ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Σχολή Χρηματοοικονομικής και Στατιστικής

Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΣΤΗΝ **ΕΦΑΡΜΟΣΜΕΝΗ ΣΤΑΤΙΣΤΙΚΗ**

ΠΡΟΒΛΕΨΗ ΚΑΙ ΑΝΑΛΥΣΗ ΤΗΣ ΑΠΟΔΟΣΗΣ ΣΕ ΑΓΩΝΕΣ ΑΥΤΟΚΙΝΗΤΟΥ FORMULA 1

Εμμανουήλ Ν. Φλωράκης

Διπλωματική Εργασία

που υποβλήθηκε στο Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στην *Εφαρμοσμένη Στατιστική*

Πειραιάς
Απρίλιος 2025

Π Α Ν Ε Π Ι Σ Τ Η Μ Ι Ο Π Ε Ι Ρ Α Ι Ω Σ

Σχολή Χρηματοοικονομικής και Στατιστικής

Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΣΤΗΝ **ΕΦΑΡΜΟΣΜΕΝΗ ΣΤΑΤΙΣΤΙΚΗ**

ΠΡΟΒΛΕΨΗ ΚΑΙ ΑΝΑΛΥΣΗ ΤΗΣ ΑΠΟΔΟΣΗΣ ΣΕ ΑΓΩΝΕΣ ΑΥΤΟΚΙΝΗΤΟΥ FORMULA 1

Εμμανουήλ Ν. Φλωράκης

Διπλωματική Εργασία

που υποβλήθηκε στο Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στην *Εφαρμοσμένη Στατιστική*

Πειραιάς
Απρίλιος 2025

Η παρούσα Διπλωματική Εργασία εγκρίθηκε ομόφωνα από την Τριμελή Εξεταστική Επιτροπή που ορίσθηκε από τη ΓΣΕΣ του Τμήματος Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς στην υπ' αριθμ. …….. συνεδρίασή του σύμφωνα με τον Εσωτερικό Κανονισμό Λειτουργίας του Προγράμματος Μεταπτυχιακών Σπουδών στην Εφαρμοσμένη Στατιστική

Τα μέλη της Επιτροπής ήταν:

- Αναπληρωτής Καθηγητής Κωνσταντίνος Πολίτης (Επιβλέπων)
- Επίκουρος Καθηγητής Μποζίκας Απόστολος
- Επίκουρος Καθηγητής Ρακιτζής Αθανάσιος

Η έγκριση της Διπλωματική Εργασίας από το Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς δεν υποδηλώνει αποδοχή των γνωμών του συγγραφέα.

UNIVERSITY OF PIRAEUS

School of Finance and Statistics

Department of Statistics and Insurance Science

POSTGRADUATE PROGRAM IN
APPLIED STATISTICS

PREDICTION AND ANALYSIS OF PERFORMANCE IN FORMULA 1 RACES

By

Emmanouil N. Florakis

MSc Dissertation

submitted to the Department of Statistics and Insurance Science of the University of Piraeus in partial fulfilment of the requirements for the degree of Master of Science in Applied Statistics

Piraeus, Greece
April 2025

Στην οικογένειά μου

Acknowledgements

The present thesis, entitled “Prediction and analysis of performance in Formula 1 races”, was conducted under the framework of the Postgraduate Program “Master in Science in Applied Statistics” of the Department of Statistics and Insurance Science at the University of Piraeus. Upon the completion of this academic journey, it is important to express my sincere gratitude to the people who stood by me throughout this effort. First and foremost, I would like to thank my supervisor, Mr. Konstantinos Politis, for giving me the opportunity to work on such a challenging and engaging topic, and for his patience, guidance, and meticulous feedback throughout the process. I would also like to thank my friends and colleagues who supported me during this journey, providing encouragement and assistance whenever needed. Lastly, I extend my heartfelt thanks to my family and my other half for their unwavering support, patience, and understanding during this demanding period.

Περίληψη

Η Φόρμουλα 1, γνωστή και ως η κορυφή του μηχανοκίνητου αθλητισμού, συνδυάζει την αιχμή της τεχνολογίας με το ταλέντο παγκόσμιας κλάσης των οδηγών και τις στρατηγικές τακτικές των ομάδων, σε ένα άθλημα όπου τα χιλιοστά του δευτερολέπτου κάνουν τη διαφορά μεταξύ νίκης και ήττας. Καθώς οι αγώνες εξελίσσονται με απρόβλεπτο τρόπο λόγω των καιρικών συνθηκών, των στρατηγικών ελαστικών, των περιστατικών στην πίστα και της μεταβαλλόμενης απόδοσης των μονοθέσιων, η Φόρμουλα 1 προσφέρει ένα εύφορο πεδίο για ανάλυση δεδομένων. Οι ομάδες αξιοποιούν προηγμένα αναλυτικά εργαλεία για να μεγιστοποιήσουν τη στρατηγική τους, να βελτιώσουν την απόδοση του μονοθέσιου και να βελτιστοποιήσουν τα pit stops. Πέρα από αυτό, η αυξανόμενη διαθεσιμότητα δεδομένων αγώνων έχει δώσει τη δυνατότητα σε ερευνητές και αναλυτές να χρησιμοποιούν προηγμένες στατιστικές μεθόδους για την αναγνώριση τάσεων, την ανάλυση της απόδοσης οδηγών και ομάδων και την ανάπτυξη προγνωστικών μοντέλων. Μέσω της χρήσης της μηχανικής μάθησης και της εφαρμοσμένης στατιστικής, είναι δυνατό να ξεπεραστούν οι παραδοσιακές προσεγγίσεις κατανόησης των αγώνων και να αποκαλυφθούν βαθύτερα πρότυπα που ορίζουν την επιτυχία στο άθλημα.

Η παρούσα διπλωματική εργασία εστιάζει στην εφαρμογή στατιστικών και μεθόδων μηχανικής μάθησης για τη μελέτη και πρόβλεψη της απόδοσης στους αγώνες της Φόρμουλα 1. Η έρευνα προηγείται από εκτενή προετοιμασία δεδομένων και δημιουργία χαρακτηριστικών, όπως διερευνητική ανάλυση μέσω περιγραφικών στατιστικών, ανάλυση συσχετίσεων και έλεγχοι κανονικότητας. Οι τεχνικές χρονοσειρών και παλινδρόμησης διευκολύνουν την παρακολούθηση και πρόβλεψη τάσεων απόδοσης μέσω ιστορικών δεδομένων. Η μείωση διαστατικότητας επιτυγχάνεται με την εφαρμογή της Ανάλυσης Κύριων Συνιστωσών (PCA) στο κεφάλαιο της μηχανικής μάθησης. Τα μοντέλα ταξινόμησης Random Forest και Support Vector Machines (SVM) χρησιμοποιούνται για την πρόβλεψη των αποτελεσμάτων των αγώνων. Τεχνικές συσταδοποίησης όπως τα K-Means και Ιεραρχική Συσταδοποίηση (Hierarchical Clustering) αναγνωρίζουν πρότυπα στην απόδοση των οδηγών. Τα αποτελέσματα αναδεικνύουν την αποτελεσματικότητα αυτών των μεθόδων στον εντοπισμό βασικών δεικτών απόδοσης και τάσεων, καταδεικνύοντας πώς οι στατιστικές και οι μεθοδολογίες μηχανικής μάθησης μπορούν να ενισχύσουν την κατανόησή μας γύρω από την ανάλυση του μηχανοκίνητου αθλητισμού, προσφέροντας δεδομενοκεντρική κατανόηση της απόδοσης και του ανταγωνισμού στους αγώνες.

Abstract

Formula 1 also known as the pinnacle of motorsport, combines cutting-edge technology with world-class driver talent and strategic team tactics in a sport where milliseconds will make all the difference between victory and defeat. As races unfold unpredictably according to weather, tire strategies, on-track mishaps and fluctuating car performance, Formula 1 presents a fertile field for data analysis. Teams utilize advanced analytics to maximize their race strategy, to improve car performance and optimize pit stops. Beyond that, the growing availability of race data has enabled researchers and analysts to utilize advanced statistical methods to find trends, to analyze driver and team performance and to develop predictive models. With the use of machine learning and applied statistics, it is possible to move beyond traditional race understandings and uncover deeper patterns that define success in the sport.

The current thesis delves into the application of statistical and machine learning methods to study and predict Formula 1 racing performance. The research is preceded by extensive data preparation and feature creation, such as exploratory analysis via descriptive statistics, correlation analysis, and normality tests. Time series forecasting and regression techniques facilitate tracking and prediction of performance trends using historic data. Dimensionality reduction is achieved through implementation of Principal Component Analysis (PCA) in the machine learning chapter. Random Forest and Support Vector Machines (SVM) classification models are employed in the prediction of race outcomes. Clustering techniques such as K-Means and Hierarchical Clustering identify patterns in driver performance. The results demonstrate the effectiveness of these methodologies for determining key performance indicators and trends, outlining how statistical and machine learning methodologies used can enhance our understanding of motorsport analytics by providing data-driven insights into race performance and competitiveness.

Table of Contents

Catalogue of Figures and Tables

Chapter 1: Introduction

The objective of the current thesis is to apply statistical and machine learning methods to analyze historical Formula 1 data. The dataset spans from the 2011 season to the summer break of the 2024 season, encompassing every driver's race results. For analytical purposes, race outcomes are categorized into three main groups: podium finishes, finishes between 4th and 10th place, and finishes outside the top ten. This classification allows for a structured approach to understanding performance trends and predicting race results through statistical methodologies.

The second chapter provides an in-depth exploration of Formula 1's history, offering insight into the evolution of rules, technological advancements, and significant milestones that have shaped the sport. A detailed review of the monocoques' development by the years, the role of the FIA, team race strategies, and the financial dynamics of the sport contextualizes the competitive landscape of Formula 1. Additionally, the chapter highlights the growing influence of data analytics and technology on performance optimization, demonstrating how modern teams leverage data-driven approaches for decision-making. The chapter concludes with an extensive literature review, examining past studies that have applied statistical analysis to Formula 1, as well as fundamental applications of statistical methodologies in motorsport research.

The third chapter focuses on descriptive statistics, laying the groundwork for the subsequent analytical methods. It begins with data presentation, processing, and the construction of key variables to establish a structured and reliable dataset for analysis. Various statistical and visualization techniques are employed to explore relationships between variables and race results. Time series plots, bar charts, and scatterplots are utilized to uncover correlations and trends within the dataset, while grouped time series analyses examine performance patterns across different race outcome categories. Further visualization techniques, such as violin and bar plots, provide deeper insight into the distribution and variability of key performance indicators. The chapter concludes with an analysis of fastest lap time progression across circuits over the years, highlighting which types of circuits influence the impact of technological advancements in the pursuit of lower fastest laps.

The fourth chapter explores the distributional properties of the dataset and the relationships between key numerical variables. It begins with normality tests, where the Kolmogorov-Smirnov (KS) test is applied alongside graphical methods such as Q-Q plots and histograms to assess the extent to which variables conform to a normal distribution. A specific focus is placed on qualifying times (Q1, Q2, and Q3), excluding zero values to ensure a more meaningful evaluation. The second part of the chapter shifts to correlation analysis, utilizing Spearman's rank correlation coefficient to identify relationships between numerical variables. Heatmaps visually represent these correlations both across the entire dataset and when grouped by race outcome, providing insight into how different performance metrics interact under varying race results.

The fifth chapter introduces Generalized Linear Models (GLMs) and their application in predicting race outcomes. The first part provides theoretical foundations, covering regression analysis, logistic regression, and multinomial logistic regression, along with key evaluation techniques. Following this, a multinomial logistic regression model is implemented, where variables are carefully selected by reducing multicollinearity and retaining only statistically significant predictors. The model's performance is assessed through accuracy scores, ROC curves, classification reports, confusion matrices, and learning curves. Additionally, Lasso cross-validation (Lasso CV) is applied as an alternative feature selection method to compare its selected variables with those retained in the multinomial logistic regression.

The sixth chapter focuses on time series analysis, particularly in the context of predicting the progression of fastest lap times in Formula 1. The theoretical foundation covers key concepts such as stationarity, trends, seasonality, and unit root tests, followed by model selection and validation techniques. The applied analysis begins by visualizing time series data and assessing stationarity through ACF and PACF plots, as well as statistical tests like the Augmented Dickey-Fuller (ADF) test. When necessary, differentiation is applied to achieve stationarity before selecting the most suitable models based on AIC, BIC, and AICc criteria. The chosen models are rigorously evaluated by analyzing residuals using tests such as Anderson-Darling, Shapiro-Wilk, Box-Ljung, normal Q-Q and ACF plots. Once validated, the models are used to forecast fastest lap times for the next two seasons. The analysis concludes with a comparative discussion on time series patterns across different circuit types and an evaluation of forecast accuracy by comparing predicted times with the most recent lap records available at the time of writing.

The seventh chapter delves into the application of machine learning techniques for analyzing and predicting Formula 1 performance. The chapter begins with an introduction to machine learning, covering key concepts in supervised, unsupervised, semi-supervised, and reinforcement learning. Feature selection methods and dimensionality reduction techniques, specifically Principal Component Analysis (PCA), are then discussed. For the implementation, PCA is applied to identify the optimal number of principal components, balancing computational efficiency with predictive performance. The classification models implemented include Random Forest and Support Vector Machines (SVM), both of which are evaluated using accuracy, cross-validation scores, classification reports, and confusion matrices. For clustering, K-Means and Hierarchical Agglomerative Clustering are utilized, with the optimal number of clusters determined using the elbow plot and dendrogram, respectively. The quality of clustering is assessed using silhouette scores, the Adjusted Rand Index, and the Calinski-Harabasz Index, alongside visualizations to inspect the separation of clusters. The chapter concludes with a comparative discussion of the applied machine learning methods, highlighting their effectiveness and suitability for Formula 1 data analysis.

Chapter 8 presents the final conclusions of the thesis, summarizing the main insights gained from the analysis and evaluating how effectively the study’s objectives were met through the applied methodologies.

Chapter 2: Analyzing Formula 1

2.1 Formula 1 History and Advancements

Formula 1 racing can trace its origins back to the European Grand Prix competitions of the 1920s and 1930s. However, the foundation of what we recognize as modern Formula 1 began in 1946 when the Fédération Internationale de l'Automobile (FIA) established a standardized set of rules. This move paved the way for the first official World Championship of Drivers, which took place in 1950.

Throughout its history, the sport has evolved in tandem with its technical regulations.

The Pre-Formula 1 Era

In the early 1900s, motorsports were about raw skill, innovation, and endurance. Key events like the Targa Florio, launched in 1906 in Sicily, tested drivers with a challenging 92-mile circuit full of treacherous terrain and sharp turns. It demanded not just speed but mechanical resilience and driver expertise. Another iconic race, the Indianapolis 500, began in 1911, becoming a prestigious competition in the U.S. and attracting top international talent. These early races laid the groundwork for what would evolve into the modern Formula 1 series

Early Years and Pre-World War II Influence (1946–1950)

In 1946, the FIA's Commission Sportive Internationale defined Formula 1 as the top tier of single-seater racing, officially starting in 1947. The new classification evolved from pre-war guidelines, balancing supercharged 1.5-litre engines with non-supercharged 4.5-litre engines.

The first official Formula 1 race is debated, with candidates including the 1946 Turin Grand Prix and the 1947 Pau Grand Prix. Championships for drivers and constructors were initially absent, with early races dominated by Italian cars, especially from Maserati. This period saw the end of pre-war legends and the rise of new talents like Alberto Ascari and Juan Manuel Fangio.

The Era of Italian and Mercedes Dominance (1950–1957)

The first official World Championship for Drivers was established by the FIA in 1950, formalizing the structure of Grand Prix racing. Italian manufacturers—Alfa Romeo, Ferrari, and Maserati—dominated the early years, with Alfa Romeo winning nearly every race in 1950 using their pre-war "Alfetta" cars. Notable drivers included Nino Farina, who won the first championship, and Juan Manuel Fangio, who claimed the title in 1951.

By 1952, Alfa Romeo had withdrawn, and Ferrari became the dominant force, securing championships for Alberto Ascari in 1952 and 1953. The championship adhered to Formula 2 rules during these years due to a lack of competition under Formula 1 regulations.

In 1954, Formula 1 reverted to a new 2.5-litre engine specification, attracting major manufacturers like Lancia and Mercedes-Benz. Fangio, driving for Mercedes, dominated with advanced cars featuring cutting-edge technology. However, Mercedes abruptly withdrew from the sport at the end of 1955 following the Le Mans tragedy. Fangio then won his fifth title with Maserati in 1957, a record that remained unbroken for nearly five decades.

British Teams and the Rear-Mid Engine Revolution (1958–1961)

In 1958, Formula 1 implemented shorter races and switched to avgas fuel, introducing the International Cup for F1 Manufacturers. Mike Hawthorn became the first British driver to win the Drivers' Championship, while Vanwall claimed the Constructors' title. Stirling Moss's victory in a mid-engined Cooper at the Argentine Grand Prix marked a significant shift in car design.

The mid-engine revolution gained momentum in 1959, with fierce competition between Jack Brabham's and Moss's Coopers. By 1960, British teams outperformed front-engined designs, leading Lotus and BRM to adopt mid-engine configurations. In 1961, the FIA limited engines to 1.5-litres, and Ferrari's new V6 156 dominated the season, with Phil Hill winning the championship.

This era also featured many important races that, while operating under Formula 1 regulations, were not part of the official World Championship.

Anglophone Drivers and 1.5-Litre Engines (1962–1967)

The 1962 season marked the debut of the Lotus 25, featuring the new Coventry-Climax V8 engine and an innovative monocoque chassis. Jim Clark, despite early reliability issues, finished second, while Graham Hill claimed the championship with his BRM.

As reliability improved, Clark dominated, winning titles in 1963 and 1965, the latter being notable for his victory in both the Formula 1 Championship and the Indianapolis 500. In 1964, John Surtees won the championship driving for Ferrari, becoming the only driver to win titles in both cars and motorcycles.

The 1966 season ushered in a new era with the introduction of 3.0-litre engines. Jack Brabham’s team triumphed with a lightweight Repco V8, while Ferrari struggled with heavier cars. In 1967, Lotus introduced the groundbreaking Lotus 49, powered by the Ford-Cosworth DFV engine, which would dominate the sport for a decade. Brabham's reliability and Hulme's consistency secured back-to-back championships for the team.

By the late 1960s, about one-third of races were held outside Europe, and British drivers, including Hill, Clark, Surtees, and Stewart, dominated the scene, winning seven championships between them.

DFV Engine, 12-Cylinder Engines, and the Arrival of Sponsorship, Safety, and Aerodynamics (1968–1976)

In 1968, Lotus lost its exclusive rights to the Ford-Cosworth DFV engine, ushering in a new era as McLaren and Ken Tyrrell's team entered the scene with Cosworth-powered chassis. This season was marked by the tragic death of two-time champion

Jim Clark at Hockenheim, prompting a significant push for improved safety measures in the sport. Innovations included unrestricted sponsorship following the withdrawal of traditional automotive sponsors, with Team Gunston and Lotus leading the way in displaying sponsor liveries. Additionally, the introduction of wings to enhance downforce began, although they resulted in accidents due to structural failures, highlighting the need for better safety protocols. The first full-face helmets made their debut, further advancing driver safety.

Safety concerns intensified after a driver boycott led to the cancellation of the 1969 Belgian Grand Prix due to inadequate track conditions. Jackie Stewart secured the 1969 title with the Matra MS80, the only chassis built in France to win a championship. Following wing-related incidents at the Spanish Grand Prix, the FIA initially banned wings, but they were reintroduced later in the season with restrictions on size and mounting. The 1970 season was marked by the tragic death of Jochen Rindt, who posthumously won the championship while driving the innovative Lotus 72, featuring advanced technologies like inboard brakes and variable flexibility suspension. Ferrari also began improving with its flat-12 engine, gaining competitiveness by the season's end.

In 1971, Stewart continued to excel, leading Tyrrell to success with the Matra MS80. The following year, Emerson Fittipaldi became the youngest world champion at 25, driving for Lotus. Stewart’s retirement at the end of the 1973 season opened the field for McLaren, which enjoyed success with the M23, driven by Fittipaldi. The 1974 season saw intense competition among McLaren, Ferrari, and Brabham, with notable advancements such as Ferrari's monocoque chassis and transverse gearbox improving weight distribution and performance.

Niki Lauda's dominance began in 1975, culminating in his first championship title. The newly formed Wolf team made waves, surprising everyone by finishing strongly in the Constructors’ Championship. The 1976 season became famous for the rivalry between Lauda and James Hunt. Lauda suffered severe injuries in a crash at Nürburgring but made an incredible recovery to return to racing just six weeks later. Hunt ultimately clinched the championship by a single point after Lauda withdrew from the rain-soaked final race in Japan, citing safety concerns. This period also saw the introduction of radical designs like the six-wheeled Tyrrell P34, which proved competitive but could not outperform the best four-wheeled cars. By 1977, Lauda secured his second title, despite tensions within the Ferrari team, solidifying his legacy as a dominant figure in the sport.

Ground-Effect Era (1977–1982)

The ground-effect era began in 1977 with Lotus's introduction of the Lotus 78, which utilized wing-profiled sidepods and sliding lexan skirts to generate significant downforce with reduced drag. This innovation helped Mario Andretti and Gunnar Nilsson secure five wins that season. Renault also made a splash with the RS01, the first turbocharged Formula 1 car, and introduced Michelin’s radial tyres, challenging Goodyear’s monopoly.

In 1978, the Lotus 79 further refined ground effect technology, allowing Andretti to win the championship. Brabham's BT46B "fan car," which exploited a regulatory loophole, won its only race before being withdrawn. Tragically, the season was overshadowed by the death of Ronnie Peterson at Monza.

By 1979, other teams, including Ligier and Ferrari, developed competitive wing cars, prompting Lotus to rush the less successful Lotus 80. The RS10 turbo car finally won at Dijon, showcasing Renault's commitment to the new technology. The era also witnessed intense political struggles between FISA (the governing body) and FOCA (the Constructors' Association) over turbo versus ground effect technologies, as teams navigated the complexities of evolving regulations.

In the early 1980s, champions included Jody Scheckter for Ferrari, Alan Jones and Keke Rosberg for Williams, and Nelson Piquet for Brabham. The tragic deaths of Patrick Depailler and Gilles Villeneuve heightened safety concerns, pushing the sport toward reform.

Amidst these changes, McLaren’s merger with Ron Dennis’s Project-4 team in 1981 led to the innovative MP4/1, the first car with a carbon fiber composite chassis, which improved safety and performance. This material quickly became the standard in Formula 1.

1.5-Litre Turbocharged Engines (1983–1988)

The 1983 season marked the first drivers' title won by a turbocharged engine, with Nelson Piquet claiming victory for the BMW-powered Brabham team. The earlier disputes between FISA and FOCA had been resolved, and by this time, it was clear that turbocharged engines provided a competitive edge, especially on high-speed and high-altitude tracks. By 1982, teams recognized the necessity of adopting turbo technology to remain competitive.

As reliability improved, by 1984, only Tyrrell continued with the outdated DFV engines. This period saw turbocharged engines achieving power outputs comparable to the legendary 640 hp of the 1937 Mercedes-Benz W125, with some qualifying engines producing over 1,350 bhp by 1986. However, restrictions on fuel consumption and turbocharger boost were introduced, limiting their performance.

The 1986 season was notable as it was the only year before 2014 in which every car used a turbocharged engine, with Williams-Honda winning the Constructors' Championship. Niki Lauda's 1984 title win over Alain Prost was one of the closest finishes in Formula 1 history, marred by controversy over half points at the Monaco Grand Prix.

The rivalry between Prost and Ayrton Senna began in earnest during this period, particularly as Prost drove for McLaren with the TAG turbo engine, winning multiple championships. In 1986, Williams cars dominated, but internal competition between Piquet and Nigel Mansell allowed Prost to stay in contention, with the Drivers' title being decided only in the final race of the season.

By 1987, the introduction of the 3.5-litre atmospheric engines marked a shift, although turbo engines continued to dominate until their eventual ban in 1989.

Williams-Honda's success continued, and McLaren’s partnership with Honda in 1988 led to a record-setting season, with Senna claiming his first World Title after winning 15 out of 16 races.

3.5-Litre Naturally Aspirated Engines, Active Suspension, and Electronic Driver Aids

In 1989, the ban on turbocharged engines ushered in a new era with regulations permitting only naturally aspirated engines up to 3.5 litres. McLaren-Honda dominated the championship for the next three years, with Alain Prost winning in 1989 and Ayrton Senna taking titles in 1990 and 1991. The V10 and V12 engines developed by Honda matched the performance of their turbo predecessors, with McLaren drivers winning 37 of 48 races during this period. However, the intense rivalry between Prost and Senna marred the seasons, highlighted by their controversial clashes at the Japanese Grands Prix in 1989 and 1990.

As the 1990s progressed, teams began integrating electronic driver aids, including active suspension, semi-automatic gearboxes, and traction control, which enhanced performance but led to concerns about the diminishing role of driver skill. Despite the FIA's eventual ban on these technologies in 1994, many believed that the enforcement was ineffective.

In 1992, Nigel Mansell finally clinched the title after years of competition, while Alain Prost secured his fourth championship in 1993, both driving for the dominant Williams-Renault team. The introduction of lightweight television cameras in the early '90s increased viewership and sponsorship opportunities, moving beyond traditional sources.

As the 1994 season approached, expectations were high, particularly with Senna moving to Williams, Schumacher driving for Benetton, and McLaren hoping for success with its new Peugeot engine. Unfortunately, the season became notorious for tragic events that overshadowed the racing.

Safety, Rules, and Regulations (1994)

By 1994, Formula 1 had not experienced a fatality in nearly a decade, fostering a false sense of safety among teams and fans. This perception was shattered during the San Marino Grand Prix weekend, which witnessed serious accidents involving Rubens Barrichello in practice, followed by the tragic deaths of Roland Ratzenberger and three-time world champion Ayrton Senna. The aftermath prompted the FIA to take immediate and decisive action to enhance safety in the sport.

Significant changes to car design were not feasible for the 1994 season, but several regulations were quickly implemented. Airboxes were required to be perforated to limit power, while special racing fuels were banned in favor of blends more similar to standard unleaded petrol. Additionally, a wooden "plank" was mandated beneath the chassis to reduce downforce; cars would be deemed illegal if the plank wore beyond a specified tolerance.

Starting in 1995, designs had to conform to a reference plane, introducing strict tolerances for cockpit sizes and aerodynamic features. The maximum engine displacement was reduced from 3.5 to 3.0 litres. Further regulations included increasing cockpit opening sizes for easier driver exit, introducing grooved tyres to decrease grip, and imposing limitations on wing sizes to curb aerodynamic downforce.

The rapid implementation of these regulations created a chaotic atmosphere in Formula 1. Michael Schumacher, driving for the Benetton team, faced numerous challenges, including suspensions due to regulatory violations. His championship victory in Australia was marked by controversy, as he collided with rival Damon Hill to secure his first World Drivers' Championship.

3-Litre Engines (1995–1999)

The 3-litre engine formula introduced in 1995 did not hinder the dominance of the Renault V10, as Michael Schumacher secured his second Drivers' Championship and helped Benetton achieve its first Constructors' title. The only significant challenge came from Williams, with Damon Hill and David Coulthard. Ferrari, meanwhile, managed just one victory at the Canadian Grand Prix, where Jean Alesi claimed his only career win.

In 1996, the FIA mandated larger cockpit areas and head protection, which ironically limited visibility and contributed to accidents. Michael Schumacher joined Ferrari, quickly revitalizing the team's performance by winning three races in his first season, though he ultimately finished behind Damon Hill, who claimed the championship after years of near misses. The following year, Jacques Villeneuve, son of a racing legend, won the Drivers' Championship for Williams, making history as the first Canadian champion. His title battle culminated in a dramatic collision with Schumacher, who faced disgrace after being stripped of points for his actions.

After the 1997 season, Renault exited Formula 1, paving the way for McLaren-Mercedes and Mika Häkkinen to dominate in 1998 and 1999. Häkkinen secured his first title, while Schumacher endured setbacks, including a leg-breaking crash at Silverstone in 1999. His teammate Eddie Irvine narrowly missed the championship, but their efforts contributed to Ferrari's first Constructors' title since 1983.

Amidst these title battles, the landscape of Formula 1 changed significantly. Established teams like Lotus and Brabham disappeared, and Ligier was sold to Alain Prost. This marked the end of the small private teams, with only Jordan, Sauber, Arrows, and Minardi remaining. Despite a brief resurgence from drivers like Damon Hill and Ralf Schumacher, the decline of privateers indicated looming challenges for the sport, as even the once-dominant Benetton struggled to remain competitive, culminating in Jackie Stewart's team transforming into Jaguar.

V10 Engines and Road Car Manufacturer Participation (2000–2005)

Following the ban on turbocharged engines, V10 engines became the standard in Formula 1, balancing power and fuel efficiency. By 2000, Schumacher and Häkkinen were the top contenders, with Schumacher winning his third championship, marking

Ferrari's first title since 1979. The 2001 season saw Schumacher dominate, winning the championship by the Hungarian Grand Prix and reintroducing electronic driver aids like traction control, which bolstered Ferrari's performance.

The early 2000s also saw an influx of car manufacturers in Formula 1. Teams like BMW and Honda returned as engine manufacturers, while Renault purchased Benetton and rebranded it. This shift allowed for greater competition, although Ferrari's dominance continued, leading to concerns over the sport's competitiveness.

Despite changes aimed at leveling the playing field, Schumacher captured another title in 2003, although the season featured eight different race winners. In 2004, Ferrari once again demonstrated its dominance, winning 12 of the first 13 races. That season also saw the Formula 1 calendar expand to include new races in Bahrain and China, further increasing the sport's global reach.

In 2005, Renault, led by the 24-year-old Fernando Alonso, rose to prominence, with Alonso becoming the youngest world champion, and securing the Constructors' title for Renault. The season also saw the decline of smaller teams, as Minardi was acquired by Red Bull, marking the end of an era for independent teams. The 2005 season concluded the V10 era, as regulations aimed to control costs and maintain competitive balance, with Renault and Ferrari achieving significant success during this time.

2.4-Litre V8 Engines (2006–2008)

The 2006 season was the last with both Bridgestone and Michelin as tyre suppliers. Renault, led by Fernando Alonso, dominated early on, securing both titles despite a mid-season challenge from Ferrari. Alonso won his second championship after Michael Schumacher's engine failure in Japan.

In 2007, McLaren introduced a competitive pairing of Alonso and rookie Lewis Hamilton. Hamilton's strong start led to tension between the teammates, and ultimately, Kimi Räikkönen of Ferrari claimed the title by just one point after a late-season surge.

The 2008 season featured fierce competition, with Hamilton clinching the Drivers' Championship in a dramatic final lap at the Brazilian Grand Prix, overtaking Timo Glock for the necessary points. Ferrari won the Constructors' Championship, marking their eighth title in ten years. This era concluded with significant regulatory changes, including the ban on traction control starting in 2009.

Cost-Cutting Measures and Departure of Car Manufacturers (2009–2013)

The 2009 Formula 1 season introduced crucial regulations to curb rising costs amid a global economic crisis. Key changes included engine RPM limits and the Kinetic Energy Recovery System (KERS), aimed at promoting overtaking and enhancing car performance. Brawn GP emerged as a surprise champion in its debut season, while established teams like Ferrari and McLaren struggled with the new dynamics.

The economic downturn led to the exit of major manufacturers. Following Honda's departure in 2008, both Toyota and BMW left the sport in 2009, and Renault

transitioned to an engine supplier role. Teams implemented stringent cost-cutting measures, including reduced testing and increased engine mileage requirements.

In 2010, the points system was revamped, and Red Bull Racing rose to prominence, with Sebastian Vettel winning his first championship in a dramatic finale. The 2011 season saw Vettel secure his second title, while 2012 brought increased competition from Alonso and Ferrari. By 2013, Vettel dominated the season, clinching his fourth consecutive title with an impressive nine race wins. Overall, this period marked a significant transformation in Formula 1, characterized by regulatory changes and the decline of prominent manufacturers.

Introduction of 1.6-litre Turbocharged V6 Hybrid Power Units and Cost Cap (2014–2021)

In 2014, Formula 1 entered its second turbocharged era, introducing 1.6-litre turbocharged V6 hybrid engines. This marked a significant shift in regulations, as cars were required to use engines limited to 15,000 rpm with a maximum fuel flow of 100 kg/hr. Alongside engine changes, the minimum weight of cars increased from 642 kg to 690 kg. The new energy recovery system (ERS) was also introduced, offering drivers an additional 160 horsepower, significantly enhancing performance compared to the previous KERS system.

Mercedes quickly dominated this hybrid era, winning 51 of 59 races from 2014 to 2016, with Lewis Hamilton securing titles in 2014 and 2015, and Nico Rosberg winning in 2016. Their success was attributed to innovative engineering, including a "split turbocharger" design that improved efficiency and power delivery. Despite strong competition from Ferrari and Sebastian Vettel in 2017 and 2018, Mercedes continued to excel, clinching titles in 2019 and 2020, with Hamilton equaling Michael Schumacher's record of seven championships.

The competition intensified in 2021, with Red Bull Racing, powered by Honda, challenging Mercedes. Max Verstappen emerged as the Drivers' Champion after a fiercely contested season. However, the championship's conclusion was marred by controversy during the final race in Abu Dhabi, where a late safety car decision by the FIA allowed Verstappen to overtake Hamilton on the last lap. Many fans and analysts criticized the decision, claiming it effectively robbed Hamilton of an eighth title. This incident sparked significant debate over the FIA's race regulations and management, highlighting the intense rivalry and drama that characterized the 2021 season.

Ground Effect Cars and Aerodynamic Changes (2022–2025)

In 2022, the FIA introduced regulations to promote "closer racing" through aerodynamic changes, reintroducing ground effect, which enhances downforce from the car’s floor. This reduces drag and "dirty air," facilitating overtaking. New designs simplified aerodynamics by eliminating bargeboards and modifying front wings, while engine regulations shifted to include 10% bio-components for sustainability.

Red Bull Racing, led by Max Verstappen and Sergio Pérez, dominated the 2022 season, with Verstappen winning the Drivers' Championship with a record 454 points. In 2023, Verstappen clinched his third consecutive title, winning 19 of 22 races,

setting the highest win percentage for a driver at 86.36%. Red Bull secured their sixth Constructors' Championship, achieving a 95.45% win rate.

In Conclusion

Formula 1 has undergone significant transformations since its inception in 1950, characterized by technological innovations, evolving regulations, and fierce competition among iconic teams. From the early dominance of manufacturers like Alfa Romeo to the formidable rivalries of McLaren and Ferrari, the sport has continuously pushed the limits of automotive engineering. The introduction of turbocharged engines and hybrid power units has reshaped racing dynamics, emphasizing both performance and efficiency.

A pivotal figure in this evolution is Michael Schumacher, whose remarkable career and unprecedented seven World Championships record redefined the standards of excellence in Formula 1. Schumacher's ability to dominate races, coupled with his influence on team dynamics and strategy, not only secured his place in the history books but also set a benchmark that future generations of drivers aspire to reach. His legacy, alongside the sport's ongoing advancements, ensures that Formula 1 remains a thrilling and innovative spectacle for fans around the world.

Other Technological Advancements

Beyond the major innovations previously discussed, several additional breakthroughs played key roles in F1's evolution. In 1952, hard-shell helmets became mandatory, shifting from cloth-covered cork to more protective materials like steel. The introduction of four-wheel-drive technology made waves in 1969, with early experimentation leading to a victory at the Oulton Park International Gold Cup in 1961. The 1970s saw a shift to side-mounted radiators, improving aerodynamics compared to traditional nose-mounted designs. In 1984, carbon-carbon brakes replaced steel brakes, significantly enhancing braking performance, despite a brief ban in 1998. A notable 1997 development was the steer-brake pedal, which allowed more precise corner braking, challenging drivers to adapt to four pedals. In 2009, the Kinetic Energy Recovery System (KERS) was introduced by some teams to capture braking energy, though it was later replaced. The F-duct appeared in 2010, enabling airflow adjustments for speed boosts before being replaced by DRS in 2011. More recently, Mercedes introduced the Dual Axis Steering (DAS) system in 2020, allowing real-time wheel alignment adjustments, though it was quickly banned in 2021.

F1 Logo

The F1 logo has evolved significantly since the inception of the sport in 1950, reflecting the dynamic nature of Formula 1 racing.

In the early decades, the logo was simple and straightforward, primarily featuring "FIA Formula 1 World Championship" with a classic, straightforward typeface. These early logos were emblematic of the traditional and formal style of motorsport at the time.

In 1994, F1 introduced a modernized logo that became iconic. It consisted of a stylized "F" with a hidden "1" between the black "F" and the red speed lines, representing speed and competition. This logo stayed in use for over two decades and became synonymous with the brand, emphasizing speed and a futuristic look.

In 2017, F1 underwent a rebranding, introducing a new, simplified logo. The updated logo uses a sleeker, more angular design with two curved lines forming an "F" and "1," symbolizing a race track and evoking a sense of motion. This redesign aimed to reflect F1's digital age, catering to modern platforms and a new generation of fans while maintaining a sense of movement and speed.

Figure 2.1: F1 logo transformation by the years

(Source: https://logos-world.net/f1-logo/)

The evolution of the F1 logo reflects the sport's transformation from a traditional motorsport competition to a cutting-edge, technology-driven global spectacle.

2.2 Formula 1 Open-Wheel Cars

In the following analysis, various visualizations have been created using data from https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020 combined with developing a relevant code in Python Programming Language. The Python code used to generate the diagrams is presented in detail in the attached appendix at the end of the Present Thesis (Appendix 1: Implementation Code of the Present Study).

Overview

Formula 1 cars represent the pinnacle of open-wheel racing, defined by their exposed wheels and a single-seat cockpit, designed for optimal speed and aerodynamic performance. The open-wheel concept traces its origins back to 1911, when Ray Harroun's innovative vehicle design won the first Indianapolis 500. This

victory laid the groundwork for future race car development, emphasizing lightweight builds and advanced aerodynamics.

In the following Time Series Plot, created by computing the average lap times at Australia's Albert Park Grand Prix Circuit, the continuous annual advancements in Formula 1 car technology contributing to the optimization of average in-race lap times can easily be observed.

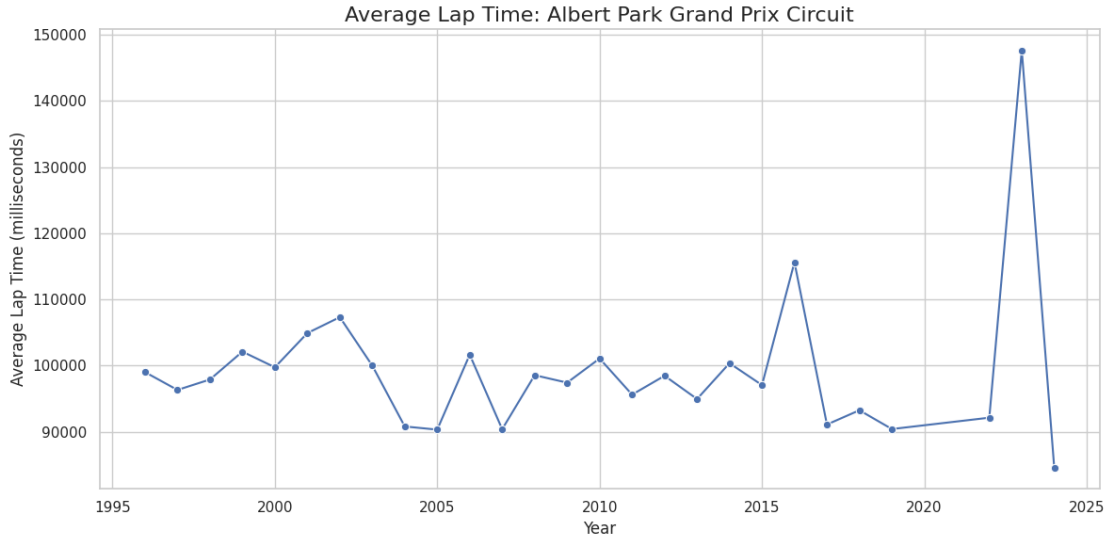

Figure 2.2: Time Series Average Lap Time in Albert Park GP per Year

As observed in the years from 1996 to the early 2000s, the implementation of new safety regulations aimed at reducing straight-line speeds, resulted in a slight upward trend in average lap times. However, starting from 2006, a consistent decline in lap times becomes evident, marking the advent of a new era in monocoque design. Notable exceptions to this trend include spikes in the 2016 and 2023 Grands Prix, where significant incidents led to 6 and 8 cars retiring the races, respectively. This overall downward trajectory in average lap times culminates in the 2024 Grand Prix, which recorded the lowest value to date.

Design Features

Modern F1 cars feature a mid-engine layout with hybrid, turbocharged power units, housed in a semi-open cockpit. Constructed primarily from lightweight yet robust carbon-fibre composites, they are both highly durable and incredibly nimble. The minimum weight, including the driver (but not fuel), is regulated to be 795 kg (1,753 lbs), with teams using strategically placed ballast to achieve the minimum weight while optimizing car balance and handling.

Aerodynamic efficiency is paramount in F1 design. Downforce is generated through sophisticated front and rear wings and ground-effect designs that maximize grip in corners. These cars can reach speeds over 360 km/h (220 mph), aided by a sleek aerodynamic profile and low-drag construction. Since 2009, the return to slick tyres has further improved grip, and suspension designs typically utilize double wishbone or multilink configurations to manage the intense forces experienced during a race.

Driving Dynamics

Precision driving is essential in F1 due to the sensitive nature of open-wheel cars, where even minor contact can result in damage. The visible tyres aid in maneuverability, allowing drivers to judge their position on the track accurately. The lightweight design contributes to faster acceleration and braking, while the exposed wheels aid in cooling the carbon disc brakes—crucial for maintaining performance on varied track conditions.

The current generation of F1 cars is powered by 1.6-liter V6 turbo hybrid engines, integrated with advanced energy recovery systems. These power units generate upwards of 780 bhp (580 kW), while strict regulations ban certain technologies like active suspension to maintain competitive integrity. Despite restrictions, modern F1 cars still generate significant downforce, enabling lateral forces of up to 3.5g in corners. This, combined with the vehicle's agility, demands exceptional physical and mental endurance from drivers, who experience high lateral forces during races.

Teams are limited to two cars each and must adhere to stringent engine regulations, with only three power units allowed per season without incurring penalties. These constraints emphasize reliability, strategy, and skill in managing the car's power and performance throughout the long F1 season.

Energy Recovery Systems (ERS) and Drag Reduction System (DRS)

ERS (Energy Recovery Systems) is a technology used in Formula 1 to improve efficiency and boost performance. Introduced in 2014, it builds on the earlier Kinetic Energy Recovery System (KERS) that debuted in 2009. ERS captures energy during braking and stores it in a battery or flywheel, allowing drivers to deploy this stored energy during acceleration, improving speed and reducing fuel consumption. ERS is divided into two systems: **MGU-K** (which recovers kinetic energy from braking) and **MGU-H** (which recovers heat energy from the exhaust).

The **Drag Reduction System (DRS)** is a technical feature that helps facilitate overtaking by reducing aerodynamic drag. The system works by allowing the driver to open a flap on the rear wing, which decreases air resistance and provides a temporary speed boost. To activate DRS, a driver must be within **one second** of the car ahead at specific detection points on the track. DRS becomes available from the **third lap** of the race, giving competitors time to position themselves strategically after the start. It can only be used in designated DRS zones, typically located on long straights to maximize overtaking opportunities. Once the overtake is completed or the one-second gap requirement is not met, the DRS deactivates, and the rear wing returns to its normal position to maintain stability.

2.3 The History and Role of the FIA in Formula 1

The Fédération Internationale de l'Automobile (FIA) has been instrumental in shaping Formula 1 since its inception. After WWII, the FIA standardized regulations for Formula 1 in 1946, leading to the inaugural World Championship for Drivers in 1950 and the World Constructors' Championship in 1958. These initiatives established a framework for competitive racing and team recognition.

Today, the FIA governs Formula 1, ensuring the sport adheres to a stringent rulebook that maintains safety, fairness and event organization in F1 , continuously adapting regulations to enhance the sport's integrity and sustainability, underscoring its vital role in Formula 1's evolution. It also certifies global records, organizes events, and promotes sustainable practices in motorsports, showcasing its enduring influence since its founding over a century ago.

Formula 1 Regulations showcase the sport’s constant balancing act between safety, competition, and technological innovation.

1950s - Early Regulations

When Formula 1 began in 1950, the regulations were relatively simple, focusing mainly on engine size. The original rules allowed either 1.5-liter supercharged engines or 4.5-liter naturally aspirated engines. There were few restrictions on the car's design, leading to a variety of innovative engineering approaches.

1960s - Introduction of Safety Measures

In the 1960s, regulations began to emphasize driver safety. Helmets became mandatory, and by the end of the decade, cars were required to have seatbelts. Technological advances in aerodynamics, including the introduction of wings, changed the racing landscape, prompting the FIA to standardize car dimensions to maintain fair competition.

1970s - Safety and Technical Advances

The 1970s saw the introduction of strict safety standards after a series of fatal accidents. Stronger crash structures, fire-resistant suits, and fuel cell safety regulations were mandated. Turbocharged engines began to dominate, leading to the first limitations on engine performance to control escalating speeds.

1980s - Turbo Era and Electronic Innovations

Turbocharging peaked in the 1980s, leading the FIA to impose limits on fuel capacity and introduce a ban on turbo engines by the end of the decade. This era also saw the first electronic driver aids, like traction control, causing debates over technology’s impact on driver skill.

1990s - Refocusing on Aerodynamics and Safety

The 1990s brought significant changes to aerodynamics. Regulations introduced narrower cars and higher wings to reduce downforce, enhancing safety by lowering

speeds. Engine sizes were standardized to 3.0-liter naturally aspirated units. The introduction of crash tests for chassis further emphasized safety improvements.

2000s - Modern Era and Technological Control

In the 2000s, the FIA imposed restrictions on technological aids to maintain driver skill as the deciding factor in races. Systems like traction control and active suspension were banned. Aerodynamic regulations were modified to limit downforce, aiming to improve overtaking and competitiveness.

2010s - Hybrid Power and Sustainability

The 2010s marked the beginning of the hybrid era, with the introduction of the 1.6-liter V6 turbocharged hybrid engines in 2014. This move aimed to make the sport more environmentally friendly. Other rule changes focused on reducing costs, with standard components and budget constraints introduced to level the playing field.

2020s - Focus on Cost, Safety, and Sustainability

Recent regulations emphasize sustainability and financial fairness. Cost caps were introduced in 2021 to control spending among teams. Safety continues to be a priority, with stricter crash tests and the introduction of the "halo" cockpit protection device in 2018. New rules for 2026 aim to increase efficiency, with lighter cars, hybrid power units, and fully sustainable fuels to ensure the sport stays relevant and eco-friendly.

Throughout its history, Formula 1 regulations have constantly evolved to keep pace with technological progress, improve safety (especially after Romain Grosjean’s horrific accident in 2020 Bahrain Grand Prix), and maintain competitive racing, shaping the sport into its modern form while keeping its core identity intact.

The 2026 Formula 1 Technical Regulations: A New Era

The upcoming 2026 Formula 1 regulations are set to revolutionize the sport, aiming for closer racing, greater sustainability, and improved safety. A key shift involves a lighter, more agile car design, reducing vehicle weight by 30 kg and decreasing wheelbase to 3400mm, and car width to 1900mm., which enhances handling and efficiency. Formula 1 will also see a new power unit that balances 50% of its output between internal combustion and electric energy, with electric power increasing to 350 kW—nearly 300% more than current engines using 100% sustainable fuels.

Aerodynamic changes include Active Aerodynamics, which will adjust wing configurations to improve cornering and straight-line speed. This, alongside the Manual Override Mode, which grants trailing cars extra electrical power, promises more overtaking opportunities and exciting racing dynamics, removing the DRS.

Sustainability is a major focus, with all cars switching to 100% sustainable fuel by 2026, and the new hybrid engines designed to make Formula 1 more eco-friendly. This initiative aligns with the FIA’s ambition to achieve Net Zero carbon emissions by 2030, with innovations intended to influence the broader automotive industry.

A record number of manufacturers, including Ferrari, Mercedes, Alpine, Honda, Audi, and Red Bull Ford, have committed to these regulations, drawn by the advanced

technology and green initiatives. Additionally, enhanced safety measures will strengthen car structures and improve testing standards, further cementing Formula 1's place at the forefront of motorsport innovation.

These changes, developed with input from F1 teams and stakeholders, aim to make the sport faster, safer, and more engaging for fans while maintaining its competitive and technological edge setting a goal of Net Zero carbon by 2030.

Formula 1's long history of drivers battling the FIA

Formula 1 has a long history of conflicts between drivers and the FIA, often centered around safety, regulations, and the governing body's authority. In the early years, disputes mostly involved teams, like Ferrari's 1950 British Grand Prix boycott over prize money issues.

In the 1970s, drivers became vocal advocates for change, particularly regarding unsafe track conditions at venues like Spa, Nürburgring, and Montjuic. Led by Jackie Stewart, they pushed for better barriers, track improvements, and enhanced emergency response. Drivers even threatened to withdraw from races, leading to direct clashes with the FIA and race organizers.

The late 1970s also saw tensions over regulations, such as ground effect aerodynamics and budget caps. In the 1980s and 1990s, figures like Niki Lauda and Ayrton Senna challenged the FIA over car specifications, political controversies, and race conditions, reinforcing the idea that drivers weren't afraid to confront authority.

In the modern era, tensions remain. Max Verstappen's recent criticism of FIA penalties for his press conference language highlights that drivers continue to challenge the FIA when they believe rules are unfair or inconsistent. These conflicts reflect a long-standing balance between safety, competition, and the FIA's efforts to control the sport's development.

The Formula 1 Teams Association (FOTA)

The Formula 1 Teams Association (FOTA) was formed in 2008 to represent F1 teams in negotiations with the FIA and Formula 1 Group. Led initially by Ferrari's Luca di Montezemolo, FOTA aimed to shape the new Concorde Agreement. A proposal for a 2010 budget cap caused a dispute, with some teams threatening a breakaway. The conflict was resolved with a revised agreement. FOTA represented all teams until 2011, after which prominent members withdrew. Internal disagreements and financial issues led to its dissolution in 2014.

2.4 Racing and Strategy

Race Weekend Format

A typical Formula 1 Grand Prix unfolds over three days, starting with two **Free Practice** sessions on Friday and concluding with a third on Saturday. The crucial **Qualifying Session** follows, setting the stage for the **Race** on Sunday. The goal of

qualifying is to determine the fastest laps, positioning drivers for the coveted **pole position**—the front-most starting spot on the grid.

Tyre Regulations

Drivers are allocated a limited number of tyres per race weekend: 13 sets of dry-weather tyres, four sets of intermediates, and three sets of full-wet tyres. The types of tyres used include three slick compounds (soft, medium, hard) and two wet-weather compounds. Teams must choose the most appropriate tyre for each session to balance speed, durability, and weather adaptation.

Qualifying Process

The qualifying format is a three-stage elimination system:

- **Q1**: 18 minutes to set fast laps; the slowest 5 are eliminated.
- **Q2**: 15 minutes for the top 15 drivers; another 5 are knocked out.
- **Q3**: A 12-minute battle among the top 10 for the best grid positions. If a driver's Q1 lap is not within 107% of the fastest time, they may not be allowed to race, though exceptions can be made.

Sprint Qualifying and Race

The **Sprint Race** format in Formula 1, introduced in 2021, is a shorter race held on selected weekends. In 2023, six Grands Prix featured sprint races: Azerbaijan, Austria, Belgium, Qatar, Texas (USA), and São Paulo. Typically around 100 km in distance, it lasts 25-30 minutes, with no mandatory pit stops. This race, until 2022, determined the starting grid for Sunday's main event and awarded points only to the top three finishers. It is preceded by a special "Sprint Shootout" qualifying session that sets the Sprint Race grid. The format encourages aggressive racing and offers fans additional competition during the race weekend, adding strategic challenges for teams.

Grid Penalties

Penalties affecting grid positions can arise from previous infractions, gearbox or engine changes, or failing scrutineering. Drivers excluded from qualifying may still start from the back of the grid at the stewards' discretion.

Race Start

The race begins with a **formation lap**, allowing drivers to warm up their tyres and check track conditions before assembling on the starting grid in qualifying order. Once all cars are in position, a light system signals the start: five red lights illuminate sequentially before extinguishing simultaneously to commence the race.

If a driver stalls or if conditions are unsafe, the start may be aborted, and a new formation lap is initiated. Races may also begin behind the **Safety Car** during hazardous conditions, with a standing restart after the track dries.

Race Structure

The winner is the first driver to cross the finish line after completing a designated number of laps. Since 1989, the standard race distance is 305 km (190 miles), although some street races, like Monaco, are shorter to stay within a two-hour time limit. Race officials can end the race early under extreme conditions, using red flags to pause the race while ensuring safety.

During the race, drivers can overtake each other, and backmarkers (slower cars) are shown a blue flag, indicating they must yield to leaders. If a driver completes less than 90% of the race distance, they are classified as "not classified."

Tyres

Formula 1 tyres have come a long way since 1950, evolving from simple, treaded designs to advanced high-performance slicks. In the 1960s, synthetic materials replaced cotton for greater durability, and in the early 1970s, slick tyres became the standard, offering more grip with increased contact area. Radial-ply tyres introduced by Michelin in 1977 brought further improvements in stiffness and handling.

In 1998, grooved tyres were introduced to limit cornering speeds, but F1 returned to full slicks in 2009 for better consistency. Pirelli became the exclusive supplier in 2011, introducing compounds with controlled degradation to add strategic complexity to races. A major shift came in 2022 with the switch to 18-inch rims, mirroring road car trends.

Tyre compounds are a key component of race strategy, providing teams with a range of options for different track conditions. Pirelli, the exclusive tyre supplier, offers six dry-weather compounds (C0 to C5), ranging from the hardest (C0) to the softest (C5). These compounds determine grip, durability, and heat resistance. Harder compounds (C0-C2) are durable, ideal for abrasive or high-temperature tracks but offer less grip. Softer compounds (C3-C5) provide better grip and quicker lap times, making them suitable for smooth, low-severity tracks but wear out faster.

Pit Stops

From the 1950s to Present Day: The Journey of Pit Stops

In the 1950s, pit stops lasted over a minute, often chaotic and lacking refinement. By the 1980s, the introduction of in-race refueling made pit stops a significant strategic element, though it raised safety concerns. Refueling incidents, such as Jos Verstappen’s 1994 and Felipe Massa’s 2008 pitstops, led to the permanent ban of in-race refueling in 2009, shifting the focus entirely to tyre changes. This change spurred teams to optimize tyre swap times, leading to today’s sub-three-second pit stops.

Speed and Precision: Modern Pit Stops

Today's pit stops are astonishingly quick. Over 20 specialists handle a car, each with a specific task—from tyre changes to minor adjustments. The current record, held by McLaren Racing, is a 1.80-second pit stop set in 2023 Qatar Grand Prix. Such feats require relentless practice, advanced equipment, and impeccable teamwork. In fact,

pit crews perform thousands of rehearsals to perfect their choreography, knowing that even a fraction of a second can impact the podium.

Pit Stops and Tyre Strategy

Pit stops in Formula 1 are pivotal moments where milliseconds can mean the difference between victory and defeat. Teams quickly change tyres and make necessary adjustments to maximize performance. Pirelli selects three tyre compounds—soft (red), medium (yellow), and hard (white)—for each race, tailored to expected track conditions. Softer tyres provide better grip for qualifying but wear out faster, necessitating more pit stops, while harder tyres offer durability, potentially saving time during the race.

Teams must use at least two different slick compounds, ensuring a minimum of one pit stop during a dry race. Wet (blue) and intermediate (green) tyres are deployed in varying conditions, each designed for specific levels of water on the track. Tyre strategy hinges on the circuit layout, weather forecasts, and anticipated tyre degradation, making the selection of compounds and timing of stops critical to maintain a driver's lead or help gain positions. Pitting during Safety Car or Virtual Safety Car periods can be advantageous.

By computing the number of pits stops per year, one can observe how pit stops strategies may differ throughout the years in the next Time Series Plot:

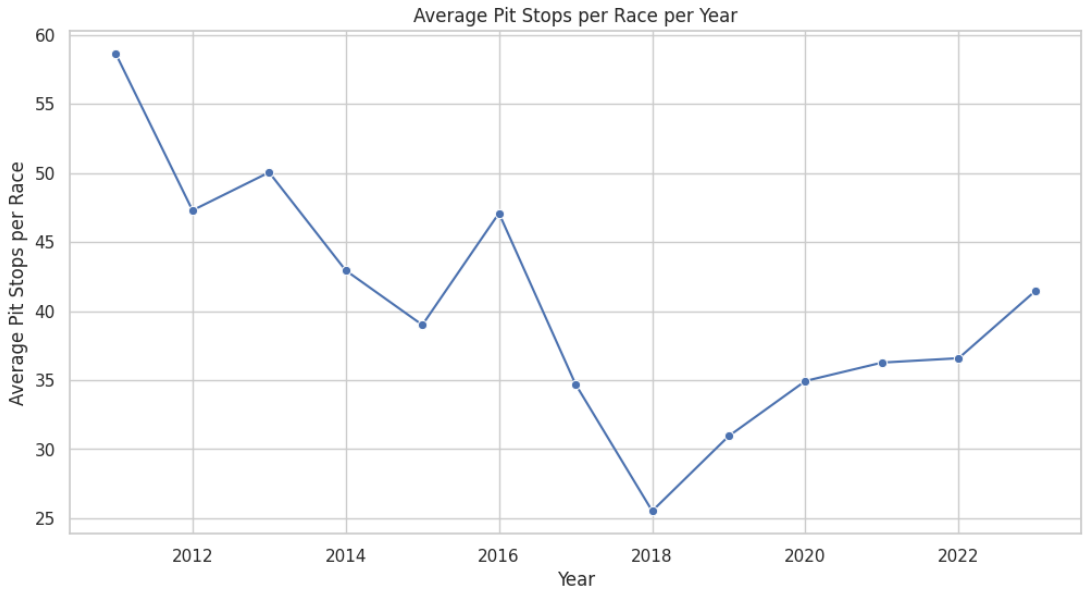

Figure 2.3: Time Series Plot Number of Pit Stops per Year

Figure 2.4 illustrates how pit stop strategies in Formula 1 differ over time. From 2011 to 2017, there is a noticeable decline, with a sharp drop in 2017 to around 25 pit stops per race, due to regulatory changes favoring durable tires and fewer stops. From 2018 onwards, pit stops gradually increase, reaching about 40 in 2022, suggesting a shift back to more aggressive strategies or circuit demands. Early years show greater variability, while later years stabilize around 35 to 40 stops. These trends highlight the

influence of regulations, tire technology, and team strategies on pit stop patterns, reflecting the sport's constant evolution.

During a race, executing a pit stop typically takes under three seconds, and precise communication between drivers and pit crews is essential. Mistakes can lead to lost time or penalties, significantly impacting race outcomes. Additionally, the sport's shift towards sustainability has led to reduced tyre blanket usage and the development of compounds that warm up quickly, further complicating strategy as teams adapt to evolving regulations while striving for optimal performance throughout the weekend.

Additionally, as Formula 1 has moved towards sustainable practices the usage of tyre blankets will be reduced and focused on compounds that warm up quickly, adapting to the FIA's evolving sustainability goals. This adds another layer of strategy as teams navigate new rules while aiming for optimal tyre performance throughout a race weekend.

Undercut and Overcut

In Formula 1, the **undercut** and **overcut** are two key pit stop strategies that can influence the outcome of a race.

Undercut involves a driver pitting earlier than a rival, sacrificing some track position for fresh tyres. The goal is to use the faster lap times on new tyres to close the gap or even overtake the rival when they pit later. This works best on tracks with high tyre wear, as worn tyres become significantly slower. In the 2019 Singapore Grand Prix, Sebastian Vettel’s early pit stop allowed him to jump ahead of rivals, securing a race win.

Overcut is achieved when a driver stays out longer on older tyres, hoping to gain an advantage as the rival struggles to bring their fresh tyres up to optimal temperature. This strategy is effective on tracks with low tyre degradation and limited overtaking opportunities. At the 2017 Monaco Grand Prix, Sebastian Vettel successfully stayed out longer than his competitors, overtaking them during his pit stop thanks to consistent lap times on older tyres.

These strategies are a complex part of the tactical battle in F1, and their success depends on track conditions, tyre wear, and a bit of luck.

Race Director and Safety Protocols

The **Race Director** oversees race management, ensuring compliance with regulations and handling incidents. The deployment of the **Safety Car** neutralizes the race during serious incidents, while the **Virtual Safety Car** slows the field without halting the race. Safety Cars can dramatically affect strategy, as they allow teams to conduct faster pit stops.

Safety Car

The **Safety Car** is deployed when an incident jeopardizes the safety of competitors or officials. It suspends the race, with drivers following the car in race order without overtaking. During this period, lapped cars may be allowed to un-lap themselves to facilitate a smoother restart. Once the danger is cleared, the race resumes with a

rolling start after the safety car returns to the pits. Pit stops during safety car periods can offer significant strategic advantages.

Mercedes-Benz has supplied the safety cars since 1996, with Aston Martin joining the rotation in 2021. Bernd Mayländer, a former racing driver, has served as the main safety car driver since 2000.

Virtual Safety Car

The **Virtual Safety Car (VSC)** was introduced after the serious accident involving Jules Bianchi at the 2014 Japanese Grand Prix. When deployed, drivers must maintain a minimum lap time, indicated by "VSC" signals displayed on their steering wheels and trackside panels. This system allows for safer race management without a full safety car deployment and was first implemented during the 2015 Monaco Grand Prix.

This format emphasizes the strategic complexity and split-second decision-making involved in each race weekend, showcasing why Formula 1 remains at the pinnacle of motorsport engineering and competition.

Points System

Formula 1 point scoring systems has evolved significantly since the sport's inception in 1950. Initially, only the top five finishers scored points, with an additional point awarded for the fastest lap. This system expanded over the years, moving to the top six finishers in 1960, and then to the top eight in 2003.

In 2010, the FIA revamped the system to award points to the top ten finishers, with the race winner earning 25 points—a significant increase compared to the earlier 10-point maximum. In addition, bonus points for the fastest lap returned in 2019, but only for those finishing in the top ten. The introduction of Sprint Races in 2021 brought additional opportunities for drivers to score points.

The current Formula 1 points system, established in 2010, awards points to the top ten finishers in each race as follows:
(From the start of the 2025 season the point for fastest lap is removed.)

Position	Race Points	Sprint Points
1st	25	8
2nd	18	7
3rd	15	6
4th	12	5
5th	10	4
6th	8	3
7th	6	2
8th	4	1
9th	2	-
10th	1	-
Fastest Lap: 1 point (only if the driver finishes in the top 10)		

Table 2.1: F1 current point system

A driver must complete at least 90% of the race distance to be eligible for points, and they can still score even if they retire, as long as they meet this requirement. Previously, from 1977 to 2021, a half points system was used if less than 75% of the race was completed, a rule last applied at the controversial 2021 Belgian Grand Prix, but it was replaced by a distance-dependent scale starting in 2022.

All-Time Points Leaders in Formula 1

(Source: https://www.mostlyf1.com/statistics/all-time-stats/driver-stats/most-points-with-the-current-points-system-driver/)

Many have tried with various parameters to calculate the best driver in existence. By making even the point calculating system through the years and considering the latest as an universal benchmark, would lead to fairly ranking the drivers based on the points they manage to gather and tell a compelling story of dominance and consistency.

1. Lewis Hamilton (5233.5 points) leads the pack, showcasing an unparalleled career marked by seven World Championships.
2. Michael Schumacher (3961 points), a pioneer of modern F1, follows closely, with numerous records to his name.
3. Sebastian Vettel (3321 points) stands as a testament to excellence, especially during his dominant years with Red Bull Racing.
4. Fernando Alonso (3193 points) remains a formidable competitor, known for his skill and versatility across teams.
5. Max Verstappen (2998.5 points) is the rising star, breaking records and redefining what's possible in the sport.
6. Kimi Räikkönen (2830 points), famed for his natural talent and unique personality, has left an indelible mark on F1.
7. Alain Prost (2508.5 points), the 'Professor,' was known for his tactical acumen and consistency.
8. Rubens Barrichello (1906 points), despite being overshadowed by teammates, remains one of the sport's most experienced drivers.
9. Ayrton Senna (1874.5 points) is revered for his incredible speed and charisma, embodying the spirit of racing.
10. Jenson Button (1844.5 points) rounded out the top ten, demonstrating resilience and adaptability throughout his career.

This unified points system highlights the legendary status of these drivers, showcasing their incredible achievements and contributions to the sport of Formula 1.

Physical Demands

Formula 1 driving is an intense physical challenge, with drivers burning around 1,000 calories per hour and losing 2–4 kg (4–9 lb) per race. They experience extreme

g-forces—up to 6.5 gs in corners and 6 gs when braking—making them feel weights as heavy as six times their body weight. The cockpit temperature can soar to 60 °C (140 °F), adding to the strain as drivers wear fireproof suits. Without power steering, steering forces reached 40–50 newton-meters (30–37 lb·ft), and braking requires about 150 kg (330 lb) of force. Lightweight physiques are crucial, as even a small increase in weight impacts performance.

Feeder Series

Most Formula 1 drivers begin in karting and progress through European single-seater series like Formula Ford, Formula Renault, and Formula 3, often advancing to GP2 (now FIA Formula 2). While champions like Lewis Hamilton and Nico Rosberg transitioned through these ranks, others, like Kimi Räikkönen, reached F1 via lower-tier competitions. American open-wheel racing has also influenced F1, with drivers like Mario Andretti and Jacques Villeneuve becoming world champions. Alternative paths, such as DTM or motorcycle racing, have occasionally led to F1 as well.

Women in Formula 1

Since 1950, only five women have raced in Formula 1, with Lella Lombardi being the sole points scorer. Women's involvement has grown beyond drivers, with roles in engineering and strategy contributing to increased female viewership, rising from 20% in 2019 to 40% in 2022, partly due to *Drive to Survive*.

F1 Academy

Launched in 2023, the F1 Academy is the first all-female single-seater series, designed to develop young female drivers. The championship uses spec Tatuus F4-T421 cars equipped with a 174-horsepower turbocharged engine from Autotecnica Motori and Pirelli tyres, ensuring equal performance across competitors.

From 2024, all F1 Academy races will align with Formula 1 weekends, featuring two races per event. This integrated schedule increases visibility for the series while granting Super Licence points to top performers and allowing wild card entries to showcase local talent.

Notable Female Drivers

Maria Teresa de Filippis was the first woman to race in F1 (1958), and Lella Lombardi remains the only woman to score points (1975). Desiré Wilson is the only woman to win an F1 race, taking victory in the 1980 Aurora AFX F1 series.

Formula 1 Grands Prix

The number of Formula 1 Grands Prix in a season has varied significantly, from just seven races in the inaugural 1950 World Championship to a peak of 22 races in 2021, 2022, and 2023. As of 2024, a record 24 races are scheduled. Originally, six of the seven races in 1950 were held in Europe, with the Indianapolis 500 being the only non-European event. Over time, F1 expanded globally, with Argentina hosting the first South American Grand Prix in 1953 and Morocco the first African race in 1958. The Monaco Grand Prix, established in 1929, is one of the sport's most prestigious

events and has been continuously held since 1955, except for 2020. Night races began with the 2008 Singapore Grand Prix, changing the traditional daytime format.

Circuits Overview

Formula 1 races must take place on FIA-grade one tracks, the highest safety rating for circuits. Since the first World Championship Grand Prix at Silverstone in 1950, 77 circuits have hosted F1 races. Some classic tracks, like Nürburgring and Spa-Francorchamps, have seen various layout changes for safety and modernization. Notably, Monza has hosted the most World Championship races, missing out only once in 1980. The most recent circuit addition is the Las Vegas Strip Circuit, which debuted in 2023. While the lap layout and distance can vary significantly, most circuits run in a clockwise direction, with a few exceptions. Each circuit typically features a main straight for the starting grid and a pit lane adjacent to it, where teams service their cars during races. While most tracks are purpose-built, several races occur on modified public streets, including famous venues like Monaco, Melbourne, Singapore, Baku, Miami, Jeddah, and Las Vegas. The Monaco Grand Prix uniquely requires a minimum race distance of only 260 km, compared to the standard 305 km for other races.

To highlight the expansion of Formula 1 over the years, the following timeseries plot illustrates the number of Grands Prix held each year:

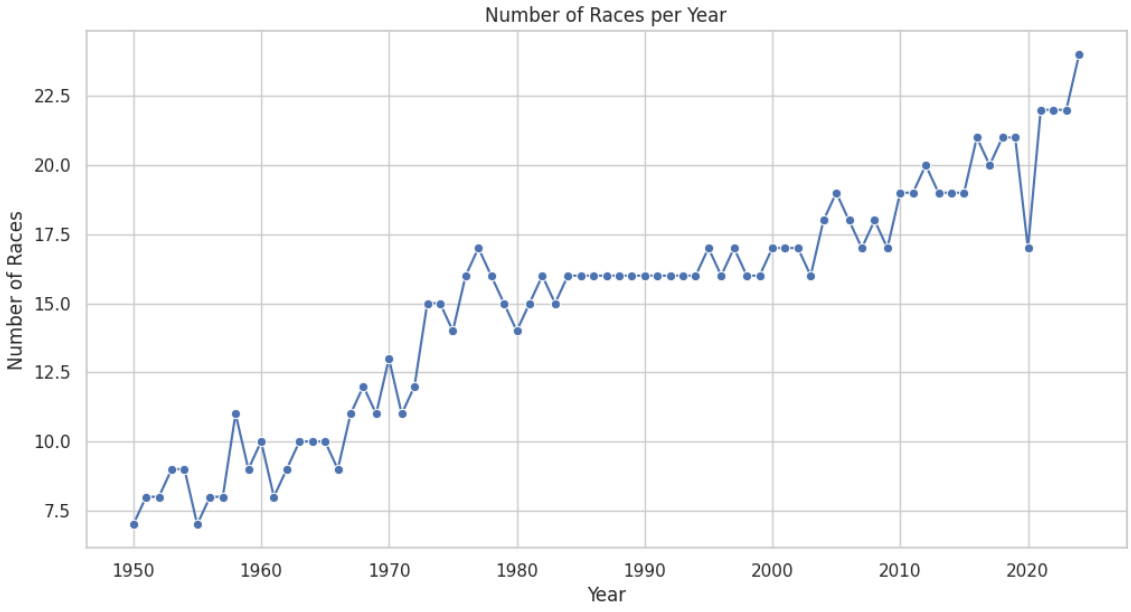

Figure 2.4: Time Series Plot Number of Races per Year

Figure 2.4 shows the steady growth in the number of Grands Prix held annually from 1950 to 2023, reflecting Formula 1's global expansion. In the early years, the calendar remained relatively stable, with 7 to 11 races per season during the 1950s and 1960s. Growth became evident in the 1970s and 1980s as more circuits and countries were added, stabilizing at around 16 races per year by the early 2000s. A significant expansion occurred after 2010, surpassing 20 races annually, with a peak of 22 races in 2023. The temporary decline around 2020-2021 corresponds to disruptions caused

by the COVID-19 pandemic. Overall, the trend highlights Formula 1's ongoing efforts to increase its global footprint and audience engagement while adapting to logistical challenges and market demands.

Circuit design has evolved to enhance driver safety, exemplified by tracks like the Bahrain International Circuit, constructed in 2004 by Hermann Tilke. Although some newer circuits have faced criticism for lacking the character of classic tracks like Spa-Francorchamps and Imola, they generally meet modern safety standards. Since 2012, several new tracks, including the Circuit of the Americas in Austin and the Baku City Circuit in Azerbaijan, have been added to the F1 calendar. Notably, the Algarve International Circuit debuted for the Portuguese Grand Prix in 2020, and the Las Vegas Grand Prix was introduced in 2023, marking a significant expansion of F1's global presence.

F1 Kids

Formula 1 recently launched the F1 Kids Broadcast at the 2023 Singapore Grand Prix, following a successful debut at the Hungarian Grand Prix. This kid-friendly broadcast uses cartoon avatars of drivers and simple “Did you know?” facts about terms like tyres and DRS, making the sport accessible for children and new fans alike.

2.5 Financial Dynamics in F1: Revenue, Costs, and Profitability

The financial landscape of Formula 1 has evolved significantly, particularly with the introduction of cost caps aimed at enhancing competitiveness among teams. In March 2007, F1 Racing estimated that total spending by all eleven teams in the 2006 season reached approximately $2.9 billion, with top teams like Toyota, Ferrari, and McLaren each spending over $400 million. In contrast, smaller teams like Super Aguri operated on budgets as low as $57 million. This initiative seeks to create a more equitable competitive environment.

However, controversies surrounding revenue sharing remain prevalent. Smaller teams have raised concerns that profit distribution favors established teams, creating an uneven playing field. In 2015, Force India and Sauber filed a complaint with the European Union regarding the governance of Formula 1 and the fairness of its financial practices. This tension highlights the ongoing challenge of ensuring that all teams can compete effectively, as smaller teams often struggle to secure adequate funding.

The costs associated with building new permanent circuits can exceed hundreds of millions of dollars, while converting public roads into temporary tracks is typically less expensive. Permanent circuits, like the Shanghai International Circuit, which cost over $300 million to construct, can generate revenue year-round through leasing for private events and other motorsport activities. Meanwhile, driver salaries in Formula 1 are among the highest in motorsport, with Lewis Hamilton earning $55 million in 2021, making him the sport’s highest-paid driver.

The economics of operating a Formula 1 team reveals that while the sport is immensely costly, with substantial expenses on logistics and components, the potential rewards extend beyond immediate financial gains. Sponsorship deals play a crucial role, often amounting to over $50 million annually, enabling teams to offset high operational costs. Additionally, the journey to becoming an F1 driver involves significant financial backing, sometimes prioritizing sponsorship over raw talent.

The COVID-19 pandemic also impacted Formula 1 finances, resulting in a reported revenue loss of $122 million in the second quarter of 2020 due to delays in the championship's start. In contrast, the company grossed $620 million in revenue during the same quarter the previous year. However, as of October 2024, Formula 1 is poised for recovery, with increasing viewership and sponsorship deals essential for maintaining the financial health of teams and ensuring continued investment in the sport.

Cost Cap

Since its inception in 1950, Formula 1 has operated without strict financial limitations, leading to a substantial performance gap between well-funded teams and their financially modest rivals. Historically, this disparity made it nearly impossible for smaller teams to compete, and some even faced the threat of bankruptcy. For example, in 2019, top teams like Mercedes and Ferrari spent around $420 million and $435 million, respectively, while smaller teams like Williams and Haas operated on budgets of only $125 million and $150 million. This growing disparity, combined with concerns over financial sustainability, led the FIA to implement a cost cap in 2021 to promote a more level playing field.

The initial cap was set at $175 million, but it was reduced to $145 million in response to the economic impacts of the COVID-19 pandemic. This figure continued to decrease to $140 million in 2022 and was further adjusted to $135 million for the 2023 season. Future adjustments are linked to inflation, and additional allowances are made for any extra races beyond the standard calendar. The cost cap aims to balance competition, emphasizing engineering skill over financial dominance while contributing to the sport’s sustainability by reducing excessive R&D spending and the associated carbon footprint.

The 2022 season was particularly crucial for assessing the cap's impact, coinciding with major regulatory changes that allowed teams to develop new cars from scratch. The cost cap covers a wide range of expenditures, including car components, equipment, and operational costs. However, it notably excludes driver salaries, the salaries of top executives, and marketing expenses. Critics argue that these exclusions may still favor larger teams with superior infrastructure.

Advocates of the cost cap highlight its potential benefits, such as improved competition, financial stability for smaller teams, and environmental responsibility. Notably, McLaren's mid-season turnaround in 2023—from a slower team to one of the fastest on the grid—underscored how strategic resource management can offset financial disadvantages. This transformation illustrates how engineering ingenuity can

make up for budgetary constraints, fostering a more dynamic and unpredictable racing environment.

The enforcement of the cost cap is managed by the Cost Cap Adjudication Panel, consisting of six judges selected by the FIA and participating teams. Penalties for breaching the cap are tiered: minor overspends (less than 5% above the limit) can result in point deductions, fines, or limitations on wind tunnel testing. More severe violations, exceeding 5%, could lead to disqualification from the championship. Despite the challenges in compliance and concerns over enforcement, the cap's introduction marks a significant shift toward a fairer, more balanced, and sustainable Formula 1, attracting interest from potential new teams and manufacturers.

2.6 F1 Impact in Automotive Life

Formula 1 has profoundly shaped the automotive industry with advancements in technology and materials.

- **Aerodynamics**: F1's expertise in airflow dynamics has enhanced road cars' efficiency, reducing drag and improving fuel economy. Modern designs are inspired by the principles of downforce and airflow learned on the racetrack.
- **Engine Technology**: F1 pioneered turbochargers and hybrid systems, now standard in consumer cars for better performance and emissions reduction. Energy Recovery Systems (ERS) have influenced modern hybrid vehicles, balancing power with efficiency.
- **Lightweight Materials**: Carbon fiber, first used in F1 for chassis strength, is now common in road cars, providing durability without weight. These materials contribute to greater agility and safety.
- **Brakes and Suspension**: F1-inspired suspension systems enhance ride quality and handling, while carbon-ceramic brakes, known for their endurance and superior stopping power, have become a feature in high-performance vehicles.
- **Transmission**: Paddle-shift gearboxes, perfected in F1 for rapid gear changes, are now standard in many modern cars, allowing drivers to change gears quickly and seamlessly.
- **Cross-Industry Influence**: F1 technology extends beyond cars—McLaren's collaboration in cycling, F1-derived boat designs for sailing, and energy storage innovations for public transportation showcase the broad impact.

Overall, F1's relentless pursuit of speed, efficiency, and safety has pushed the boundaries of what's possible in automotive design, making everyday vehicles safer, more efficient, and more enjoyable to drive.

Future of F1 Technology

The future of Formula 1 is set to bring transformative changes, blending performance with sustainability. A key focus is the introduction of 100% sustainable

fuels by 2026, aiming for carbon neutrality. This shift could revolutionize the automotive industry's environmental impact. Battery advancements will enhance hybrid power units with greater efficiency, energy density, and reduced weight. Although fully autonomous F1 cars aren't imminent, technology like advanced AI for strategy and safety features is likely. Additionally, regulatory changes by the FIA will guide innovations like electric propulsion and bio-composite materials, keeping F1 competitive and eco-friendly.

2.7 The Impact of Data Analytics & Technology on F1 Performance

In modern Formula 1, data analytics and advanced technology have fundamentally reshaped the sport, significantly influencing car performance, design, driver training, race strategies, and overall team operations. Cutting-edge tools like Catapult's RaceWatch and Circuit Manager are at the forefront of this transformation, providing teams with detailed insights that are crucial for optimizing every aspect of their racing activities. As a result, Formula 1 has evolved into a highly data-driven discipline, where strategic precision often takes precedence over raw speed.

Data plays a multifaceted role in Formula 1, supporting everything from performance optimization to predictive race strategies. By closely monitoring parameters such as tyre pressure, aerodynamics, and driver behavior, teams can make targeted adjustments that maximize car efficiency and reliability. Advanced data analysis tools allow teams to forecast key variables like race conditions, fuel consumption, and tyre wear. This predictive capability is instrumental in developing flexible race strategies that can adapt to shifting conditions throughout a Grand Prix.

Real-Time Insights and Telemetry

Telemetry technology is a cornerstone of Formula 1, with each car equipped with over 300 sensors that collect real-time data on critical performance metrics. These metrics include engine output, tyre temperature, fuel levels, and aerodynamic efficiency. This vast amount of data is transmitted wirelessly to teams, who utilize it to make split-second strategic decisions—whether adjusting tyre strategies, optimizing fuel consumption, or making immediate alterations to car setup. The integration of machine learning and predictive analytics further enhances this process by forecasting tyre degradation, fuel efficiency, and potential race scenarios, allowing teams to swiftly adapt their strategies as the race unfolds.

Data-Driven Car Development and Maintenance

The influence of data-driven decision-making extends beyond the track into the realms of car development and maintenance. Teams analyze historical race data and simulations to refine aerodynamics, select optimal pit stop windows, and perfect pit stop routines down to fractions of a second. The use of machine learning enables teams to simulate millions of race scenarios, saving valuable time and resources. This has led to improved design efficiency, reduced costs, and a deeper understanding of how each component contributes to overall performance.

Leveraging AI and Machine Learning

AI and machine learning have become indispensable tools in Formula 1, helping teams interpret and utilize the enormous volume of data generated during races and testing. These technologies have been integrated into various aspects of F1 operations, from optimizing car setups to forecasting race outcomes. Teams rely on AI-powered simulations to predict tyre wear, identify aerodynamic weaknesses, and model countless race scenarios under different conditions. These insights not only help engineers fine-tune car design but also provide drivers with virtual training opportunities, improving their performance without risking damage to valuable assets.

Cloud Computing and Collaboration

Cloud computing platforms, such as those provided by AWS, have made a significant impact on how data is processed and shared among teams. Engineers can now collaborate in real-time from various locations, analyzing data simultaneously and contributing to faster decision-making processes. This has been particularly useful for mid-race adjustments and for building "digital twins"—virtual replicas of physical cars that allow teams to experiment with design changes and race strategies in a virtual environment before implementing them on track.

Enhancing the Fan Experience

Beyond the technical aspects, data analytics and AI are also revolutionizing the fan experience. Machine learning algorithms are used to generate real-time insights during races, which are shared with audiences through broadcasts, providing deeper understanding and engagement. Advanced analytics allow fans to track car performance, follow live race dynamics, and access detailed statistics, bringing them closer to the action. Teams are exploring augmented reality (AR) and virtual reality (VR) platforms to provide immersive experiences, turning fans into active participants in the sport.

The Future of Data in Formula 1

Looking ahead, Formula 1 is poised to embrace even more sophisticated technologies. Big Data, AI, and the Internet of Things (IoT) are expected to dominate the future landscape, enabling teams to refine car configurations, make real-time adjustments with unparalleled precision, and predict technical failures before they occur. These innovations will not only enhance racing performance but will also contribute to sustainability efforts, allowing teams to monitor and reduce carbon emissions through precise data analysis.

Cross-Industry Relevance of F1 Data Analytics

The data-driven methodologies employed in Formula 1 extend beyond the racetrack, offering valuable lessons for other industries. The predictive models, real-time monitoring, and efficiency improvements that define F1 racing are increasingly relevant to fields like manufacturing, healthcare, and finance. The sport’s emphasis on precision, adaptability, and innovation illustrates the powerful role data analytics can play in optimizing performance and decision-making across sectors.

2.8 Honorable Remarks

Top 10 F1 Drivers by Win Percentage

Measuring the greatest drivers in F1 history is challenging due to the evolving nature of the sport, but win percentage offers a unique metric. Leading the list is Juan Manuel Fangio, who won almost half of his races in the 1950s, dominating the early years of F1. Alberto Ascari follows with an impressive 40.63%, achieving major success with Ferrari in the early 1950s. Jim Clark stands out as a 1960s icon with a 34.72% win rate, displaying unmatched speed before his untimely death. In the modern era, Lewis Hamilton and Max Verstappen have climbed the ranks, each nearing 30% thanks to dominant periods with Mercedes and Red Bull, respectively. Michael Schumacher, a seven-time champion, maintains a nearly identical win rate, having set records with Ferrari. Legendary figures like Jackie Stewart, Ayrton Senna, Alain Prost, and Stirling Moss round out the top ten, each defining their era with remarkable success and unforgettable rivalries. These drivers' win percentages highlight their exceptional talent, despite the vastly different challenges and seasons they faced.

Additionally, creating bar plots for all-time race winners and the accumulated constructor points across various point systems provides valuable insights into the leading figures and teams in the industry.

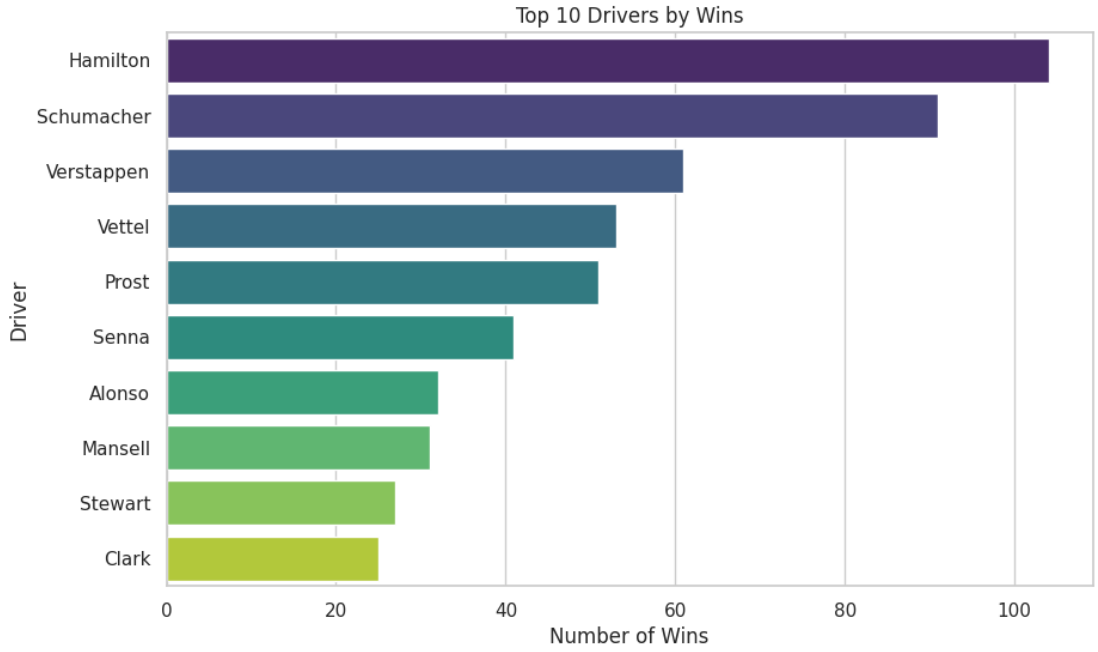

Figure 2.5: Bar Plot of the Top 10 Drivers by Wins

As observed, Lewis Hamilton leads with over 100 victories, showcasing his unparalleled dominance in the history of Formula 1, followed by Michael Schumacher with approximately 90 wins, a testament to his legendary career. Max Verstappen, ranked third, stands out as an active driver with a notable win count, suggesting his potential to challenge long-standing records in the future. Sebastian Vettel and Ayrton Senna follow closely behind, with Senna's record being particularly remarkable given

his untimely passing. Both Senna's and Schumacher's careers invite speculation about what more they could have achieved had circumstances allowed them to continue competing in the sport.

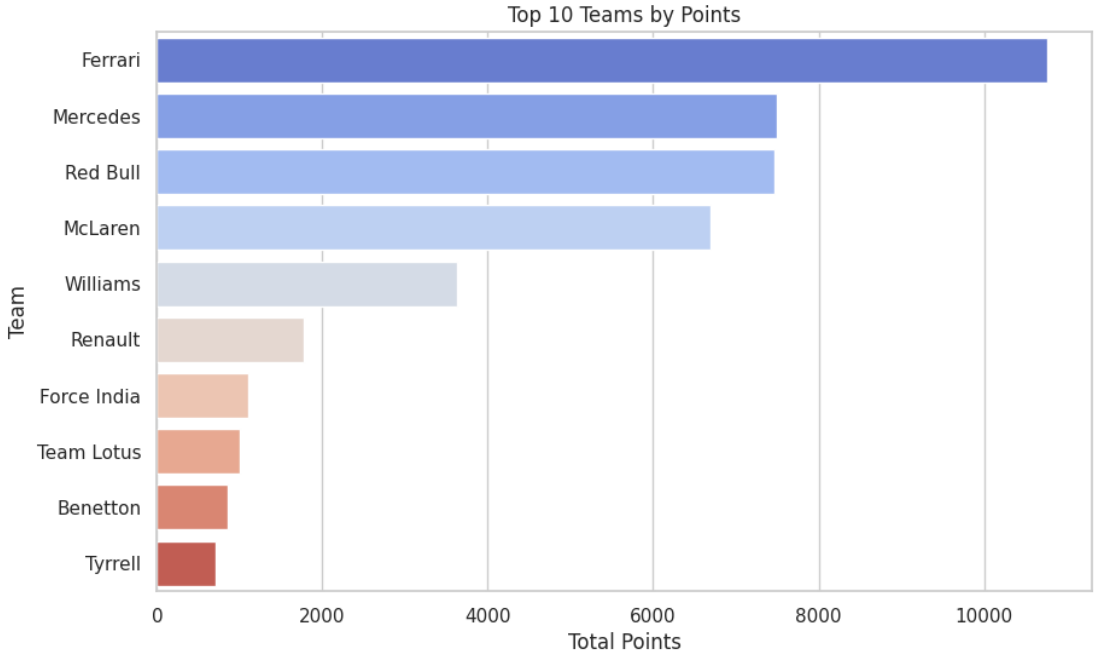

Figure 2.6: Bar Plot of the Top 10 Constructors by Points

As one can see, Ferrari leads by a significant margin, with over 10,000 points, reflecting the team's long-standing success and dominance in the sport. Mercedes and Red Bull follow in second and third places, respectively, showcasing their remarkable performances in the modern era of Formula 1. McLaren and Williams complete the top five, underscoring their historical significance despite recent challenges. It is important to note that the total points shown have been collected under various points systems used throughout Formula 1's history. In recent years, the points awarded per race have increased significantly compared to earlier periods, giving an advantage to more recently established teams such as Mercedes and Red Bull, which have benefited from the modern scoring systems.

Ayrton Senna: Life and Legacy

Ayrton Senna was born on March 21, 1960, in São Paulo, Brazil. He began karting at a young age and quickly demonstrated remarkable talent, leading to his entry into Formula 1 in 1984 with Toleman. Over his career, Senna became known for his fierce determination and exceptional skill, particularly in qualifying sessions. He won three World Championships (1988, 1990, and 1991) driving for McLaren and was celebrated for his intense rivalry with Alain Prost, which captivated motorsport fans worldwide.

Senna's contributions to Formula 1 extended beyond his racing achievements. He was a vocal advocate for safety in the sport, pushing for improved regulations and practices after witnessing several tragic accidents. His philanthropic efforts were equally notable; he founded the Ayrton Senna Institute, which aimed to improve

education for underprivileged children in Brazil. Senna's legacy continues to inspire drivers and fans alike, highlighting the importance of both skill and compassion in the world of motorsport.

Ayrton Senna’s death during the 1994 San Marino Grand Prix led to extensive legal repercussions due to Italian law requiring investigations into fatalities. Following the crash, which was attributed to a steering column failure caused by modifications made by the Williams team, key team members were charged with manslaughter. While the initial trial in 1997 resulted in acquittals, the Supreme Court of Cassation later determined that poorly designed modifications contributed to the accident. His death led to a significant overhaul of safety regulations in Formula 1, resulting in improved car designs, better circuit safety standards, and increased medical protocols. These changes have contributed to making the sport safer for drivers, ensuring that Senna's legacy endures not only through his remarkable achievements but also through the advancements in safety that followed his untimely death.

Niki Lauda: A Racing Legend

Niki Lauda, born on February 22, 1949, in Vienna, Austria, was one of the most iconic figures in Formula 1 history. Known for his intense work ethic and analytical mind, Lauda won his first World Championship in 1975 driving for Ferrari, followed by another title in 1977. However, his career was marked by a life-threatening crash at the Nürburgring in 1976, where he suffered severe burns and inhaled toxic gases, leading to a lengthy recovery. Remarkably, he returned to racing just six weeks later, showcasing his incredible determination.

After his accident, Lauda became a key advocate for improved safety measures in Formula 1, pushing for changes that would protect drivers. He secured his third championship in 1984 driving for McLaren, further solidifying his legacy. In addition to his racing career, Lauda founded and managed several airlines, including Lauda Air, demonstrating his entrepreneurial spirit. Lauda's influence extended beyond the track, as he became a respected commentator and a mentor for young drivers. His passing on May 20, 2019, marked the end of an era, but his contributions to motorsport and aviation continue to inspire many.

Scuderia Ferrari: Formula 1 Most Successful Team

Scuderia Ferrari, established by Enzo Ferrari in 1929, has been a cornerstone of Formula 1 since the championship's inception in 1950. The team holds records for the most Constructors' Championships (16) and Drivers' Championships (15), cementing its place as the sport's most successful outfit. Ferrari's roster has included celebrated drivers like Michael Schumacher, who won five consecutive titles (2000–2004), and Niki Lauda, known for his precision and technical expertise.

Operating from Maranello, Italy, Ferrari is renowned for its engineering capabilities, with a focus on aerodynamics, power unit design, and continuous innovation. The Fiorano Circuit serves as its dedicated testing facility, enabling on-site development of race cars.

While Ferrari's history includes periods of both dominance and challenges, the team remains a central figure in Formula 1. Its contributions to the sport include advancements in technology and a reputation for fostering top-tier talent in drivers and engineers. Scuderia Ferrari continues to compete at the highest level, upholding its legacy of excellence.

2.9 Literature Review

This section provides a review of recent articles from around the world that align with our analysis, and in some cases, utilize the same datasets. These works explore similar themes, offering valuable insights and methodologies that parallel our own research, enriching the understanding of data analysis and technology's impact on Formula 1.

Jenkins (2010) examined the relationship between technological changes and competitive performance in Formula 1 racing over a 57-year period. The findings showed that established firms often struggle to adapt to sudden changes, which allows new competitors to enter the market. The study revealed a few firms that managed to maintain their competitive edge despite multiple technological shifts. It suggests that, alongside the ability to create new advantages (dynamic capabilities), these firms also possess what are called sustaining capabilities—resource configurations that give them extra time to adjust to technological advancements.

Calmon and Albi (2020) introduced innovative methods for estimating the number of clusters in a finite population of objects ranked by several judges, based on a hierarchical adaptation of the classical Plackett–Luce model. The two methods developed, HPL and FHPL, cater to different assumptions, with HPL offering greater flexibility and FHPL providing faster computation by restricting the parameter space. A comprehensive simulation study demonstrated the methods' effectiveness, with both achieving high accuracy in cluster estimation across various datasets. Notably, they outperformed established methods, showing the highest success rates and smallest errors. The FHPL method is recommended for large datasets due to its simplicity, while both methods are equally useful for smaller datasets.

To demonstrate the practical application of these methods, the study analyzed a Formula 1 dataset that included rankings derived from the fastest lap times of seven drivers in the 2015–2018 seasons. Both methods indicated seven clusters, suggesting no statistical ties between drivers, with Lewis Hamilton consistently ranked as the best. Future research could relax the assumption of homogeneity among judges (in Formula 1, the FIA race officials), explore different prior distributions, and adopt simulation techniques to enhance the computational efficiency of HPL. Additionally, integrating HPL with multistage models and comparing the results with alternative clustering methods presents promising directions for further exploration.

Wesselbaum and Owen (2020) investigated the impact of Pole Position in Formula 1 racing on race outcomes, specifically addressing whether it provides a significant advantage to drivers. The analysis reveals that securing Pole Position offers a notable edge, translating to approximately two positions gained at the finish line or a roughly

10 percentage point increase in the probability of winning. These findings account for various confounding factors, including starting position, weather conditions, driver skill, track characteristics, and constructor performance. Additionally, the advantage associated with Pole Position varies over time, influenced by changing rules, technological advancements, and other factors. Overall, the study highlights the critical role that Pole Position plays in shaping race outcomes in Formula 1 history.

Budzinski and Feddersen (2020) assessed competitive balance within Formula 1, examining it through three dimensions: within-race, within-season, and inter-season. While some indicators show a gradual improvement in competitive balance over time, evidence suggests a trend reversal since 2010, with certain metrics indicating reduced balance. The study highlights that using different methodologies, such as Markov chain-based analyses or Spearman rank correlations, could enhance the dynamic measurement of competitive balance. Additionally, expanding the analysis to include factors like fan preferences for local heroes may offer a deeper understanding of how competitive balance affects audience engagement in F1.

A.Piquero et al. (2021) confirmed the ongoing interest in ranking professional athletes, particularly Formula 1 drivers in the turbo era. Using group-based trajectory methods, it successfully identified Lewis Hamilton and Nico Rosberg as the top performers based on accumulated points from 2014 to 2019. The analysis categorized the 45 drivers competing during this period into three distinct groups: the top-performing group consisting of Hamilton and Rosberg, a second group of six drivers—including Bottas, Leclerc, Räikkönen, Ricciardo, Verstappen, and Vettel—and a third low-performing group, where many drivers scored zero points.

Furthermore, this research highlights the potential for future studies to track not only drivers and their points but also car manufacturers and upcoming changes to rules and regulations, such as team budgets. Such factors may influence how drivers are classified based on performance. As the sport continues to evolve, understanding these dynamics will be crucial for analyzing competitive success in Formula 1 and refining the methodologies used for ranking drivers.

Von Schleinitz et al. (2021) proposed the VASP framework for robust time series prediction outperformed the other investigated approaches on our dataset, particularly in scenarios where anomalies were present. This framework combines a variational autoencoder (VAE), an anomaly detector, and LSTM predictors, allowing for selective replacement of anomalous input data before making predictions. This approach is innovative, as it has not been published before, and it effectively addresses the significant accuracy drop often associated with anomalies in input signals. The selective prediction strategy exhibited almost no disadvantages, maintaining high accuracy with regular data while reducing error by 13% to 33% on average, and up to 70% in specific cases when faced with anomalous inputs.

One of the key advantages of the VASP framework is its seamless integration into existing models; it can enhance any prediction model by simply adding a VAE and an anomaly detector, without the need for retraining or labeled anomalies. This makes it particularly suitable for application in motorsport, where the use of machine learning models is limited due to interpretability issues and uncertainty in model behavior. The

study aimed to robustify these machine learning models, addressing a crucial aspect for their adoption by experts in the field. Future research could involve applying our method to various datasets from motorsport and other areas, as well as investigating the impact of prediction signal properties on accuracy, given the notable differences observed in our study.

Sicoie (2022) researched the effectiveness of ensemble learning methods and regression techniques in predicting Formula 1 (F1) championship standings. Data was collected using API retrieval and web scraping from reliable sources, with careful selection guided by domain knowledge. After preprocessing and feature creation, data modeling was conducted using pipelines. Although initial predictions were unsatisfactory, results aggregated over the season showed a high correlation with actual standings, with fine-tuned models achieving notable correlation scores (RFR ρ: 0.902, GBR ρ: 0.903, SVR ρ: 0.883). For the top 10 positions, 63% of predictions were accurate within a margin of 3 positions, and over 53% were accurate within a stricter margin of 2 positions.

Future research could improve performance by fine-tuning algorithms, exploring deep learning techniques, and incorporating additional features such as lap times and racing incidents. The relationship between drivers and constructor teams also requires further investigation. In conclusion, the framework has shown considerable potential in predicting F1 championship standings. Further refinement and the inclusion of more granular data could enhance accuracy, with ongoing collaborative efforts in F1 predictive analytics leveraging various data sources to improve insights and forecasts.

Rondelli (2022) explored the application of deep learning and neural network algorithms to predict tyre strategy during Formula 1 races. The objective was to demonstrate how these algorithms can effectively determine the optimal timing and compounds for tyre changes in a high-performance environment. Utilizing telemetry data from the "FastF1" API, the authors developed LSTM, GRU, and MLP models, achieving promising results—specifically, the GRU model outperformed the blind classifier by 25% in accuracy. These findings underscore the potential of deep learning in tyre prediction, paving the way for more advanced models in the future. In conclusion, the integration of deep learning and AI in tyre strategy could revolutionize Formula 1, enabling teams to analyze vast data sets and make informed decisions that enhance performance and success on the track. As technology advances, the authors anticipate an even greater role for these methods in motorsports.

Niamh (2023) analyzed the impact of weather on Formula 1 races, pioneering an underexplored area by creating and exploring a comprehensive dataset spanning the 2005-2022 seasons across 348 races on 37 racetracks worldwide. Data integration involved combining F1 performance statistics with weather conditions, collected for six hours around each race’s scheduled local time. Visualization tools unveiled complex relationships between precipitation and race speeds, revealing, for example, how increased rainfall did not consistently correlate with slower lap times. The project produced an interactive tool allowing users to access race weather data, lap times, and precipitation insights across different years and circuits. Machine learning models were also applied, comparing scenarios with scaled and unscaled data to

measure performance variations, though limitations were noted due to gaps in pre-2005 race time data, suggesting that a larger dataset could improve model accuracy.

Tejada (2023) investigated the application of various hyperparameter settings to improve the performance of three machine learning algorithms—Random Forest, Support Vector Machines (SVM), and Neural Networks—in predicting whether a driver should make a pit stop during Formula 1 races from 2019 to 2022. Two target variables were analyzed: "has pit stop," which indicates if a pit stop occurred, and "good pit stop," assessing whether the stop positively impacted the driver's position. The findings highlight the complexity of predicting pit stops, showing promise in determining occurrences but indicating that accuracy still requires improvement. Predicting "good pit stops" proved particularly challenging due to the uncertainty of outcomes, as not all drivers can benefit simultaneously from pit stops.

While the models can assist teams in making informed decisions, they should not be solely relied upon. The study underscores the potential of machine learning to enhance pit stop predictions, despite the difficulty of achieving perfect accuracy. Future research could expand the variables in the models by collaborating with teams for unique data and analyzing past lap trends. Moreover, developing a real-time pit stop prediction system could revolutionize strategies by providing timely insights during races, enabling adaptive and strategic decision-making that enhances competitiveness in Formula 1.

Aiginiti (2024) presented a thesis that applies machine learning to predict F1 race outcomes using public data on drivers, constructors, qualifying results, and race performance. Several algorithms, including regression models, ensemble decision-tree methods, and neural networks, were tested, with piecewise polynomial splines and Extreme Gradient Boosting (XGBoost) yielding the best predictive results. Despite these models performing better, the predictions were still limited by the complexity of racing dynamics and the influence of unpredictable factors like luck. This study illustrates the potential of machine learning for strategic insights, while highlighting the challenges of achieving accurate predictions in such a fast-paced and variable environment.

Belgaid (2024) examined the impact of FIA regulations on safety, racing dynamics, and excitement in Formula 1 from 1990 to 2023. Findings indicate that while regulations aimed at improving driver safety have not shown a statistically significant link to race fatalities—likely due to their rarity and methodological constraints—they have not negatively impacted overtaking opportunities. The introduction of DRS (Drag Reduction System) significantly increased overtakes, counteracting some adverse effects of regulations that have led to heavier cars. Overall, FIA regulations have contributed to long-term safety improvements and enhanced racing excitement, but ongoing evaluation and refinement are essential to ensure they continue to benefit the sport effectively.

Garcia (2024) utilized the "Formula 1 World Championship (1950-2020)" dataset to explore tyre wear's influence on lap times and optimize race strategy in Formula 1. Data preprocessing involved applying the Interquartile method for outlier removal, leading to more accurate regression models of tyre performance over time. Initial

strategy was refined using prior race data to compensate for sparse early-stage information. The optimization of tyre choices and stint lengths was formulated as a Mixed-Integer Quadratic Programming (MIQP) problem, capable of near-instant solutions. This approach was tested on 2022 season races, showing notable differences from some real-world strategies—especially in cases where drivers deviated from optimal tyre usage. It was observed that while tyre strategy impacts race outcomes, other factors like driver skill and vehicle performance also significantly influence final results. Future work could incorporate additional race variables like weather conditions and yellow flags to enhance realism in race strategy modeling.

R.Nimmala and J.Nimmala (2024) explored Formula 1's transformation into a data-centric sport, where advanced sensor networks, real-time data streaming, and analytics shape competitive strategies and performance optimization. Each F1 car is outfitted with hundreds of sensors, generating data on critical aspects such as tyre wear, engine performance, and aerodynamic efficiency, all channeled through the Engine Control Unit (ECU) and transmitted using technologies like Apache Kafka. This setup enables immediate, high-throughput data analysis crucial for split-second decisions. Case studies, such as Mercedes F1's use of sensor data to adjust strategies mid-race, underscore how data drives competitive advantage. Tools like Alteryx and predictive analytics support teams in anticipating conditions, optimizing setups, and making real-time adjustments. Looking ahead, the integration of AI, IoT, and machine learning is expected to further enhance race strategy, fan engagement, and car design, positioning F1 as a model of cutting-edge data science in sports.

Chapter 3: Descriptive Statistics

The current chapter outlines the source of raw data, which underwent extensive processing and cleaning to produce a merged working dataset spanning 13 closed seasons (2011 till present date). The steps taken to ensure consistency and reliability are described in detail, including the selection and creation of key numerical and categorical variables. The chapter also presents several depictions of the working dataset variables both statistical and graphical to provide an overview of their structure and characteristics. Finally, comparisons grouped by key variables are performed to explore potential dependencies and relationships within the data. All the data processing, analysis, and visualization tasks were performed using Python Programming Language. The Python code used to generate the diagrams is presented in detail in the attached appendix at the end of the present thesis (Appendix 1: Implementation Code of the Present Study).

3.1 Data Presentation, Preparation and Processing

The raw data originates from https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020 that derives data from http://ergast.com/mrd/ and provides a wide variety of data in csv Excel files. The chronological line starts from the start of the sport 1950 up to 6 months back from the current date thus offering many prospectives to analyze. Specifically, the data extends up to the British Grand Prix, held in July 2024.

These specific raw data were chosen as the primary source for this analysis due to its extensive use in various articles and analyses over the years, making it a trusted and well-documented resource. Its popularity among the data science community has resulted in the dataset being validated and enriched with meaningful insights, ensuring a high level of reliability and consistency. Furthermore, the dataset's comprehensive nature, covering numerous aspects of Formula 1 racing history, provided a robust foundation for creating a highly detailed working dataset. Its structure and availability of key relational variables also made it well-suited for merging and analysis, aligning perfectly with the objectives of this study.

More specifically the files that were useful to the construction of the working dataset where:

- drivers.csv (Driver Information)
- constructors.csv (Constructor Information)
- results.csv (Race Results)
- races.csv (Race Information)
- pit_stops.csv (Pit Stop Details)
- qualifying.csv (Qualifying Session Results)
- circuits.csv (Circuit Information)
- status.csv (Race Status Indicators)

From all the above combined with a relevant code, a raw merged dataset was constructed to facilitate a better understanding of the cleaning work needed before the creation of the final working dataset. The only CSV file that required preprocessing before the data merging was the **Pit Stops** archive, due to its structure. This file contained multiple lines for the same race, with each line corresponding to a separate pit stop and its associated time. To make the data suitable for analysis, the file was aggregated by summing the total number of pit stops for each driver in each race and calculating the total pit stop duration. The merging process was carried out using the key variables (driver ID, constructor ID, circuit ID, and race ID) that were present in each csv Excel file. By selecting specific variables from the entire raw csv files, the initial raw dataset included the following variables, (some of which got renamed for clarity):

Title	Final Title	Derived From CSV
driverId	driverId	drivers.csv
forename	driver_forename	drivers.csv
surname	driver_surname	drivers.csv
nationality	driver_nationality	drivers.csv
dob	dob	drivers.csv
constructorId	constructorId	constructors.csv
name	constructor_name	constructors.csv
nationality	constructor_nationality	constructors.csv
raceId	raceId	races.csv
date	race_date	races.csv
circuitId	circuitId	circuits.csv
name	circuit_name	circuits.csv
year	year	races.csv
round	round	races.csv
position	qualifying_position	qualifying.csv
q1	q1	qualifying.csv
q2	q2	qualifying.csv
q3	q3	qualifying.csv
grid	grid_position	results.csv
points	driver_points	results.csv
fastestLapTime	fastest_lap_time	results.csv
statusId	race_status	results.csv
laps	laps	results.csv
positionOrder	positionOrder	results.csv
total_pit_stops	total_pit_stops	pit_stops.csv
total_duration_ms	total_duration_ms	pit_stops.csv

Table 3.1: Variables of the Raw merged Dataset

With the above preprocessing completed (index for variables will be given in the enumeration of the working dataset’s variables), the next step involved merging the driver's names and forenames into a single column named driver_name for a further simplification of the dataset structure and in the same time sorting it by the race_date.

Continuing further, additional columns were created to support the upcoming analysis in this thesis. The following new columns were designed to provide deeper insights and enable more sophisticated exploration of the data:

Title	Index
age	**Each driver's age calculated from the dob variable.**
years_experience	**Each driver's experience in years calculated from the first race that he appeared.**
race_starts	**Cumulative driver's race starts calculated by the number of appearances.**
cumulative_season_driver_points	**Cumulative driver's points in each season up to the certain race.**
years_with_constructor	**Continuous years of working with the current constructor calculated by the combined appearances.**
top_constructor	**Constructor with the most years of working together with the driver derived from the highest number of the previous variable.**
top_3_positions	**Driver's cumulative times finishing in podium calculated by the corresponding result each year.**
positions_4_10	**Driver's cumulative times finishing in top 10 but out of podium calculated by the corresponding result each year.**
outside_10_positions	**Driver's cumulative times finishing out of points (current point system) by the corresponding result each year.**
race_outcome *(Podium finish, Points without podium, No points)*	**Nominal variable useful for classification etc., calculated by each race result.**

Table 3.2: Added Variables to the Raw merged Dataset

With the upper additions completed and given the plan of this thesis, to expand the scope of analysis by including additional variables, certain adjustments were necessary. Upon reviewing the dataset, it became apparent that a substantial amount of missing data was present. The verdict was that from 2011 onwards, the data for all qualifying sessions was more consistent and detailed records of pit stops began to be systematically noted. Thus, through appropriate filtering, a working dataset was compiled, covering the period from 2011 to the present.

Before finalizing the working dataset, some further adjustments were necessary to address, regarding missing data and logical NaN values. To begin with, the fastest_lap_time variable both contained NaN and entries represented as "\N" probably due to the way Ergast API (http://ergast.com/mrd/) compiles the data, thus with the relevant code, they were changed to NaN for alignment purposes.

Subsequently, missing values in the variables “qualifying_position”, “fastest_lap_time”, “total_pit_stops”, and “total_duration_ms” were analyzed. Upon inspection these NaN values were associated with various cases of “Did Not Finish” DNF or “Did Not Start” DNS, due to a variety of reasons, as the information in race_status variable indicated. Since these cases were not critical to the analysis, 445 NaN entries were removed, resulting in a cleaned dataset comprising 5,295 rows.

For the variables “q1” , “q2” , and “q3”, in cases where the drivers did not progress to the Q2 or Q3, the entry was filled with “\N”. To ensure the dataset was suitable for analysis, the “\N” values were replaced with “0:00.000” indicating that no time was set. To address potential misinterpretation of the “0:00.000” values as valid times, the additional binary variables “valid_time_set_for_q1”, “qualified_for_q2” and “qualified_for_q3” were created. These variables serve to identify whether a driver set a valid time in Q1 or advanced to Q2 and Q3, preventing any misclassification of “0:00.000” as the best time in each session.

Additionally, the variables “fastest_lap_time”, “q1” , “q2” , and “q3” were converted from the *X:XX.XXX* format to seconds as needed for further analysis. Regarding the “race_status” variable, which contained numerous distinct categories, a logical grouping was performed based on the nature of the outcomes. The categories were consolidated as follows:

- 'FINISHED': Completed the race without complications.
- 'DSQ': Disqualified (e.g., non-compliance with FIA regulations).
- 'RETIRED': Retired from the race.
- 'WITHDREW': Withdrew before or during the race.
- 'DRIVER RELATED': Driver-specific issues (e.g., illness).
- 'LAPPED': Lapped by leading cars during the race.
- 'RACE INCIDENT': Complications arising during the race (e.g., collisions).
- 'CAR RELATED': Issues related to the car (e.g., engine failure).

With all the abovementioned completed, a working dataset of 32 variables and 5295 rows is compiled and simultaneously checked for NaN or \N entries. The id, “dob” and “race_date” variables were not transferred from the raw dataset to the working one due to no further usage. By rearranging the variables, the working dataset comprises the following variables:

Title	Index
driver_name	**Each driver’s full name.**
driver_nationality	**Each driver’s nationality.**
age	**Each driver’s age calculated from the dob variable.**
years_experience	**Each driver’s experience in years calculated from the first race that he appeared.**
race_starts	**Cumulative driver’s race starts calculated by the number of appearances.**
constructor_name	**Each Constructor’s name.**
constructor_nationality	**Each Constructor’s nationality.**

years_with_constructor	**Continuous years of working with the current constructor calculated by the combined appearances.**
top_constructor	**Constructor with the most years of working together with the driver derived from the previous variable.**
year	**Year of the related race.**
round	**Round of the Grand Prix.**
circuit_name	**Name of the Circuit.**
valid_time_set_in_q1	**1 – Driver managed to set valid time in Q1** **0 – Driver didn't set a valid time in Q1**
q1	**Time set in Q1.**
qualified_for_q2	**1 – Driver proceeded in Q2** **0 – Driver didn't proceeded in Q2**
q2	**Time set in Q2.**
qualified_for_q3	**1 – Driver proceeded in Q3** **0 – Driver didn't proceeded in Q3**
q3	**Time set in Q3.**
qualifying_position	**Position of driver from Qualifying.**
grid_position	**Position of driver at the start of the race.**
positionOrder	**Race result of the driver.**
race_status *(FINISHED, DSQ ,RETIRED, LAPPED, WITHDREW, DRIVER RELATED, RACE INCIDENT, CAR RELATED)*	**Additional Info of each driver's race.**
driver_points	**Driver's points after the race.**
cumulative_season_driver_points	**Cumulative driver's points in each season up to the certain race.**
fastest_lap_time	**Fastest in-race lap time for each driver.**
laps	**Completed laps for each driver.**
total_pit_stops	**Total number of pit stops in each race.**
total_duration_ms	**Duration of the above. (Penalties included)**
top_3_positions	**Driver's cumulative times finishing in podium calculated from corresponding result per year.**
positions_4_10	**Driver's cumulative times finishing in top 10 but out of podium calculated by the corresponding result each year.**
outside_10_positions	**Driver's cumulative times finishing out of points (current point system) by the corresponding result each year.**
race_outcome *(Podium finish, Points without podium, No points)*	**Nominal variable useful for classification etc, calculated by each race result.**

Table 3.3: Variables of the Final Working Dataset

3.2 Characteristics of Variables

To begin with, providing a relevant describe code in Python lead to the following descriptive metrics regarding the Min (Minimum price), 1st Q (First quarter of data), Median, Mean, 3rd Q (Third quarter of data), Max (Maximum price) for the numeric variables of the working dataset as follows:

Variable	Min	1st Q	Median	Mean	3rd Q	Max
age	18	24	27	27,959	31	43
years_experience	0	2	5	6,035	9	23
race_starts	1	36	89	112,124	174	392
years_with_constructor	0	0	1	1,718	3	11
year	2011	2014	2017	2017,213	2021	2024
round	1	5	10	10,452	15	22
q1	0	78,3355	88,476	87,954	97,268	141,611
q2	0	0	80,916	64,494	93,1425	132,47
q3	0	0	0	40,722	86,1445	129,776
qualifying_position	1	6	11	10,898	16	24
grid_position	0	5	11	10,690	16	24
positionOrder	1	5	10	10,372	15	24
driver_points	0	0	1	5,215	10	50
cumulative_season_driver_points	0	2	20	51,274	68	530
fastest_lap_time	55,404	81,2025	91,684	91,349	100,6165	176,181
laps	2	52	56	57,056	66	87
total_pit_stops	1	1	2	2,061	3	7
total_duration_ms	17434	28598,5	47014	168549,162	67144,5	3703013
top_3_positions	0	0	3	24,195	33	197
positions_4_10	0	5	29	38,440	63	151
outside_10_positions	0	18	34	39,236	56,5	136

Table 3.4: Descriptive Metrics of the Numeric Variables of the Working Dataset

Table 3.4 metrics table easily one can see the mean and median of age are close to 27 and driver ages span to 43 years. Driver experience, with a maximum of 23 years, has an average of 6 years, indicating substantial variation. The number of race starts also reflects significant variability, with a mean of 112 races, ranging from just 1 to a maximum of 392. Drivers typically remain with the same constructor approximately 2 years as the 3rd Q shows 3 and maxes to 11 mainly as an outlier. The dataset spans from 2011 to 2024, with approximately 20 rounds per season. Lap times for Q1, Q2, and Q3 progressively decrease, as reflected in their 3rd quartile and maximum values. Qualifying, grid positions and results (positionOrder) exhibit similar ranges, with slight decreases in their mean values. Driver points per race average 5, with a maximum of 50 due to the one-time doubling of points test in the final race of the 2014 season in Abu Dhabi. Cumulative season driver points show a mean of 51, but the maximum value highlights Max Verstappen's record-breaking last season. Fastest in-race average 91 milliseconds, fairly slower even from Q1 times due to race variables (tire wear, fuel management etc). Laps with an average of 57 show their differences amongst the circuits, so as the different pit stop strategies with a mean of 2 per race. The vast variance shown by pit stop duration can be attributed to the various in-race incidents (front wing change, execution of time penalties etc) with a mean of

17.4, from the time the car entries the pit lane, till the time of the exit. Cumulative driver positions show a decrease in their mean values from podiums to out of points finishes, but also a tendency of keeping the front runners in the grid for longer periods as the maximum prices suppose.

To describe the binary variables the following table was created to focus on the Means of each one:

Variable	Mean
valid_time_set_in_q1	0,990
qualified_for_q2	0,734
qualified_for_q3	0,467

Table 3.5: Descriptive Metrics of the Binary Variables of the Working Dataset

As shown in Table 3.5, it is evident that with a mean of 0.99, there is a high likelihood that drivers set a valid Q1 time. However, the mean values for qualifying through Q2 and Q3 decrease to 0.74 and 0.48, respectively. This trend indicates a reduction in qualification probabilities as drivers advance through the stages.

Moving towards the data visualization for the numerical values, an ideal statistical tool that illustrates the values over time are the Time Series plots. They display data values as they change across time and can be particularly useful as they can illustrate data trends, reveal potential patterns in the values, and serve as a tool for assessing the randomness of the points. Since the 2024 season is still ongoing at the time of writing, summed variables should be out of visual focus. Additionally, to provide a clearer view, zeros from grid positions and qualifying times were omitted from the averages presented below (explained 3.1 paragraph) and as so the following plots were created:

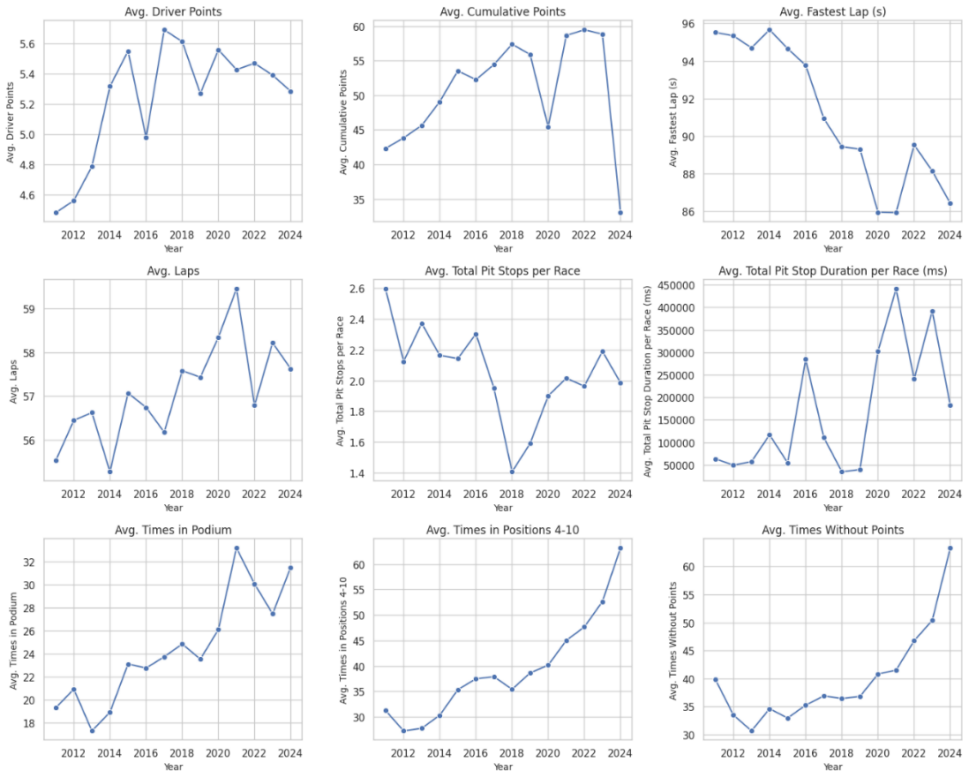

Figure 3.1: Time Series plots of variable's mean values (1)

As observed, the average driver points per race appear to increase from 2011 to 2018 before stabilizing to around 5.4 per race. This trend was influenced by the changes of the grid size gradually declining to 20 drivers sharing 101 points per race, with potential retirements also playing a role in the fluctuations. Average cumulative points upward trend only shows a noticeable dip during the pandemic-affected season with a reduced number of Grands Prix. The decreasing fastest lap times over the years reflect the sport's evolution and technological advancements. As seen in the Figure 2.4 circuits change by the years but Figure 2.2 and the wide gather of data are keeping the decreasing of lap times unbiased. Additionally, variables avg. pit stops and pit stop duration, show some relation as well as in the more noticeable avg. finishing positions. Pit stops show a decreasing trend until 2018, while finishing positions demonstrate a steady upward trend, further illustrating changes in racing strategies and driver experience.

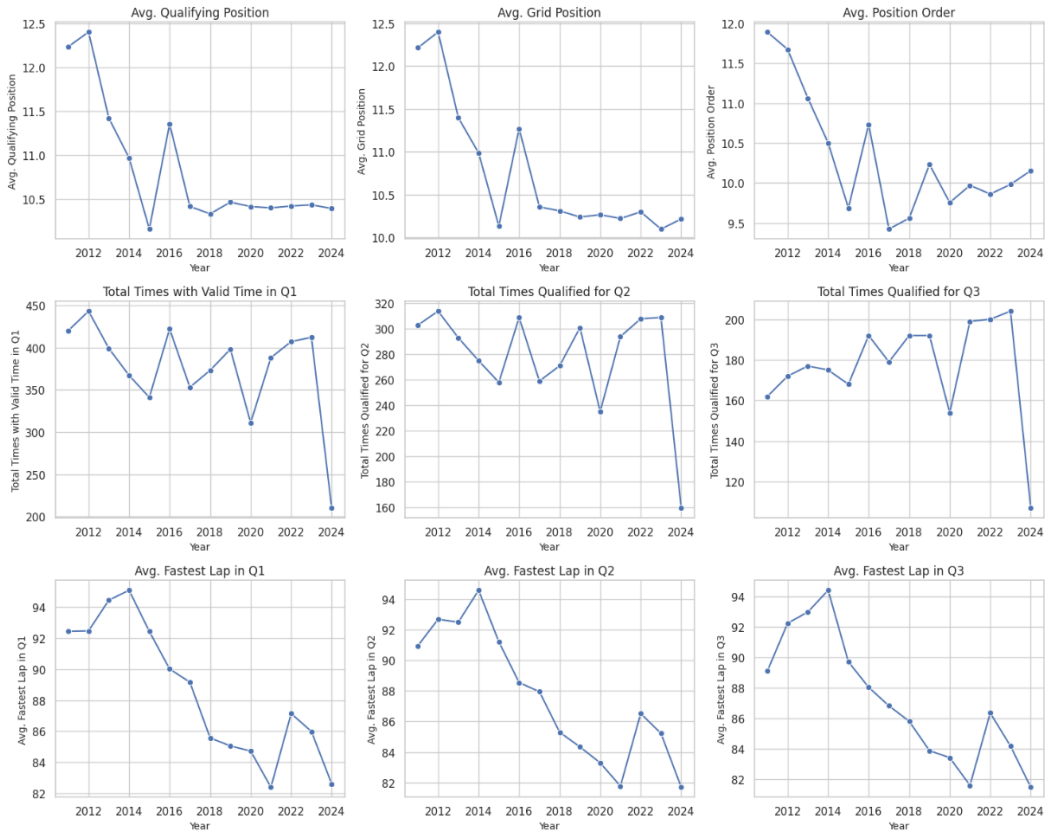

Figure 3.2: Time Series plots of variable's mean values (2)

Figure 3.2 also can visually highlight high relations per 3 variables. Specifically, average qualifying, grid position and final position order show a declining trend from 2011, stabilizing after 2015 mainly due to the number of drivers per season. Average final position order shows a slight upward trend indicating increasing performance variability. Total Times with Valid Time in Q1, Qualified for Q2, and Qualified for Q3 display variability but generally increases over time, peaking in the later years, reflecting fewer failures or disqualifications in the sessions. The pandemic season, with fewer races, creates noticeable drops in these totals. Average lap times for all qualifying sessions display a consistent downward trend, underscoring advancements in car performance, track conditions, and strategy optimization over the years.

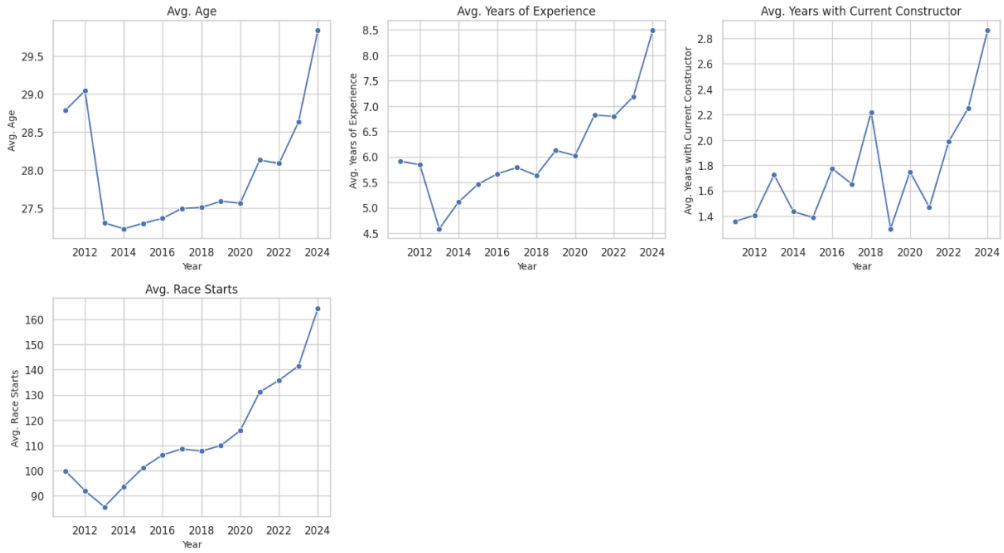

Figure 3.3: Time Series plots of variable's mean values (3)

Figure 3.3 provide insights to key driver-related variables over time, thus one can clearly see correlation in-between them. The average age of drivers exhibits a clear upward trend, particularly in recent years that can also reflect a reduction in the influx of younger drivers entering the sport. Average years of experience have been steadily increasing since 2014, with a sharp rise in recent years indicating a growing emphasis on retaining experienced drivers. The average tenure with each driver's current constructor (at the time of each race) shows greater variability but a general upward trend that may point an increasing focus on driver-team continuity. Race starts has also been steadily rising, reflecting the accumulation of experience amongst the grid.

Continuing with a visualization for the categorical variables, a statistical tool to represent their distribution is the Bar Plot. Bar plots effectively display the frequency or proportion of categories within a variable, marking them as ideal for identifying patterns, highlighting dominant categories, and comparing distributions across different groups or time periods. These visualizations allow for an intuitive understanding of categorical data making them valuable tool for analyzing trends or shifts and with a relevant code from Python the following figures were constructed:

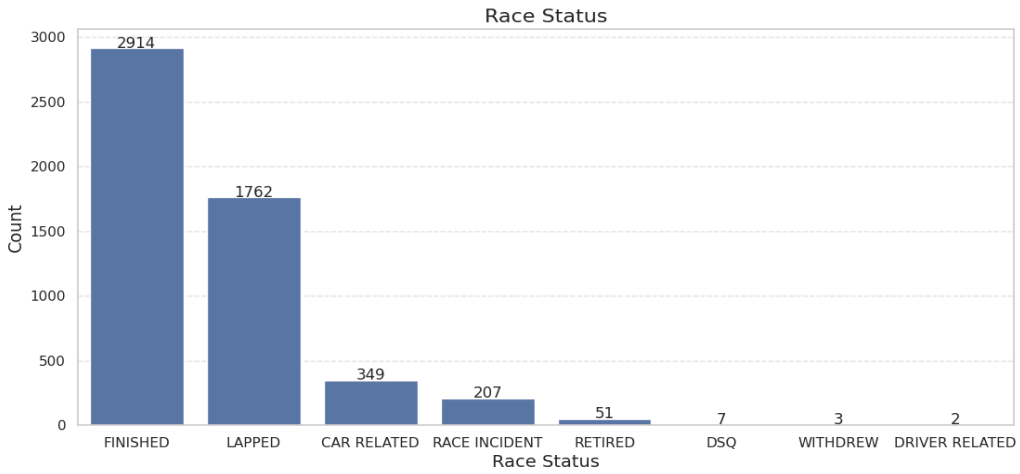

Figure 3.4: Race Status Variable Bar Plot

Figure 3.4 represents the grouped variable of the many distinct categories that once had (as seen in 3.1 paragraph) with a count on each outcome. The majority of drivers either finish the race (2914 counts) or are lapped (1762 counts). Outcomes such as Car Related (349 counts) and Race Incident (207 counts) are the primary reasons for DNFs “Did not Finish”, reflecting the challenges posed by mechanical failures and in-race collisions. Categories like Retired (51 counts), DSQ “Disqualified”, Withdrew and Driver Related are significantly less frequent, representing uncommon situations.

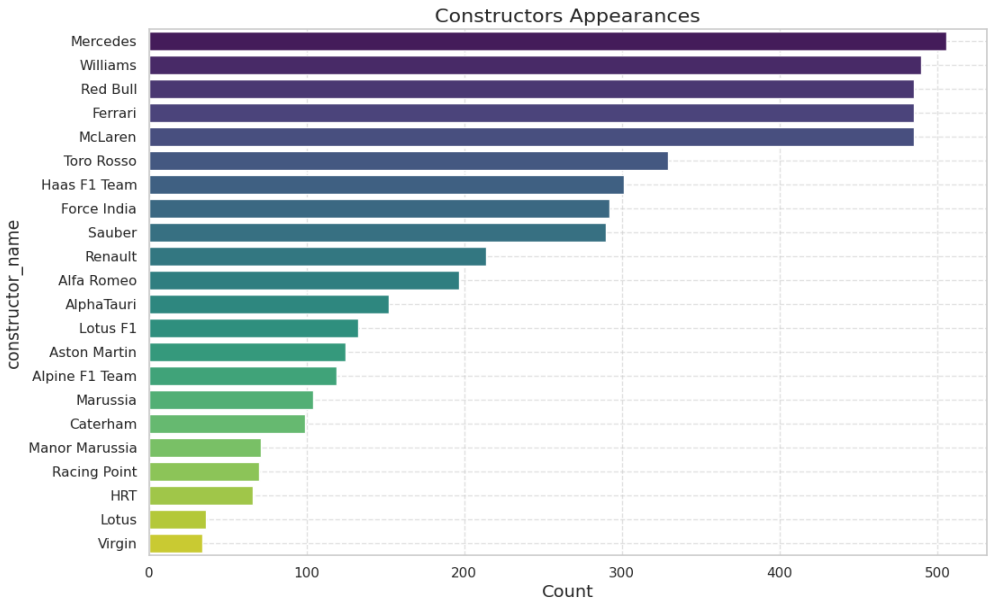

Figure 3.5: Constructor Appearances Bar Plot

Figure 3.5 presents the number of race appearances for each Formula 1 constructor from 2011 onward. Mercedes leads with nearly 500 appearances, reflecting consistent participation during this period. Other historic teams such as Williams, Red Bull, Ferrari, and McLaren also maintain high counts, showcasing their long-standing presence. Midfield teams like Toro Rosso, Haas, and Force India exhibit moderate participation, with name changes and rebranding affecting their continuity.

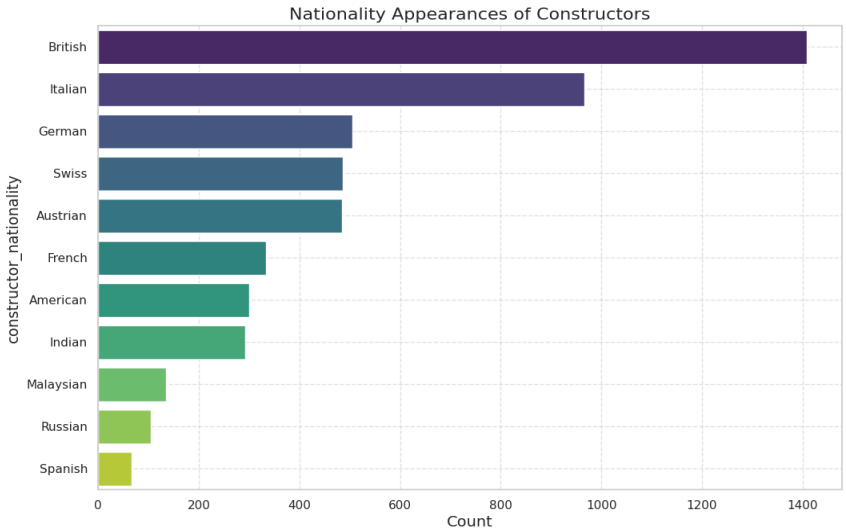

Figure 3.6: Nationality Appearances of Constructors Bar Plot

Showcasing the diversity of nationalities among constructors from 2011 to the present, Figure 3.6 highlights the global reach of Formula 1. British teams dominate the list, reflecting their long-standing heritage in the sport, with Italy following closely, representing an iconic presence. Teams from the USA, India, Malaysia, and Russia also make it into the top ten, showcasing the international appeal and participation of constructors across various regions.

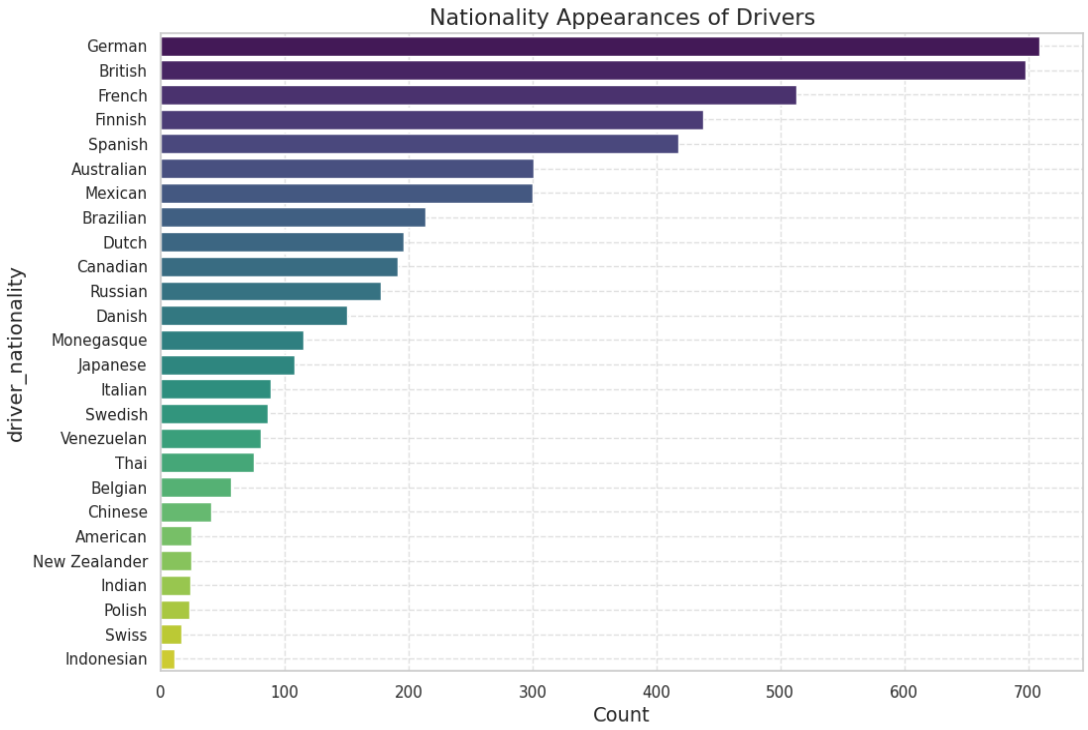

Figure 3.7: Nationality Appearances of Drivers Bar Plot

Figure 3.7 illustrates the distribution of driver nationalities in Formula 1 from 2011 to the present, revealing the global diversity within the sport. German and British drivers lead the chart, reflecting their consistent representation and success in Formula 1 during this period. They are followed by French and Finnish drivers, showcasing strong European dominance. Notably, drivers from non-European countries like Australia, Mexico, Brazil and Japan also make a significant impact, highlighting the worldwide appeal and inclusivity of the sport. Meanwhile, the lower representation of nations such as India, Indonesia, and Poland suggest emerging or less-established Formula 1 driver programs in those regions.

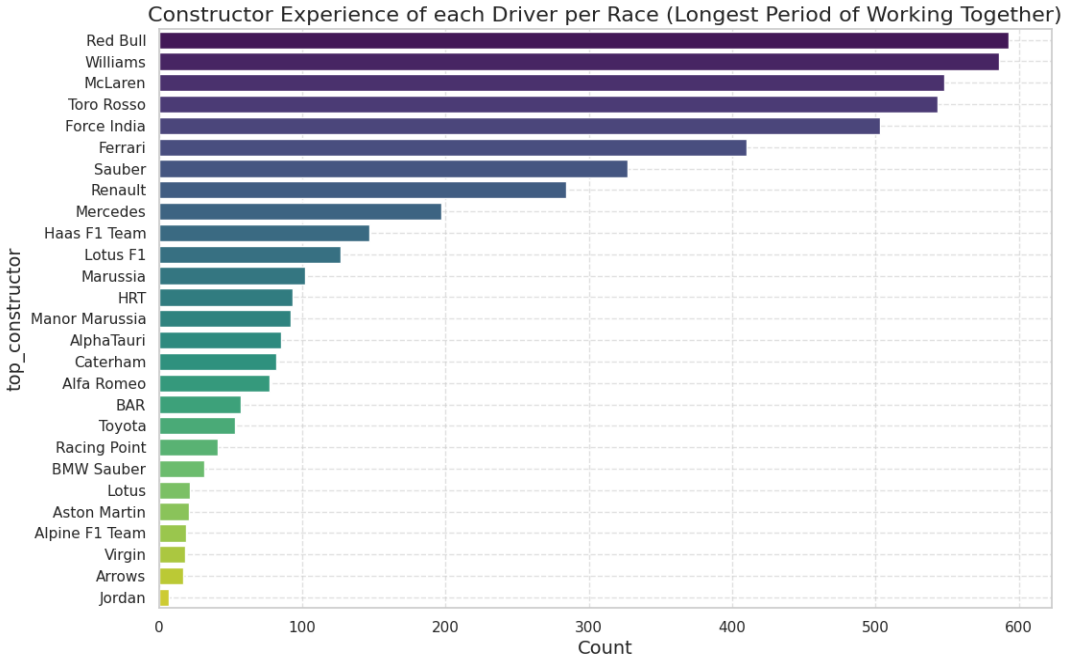

Figure 3.8: Constructor Experience of each Driver per Race Bar Plot

Figure 3.8 illustrates the constructors with which drivers have had the longest experience throughout their careers. The chart highlights teams where drivers have built the majority of their career familiarity and skill, showcasing notable constructors such as Red Bull, Williams, and McLaren as key contributors to driver development. Mid-tier teams like Toro Rosso and Force India also feature prominently, reflecting their roles in nurturing talent. On the other hand, constructors with lower counts, such as Caterham and HRT, represent teams with shorter stints in Formula 1, offering fewer opportunities for drivers to accumulate extended experience.

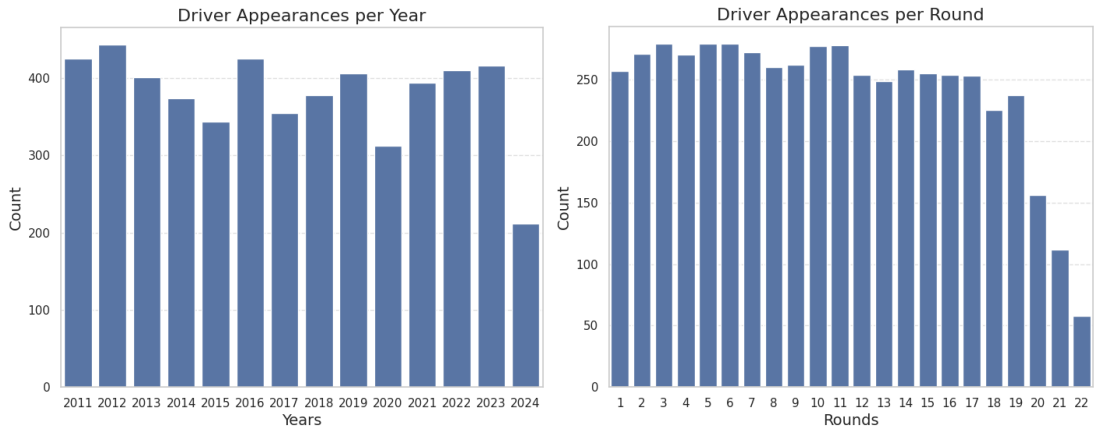

Figure 3.9: Driver Appearances per Year and per Round Bar Plots

Figure 3.9 displays driver appearances both by year and by round. The left bar plot highlights annual variations in driver participation, with noticeable dips during years like 2020 due to the pandemic-shortened season. The right bar plot breaks down appearances per race round, illustrating uniform participation across most rounds but tapering off toward the final few rounds as to increasing number of rounds the latest years. (as seen in the Figure 2.4)

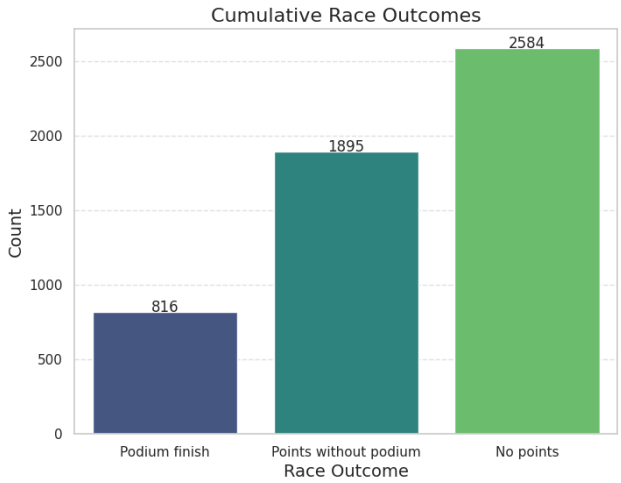

Figure 3.10: Race Outcome cumulative results Bar Plot

Figure 3.10 presents the distribution of the abovementioned categorical variable derived from race results across the dataset. The outcome aligns logically with race results, which range from 1st to 20th position (in recent years). A clear and expected pattern emerges: the number of appearances increases as the finishing position moves away from podium finishes. Specifically, there are 816 appearances in podium positions (1st to 3rd), increasing to 1,895 appearances for drivers finishing in 4th to 10th (points-scoring positions), and peaking at 2,584 appearances for drivers finishing outside the points.

To showcase the categorical variables “driver_name” and “circuit_name”, Bar plots were also implemented and can be observed at the end of the present thesis (Appendix 2: Figures). Figure A.1 illustrates the accumulated race entries of each driver from 2011 onwards. Lewis Hamilton leads with over 260 appearances followed by other well-established drivers who have surpassed 200 entries. Meanwhile, frontrunners like Max Verstappen and Charles Leclers only appear around 175 and 125 times respectively. Figure A.2 highlights the total number of appearances for each Formula 1 circuit during the same period. Silverstone Circuit tops the list, closely followed by iconic tracks such as Circuit de Barcelona-Catalunya, Hungaroring, and Bahrain International Circuit. These circuits are staples in the Formula 1 calendar, reflecting their historical significance and consistent inclusion over the years. In contrast, newer circuits like the Las Vegas Street Circuit and the Losail International Circuit show much fewer appearances due to their recent addition to the calendar.

3.3 Characteristics of Variables versus Race Results

This section examines the relationships between the variables in the working dataset and race results, offering visual insights into the factors influencing performance outcomes. Scatterplots, also known as scatter diagrams, are valuable visual tools for examining the relationships between variables. These plots depict the values of observed variables on a two-dimensional plane, allowing us to assess potential correlations and trends. By plotting each numerical variable against the race performance variable (Position Order), we can categorize their relationships into three main types:

1. **Positive Correlation:** Variables with a positive correlation exhibit an increasing relationship, where higher values of the variable on the x-axis correspond to higher values on the y-axis. For instance, fastest lap time or total pit stops may show a positive correlation with Position Order. Lower lap times or fewer pit stops generally indicate better performance, which correlates with a lower (better) finishing position.

2. **Negative Correlation:** A negative correlation occurs when an increase in one variable corresponds to a decrease in the other. For example, driver points and position order demonstrate a negative correlation, as a lower (better) finishing position is associated with a higher point result.

3. **No Correlation:** Some variables show no clear relationship with Position Order, meaning changes in one variable do not systematically correspond to changes in the other. For example, variables like age or years with constructor might fall into this category, as their impact on race results may be more nuanced or context dependent. (Figure 3.11)

By analyzing both numerical and categorical variables in relation to race results, patterns and trends emerge, showcasing the competitive dynamics influenced by various factors. The following Scatterplots, generated using Python, illustrate the interactions between numerical variables and the final position order variable:

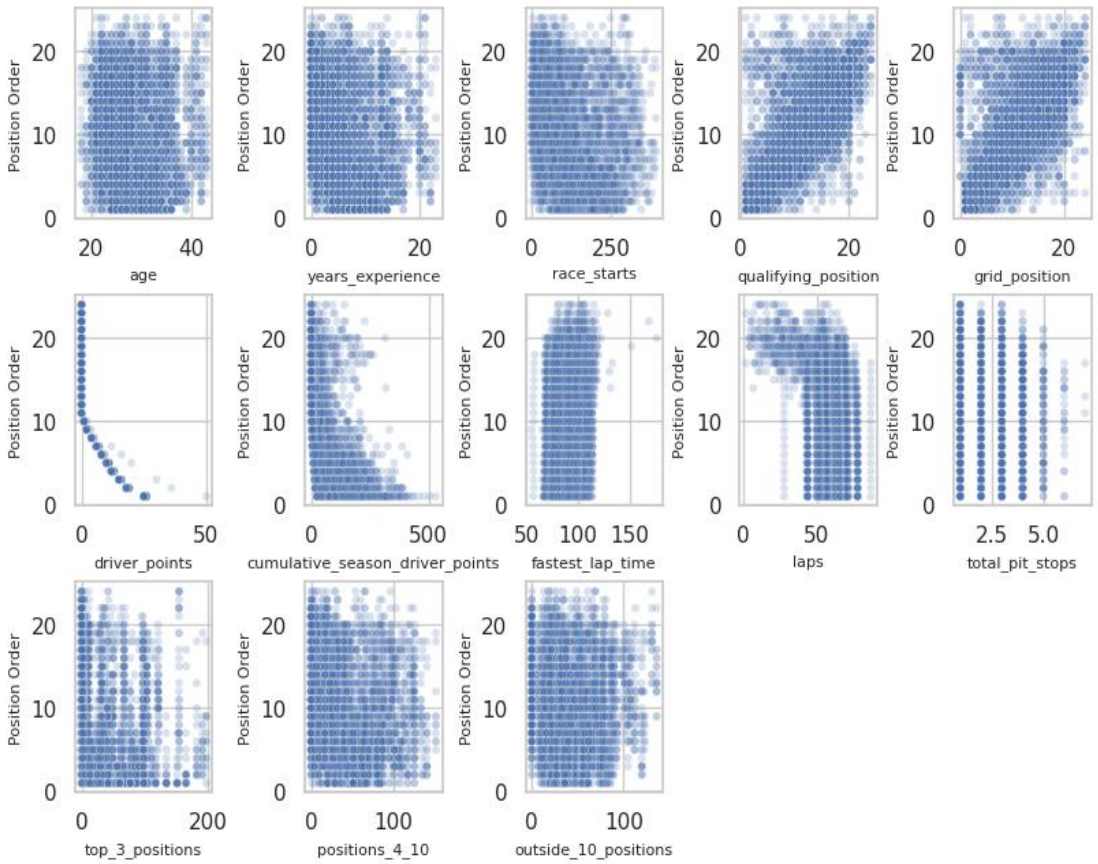

Figure 3.11: Numeric Variables vs Position Order Scatterplots

The scatter plot matrix above provides a detailed view of how different numerical variables relate to the "Position Order," a representation of drivers' final standings in each race. All non-categorical variables vs Position Order Scatterplots can be seen at the end of the present thesis (Appendix 2: Figures). Experience-related Variables show dispersed patterns, though certain trends can be discerned. For example, younger drivers and those with fewer race starts are more likely to finish in lower positions, while more experienced drivers tend to cluster towards higher positions. Outliers suggest that raw experience alone does not guarantee success, reflecting the influence of external factors such as car performance and team strategy.

As for qualifying and grid start positions, there is a strong, direct relationship between them and final position order, as seen in the tight clustering around a diagonal line in their scatter plot. Drivers starting higher on the grid are more likely to finish in better positions, underscoring the critical role of qualifying sessions in race outcomes. Driver Points and season's Cumulative Driver Points display clear inverse relationships with position order showing that a higher cumulative or individual race points tend to a higher finish, highlighting the consistency of high-performing drivers across races.

Fastest lap times and fewer pit stops are indicative of better race performance, reflected by the denser clustering of points in the upper position range. Conversely, more pit stops generally correspond to lower finishing positions. At last, the tracking variables for cumulative finishes -on the podium, in points but outside the podium and

out of points- highlight the consistent performance of frontrunners, who regularly achieve better results. In contrast, backmarkers show no clear pattern or direct correlation, reflecting the variability and challenges faced by those further down the grid.

To delve deeper into the dataset, the following Time Series Plots focus on grouping data by the race outcome variable. This approach highlights the progression of key categories, such as podium finishes, mid-field positions, and non-point-scoring results, over time. Examining these groupings uncovers trends, shifts, or consistencies in performance outcomes across seasons. These time series plots provide a dynamic perspective, aligning race results with the evolving temporal patterns of the sport.

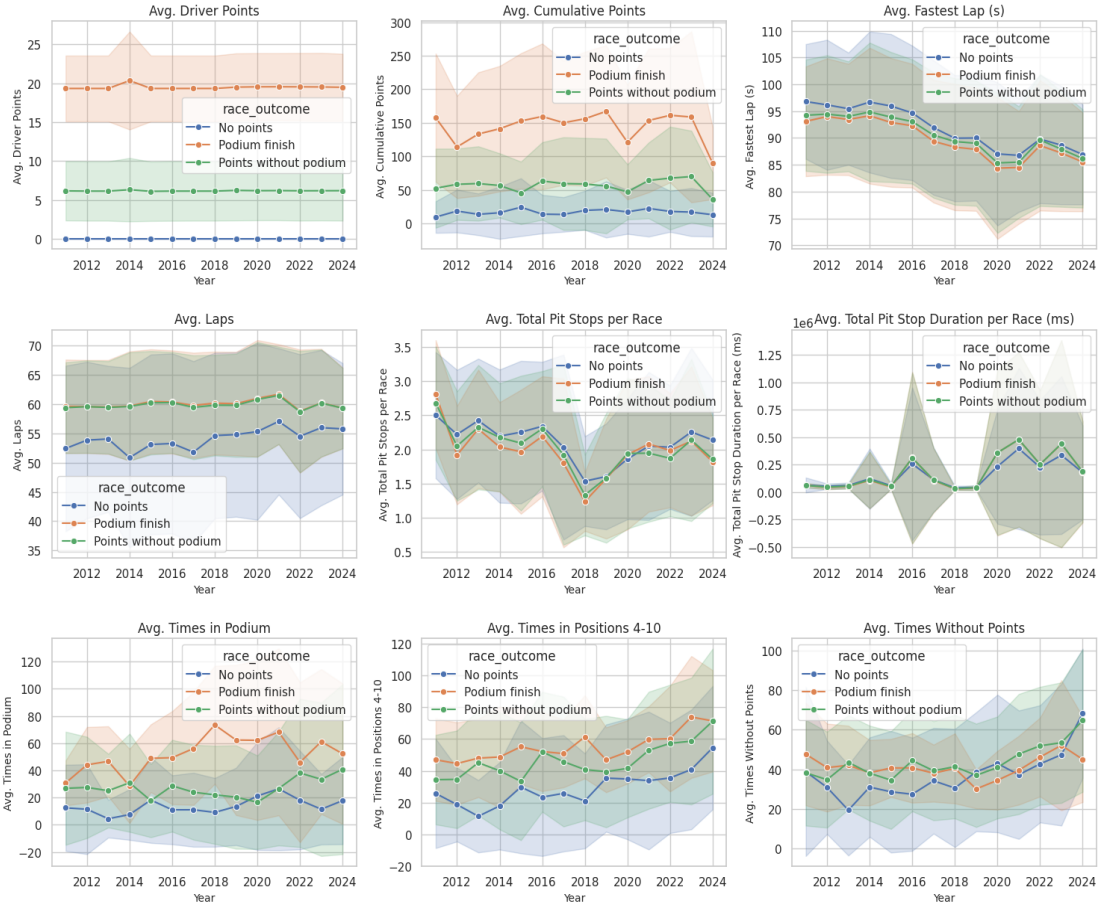

Figure 3.12: Numeric Variables vs Race Outcome Time Series plots (1)

The Time Series Plots include driver points, cumulative points, fastest laps, pit stop durations, and times spent in various positions during the races. Podium finishers consistently show superior performance with higher driver and cumulative points, faster lap times, fewer pit stops, and less pit stop durations. In contrast, Points finishes without podium exhibit moderate points accumulation, with fluctuating performance in lap times and pit stop durations. Non-point finishers generally have the lowest points totals, slower lap times, and longer pit stop durations.

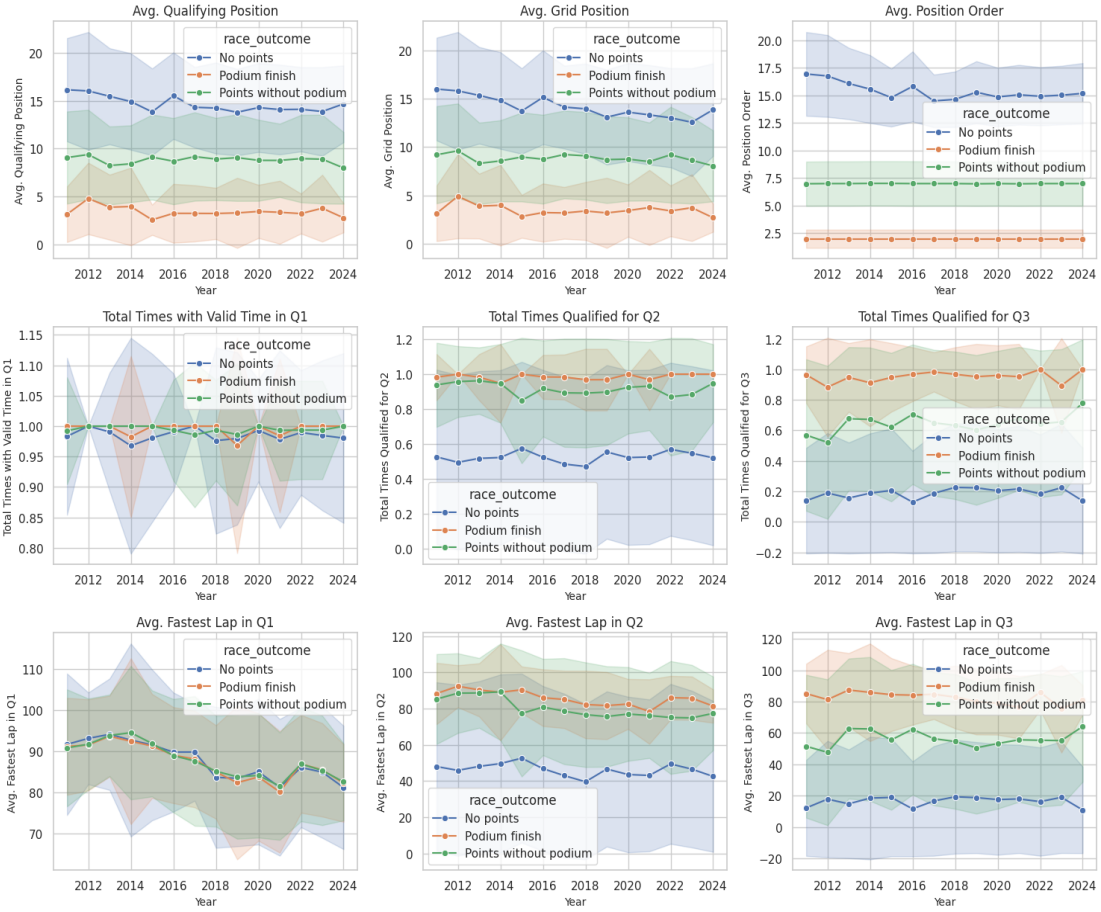

Figure 3.13: Numeric Variables vs Race Outcome Time Series plots (2)

Figure 3.13 pertains to qualifying and grid positions, average position order, fastest lap times, and progression through qualifying sessions (Q1, Q2, Q3). Distinct trends can emerge for each outcome category, like for Podium finishes consistently show superior performance, with lower average grid, faster lap times, qualifying progression and positions. Points finishes without podiums exhibit mid-range performance metrics, bridging the gap between podium finishes and non-point results. Non-point finishes indicate the most challenging outcomes, with higher grid positions and slower lap times.

Figure 3.14: Numeric Variables vs Race Outcome Time Series plots (3)

Figure 3.14 comprises key driver-related variables over time, grouped by race outcomes. Across all four variables—age, experience, continuous work with the same constructor, and race starts—a consistent trend is evident: as time progresses, drivers tend to achieve better results. This suggests that experience, stability with a constructor, and accumulated race participation positively influence performance outcomes.

To further explore the distribution of variables across race outcomes, the analysis transitions to violin plots, a sophisticated extension of box plots. Box plots are a powerful tool for summarizing data distributions by visualizing the median, quartiles, and potential outliers within a dataset. They provide a concise summary of key statistical properties, including the central tendency (median), spread (interquartile range), and extreme values (outliers). However, box plots do not reveal the underlying density or shape of the distribution, potentially obscuring nuanced patterns in the data. Violin plots address this limitation by incorporating a kernel density estimate (KDE) on each side of the box plot structure, effectively merging a box plot with a smoothed density plot. This enhancement allows for a more detailed understanding of the data, as it displays the distribution's shape and variability alongside the core summary statistics. By grouping the numeric variables by race outcome, violin plots not only capture central tendencies and spread but also reveal deeper patterns, such as skewness, multimodality, or clusters, providing richer insights into performance metrics across different categories. The violin plots occasionally display areas below zero on the y-axis, even though all data points are non-negative. This visual artifact arises because the smoothing function used to create the violin shape extrapolates slightly beyond the range of the actual data points, producing an unrealistic appearance of negative values. Addressing this issue could involve restricting the y-axis to start at zero, though doing so cuts off the violin shapes, making them appear less smooth or visually consistent.

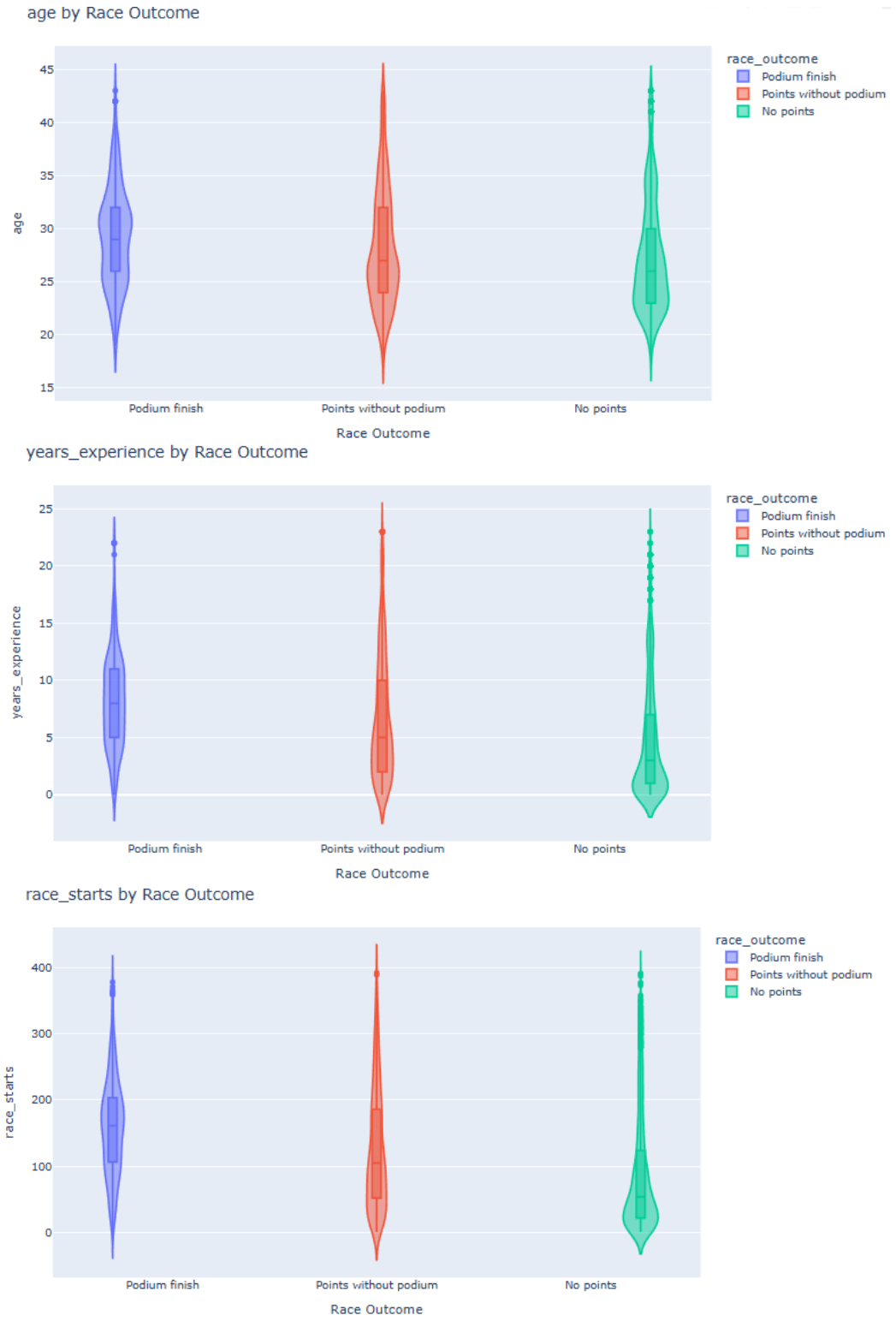

Figure 3.15: Numeric Variables vs Race Outcome Violin plots (1)

Figure 3.15 presents violin plots depicting the distribution of driver-related variables grouped by the race outcome categories. These plots illustrate a consistent trend where experienced drivers tend to perform better, frequently leading to podium finishes, with potential extreme values (e.g. Fernando Alonso) across all plots. The violin shapes highlight the density peaks within each group, aligning closely with the interquartile ranges represented by the embedded box plots. ranges.

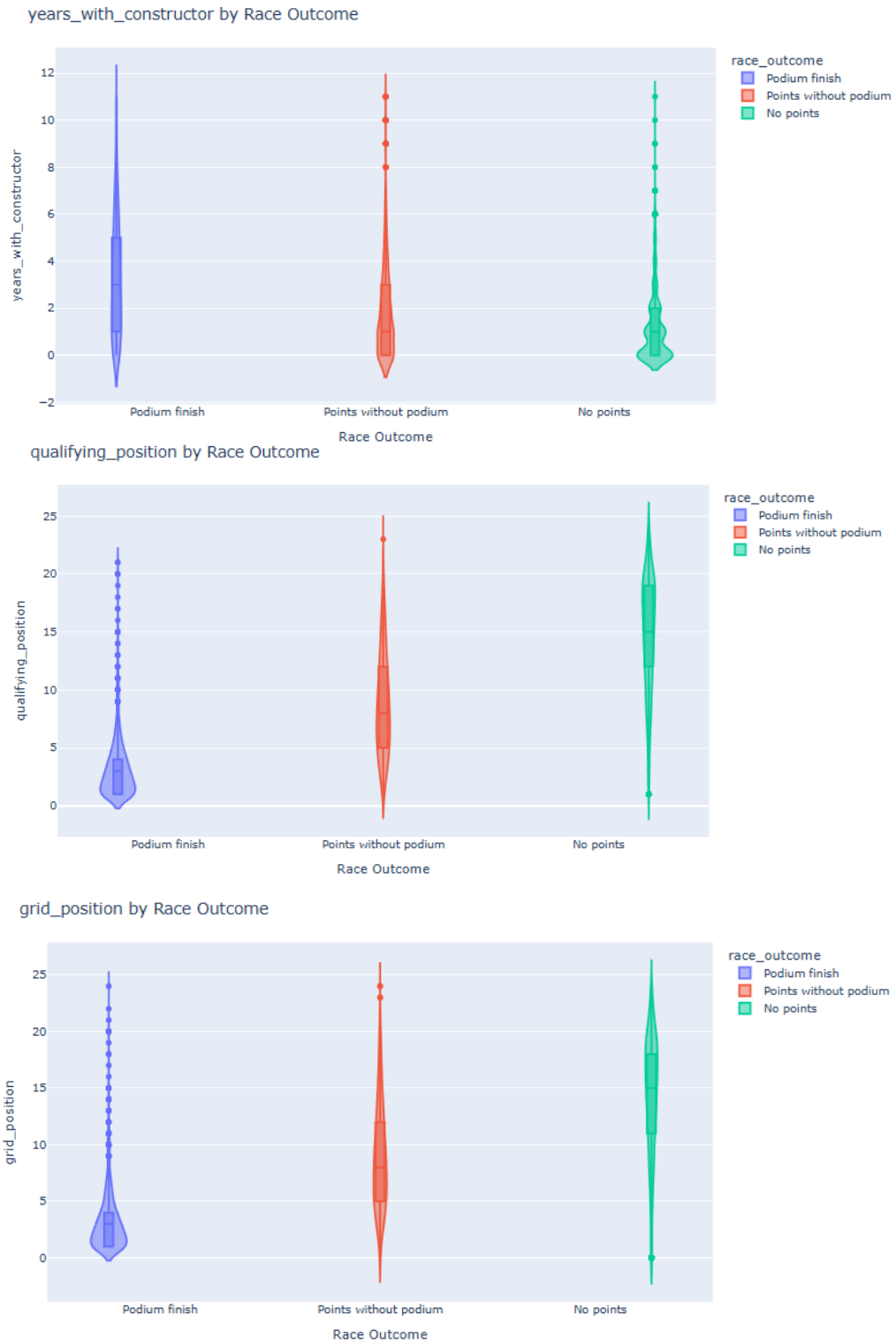

Figure 3.16: Numeric Variables vs Race Outcome Violin plots (2)

Figure 3.16 showcases violin plots for various numeric variables against the race outcome categories. The plot for years with same constructor highlights that drivers with longer tenures with a constructor tend to achieve podium finishes, as evidenced by the higher density of values on the left side. Qualifying Results and Starting Grid Position highlight a clear pattern where superior qualifying and starting positions are strongly associated with better race outcomes.

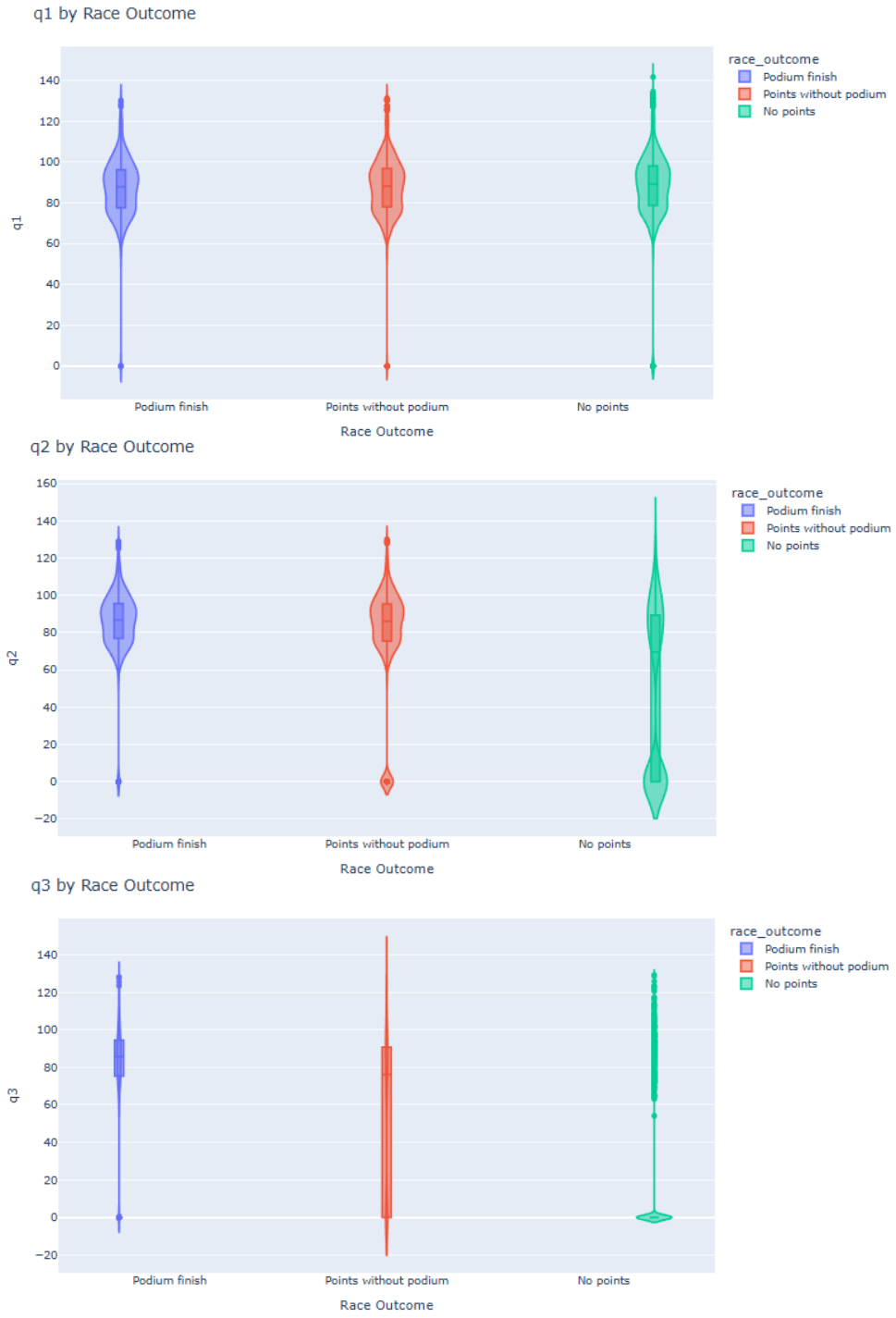

Figure 3.17: Numeric Variables vs Race Outcome Violin plots (3)

Figure 3.17 delves into the detailed analysis of qualifying times across Q1, Q2, and Q3 stages, complementing the observations from the previous figure. Q1 and Q2 times reveal minimal differentiation among the top 10 finishers, with no point finishers mainly demonstrate visibly slower times. For Q3 times, the visualization is notably influenced by non-points finishers who did not set times in this session. These zero-time entries create a pronounced visual gap, emphasizing the absence of these drivers in the final qualifying stage. Podium finishers, in contrast, consistently showcase steady, faster times, with minimal variance, underscoring their strong qualifying performance.

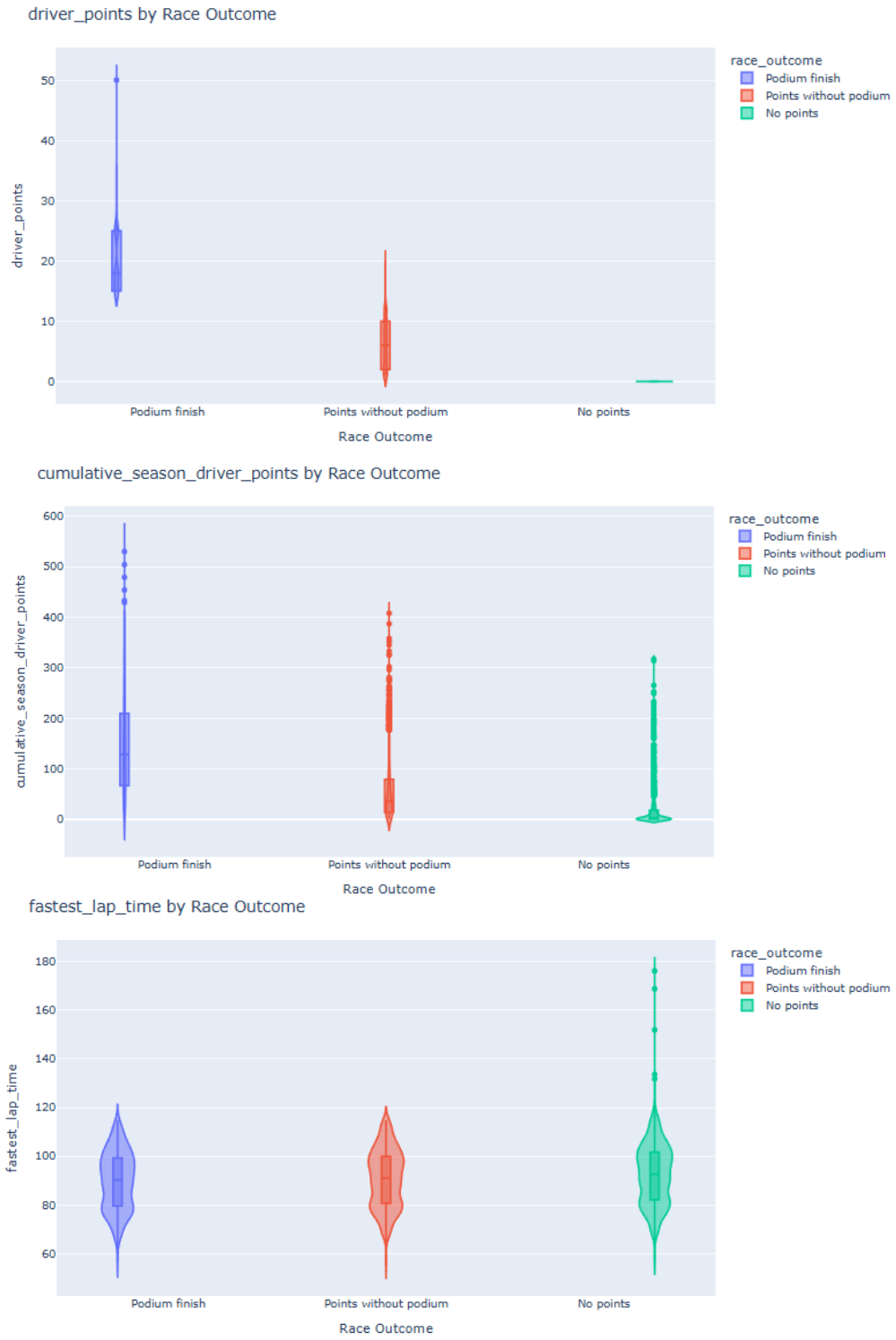

Figure 3.18: Numeric Variables vs Race Outcome Violin plots (4)

Figure 3.18 highlights the distribution of driver points and fastest lap times by race outcomes. Podium finishers show broader consistency with notable extreme values in cumulative points, while fastest laps exhibit minimal differentiation, often attributed to frontrunners but occasionally achieved by all finishers who had the chance for a last-minute tire change. Additionally, lots of potential extreme high values are seen in the non-pointers.

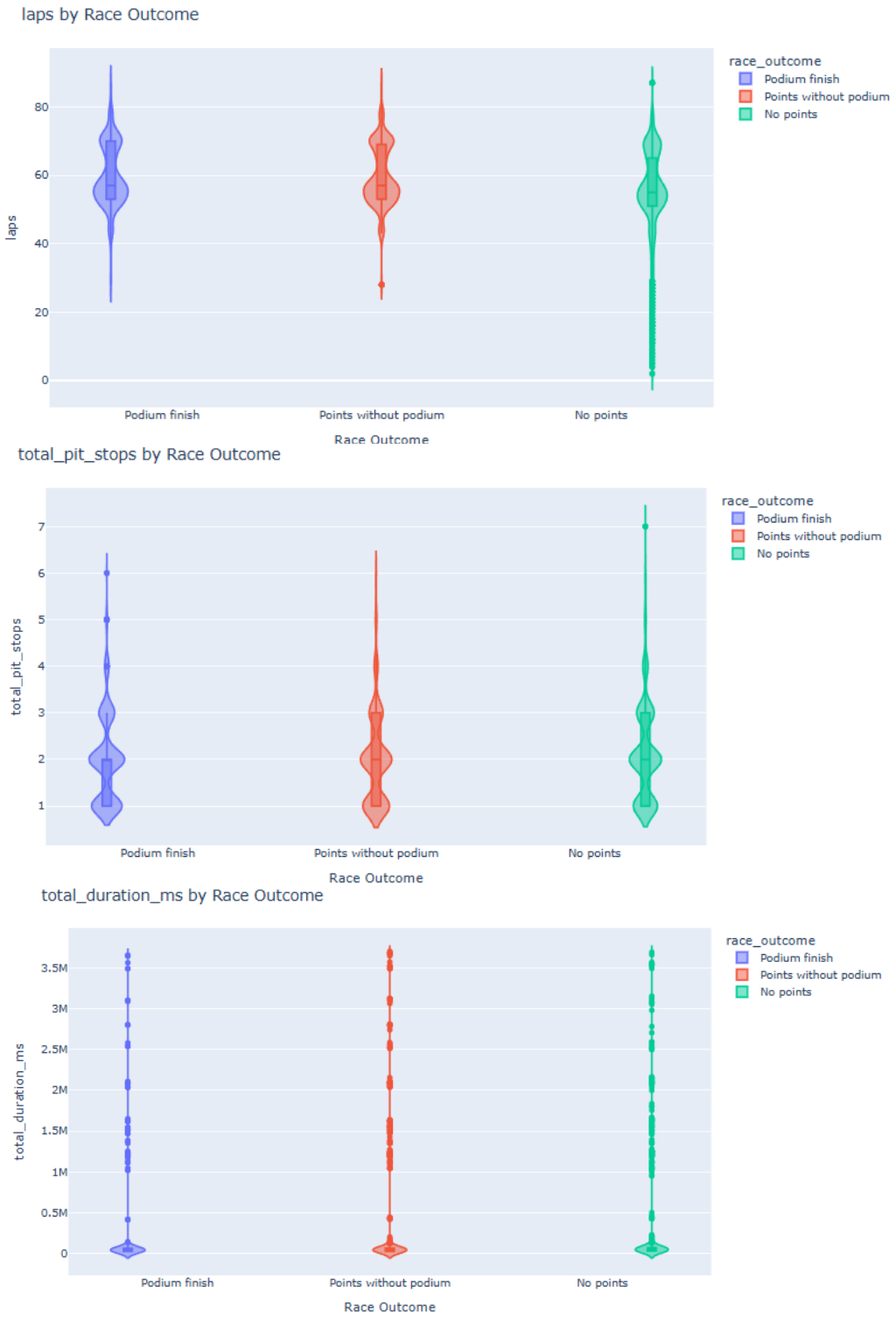

Figure 3.19: Numeric Variables vs Race Outcome Violin plots (5)

Moving towards Figure 3.19, variables such as laps, pit stops, and their durations show less differentiation when grouped by race outcome. It is observed that drivers in the top ten generally complete every lap, while fewer pit stops tend to enhance outcomes for podium finishers, as reflected in the narrower interquartile ranges.

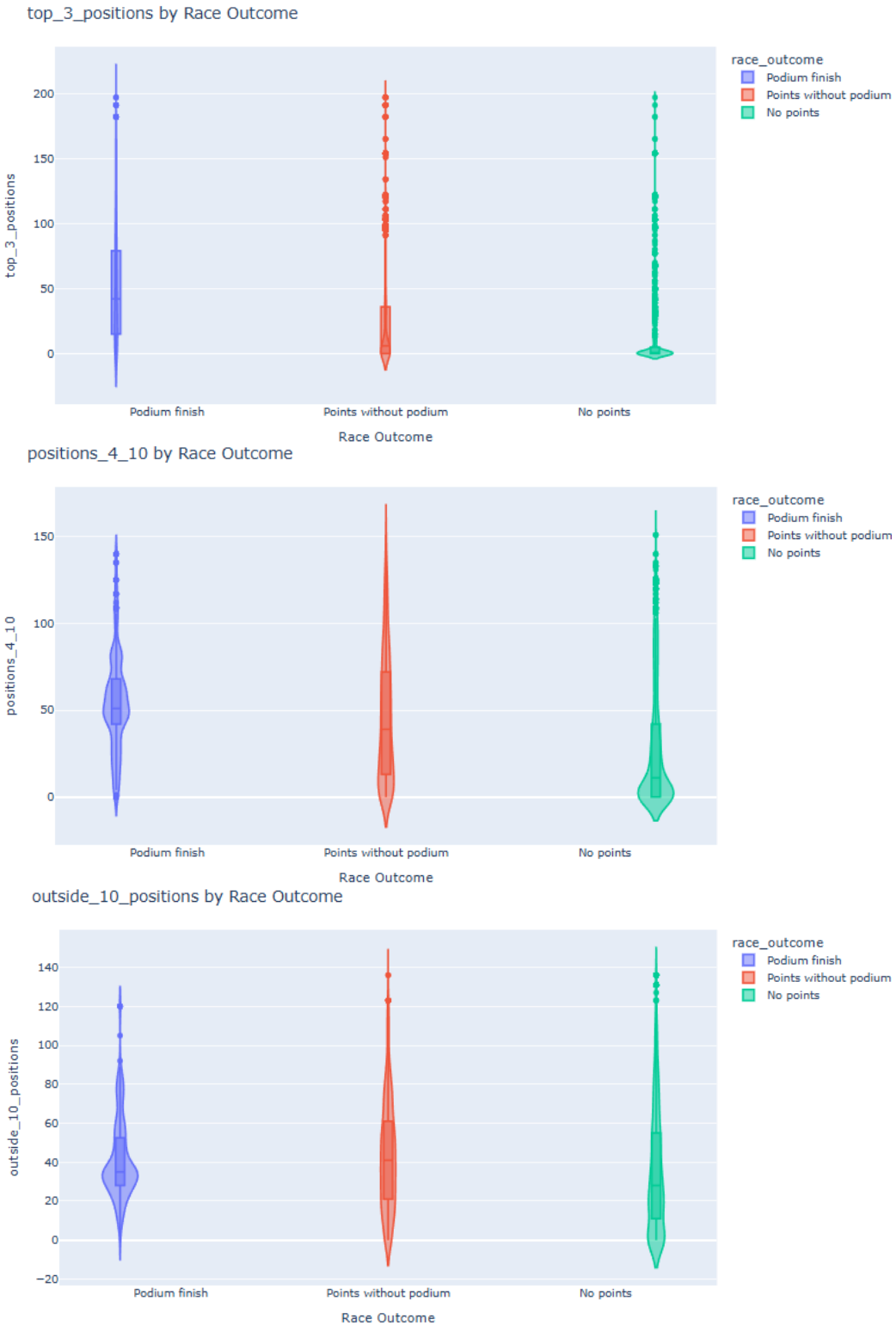

Figure 3.20: Numeric Variables vs Race Outcome Violin plots (6)

Figure 3.20 highlights the distribution of positions across race outcomes. Podium finishers dominate the top 3 positions, showing consistent results. Drivers in positions 4-10 mostly achieve points without podiums, while non-pointers dominate the outside 10 positions, reflecting their struggle for competitive results.

The remaining Violin plots for non-categorical variables versus Race Outcome are included in Appendix 2: Figures. Figure A.5 highlights key patterns: in the Q1 valid time plot, most finishers set valid times. The Q2 qualification plot shows Podium and Points finishers consistently qualifying, while non-points finishers drop significantly. Lastly, the Q3 qualification plot reveals Podium finishers dominate, Points finishers show moderate success, and non-points outcomes rarely qualify.

Figure 3.21: Categorical Variables vs Race Outcome Bar plots (1)

Figure 3.21 compares constructor and driver nationalities with race outcomes since 2011. British constructors and drivers dominate in podium finishes and points without podiums, showcasing their historical strength in Formula 1. Italian constructors excel in securing points without podium finishes, while Swiss constructors, along with German, Spanish, and French drivers, also stand out for their visibility. On the other hand, constructors and drivers from smaller or less prominent nations are more often represented in the "no points" category, highlighting their struggles in maintaining competitive performance.

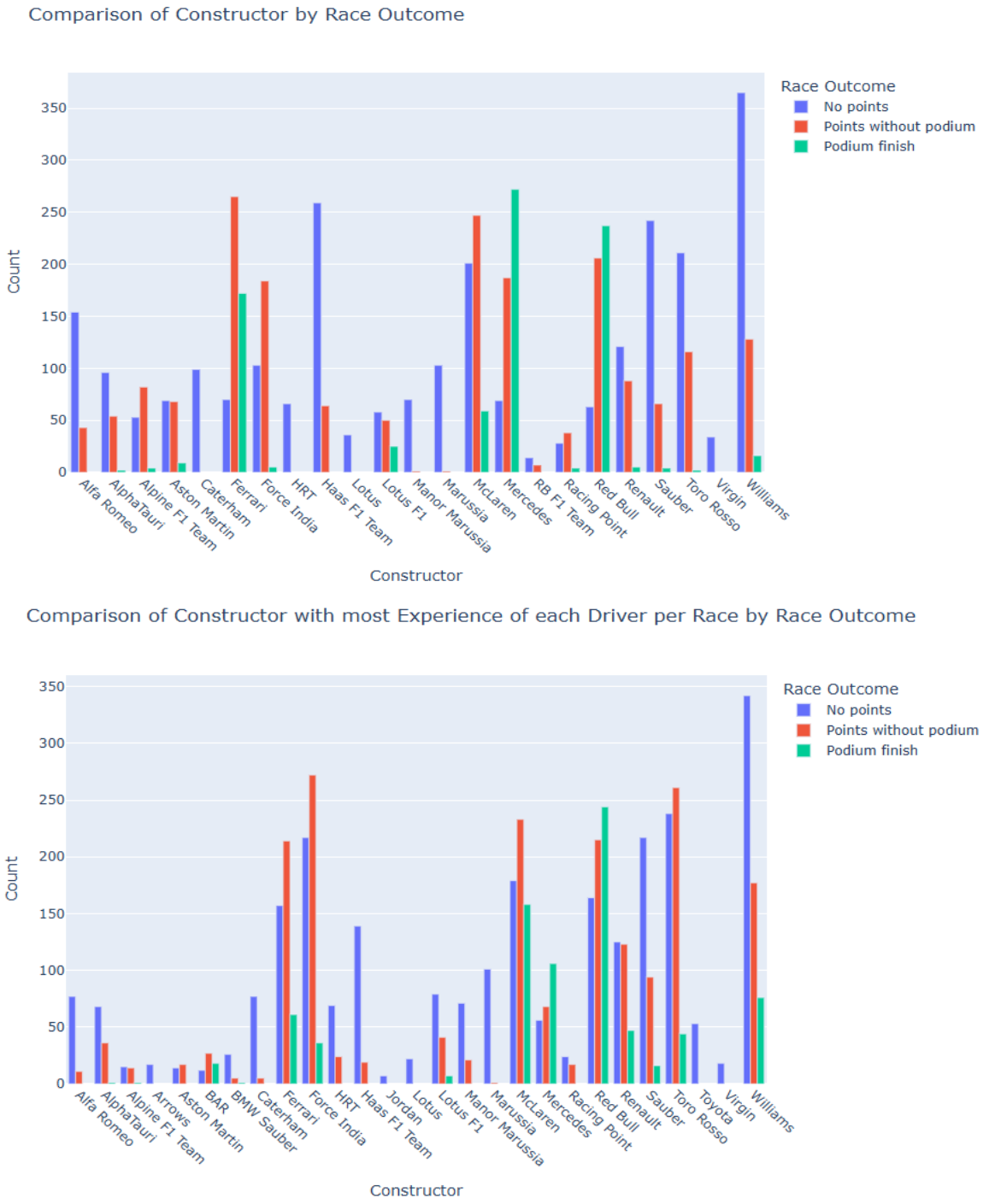

Figure 3.22: Categorical Variables vs Race Outcome Bar plots (2)

Figure 3.22 showcase constructors based on race outcomes. Leading teams like Ferrari, Mercedes, and Red Bull demonstrate a strong link to podium finishes and point-scoring positions, emphasizing their competitive reliability. In contrast, smaller teams like HRT and Manor are predominantly featured in the "no points" category, reflecting limited resources and weaker performance. The second plot highlights constructors with whom drivers have the most career experience, showing that experience with top teams like Red Bull and McLaren often aligns with better results. However, teams like Toro Rosso and Williams sometimes fall short in maintaining high-tier development programs despite driver experience.

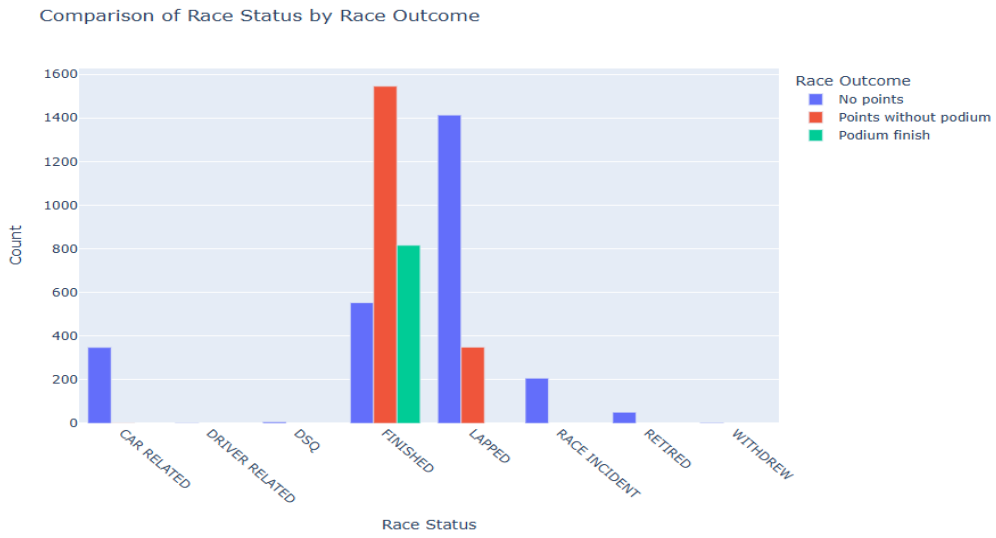

Figure 3.23: Race Status vs Race Outcome Bar plot

Figure 3.23 illustrates the distribution of the race status variable across race outcomes. Drivers who finish the race are predominantly in point-scoring positions, while those who are lapped tend to fall outside the points. Incidents, retirements, and car-related issues, even if the drivers manage to complete the race, result in non-point finishes.

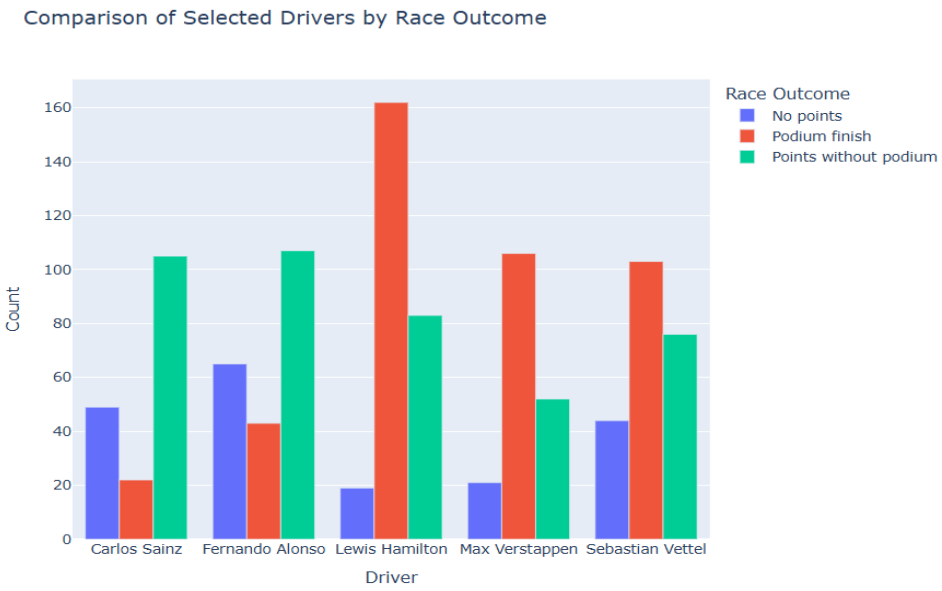

Figure 3.24: Drivers vs Race Outcome Bar Plot

Figure 3.24 compares individual drivers by their race outcomes from 2011, highlighting varying performance levels among them. Drivers like Lewis Hamilton, Sebastian Vettel, and Max Verstappen dominate the podium finishes, reflecting their consistent top-tier performances and ability to compete at the front of the grid. Meanwhile, drivers such as Sergio Perez, Carlos Sainz, and Fernando Alonso show high counts in the 'Points without Podium' category, indicating regular point finishes without frequent podium appearances. At the end of the thesis (Appendix 2: Figures), Figure A.6 illustrates the distribution across all drivers, highlighting how backmarkers and reserve drivers accumulate more 'No Points' outcomes. These figures emphasize the competitive hierarchy in Formula 1, distinguishing the elite performers from those with sporadic or limited success.

Concluding with the categorical variable of Circuit Names vs Race Outcome Bar plot, as expected, all circuits display similar distributions across race outcomes. This plot primarily mirrors the purpose of Figure A.2, providing an overview of the circuits with the highest number of appearances. The Figure A.7 can be seen at the end of the thesis (Appendix 2: Figures)

3.4 A “Dive” into Fastest Lap progression by Circuit

As discussed in Chapter 2 (Figure 2.2), the continuous technical progression of monocoques has contributed to lower average lap times, as well as the fastest lap times observed in Figure 3.1 above. While some circuits, such as Yas Marina, test maximum straight-line speed, others, like Monaco, are more technical and demand higher cornering speeds and acceleration. This raises the question of which circuit has experienced the greatest reductions in lap times. Since average lap times can be influenced by various in-race incidents, this analysis will focus on the average in-race fastest lap times. To investigate this, a time series plots of average fastest lap times has been generated using a relevant code for the circuits of most continuing appearances in the Formula 1 calendar from 2011 to 2023:

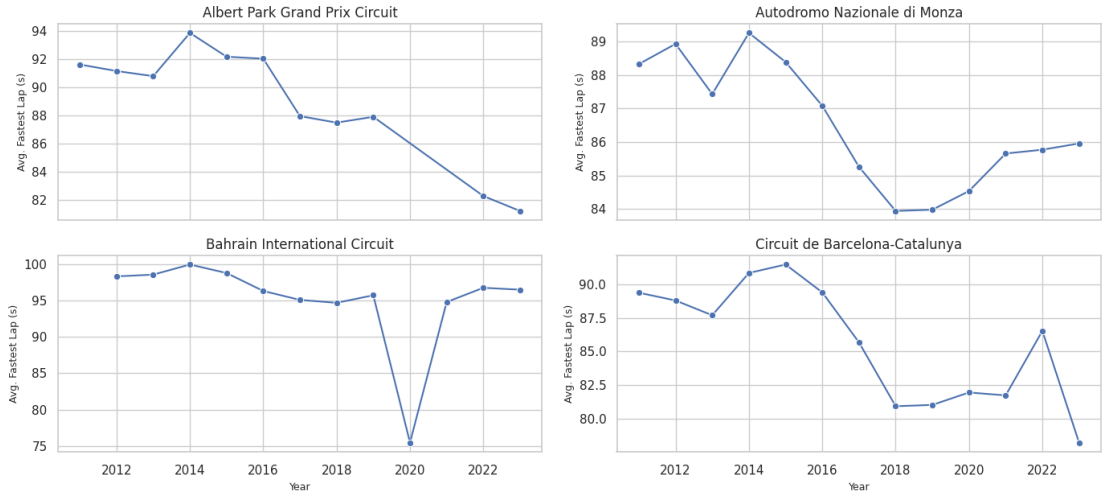

Figure 3.25: Avg. Fastest Lap Times Time Series Plot by sel. Circuits

The time series plots illustrate a clear downward trend in fastest lap times across all selected circuits, whether they are traditional racetracks or street circuits all show a severe deduction. Despite occasional outliers, the overall tendency is a significant reduction in lap times, reflecting continuous advancements in car performance and aerodynamics. Namely the most eye-catching outliers which shouldn't be considered to the conclusion are Bahrain 2020 Grand Prix as the race was conducted in a different route, Catalunya 2023 Grand Prix, with an official fastest lap attempt after straightening the last chicane (adjacent corners) and Singapore (Marina Bay) 2023 Grand Prix also having a held in a new race layout. The most substantial reductions in average fastest lap times can be observed by analyzing the vertical differences in the Y-axis (seconds) across the years. Based on the observed data, Australia Grand Prix held in Albert Park Circuit exhibits the largest reduction in average fastest lap times, with a decrease exceeding ten seconds over the analyzed period. Catalunya and Silverstone also experience respectable declines around 9 and 6 seconds respectively. Marina Bay and Monza are also showing a decent number of reductions with the rest of the circuits generally in the two-second range.

Regarding the type of circuits that show the most significant reductions in fastest lap times, one can observe that race circuits enhance the advance in technical engineering of the monocoques. Ground effect technology, along with upgrades to chassis and wings allows for greater downforce generation compared to normal road-tarmac circuits like Monaco. Additionally, a general trend of decreasing lap times can be observed from 2014 to 2020 , followed by an upward curve in recent years. These trends are a direct result of FIA regulation changes with the *Introduction of 1.6-litre Turbocharged V6 Hybrid Power Units and Cost Cap (2014–2021)* and the *Ground Effect Cars and Aerodynamic Changes (2022–2025)*, both of which were discussed in Chapter 2.

Chapter 4: Normality Tests & Correlations

In this chapter, normality tests will be conducted for the quantitative variables included in the working dataset. Assessing normality is a crucial step, as it informs the choice of subsequent statistical tests and ensures the robustness of each final finding. Additionally, correlations among these variables will be examined to identify potential relationships that influence a team's performance as a validation to Chapter's 3 statements. This exploratory analysis provides a foundational overview of the data and highlights key associations, though it is not definitive, as it lacks the precision and predictive capability of a fully developed statistical model. All analyses are conducted at a 5% significance level, ensuring that the results meet established statistical standards, executed using the Python programming language. The Python code can be found in the attached appendix at the end of this thesis (Appendix 1: Implementation Code of the Present Study).

4.1 Normality Tests

One of the most fundamental statistical hypothesis tests is the normality test, which will be examined in this section. By "normality of data," we refer to whether the sample being analyzed is drawn from a population that follows a normal distribution. This test is considered essential in statistics as its outcome determines whether parametric or non-parametric methods will be used in subsequent hypothesis testing, involving means, medians, or other statistical measures. The structure of this type of test is as follows:

- **Null Hypothesis (H_0):** The sample originates from normal distribution.

  vs

- **Alternative Hypothesis (H_1):** The distribution from which the sample originates, is not normal.

To decide between the null and alternative hypotheses in a normality test, a significance level (α) is predetermined, commonly set at 5% (0.05). If the p-value obtained from the test is less than or equal to 0.05, the null hypothesis (H_0) is rejected, indicating that the data does not follow a normal distribution. Conversely, if the p-value exceeds 0.05, we fail to reject the null hypothesis, suggesting that the data could reasonably be considered normally distributed.

There are various methods to assess normality, including graphical techniques and statistical tests. Graphical methods, such as Q-Q plots or histograms, provide intuitive visualizations of the data's distribution but may lack the precision needed for definitive conclusions. Statistical tests, such as the Shapiro-Wilk and Kolmogorov-Smirnov tests, deliver numerical results that aid in making objective decisions. These tests differ in sensitivity to sample size and applicability (Evangelaras, 2022).

Kolmogorov-Smirnov (K-S) Test

The Kolmogorov-Smirnov (K-S) test is used to evaluate the goodness-of-fit of a sample to a specified continuous distribution such as the normal distribution. It calculates the maximum difference between the empirical cumulative distribution function (ECDF) of the sample and the cumulative distribution function (CDF) of the hypothesized distribution. This maximum discrepancy is used to compute a test statistic, which is then compared to a critical value to determine whether to reject the null hypothesis.

According to the above, $H_0 : X_i \sim F_0$ is rejected when the test statistic

$$D = d_k\left(\hat{F}, F_0\right) = sup_{x\in\mathbb{R}}|\hat{F}(x) - F_0(x)|$$

takes unusually large values, that is, when $D > c$. This criterion is known as the Kolmogorov–Smirnov test (and the test statistic D is called the Kolmogorov–Smirnov statistic).

A notable advantage of the KS test is its simplicity and versatility, as it can be applied to various distributions. However, its major limitation lies in its assumption that the parameters of the distribution under the null hypothesis are fully specified in advance. When the parameters are estimated from the data, the KS test becomes less reliable because it does not account for the variability introduced by parameter estimation. This makes it less effective for real-world scenarios where the parameters of the distribution are often unknown (Boutsikas, 2004).

Shapiro-Wilk Test

The Shapiro-Wilk test is widely regarded as one of the most powerful tests for normality, particularly for small sample sizes ($n < 50$). Unlike the KS test, it is specifically designed to assess normality by comparing the empirical percentiles of the sample and the theoretical percentiles of the corresponding normal distribution (Antzoulakos, 2021). By measuring this closeness, it provides a robust indication of whether the data conforms to normality, making it a reliable choice for small datasets.

The Shapiro-Wilk test's higher sensitivity makes it a preferred choice in many practical applications. It performs well across a range of sample sizes and is robust even with moderate deviations from normality. However, for very large samples ($n > 2000$), the test may become overly sensitive to minor deviations that are not practically significant, potentially leading to the rejection of the null hypothesis for data that is reasonably close to normal.

Lilliefors Test

The Lilliefors test is a modification of the Kolmogorov-Smirnov (KS) test, designed specifically to address one of its key limitations, it being the requirement for the population parameters (mean and standard deviation) to be known in advance. The Lilliefors test adjusts for this by accounting for the variability introduced when

population parameters are estimated from the sample, thus providing a more robust and reliable test for normality in such cases. This test is particularly useful in applied statistics and research, where sample data is often used to infer properties about the underlying population without prior knowledge of its exact parameters (Antzoulakos, 2021).

Other Normality Tests

In addition to the above widely known tests, several other statistical tests are available to evaluate normality:

- **Anderson-Darling Test:** A general-purpose test that gives greater weight to the tails of the distribution, making it particularly sensitive to deviations in the extremes.
- **Shapiro-Francia Test:** Similar to the Shapiro-Wilk test but optimized for larger datasets, it is often used as an alternative for samples where $n > 50$.
- **Cramér-von Mises Test:** Focuses on the cumulative distribution function and is more sensitive to subtle deviations across the entire distribution.
- **Pearson's Chi-Square Test:** Divides data into bins and compares the observed frequencies to the expected frequencies under the normal distribution. While versatile, it requires a large sample size to ensure reliable results.

These tests complement one another, offering varied approaches to normality assessment, depending on sample size, distribution characteristics, and the specific requirements of the analysis. By each test having its own strengths and limitations, the choice of test should be guided by the data's context and the goals of the study.

Q-Q Plot

In addition to statistical tests, normality can be visually assessed through graphical methods, such as the Q-Q (Quantile-Quantile) Plot, which compares the quantiles of a dataset to the theoretical quantiles of a specified distribution (typically normal). The theoretical quantiles (X-axis) are plotted against the sample’s empirical quantiles (Y-axis). If the data follows a normal distribution, the points align closely along a 45-degree diagonal line. Deviations from this line indicate departures from normality, with curves or S-shapes suggesting skewness or kurtosis issues and outliers appearing as distant points, particularly at the tails. While less precise than statistical tests, Q-Q plots offer an intuitive visual assessment of normality and effectively highlight areas of deviation, making them a valuable complement in exploratory data analysis (Manolesou, 2015).

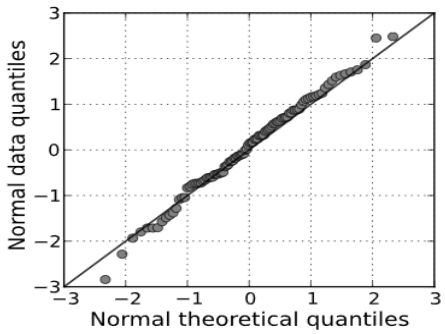

Figure 4.1: Q-Q Plot Example

(Source: https://en.wikipedia.org/wiki/Q%E2%80%93Q_plot)

Figure 4.1 compares the theoretical quantiles of a normal distribution (X-axis) with the empirical quantiles of the dataset (Y-axis). Points aligning closely with the 45-degree diagonal, indicate that the data fit a normal distribution reasonably well. However, slight deviations at the tails suggest minor departures from normality, potentially due to skewness, kurtosis, or outliers.

Histogram

Histograms are another graphical method for assessing the distribution of a dataset by visualizing the frequency distribution of data grouped into bins. When data originate from a normal distribution, the histogram takes the shape of a symmetric, bell-shaped curve. Deviations from this shape can indicate departures from normality, such as asymmetry suggesting skewness, thick or thin tails indicating kurtosis issues, multiple peaks pointing to multimodality, or isolated bars at the extremes that may signal potential outliers. Histograms offer a quick and intuitive way to evaluate normality, but lack the precision of formal statistical tests influenced by factors like bin width or sample size. Therefore, they are best used alongside other methods, such as Q-Q plots or normality tests, to provide a comprehensive understanding of the dataset (Manolesou, 2015).

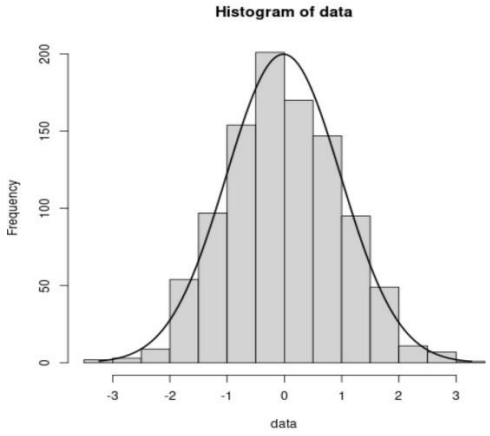

Figure 4.2: Histogram with Normal Distribution Curve Example

(Source: https://www.statology.org/overlay-normal-curve-histogram-in-r/)

The histogram in Figure 4.2 illustrates the frequency distribution of a dataset overlaid with a normal distribution curve. The bars represent the empirical frequencies, while the curve represents the theoretical normal distribution. The close alignment between the histogram bars and the curve suggests that the data closely follows a normal distribution. However, slight deviations from the curve, particularly at the tails, may indicate minor departures from normality, potentially due to skewness or kurtosis. This visualization effectively highlights the dataset's general adherence to normality while allowing for the identification of potential areas of divergence.

4.2 Normality Tests for the Numeric Variables

For the analysis of our data regarding the case of normality, the Kolmogorov-Smirnov (K-S) test was selected as it is particularly well-suited for evaluating large datasets, such as the one under study. Its non-parametric nature allows it to assess deviations from normality, making it an appropriate choice for the variables. These variables exhibit diverse distributions, shaped by the dynamic and competitive nature of Formula 1 as a sport. The statistical values, p-values, and graphical representations for selected cases are presented as follows:

Title	K-S Statistic	K-S p-value
Age	1.000	0.000
Years Experience	0.739	0.000
Race Starts	0.983	0.000
Years with Constructor	0.503	0.000
Q1	0.990	0.000
Q2	0.734	0.000
Q3	0.500	0.000
Qualifying Position	0.928	0.000
Grid Position	0.918	0.000
Position Order	0.926	0.000
Driver Points	0.500	0.000
Cumulative Season Driver Points	0.751	0.000
Fastest Lap Time	1.000	0.000
Laps	1.000	0.000
Total Pit Stops	0.841	0.000
Total Duration ms	1.000	0.000
Top 3 Positions	0.521	0.000
Positions 4-10	0.788	0.000
Outside 10 Positions	0.866	0.000

Table 4.1: K-S tests of the Numeric Variables

The results of the Kolmogorov-Smirnov (K-S) test indicate that none of the numeric variables follow a normal distribution, as all p-values are $< 10^{-3}$, rejecting the null hypothesis of normality. Variables such as Age, Fastest Lap Time, and Laps show extreme deviations, while performance metrics (e.g., Driver Points, Total Duration ms) and grid-related variables (e.g., Grid Position) reflect the variability inherent in

Formula 1 data. The observed non-normality highlights the complex and dynamic nature of the sport, suggesting that non-parametric methods or data transformations may be necessary for further analysis.

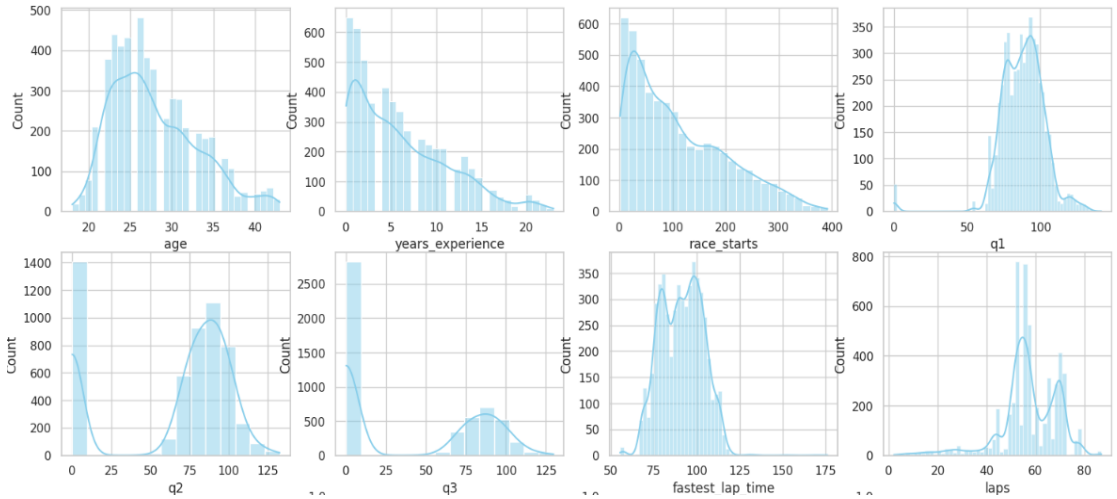

Figure 4.3: Histograms of Selected Variables

The histograms in Figure 4.3 provide a visual representation of the distribution of selected numeric variables, further highlighting their deviations from normality. Variables such as Age and Race Starts exhibit right-skewed distributions, with a concentration of lower values and a long tail towards higher values. Years Experience also shows a clear right skew, reflecting the progression of drivers over time while Laps though more concentrated, also demonstrate asymmetry. Q1, Q2 and Q3 variables can be distinguished for their "normal" appearance which will be investigated later.

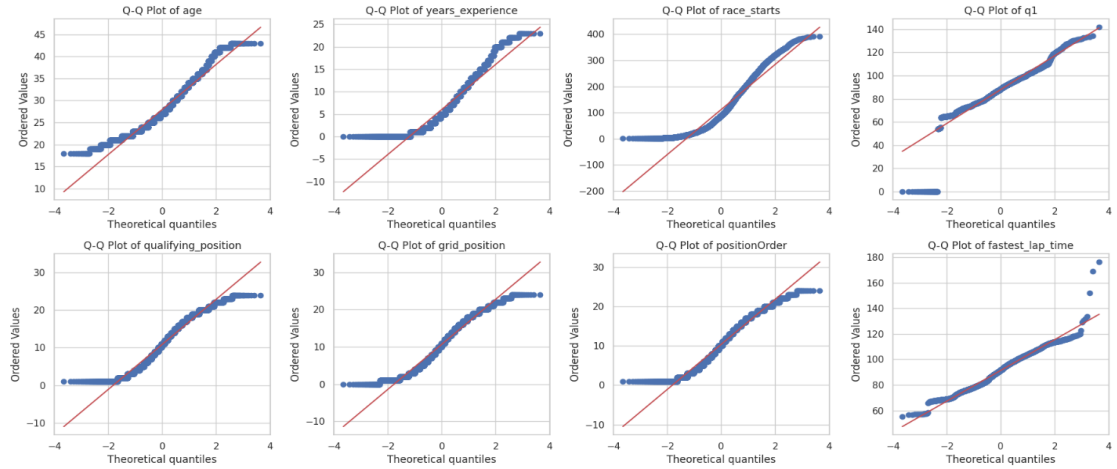

Figure 4.4: Q-Q plots of Selected Variables

The Q-Q plots in Figure 4.4 provide a graphical assessment of the normality of selected variables by comparing their empirical quantiles to the theoretical quantiles of a normal distribution. Most variables exhibit clear deviations from the 45^{o} degree diagonal line with S-shaped patterns, indicating departures from normality. Q1 and Fastest Lap Time align closer to the diagonal in their mid-ranges, their tails deviate significantly, confirming the influence of outliers or non-normal distributions.

4.3 Normality Tests for Q1, Q2, and Q3 (Zeros Excluded)

To further investigate whether the Q1, Q2, and Q3 variables follow a normal distribution -after excluding zero times from drivers who did not compete in each session- the tests were rerun, yielding the following results:

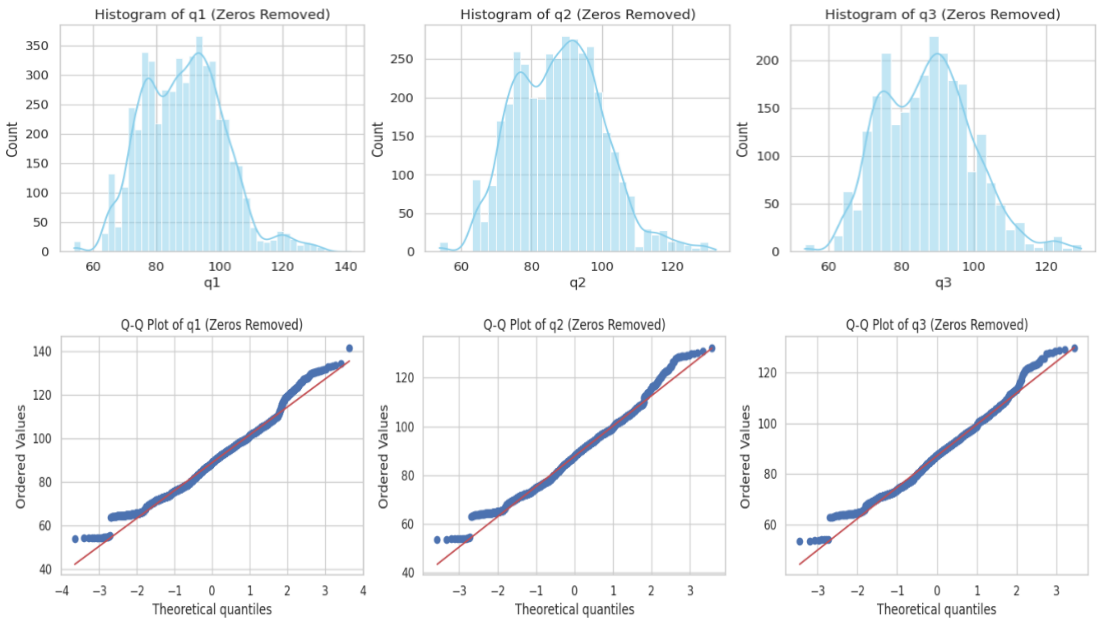

Figure 4.5: Histograms & Q-Q plots of Q1-3 (Zeros Removed)

Figure 4.5 presents histograms and Q-Q plots for the Q1, Q2, and Q3 variables after excluding zero values from drivers who did not compete in the respective sessions. Histograms indicate a distribution that appears roughly normal but exhibits some skewness and potential deviations from a perfect Gaussian shape. Q-Q plots further illustrate how the data align with a theoretical normal distribution. While the points generally follow the expected diagonal line, deviations at the tails suggest the presence of outliers or slight non-normality. These findings highlight the need for additional statistical testing to confirm the normality assumption, as follows:

Title	K-S Statistic	K-S p-value
Q1	1.000	0.000
Q2	1.000	0.000
Q3	1.000	0.000

Table 4.2: K-S tests of Q1-3 (Zeros Removed)

The results of the Kolmogorov-Smirnov (K-S) test suggest a significant deviation from normality for the Q1, Q2, and Q3 qualifying times, as indicated by the p-values of $< 10^{-3}$ for all three variables. However, it is important to consider the context of the data. While the statistical tests reject normality, the data may still exhibit a roughly normal shape, particularly in the central region of the distribution. The extreme values seen in the Q-Q plots could be attributed to outliers from race incidents or changing track conditions. Given the competitive nature of the sport, qualifying times may not follow a perfect Gaussian distribution but could still be considered slightly deviating from normality.

4.4 Correlations

A fundamental tool in statistics is the correlation coefficient, which measures the relationship between variables. Calculating the correlations between variables allows for an evaluation of the degree of interdependence among them and provides valuable insights into the structure of a dataset. This process helps identify patterns, trends, or potential causal links that can guide further analysis. Two of the most widely used correlation coefficients are Pearson's correlation, which measures the strength and direction of linear relationships and Spearman's correlation, which captures monotonic relationships and is particularly effective for non-linear associations.

Pearson's Correlation

The Pearson correlation coefficient, commonly denoted as $\rho(X,Y)$ for population data or r for sample data, is a statistical measure used to quantify the strength and direction of a linear relationship between two continuous random variables. It is one of the most widely used measures of association in statistics due to its simplicity and effectiveness in evaluating linear dependencies.

Definition and Formula

The Pearson correlation coefficient is calculated using the formula:

$$\rho(X,Y) = \frac{cov(X,Y)}{\sqrt{Var(X) \cdot Var(Y)}}$$

Where:

- $cov(X,Y)$ is the covariance between variables X and Y, which measures how changes in X are associated with changes in Y.
- $Var(X)$ and $Var(Y)$ represent the variances of X and Y, respectively.

The covariance is defined as:

$$cov(X,Y) = E[(X - E[X])(Y - E[Y])] = E[XY] - E[X]E[Y]$$

The Pearson coefficient ranges from -1 to 1, with the following interpretations:

- **$0 < \rho \leq 1$**: A positive correlation, meaning that as one variable increases, the other tends to increase as well. A value close to +1 indicates a strong positive linear relationship.
- **$-1 \leq \rho < 0$:** A negative correlation, indicating that as one variable increases, the other tends to decrease. A value close to -1 reflects a strong negative linear relationship.
- **$\rho = 0$:** No linear correlation; the variables are uncorrelated or their relationship is not linear. (Kolyva-Machaira and Bora-Senta, 2013)

Sample Correlation Coefficient

In practice, the Pearson coefficient is often computed using sample data, where it is denoted as r and calculated as:

$$r_{x,y} = \frac{\Sigma_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\Sigma_{i=1}^{n}(y_i - \bar{y})^2}}$$

Here:

- x_i and y_i are the individual data points for the two variables.
- $\bar{x}$ and $\bar{y}$ are the sample means of X and Y, respectively.
- The summation runs over all n observations in the sample.

The sample correlation coefficient also ranges between -1 and 1, with the same interpretation as for the population coefficient. It is important to note that the Pearson coefficient only captures linear relationships. Nonlinear relationships, even if strong, may yield a Pearson coefficient near 0, which can be misleading.

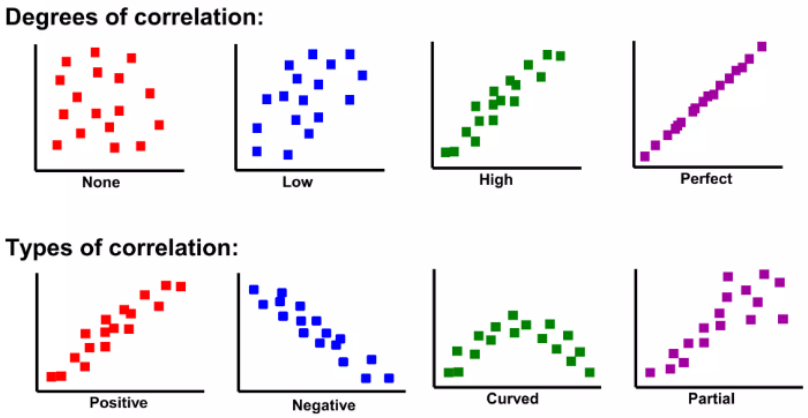

Figure 4.6: Scatterplot Correlation Examples

(Source: https://mudassiriqbal.net/scatter-diagram/)

Figure 4.6 provides a visual representation of different correlation scenarios, with the first four scatterplots illustrating the progression from no correlation (random distribution of points) to a strong linear correlation. As the correlation increases, the points align more closely along a straight line. The next four scatterplots showcase more specific patterns: positive correlation (where both variables increase together), negative correlation (where one variable increases as the other decreases), curved (non-linear) correlation, and partial correlation (where a relationship is observed despite the presence of additional influencing factors). These visualizations highlight the diverse relationships that may exist between variables.

Spearman's Rank Correlation Coefficient

Spearman's rank correlation coefficient, denoted as $\rho(rho)$ or r_s, is a statistical measure named after Charles Spearman. It is a non-parametric method that assesses the strength and direction of the relationship between two variables based on their ranks, rather than their raw values. This makes it especially useful when the data does not meet the assumptions of normality or when the relationship between variables is monotonic but not necessarily linear.

Unlike Pearson's correlation, which measures linear relationships, Spearman's correlation evaluates how well the relationship between two variables can be described by a monotonic function. In simple terms, if one variable tends to increase (or decrease) as the other increases, the Spearman coefficient will capture this trend.

The formula for Spearman's correlation is:

$$\rho_s = \frac{cov(rg_x, rg_y)}{\sigma_{rg_x} \cdot \sigma_{rg_y}}$$

Where:

- rg_x and rg_y are the rank values of X and Y,
- $cov(rg_x, rg_y)$ is the covariance of the ranks,
- σ_{rg_x} and σ_{rg_y} are the standard deviations of the ranks.

Properties of Spearman's Correlation

- **Range**: The Spearman coefficient takes values between -1 and +1.
  - A value of +1 indicates a perfect positive monotonic relationship.
  - A value of -1 indicates a perfect negative monotonic relationship.
  - A value close to 0 suggests no monotonic relationship between the variables.
- **Sign**: The sign of ρ indicates the direction of the relationship:
  - Positive ρ: As X increases, Y also tends to increase.
  - Negative ρ: As X increases, Y tends to decrease.

When there are tied ranks (i.e., when multiple observations share the same value), the rank values are averaged. Spearman's formula can still be applied, and the coefficient retains its properties of measuring monotonicity. Spearman's correlation is particularly suitable for Ordinal Data (ranked variables but without consistent intervals), Non-linear Relationships (monotonic but not linear) and for Small Sample Sizes since it does not assume normality (Triantafyllou, 2023).

4.5 Correlations between the Numeric Variables

This section explores the correlations between the numeric variables of the working dataset. A common and effective method for visualizing correlations is through heatmaps, which represent the correlation matrix using color gradients to indicate the strength and direction of relationships. Given that the data in this analysis is not normally distributed, Spearman rank correlation is used to assess the strength and direction of the monotonic relationship between pairs of variables. With the use of a relevant code the following heatmap was produced:

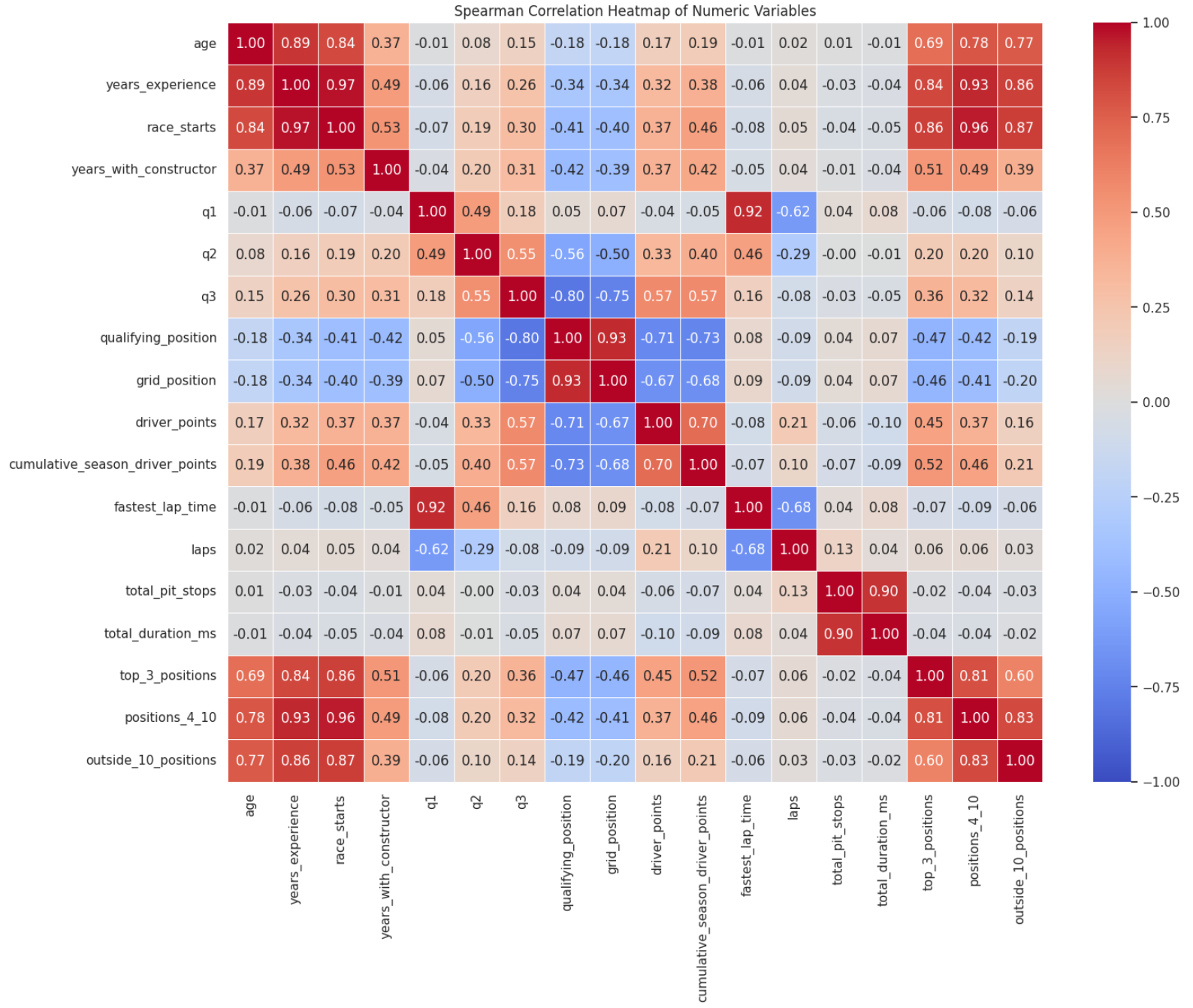

Figure 4.7: Numeric Variables Heatmap

The heatmap reveals several significant relationships among the numeric variables, offering key insights into driver performance and race dynamics. A strong positive correlation is observed between age, years_experience, and race_starts (correlation values exceeding 0.84), illustrating that older drivers tend to have more experience and a higher number of race participations. Notably, Q1 times exhibit a strong positive correlation with fastest lap times (0.92), suggesting that drivers with slower Q1 laps are also less likely to achieve the fastest laps during races. This correlation weakens in subsequent qualifying sessions, with Q2 times and fastest lap times showing a correlation of 0.46, and Q3 times dropping further to 0.16, reflecting the natural elimination process and increased competition in later qualifying stages.

Generally, qualifying times are intercorrelated, with correlations of 0.49 between Q1 and Q2 and 0.55 between Q2 and Q3. As drivers progress through qualifying, these times show increasingly negative correlations with qualifying positions (-0.80 for Q3), grid positions (-0.75 for Q3), and final positions (-0.54 for Q3), translating into a positive correlation with driver points (0.57). The close relationship between qualifying positions and grid positions (0.93) reflects how initial qualifying efforts significantly shape race-day starting positions. Furthermore, experience-related variables, show weak to moderate negative correlations with grid positions and final positions, highlighting the stability and skill of experienced frontrunners.

Positive correlations are also observed for driver point variables with years_with_constructor (0.37 for driver_points and 0.42 for cumulative_season_driver_points), indicating that drivers who stay longer with a team tend to accumulate more points, likely benefiting from a stable and supportive working environment. Additionally, laps completed are negatively correlated with Q1 times (-0.62) and fastest lap times (-0.68), suggesting that backmarkers—drivers with slower laps—are more likely to experience race incidents or fail to complete races.

Finally, the cumulative performance categories (top 3 positions, positions 4-10, and outside 10 positions) show strong positive correlations with various factors as experience variables, while correlating negatively with qualifying and grid positioning, emphasizing the importance of strong starts for podium finishes.

These findings provide a comprehensive understanding of the interplay between driver characteristics, race outcomes, and performance metrics, with precise correlations illustrating key relationships. This analysis lays a robust foundation for advanced techniques such as clustering or predictive modeling.

4.6 Correlations for the Numeric Variables grouped by the Race Outcome Variable

This section examines the correlations between numeric variables in the working dataset, grouped by the Race Outcome variable, which includes the categories Podium finish, Points without podium, and No points. Spearman rank correlation is used within each outcome group and by analyzing these correlations separately for each race outcome, the goal is to uncover how indifferent factors to the race results, providing insights for subsequent analyses such as clustering and classification. Once again using heatmap for the subgroups the result was the following:

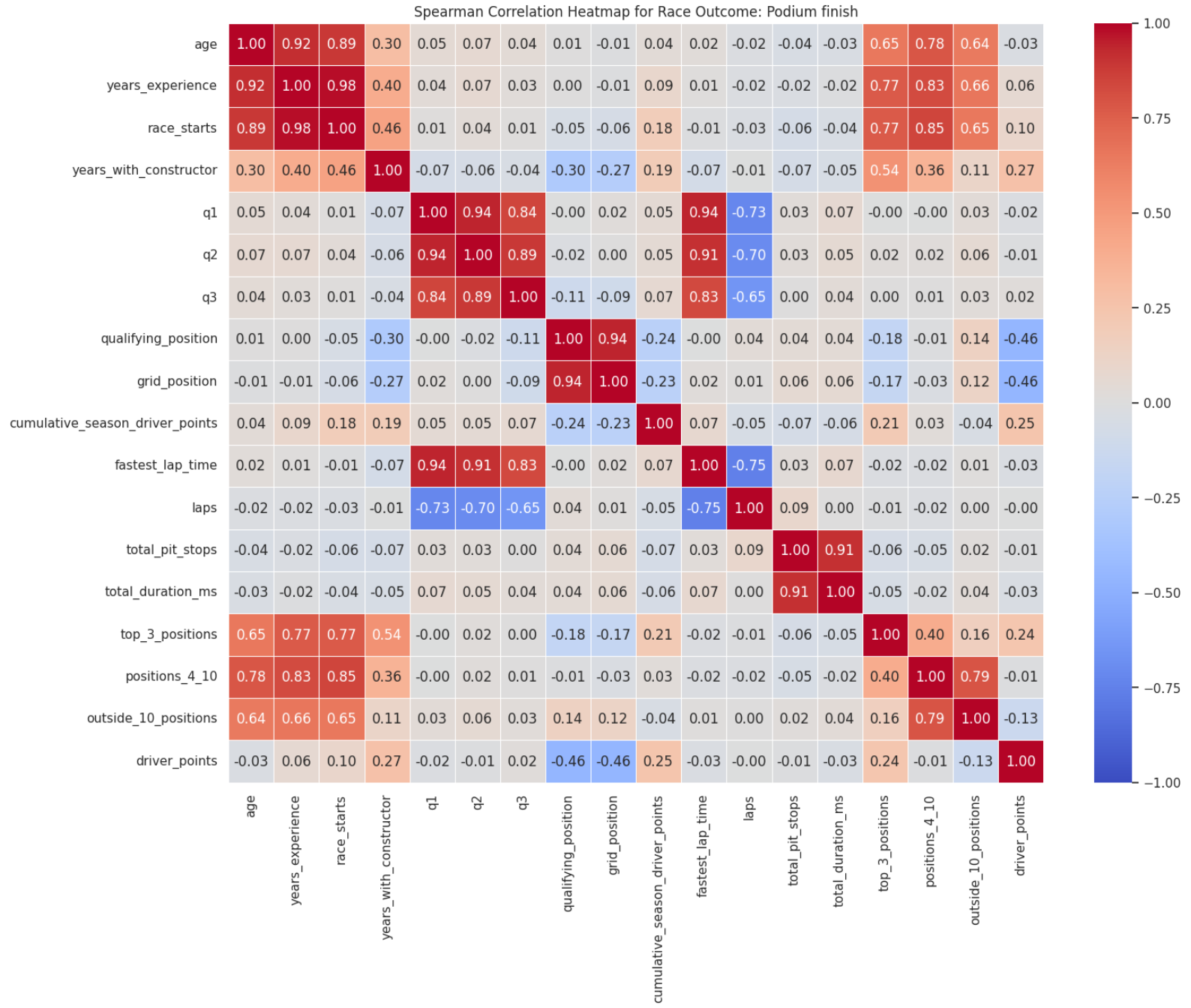

Figure 4.8: Numeric Variables Heatmap for Podium finishes

Figure 4.8 presents the correlation values specifically for podium finishes, revealing notable differences compared to the non-grouped heatmap. Experience-related variables, such as age, years of experience, and race starts, exhibit weaker correlations with positioning and driver points, indicating that experience has a diminished impact on podium outcomes. Qualifying times (Q1, Q2, and Q3) show strong positive correlations with each other, highlighting consistency throughout the qualifying sessions. Furthermore, the relationship between positioning variables and lap times becomes increasingly pronounced, demonstrating tighter performance metrics among podium finishers.

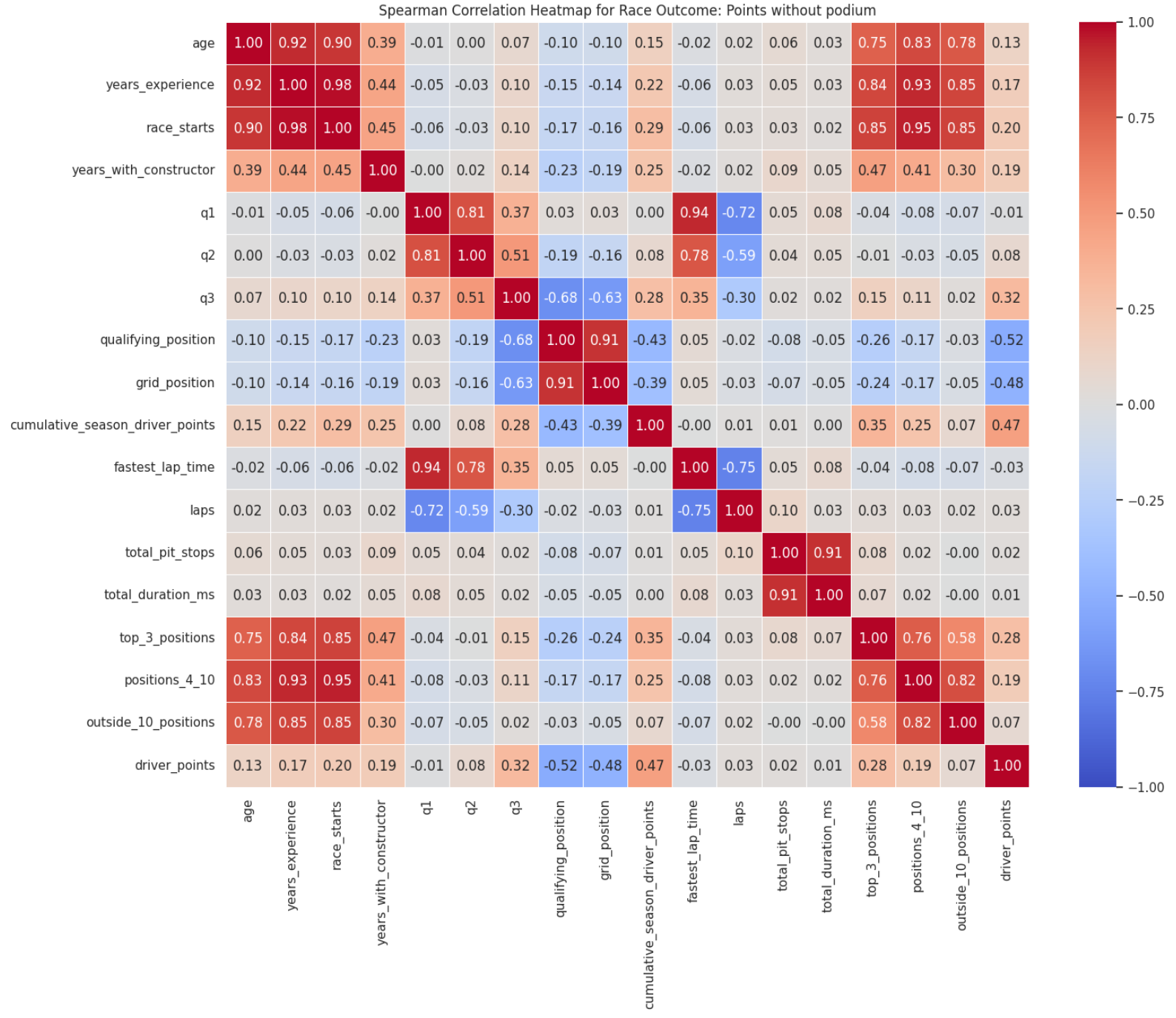

Figure 4.9: Numeric Variables Heatmap for Points without Podium finishes

Figure 4.9 illustrates the correlation values for races resulting in points without podium finishes, showcasing several shifts compared to the podium-focused heatmap. Experience-related variables, such as age, years of experience, and race starts, begin to play a stronger role in influencing positioning and points, reflecting the value of consistency and experience in achieving mid-range success. In contrast, the influence of qualifying (Q1, Q2, Q3) and fastest lap times begins to fade, as their correlations with positioning and performance metrics weaken. Additionally, the correlations between qualifying sessions themselves become less pronounced, indicating reduced consistency among non-podium point scorers. Notably, the relationship between Q3 times and positioning weakens, suggesting that final qualifying performance has less impact on finishing positions.

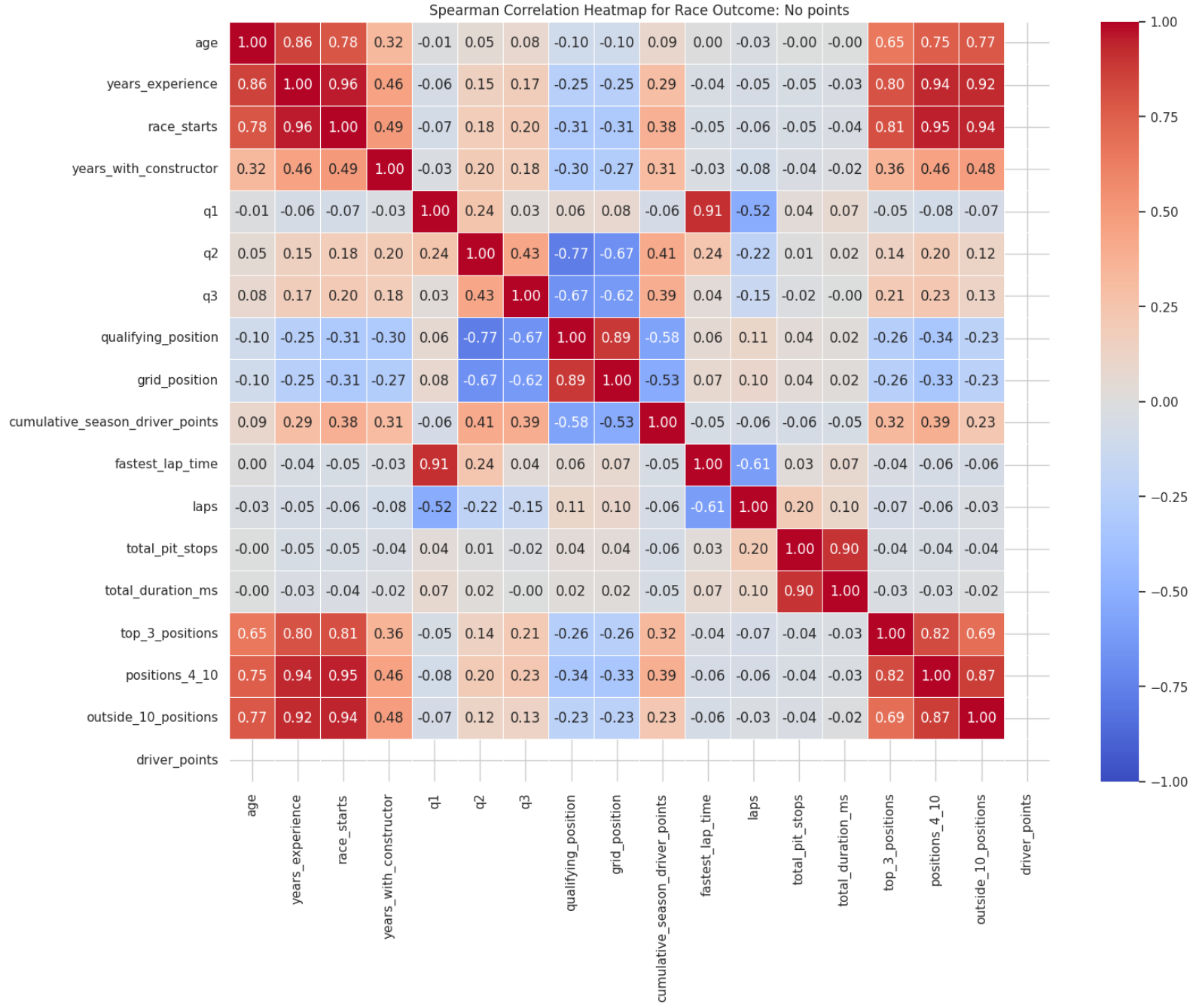

Figure 4.10: Numeric Variables Heatmap for No points finishes

Finally, Figure 4.10 presents the correlation values for races resulting in no points finishes, highlighting several notable shifts compared to the previous heatmaps. The overall structure of the heatmap closely resembles that of the non-grouped heatmap, reinforcing trends observed in Figure 4.8, albeit without the inclusion of the driver points variable for logical reasons. Correlations between experience-related variables and positioning become even more pronounced, emphasizing how experience plays a more substantial role in determining finishing positions for non-podium finishers. In this scenario, qualifying times exhibit a higher correlation primarily with starting positions, both qualifying and grid position, indicating that the qualifying performance influences the initial placement more than the final results, while only Q1 lap times correlating highly to the fastest laps. These shifts reflect the significant impact of experience and qualifying positions for those who do not secure points.

In conclusion, the analysis of non-grouped and grouped heatmaps highlights how correlations between key metrics shift based on race outcomes. While the non-grouped analysis reveals general trends, such as the strong influence of experience and qualifying on positioning, the grouped analyses provide more nuanced insights. For podium finishes, qualifying times show strong internal consistency, but experience plays a lesser role. In races resulting in points without podium, experience becomes more impactful, while qualifying metrics weaken. In no-point finishes, experience and starting positions dominate, closely mirroring the non-grouped patterns. These findings emphasize how race outcomes reshape performance dynamics, guiding further analytical approaches.

Chapter 5: Generalized Linear Models

The objective of this chapter is to analyze the significance of various variables in relation to the final grouped outcomes of the races. To address this, we will implement generalized linear models, ensuring they are appropriately adjusted to account for the response variable *Race_Outcome*. This variable is categorical and divided into three distinct categories: Podium finish, Points without podium, and No points (as seen previously). By exploring these categories, we aim to better understand how different factors contribute to the race results, ultimately helping to identify key determinants of success in these events.

5.1 Regression Analysis

It is quite a common need to study two or more variables simultaneously, with the goal of determining how these variables are related to each other. The most typical form of the relationship between two variables, X and Y, aimed at predicting one variable through the other, is called simple regression analysis. In this context, the variable X is referred to as the independent variable because its values (x_i) are determined by the analyst, while Y is referred to as the dependent variable because its values (y_i) depend on the values of X.

When the regression analysis involves more than one independent variable, i.e., $X^1, X^2, \dots, X_p$, the model applied is called multiple regression, which is a more generalized form of simple regression. The general formula for this model is as follows:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \dots + \beta_p X_{ip} + \varepsilon_i$$

Where Y_i $(i = 1, \dots, \nu)$ is the i-th value of the dependent variable Y, X_i are the i-th values of the p independent variables X_i, and $\beta_0, \beta_1, \dots, \beta_p$ are the parameters of the model, while ε_i represents the errors.

For the above multiple regression model to be valid, in addition to the normality of the errors and, consequently, the dependent variable, the following conditions must be met:

- The quantities $\beta_0, \beta_1, \dots, \beta_p$ are unknown parameters.
- The $x_{i1}, \dots, x_{ip}$ are known values, specifically the values of the independent (or explanatory) variables for the i-th repetition of the experiment. These values are determined by the researcher conducting the experiment.
- Y_i is the value of the dependent variable (or response variable) for the i-th repetition of the experiment. Y_i is a random variable.
- The ε_i, $i = 1, \dots, \nu$, are random errors with a mean of 0 and variance σ^2.

- The errors ε_i and ε_j corresponding to different repetitions of the experiment ($j \neq i$) are considered uncorrelated, meaning that $Cov(\varepsilon_i, \varepsilon_j) = 0$ for $j \neq i$. (Koutras, 2020)

As previously mentioned, the analysis conducted in this chapter focuses on the categorical variable that interprets the final result of a driver in a race. Given its structure—three possible outcomes and being a categorical variable—it is not appropriate to assess the normality of the variable, and therefore, performing a regression model would not be suitable. For this reason, we will analyze the data using generalized linear models, which are an extension of the regression model.

5.2 Generalized Linear Models

A simple way to understand that generalized linear models are an extension of regression models is by recognizing that, instead of using the mean value of Y, a function of this mean is used, denoted as g. In short, a link function g connects the stochastic part of the model (the mean of Y) with the non-stochastic part, which is a linear combination of the explanatory variables X_i. Therefore, the prediction function is formulated as follows:

$$\eta_i = g(\mu_i) = \beta_0 + \sum_{j=i}^{k} \beta_j X_{ij} \ , for\ j = 1, \ldots, p$$

where X_{ij} represents the value of the variable X_j for the i-th observation.

The formula mentioned for the function $g(\mu_i)$ pertains to distributions that belong to the exponential family of distributions. One of the main advantages of generalized linear models is that no assumptions are made about the distribution of errors, unlike regression, which assumes normally distributed errors. Moreover, it is evident that these models have a broader range of applications and that parameter estimates are obtained through the maximum likelihood method, which ensures a set of desirable properties.

Finally, in most cases, there is no need to assume constant variance for the values of Y, nor is there a need to use different models based on whether the explanatory variables are quantitative or qualitative. However, it is noted that in regression analysis, quantitative variables are used (Politis, 2021).

5.3 Logistic Regression

The logit model, widely recognized in the field of Statistics as the logistic regression model, is a special case of a generalized linear model (GLM). It is used for describing data and investigating the relationship between a categorical response variable and one or more explanatory variables, with the connection being established through the logit link function.

Depending on the nature of the response variable's categories, logistic regression is classified into three types:

- Binary logistic regression (for two categories).
- Multinomial logistic regression (for more than two categories).
- Ordinal logistic regression (when the categories follow an increasing order).

Despite the existence of multiple types, the binary logistic regression model receives the most attention, as it is widely applied in various fields.

Essentially, logistic regression is a classification model based on probability theory. The key difference between logistic and linear regression lies in the nature of the response variable (in logistic regression, it is categorical, whereas in linear regression, it is continuous and quantitative). Moreover, in classical linear regression, parameter estimation is performed using the least squares method, while in logistic regression, it is based on the maximum likelihood estimation (MLE), which is commonly used in generalized linear models (GLMs).

The logit model is utilized in a wide range of scientific disciplines, including political and social sciences, medicine, machine learning, economics, and more. For a more detailed description of logistic regression methods, one can refer to the works of Cox & Snell (1989), Hosmer & Lemeshow (2000), and Long & Freese (2014).

Logit Link Function

In statistics, the logit link function is the logarithm of the odds ratio for a given event. The odds ratio for an event with probability π in the interval (0,1) is defined as the ratio of the probability of occurrence π to the probability of non-occurrence 1-π:

$$odds = \frac{\pi}{1-\pi}, where\ odds \in (0, \infty)$$

$$\log odds = logit(\pi) = log\left(\frac{\pi}{1-\pi}\right), \qquad LO\ \epsilon\ (-\infty, +\infty)$$

The logit function has a domain of (0,1) and a range of $\mathbb{R}$. Thus with $\pi\ \epsilon\ (0,1)$, it is defined as the logarithm of the odds ratio (log odds - LO):

$$logit^{-1}(x) = \frac{e^x}{1+e^x}$$

Additionally, the logit function is monotonically increasing, and the following properties hold:

$$\pi_i < \frac{1}{2} \Leftrightarrow odds < 1 \Leftrightarrow logit(\pi_i) < 0$$

$$\pi_i = \frac{1}{2} \Leftrightarrow odds = 1 \Leftrightarrow logit(\pi_i) = 0$$

$$\pi_i > \frac{1}{2} \Leftrightarrow odds > 1 \Leftrightarrow logit(\pi_i) > 0$$

The Logit Model

As previously stated, the logit model is a GLM model with a binary response and the logit function as its link function. Essentially, the cumulative distribution function (CDF) of this model follows the standard logistic distribution:

$$\Lambda(x) = \frac{e^x}{1 + e^x}, \quad for\ x \in \mathbb{R},$$

with probability density function (PDF):

$$l(x) = \Lambda'(x) = \frac{e^x}{(1 + e^x)^2}, \quad for\ x \in \mathbb{R}.$$

The standard logistic distribution is symmetric around zero, with a mean of 0 and variance $\pi^2/3$ (Goutos, 2021).

Now, let Y_i be a binary response variable, where:

$$E(Y_i) = P(Y_i = 1) = \pi_i$$

For a set of parameters $\beta = (\beta_0, \beta_1, \ldots, \beta_p)'$ and explanatory variables $x = (1, x_1, x_2, \ldots, x_p)'$, replacing the cumulative function F(·) with the standard logistic distribution Λ(·), we derive the logit model:

$$logit(\pi) = log\left(\frac{\pi}{1 - \pi}\right) = x'\beta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p ..$$

By inverting this function, we obtain:

$$\pi = \Lambda(x'\beta) = \frac{e^{x'\beta}}{1 + e^{x'\beta}} .$$

Probit Regression

Similar to the logit model, the probit model is a popular choice for binary response variables and is a type of Generalized Linear Model (GLM). Like logit, parameter estimation in the probit model is performed using the maximum likelihood method. The model was introduced by Bliss (1935), who coined the term from "probability unit." Bliss initially applied the probit model in toxicology experiments, where he divided test animals or insects into groups (not necessarily of equal size) and administered different doses of a drug to each group. He then recorded the number of surviving subjects in each group and modeled survival probability as a function of dosage using the probit model. (McCullagh & Nelder, 1989)

The Probit Model

The structure of the probit model is similar to the logit model, but it differs in the choice of link function. Instead of the logistic function, the probit model uses the cumulative distribution function (CDF) of the standard normal distribution, denoted as Φ(·). This means that:

$$\pi = \Phi(x'\beta)$$

where Φ(·) represents the CDF of the standard normal distribution, which is symmetric around zero, with a mean of 0 and variance of 1.

The inverse function of the probit model is:

$$\Phi^{-1}(\pi) = x'\beta$$

Because of its reliance on the normal distribution, the probit model is widely used in economics and social sciences, particularly when modeling binary outcomes where normality assumptions are reasonable.

Complementary Log-Log Regression

The complementary log-log (cloglog) model was first introduced in 1922 by the British statistician R.A. Fisher. Fisher described an experiment in which he took a soil or water sample and performed a series of dilutions to determine the presence or absence of a microbial contaminant. To model this, he applied the cloglog transformation and used maximum likelihood estimation. (Piegorsch, 1992)

The Complementary Log-Log Model

The cumulative distribution function (CDF) of this model follows the standard Gumbel distribution:

$$W(x) = 1 - e^{-e^x} \ , -\infty < x < \infty$$

with its probability density function given by:

$$w(x) = W'(x) = e^x(1 - W(x)) = e^{x-e^x}$$

Unlike the normal and logistic distributions, the Gumbel distribution is not symmetric around zero. It has a mean of approximately -0.5772 and a variance of $\pi^2/6$. A key characteristic of the cloglog model is that it performs better in cases where the probability π approaches 0 slowly but reaches 1 more rapidly as a function of the explanatory variables (Goutos, 2021).

Replacing F(·) with the Gumbel distribution W(·), we obtain the complementary log-log model:

$$\pi = W(x'\beta) = 1 - e - e^{-e^{x'\beta}}$$

with the inverse function:

$$log(-log(1-\pi)) = x'\beta$$

This model is particularly useful in survival analysis and applications where events are rare but become more likely under certain conditions (Goutos, 2021).

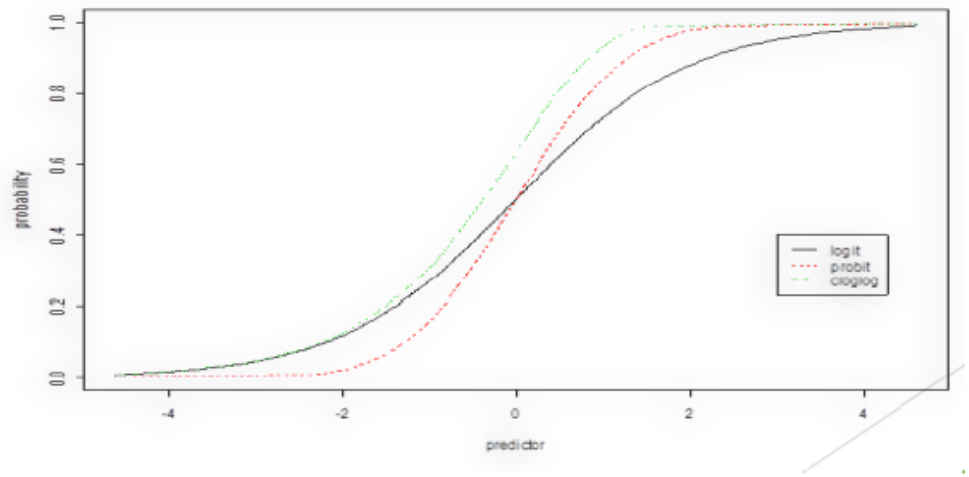

Figure 5.1: Graphical representations of the link functions

(Source: Politis, 2021)

5.4 Multinomial Logistic Regression

Multinomial logistic regression is a generalization of logistic regression used when the response variable has more than two categorical outcomes as can been seen in the Race_Outcome variable.

Let Y be a categorical response variable with J > 2 possible outcomes, occurring with probabilities:

$$\pi_1, \pi_2, \dots, \pi_J \qquad \text{with} \qquad \sum_{j=1}^{J} \pi_j = 1$$

Thus, the probability of observing category *j* is given by:

$$P(Y = j) = \pi_j, j = 1, \dots, J.$$

In the case of a binary response variable (J = 2), this reduces to the standard logistic regression model, where $\pi_1 = \pi$ and $\pi_2 = 1 - \pi$.

Now, suppose there are *p* explanatory variables, such that each observation *Y* is associated with a predictor vector:

$$x = (x_0, x_1, \dots, x_p)'$$

where, typically, $x_0 = 1$ to account for the intercept.

It is important to note that, in a more general framework, the explanatory variable vector *x* may include not only the main effects of continuous variables but also interaction terms, dummy variables, and transformations of the original predictors.

The objective is to model the probabilities π_j as a function of the predictor variables. Since these probabilities are constrained to sum to one, they are not independent. To address this, one category is selected as the reference category, against which all other categories are compared. Following standard practice, and consistent with the default implementation in R, we assume that the first category serves as the baseline.

The log-odds (logits) relative to the baseline category ($j = 1$) are defined as:

$$log\left(\frac{\pi_j}{\pi_1}\right), j = 2, \dots, J.$$

The multinomial logistic regression model is then specified as follows:

$$log\left(\frac{\pi_j}{\pi_1}\right) = x'\beta_j = \beta_{j0} + \beta_{j1}x_1 + \cdots + \beta_{jp}x_p, j = 2, \dots, J.$$

where x_0= 1 to include an intercept term.

This formulation allows for the estimation of separate parameter vectors β_j for each category *j*, enabling the modeling of categorical outcomes with more than two levels while maintaining a flexible and interpretable structure (Iliopoulos, 2023).

Multinomial Probit Model

The multinomial probit model is similar to the multinomial logit model but assumes that the error terms follow a multivariate normal distribution rather than a logistic distribution. Instead of using the logit function, it applies the cumulative normal distribution function as the link function.

The model takes the form:

$$P(Y = j) = \Phi(\beta_{0j} + \beta_{1j}X_1 + \beta_{2j}X_2 + \cdots + \beta_{pj}X_p)$$

where Φ(·) represents the cumulative normal distribution function.

Advantages of the multinomial probit model:

- It relaxes the IIA assumption, making it more flexible than the multinomial logit model.
- It allows for correlation between choices, which is useful in cases where the alternatives are not strictly independent.

However, it is computationally more complex and requires numerical methods such as simulation-based estimation (e.g., Monte Carlo methods) to estimate the model parameters (Agresti, 2018).

Multinomial Complementary Log-Log (Cloglog) Model

The multinomial complementary log-log (cloglog) model is used when the response variable exhibits asymmetry, meaning that the probability distribution is skewed rather than symmetric. This makes it particularly useful for rare events or situations where one category is much more frequent than others.

Instead of modeling log-odds (like in the logit model), the cloglog model transforms probabilities using the complementary log-log function:

$$log(-log(1 - P(Y = j))) = \beta_{0j} + \beta_{1j}X_1 + \beta_{2j}X_2 + \cdots + \beta_{pj}X_p$$

where:

- $log(-log(1 - P(Y = j)))$ is the complementary log-log transformation of probabilities.

Advantages of the cloglog model:

- It is useful when one category has an extremely high or low probability compared to others.
- It is often used in hazard models, survival analysis, and event history analysis.

One limitation of this model is that it does not assume symmetry, making it less interpretable in standard classification problems (Mustafa, 2023).

5.5 Goodness-of-fit measures

Once the most appropriate model for our data is fitted, we need to choose the one that best describes the response variable. In usual regression, a numerical measure that evaluates the fit of a model is the R^2 coefficient. For logistic models (or, generally, GLMs), there is no widely accepted measure equivalent to R^2. However, several measures have been proposed (typically known as pseudo-R^2). Next, L_M represents the maximum likelihood under model M, and $\mathbf{L_0}$ is the maximum likelihood of the model with only the intercept. The most important goodness-of-fit measures are presented below.

AIC

One of the most popular criteria for model selection is the Akaike Information Criterion (AIC). For a model with **k** parameters, the criterion is defined as:

$$AIC = -2\,log\,L_M + 2k$$

Among different models, the one with the lowest AIC value is selected as the best.

BIC

This criterion was proposed by Schwarz in 1978 and is used when choosing between two or more models. The BIC is more stringent than the AIC because it applies a greater penalty for the number of parameters in the model. The formula is defined as:

$$BIC = log(n)\,k - 2\,log(L)$$

McFadden

The McFadden criterion is the most popular goodness-of-fit measure related to pseudo-R^2. Values between 0.2 and 0.4 indicate a good fit. Its formula is:

$$1 - \frac{logL_M}{logL_0}$$

Cox and Snell

The formula is:

$$1 - \left(\frac{L_0}{L_M}\right)^{\frac{2}{n}}$$

Nagelkerke / Cragg and Uhler

A modification of the above, when divided by the maximum value. Its formula is:

$$\frac{1 - (L_0 - L_M)^{\frac{2}{n}}}{1 - L_0{}^{\frac{2}{n}}}$$

5.6 Statistical Significance Tests for Variables

After selecting the most appropriate model for the analysis, it is necessary to test the significance of the variables that are useful for explaining variability. One method to test the significance of each variable in a logistic regression model is the Wald test. According to estimation theory, the maximum likelihood estimators (MLEs) for the parameters of a generalized linear model asymptotically follow a normal distribution. Therefore, for a large number of observations (typically greater than 30), we can use the normal distribution to test whether a parameter β significantly differs from zero. The null and alternative hypotheses tested by the Wald test are:

Null hypothesis H_0: $\beta = 0$

vs

Alternative hypothesis H_1: $\beta \neq 0$

The statistical function used for testing the null hypothesis of statistical significance of the model's variable is:

$$z = \frac{\hat{\beta}}{s.e(\hat{\beta})}$$

where, under the null hypothesis H_0: $\beta = 0$, the statistic z asymptotically follows a standard normal distribution N(0,1).

Although the Wald test is an important test for statistical significance in a logistic regression model, it is not the only test that can be performed. The main concept used to assess the goodness of fit of a model is the deviance (Politis, 2021). This concept is similar to the sum of squared residuals in standard linear models, as it represents a measure of the unexplained variability of the model with a constant. The deviance is defined as the logarithm of the likelihood of the fitted model. Intuitively, the smaller the deviance of a model, the closer it is to the saturated model, which indicates a better fit.

In general, the distribution of the deviance is not known. To address this, the likelihood ratio test is used, which essentially compares the difference in deviance between the saturated model (which contains as many variables as there are observations) and the fitted model. This is expressed by the following formula:

$$D1 - D2 = -2\left(\frac{logL(reducedmodel)}{logL(saturatedmodel)}\right)$$

Under the null hypothesis, that the model does not differ from the saturated model, this quantity is known to follow a X_p^2 distribution, where $p = df1 - df2$, and L() is the likelihood function.

5.7 Multicollinearity Test (VIF)

Another important test for model evaluation is checking for multicollinearity among the variables. Multicollinearity does not affect the prediction of the response variable but causes confusion in estimating the model coefficients (leading to large standard errors), meaning the effects of explanatory variables on the response variable cannot be determined accurately. Multicollinearity can be detected primarily through tolerance and the Variance Inflation Factor (VIF).

Tolerance (Senaviratna & A. Cooray, 2019) is the proportion of the variance of an explanatory variable that cannot be explained by the other independent variables. By definition, the tolerance of any particular explanatory variable is equal to $1 - R_i^2$, where R_i^2 is the coefficient of determination obtained by regressing the other variables on the i-th variable. Tolerance values close to 1 indicate low multicollinearity, while values near 0 suggest that multicollinearity could be problematic. The Variance Inflation Factor (VIF) is defined as:

$$VIF = \frac{1}{1 - R^2}$$

It shows how much the variance of the estimated coefficient is inflated due to multicollinearity. The square root of the VIF indicates how much larger the standard error is compared to what it would be if the variable were not correlated with the other explanatory variables. VIF values exceeding 10 are often considered to indicate problematic multicollinearity, but in weaker models (as often happens in logistic regression), values above 5 may raise concerns (Chrysis, 2024).

5.8 Application of Multinomial Logistic Regression

In this section, the previously discussed methodology is applied to identify the most significant variables in the working dataset, aiming to determine the optimal predictors for the Race_Outcome variable. Given that the dependent variable comprises three categories, Multinomial Logistic Regression is the most suitable approach. Furthermore, as the dataset does not follow a normal distribution, this method provides a more flexible and robust framework for analyzing the categorical response variable. The Python code can be found in the attached appendix at the end of this thesis (Appendix 1: Implementation Code of the Present Study).

To ensure the proper structuring of variables and facilitate a smooth execution of the relevant code, categorical variables were encoded. The LabelEncoder function from the Scikit-Learn library was utilized to systematically encode the following variables: driver_name, constructor_name, top_constructor constructor_nationality and race_status. Additionally, for improved interpretability of the outputs, the Race_Outcome variable was manually encoded into three distinct categories: "Podium Finish" (0), "Points Without Podium" (1), and "No Points" (2). With these transformations applied, the variable structure was appropriately formatted for the multinomial logistic regression analysis.

To reduce the number of categories in the ‘driver_nationality’ variable, nationalities were grouped into European and Non-European. The ‘circuit_name’ variable was also encoded into four categories based on overtaking data from https://racingpass.net/circuits/. Circuits were grouped by average overtakes per race: 0–25, 25–50, 50–75, and 75–100. For instance, Monaco, with an average of 11.3 overtakes, falls into the lowest category, as expected for a low-speed, narrow street circuit. Finally, all necessary variables were converted to categorical.

Initial testing, incorporating all variables, demonstrated a moderate accuracy of approximately 67%, with minimal overfitting. However, notable issues related to economic interpretability and multicollinearity were identified. To address these concerns, variables exhibiting the highest levels of multicollinearity and the strongest correlations -previously discussed in the preceding chapter- were systematically examined for their relevance in the final model. The VIF (Multicollinearity) prices can be seen in the table below:

Variable	VIF
driver_name	3,98
driver_nationality	11,04
age	274,93
years_experience	85,78
race_starts	11574,04
constructor_name	6,53
constructor_nationality	3,93
years_with_constructor	3,08
top_constructor	9,49
year	934,73
round	80,22
circuit_name	18,96

valid_time_set_in_q1	450,51
q1	351,13
qualified_for_q2	244,75
q2	232,83
qualified_for_q3	131,20
q3	121,70
qualifying_position	68
grid_position	28,58
positionOrder	22,93
race_status	11,96
driver_points	8,08
cumulative_season_driver_points	4,48
fastest_lap_time	392,39
laps	61,65
total_pit_stops	6,73
total_duration_ms	1,36
top_3_positions	1199,35
positions_4_10	1570,38
outside_10_positions	1361,53

Table 5.1: VIF prices of the Variables

Starting with the variables exhibiting the highest multicollinearity levels, and referring back to the heatmaps in Chapter 4, each pair of variables with a high VIF was analyzed. The variable from each pair that showed the lowest correlations in the heatmaps was retained. This process was repeated until all VIF values were reduced to 5 or lower, resulting in the selection of the following variables:

- ‘constructor_name’
- ‘constructor_nationality’
- ‘years_with_constructor’
- ‘qualifying_position’
- ‘driver_points’
- ‘cumulative_season_driver_points
- ‘total_pit_stops’
- ‘total_duration_ms’
- ‘top_3_positions’
- ‘outside_10_positions’

Continuing the process by further examining intermediate correlations from the heatmaps, the variables ‘driver_points’, ‘total_duration_ms’, and ‘top_3_positions’ were also removed. The final selection of variables was determined using Wald significance tests, which indicated that ‘constructor_name’, ‘constructor_nationality’, and ‘outside_10_positions’ should also be removed. This resulted in a very economical whilst efficient model without significantly compromising accuracy or AIC metrics. Once again, the VIF prices of the selected variables can be seen in the table below:

Variable	VIF
years_with_constructor	1.975231
qualifying_position	2.927308
cumulative_season_driver_points	1.845008
total_pit_stops	3.871200

Table 5.2: VIF prices of the selected Variables

As the selected variables show minimum levels of multicollinearity, the final MNL model was established and the hyperparameters optimized, so the following output was obtained:

```
                          MNLogit Regression Results
==============================================================================
Dep. Variable:           race_outcome   No. Observations:                 3971
Model:                        MNLogit   Df Residuals:                     3961
Method:                           MLE   Df Model:                            8
Date:                Tue, 04 Mar 2025   Pseudo R-squ.:                  0.3513
Time:                        20:02:14   Log-Likelihood:                -2603.2
converged:                       True   LL-Null:                       -4013.0
Covariance Type:            nonrobust   LLR p-value:                     0.000
=================================================================================================
                race_outcome=1       coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------------------
const                            -0.4634      0.219     -2.115      0.034      -0.893      -0.034
years_with_constructor           -0.0765      0.027     -2.882      0.004      -0.129      -0.024
qualifying_position               0.3447      0.023     15.136      0.000       0.300       0.389
cumulative_season_driver_points  -0.0066      0.001     -8.231      0.000      -0.008      -0.005
total_pit_stops                   0.1046      0.061      1.714      0.087      -0.015       0.224
-------------------------------------------------------------------------------------------------
                race_outcome=2       coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------------------
const                            -2.3751      0.261     -9.098      0.000      -2.887      -1.863
years_with_constructor           -0.1438      0.035     -4.121      0.000      -0.212      -0.075
qualifying_position               0.5447      0.024     22.290      0.000       0.497       0.593
cumulative_season_driver_points  -0.0173      0.001    -12.393      0.000      -0.020      -0.015
total_pit_stops                   0.2341      0.070      3.323      0.001       0.096       0.372
=================================================================================================
```

Figure 5.2: Output of the fitted model with Race_Outcome dep. Variable

Figure 5.2 shows the results of the multinomial logistic regression model and provides valuable insights into the factors influencing Race_Outcome. The model demonstrates statistical significance, as indicated by the LLR p-value closing to zero suggesting that the independent variables contribute meaningfully to the prediction of the dependent variable. The pseudo R-squared value of 0.3513 suggests a moderate level of explanatory power, expected for our dataset's characteristics.

Examining the estimated coefficients, qualifying_position has a strong positive impact on the likelihood of finishing in either category 1 (Points without podium) or category 2 (No points), relative to the baseline category (Podium finish). Specifically, an increase in qualifying_position is associated with higher odds of a lower race outcome. Similarly, total_pit_stops is positively associated with both Race_Outcome = 1 and Race_Outcome = 2, implying that with a lesser pitstop strategy there are higher probabilities of a better race outcome.

Conversely, cumulative_season_driver_points has a negative coefficient for both categories, indicating that higher accumulated points throughout the season decrease

the probability of finishing in Race_Outcome = 1 or Race_Outcome = 2, reinforcing the consistency of high-performing drivers. Years_with_constructor also show a negative relation with the Race Outcome suggesting that longer tenures and steady driver-constructor environments show better results. All coefficients are statistically significant as one can see from each Wald test's ($P > |z|$) low p-value.

Additionally, gathered below, various metrics of the fitted model can be seen:

Metric	Value
Train Accuracy	0.7137
Test Accuracy	0.6971
Balanced Accuracy	0.6684
ROC-AUC Score	0.8524

Table 5.3: Metrics of the fitted model with Race_Outcome dep. Variable

Table's 5.3 evaluation metrics indicate a well-performing classification framework for predicting Race_Outcome. The training accuracy of 71.37% and test accuracy of 69.71% suggest that the model performs well, with minimal overfitting. The balanced accuracy of 66.84% accounts for potential class imbalances, providing a more nuanced measure of the model's performance across all categories of the dependent variable. ROC-AUC score of 0.8524 demonstrates strong discriminatory power in distinguishing between the different race outcomes. Overall, these metrics confirm the model's robustness and suitability for the given dataset, ensuring reliable insights into the factors influencing race results.

Closing with the fitted model's evaluation, the Classification Report is presented:

Classification Report				
	Precision	Recall	F1-Score	Support
Class 0	0.67	0.63	0.65	190
Class 1	0.59	0.58	0.58	467
Class 2	0.78	0.80	0.79	667
Accuracy			0.70	1324
Macro Avg	0.68	0.67	0.67	1324
Weighted Avg	0.69	0.70	0.70	1324

Table 5.4: Classification Report of the fitted model

Table 5.4 provides a comprehensive evaluation of the model's predictive performance across the three categories of the Race_Outcome variable. The precision values indicate the proportion of correctly predicted instances for each class, while recall measures the ability of the model to capture all actual instances of a given category. The F1-score, which balances precision and recall, suggests that the model performs best in distinguishing non-point finishes (Class 2) with an F1-score of 0.79, followed by podium finishes (Class 0) and point-scoring finishes (Class 1). The overall accuracy of 0.70 aligns with previous model evaluation metrics, reinforcing

the model's ability to provide reasonably accurate classifications. The above metrics are derived from the Confusion Matrix (Figure 5.3) as follows:

- True Positives (TP): The number of cases correctly predicted as positive by the model.
- True Negatives (TN): The number of cases correctly predicted as negative by the model.
- False Positives (FP): The number of cases incorrectly predicted as positive by the model when they were actually negative.
- False Negatives (FN): The number of cases incorrectly predicted as negative by the model when they were actually positive.

The key evaluation metrics are calculated as:

- **Accuracy:** $\frac{TP+TN}{TP+TN+FP+FN}$
- **Precision:** $\frac{TP}{TP+FP}$
- **Recall:** $\frac{TP}{TP+FN}$
- **F1 Score:** $2 \times \frac{Precision \times Recall}{Precision+Recall}$

(Kamitsis, 2023)

Confusion matrices provide a detailed breakdown of the model's predictions, illustrating the distribution of correct and incorrect classifications across the three categories of the Race_Outcome variable. The following figure presents the results:

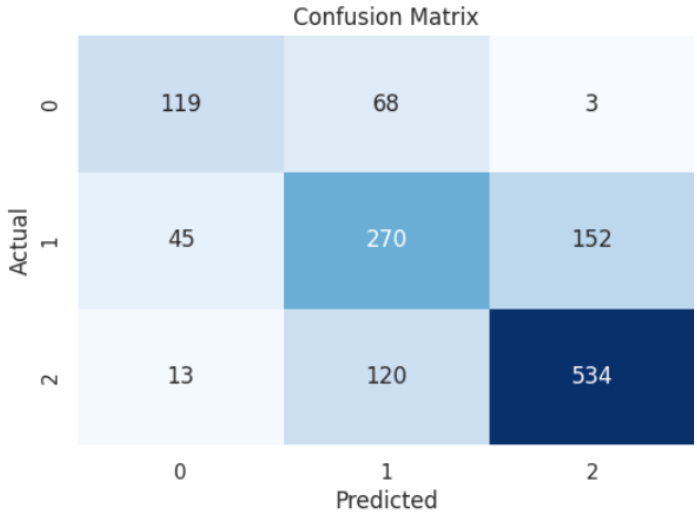

Figure 5.3: Confusion Matrix of the fitted model

Figure 5.3 illustrates the distribution of correct and incorrect predictions for each category of Race_Outcome. A higher concentration of values along the diagonal is shown indicating strong predictive performance, while off-diagonal values represent misclassifications. The most significant misclassification occurs in predicting Class 1 as Class 2, with 152 instances. Given the raw nature of the non-normally distributed large dataset, such misclassifications are expected to some extent.

The ROC curves provide a visual representation of the model's ability to distinguish between the different categories of **Race_Outcome**. By plotting the true positive rate against the false positive rate for each class, the curves help assess overall classification performance, with a higher area under the curve (AUC) indicating better discriminative power. The curves for each class can be seen in the next Figure.

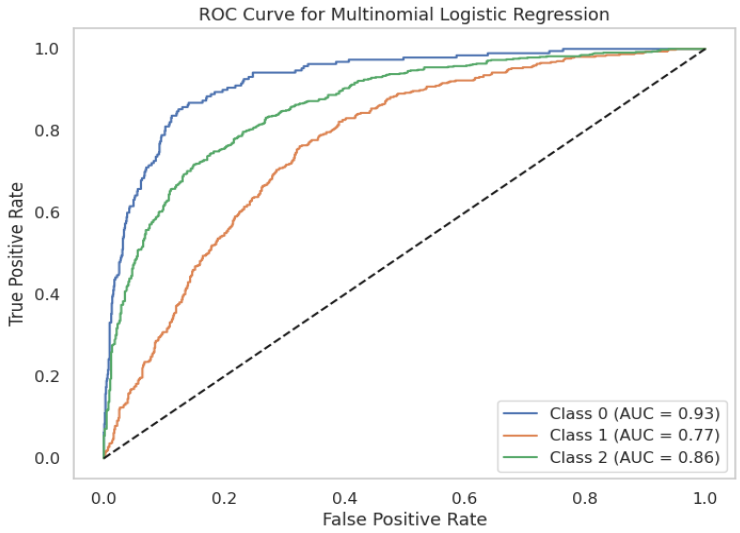

Figure 5.4: ROC Curves for each Class of the fitted model

Figure 5.4 shows the ROC curves of the fitted model across the three categories of Race_Outcome. The AUC (Area Under the Curve) values indicate the model's ability to distinguish between classes, with Class 0 (Podium Finish) achieving the highest AUC at 0.94, suggesting strong predictive accuracy. Class 1 (Points without Podium) exhibits moderate separability with an AUC of 0.77, while Class 2 (No Points) has a relatively better distinction at 0.87. The overall performance suggests that the model is most effective at identifying podium finishes and non-point finishes, with slightly reduced accuracy in differentiating mid-tier outcomes.

Learning Curves provide insight into the model's performance across different training sizes, helping assess bias, variance, and potential overfitting or underfitting. Training and Validation learning curves of the fitted model are presented below

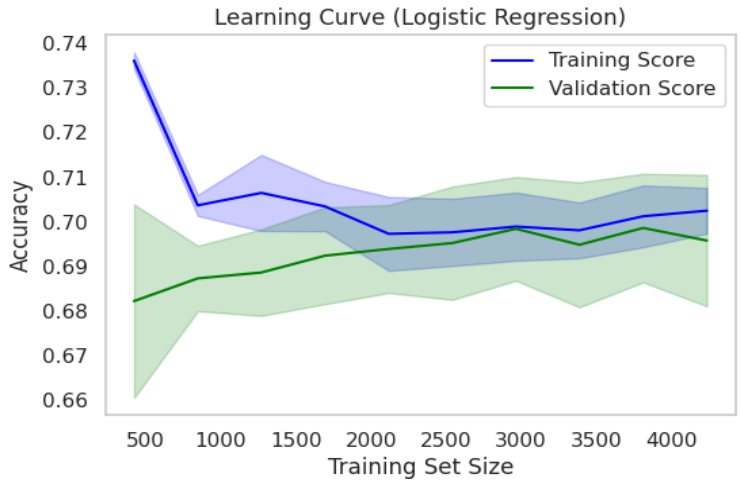

Figure 5.5: Learning Curve of the fitted model

In the Figure 5.5 one can initially focus on the training score being higher due to the model fitting closely to a smaller dataset, but as the training size increases, it stabilizes. The validation score follows a similar trend, showing a slight gap between the two curves, indicating minimal overfitting. The convergence of both scores suggests that the model has learned effectively from the data without significant underfitting or overfitting. The presence of fluctuations, particularly in smaller training sizes, is expected due to variance in limited data availability.

As part of the model diagnostics, an analysis of the residuals was conducted to assess the model's fit. Given the characteristics of our data, which include complexities such as non-normality and various outliers, the residuals indicate a reasonably good fit, with no significant issues or major patterns identified. These findings suggest that the model is well-specified and effectively captures the underlying structure of the data, reinforcing the validity of its predictions.

LassoCV Implementation

In the context of generalized linear models (GLMs) such as multinomial logistic regression (MNL), variable selection is crucial for improving model performance and interpretability. Lasso regression, introduced by Tibshirani (1996), applies L1 regularization to the regression coefficients, encouraging sparsity by shrinking some coefficients to zero. This leads to a simpler model by excluding irrelevant predictors, which is particularly useful in high-dimensional data where collinearity or irrelevant variables may otherwise complicate the model.

The extension of Lasso through LassoCV involves cross-validation to automatically select the optimal regularization parameter, λ. By evaluating the model's performance across different subsets of the data, LassoCV identifies the λ that minimizes prediction error, ensuring an optimal balance between model complexity and predictive accuracy (Hastie et al., 2009). This cross-validation approach helps prevent overfitting, making the model more robust when applied to unseen data.

In our analysis, the variables selected by LassoCV align closely with those chosen in our model, supporting the validity of our feature selection process and indicating that the most relevant predictors have been retained for the multinomial logistic regression model. Specifically, LassoCV select the variables: 'years_with_constructor', 'top_constructor', 'q2', 'qualifying_position', 'grid_position', 'driver_points', 'laps', 'total_pit_stops' , from which in our selection we have gathered the most significant ones and the ones that give higher accuracy and lower AIC and VIF metrics.

Chapter 6: Timeseries

This chapter examines the progression of average fastest lap times from 2004 at Albert Park and Monaco, two iconic Formula 1 circuits with distinct characteristics, to identify trends in performance over time. ARIMA models are applied to explore different configurations and determine the most suitable model for each circuit, with performance evaluated through statistical tests and residual diagnostics. Once the best-fitting models are selected, they are used to forecast lap times for the next two seasons, offering insights into potential future trends. This analysis assesses whether lap times follow a predictable trajectory or are influenced by external factors, providing a data-driven perspective on the evolution of performance in Formula 1. The Python code can be found in the attached appendix at the end of this thesis (Appendix 1: Implementation Code of the Present Study).

6.1 Timeseries in Theory

A set of observations that express the chronical evolution of a characteristic of a stochastic phenomenon constitutes a time series. The observations are recorded sequentially over time and are typically equally spaced. Since the evolution of the characteristic's value occurs in a random manner, it can be described by a family of random variables X_t , $t = 1, 2, \dots, T$. The objective of time series analysis is to use the observed values to construct a stochastic model that enables predictions of future values of the characteristic being analyzed. Initially, for a time series to be analyzed, it must satisfy the conditions of stationarity (Triantafyllou, 2023).

6.2 Time Series Stationarity

Non-stationarity is a significant issue in time series analysis, especially when making predictions, as it can lead to inaccurate results. A time series $\{X_t, t = 1, 2, \dots\}$ is considered stationary if it satisfies the following three conditions:

- $E(X_t) = \mu$, for every $t = 1, 2, \dots$ (constant mean)
- $Var(X_t) = \sigma^2$, for every $t = 1, 2, \dots$ (constant variance)
- $Cov(X_{t_s}, X_{t_u}) = \gamma_{t_s - t_u}$,

where $\gamma_j = \gamma_{t_s - t_u}$ is the autocovariance function with lag j. The autocovariance function expresses the covariance between time series values at two different time points separated by j units. This function takes values within the range $(-\infty, +\infty)$ and is influenced by the measurement unit of the analyzed characteristic.

As previously mentioned, stationarity is a fundamental prerequisite for time series analysis. However, in practice, most time series are non-stationary, requiring transformations to ensure stationarity by smoothing violations. Violations of stationarity generally fall into two major categories: trending time series and time series with both trend and seasonality (Triantafyllou, 2023).

6.2.1 Time Series with Trend

One of the primary violations of stationarity in time series is the presence of a trend. A time series with a trend can be expressed as:

$$X_t = m_t + Y_t, t = 1,2, \dots, T$$

where m_t represents the trend function, which must be removed to achieve stationarity, and Y_t is a stationary time series with a mean of 0 and constant variance σ^2. The objective of trend-removal methods is to accurately estimate the trend function and eliminate it to obtain a stationary series. Three common methods for handling trends in time series are presented below.

Linear Trend Method

In this method, it is assumed that the trend in the time series follows a linear form:

$$m_t = \beta_0 + \beta_1 t, t = 1,2, \dots, T$$

where β_0 and β_1 are unknown parameters. To determine the best-fitting model, the sum of squared differences between the observed time series values and the estimated trend line $\widehat{m} = \hat{\beta}_0 + \hat{\beta}_1 t$ must be minimized. Specifically, the least squares estimators $\hat{\beta}_0$ and $\hat{\beta}_1$ to minimize the sum $SS = \sum_{t=1}^{T}(X_t - \beta_0 - \beta_1 t)^2$ and are obtained by using:

$$\hat{\beta}_1 = \frac{\sum_{t=1}^{T}(t-\bar{t})(X_t-\bar{X})}{\sum_{t=1}^{T}(t-\bar{t})^2} \text{ and } \hat{\beta}_0 = \bar{X} - \hat{\beta}_1 t ,$$

where $\bar{X}$ and $\bar{t}$ are the mean values of the observations and time indices, respectively. The trend is then removed from the time series, resulting in a stationary series $\widehat{Y_t}$:

$$\widehat{Y_t} = X_t - (\hat{\beta}_0 + \hat{\beta}_1 t).$$

Polynomial Trend Method

This method assumes that the trend follows a polynomial function of degree k:

$$m_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \dots + \beta_k t^k , t = 1,2, \dots, T$$

where β_0, β_1, β_2, …, β_k are unknown parameters. The least squares estimation approach, similar to the linear trend method, is applied to estimate these parameters. Once estimated, the trend is removed, resulting in a stationary time series:

$$\widehat{Y_t} = X_t - (\hat{\beta}_0 + \hat{\beta}_1 t + \hat{\beta}_2 t^2 + \dots + \hat{\beta}_k t^k) , , t = 1,2, \dots, T$$

Differencing Method

Unlike the previous methods, differencing does not estimate the trend but directly removes it. The differencing operator ∇X_t transforms the time series into a new stationary series. The differencing method varies based on the trend type:

- For a linear trend ($m_t = \beta_0 + \beta_1 t$), applying the first difference $= X_t - X_{t-1}$ results in:

$$\nabla X_t = \nabla(m_t + Y_t) = \beta_1 + \nabla Y_t$$

- For a polynomial trend of degree k ($m_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \ldots + \beta_k t^k$), applying the k-th order difference removes the trend:

$$\nabla^k X_t = \nabla^k(\beta_0 + \beta_1 t + \beta_2 t^2 + \ldots + \beta_k t^k + Y_t) = k!\beta_k + \nabla^k Y_t$$

(Triantafyllou, 2023).

6.2.2 Time Series with Trend and Seasonality

In this category, the time series X_t is described by the following equation:

$$X_t = m_t + s_t + Y_t, t = 1,2,\ldots,T$$

where Y_t is a stationary time series with a mean of 0 and constant variance σ^2, m_t represents the trend function, and s_t denotes the seasonal component of the time series. The goal of the method discussed below is to eliminate both the trend and seasonality without directly estimating them.

Differencing Method

If the time series exhibits seasonality with a period τ, meaning that $s_t = s_{t+\tau}$, then the seasonal differences of order τ are computed as:

$$\nabla_\tau X_t = X_t - X_{t-\tau} = m_t + s_t + Y_t - m_{t+\tau} - s_{t+\tau} - Y_{t+\tau} = (m_t - m_{t+\tau}) + (Y_t - Y_{t+\tau})$$

This transformation results in a new time series that still contains a trend component $(m_t - m_{t+\tau})$ but removes the seasonal component, while also producing a stationary series $\nabla_\tau Y_t = Y_t - Y_{t+\tau}$. At this stage, the seasonal component has been eliminated from the time series.

Next, one of the trend removal methods discussed in the previous section is applied to further transform the series into a stationary one.

When a time series exhibits both trend and seasonality, the preferred approach is to first apply seasonal differencing to remove seasonality, followed by trend differencing to eliminate the trend. To ensure stationarity, it is also necessary to conduct a unit root test, verifying that the transformed time series meets the conditions for stationarity (Triantafyllou, 2023).

6.3 Unit Root Test

The most well-known stationarity test for a time series is the Dickey-Fuller test, developed by Dickey and Fuller. This test examines the presence of a unit root, which indicates whether a time series is non-stationary. The extended version, known as the Augmented Dickey-Fuller (ADF) test, is commonly used in time series analysis. The test is applied in three different forms, corresponding to three different model specifications:

- Without a constant and trend:

$$\nabla X_t = \delta X_{t-1} + \sum_{i=1}^{k} \nabla X_{t-i} + W_t$$

- With a constant (intercept):

$$\nabla X_t = \mu + \delta X_{t-1} + \sum_{i=1}^{k} \gamma_i \nabla X_{t-i} + W_t$$

- With a constant and a time trend:

$$\nabla X_t = \mu + \beta_t + \delta X_{t-1} + \sum_{i=1}^{k} \gamma_i \nabla X_{t-i} + W_t$$

Hypotheses of the ADF Test:

- Null hypothesis (H_0): $\delta = 0$ (The time series has a unit root and is non-stationary).
- Alternative hypothesis (H_1): $\delta < 0$ (The time series is stationary).

If the p-value > 0.05, there is not enough evidence to reject the null hypothesis, indicating the presence of a unit root in the time series. In such cases, the series is non-stationary, requiring further transformations (such as differencing) to stabilize the mean and remove trends or seasonality (Chrysis, 2024).

6.4 Model Selection and Validation

Once the previous steps have been completed and a stationary time series has been obtained, it is possible to proceed with its analysis. The first step in selecting an appropriate model is examining the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots. These visualizations help determine the best-fitting ARIMA(p,d,q)(P,D,Q)s model.

Since different analysts may choose different models for the same time series, it is necessary to validate model selection. If multiple candidate ARIMA models fit the data, statistical criteria such as AIC, BIC, and AICc are used to determine the optimal model. The model with the lowest value in these criteria is considered the best. Once the best model is chosen, residual diagnostics are performed to assess its fit and validity.

Model Selection

The ARIMA(p,d,q)(P,D,Q)s models are widely used for time series forecasting. The seasonal component (P,D,Q)s accounts for seasonality in the data. If a time series lacks seasonality, the seasonal parameters are set to zero: (0,0,0)0.

General ARIMA Equation with Seasonality**:**

$$\Phi_P(B^s)X_t = \Theta_Q(B^s)W_t \,,$$

where: $\Phi_P(B^s) = 1 - \Phi_1 B^s - \Phi_2 B^{2s} - \ldots - \Phi_P B^{ps}$, $\Theta_Q(B^s) = 1 - \Theta_1 B^s - \Theta_2 B^s - \ldots - \Theta_Q B^{Qs}$.

For non-seasonal ARIMA models, the equation simplifies to:

$$X_t = \varphi_1 X_{t-1} + \varphi_2 X_{t-2} + \ \ldots \ + \varphi_p X_{t-p} + W_t + \theta_1 W_{t-1} + \ \ldots \ + \ \theta_q W_{t-q} \,,$$

where the parameters $\varphi_1, \varphi_2, \dots, \varphi_p \in \mathbb{R}$ and $\theta_1, \theta_2, \dots, \theta_p \in \mathbb{R}$ weigh the contribution of the terms of the under analysis timeseries.

After selecting the most promising models, the best model is identified using AIC, BIC, and AICc criteria and checked with respective diagnostics.

Akaike Information Criterion (AIC)

The AIC formula is:

$$AIC = 2k - 2lnL\left(\widehat{\varphi_1}, \widehat{\varphi_2}, \dots, \widehat{\varphi_p}, \widehat{\theta_1}, \dots, \widehat{\theta_q}, \widehat{\mu}, \widehat{\sigma^2}\right),$$

where: $k = p + q + 1$ or $k = p + q + 2$

Bayesian Information Criterion (BIC)

The BIC formula is:

$$BIC = 2ln(n) - 2lnL\left(\widehat{\varphi_1}, \widehat{\varphi_2}, \dots, \widehat{\varphi_p}, \widehat{\theta_1}, \dots, \widehat{\theta_q}, \widehat{\mu}, \widehat{\sigma^2}\right).$$

Since BIC includes a penalty term ln(n), it tends to favor simpler models by discouraging unnecessary parameters. Typically, BIC selects the same model as AIC or one with fewer parameters.

Corrected AIC (AICc)

For small sample sizes, the corrected AIC (AICc) is preferred:

$$AICc = AIC + \frac{2k^2 + 2k}{n - k - 1},$$

as $n \to \infty$, AICc approaches AIC.

Residual Diagnostics

Normality Test

A critical validation step is checking whether residuals follow a normal distribution. This can be done graphically (histograms, Q-Q plots) or using statistical tests. The Anderson-Darling test is the primary normality test used here, though Shapiro-Wilk test is computed for reference but is less reliable for time series data. Both tests where discussed in Chapter 4

Independence Test

The Box-Ljung test assesses the independence of residuals, for whether the residuals are uncorrelated:

- Null hypothesis (H_0): The residuals are independent.

  vs

- Alternative hypothesis (H_1): The residuals are autocorrelated.

If the test rejects H_0, it means the model does not sufficiently capture the dependencies in the data (Rakintzis, 2023).

If a model does not satisfy normality and independence assumptions, its order is increased by one until these conditions are met.

6.5 Applied Timeseries Methods in F1 Fastest Lap Times

Next, the aforementioned methodology will be applied to analyze the fastest lap time variable as a timeseries, using average lap times from the Australian Grand Prix (Albert Park) and the Monaco Grand Prix, leading to a forecast for the next two seasons. To enhance forecasting accuracy, the dataset was backward extended using data from Chapter's 3 raw data, where results.csv, races.csv, and circuits.csv were merged to construct a timeseries for each Grand Prix. However, both time series have limited data points (20 and 19, respectively, since 2004), placing them at the threshold of statistical reliability (McCleary et al., 1980, p.20). Despite this limitation, the analysis will proceed to demonstrate model selection techniques.

6.5.1 Australian Grand Prix (Albert Park)

Starting with data derived from the Albert Park Race Circuit, the analysis will start by reviewing the plot of the time series to access its stationarity. Up next this very plot is presented alongside with the ACF and the pACF plot of the timeseries.

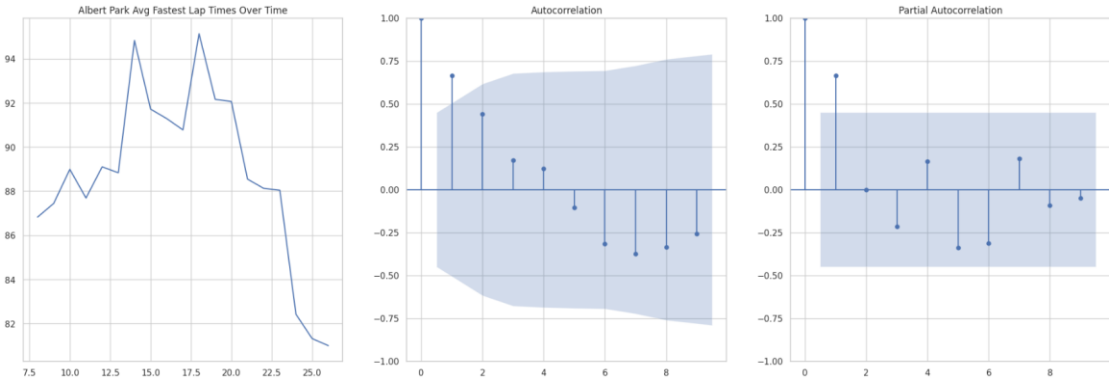

Figure 6.1: Plots of the avg Fastest Laps in Albert Park Timeseries

From the first plot, we observe fluctuations in the fastest lap times over the years, with noticeable peaks and declines. This suggests potential underlying trends and variability in performance. The ACF plot in the middle displays significant autocorrelation at lag 1, gradually decreasing. The PACF plot on the right shows a sharp drop after lag 1, suggesting an autoregressive component in the time series. These observations are depicting a non-stationary timeseries which is proven by the ADF test that gave a p-value of 0.5442, thus not rejecting the null hypothesis of a unit root.

To address the non-stationarity of the time series, the first-order differencing method was applied, followed by an Augmented Dickey-Fuller (ADF) test to assess stationarity. With a p-value of 0.5997, the time series remained non-stationary, necessitating a second-order differencing. This adjustment resulted in an ADF test p-value of 0.0077, leading to the rejection of the null hypothesis and confirming that the time series is now stationary, giving the next plots:

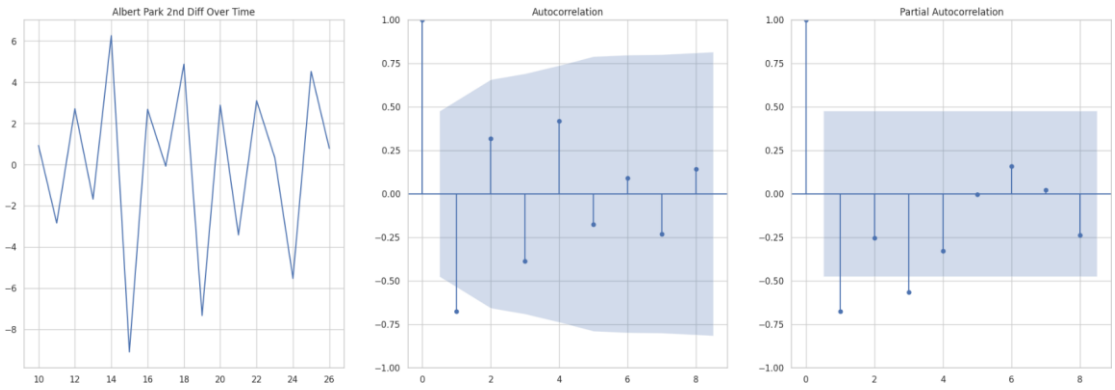

Figure 6.2: Plots of the avg Fastest Laps in Albert Park Timeseries (Stationary)

With the time series now stationary, the next step involves model selection using the ACF and PACF plots. These plots help determine the appropriate ARIMA parameters by identifying significant lags in autocorrelation and partial autocorrelation, guiding the choice of p and q values for the model. With 1 and 2 lags being significant in the Figure's 6.2 plots respectively, the analysis should focus on the ARIMA(1,0,1), ARIMA(2,0,1), ARIMA(1,0,2) and ARIMA(2,0,2) models to access the best fit. Via a relevant code the next model selection criteria emerged:

Model	ARIMA(1,0,1)	ARIMA(2,0,1)	ARIMA(1,0,2)	ARIMA(2,0,2)
AIC	87.085268	88.579378	86.639054	88.637877
BIC	90.418121	92.745444	90.805121	93.637157
AICc	88.931422	91.912711	89.972387	94.092423

Table 6.1: AIC, BIC and AICc for the Albert Park Timeseries

As observed, ARIMA(1,0,1) and ARIMA(1,0,2) yield the lowest values. Although ARIMA(1,0,2) has the lowest AIC, the best model is chosen based on AICc due to the limited number of data points in the time series. Consequently, ARIMA(1,0,1) is selected as the optimal model. With the model finalized, the next set of tests for the residuals was conducted:

Test	a = 5%	
Anderson Darling	***Statistic***	0.5123 < 0.685 (Cr.v.)
Shapiro Wilk	***p-value***	0.1788
Box-Ljung	***p-value***	0.0882

Table 6.2: Residual Tests for the fitted model of the Albert Park Timeseries

Table 6.2 presents the diagnostic test results for the residuals of the fitted model for the Albert Park timeseries. The Shapiro-Wilk test yields a p-value of 0.1788, which is greater than 0.05, indicating that the residuals do not significantly deviate from

normality. Similarly, the Anderson-Darling test statistic (0.5123) is lower than the critical value at the 5% significance level (0.685), further supporting the assumption of normality. Additionally, the Box-Ljung test results in a p-value of 0.0882, suggesting that there is no significant autocorrelation in the residuals. These findings indicate that the residuals exhibit both normality and a lack of autocorrelation, making the model a reliable choice for forecasting. These can also be observed in the residual plots below:

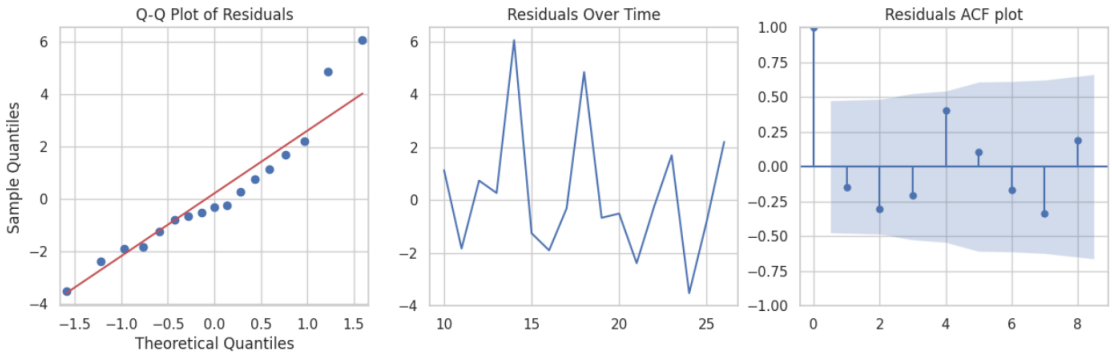

Figure 6.3: Plots of the Residuals of the fitted model (Albert Park TS)

Figure 6.3 presents the residual analysis of the fitted model for the Albert Park timeseries. The normal Q-Q plot on the left shows that the residuals generally follow a normal distribution, with slight deviations in the tails. The middle plot displays the residuals over time, indicating no clear pattern or trend, which supports the assumption of stationarity. The ACF plot on the right confirms the absence of significant autocorrelation, as all lag values remain within the confidence bounds. These visual diagnostics, in combination with the statistical tests from Table 6.2, suggest that the residuals meet the necessary assumptions to continue in forecasting the next 2 seasons. With a relevant code the next plot was structured:

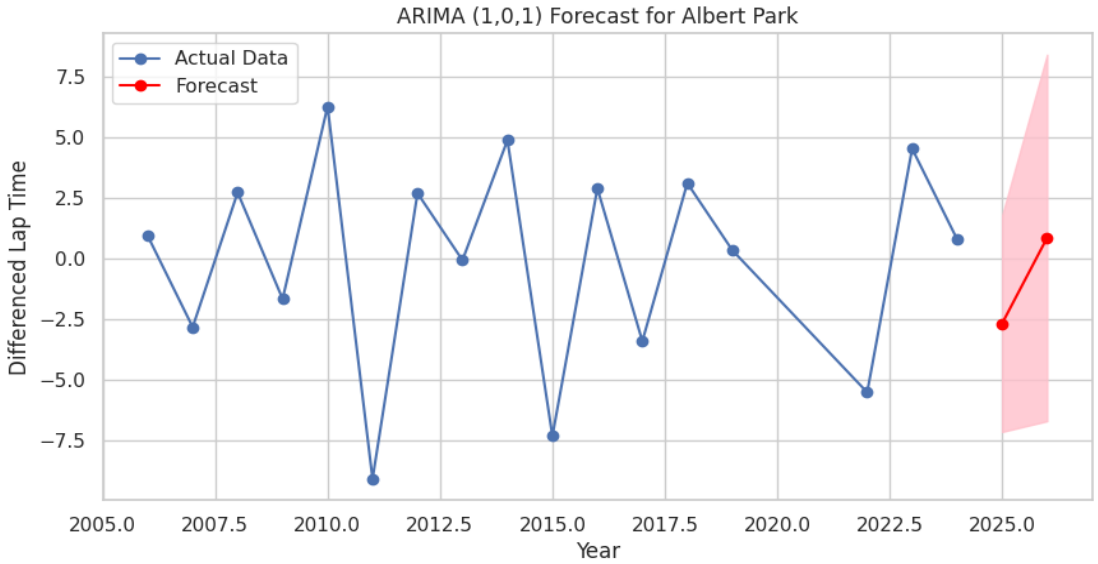

Figure 6.4: Plot of the Albert Park Timeseries with the next 2 seasons (FC)

The forecast for the next two seasons, as shown in Figure 6.4, reveals an upward trend in the differenced time series, indicating a potential shift in the rate of lap time

improvements at Albert Park. Since the series underwent second-order differencing, this does not imply a direct increase in lap times but rather a slowing down—or possible reversal—of the trend of continuous improvement seen in previous years. In other words, while lap times have generally been getting faster, the forecast suggests that these improvements may taper off, stabilizing or even regressing slightly. The widening confidence interval further emphasizes the uncertainty in these predictions, highlighting the potential influence of external factors such as evolving car regulations, weather conditions, or changes in race strategies. As a result, while the ARIMA(1,0,1) model provides valuable insights into future trends, real-world racing dynamics must be considered to fully interpret its implications.

6.5.2 Monaco Grand Prix

To continue with the data from the Monaco Street Circuit, the time series of average fastest lap times has been structured. The following plots present the time series alongside its ACF and PACF, providing an initial insight into its temporal dependencies and potential stationarity:

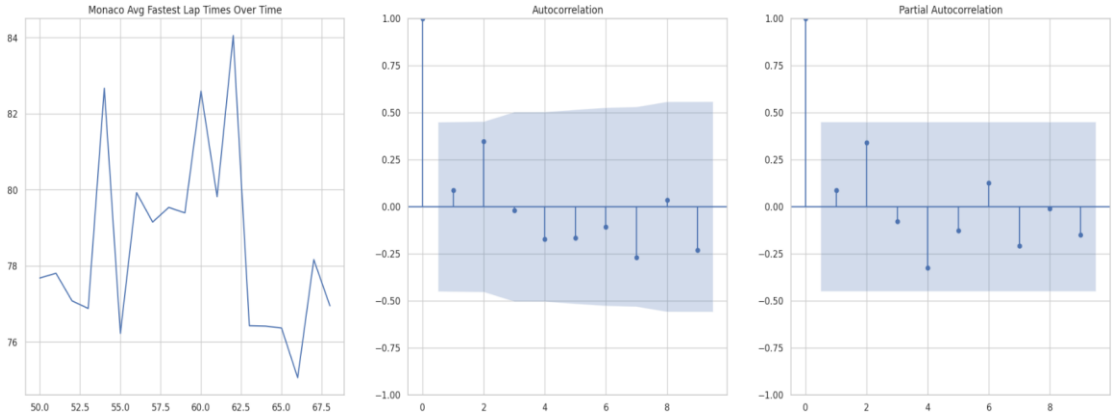

Figure 6.5: Plots of the avg Fastest Laps in Monaco Timeseries

The time series plot on the left illustrates variations in the fastest lap times over the years, marked by distinct peaks and drops, suggesting possible trends and performance fluctuations. The ACF and PACF plots in the middle show no significant autocorrelation, suggesting that the series may follow a white noise model. While white noise processes are inherently stationary, we proceed with the Augmented Dickey-Fuller (ADF) test to assess whether first-order differencing could enhance the forecasting potential of the time series. With a p-value of 0.0052, the ADF test is closely rejecting the null hypothesis of a unit root, indicating potential stationarity. However, to further explore any hidden patterns and improve predictive power, we will apply first-order differencing and reassess the results.

Applying first-order differencing, the Augmented Dickey-Fuller (ADF) test yields a p-value approaching zero (approximately 3.67×10^{-13}), strongly confirming the stationarity of the time series. With this result, we can now proceed to analyzing the subsequent plots:

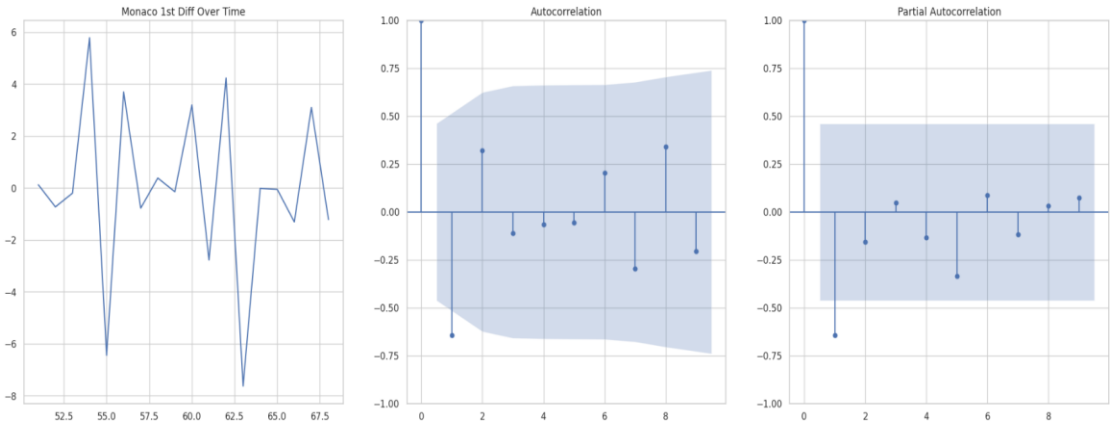

Figure 6.6: Plots of the avg Fastest Laps in Monaco Timeseries (1st Diff)

Figure's 6.6 plots show that, first-order differencing successfully extracted the forecasting power necessary to proceed with the analysis of upcoming seasons. With the first lags now showing significance in both the ACF and pACF plots, the model selection process will consider ARIMA(1,0,0), ARIMA(0,0,1), and ARIMA(1,0,1) to determine the best fit. Using relevant code, the next step involves calculating model selection criteria to finalize the optimal ARIMA configuration:

Model	ARIMA(1,0,0)	ARIMA(0,0,1)	ARIMA(1,0,1)
AIC	90.962887	91.759557	92.643365
BIC	93.634003	94.430672	96.204852
AICc	91.762887	92.559557	94.357651

Table 6.3: AIC, BIC and AICc for the Monaco Timeseries

As observed, the ARIMA(1,0,0) model yielded the lowest values among the considered options, making it the best fit based on the selection criteria. With this model chosen, the next step involves conducting residual diagnostics to ensure the validity of the fit, assessing normality, independence, and homoscedasticity through statistical tests and visual inspections.

Test	a = 5%	
Anderson Darling	***Statistic***	0.4388 < 0.687 (Cr.v.)
Shapiro Wilk	***p-value***	0.5130
Box-Ljung	***p-value***	0.5557

Table 6.4: Residual Tests for the fitted model of the Monaco Timeseries

Table 6.4 summarizes the diagnostic tests for the residuals of the ARIMA(1,0,0) model applied to the Monaco time series. The Shapiro-Wilk test produces a p-value of 0.5130, which exceeds the 5% significance level (0.05), indicating that the residuals follow a normal distribution. Likewise, the Anderson-Darling test statistic (0.4388) is below the critical threshold of 0.687, further supporting the normality assumption. Box-Ljung test returns a p-value of 0.5557, suggesting no significant autocorrelation in the residuals. These results confirm that the residuals exhibit both normality and independence, reinforcing the suitability of the model for forecasting. The subsequent residual plots visually support these findings.

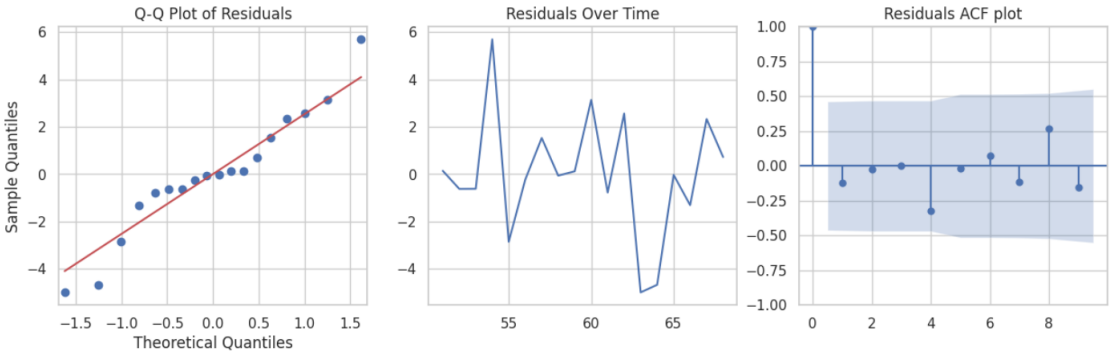

Figure 6.7: Plots of the Residuals of the fitted model (Monaco TS)

Figure 6.7 illustrates the residual analysis for the fitted model of the Monaco time series. The Q-Q plot on the left indicates that the residuals largely conform to a normal distribution, with slight deviations. The middle plot, which depicts residuals over time, reveals no discernible patterns or trends, reinforcing the assumption of stationarity. Meanwhile, the ACF plot on the right demonstrates the absence of significant autocorrelation, as all lag values remain well within the confidence limits. These visual diagnostics, combined with the statistical results from Table 6.4, confirm that the residuals satisfy the necessary assumptions for reliable forecasting. Using the appropriate code, the next forecast plot was generated:

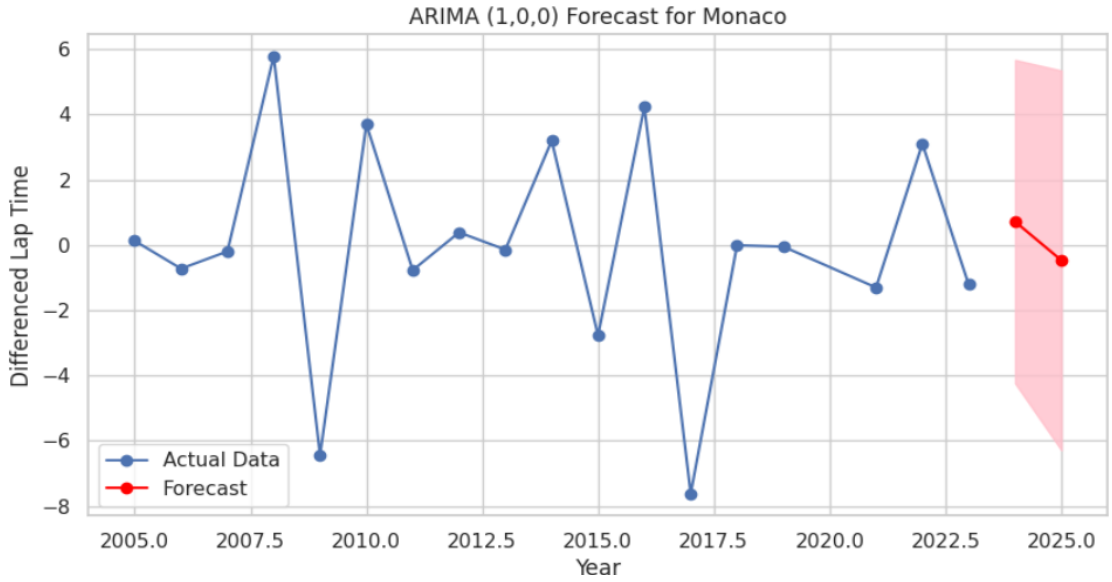

Figure 6.8: Plot of the Monaco Timeseries with the next 2 seasons (FC)

The forecast for the Monaco timeseries, shown in Figure 6.8, presents the first-differenced values of the fastest lap times. The blue line represents the historical changes in lap times, while the red points and shaded area depict the predicted changes and their confidence intervals. The forecast shows a downward trend, which does not indicate an absolute increase or decrease in lap times but rather suggests that the rate of change is negative. This implies that lap times are expected to improve (decrease) over the next two seasons. The confidence interval widens as the forecast extends further, reflecting increased uncertainty in long-term predictions. This result aligns with the fitted ARIMA(1,0,0) model, supporting its suitability for capturing the dynamics of the Monaco timeseries.

6.5.3 Conclusions from the 2 Timeseries

The analysis of the Albert Park and Monaco timeseries provided valuable insights into the trends and forecasting of lap time performance in two distinct racing environments: a permanent race circuit and a street race. While both timeseries required differencing to achieve stationarity—second-order differencing for Albert Park and first-order differencing for Monaco (mainly for illustrative purposes)—the underlying patterns and forecast behaviors showcased key differences between the two, further complementing the insights presented in Chapter 3, specifically Section 3.4, which explores the progression of fastest lap times by circuit.

Albert Park, as a dedicated race circuit, exhibited more structured fluctuations in lap times, with its second-order differencing indicating a more complex trend and variability in performance over time. The forecast for this circuit suggested an upward trend in the differenced values, implying an increasing trend in lap times for the upcoming seasons, potentially due to regulatory changes or the circuit approaching a performance ceiling, where further lap time reductions become increasingly difficult. It should be noted that FIA regulations prioritize driver safety, leading to design constraints that may influence lap times.

In contrast, the Monaco street circuit, known for its tight corners and unique layout, displayed a more immediate stationarity with first-order differencing. The forecasted values indicated a downward trend in the differenced series, suggesting an improvement in lap times over the next seasons. This aligns with the nature of street circuits, where driver adaptation, team strategies, and ongoing improvements in vehicle performance might contribute to continuous lap time reductions. This is largely due to the emphasis on cornering performance rather than straight-line speed on street circuits.

Despite these differences, both stationary timeseries -after respective differentiations- demonstrated strong model fits, with ARIMA(1,0,1) for Albert Park and ARIMA(1,0,0) for Monaco emerging as the best-performing models based on AIC and diagnostic tests. The residual analysis confirmed the normality and lack of autocorrelation in both models, validating their use for forecasting. Ultimately, while each circuit presents its own challenges and trends, both analyses highlight the evolving nature of race performance and the importance of statistical modeling in understanding and predicting future outcomes.

6.6 Comparing with up-to-date fastest lap times

By the time of this analysis, the 2025 Australian Grand Prix and the 2024 Monaco Grand Prix had taken place. As illustrated in Figure 6.4, the fastest lap at Albert Park Circuit was indeed slower than in 2024, with a gap of approximately 2.3 seconds, primarily due to wet conditions, which frequently affect the race. Conversely, the fastest lap at the 2024 Monaco Grand Prix improved by approximately 1.5 seconds compared to 2023 (as shown in Figure 6.8). While it did not break the lap record, it reaffirmed the downward trend in lap times. These observations demonstrate that the ARIMA models effectively captured the time series patterns, providing valuable insights into forecasting future race performances.

Chapter 7: Machine Learning

This chapter explores the application of machine learning techniques in race performance analysis, integrating dimensionality reduction, classification, and clustering to uncover patterns and enhance predictive accuracy. Principal Component Analysis (PCA) is first employed to reduce dimensionality while preserving key features, improving computational efficiency and interpretability. For classification, Random Forest (RF) and Support Vector Machine (SVM) methods are implemented to predict race outcomes , using the *Race_Outcome* variable as target and evaluating the model's performance through accuracy, cross-validation, and confusion matrices. Moving to unsupervised learning, we apply K-Means and Hierarchical (Agglomerative) Clustering to identify underlying groupings within the data, assessing their effectiveness using Silhouette Score, Adjusted Rand Index (to compare against the *Race_Outcome* variable), and the Calinski-Harabasz Index. These combined approaches provide a comprehensive analysis of race dynamics, offering insights into driver performance and competitive strategies. The Python code can be found in the attached appendix at the end of this thesis (Appendix 1: Implementation Code of the Present Study).

7.1 Data Mining

Data mining refers to the process of discovering meaningful patterns and knowledge from large datasets using clustering or classification algorithms, along with principles from statistics, artificial intelligence, machine learning, and database systems. The data sources can include databases, data warehouses, the internet, and other information repositories, as well as dynamically transmitted data. The goal of data mining is to extract information and identify patterns in a structured, human-understandable way, facilitating decision-making for further analysis, machine learning applications, and predictive analytics (Han et al., 2012).

Steps in the Data Mining Process

The data used in machine learning algorithms must be of high quality to achieve the desired objectives. Therefore, preprocessing is a crucial step in the data mining process. The key steps involved in this process are outlined below:

- **Data Cleaning:** Data cleaning is the first step in data mining. It is essential because if raw, unclean data is used, it may cause confusion in subsequent processes and lead to inaccurate results. This step involves removing noise and handling missing values in the dataset.
- **Data Integration:** In this step, multiple heterogeneous data sources—such as databases, data cubes, or files—are combined for analysis. This process is known as data integration.

- **Data Reduction/Selection:** This technique is used to obtain the most relevant data for analysis. The goal is to reduce the size of the dataset while preserving its integrity. Specifically, it involves dimensionality reduction, feature selection, and data compression to enhance efficiency.
- **Data Transformation:** Data is transformed into a suitable format for mining by applying strategies such as smoothing, aggregation, normalization, and discretization.
- **Data Mining:** At this stage, classification and clustering techniques are applied to uncover meaningful patterns and knowledge within the dataset.
- **Pattern Evaluation:** The most significant patterns representing useful knowledge are identified based on interest measures.
- **Knowledge Representation:** Finally, visualization and data representation techniques are used to present the extracted knowledge in an interpretable manner (Han et al., 2012).

7.2 Machine Learning

As a scientific field, machine learning focuses on developing algorithms that can learn from data and make predictions based on it. It is categorized into four main types, each serving different analytical purposes:

- **Supervised Machine Learning**: Algorithms are trained on labeled data, where input-output relationships are explicitly provided, enabling them to make accurate predictions on new data.
- **Unsupervised Machine Learning**: These algorithms analyze unlabeled data to identify hidden patterns and structures, commonly used for clustering and anomaly detection.
- **Semi-Supervised Machine Learning**: A hybrid approach that combines labeled and unlabeled data, leveraging the small amount of labeled data to improve learning efficiency.
- **Reinforcement Learning**: A learning paradigm where an agent interacts with an environment and learns optimal actions through rewards and penalties, widely used in decision-making applications.

7.2.1 Supervised Machine Learning

Being a fundamental category of machine learning, a supervised learning algorithm is trained using labeled data, meaning that each input is paired with a known output. This approach enables the model to learn the relationship between input features and target labels, allowing it to make predictions on new, unseen data. Supervised learning is broadly divided into classification and regression tasks.

- Classification involves assigning inputs to discrete predefined categories based on observed patterns. Algorithms such as Support Vector Machines (SVMs), Random Forests, k-Nearest Neighbors (k-NN), and Decision Trees are commonly used for classification problems.
- Regression, on the other hand, predicts continuous values, aiming to model the relationship between dependent and independent variables. Common regression techniques include Linear Regression, Ridge Regression, and Polynomial Regression.

Supervised learning is widely used in applications such as fraud detection, medical diagnosis, speech recognition, and financial forecasting, where past labeled data informs future decision-making (James et al., 2023).

7.2.2 Unsupervised Machine Learning

Unsupervised learning differs from supervised learning in that the model is given data without labeled outputs and must identify inherent structures or patterns independently. As before, unsupervised learning is primarily used for clustering and dimensionality reduction.

- Clustering involves grouping similar data points together based on shared characteristics. Algorithms such as K-Means, Hierarchical Clustering, and DBSCAN are commonly used to discover natural patterns in data.
- Dimensionality Reduction techniques, such as Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE), help reduce the number of features while retaining essential information, improving computational efficiency and visualization.

Unsupervised learning is widely applied in customer segmentation, anomaly detection, gene expression analysis, and recommendation systems, where structured labels are unavailable (James et al., 2023).

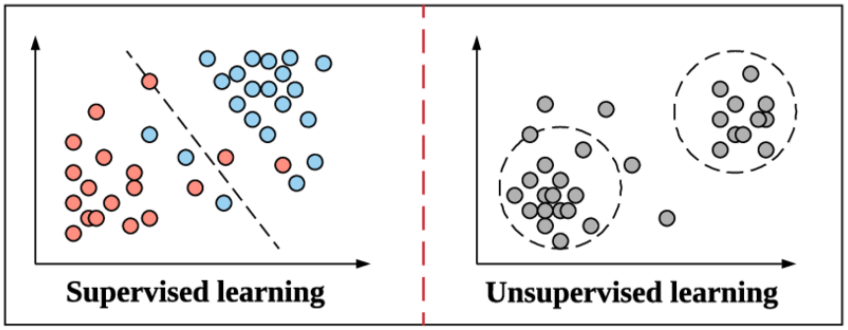

Figure 7.1: Supervised vs Unsupervised Learning

(Source: Qian et al., 2019)

7.2.3 Semi-supervised Machine Learning

Semi-supervised learning is a machine learning approach that combines a small amount of labeled data with a larger set of unlabeled data during training. This method is particularly useful when labeling data is expensive or time-consuming, allowing models to learn from the inherent structure of the data.

In semi-supervised learning, the model leverages the labeled data to guide its learning process, while also utilizing the unlabeled data to capture the underlying patterns and structures. This approach bridges the gap between supervised learning, which relies solely on labeled data, and unsupervised learning, which works with unlabeled data. This method mainly applies when manually labeling is impractical, so semi-supervised algorithms are employed to improve classification accuracy.

However, semi-supervised learning also has its challenges. The quality of the unlabeled data and the assumptions made about its distribution can significantly impact the model's performance. Therefore, careful consideration is required when selecting and implementing semi-supervised learning techniques. (https://www.geeksforgeeks.org/ml-semi-supervised-learning/)

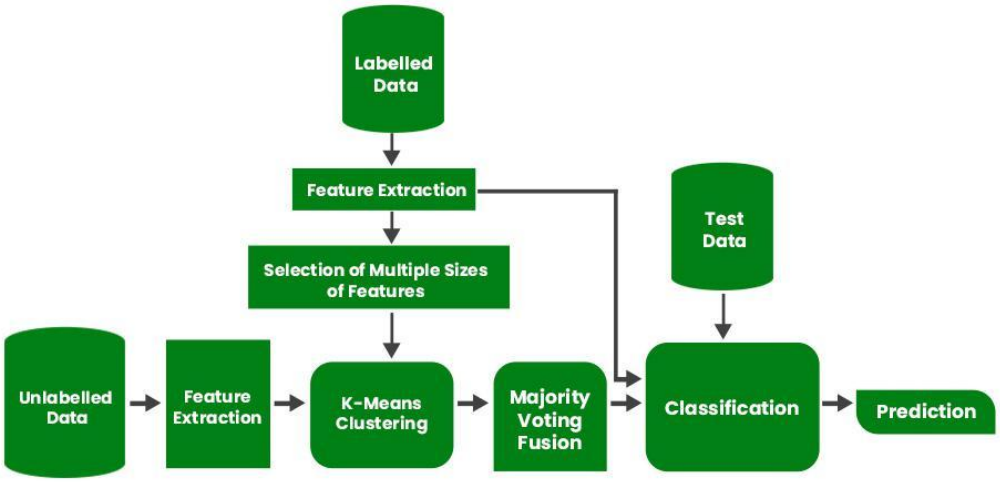

Figure 7.2: Semi-supervised Learning Flow Chart

(Source: https://www.geeksforgeeks.org/ml-semi-supervised-learning/)

7.2.4 Reinforcement Machine Learning

Reinforcement learning is a framework in which an agent interacts with an environment to learn optimal behaviors through trial and error. The agent takes actions, receives feedback in the form of rewards, and adjusts its strategy to maximize cumulative rewards over time. Unlike supervised learning, where correct outputs are provided, reinforcement learning relies on the agent exploring different actions and learning from the consequences.

A key aspect of reinforcement learning is the trade-off between exploration and exploitation. Exploration allows the agent to gather new information about the environment, while exploitation enables it to use known information to maximize rewards. This balance is crucial for achieving optimal performance, especially in dynamic and uncertain environments.

A central concept in reinforcement learning is the Markov Decision Process (MDP), which provides a mathematical framework for decision-making. MDPs define an environment in terms of states, actions, transition probabilities, and reward functions, enabling the agent to make sequential decisions effectively. Policy-based and value-based methods, such as Q-learning and policy gradient approaches, are commonly used to solve reinforcement learning problems (Sutton, 2018).

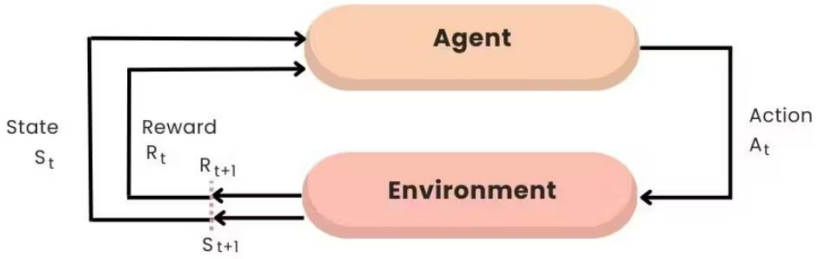

Figure 7.3: Reinforcement Learning Cycle

(Source: https://www.turing.com/kb/reinforcement-learning-algorithms-types-examples)

7.3 Feature Selection

As a first step for applying data mining algorithms, it is essential to reduce the number of variables, particularly when the dataset contains a large number of them. In this study, several variables have been recorded, and consequently, it is necessary to select the most relevant ones. This process involves choosing a subset of variables/features from a broader set to construct a predictive model. It is typically performed to enhance the model's accuracy, prevent overfitting, and expedite the training process. There are two primary categories of methods used for selecting the most appropriate variables: unsupervised methods and supervised methods. These categories align with the previously mentioned types of machine learning, sharing similar characteristics. Specifically, in supervised methods, the output category is known, and the objective is to select variables that improve the model's performance. In contrast, unsupervised methods do not rely on the output category for feature selection. More specifically, supervised methods can be further divided into three subcategories:

- **Filter Method:** This method involves selecting variables based on their relationship with the output category or their correlation with it. The correlation is used to determine whether the variables are positively or negatively associated with the output variable. Common techniques for assessing correlation include the Chi-Square Test, Fisher's Score, and others.

- **Wrapper Method:** In this category, the data is initially divided into subsets, and a model is trained using these subsets. Based on the model's output, features are removed, and the model is retrained. This method employs a "greedy" approach to generate subsets and evaluates the accuracy of all possible combinations of features. Examples of such methods include Forward Selection, Backward Elimination, and others.

- **Embedded Method:** This method combines elements of the previous two to form the optimal subset. Examples of such methods include Lasso and Ridge regression (Kamitsis, 2023).

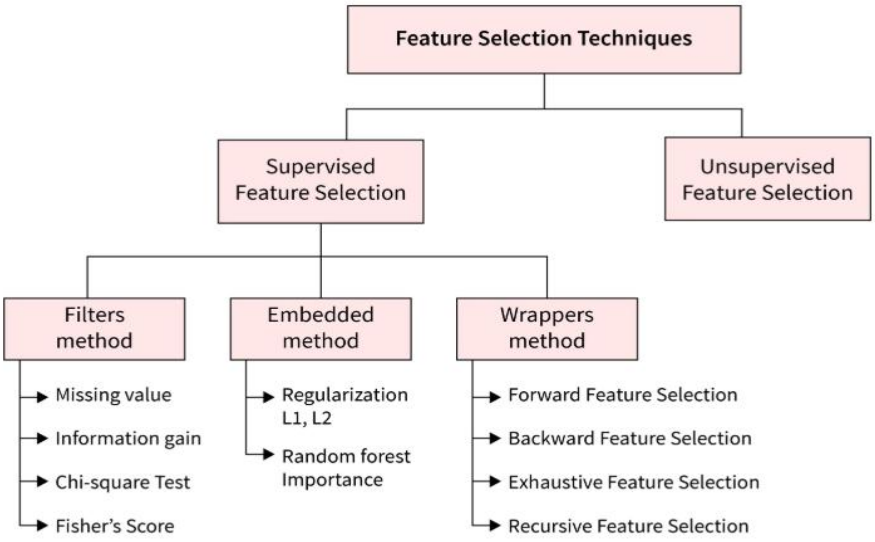

Figure 7.4: Feature Selection Methods/Techniques

(Source: https://www.scaler.com/topics/machine-learning/feature-selection-in-machine-learning/)

7.4 Principal Component Analysis

Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction that allows for the transformation of a large set of variables into a smaller one while retaining as much information as possible. The goal of PCA is to identify the most significant features, or principal components, in a dataset and to reduce the number of dimensions in such a way that the variance of the data is preserved. PCA is frequently applied to improve the computational efficiency of algorithms and to aid in the visualization of high-dimensional data.

The process of PCA involves calculating the eigenvectors and eigenvalues of the data's covariance matrix. These eigenvectors correspond to the directions of maximum variance in the data, while the eigenvalues represent the magnitude of the variance in those directions. By selecting the top eigenvectors (principal components) based on their associated eigenvalues, PCA reduces the dimensionality of the data

without significant loss of information. This transformation allows for simpler analysis, improved accuracy in subsequent modeling, and reduced noise.

Jolliffe (2002) explains that PCA is especially valuable in fields like image processing, gene expression analysis, and other domains where large, high-dimensional datasets are common. It can enhance the performance of machine learning algorithms by removing redundancy and improving the interpretability of data.

7.5 Classification

Classification is a fundamental technique in machine learning and involves predicting the class or category of data. Classification methods are algorithms that use labeled data to discover a decision boundary that can be used to classify new, unlabeled data into one or more predefined classes. For this reason, classification methods require the division of data into training and testing sets. In this way, part of the data is used to train the model and then tested on the remaining portion (Chrysis, 2024). In the present study, the data were split in an 80% to 20% ratio between the training and testing sets. The goal is to train a model that can accurately predict the class of new observations. Overall, classification algorithms are powerful tools for solving a variety of problems.

Random Forests

A widely known machine learning algorithm used for classification is the Random Forest. This process was first introduced by Leo Breiman and Adele Cutler in 2001. It is a technique aimed at improving the accuracy of predictions by combining multiple decision trees and reducing overfitting. Figure 7.5 illustrates the algorithm's process for classifying a dataset.

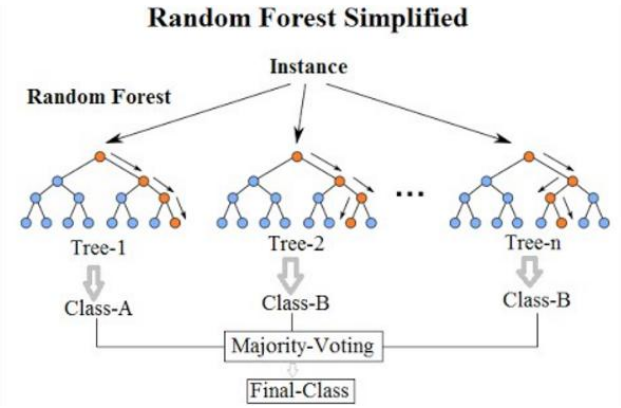

Figure 7.5: Random Forest Classifier

(Source: https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d)

In a Random Forest, each decision tree is trained on a random subset of the input data. Initially, the process creates multiple new training sets, and let's assume the number of new sets is K. Based on each set K_i from the K sets, a tree is constructed using a base classifier. The difference from other methods is that not all the available features are used when adding new nodes; only a subset of them is utilized. The number of features used is fixed during the execution of the algorithm. The final prediction is primarily determined through a majority voting mechanism, where the most frequently predicted class across all decision trees is selected as the final output. (Breiman, 2001)

Support Vector Machines

Another machine learning algorithm used for the classification process in this work is Support Vector Machines (SVM), which can also be applied to regression tasks. This algorithm is built around the idea of finding a hyperplane that separates elements of two classes in a higher-dimensional space. Its goal is to determine the hyperplane that maximizes the margin, i.e., the distance between the hyperplane and the nearest points from each class. These nearest points are known as support vectors (Hastie et al., 2009).

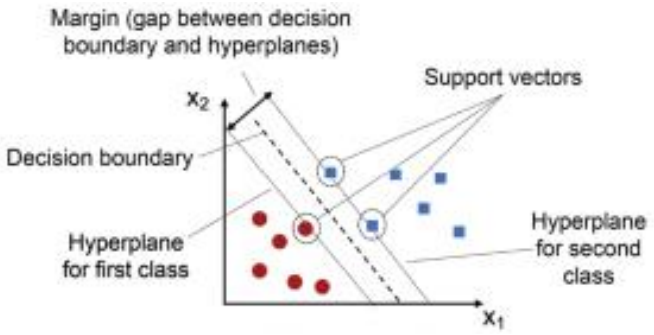

Figure 7.6: Support Vector Machine Example

(Source: https://vitalflux.com/classification-model-svm-classifier-python-example/)

Initially, this algorithm transforms the data into a higher-dimensional space where it is likely that the data becomes linearly separable. To achieve this transformation, a kernel function is used. The kernel function maps the data into the higher-dimensional space without directly computing the coordinates of the data in the new space. The choice of kernel is crucial as it determines the shape of the decision boundary. Below are some common kernel functions:

- **Polynomial Kernel Function:** This transforms the data points by using the dot product and mapping the data into an "n-dimensional" space, where the transformation is either a square product or a higher-order product. Therefore, the data is represented in a higher-dimensional space using the newly transformed points. The formula used by this kernel function is:

$$k(x, y) = (ax^T y + c)^d$$

where d is the degree of the polynomial function, which is a parameter to be tuned, and α, c control the influence of higher-order terms.

- **Radial Basis Function (RBF) Kernel:** This kernel function initially transforms the data by representing it in an infinite-dimensional space and then classifies it using a weighted nearest neighbor approach. The RBF kernel can be Gaussian or Laplace, depending on a hyperparameter known as gamma, which controls the width of the Gaussian function and must be tuned. The formula for the RBF kernel is:

$$k(x, y) = exp(\gamma \|x - y\|^2)$$

- **Linear Kernel:** This kernel simply represents the data points using a linear relationship. Specifically, it computes the dot product between two input vectors in their original space. The formula is:

$$k(x, y) = x^T y$$

(Schölkopf & Smola, 2002)

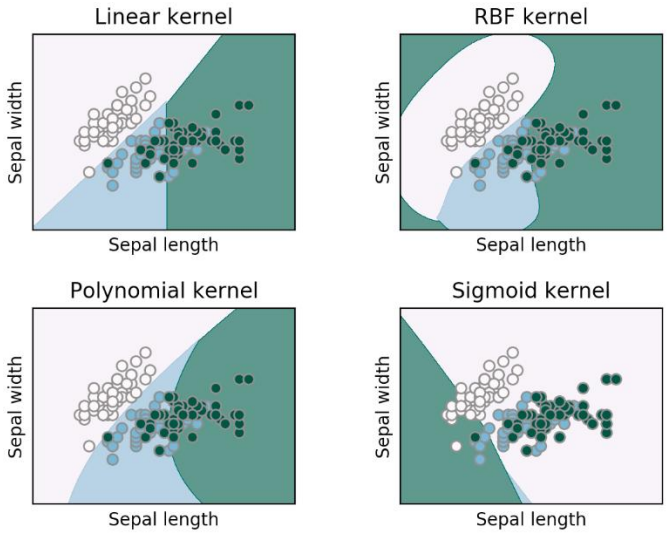

Figure 7.7: SVM Kernel Examples

(Source: Premanand, 2023)

7.6 Evaluation Techniques for Classification Methods

Evaluation of classification methods is a crucial step in assessing the performance of a model. As we have discussed in Chapter 5, Section 5.8, common evaluation metrics include accuracy, precision, recall, F1 score, and the confusion matrix. Accuracy measures the percentage of correct predictions, but it may not be sufficient for imbalanced data. Precision focuses on the percentage of true positive predictions out of all positive predictions, while recall measures how many actual positive cases are correctly identified. The F1 score combines precision and recall into a single metric, providing a balance between the two. Finally, the confusion matrix helps visualize the model's performance by showing true positives, true negatives, false positives, and false negatives, providing deeper insight into the model's strengths and weaknesses. These metrics are essential for evaluating and improving the performance of classification models.

7.7 Clustering

Cluster analysis (clustering) is one of the fundamental techniques in data mining and machine learning, which involves grouping a set of objects into clusters so that the objects within each cluster are more similar to each other, i.e., more homogeneous, compared to those in other clusters. Clustering is used for various purposes, such as identifying natural groupings of similar objects, detecting outliers or anomalies in data, and reducing dimensionality in large datasets. The choice of algorithm depends on the nature of the data and the objectives of the analysis. (Tan et al., 2018)

K-Means

The K-means algorithm is a popular clustering algorithm used in data mining and machine learning. The main idea behind K-means is to partition a set of n data points into k clusters, where k is a user-defined parameter. The algorithm follows the steps outlined below:

1. **Initialization:** The user selects k initial centroids randomly from the data set.
2. **Assignment**: Each data point is assigned to the cluster whose centroid is closest, based on a distance measure such as Euclidean distance.
3. **Update:** The centroids of the k clusters are updated by computing the mean of all the points assigned to each cluster.
4. **Iteration:** Steps 2 and 3 are repeated until convergence, meaning the cluster assignments no longer change.

While the K-means algorithm is relatively simple and efficient, it is sensitive to the initial placement of centroids and may converge to a local minimum. For this reason, it is often run multiple times with different initializations to increase the chances of finding a good solution.

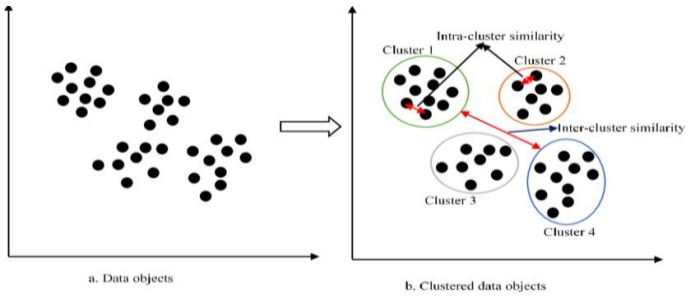

Figure 7.8: K-Means Clusters

(Source: Ezugwu et al., 2021)

The algorithm also has some limitations. One limitation is that it assumes the clusters are spherical and have equal variance. This assumption may lead to poor performance when the data points are not well separated or when the clusters have irregular shapes. Another limitation is that the user must specify the number of clusters k beforehand, which may not always be known. To address this, techniques such as the elbow method and silhouette analysis are commonly used to help determine an appropriate value for k (Tan et al., 2018).

Elbow Method

The Elbow Method is a technique used to determine the optimal number of clusters in K-means clustering. The core idea behind the Elbow Method is to plot the within-cluster sum of squares (WCSS) as a function of the number of clusters k and identify an "elbow" or bend in the curve. The WCSS measures the sum of squared distances between each data point and the centroid of the cluster it is assigned to, given by the formula:

$$WCSS = \sum_{j} \sum_{i} || x_i - c_j ||^2$$

where x_i represents the *i-th* observation, c_j is the centroid of the *j-th* cluster, and ||.|| denotes the Euclidean distance.

To apply the Elbow Method, the following steps are performed:

1. The K-means algorithm is executed for a range of k values (e.g., from 1 to 10).
2. For each value of k, the WCSS is calculated.
3. The WCSS is plotted as a function of k.
4. The elbow point is identified, which is where the WCSS starts to level off or decrease at a slower rate.

The intuition behind the Elbow Method is that as k increases, the WCSS should decrease, since smaller clusters are more compact. However, beyond a certain k, the improvement in WCSS becomes marginal, as clusters become too small to provide meaningful separation. The Elbow Method seeks to identify this optimal number of clusters (Tan et al., 2018).

It is important to note that the Elbow Method is not always foolproof and may not provide a clear indication of the best k. In practice, it is often used as an initial guideline, while other methods, such as Silhouette Analysis, can be used for further validation. Figure 7.8 shows that a good choice for the number of clusters is k = 2, as the WCSS curve shows a noticeable drop up to this point, after which the slope decreases significantly for higher values of k.

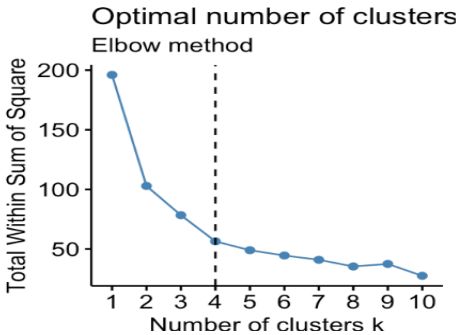

Figure 7.9: Elbow Plot Example

(Source: https://www.datanovia.com/en/lessons/determining-the-optimal-number-of-clusters-3-must-know-methods/#google_vignette)

Hierarchical Agglomerative Clustering (AGNES)

Hierarchical Agglomerative Clustering (HAC), also known as Agglomerative Nesting (AGNES), is a clustering technique that begins with each data point as its own individual cluster and iteratively merges the closest pair of clusters until all points belong to a single cluster. HAC offers several advantages, including simplicity and the ability to handle different types of data. However, it can be computationally expensive for large datasets, and its results can be sensitive to the choice of distance metric and linkage method, which are discussed below.

This algorithm is often represented using a dendrogram, a tree-like diagram that illustrates the sequence and distances of cluster mergers during the process.

Steps of Hierarchical Agglomerative Clustering:

1. Initialize each data point as an individual cluster.
2. Compute the distance matrix for all pairs of clusters.
3. Identify the closest pair of clusters based on a chosen distance metric, such as Euclidean distance.
4. Merge the two closest clusters into a single cluster.
5. Update the distance matrix to reflect the new distances between the merged cluster and the remaining clusters.
6. Repeat steps 3-5 until all data points belong to a single cluster or until a stopping criterion is met (e.g., a desired number of clusters is reached).

Linkage Methods for Cluster Distance Calculation:

Different methods can be used to determine the distance between clusters:

- Single Linkage (Minimum Linkage): Defines the distance between two clusters as the shortest distance between any two points within them. This method tends to create long, loose clusters.
- Complete Linkage (Maximum Linkage): Defines the distance between two clusters as the longest distance between any two points within them. It generally produces compact clusters.
- Average Linkage (Mean Linkage): Computes the distance as the average of all pairwise distances between points in the two clusters.
- Centroid Linkage: Calculates the centroids of both clusters and uses the distance between these centroids.
- Ward's Method: Minimizes the total variance within clusters by merging the pair of clusters that result in the smallest increase in total within-cluster variance. This method tends to create compact, spherical clusters, but it can be sensitive to outliers and may struggle with non-spherical clusters or mixed data types. Ward's Method is going to be the choice for our analysis, as it generally produces well-separated and compact clusters (Tan et al., 2018).

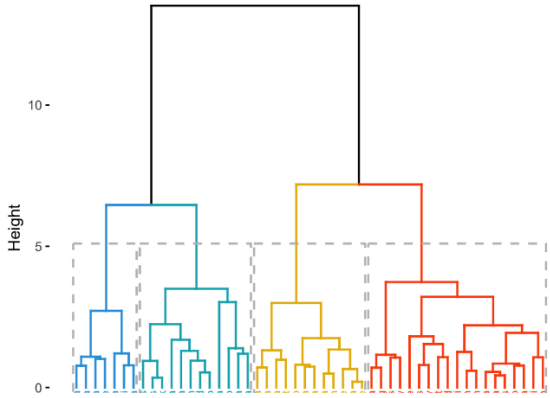

Figure 7.10: Cluster Dendrogram HAC

(Source: https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/)

7.8 Evaluation Techniques for Clustering Methods

There are several methods for evaluating clustering quality, categorized into internal, external, and relative measures. The choice of an evaluation metric depends on the specific application and the clustering objectives.

- Internal measures assess clustering quality based solely on the data and the clustering algorithm, without using external information or class labels.
- External measures evaluate clustering performance by comparing the results with an external reference, such as predefined class labels.
- Relative measures compare the results of different clustering algorithms or different parameter settings for the same algorithm.

In this study, we use the Silhouette Coefficient, the Adjusted Rand Index (ARI), and the Calinski-Harabasz Index for evaluating clustering results.

Silhouette Coefficient

One of the most widely used methods for assessing clustering performance is the Silhouette Coefficient (SC). This metric is calculated for each data point and is based on two key distance measures:

- $a_{(i)}$**:**The average distance between a point *i* and all other points within the same cluster.
- $b_{(i)}$**:** The average distance between a point *i* and all points in the closest neighboring cluster.

The Silhouette Coefficient for a data point is computed as:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

The overall Silhouette Coefficient is the mean of $s_{(i)}$ values across all data points. The coefficient ranges from **-1 to 1**:

- Values close to **1** indicate well-clustered data.
- Values near **0** suggest overlapping clusters.
- Values close to **-1** indicate poor clustering.

(Tan et al., 2018)

Adjusted Rand Index (ARI)

The Adjusted Rand Index (ARI) measures the agreement between two clusterings of the same dataset. It is a modification of the Rand Index (RI), which quantifies the proportion of point pairs assigned to the same or different clusters in both partitions. Unlike RI, ARI adjusts for chance agreement, making it a more robust metric.

ARI is defined as:

$$ARI = \frac{RI - Expected\_RI}{max\{RI\} - Expected\_RI}$$

where:

- *RI* is the Rand Index,
- *Expected_RI* is the expected value of *RI* under a random clustering assumption,
- *max{RI}* is the maximum possible *RI* value.

The ARI score ranges from -1 to 1:

- 1: Perfect agreement between the two clusterings.
- 0: Random clustering assignment.
- -1: Complete disagreement.

The ARI has several advantages over other clustering evaluation metrics. First, it is not affected by the number of clusters or the dataset size, unlike some other measures. Second, it is symmetric, meaning it does not favor one clustering over another. Third, it accounts for the probability of agreement by chance, making it more robust to noise and random fluctuations in the data (Tan et al., 2018).

Calinski-Harabasz Index

The Calinski-Harabasz Index (CH Index) is used to compare different clustering solutions for the same dataset. It measures cluster cohesion and separation based on the distances between cluster centroids and the data points within them. Higher values indicate better clustering results. The formula for CH Index is:

$$CH = \frac{\sum_{k=1}^{K} n_k \,||\, c_k - c \,||^2/(k-1)}{\sum_{k=1}^{K}\sum_{i=1}^{n_k} ||\, d_i - c_k \,||^2/(N-K)}$$

where:

- K is the number of clusters,
- n_k is the number of points in cluster k,
- c_k is the centroid of cluster k,
- c is the centroid of the entire dataset,
- d_i represents a data point,
- N is the total number of data points.

A higher CH Index suggests a more distinct and well-separated clustering structure (Tan et al., 2018).

7.9 Application of Machine Learning Methods

In this section, the application of classification and clustering techniques is conducted. The dataset is sourced directly from the Encoding section of Chapter 5 and is utilized for the purposes of this chapter. Prior to implementing these methods, the variable *Race_Outcome* was retained and designated as the target variable (y), with the remaining dataset undergoing Scaling to enhance uniformity and improve model performance excluding the target variable. Additionally, the variables *driver_name* and *positionOrder* were removed to eliminate potential biases in the selection and creation of Principal Component Analysis (PCA) components.

7.9.1 Dimensionality Reduction with PCA

With the data prepared for Principal Component Analysis (PCA), the components were generated using the appropriate code implementation. To determine the optimal number of principal components to retain, a Cumulative Explained Variance Plot was constructed, providing a visual representation to guide the selection process:

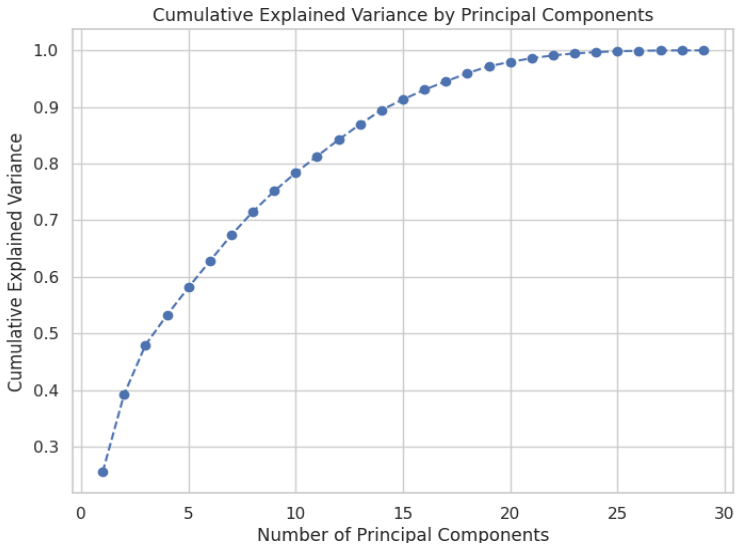

Figure 7.11: Cumulative Explained Variance from PCA

The Cumulative Explained Variance Plot in Figure 7.9 illustrates the proportion of total variance captured by the principal components. As seen in the graph, the explained variance increases rapidly with the first few components before gradually leveling off. To balance dimensionality reduction with information retention, we selected the first 10 principal components, which together explain approximately 79% of the total variance. This threshold ensures that a significant portion of the dataset's variability is preserved while reducing computational complexity and mitigating potential noise from less informative components. Table 7.1 shows the top 3 contributing features for each principal component:

Principal Component	Most Contributing Features
PC1	'qualifying_position', 'race_starts', 'grid_position'
PC2	'age', 'years_experience', 'outside_10_positions'
PC3	'q1', 'fastest_lap_time', 'laps'
PC4	'constructor_name', 'top_constructor', 'cumulative_season_driver_points'
PC5	'round', 'driver_nationality', 'constructor_name'
PC6	'total_duration_ms', 'total_pit_stops', 'constructor_nationality'
PC7	'total_pit_stops', 'year', 'total_duration_ms'
PC8	'valid_time_set_in_q1', 'round', 'race_status'
PC9	'round', 'valid_time_set_in_q1', 'driver_nationality'
PC10	'race_status', 'circuit_name', 'round'

Table 7.1: Most Contributing Features of the Principal Components

7.9.2 Classification with Random Forest

In this section, the Random Forest algorithm is implemented as a classification technique to predict the Race_Outcome variable. Using the 10 principal components selected from PCA, the dataset was split into training and testing sets to evaluate the model's performance. The classification process involved hyperparameter tuning to optimize accuracy while ensuring generalization to unseen data, Specifically, the class weight parameter was set to 'balanced' to equal the outcomes chances. With a fairly good Cross-Validated score of 0.77 the next Confusion Matrix was created:

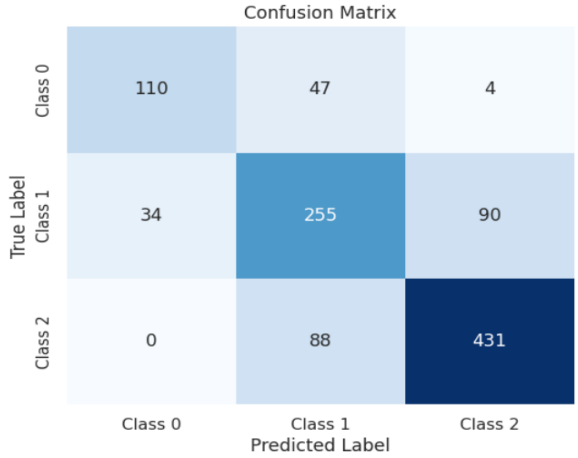

Figure 7.12: Confusion Matrix of the RF model

Figure 7.12 presents the Confusion Matrix for the Random Forest model, illustrating its classification performance. Class 2 (non-point finishes) achieved the highest accuracy with 431 correct predictions, while Class 1 (points but out of podium) had the most misclassifications (90), often confused with Class 2. These insights contribute to the calculation of next classification report, where precision, recall, and F1-score metrics provide a more detailed evaluation of the model's performance across all classes:

Classification Report				
	Precision	Recall	F1-Score	Support
Class 0	0.76	0.68	0.72	161
Class 1	0.65	0.67	0.66	379
Class 2	0.82	0.83	0.83	519
Accuracy			0.75	1059
Macro Avg	0.75	0.73	0.74	1059
Weighted Avg	0.75	0.75	0.75	1059

Table 7.2: Classification Report of the RF model

Table 7.2 highlights the key performance metrics of the Random Forest model. With an overall accuracy of 0.75, the model demonstrates a solid ability to classify race outcomes. Precision, recall, and F1-score exhibit the same variations across the three classes, with Class 2 achieving the highest scores (above 0.8) and Class 1 showing the lowest (below 0.7). This suggests that the model is more effective in identifying out-of-points finishes than other categories. Overall, results indicate a decent classification performance.

7.9.3 Classification with Support Vector Machines

Moving forward with the implementation of the Support Vector Machine (SVM) technique, the same scaled independent variables were split into training and testing subsets. To optimize performance, GridSearch was employed with 5-fold CV to fine-tune the model's hyperparameters. As a result, the radial basis function (rbf) kernel was selected, with the kernel coefficient "gamma" set to 0.01 and the regularization parameter "C" set to 10. The model achieved a higher cross-validated accuracy of 0.78, and the corresponding Confusion Matrix is presented in Figure 7.13.

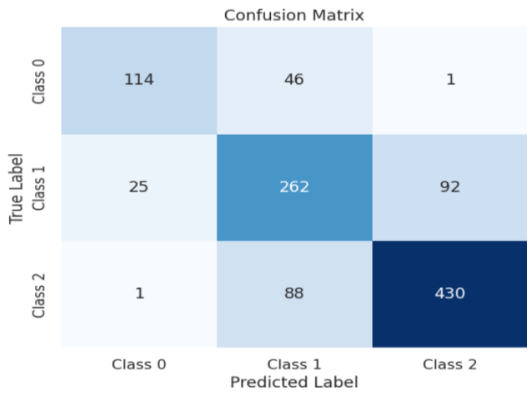

Figure 7.13: Confusion Matrix of the SVM model

Figure 7.13 presents the Confusion Matrix for the Support Vector Machine (SVM) model, showcasing its classification performance across the three classes. Class 2 once again achieved the highest accuracy with 430 correct predictions, while Class 1 exhibited the highest misclassification rate, with 92 instances misclassified as Class 2. Additionally, Class 0 showed relatively strong performance, with 114 correctly predicted instances, though 46 were misclassified as Class 1. Overall, the model demonstrates a solid classification capability, particularly for Class 2, while struggling slightly between Classes 1 and 2 as seen before, though with a better performance.

Classification Report				
	Precision	Recall	F1-Score	Support
Class 0	0.81	0.71	0.76	161
Class 1	0.66	0.69	0.68	379
Class 2	0.82	0.83	0.83	519
Accuracy			0.76	1059
Macro Avg	0.77	0.74	0.75	1059
Weighted Avg	0.76	0.76	0.76	1059

Table 7.3: Classification Report of the SVM model

Table 7.3 presents the Classification Report for the SVM model, summarizing key performance metrics, including precision, recall, and F1-score across the three classes. The model achieved an overall accuracy of 0.76, demonstrating reliable classification performance. Class 2 exhibited the highest precision (0.82) and recall (0.83), indicating that the model effectively identifies instances belonging to this category. Class 1, on the other hand, showed the lowest precision (0.66) and F1-score (0.68), suggesting greater difficulty in distinguishing this class from the others. Overall higher metrics are observed from the RF model's classification report.

7.9.4 Clustering with K-Means

To initiate the clustering process with the K-Means method, the independent scaled variables from the classification techniques were utilized without any train-test split or target variable, as clustering operates in an unsupervised learning framework. To determine the optimal number of clusters, the Elbow Plot in Figure 7.14 was generated, providing insights into the most suitable number of clusters for further analysis.

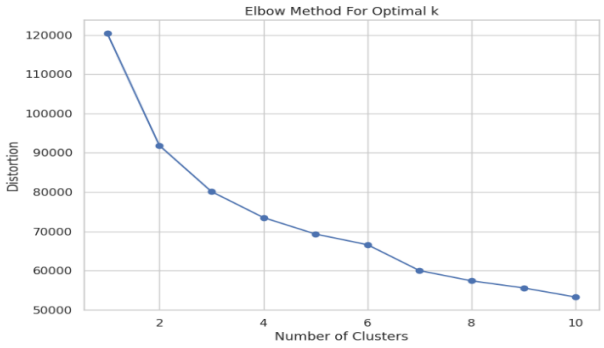

Figure 7.14: Elbow Plot for K-Means

As observed in Figure 7.14, the elbow plot suggests that the optimal number of clusters is between 2 and 3, where the curve begins to level off. To facilitate a meaningful comparison between the clustering results and the actual Race_Outcome variable, the number of clusters was set to 3. This decision enables the implementation of the Adjusted Rand Index (ARI) to evaluate clustering performance. With the clusters defined, the visualization of the clustering separation was generated in Figure 7.15.

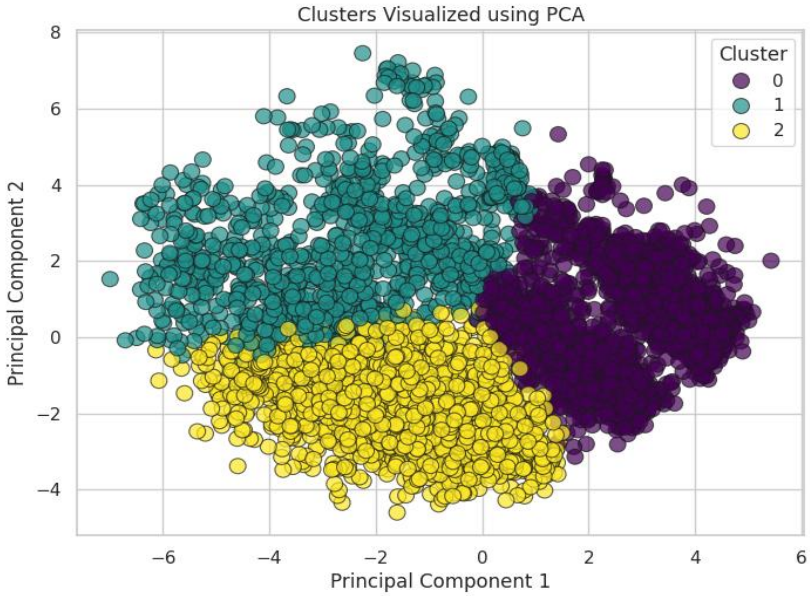

Figure 7.15: Visualized Cluster of K-Means

Figure 7.15 presents the visualized clusters obtained from the K-Means algorithm, demonstrating a relatively well-defined separation among the three clusters, albeit with variations in their densities. Cluster 0 -likely corresponding to podium finishes- show the most compact data points, while Clusters 1 and 2 spread wider into the plot. Class 1 points appear more scattered which combined with Class 2 volume it shows a logical alignment with the *Race_Outcome* variable distribution. These findings suggest that the clustering structure shows signs of relation with the *Race_Outcome* variable, which will be further examined in the subsequent evaluation tests.

Evaluation Technique	Value
Silhouette Coefficient	0.224
Adjusted Rand Index	0.193
Calinski-Harabasz Index	1463.455

Table 7.4: Evaluation Tests for K-Means

Table 7.4 presents the evaluation metrics used to assess the performance of the K-Means clustering model. The Silhouette Coefficient with a value of 0.224, indicate a moderate level of cohesion and separation. The Adjusted Rand Index (ARI), which compares the clustering results to the actual *Race_Outcome* variable, is 0.193, suggesting a weak but present alignment between the clusters and the ground truth labels. Lastly, the Calinski-Harabasz Index show a relatively high value of 1463.455, indicating a well-defined clustering structure.

7.9.5 Clustering with Hierarchical Agglomerative Clustering

At last, the Agglomerative Clustering method is applied using the same set of variables as in the previous clustering approach. To determine the optimal number of clusters, a dendrogram is constructed, providing a visual representation of hierarchical relationships within the data. By analyzing the distinct branches in the dendrogram, the most appropriate cluster divisions can be identified, capturing the underlying structure of the dataset effectively.

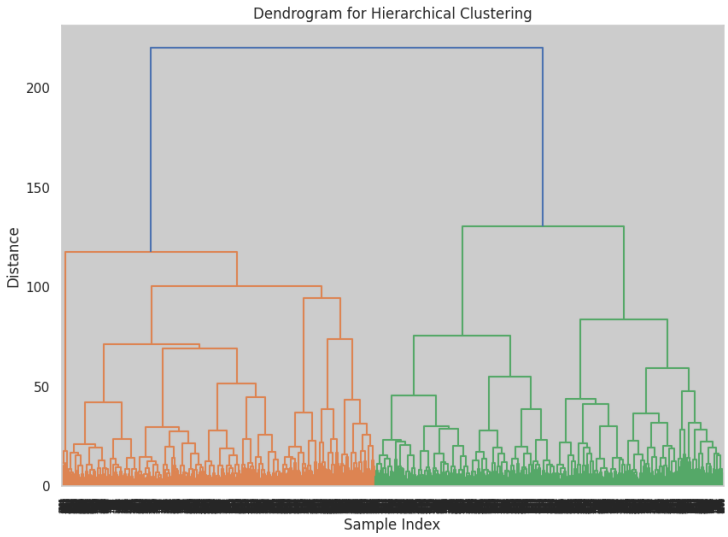

Figure 7.16: Dendrogram of HAC

As observed in the dendrogram, the observations are primarily divided into two main branches, suggesting an optimal cluster number of two. However, the right green branch further splits almost evenly, indicating that three clusters could also be a viable choice. To ensure a meaningful Adjusted Rand Index (ARI) test and facilitate a direct comparison with the previous clustering method, the number of clusters was set to three. With this decision finalized, the visualization of the clusters was created.

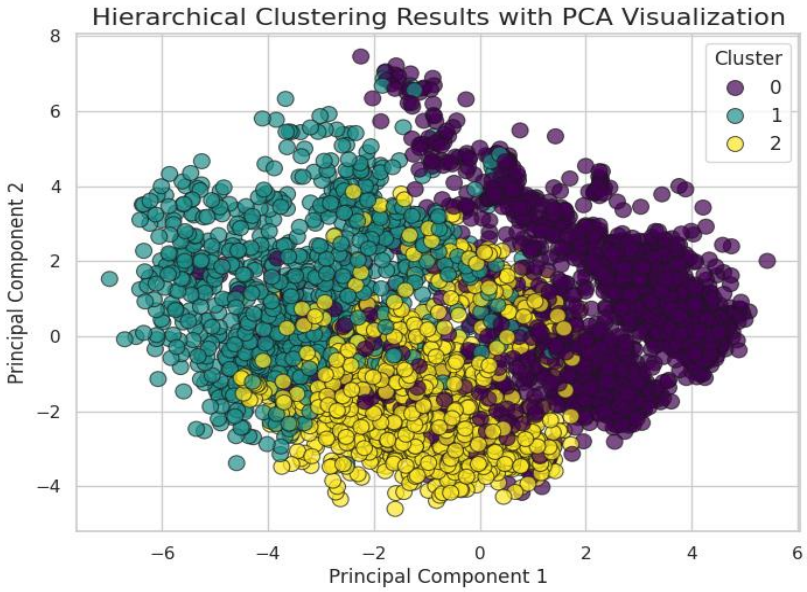

Figure 7.17: Visualized Cluster of HAC

Figure 7.17 illustrates the separation of clusters obtained through the Hierarchical Clustering method. Compared to the previous clustering approach, the three clusters are still distinguishable, though many points extend beyond the cluster boundaries. Notably, Class 2 points appear more compact, whereas the other two clusters' points are scattered throughout the plot. While Class 1 continues to show the fewest points, Class 2 -on the contrary from the previous method- also appear to enumerate around the same points as Class 1. Overall, the visualization demonstrates a reasonable degree of separation between clusters, suggesting that the method may yield promising results in the subsequent evaluation tests.

Evaluation Technique	Value
Silhouette Coefficient	0.120
Adjusted Rand Index	0.250
Calinski-Harabasz Index	990.879

Table 7.5: Evaluation Tests for the HAC

Table 7.5 provides key evaluation metrics for the Hierarchical Clustering model. The Silhouette Coefficient of 0.120, though lower than in the previous method, still indicates a reasonable level of cohesion and separation among the clusters. The Adjusted Rand Index (ARI) registers an improvement at 0.250, suggesting a slightly better alignment with the *Race_Outcome* variable. Additionally, the Calinski-Harabasz Index, at 990.879, remains relatively high -though lower than the previous technique- reinforcing the presence of a distinct clustering structure despite some dispersion in the data.

7.9.6 Conclusions from the Machine Learning Applications

This chapter explored the implementation of machine learning techniques, covering both classification and clustering methods. Initially, Principal Component Analysis was applied to reduce dimensionality, selecting the first 10 components, which accounted for approximately 79% of the explained variance. This step ensured that the most relevant features were retained while minimizing redundancy.

For classification, both Random Forest and Support Vector Machine models were employed to predict the *Race_Outcome* variable. The Random Forest model achieved an overall accuracy of 75%, with Class 2 demonstrating the highest classification performance. The SVM model, with optimized hyperparameters using Grid Search and cross-validation, slightly outperformed Random Forest with an accuracy of 76%, showing better balance across precision, recall, and F1-score.

In the unsupervised learning, the K-Means and Hierarchical Clustering techniques were implemented. The optimal number of clusters was determined to be three based on the elbow and dendrogram, while providing insightful comparison with the *Race_Outcome* variable. Evaluation metrics showed that K-Means yield slightly better separation, while Adjusted Rand Index showed that HAC's label distribution was closer to the *Race_Outcome* original data. The Calinski-Harabasz Indexes both show high values -whilst lower for HAC-, indicating well-defined clustering structures.

For an in-depth analysis of the in between label comparison of the two clustering techniques the bar plot was created, enumerating each label per Class:

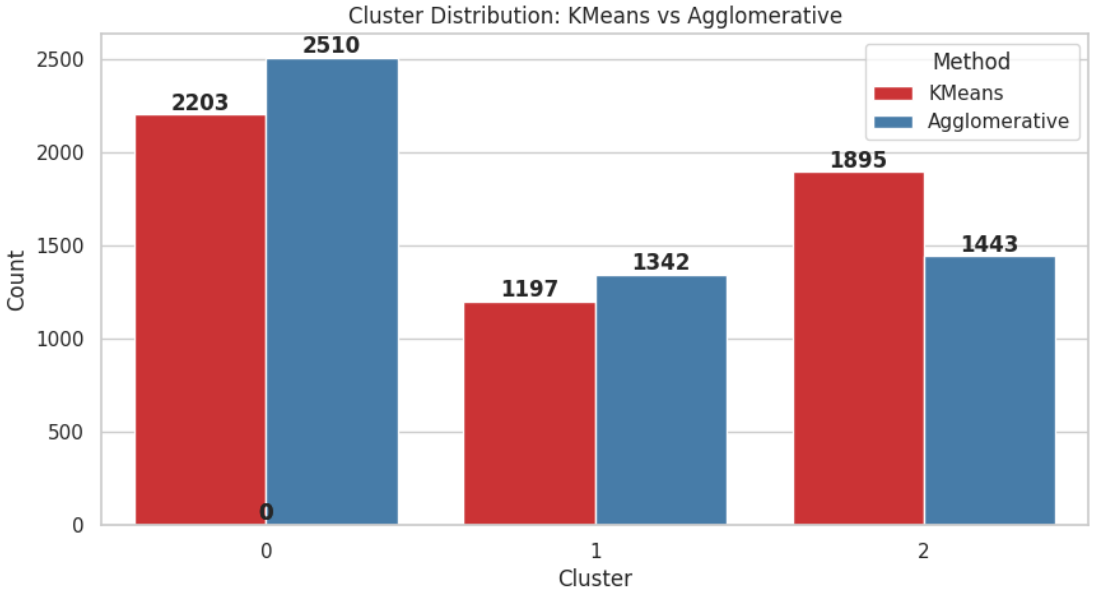

Figure 7.18: K-Means vs HAC cluster point distribution

Figure 7.18 illustrates the distribution of data points across clusters obtained from K-Means and Hierarchical Agglomerative Clustering (HAC). We can observe that the two techniques show a close point distribution with some slight fluctuations. Class 0 cluster contains 2,203 points in K-Means, while HAC assigns 2,510 points thus taking points from the other 2 classes which showed lower points compared to K-Means technique. Class 2 show the most significant gap between the 2 distributions, which explains the evaluation test differences. These differences highlight variations in how each clustering algorithm partitions the data. To further investigate the differences the next confusion matrix was created:

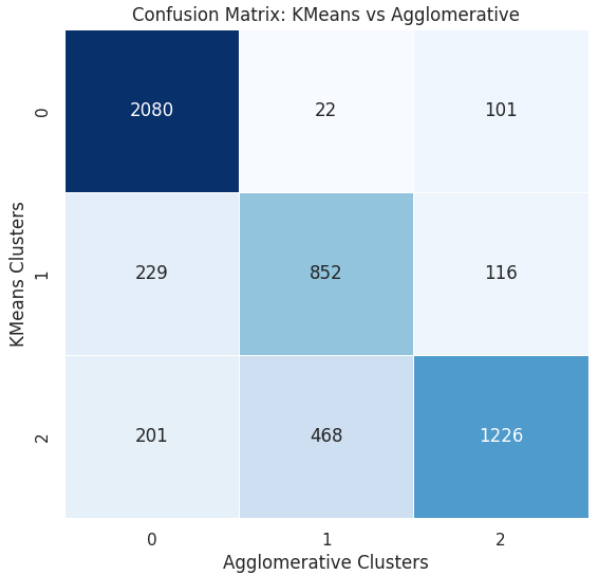

Figure 7.19: Confusion Matrix of the K-Means vs HAC cluster point distribution

Figure 7.19 presents the confusion matrix comparing K-means and Agglomerative Hierarchical Clustering (HAC), revealing how the two algorithms align in their cluster assignments. The large diagonal values (2080, 852, 1226) suggest that both methods largely agree on the grouping of data points, with K-means and HAC clustering most points into the same categories. However, the off-diagonal entries indicate that there are some discrepancies between the two, with certain data points being assigned to different clusters by each method. These differences highlight the inherent characteristics of K-means, which is sensitive to initialization and tends to form clusters based on centroid proximity, and HAC, which builds clusters iteratively based on data similarity. Additionally, the percentage of matching labels between the two methods was calculated to be 78.53%, further demonstrating a substantial, though not perfect, agreement between the clusterings. Overall, while both algorithms show a strong correlation in their clustering results, the confusion matrix underscores how their different approaches can lead to slight variations in the final cluster assignments.

Overall, results demonstrated that machine learning techniques can effectively analyze and categorize racing outcomes. While classification models provided strong predictive capabilities, clustering methods highlighted various underlying patterns.

Chapter 8: Conclusions

This study applied a range of statistical and machine learning techniques to analyze and predict performance in Formula 1 races. Beginning with extensive data preprocessing and feature engineering, the research established a structured dataset suitable for advanced analysis. Descriptive statistics and correlation studies provided initial insights into the relationships between variables, while normality tests ensured appropriate methodological choices. Regression models and time series analysis were employed to identify key performance trends and forecast future results. Machine learning techniques, including Principal Component Analysis, classification models, and clustering methods, further enhanced the understanding of race outcomes and driver performance. This chapter summarizes the key findings from each methodological approach, comparing their effectiveness and outlining the most significant results.

The second chapter provided a comprehensive background on Formula 1, covering its historical evolution, technical advancements, and regulatory changes that have shaped the sport. Additionally, it explored the financial dynamics, strategic elements, and the growing impact of data analytics in performance optimization. A literature review highlighted previous research in F1 analytics, establishing a foundation for the statistical and machine learning methodologies applied in this study.

Chapter 3 provided a detailed descriptive analysis of Formula 1 data, utilizing visualization techniques such as time series plots, scatter plots, and violin plots to reveal key trends. Fastest lap times showed a clear downward trend, highlighting technological advancements, while qualifying and grid positions strongly correlated with final race results. Driver-related attributes, including experience and tenure with a constructor, indicated that seasoned drivers achieve more consistent performance, as Garcia (2024) also emphasized in which factors mostly influence final results. Grouping race results into podium, points, and non-points finishes further emphasized performance differences, with podium finishers consistently achieving faster lap times with fewer pit stops. A focused analysis on circuit-specific fastest lap progression revealed significant reductions, particularly in race circuits rather than street circuits, as they better capture the impact of monocoque engineering advancements on lap time reductions. Additionally, as Jenkins (2020) illustrated, major regulatory changes are challenging to adapt to, further verifying that the temporary upward curves in lap times are indeed following new FIA rule implementations (observed in Figure 3.24).

Chapter 4 delved into the statistical characteristics and interrelationships of numerical variables within the dataset, revealing the inherent complexity of Formula 1 data. Initial normality checks using the Kolmogorov-Smirnov test confirmed that none of the variables followed a normal distribution, with variables like Age and Fastest Lap Time showing wider deviations. Visual tools like histograms and Q-Q plots highlighted right-skewed distributions and tail deviations, especially in qualifying times, even after excluding zero values. Correlation analysis through a heatmap demonstrated strong internal links between experience-related variables—Age, Race Starts, and Years of Experience—as well as notable ties between qualifying performance and race outcomes. Q1 times, in particular, displayed a high correlation

with both Fastest Lap Time and race completion metrics. When the data was grouped by race outcome (podium, points, no points), a clearer stratification of relationships emerged. Podium finishers showed tight performance clustering and strong qualifying consistency, but a diminished impact of experience. Mid-field drivers benefited more from experience, while non-point finishers reflected stronger dependencies on starting positions and Q1 performance. These findings highlight how experience, consistency, and qualifying performance interact differently across performance tiers, setting the stage for advanced modeling approaches.

Chapter 5 applied the previously outlined methodology to build a predictive model for race outcomes, using Multinomial Logistic Regression (MNL) due to the categorical nature of the dependent variable and the dataset's non-normal distribution. Categorical variables were encoded to prepare the data, with preprocessing steps such as grouping driver nationalities and classifying circuits by overtaking difficulty. Initial model testing achieved around 67% accuracy but required refinement due to multicollinearity and interpretability concerns. High VIF variables were systematically reduced using correlation insights from Chapter 4, followed by Wald significance testing, resulting in a streamlined and efficient model. The final model retained four key variables: *years_with_constructor*, *qualifying_position*, *cumulative_season_driver_points*, and *total_pit_stops*. These variables demonstrated strong explanatory power and statistical significance, aligning with literature findings. Wesselbaum and Owen (2020) emphasized the critical role of pole position, mirrored by the *qualifying_position* variable in the final model. Similarly, Sicoie (2022) highlighted the importance of the driver-constructor relationship, reflected by the inclusion of *years_with_constructor* in the final model. The model achieved a pseudo R^2 of 0.3513, with training and test accuracies of 71.4% and 69.7% respectively, balanced accuracy of 66.8%, and an ROC-AUC of 0.85, indicating robust classification performance. ROC curves showed especially high AUC for podium (0.94) and non-point finishes (0.87). Learning curves revealed minimal overfitting, and residual analysis confirmed a decent model fit. Finally, LassoCV supported the feature selection process by identifying largely overlapping variables, reinforcing the reliability and predictive validity of the model.

The sixth chapter implemented time series forecasting techniques to analyze and predict fastest lap times in Formula 1, focusing on the Australian Grand Prix (Albert Park) and the Monaco Grand Prix. Using average lap time data reconstructed from Chapter 3’s raw dataset, two separate time series were created—each spanning from 2004 to the present. At Albert Park, the time series required second-order differencing to achieve stationarity, after which ARIMA(1,0,1) was selected as the optimal model based on AICc and residual diagnostics. The forecast indicated a potential slowdown or reversal in the trend of continuous lap time improvement, with external factors such as regulation changes likely influencing performance. In contrast, the Monaco series achieved stationarity with first-order differencing, and ARIMA(1,0,0) emerged as the best-fitting model. Its forecast suggested a continued improvement in lap times, reflecting the influence of cornering dynamics and team strategies. Both models passed diagnostic tests for residual normality and independence, supporting their reliability for short-term forecasting. When compared with real-world data from the

2025 Australian Grand Prix and the 2024 Monaco Grand Prix, the forecasts aligned with observed trends, further validating the models and illustrating how statistical methods can anticipate performance outcomes in the ever-evolving landscape of Formula 1.

Chapter 7 explored the application of machine learning techniques -both supervised and unsupervised- to analyze and predict Formula 1 race outcomes. Leveraging the encoded dataset from Chapter 5, Race_Outcome was designated as the target variable, while other features were scaled to standardize inputs and enhance model performance. Principal Component Analysis (PCA) was employed as a dimensionality reduction technique, with the first 10 components retained based on a cumulative explained variance of 79%, ensuring a balance between information preservation and computational efficiency. Random Forest and Support Vector Machine (SVM) algorithms were implemented to classify race outcomes. The Random Forest model, with a cross-validated accuracy of 75%, performed best in identifying out-of-points finishes (Class 2), though it struggled with intermediate classifications (Class 1). The SVM model, optimized using grid search and a radial basis function kernel, slightly outperformed Random Forest with a 76% accuracy, offering improved balance in precision and recall across classes. Clustering was then approached using K-Means and Hierarchical Agglomerative Clustering (HAC), revealing distinct and partially overlapping cluster structures respectively. K-Means achieved a higher Silhouette Coefficient (0.224) and Calinski-Harabasz Index (1463.46), indicating better separation, whereas HAC demonstrated a slightly stronger alignment with the Race_Outcome variable via an improved Adjusted Rand Index of 0.250. Visualizations and comparative evaluations, including confusion matrices and label distribution analyses, revealed that while both clustering methods aligned in roughly 78.5% of their assignments, subtle differences arose due to their underlying mechanics. Overall, the chapter demonstrates how machine learning can both predict race outcomes and uncover latent structures in race data, offering valuable insights into the dynamics of Formula 1 performance.

In conclusion, this thesis presented a comprehensive analysis of Formula 1 race data using a variety of data science techniques. Through these methods, meaningful insights were drawn about the structure of race outcomes, the influence of driver and car-related variables, and the capacity of machine learning models to predict performance. One crucial factor that, while not directly analyzed, remains undeniably influential in the world of Formula 1 is weather. Strategic decisions such as tire selection, pit timing, and even race pace are heavily dictated by weather conditions, while driver performance can vary significantly under changing climates. As highlighted by Niamh (2023), weather introduces a level of uncertainty that can reshape the outcome of an entire race weekend. Furthermore, this thesis encountered challenges typical of motorsport data analysis. The nature of the data posed limitations on both feature consistency and model precision. These complexities, also noted by Aiginiti (2024), underscore the difficulty of building fully generalizable models in such a dynamic and multifactorial sport. Nevertheless, the findings demonstrate the value of data-driven approaches in Formula 1, offering predictive insights and revealing latent patterns that could support teams, analysts, or fans alike.

Bibliography

Greek

Aiginiti, C. I. (2024). Applications in Machine Learning Methods in Motosport. Thesis , Department of Statistics and Insurance Science, University of Piraeus, MSc "Applied Statistics"
https://dione.lib.unipi.gr/xmlui/handle/unipi/16849

Antzoulakos, D. (2021). Course notes "Data Analysis Using Statistical Packages: Introduction to R", Department of Statistics and Insurance Science, University of Piraeus, MSc "Applied Statistics", C. 6 p. 27.

Boutsikas, M. (2004). Course notes "Statistical Programs", Department of Statistics and Insurance Science, University of Piraeus, p. 32.

Chalikias, M., Manolesou, A., Panagiota, L. (2015). Research Methodology and Introduction to Statistical Data Analysis with IBM SPSS STATISTICS.
https://eclass.aegean.gr/modules/document/file.php/MATH113/%CE%92%CE%99%CE%92%CE%9B%CE%99%CE%91/spss_book1.pdf

Chrysis, V. (2024). Regular season and playoffs at the NBA : comparison and predictions using statistical methods. Thesis , Department of Statistics and Insurance Science, University of Piraeus, MSc "Applied Statistics"
https://dione.lib.unipi.gr/xmlui/handle/unipi/16349

Evangelaras, Ch. (2022). Course notes "Data Analysis Using Statistical Packages: Notes on IBM SPSS Statistics", Department of Statistics and Insurance Science, University of Piraeus, MSc "Applied Statistics", p. 46.

Goutos, K. (2021). Marginal effects of explanatory variables in a generalized linear model: Interpretation and applications. Thesis , Department of Statistics and Insurance Science, University of Piraeus, MSc "Applied Statistics"
https://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/13574/Goutos_18011.pdf?sequence=3&isAllowed=y

Kamitsis, A. (2023). Indices for evaluating player and team performance in basketball games, and factors affecting these indices. Thesis , Department of Statistics and Insurance Science, University of Piraeus, MSc "Applied Statistics"
https://dione.lib.unipi.gr/xmlui/handle/unipi/15593

Kolyva-Machaira, F., Bora-Senta, E. (2013). Statistics, Ziti Publications, pp. 331–334.

Koutras, M. (2020). Course notes "Regression Analysis and Analysis of Variance", Department of Statistics and Insurance Science, University of Piraeus, MSc "Applied Statistics", p. 114.

Pelekis, N. (2022). Course notes "Statistical Methods for Data Mining", Department of Statistics and Insurance Science, University of Piraeus, MSc "Applied Statistics", C. 1 p. 7.

Politis, K. (2021). Course notes "Generalized Linear Models", Department of Statistics and Insurance Science, University of Piraeus, MSc "Applied Statistics", p. 13.

Rakitzis, A. (2023). Course notes "Forecasting – Time Series", Department of Statistics and Insurance Science, University of Piraeus, MSc "Applied Statistics".

Triantafyllou, I. S. (2023). Course notes "Forecasting – Time Series", Department of Statistics and Insurance Science, University of Piraeus, MSc "Applied Statistics".

Foreign

Belgaid, A. (2024). Statistical Analysis of the Impact of FIA Regulations on Safety, Racing Dynamics, and Spectacle in Formula 1. https://arxiv.org/abs/2410.11375

Bobbit, Z. (2022). How to Overlay Normal Curve on Histogram in R (2 Examples). https://www.statology.org/overlay-normal-curve-histogram-in-r/

Breiman, L. (2001). Random Forests. Machine Learning, 45, pp 5–32. https://link.springer.com/article/10.1023/A:1010933404324

Budzinski, O., Feddersen, A. (2020). Chapter 1: Measuring Competitive Balance in Formula One Racing. https://www.elgaronline.com/edcollchap/edcoll/9781839102165/9781839102165.00006.xml

Calmon, W., Albi, M. (2020). Estimating the Number of Clusters in a Ranking Data Context. Information Sciences 546, pp 977-995. https://www.sciencedirect.com/science/article/pii/S002002552030966X

Ezugwu A.E.-S. et al. (2021). Automatic clustering algorithms: a systematic review and bibliometric analysis of relevant literature, Neural Computing and Applications, pp 14-15. https://www.researchgate.net/publication/344590665_Automatic_clustering_algorithms_a_systematic_review_and_bibliometric_analysis_of_relevant_literature

Han, J., Kamber, M., Pei, J. (2012). Data Mining Concepts and Techniques, 3rd edition. Cambridge, MA: Morgan Kaufmann. https://myweb.sabanciuniv.edu/rdehkharghani/files/2016/02/The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011.pdf

Hastie, T., Friedman, J., Tibshirani, R. (2009). The Elements of Statistical Learning Data Mining, Inference, and Prediction, 2nd edition. Springer. https://www.sas.upenn.edu/~fdiebold/NoHesitations/BookAdvanced.pdf

Iqbal, M. (2023). Scatter Diagram. https://mudassiriqbal.net/scatter-diagram/

James, G. et al. (2023). An Introduction to Statistical Learning with Applications in Python. Springer.
https://www.statlearning.com/

Jenkins, M. (2010). Technological Discontinuities and Competitive Advantage: A Historical Perspective on Formula 1 Motor Racing 1950–2006. Journal of Management Studies. Volume 47, Issue 5, pp 884 – 910.
https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-6486.2010.00928.x

Jolliffe, I.T. (2002). Principal Component Analysis, Second Edition. Springer.
http://cda.psych.uiuc.edu/statistical_learning_course/Jolliffe%20I.%20Principal%20Component%20Analysis%20(2ed.,%20Springer,%202002)(518s)_MVsa_.pdf

Koehrsen, W. (2017). Random Forest Simple Explanation.
https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d

Kumar, A. (2023). Support Vector Machine (SVM) Python Example.
https://vitalflux.com/classification-model-svm-classifier-python-example/

Long, J.S., Freese, J. (2014). Regression Models for Categorical Dependent Variables Using Stata, 3rd edition. Stata Press.
https://www.stata-press.com/books/preview/long3-preview.pdf

Mustafa, A. (2023). A Gentle Introduction to Complementary Log-Log Regression.
https://towardsdatascience.com/a-gentle-introduction-to-complementary-log-log-regression-8ac3c5c1cd83/

Niamh, D. (2023). Racing Against the Elements: Analysing the Impact of Weather on Formula One Races: Data Science Report.
https://norma.ncirl.ie/6920/

Nimmala, R., Nimmala, J. (2024). Racing into the Data Age: Sensor Intelligence, Advanced Analytics, and Kafka in Formula 1 Race Car. International Journal of Artificial Intelligence & Machine Learning (IJAIML), Volume 3, Issue 1, pp. 69-74.
https://iaeme-library.com/index.php/IJAIML/article/view/IJAIML_03_01_006

Piquero, A.R., Piquero, N. L., Han, S. (2021). Identifying the Most Successful Formula 1 Drivers in the Turbo Era. The Open Sports Sciences Journal, 14, pp 151-157.
https://research.monash.edu/en/publications/identifying-the-most-successful-formula-1-drivers-in-the-turbo-er

Premanand, S. (2023). The A-Z guide to Support Vector Machine.
https://www.analyticsvidhya.com/blog/2021/06/support-vector-machine-better-understanding/

Qian et al (2019). Orchestrating the Development Lifecycle of Machine Learning-Based IoT Applications: A Taxonomy and Survey.
https://www.researchgate.net/publication/336642133_Orchestrating_the_Development_Lifecycle_of_Machine_Learning-Based_IoT_Applications_A_Taxonomy_and_Survey

Quiroga García, M.P. (2024). Optimization of Pit Stop Strategies in Formula 1 Racing: A Data-Driven Approach.
https://repositorio.uchile.cl/handle/2250/199664

Rondelli, M. (2022). The Future of Formula 1 Racing: Neural Networks to Predict Tyre Strategy.
https://amslaurea.unibo.it/27922/1/Tesi_Massimo_Rondelli.pdf

Senaviratna, N.A.M.R., Cooray, T.M.J.A. (2019). Diagnosing Multicollinearity of Logistic Regression Model. Asian Journal of Probability and Statistics, Vol. 5, Issue 2, pp 1-9.
https://journalajpas.com/index.php/AJPAS/article/view/96

Shah, R. D. (2014). Statistical modelling.
http://www.statslab.cam.ac.uk/~rds37/teaching/statistical_modelling/notes.pdf

Sicoie, H. (2022). Machine Learning Framework for Formula 1 Race Winner and Championship Standings Predictor.
https://arno.uvt.nl/show.cgi?fid=157635

Smola, A.J. (2002). Support Vector Machines and Kernel Algorithms.
https://alex.smola.org/papers/2002/SchSmo02b.pdf

Sutton, R.S., Barto, A.G. (1992). Reinforcement Learning: An Introduction, 2nd edition. MIT Press.
https://mitpress.ublish.com/ebook/reinforcement-learning-an-introduction-2-preview/2351/Cover

Tan, P.-N. et al. (2018) Introduction to data mining, 2nd edition. Pearson Education.

Tejada, L.G. (2023). Applying Machine Learning to Forecast Formula 1 Race Outcomes.
https://aaltodoc.aalto.fi/server/api/core/bitstreams/70d5a580-c282-4278-8462-94d061471546/content

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. Journal of the Royal Statistical Society. Series B (Methodological), Vol. 58, No. 1, pp. 267-288.
https://www.jstor.org/stable/2346178

Turney, S. (2024). Pearson Correlation Coefficient (r) | Guide & Examples.
https://www.scribbr.com/statistics/pearson-correlation-coefficient/

Von Schleinitz, J. et al. (2021). VASP: An Autoencoder-Based Approach for Multivariate Anomaly Detection and Robust Time Series Prediction with Application in Motorsport. Engineering Applications of Artificial Intelligence, 104.
https://www.sciencedirect.com/science/article/pii/S0952197621002025

Wesselbaum, D., Owen, P.D. (2020). The Value of Pole Position in Formula 1 History. The Australian Economic Review, vol. 54, no. 1, pp. 164–173
https://onlinelibrary.wiley.com/doi/epdf/10.1111/1467-8462.12401

Links

https://en.wikipedia.org/wiki/Q%E2%80%93Q_plot

https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/regression-and-correlation/strength-of-correlation.html

https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

https://racingpass.net/circuits/

https://github.com/liannewriting/YouTube-videos-public/tree/main/arima-model-time-series-prediction-python

https://www.twinkl.gr/teaching-wiki/history-of-f1

https://en.wikipedia.org/wiki/History_of_Formula_One

https://en.wikipedia.org/wiki/Formula_One

https://en.wikipedia.org/wiki/Open-wheel_car

https://racingnews365.com/every-world-champion-in-formula-1-history

https://www.gulfoilltd.com/technological-advancements-f1-racing

https://medium.com/@shrimangalevallabh789/all-you-need-to-know-about-f1-cars-technology-43177a84274b

https://www.f1-grandprix.com/history1.html#Origins

https://feature.com/blogs/feature-sneaker-boutique/the-history-of-formula-one-racing?srsltid=AfmBOor-MFrNptBec4VsLOjELQwTN8Qq63tzkmCjQvdM4jA1NgKT_JU8

https://motorsportstats.com/series/fia-formula-one-world-championship/summary/2024

https://www.planetf1.com/features/fia-governing-body-explained

https://www.fia.com/news/new-era-competition-fia-showcases-future-focused-formula-1-regulations-2026-and-beyond

https://www.planetf1.com/features/formula-1-history-drivers-vs-fia

https://en.wikipedia.org/wiki/History_of_Formula_One_regulations

https://en.wikipedia.org/wiki/Formula_One_regulations#Scoring

https://en.wikipedia.org/wiki/List_of_Formula_One_World_Championship_points_scoring_systems

https://www.planetf1.com/features/top-10-highest-f1-win-percentages

https://en.wikipedia.org/wiki/F1_Academy

https://en.wikipedia.org/wiki/List_of_Formula_One_driver_records#Highest_average_points_per_championship

https://en.wikipedia.org/wiki/Formula_One_car

https://en.wikipedia.org/wiki/Formula_One_racing#Historical_methods_2

https://en.wikipedia.org/wiki/Scuderia_Ferrari

https://en.wikipedia.org/wiki/Fiorano_Circuit

https://en.wikipedia.org/wiki/Ayrton_Senna

https://en.wikipedia.org/wiki/Niki_Lauda

https://simple.wikipedia.org/wiki/Formula_One_car

https://www.formula1.com/en/latest/article/the-beginners-guide-to-formula-1-tyres.61SvF0Kfg29UR2SPhakDqd

https://www.pirelli.com/tyres/en-gb/motorsport/f1/tyres

https://www.autosport.com/f1/news/f1-tyres-what-are-the-compounds-and-what-do-they-mean/10344284/

https://automobilist.com/en-eu/blogs/stories/formula-1-tyre-evolution?srsltid=AfmBOoqCmF-X5cR4Kl0TOgye3JSzXjdF9kcBQm5A4OdtQHeA-ctHEDic

https://en.wikipedia.org/wiki/Formula_One_Teams_Association

https://www.motorsport.com/f1/news/f1-cost-cap-what-is-it-how-it-works/10379799/

https://www.globalsportsadvocates.com/blog/understanding-the-f1-cost-cap.cfm

https://worldinsport.com/explained-the-history-of-formula-ones-budget-cap-implementation-and-rationale/

https://www.dive-bomb.com/article/driving-business-the-economics-of-a-formula-1-team

https://www.linkedin.com/pulse/price-tag-formula-1-complete-breakdown-r-daffa-favian-d-a--x1ole/

https://www.youtube.com/watch?v=4_yDVVutU38

https://www.youtube.com/watch?v=b2N-lZ1MWnY

https://www.mostlyf1.com/statistics/all-time-stats/driver-stats/most-points-with-the-current-points-system-driver/

https://en.wikipedia.org/wiki/List_of_Formula_One_circuits

https://www.catapult.com/blog/f1-data-analysis-transforming-performance

https://medium.com/@akshat.gattani/unleashing-the-speed-of-data-overview-on-how-formula-1-harnesses-data-science-16ac9684f3d9

https://www.intrafocus.com/2024/04/the-data-driven-race-to-victory/

https://www.forbes.com/sites/joelshapiro/2023/01/26/data-driven-at-200-mph-how-analytics-transforms-formula-one-racing/

https://www.forbes.com/sites/bernardmarr/2023/07/10/how-artificial-intelligence-data-and-analytics-are-transforming-formula-one-in-2023/

https://www.grandprix247.com/2024/07/29/the-evolution-of-pit-stops-in-formula-1/

https://www.f1hoesblog.com/post/the-history-of-formula-one-pitstops

https://statathlon.com/analysis-of-the-pit-stop-strategy-in-f1/

https://www.gpfans.com/en/f1-news/1016512/f1-undercut-overcut-explained/

https://logos-world.net/f1-logo/

https://www.dive-bomb.com/article/introducing-the-new-formula-1-kid-tailored-broadcast

https://www.scaler.com/topics/machine-learning/feature-selection-in-machine-learning/

https://www.turing.com/kb/reinforcement-learning-algorithms-types-examples

https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/

https://www.datanovia.com/en/lessons/determining-the-optimal-number-of-clusters-3-must-know-methods/#google_vignette

Appendix

A.1 Implementation Code of the Present Study

```
#Neccesary Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math
import plotly.express as px
import scipy.stats as stats
from sklearn.model_selection import train_test_split, learning_curve
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import balanced_accuracy_score, roc_auc_score, roc_curve, auc
import statsmodels.api as sm
from sklearn.preprocessing import LabelEncoder
from statsmodels.stats.outliers_influence import variance_inflation_factor
import random
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
from scipy.stats import shapiro
from scipy.stats import anderson
from statsmodels.stats.diagnostic import acorr_ljungbox
from statsmodels.tsa.arima.model import ARIMA
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from pprint import pprint
from sklearn.svm import SVC
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.metrics import adjusted_rand_score
from sklearn.metrics import calinski_harabasz_score
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage

np.random.seed(1234)
random.seed(1234)

# Seaborn style for better visualizations
sns.set(style="whitegrid")

# Loading Data from the CSV files
drivers = pd.read_csv("drivers.csv")
```

```
constructors = pd.read_csv("constructors.csv")
results = pd.read_csv("results.csv")
races = pd.read_csv("races.csv")
lap_times = pd.read_csv("lap_times.csv")
pit_stops = pd.read_csv("pit_stops.csv")
qualifying = pd.read_csv("qualifying.csv")
driver_standings = pd.read_csv("driver_standings.csv")
constructor_standings = pd.read_csv("constructor_standings.csv")
seasons = pd.read_csv("seasons.csv")
circuits = pd.read_csv("circuits.csv")
status = pd.read_csv("status.csv")

"""FIGURES FOR THEORY PART CHAPTER 2"""

"""Wins per Driver"""

# Filter results for position 1
wins_per_driver = results[results["positionOrder"] == 1].groupby("driverId").size()

# Merge with driver names
wins_per_driver = wins_per_driver.reset_index()
wins_per_driver.columns = ["driverId", "wins"]
wins_per_driver = wins_per_driver.merge(drivers[["driverId", "surname"]], on="driverId")

# Sort and display top 10 drivers
top_drivers = wins_per_driver.sort_values(by="wins", ascending=False).head(10)

# Visualization
plt.figure(figsize=(10, 6))
sns.barplot(x="wins", y="surname", data=top_drivers, palette="viridis")
plt.title("Top 10 Drivers by Wins")
plt.xlabel("Number of Wins")
plt.ylabel("Driver")
plt.show()

"""Points per Team"""

# Total points per team
points_per_team = results.groupby("constructorId")["points"].sum().reset_index()
points_per_team = points_per_team.merge(constructors[["constructorId", "name"]],
on="constructorId")

# Sort and display top 10 teams
top_teams = points_per_team.sort_values(by="points", ascending=False).head(10)

# Visualization
plt.figure(figsize=(10, 6))
sns.barplot(x="points", y="name", data=top_teams, palette="coolwarm")
plt.title("Top 10 Teams by Points")
plt.xlabel("Total Points")
plt.ylabel("Team")
```

```
plt.show()

"""Number of Races per Year"""

# Number of races per year
races_per_year = races.groupby("year").size().reset_index(name="race_count")

# Visualization
plt.figure(figsize=(12, 6))
sns.lineplot(x="year", y="race_count", data=races_per_year, marker="o")
plt.title("Number of Races per Year")
plt.xlabel("Year")
plt.ylabel("Number of Races")
plt.show()

"""Fastest Lap Timeseries"""

# Convert lap times to numeric
lap_times["milliseconds"] = pd.to_numeric(lap_times["milliseconds"], errors="coerce")

# Calculate average lap time per circuit and year
avg_lap_time_per_circuit =
lap_times.groupby(["raceId"])["milliseconds"].mean().reset_index()
avg_lap_time_per_circuit = avg_lap_time_per_circuit.merge(races[["raceId", "year",
"circuitId"]], on="raceId")
avg_lap_time_per_circuit = avg_lap_time_per_circuit.merge(circuits[["circuitId", "name"]],
on="circuitId")

# Filter only for Albert Park Grand Prix Circuit
filtered_circuits = ["Albert Park Grand Prix Circuit"]
avg_lap_time_filtered =
avg_lap_time_per_circuit[avg_lap_time_per_circuit["name"].isin(filtered_circuits)]

# Calculate average lap time per circuit and year
avg_lap_time_filtered = avg_lap_time_filtered.groupby(["year",
"name"])["milliseconds"].mean().reset_index()

# Visualization
plt.figure(figsize=(12, 6))
sns.lineplot(x="year", y="milliseconds", data=avg_lap_time_filtered, marker="o")
plt.title("Average Lap Time: Albert Park Grand Prix Circuit", fontsize=16)
plt.xlabel("Year", fontsize=12)
plt.ylabel("Average Lap Time (milliseconds)", fontsize=12)
plt.tight_layout()
plt.show()

"""Pit Stops Analysis"""

# Calculate the average number of pit stops per race per year
pit_stops_per_year = pit_stops.merge(races[["raceId", "year"]], on="raceId")
pit_stops_per_race = (pit_stops_per_year.groupby("year").size()  # Total pit stops per year
```

```
                / races.groupby("year").size()          # Total races per year
                ).reset_index(name="average_pit_stops_per_race")

# Only for finished periods
pit_stops_per_race = pit_stops_per_race[pit_stops_per_race["year"] <= 2023]

# Visualization
plt.figure(figsize=(12, 6))
sns.lineplot(x="year", y="average_pit_stops_per_race", data=pit_stops_per_race,
marker="o")
plt.title("Average Pit Stops per Race per Year")
plt.xlabel("Year")
plt.ylabel("Average Pit Stops per Race")
plt.grid(True)
plt.show()

"""Creating a unified dataset for our analysis's purposes"""

# Grouping data by race and driver, calculating total pit stops and total duration
Cumulative_pit_stops = pit_stops.groupby(['raceId', 'driverId']).agg(
    total_pit_stops=('stop', 'count'),      # Total pit stops for the driver in the race
    total_duration_ms=('milliseconds', 'sum') # Total duration in milliseconds
).reset_index()

# Updated selection of important columns
drivers_selected = drivers[['driverId', 'forename', 'surname', 'nationality', 'dob']]
constructors_selected = constructors[['constructorId', 'name', 'nationality']]
races_selected = races[['raceId', 'date', 'circuitId', 'year', 'round']]
qualifying_selected = qualifying[['raceId', 'driverId', 'position', 'q1', 'q2', 'q3']]
results_selected = results[['raceId', 'driverId', 'constructorId', 'grid', 'points', 'fastestLapTime',
'statusId', 'laps', 'positionOrder']]
circuits_selected = circuits[['circuitId', 'name']]
pit_stops_selected = Cumulative_pit_stops[['raceId', 'driverId',
'total_pit_stops','total_duration_ms']]

# Renaming columns for clarity
drivers_selected = drivers_selected.rename(columns={
    'forename': 'driver_forename',
    'surname': 'driver_surname',
    'nationality': 'driver_nationality'
})
constructors_selected = constructors_selected.rename(columns={
    'name': 'constructor_name',
    'nationality': 'constructor_nationality'
})
races_selected = races_selected.rename(columns={
    'date': 'race_date'
})
qualifying_selected = qualifying_selected.rename(columns={
    'position': 'qualifying_position'
})
```

```
results_selected = results_selected.rename(columns={
    'grid': 'grid_position',
    'points': 'driver_points',
    'fastestLapTime': 'fastest_lap_time',
    'statusId': 'race_status'
})
circuits_selected = circuits_selected.rename(columns={
    'name': 'circuit_name'})

# Merging DataFrames
df_merged = results_selected.merge(qualifying_selected, on=['raceId', 'driverId'], how='left') \
                        .merge(drivers_selected, on='driverId', how='left') \
                        .merge(constructors_selected, on='constructorId', how='left') \
                        .merge(races_selected, on='raceId', how='left') \
                        .merge(circuits_selected, on='circuitId', how='left') \
                        .merge(pit_stops_selected, on=['raceId', 'driverId'], how='left')

# Selecting the final columns
final_columns = ['driverId', 'driver_forename', 'driver_surname', 'driver_nationality', 'dob',
                 'constructorId', 'constructor_name', 'constructor_nationality',
                 'raceId',  'race_date','circuitId','circuit_name','year',  'round',  'qualifying_position',
'q1', 'q2', 'q3',
                 'grid_position','driver_points', 'fastest_lap_time', 'race_status','laps', 'positionOrder',
                 'total_pit_stops','total_duration_ms']

# Creating the final DataFrame
final_df = df_merged[final_columns]

"""Making one column for the drivers"""

# Concatenate 'driver_forename' and 'driver_surname' with a space in between
final_df['driver_name']  =  final_df['driver_forename'].astype(str)  +  '  '  +
final_df['driver_surname'].astype(str)

# Drop the original columns
final_df = final_df.drop(columns=['driver_forename', 'driver_surname'])

"""Sort by Race date"""

# Sorting the final DataFrame by 'race_date' in ascending order
final_df = final_df.sort_values(by='race_date').reset_index(drop=True)

"""Additional Columns"""

# Convert 'dob' to datetime format if not already done
final_df['dob'] = pd.to_datetime(final_df['dob'], errors='coerce')

# Calculate Age for each race
final_df['age'] = final_df['year'] - final_df['dob'].dt.year

# Calculate Years of Experience
```

```
# Find the first year each driver participated in a race
first_race_year = final_df.groupby('driverId')['year'].transform('min')
final_df['years_experience'] = final_df['year'] - first_race_year

# Calculate cumulative race starts for each driver within the season up to each race
final_df['race_starts'] = final_df.groupby('driverId').cumcount() + 1

# Calculate cumulative points for each driver within the season up to each race
final_df['cumulative_season_driver_points'] = final_df.groupby(['year', 'driverId'])['driver_points'].cumsum()

"""Adding the Constructor with the most years of co-working with each driver to each date."""

# First the necessary fields, excluding the year of the current race to only count previous collaborations
driver_constructor_years = final_df[['driverId', 'constructor_name', 'year']].drop_duplicates()
driver_constructor_years['years_with_constructor'] = driver_constructor_years.groupby(['driverId', 'constructor_name'])['year'].cumcount()

# Merging with the final DataFrame to add the cumulative collaboration information
final_df = final_df.merge(driver_constructor_years[['driverId', 'constructor_name', 'year', 'years_with_constructor']], on=['driverId', 'constructor_name', 'year'], how='left')

# Finding the constructor with the most cumulative collaboration for each driver up to the previous year
# Ignoring the current year when calculating the dominant constructor
def get_top_constructor(row):
    driver_data = final_df[(final_df['driverId'] == row['driverId']) & (final_df['year'] < row['year'])]
    if not driver_data.empty:
        return driver_data.groupby('constructor_name')['years_with_constructor'].max().idxmax()
    return row['constructor_name']

# Applying the function to each row of the DataFrame
final_df['top_constructor'] = final_df.apply(get_top_constructor, axis=1)

"""Adding Top3, Top10 (4-10) and OutsideTop10"""

# Create a new function to calculate the positions
def calculate_positions(row):
    # Filter the data for the driver up to the current year
    driver_data = final_df[(final_df['driverId'] == row['driverId']) & (final_df['year'] < row['year'])]

    # Calculate the positions in the Top 3 category (positions 1, 2, 3)
    top_3_count = driver_data[driver_data['positionOrder'].isin([1, 2, 3])].shape[0]

    # Calculate the positions in the 4th to 10th category (positions 4-10)
```

```
    top_10_count = driver_data[(driver_data['positionOrder'] >= 4) & (driver_data['positionOrder'] <= 10)].shape[0]

    # Calculate the positions outside the Top 10 (positions > 10)
    outside_10_count = driver_data[driver_data['positionOrder'] > 10].shape[0]

    return pd.Series([top_3_count, top_10_count, outside_10_count], index=['top_3_positions', 'positions_4_10', 'outside_10_positions'])

# Apply the function to each row of the DataFrame
final_df[['top_3_positions', 'positions_4_10', 'outside_10_positions']] = final_df.apply(calculate_positions, axis=1)

"""Creating column for further analysis"""

# Define a function to classify the race outcome
def classify_race_outcome(row):
    if row['positionOrder'] in [1, 2, 3]:
        return 'Podium finish'
    elif 4 <= row['positionOrder'] <= 10:
        return 'Points without podium'
    else:
        return 'No points'

# Apply the function to create the new 'race_outcome' column
final_df['race_outcome'] = final_df.apply(classify_race_outcome, axis=1)

"""Making our working df from 2011, to have exact metrics for qualifying and pitstops"""

# Filter final_df to include only rows where 'year' is 2011 or later
working_df = final_df[final_df['year'] >= 2011].copy()

"""Dropping rows with NaN that didn't contribute to the analysis"""

# Replace '\N' with NaN only in the 'fastest_lap_time' column
working_df['fastest_lap_time'].replace('\\N', np.nan, inplace=True)

# Drop rows with NaN in any of the specified columns
working_df.dropna(subset=['qualifying_position','fastest_lap_time', 'total_pit_stops', 'total_duration_ms'], inplace=True)

# Verify that there are no more NaN values in the specified columns
print(working_df[['qualifying_position','fastest_lap_time', 'total_pit_stops', 'total_duration_ms']].isnull().sum())

"""Dealing with NaNs from qualifying sessions"""

# Replace '\N' with NaN in q1, q2, and q3 columns to standardize missing values
working_df['q1'].replace('\\N', np.nan, inplace=True)
working_df['q2'].replace('\\N', np.nan, inplace=True)
working_df['q3'].replace('\\N', np.nan, inplace=True)
```

```
# Add binary indicators for participation in Q1, Q2, and Q3 (to not be misinterpreted as an actual time)
working_df['valid_time_set_in_q1'] = working_df['q1'].notna().astype(int)
working_df['qualified_for_q2'] = working_df['q2'].notna().astype(int)
working_df['qualified_for_q3'] = working_df['q3'].notna().astype(int)

# Replace NaN values in Q2 and Q3 with '0:00.000' as the time for non-participation
working_df['q1'].fillna('0:00.000', inplace=True)
working_df['q2'].fillna('0:00.000', inplace=True)
working_df['q3'].fillna('0:00.000', inplace=True)

"""Rearranging and keeping the needed columns"""

# Dropping specified columns
working_df = working_df.drop(['driverId', 'dob', 'constructorId', 'raceId', 'race_date', 'circuitId'], axis=1)

# Rearranging the remaining columns in a specific order
# Define the new column order as a list of column names
new_column_order = [
    'driver_name', 'driver_nationality', 'age', 'years_experience', 'race_starts',
    'constructor_name', 'constructor_nationality','years_with_constructor', 'top_constructor',
    'year', 'round', 'circuit_name', 'valid_time_set_in_q1','q1','qualified_for_q2','q2', 'qualified_for_q3','q3', 'qualifying_position',
    'grid_position','positionOrder','race_status','driver_points', 'cumulative_season_driver_points','fastest_lap_time', 'laps',
    'total_pit_stops', 'total_duration_ms','top_3_positions', 'positions_4_10', 'outside_10_positions', 'race_outcome',
]

# Reordering the DataFrame columns
working_df = working_df[new_column_order]

# Convert Q1, Q2, Q3, and Fastest_lap_time to seconds
def time_to_seconds(time_str):
    try:
        mins, secs = map(float, time_str.split(':'))
        return mins * 60 + secs
    except:
        return None

for col in ['q1', 'q2', 'q3', 'fastest_lap_time']:
    working_df[col] = working_df[col].apply(time_to_seconds)

# Grouping race_status variable

# Define groupings of numbers for categories
group_mapping = {
    'FINISHED': [1],
    'DSQ': [2, 96],
```

```
    'RETIRED': [31],
    'WITHDREW': [54],
    'DRIVER RELATED': [139],
    'LAPPED': [11,12,13,14,15,16,17,18,19,45,88,111],
    'RACE INCIDENT': [3,4,20,29,130,137],
    'CAR RELATED': [5, 6, 7, 8, 9, 10, 21, 22, 23, 24, 25, 26, 27, 30, 32, 33, 34, 36, 37, 38, 39,
40, 42, 43, 44, 47, 48, 51, 60, 61, 63, 65, 75, 76, 79, 84, 91, 95, 101, 103, 131, 132, 135, 136,
140, 141]
}

# Reverse the group_mapping to map each number to its category
rename_mapping = {num: category for category, nums in group_mapping.items() for num in
nums}

# Map the 'race_status' column
working_df['race_status'] = working_df['race_status'].map(rename_mapping)

"""Checking the whole working_df for NaNs"""

# Check for NaN values across all columns
nan_counts = working_df.isna().sum()
print("NaN Counts:")
print(nan_counts[nan_counts > 0])  # Display only columns with NaN values

# Check for "\N" values across all columns
backslash_n_counts = (working_df == r'\N').sum()
print("\\N Counts:")
print(backslash_n_counts[backslash_n_counts > 0])  # Display only columns with \N values

print(working_df)

# Export merged_df to an Excel file in the current directory
final_df.to_excel("final_f1_data_V16.xlsx", index=False, engine="openpyxl")

# Export merged_df to an Excel file in the current directory
working_df.to_excel("working_f1_data_V5.xlsx", index=False, engine="openpyxl")

"""Descriptive Statistics"""

# Read the created working excel
working_df = pd.read_excel("working_f1_data_V5.xlsx")

"""Metrics"""

# Calculate descriptive statistics for numerical columns
descriptive_stats = working_df.describe(percentiles=[0.25, 0.5, 0.75]).T

# Keep only the columns that we are interested in
descriptive_stats = descriptive_stats[['min', '25%', '50%', 'mean', '75%', 'max']]

# Rename columns for clearer display
```

```
descriptive_stats.columns = ['Min', '1st Q (25%)', 'Median (50%)', 'Mean', '3rd Q (75%)',
'Max']

# Display the results
print(descriptive_stats)

"""Timeseries Plots"""

# Calculate the mean per year excluding zero values
def custom_mean(series):
    # Filter out zero values (0)
    valid_values = series[series > 0]
    return valid_values.mean()

# Grouping data by year
yearly_data = working_df.groupby('year').agg({
     'driver_points': 'mean',
    'cumulative_season_driver_points': 'mean',
    'fastest_lap_time': 'mean',
    'laps': 'mean',
    'total_pit_stops': 'mean',
    'total_duration_ms': 'mean',
    'top_3_positions': 'mean',
    'positions_4_10': 'mean',
    'outside_10_positions': 'mean'
}).reset_index()

# Defining variables for plotting
plot_vars = {
    'driver_points': 'Avg. Driver Points',
    'cumulative_season_driver_points': 'Avg. Cumulative Points',
    'fastest_lap_time': 'Avg. Fastest Lap (s)',
    'laps': 'Avg. Laps',
    'total_pit_stops': 'Avg. Total Pit Stops per Race',
    'total_duration_ms': 'Avg. Total Pit Stop Duration per Race (ms)',
    'top_3_positions': 'Avg. Times in Podium',
    'positions_4_10': 'Avg. Times in Positions 4-10',
    'outside_10_positions': 'Avg. Times Without Points'
}

# Creating a grid of plots
fig, axes = plt.subplots(3, 3, figsize=(15, 12))
axes = axes.flatten()

for i, (var, label) in enumerate(plot_vars.items()):
    ax = axes[i]
    sns.lineplot(data=yearly_data, x='year', y=var, ax=ax, marker='o')
    ax.set_title(label, fontsize=12)
    ax.set_xlabel('Year', fontsize=10)
    ax.set_ylabel(label, fontsize=10)
    ax.grid(True)
```

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

# Grouping data by year
yearly_data = working_df.groupby('year').agg({
    'qualifying_position': 'mean',
    'grid_position': custom_mean,
    'positionOrder': 'mean',
    'valid_time_set_in_q1': 'sum',
    'qualified_for_q2': 'sum',
    'qualified_for_q3': 'sum',
    'q1': custom_mean,
    'q2': custom_mean,
    'q3': custom_mean,
}).reset_index()

# Defining variables for plotting
plot_vars = {
    'qualifying_position': 'Avg. Qualifying Position',
    'grid_position': 'Avg. Grid Position',
    'positionOrder': 'Avg. Position Order',
    'valid_time_set_in_q1': 'Total Times with Valid Time in Q1',
    'qualified_for_q2': 'Total Times Qualified for Q2',
    'qualified_for_q3': 'Total Times Qualified for Q3',
    'q1': 'Avg. Fastest Lap in Q1',
    'q2': 'Avg. Fastest Lap in Q2',
    'q3': 'Avg. Fastest Lap in Q3'
}

# Creating a grid of plots
fig, axes = plt.subplots(3, 3, figsize=(15, 12))
axes = axes.flatten()

for i, (var, label) in enumerate(plot_vars.items()):
    ax = axes[i]
    sns.lineplot(data=yearly_data, x='year', y=var, ax=ax, marker='o')
    ax.set_title(label, fontsize=12)
    ax.set_xlabel('Year', fontsize=10)
    ax.set_ylabel(label, fontsize=10)
    ax.grid(True)

fig.tight_layout()
plt.show()

# Grouping data by year
yearly_data = working_df.groupby('year').agg({
    'age': 'mean',
    'years_experience': 'mean',
    'years_with_constructor': 'mean',
    'race_starts': 'mean'
```

```
}).reset_index()

# Defining variables for plotting
plot_vars = {
    'age': 'Avg. Age',
    'years_experience': 'Avg. Years of Experience',
    'years_with_constructor': 'Avg. Years with Current Constructor',
    'race_starts': 'Avg. Race Starts'
}

# Creating a grid of plots
fig, axes = plt.subplots(3, 3, figsize=(15, 12))
axes = axes.flatten()

for i, (var, label) in enumerate(plot_vars.items()):
    ax = axes[i]
    sns.lineplot(data=yearly_data, x='year', y=var, ax=ax, marker='o')
    ax.set_title(label, fontsize=12)
    ax.set_xlabel('Year', fontsize=10)
    ax.set_ylabel(label, fontsize=10)
    ax.grid(True)

# Removing empty subplots (we have fewer than 9 plots)
for j in range(len(plot_vars), len(axes)):
    fig.delaxes(axes[j])

fig.tight_layout()
plt.show()

# Bar Plot for driver_name

plt.figure(figsize=(12, 22))
data = working_df['driver_name'].value_counts()
sns.barplot(x=data.values, y=data.index, palette='viridis')
plt.title('Driver Appearances', fontsize=16)
plt.xlabel('Count', fontsize=14)
plt.ylabel('Driver Names', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()

# Variables to create Barplots
barplot_vars = ['driver_nationality', 'constructor_name',
'constructor_nationality','top_constructor','circuit_name']

# Custom Titles for Each Plot
custom_titles = [
    "Nationality Appearances of Drivers",
    "Constructors Appearances",
    "Nationality Appearances of Constructors",
    "Constructor Experience of each Driver per Race (Longest Period of Working Together)",
    "Circuit Appearances"
```

```
]

# Bar Plots with Custom Titles
for var, title in zip(barplot_vars, custom_titles):
    plt.figure(figsize=(12, 8))  # Adjust the figure size to fit all categories
    data = working_df[var].value_counts()  # Count all categories
    sns.barplot(x=data.values, y=data.index, palette='viridis')
    plt.title(title, fontsize=16)  # Use the custom title here
    plt.xlabel('Count', fontsize=14)
    plt.ylabel(var, fontsize=14)
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.show()

# Define the custom order for the race outcome categories
custom_order = ["Podium finish", "Points without podium", "No points"]

# Count the occurrences of each race outcome category
outcome_counts = working_df['race_outcome'].value_counts()

# Reorder the data according to the custom order
outcome_counts = outcome_counts.reindex(custom_order)

# Plotting the vertical bar plot with reordered categories
plt.figure(figsize=(8, 6))
sns.barplot(x=outcome_counts.index, y=outcome_counts.values, palette='viridis')

# Adding labels and a title
for i, count in enumerate(outcome_counts.values):
    plt.text(i, count + 0.5, str(count), ha='center', fontsize=12)

plt.title("Cumulative Race Outcomes", fontsize=16)
plt.xlabel("Race Outcome", fontsize=14)
plt.ylabel("Count", fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.show()

# Driver Appearances per Year Bar Plot
plt.figure(figsize=(8, 6))
data = working_df['year'].value_counts()  # Count all categories
sns.barplot(x=data.index, y=data.values)
plt.title("Driver Appearances per Year", fontsize=16)
plt.xlabel("Years", fontsize=14)
plt.ylabel("Count", fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.show()

# Driver Appearances per Round Bar Plot
plt.figure(figsize=(8, 6))
data = working_df['round'].value_counts()  # Count all categories
sns.barplot(x=data.index, y=data.values)
plt.title("Driver Appearances per Round", fontsize=16)
```

```
plt.xlabel("Rounds", fontsize=14)
plt.ylabel("Count", fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.show()

# Count occurrences of each race_status
status_counts = working_df['race_status'].value_counts()

# Plot the distribution of race_status
plt.figure(figsize=(13, 6))
sns.barplot(x=status_counts.index, y=status_counts.values)
plt.title("Race Status", fontsize=16)
plt.xlabel("Race Status", fontsize=14)
plt.ylabel("Count", fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.6)

# Adding counts on the bars
for i, count in enumerate(status_counts.values):
    plt.text(i, count + 5, str(count), ha='center', fontsize=12)
plt.show()

"""Scatter Plots vs Position Order"""

# Identify numeric columns to plot
numeric_columns = working_df.select_dtypes(include=['float64', 'int64']).columns

# Determine the grid size
num_vars = len(numeric_columns)
cols = 4  # Number of columns in the grid
rows = math.ceil(num_vars / cols)

# Create subplots with a shared figure
fig, axes = plt.subplots(rows, cols, figsize=(8, rows * 2), constrained_layout=True)
axes = axes.flatten()  # Flatten for easy indexing

# Loop through numeric columns and create scatterplots
for i, column in enumerate(numeric_columns):
    sns.scatterplot(
        data=working_df,
        x=column,
        y='positionOrder',  # PositionOrder as the dependent variable
        alpha=0.2,
        size=300,
        legend=False,
        ax=axes[i]
    )
    axes[i].set_xlabel(column, fontsize=8)
    axes[i].set_ylabel('Position Order', fontsize=8)

# Turn off unused axes
for j in range(i + 1, len(axes)):
```

```
    axes[j].axis('off')

plt.show()

# Identify numeric columns to plot
numeric_columns = working_df.select_dtypes(include=['float64', 'int64']).columns

# Exclude specific columns from the numeric columns
columns_to_exclude = ['years_with_constructor', 'year', 'round', 'valid_time_set_in_q1',
                      'q1', 'qualified_for_q2', 'q2', 'qualified_for_q3', 'q3',
                      'positionOrder', 'total_duration_ms']
numeric_columns = [col for col in numeric_columns if col not in columns_to_exclude]

# Determine the grid size
num_vars = len(numeric_columns)
cols = 5  # Number of columns in the grid
rows = math.ceil(num_vars / cols)

# Create subplots with a shared figure
fig, axes = plt.subplots(rows, cols, figsize=(8, rows * 2), constrained_layout=True)
axes = axes.flatten()  # Flatten for easy indexing

# Loop through numeric columns and create scatterplots
for i, column in enumerate(numeric_columns):
    sns.scatterplot(
        data=working_df,
        x=column,
        y='positionOrder',  # PositionOrder as the dependent variable
        alpha=0.2,
        size=300,
        legend=False,
        ax=axes[i]
    )
    axes[i].set_xlabel(column, fontsize=8)
    axes[i].set_ylabel('Position Order', fontsize=8)

# Turn off unused axes
for j in range(i + 1, len(axes)):
    axes[j].axis('off')

plt.show()

"""Time Series Plots groupped by Race Outcome"""

# Grouping data by year
yearly_data = working_df.groupby('year').agg({
     'driver_points': 'mean',
    'cumulative_season_driver_points': 'mean',
    'fastest_lap_time': 'mean',
    'laps': 'mean',
    'total_pit_stops': 'mean',
```

```
    'total_duration_ms': 'mean',
    'top_3_positions': 'mean',
    'positions_4_10': 'mean',
    'outside_10_positions': 'mean'
}).reset_index()

# Defining variables for plotting
plot_vars = {
    'driver_points': 'Avg. Driver Points',
    'cumulative_season_driver_points': 'Avg. Cumulative Points',
    'fastest_lap_time': 'Avg. Fastest Lap (s)',
    'laps': 'Avg. Laps',
    'total_pit_stops': 'Avg. Total Pit Stops per Race',
    'total_duration_ms': 'Avg. Total Pit Stop Duration per Race (ms)',
    'top_3_positions': 'Avg. Times in Podium',
    'positions_4_10': 'Avg. Times in Positions 4-10',
    'outside_10_positions': 'Avg. Times Without Points'
}

# Creating time series with separation by race_outcome
fig, axes = plt.subplots(3, 3, figsize=(15, 12))  # Create a grid of plots
axes = axes.flatten()

for i, (var, label) in enumerate(plot_vars.items()):
    ax = axes[i]
    sns.lineplot(
        data=working_df,
        x='year',
        y=var,
        hue='race_outcome',  # Separation by the categorical variable
        ax=ax,
        marker='o',
        errorbar='sd'  # Displaying dispersion (standard deviation)
    )
    ax.set_title(label, fontsize=12)
    ax.set_xlabel('Year', fontsize=10)
    ax.set_ylabel(label, fontsize=10)
    ax.grid(True)

fig.tight_layout()
plt.show()

# Grouping data by year
yearly_data = working_df.groupby('year').agg({
    'qualifying_position': 'mean',
    'grid_position': custom_mean,
    'positionOrder': 'mean',
    'valid_time_set_in_q1': 'sum',
    'qualified_for_q2': 'sum',
    'qualified_for_q3': 'sum',
    'q1': custom_mean,
```

```
    'q2': custom_mean,
    'q3': custom_mean,
}).reset_index()

# Defining variables for plotting
plot_vars = {
    'qualifying_position': 'Avg. Qualifying Position',
    'grid_position': 'Avg. Grid Position',
    'positionOrder': 'Avg. Position Order',
    'valid_time_set_in_q1': 'Total Times with Valid Time in Q1',
    'qualified_for_q2': 'Total Times Qualified for Q2',
    'qualified_for_q3': 'Total Times Qualified for Q3',
    'q1': 'Avg. Fastest Lap in Q1',
    'q2': 'Avg. Fastest Lap in Q2',
    'q3': 'Avg. Fastest Lap in Q3'
}

# Creating time series with separation by race_outcome
fig, axes = plt.subplots(3, 3, figsize=(15, 12))  # Create a grid of plots
axes = axes.flatten()

for i, (var, label) in enumerate(plot_vars.items()):
    ax = axes[i]
    sns.lineplot(
        data=working_df,
        x='year',
        y=var,
        hue='race_outcome',  # Separation by the categorical variable
        ax=ax,
        marker='o',
        errorbar='sd'  # Displaying dispersion (standard deviation)
    )
    ax.set_title(label, fontsize=12)
    ax.set_xlabel('Year', fontsize=10)
    ax.set_ylabel(label, fontsize=10)
    ax.grid(True)

fig.tight_layout()
plt.show()

# Grouping data by year
yearly_data = working_df.groupby('year').agg({
    'age': 'mean',
    'years_experience': 'mean',
    'years_with_constructor': 'mean',
    'race_starts': 'mean'
}).reset_index()

# Defining variables for plotting
plot_vars = {
    'age': 'Avg. Age',
```

```
    'years_experience': 'Avg. Years of Experience',
    'years_with_constructor': 'Avg. Years with Current Constructor',
    'race_starts': 'Avg. Race Starts'
}

# Creating time series with separation by race_outcome
fig, axes = plt.subplots(3, 3, figsize=(15, 12))  # Create a grid of plots
axes = axes.flatten()

for i, (var, label) in enumerate(plot_vars.items()):
    ax = axes[i]
    sns.lineplot(
        data=working_df,
        x='year',
        y=var,
        hue='race_outcome',  # Separation by the categorical variable
        ax=ax,
        marker='o',
        errorbar='sd'  # Displaying dispersion (standard deviation)
    )
    ax.set_title(label, fontsize=12)
    ax.set_xlabel('Year', fontsize=10)
    ax.set_ylabel(label, fontsize=10)
    ax.grid(True)

# Removing unused plots
for j in range(len(plot_vars), len(axes)):
    fig.delaxes(axes[j])

fig.tight_layout()
plt.show()

"""Violin Plots"""

# Selecting numerical columns from the working_df
numeric_columns = working_df.select_dtypes(include=['float64', 'int64']).columns

# Creating a violin plot for each numerical column
for column in numeric_columns:
    fig = px.violin(
        working_df,
        x='race_outcome',         # Categories
        y=column,                 # The current numerical column
        color='race_outcome',     # Coloring by category
        box=True,                 # Adding boxplot inside the violin plot
        hover_data=working_df.columns,  # Display data on hover
        width=1000,               # Width of the plot
        category_orders={         # Setting the category order in the plot
            'race_outcome': ['Podium finish', 'Points without podium', 'No points']
        }
    )
```

```
    # Adjusting title and axis labels
    fig.update_layout(
        title=f'{column} by Race Outcome',
        xaxis_title='Race Outcome',
        yaxis_title=column,
        violinmode='group',      # Sorting by group
        font=dict(size=12)       # Font size
    )

    # Display the plot
    fig.show()

# Pivot the dataset for measurements by category
grouped_data = working_df.groupby(['constructor_nationality',
'race_outcome']).size().reset_index(name='counts')

# Plot with Plotly (Grouped Bar Chart)
fig = px.bar(
    grouped_data,
    x='constructor_nationality',       # X-axis: Constructor Nationality
    y='counts',                        # Y-axis: Count of occurrences
    color='race_outcome',              # Color differentiation by Race Outcome
    barmode='group',                   # Grouped Bar Plot
    title='Comparison of Constructor Nationalities by Race Outcome', # Title of the chart
    labels={'constructor_nationality': 'Constructor Nationality', 'counts': 'Count',
'race_outcome': 'Race Outcome'}, # Axis labels
    width=900,
    height=600
)

# Layout customization
fig.update_layout(
    xaxis=dict(tickangle=45),          # Rotate X-axis labels by 45 degrees
    font=dict(size=12),                # Set font size for readability
    legend=dict(title='Race Outcome')    # Add a title to the legend
)

# Show the plot
fig.show()

# Pivot the dataset to count occurrences by driver nationality and race outcome
grouped_data = working_df.groupby(['driver_nationality',
'race_outcome']).size().reset_index(name='counts')

# Create a grouped bar chart using Plotly
fig = px.bar(
    grouped_data,
    x='driver_nationality',       # X-axis: Driver Nationality
    y='counts',                   # Y-axis: Count of occurrences
    color='race_outcome',         # Color the bars by race outcome
```

```
    barmode='group',              # Group the bars by nationality and outcome
    title='Comparison of Driver Nationality by Race Outcome',  # Chart title
    labels={'driver_nationality': 'Driver Nationality', 'counts': 'Count', 'race_outcome': 'Race Outcome'},  # Axis labels
    width=900,                    # Set the width of the plot
    height=600                    # Set the height of the plot
)

# Customize the layout of the plot
fig.update_layout(
    xaxis=dict(tickangle=45),             # Rotate the X-axis labels by 45 degrees for better readability
    font=dict(size=12),             # Set the font size for labels and title
    legend=dict(title='Race Outcome')  # Set the title for the legend
)

# Display the plot
fig.show()

# Group the data by constructor and race outcome and count the occurrences
grouped_data = working_df.groupby(['constructor_name', 'race_outcome']).size().reset_index(name='counts')

# Create a grouped bar chart using Plotly
fig = px.bar(
    grouped_data,
    x='constructor_name',       # X-axis: Constructor Names
    y='counts',               # Y-axis: Count of occurrences
    color='race_outcome',       # Color the bars by race outcome
    barmode='group',            # Group the bars by constructor and outcome
    title='Comparison of Constructor by Race Outcome',  # Chart title
    labels={'constructor_name': 'Constructor', 'counts': 'Count', 'race_outcome': 'Race Outcome'},  # Axis labels
    width=900,                # Set the width of the plot
    height=600                # Set the height of the plot
)

# Customize the layout of the plot
fig.update_layout(
    xaxis=dict(tickangle=45),             # Rotate the X-axis labels by 45 degrees for better readability
    font=dict(size=12),             # Set the font size for labels and title
    legend=dict(title='Race Outcome')  # Set the title for the legend
)

# Display the plot
fig.show()

# Pivot the dataset to get counts by top constructor and race outcome
grouped_data = working_df.groupby(['top_constructor', 'race_outcome']).size().reset_index(name='counts')
```

```
# Create a grouped bar chart using Plotly
fig = px.bar(
    grouped_data,
    x='top_constructor',              # X-axis: Constructor Names
    y='counts',                       # Y-axis: Count of occurrences
    color='race_outcome',             # Color the bars by race outcome
    barmode='group',                  # Group the bars by constructor and outcome
    title='Comparison of Constructor with most Experience of each Driver per Race by Race Outcome',  # Chart title
    labels={'top_constructor': 'Constructor', 'counts': 'Count', 'race_outcome': 'Race Outcome'},  # Axis labels
    width=900,                        # Set the width of the plot
    height=600                        # Set the height of the plot
)

# Customize the layout of the plot
fig.update_layout(
    xaxis=dict(tickangle=45),         # Rotate the X-axis labels by 45 degrees for better readability
    font=dict(size=12),               # Set the font size for labels and title
    legend=dict(title='Race Outcome')    # Set the title for the legend
)

# Display the plot
fig.show()

# Pivot the dataset to get counts by circuit and race outcome
grouped_data = working_df.groupby(['circuit_name', 'race_outcome']).size().reset_index(name='counts')

# Create a grouped bar chart using Plotly
fig = px.bar(
    grouped_data,
    x='circuit_name',              # X-axis: Circuit names
    y='counts',                    # Y-axis: Count of occurrences
    color='race_outcome',          # Color the bars by race outcome
    barmode='group',               # Group the bars by circuit and outcome
    title='Comparison of Circuit by Race Outcome',  # Chart title
    labels={'circuit_name': 'Circuit', 'counts': 'Count', 'race_outcome': 'Race Outcome'},  # Axis labels
    width=900,                     # Set the width of the plot
    height=600                     # Set the height of the plot
)

# Customize the layout of the plot
fig.update_layout(
    xaxis=dict(tickangle=45),      # Rotate the X-axis labels by 45 degrees for better readability
    font=dict(size=12),            # Set the font size for labels and title
    legend=dict(title='Race Outcome')  # Set the title for the legend
)
```

```
# Display the plot
fig.show()

# Pivot the dataset to get counts by race status and race outcome
grouped_data = working_df.groupby(['race_status', 'race_outcome']).size().reset_index(name='counts')

# Create a grouped bar chart using Plotly
fig = px.bar(
    grouped_data,
    x='race_status',              # X-axis: Race status categories
    y='counts',                   # Y-axis: Count of occurrences
    color='race_outcome',         # Color the bars by race outcome
    barmode='group',              # Group the bars by race status and outcome
    title='Comparison of Race Status by Race Outcome',  # Chart title
    labels={'race_status': 'Race Status', 'counts': 'Count', 'race_outcome': 'Race Outcome'},  # Axis labels
    width=900,                    # Set the width of the plot
    height=600                    # Set the height of the plot
)

# Customize the layout of the plot
fig.update_layout(
    xaxis=dict(tickangle=45),     # Rotate the X-axis labels by 45 degrees for better readability
    font=dict(size=12),           # Set the font size for labels and title
    legend=dict(title='Race Outcome')  # Set the title for the legend
)

# Display the plot
fig.show()

# Pivot the dataset to get counts by driver name and race outcome
grouped_data = working_df.groupby(['driver_name', 'race_outcome']).size().reset_index(name='counts')

# Create a grouped bar chart using Plotly
fig = px.bar(
    grouped_data,
    x='driver_name',              # X-axis: Driver names
    y='counts',                   # Y-axis: Count of occurrences
    color='race_outcome',         # Color the bars by race outcome
    barmode='group',              # Group the bars by race outcome
    title='Comparison of Driver by Race Outcome',  # Chart title
    labels={'driver_name': 'Driver', 'counts': 'Count', 'race_outcome': 'Race Outcome'},  # Axis labels
    width=1300,                   # Set the width of the plot
    height=600                    # Set the height of the plot
)

# Customize the layout of the plot
```

```
fig.update_layout(
    xaxis=dict(tickangle=45),       # Rotate the X-axis labels by 45 degrees for better readability
    font=dict(size=10),             # Set the font size for labels and title
    legend=dict(title='Race Outcome')  # Set the title for the legend
)

# Display the plot
fig.show()

# List of specific drivers to include
selected_drivers = ['Lewis Hamilton', 'Sebastian Vettel', 'Max Verstappen', 'Sergio Perez', 'Carlos Sainz', 'Fernando Alonso']

# Filter the dataset to include only the selected drivers
filtered_data = grouped_data[grouped_data['driver_name'].isin(selected_drivers)]

# Create a grouped bar chart using Plotly for the filtered dataset
fig = px.bar(
    filtered_data,
    x='driver_name',              # X-axis: Driver names
    y='counts',                   # Y-axis: Count of occurrences
    color='race_outcome',          # Color the bars by race outcome
    barmode='group',              # Group the bars by race outcome
    title='Comparison of Selected Drivers by Race Outcome',  # Chart title
    labels={'driver_name': 'Driver', 'counts': 'Count', 'race_outcome': 'Race Outcome'},  # Axis labels
    width=800,                    # Set the width of the plot
    height=600                    # Set the height of the plot
)

# Customize the layout of the plot
fig.update_layout(
    font=dict(size=12),             # Set the font size for labels and title
    legend=dict(title='Race Outcome')  # Set the title for the legend
)

# Display the plot
fig.show()

"""Fastest Laps by Circuits"""

import matplotlib.pyplot as plt
import seaborn as sns

# Group by 'circuit_name' and 'year' to calculate the mean fastest lap time
circuit_data = working_df.groupby(['circuit_name', 'year']).agg({
    'fastest_lap_time': 'mean'
}).reset_index()

# Get a list of unique circuits
circuits = circuit_data['circuit_name'].unique()
```

```
# Define figure size based on the number of circuits
fig, axes = plt.subplots(len(circuits), 1, figsize=(12, 4 * len(circuits)), sharex=True)

# If there's only one circuit, make axes a list for consistency
if len(circuits) == 1:
    axes = [axes]

# Plot fastest lap times for each circuit
for ax, circuit in zip(axes, circuits):
    data = circuit_data[circuit_data['circuit_name'] == circuit]
    sns.lineplot(data=data, x='year', y='fastest_lap_time', ax=ax, marker='o')
    ax.set_title(f"Avg. Fastest Lap Times - {circuit}", fontsize=12)
    ax.set_xlabel('Year', fontsize=10)
    ax.set_ylabel('Avg. Fastest Lap (s)', fontsize=10)
    ax.grid(True)

plt.tight_layout()
plt.show()

import matplotlib.pyplot as plt
import seaborn as sns

# List of selected circuits
selected_circuits = [
    "Albert Park Grand Prix Circuit",
    "Autodromo Nazionale di Monza",
    "Bahrain International Circuit",
    "Circuit de Barcelona-Catalunya",
    "Circuit de Monaco",
    "Circuit de Spa-Francorchamps",
    "Circuit of the Americas",
    "Hungaroring",
    "Marina Bay Street Circuit",
    "Silverstone Circuit"
]

# Filter data for selected circuits and years up to 2023
filtered_data = working_df[
    (working_df['circuit_name'].isin(selected_circuits)) &
    (working_df['year'] <= 2023)
]

# Group by 'circuit_name' and 'year' to calculate the mean fastest lap time
circuit_data = filtered_data.groupby(['circuit_name', 'year']).agg({
    'fastest_lap_time': 'mean'
}).reset_index()

# Get the number of selected circuits
num_circuits = len(selected_circuits)
```

```
# Set up a grid with 2 columns
fig, axes = plt.subplots((num_circuits + 1) // 2, 2, figsize=(14, 3 * (num_circuits // 2)),
sharex=True)

# Flatten axes for easy iteration
axes = axes.flatten()

# Plot fastest lap times for each selected circuit
for ax, circuit in zip(axes, selected_circuits):
    data = circuit_data[circuit_data['circuit_name'] == circuit]
    sns.lineplot(data=data, x='year', y='fastest_lap_time', ax=ax, marker='o')
    ax.set_title(f"{circuit}", fontsize=12)
    ax.set_xlabel('Year', fontsize=9)
    ax.set_ylabel('Avg. Fastest Lap (s)', fontsize=9)
    ax.grid(True)

# Remove any empty subplots (if the number of circuits is odd)
for i in range(len(selected_circuits), len(axes)):
    fig.delaxes(axes[i])

plt.tight_layout()
plt.show()

"""Chapter 4: Hypothesis Testing and Correlation"""

# Read the created working excel
working_df = pd.read_excel("working_f1_data_V5.xlsx")

"""Kolmogorov-Smirnov Normality Test"""

# List of numeric variables
numeric_variables = working_df.select_dtypes(include=['float64', 'int64']).columns

# Prepare the plot layout for 6x4 grid of histograms and QQ plots (in separate axes)
fig, axes = plt.subplots(6, 4, figsize=(18, 24))
fig_qq, axes_qq = plt.subplots(6, 4, figsize=(18, 24))

# Flatten axes for easy indexing
axes = axes.flatten()
axes_qq = axes_qq.flatten()

# Initialize a list to store KS statistics and p-values
ks_stats = []
ks_pvalues = []

# Loop through each numeric variable to generate plots and calculate KS test
for idx, var in enumerate(numeric_variables):
    data = working_df[var]

    # Kolmogorov-Smirnov test (for normality)
    stat_ks, p_value_ks = stats.kstest(data, 'norm')
```

```
    # Store the KS statistic and p-value for later use
    ks_stats.append(stat_ks)
    ks_pvalues.append(p_value_ks)

    # Histogram
    sns.histplot(data, kde=True, color='skyblue', ax=axes[idx])

    # Q-Q Plot
    stats.probplot(data, dist="norm", plot=axes_qq[idx])
    axes_qq[idx].set_title(f'Q-Q Plot of {var}')

    # Tighten layout for histograms and QQ plots
    plt.tight_layout()

# Create a board to display the KS statistics and p-values with more decimals
fig_ks, ax_ks = plt.subplots(figsize=(12, 6))
ax_ks.axis('tight')
ax_ks.axis('off')

# Create a DataFrame to display KS statistic and p-value
ks_values_df = pd.DataFrame({
    'Variable': numeric_variables,
    'KS Statistic': [f'{stat:.3f}' for stat in ks_stats],
    'KS p-value': [f'{p_val:.3f}' for p_val in ks_pvalues]
})

# Display the KS statistic and p-value table
ax_ks.table(cellText=ks_values_df.values,    colLabels=ks_values_df.columns,    loc='center',
cellLoc='center')

plt.show()

#Checking Qualy times without zeros
variables_to_check = ['q1', 'q2', 'q3']

# Create figure layouts for histograms and Q-Q plots
fig, axes = plt.subplots(1,len(variables_to_check), figsize=(16, 4))
fig_qq, axes_qq = plt.subplots(1,len(variables_to_check), figsize=(16,4))

# Ensure axes are iterable even if there is one variable
if len(variables_to_check) == 1:
    axes = [axes]
    axes_qq = [axes_qq]
else:
    axes = axes.flatten()
    axes_qq = axes_qq.flatten()

# Initialize lists for KS statistics and p-values
ks_stats = []
ks_pvalues = []
```

```
# Loop through each variable separately
for idx, var in enumerate(variables_to_check):
    data = working_df[var]

    # Remove only the zero values for this specific variable
    data_no_zeros = data[data != 0]

    # Kolmogorov-Smirnov test (for normality)
    stat_ks, p_value_ks = stats.kstest(data_no_zeros, 'norm')

    # Store KS statistic and p-value
    ks_stats.append(stat_ks)
    ks_pvalues.append(p_value_ks)

    # Histogram
    sns.histplot(data_no_zeros, kde=True, color='skyblue', ax=axes[idx])
    axes[idx].set_title(f'Histogram of {var} (Zeros Removed)')

    # Q-Q Plot
    stats.probplot(data_no_zeros, dist="norm", plot=axes_qq[idx])
    axes_qq[idx].set_title(f'Q-Q Plot of {var} (Zeros Removed)')

# Adjust layout for better visualization
plt.tight_layout()

# Display KS statistics in a table
fig_ks, ax_ks = plt.subplots(figsize=(8, 2))
ax_ks.axis('tight')
ax_ks.axis('off')

# Create DataFrame to display KS statistic and p-value
ks_values_df = pd.DataFrame({
    'Variable': variables_to_check,
    'KS Statistic': [f'{stat:.3f}' for stat in ks_stats],
    'KS p-value': [f'{p_val:.3f}' for p_val in ks_pvalues]
})

# Display KS statistic and p-value table
ax_ks.table(cellText=ks_values_df.values,    colLabels=ks_values_df.columns,    loc='center',
cellLoc='center')

plt.show()

"""Kolmogorov-Smirnov Normality Test Groupped"""

# List of numeric variables
numeric_variables = ['age', 'years_experience', 'race_starts', 'years_with_constructor',
                     'q1', 'q2', 'q3', 'qualifying_position', 'grid_position',
                     'positionOrder', 'driver_points', 'cumulative_season_driver_points',
                     'fastest_lap_time', 'laps', 'total_pit_stops', 'total_duration_ms',
```

```
                'top_3_positions', 'positions_4_10', 'outside_10_positions']

# Define the Race Outcome categories
race_outcomes = ['Podium finish', 'Points without podium', 'No points']

# Loop through each Race Outcome category
for outcome in race_outcomes:
    # Filter the data for the specific Race Outcome
    subgroup_data = working_df[working_df['race_outcome'] == outcome]

    # Initialize lists to store K-S statistics and p-values
    ks_stats = []
    ks_pvalues = []

    # Perform K-S test for each numeric variable
    for var in numeric_variables:
        data = subgroup_data[var]

        # Kolmogorov-Smirnov test (for normality)
        stat_ks, p_value_ks = stats.kstest(data, 'norm')

        # Append results
        ks_stats.append(stat_ks)
        ks_pvalues.append(p_value_ks)

    # Print the results in a text format
    print(f"K-S Test Results for Race Outcome: {outcome}")
    print(f"{'Variable':<30}{'KS Statistic':<15}{'KS p-value':<15}")
    print("-" * 60)
    for i, var in enumerate(numeric_variables):
        print(f"{var:<30}{ks_stats[i]:<15.3f}{ks_pvalues[i]:<15.3f}")
    print("\n")

"""Correlations"""

# Selecting only the numeric variables
numeric_variables = ['age', 'years_experience', 'race_starts', 'years_with_constructor',
                'q1', 'q2', 'q3', 'qualifying_position', 'grid_position',
                'driver_points', 'cumulative_season_driver_points',
                'fastest_lap_time', 'laps', 'total_pit_stops', 'total_duration_ms',
                'top_3_positions', 'positions_4_10', 'outside_10_positions']

# Subset the DataFrame to only include the numeric variables
numeric_df = working_df[numeric_variables]

# Calculate the correlation matrix using Spearman correlation
correlation_matrix = numeric_df.corr(method='spearman')

# Plotting the heatmap
plt.figure(figsize=(16, 12))
```

```
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5,
vmin=-1, vmax=1)
plt.title("Spearman Correlation Heatmap of Numeric Variables")
plt.show()

"""Correlations Grouped"""

# Selecting only the numeric variables
numeric_variables = ['age', 'years_experience', 'race_starts', 'years_with_constructor',
                     'q1', 'q2', 'q3', 'qualifying_position', 'grid_position',
                     'cumulative_season_driver_points',
                     'fastest_lap_time', 'laps', 'total_pit_stops', 'total_duration_ms',
                     'top_3_positions', 'positions_4_10', 'outside_10_positions', 'driver_points']

# Add the Race Outcome variable to the DataFrame
working_df = working_df.copy()
working_df['race_outcome'] = working_df['race_outcome']

# Subset the DataFrame to only include the numeric variables
numeric_df = working_df[numeric_variables]

# Iterate over each unique category in the Race Outcome variable
race_outcome_categories = working_df['race_outcome'].unique()

for outcome in race_outcome_categories:
    # Filter the DataFrame for the current category
    subset_df = numeric_df[working_df['race_outcome'] == outcome]

    # Calculate the Spearman correlation matrix for this subset
    correlation_matrix = subset_df.corr(method='spearman')

    # Plotting the heatmap
    plt.figure(figsize=(16, 12))
    sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f",
linewidths=0.5, vmin=-1, vmax=1)
    plt.title(f"Spearman Correlation Heatmap for Race Outcome: {outcome}")
    plt.show()

"""Chapter 5: Regression"""

# Read the created working excel
working_df = pd.read_excel("working_f1_data_V5.xlsx")

# Load Data
data = working_df
data.head()

"""Grouping Circuits and Nationalities"""

print(data['circuit_name'].unique())
```

```
circuit_df = pd.read_excel("circuits_overtakes.xlsx")
circuit_df.head()

circuit_df["Avg_Overtakes"] =
circuit_df["Avg_Overtakes"].fillna(circuit_df["Avg_Overtakes"].mean())
circuit_df.head()

# Define bins and labels
bins = [0, 24.99, 49.99, 74.99, 100]  # 4 groups
labels = [1, 2, 3, 4]

# Create a new column for grouping
circuit_df["Overtake_Group"] = pd.cut(circuit_df["Avg_Overtakes"], bins=bins,
labels=labels, include_lowest=True)

circuit_df.head()

# Save to an Excel file
circuit_df.to_excel("circuits_overtakesV5.xlsx", index=False)

print(data['driver_nationality'].unique())

nationalities_df = pd.read_excel("Nationalities.xlsx")
nationalities_df.head()

"""Encoding"""

# List with categorical values for Encoding
categorical_columns = ['driver_name', 'constructor_name','constructor_nationality',
                       'top_constructor', 'race_status'
]

# Applying LabelEncoder in each column
for col in categorical_columns:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])

data.head()

# Define the label mapping from your custom labels to numeric values
outcome_labels = {"Podium finish": 0, "Points without podium": 1, "No points": 2}

# Apply the mapping to the 'race_outcome' column
data['race_outcome'] = data['race_outcome'].map(outcome_labels)

# Check the result
print(data['race_outcome'])

data['circuit_name'] = data['circuit_name'].map(circuit_df.set_index('Circuit
Name')['Overtake_Group'])
data['circuit_name'].head()
```

```
data['driver_nationality'] = data['driver_nationality'].map(nationalities_df.set_index('Driver Nationalities')['European'])
data['driver_nationality'].head()

"""Early Testing"""

# # Select Variables for x
x = data.iloc[:,0:31]

# Select Outcome as target
y = data.iloc[:,31]

# Create a 75% random split of data for training/testing
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.75, random_state=1234)

# Initialize Logistic Regression model
log_reg_model = LogisticRegression(class_weight='balanced', solver='lbfgs', random_state=1234, max_iter=100)

# Fit the model
log_reg_model.fit(x_train, y_train)

# Predict on the test set
y_pred = log_reg_model.predict(x_test)

# Evaluate model performance
accuracy_train = accuracy_score(y_train, log_reg_model.predict(x_train))
accuracy_test = accuracy_score(y_test, y_pred)

# Classification Report
class_report = classification_report(y_test, y_pred)

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Calculate balanced accuracy
balanced_acc = balanced_accuracy_score(y_test, log_reg_model.predict(x_test))

# Get probabilities for each class
y_proba = log_reg_model.predict_proba(x_test)
# Compute ROC-AUC (one-vs-rest)
roc_auc = roc_auc_score(y_test, y_proba, multi_class='ovr')

# Print results
print("Train Accuracy for Logistic Regression: {:.4f}".format(accuracy_train))
print("Test Accuracy for Logistic Regression: {:.4f}".format(accuracy_test))
print("\nClassification Report:")
print(class_report)
print("\nConfusion Matrix:")
print(conf_matrix)
```

```
print("\nBalanced Accuracy: {:.4f}".format(balanced_acc))
print("ROC-AUC Score: {:.4f}".format(roc_auc))

"""VIF"""

# Calculating VIF
vif_data = pd.DataFrame()
vif_data["feature"] = data.columns[:31]
vif_data["VIF"] = [variance_inflation_factor(x, i) for i in range(x.shape[1])]

print(vif_data)

# Select Variables for x
x = data.iloc[:,[5,6,7,18,22,23,26,27,28,30]]

# Calculating VIF
selected_columns = [5,6,7,18,22,23,26,27,28,30]
vif_data = pd.DataFrame()
vif_data["feature"] = data.columns[selected_columns]
vif_data["VIF"] = [variance_inflation_factor(x, i) for i in range(x.shape[1])]

print(vif_data)

"""Correlations as is"""

# Selecting the variables
variables = ['constructor_name', 'constructor_nationality','years_with_constructor',
'qualifying_position',
             'driver_points','cumulative_season_driver_points','total_pit_stops',
             'total_duration_ms', 'top_3_positions','outside_10_positions']

# Subset the DataFrame to only include the numeric variables
numeric_df = working_df[variables]

# Calculate the correlation matrix using Spearman correlation
correlation_matrix = numeric_df.corr(method='spearman')

# Plotting the heatmap
plt.figure(figsize=(16, 12))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5,
vmin=-1, vmax=1)
plt.title("Spearman Correlation Heatmap of Numeric Variables")
plt.show()

#Variables after Correlation investigating
x = data.iloc[:,[5,6,7,18,23,26,30]]

#Final Variables after Wald checks AIC and BIC compairing etc
x = data.iloc[:,[7,18,23,26]]

# Calculating VIF
```

```
selected_columns = [7,18,23,26]
vif_data = pd.DataFrame()
vif_data["feature"] = data.columns[selected_columns]
vif_data["VIF"] = [variance_inflation_factor(x, i) for i in range(x.shape[1])]

print(vif_data)

"""Multinomial Final"""

# Split data into training and testing sets (75% training, 25% testing)
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.75, random_state=1234)

# Fit MNLogit model using statsmodels
x_train_with_const = sm.add_constant(x_train)
x_test_with_const = sm.add_constant(x_test)

mnlogit_model = sm.MNLogit(y_train, x_train_with_const)
mnlogit_result = mnlogit_model.fit(method='newton')

# Predict on test set
y_pred_proba = mnlogit_result.predict(x_test_with_const)  # Predicted probabilities
y_pred = y_pred_proba.idxmax(axis=1)  # Convert probabilities to class predictions

# Metrics Calculation
accuracy_train = accuracy_score(y_train, mnlogit_result.predict(x_train_with_const).idxmax(axis=1))
accuracy_test = accuracy_score(y_test, y_pred)
balanced_acc = balanced_accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# ROC-AUC Calculation (One-vs-Rest)
roc_auc = roc_auc_score(y_test, y_pred_proba, multi_class='ovr')

# Get AIC and BIC from the fitted model
aic = mnlogit_result.aic
bic = mnlogit_result.bic

# Print AIC and BIC scores
print("AIC: {:.4f}".format(aic))
print("BIC: {:.4f}".format(bic))
# Print model summary
print(mnlogit_result.summary())

# Print metrics
print(f"Train Accuracy: {accuracy_train:.4f}")
print(f"Test Accuracy: {accuracy_test:.4f}")
print(f"Balanced Accuracy: {balanced_acc:.4f}")
print(f"ROC-AUC Score: {roc_auc:.4f}")
print("\nClassification Report:")
print(class_report)
```

```
# Visualize Confusion Matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()


# ROC Curve for each class
y_pred_proba = pd.DataFrame(mnlogit_result.predict(sm.add_constant(x_test)))
# Loop through each class and compute ROC curve
plt.figure(figsize=(8, 6))
for i in range(y_pred_proba.shape[1]):  # Iterate through available classes
    fpr, tpr, _ = roc_curve(y_test, y_pred_proba.iloc[:, i], pos_label=i)
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f"Class {i} (AUC = {roc_auc:.2f})")
# Add diagonal line and labels
plt.plot([0, 1], [0, 1], 'k--')  # Random guess line
plt.title("ROC Curve for Multinomial Logistic Regression")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="best")
plt.grid()
plt.show()

"""Learning Curve"""

from sklearn.model_selection import learning_curve
import numpy as np
import matplotlib.pyplot as plt

# Define the model
log_reg_model = LogisticRegression()

# Generate learning curve data
train_sizes, train_scores, test_scores = learning_curve(
    log_reg_model, x, y, cv=5, scoring='accuracy', n_jobs=-1, train_sizes=np.linspace(0.1, 1.0,
10)
)

# Calculate mean and standard deviation of train and test scores
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

# Plot learning curve
plt.figure(figsize=(6, 4))
```

```
plt.plot(train_sizes, train_scores_mean, label="Training Score", color="blue")
plt.fill_between(train_sizes, train_scores_mean - train_scores_std, train_scores_mean + train_scores_std, color="blue", alpha=0.2)
plt.plot(train_sizes, test_scores_mean, label="Validation Score", color="green")
plt.fill_between(train_sizes, test_scores_mean - test_scores_std, test_scores_mean + test_scores_std, color="green", alpha=0.2)
plt.title("Learning Curve (Logistic Regression)")
plt.xlabel("Training Set Size")
plt.ylabel("Accuracy")
plt.legend(loc="best")
plt.grid()
plt.show()

"""Residuals"""

# Extract predicted probabilities for the actual classes
y_test_numeric = y_test.astype(int).to_numpy()  # Convert to numpy array for indexing
# y_pred_proba_actual = y_pred_proba[np.arange(len(y_test_numeric)), y_test_numeric]
y_pred_proba_actual = y_pred_proba.values[np.arange(len(y_test_numeric)), y_test_numeric] # Access the underlying NumPy array

# Compute Pearson Residuals
pearson_residuals = (y_test_numeric - y_pred_proba_actual) / np.sqrt(y_pred_proba_actual * (1 - y_pred_proba_actual))

# --- Q-Q Plot (Check Normality) ---
sm.qqplot(pearson_residuals, line='s')
plt.title("Q-Q Plot of Pearson Residuals")
plt.show()

# --- Scatter Plot (Check Independence) ---
plt.figure(figsize=(5, 3))
plt.scatter(np.arange(len(pearson_residuals)), pearson_residuals, alpha=0.5, color="blue")
plt.xlabel("Sample Index")
plt.ylabel("Pearson Residuals")
plt.title("Residuals Scatter Plot")
plt.legend()
plt.show()

"""LassoCV"""

#LassoCV auto selecting features
x = data.iloc[:,[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,21,22,23,24,25,26,27,28,29,30]]
def select_best_variables_lasso(x, y, penalty_strengths=np.logspace(-4, 1, 30), cv=5):

    # Standardize features
    scaler = StandardScaler()
    x_scaled = scaler.fit_transform(x)

    # Lasso-based logistic regression with cross-validation
```

```
    lasso = LogisticRegressionCV(
        Cs=penalty_strengths,
        cv=5,
        penalty="l1",
        solver="saga",  # Faster than liblinear
        multi_class="multinomial",
        max_iter=1000,
        n_jobs=-1  # Use all available CPU cores

    )

    # Fit model
    lasso.fit(x_scaled, y)

    # Identify non-zero coefficients (selected variables)
    selected_mask = np.any(lasso.coef_ != 0, axis=0)   # Keeps features with non-zero coefficients
    selected_features = x.columns[selected_mask]
    removed_features = x.columns[~selected_mask]  # Features removed by Lasso

    print("\nSelected Features for MNL Model:")
    print(selected_features.tolist())

    print("\nRemoved Features (Lasso Shrunk to Zero):")
    print(removed_features.tolist())

    # Return reduced dataset with selected features
    return x[selected_features], removed_features.tolist()

# Example Usage:
x_selected_lasso, removed_features_lasso = select_best_variables_lasso(x, y)

"""Chapter 6: Timeseries"""

# Load necessary datasets
results = pd.read_csv("results.csv")  # Contains fastest lap times
races = pd.read_csv("races.csv")  # Contains race locations & years
circuits = pd.read_csv("circuits.csv")  # Contains circuit details

# Merge races with circuits
races = races.merge(circuits, on="circuitId")

# Merge results with races to get circuit information
df = results.merge(races, on="raceId")

#AICc Calculation
def calculate_aicc(aic, k, n):
    """Calculate AICc (Corrected AIC) for small sample sizes."""
    return aic + (2 * k * (k + 1)) / (n - k - 1)

"""ALBERT PARK"""
```

```
# Keep only Albert Park races
albert_park_races = df[df["circuitRef"] == "albert_park"]

# Convert fastest lap time to seconds
def time_to_seconds(time_str):
    try:
        m, s = map(float, time_str.split(":"))
        return m * 60 + s
    except:
        return None  # Handle missing or incorrect values

albert_park_races["fastestLapTime_seconds"] =
albert_park_races["fastestLapTime"].apply(time_to_seconds)

# Compute average fastest lap time per race (year)
tsdf = albert_park_races.groupby("year")["fastestLapTime_seconds"].mean().reset_index()
tsdf = tsdf[tsdf["year"] >= 2004].copy()

# TS plot, ACF, PACF
fig, axes = plt.subplots(1, 3, figsize=(25, 8))
tsdf["fastestLapTime_seconds"].plot(title="Albert Park Avg Fastest Lap Times Over Time",
ax=axes[0])
plot_acf(tsdf["fastestLapTime_seconds"], ax=axes[1])
plot_pacf(tsdf["fastestLapTime_seconds"], ax=axes[2])
plt.show()

# ADF Test
result = adfuller(tsdf["fastestLapTime_seconds"])
print(f"ADF Statistic: {result[0]}")
print(f"p-value: {result[1]}")

if result[1] < 0.05:
    print("Time series is stationary.")
else:
    print("Time series is NOT stationary.")

# 1st Diff
tsdf["lap_time_diff"] = tsdf["fastestLapTime_seconds"].diff()

# Drop the first NaN row
tsdf.dropna(inplace=True)

# ADF Test
result = adfuller(tsdf["lap_time_diff"])
print(f"ADF Statistic: {result[0]}")
print(f"p-value: {result[1]}")

if result[1] < 0.05:
    print("Time series is now stationary.")
else:
```

```
    print("Still not stationary.")

# 2nd Diff
tsdf["lap_time_diff2"] = tsdf["lap_time_diff"].diff()
tsdf.dropna(inplace=True)

# ADF Test Again
result = adfuller(tsdf["lap_time_diff2"])
print(f"ADF Statistic: {result[0]}")
print(f"p-value: {result[1]}")

# TS plot, ACF, PACF
fig, axes = plt.subplots(1, 3, figsize=(25, 8))
tsdf["lap_time_diff2"].plot(title="Albert Park 2nd Diff Over Time", ax=axes[0])
plot_acf(tsdf["lap_time_diff2"], ax=axes[1])
plot_pacf(tsdf["lap_time_diff2"], ax=axes[2])
plt.show()

# Define ARIMA models to compare
models = [(1,0,1), (2,0,1), (1,0,2), (2,0,2)]
results = []

for order in models:
    model = sm.tsa.ARIMA(tsdf["lap_time_diff2"], order=order)
    fit = model.fit()

    # Number of parameters (p + d + q + 1 for variance)
    k = sum(order) + 1
    n = len(tsdf["lap_time_diff2"])

    results.append({
        "Model": f"ARIMA{order}",
        "AIC": fit.aic,
        "BIC": fit.bic,
        "AICc": calculate_aicc(fit.aic, k, n)
    })

# Convert results to a DataFrame and display
Results = pd.DataFrame(results)
print(Results)

# Residual Testing
fit = sm.tsa.ARIMA(tsdf["lap_time_diff2"], order=(1, 0, 1)).fit()
residuals = fit.resid

fig, axes = plt.subplots(1, 3, figsize=(15, 4))

# Q-Q plot
sm.qqplot(residuals, line="s", ax=axes[0])
axes[0].set_title("Q-Q Plot of Residuals")
# Residuals over time
```

```
axes[1].plot(residuals)
axes[1].set_title("Residuals Over Time")

# Autocorrelation plot
plot_acf(residuals, ax=axes[2])
axes[2].set_title("Residuals ACF plot")
plt.show()

# Testing
# Residual Shapiro
stat, p_shapiro = shapiro(fit.resid)  # Residuals from Auto-ARIMA
print(f"Shapiro-Wilk p-value: {p_shapiro:.4f}")

# Res Anderson-Darling
result = anderson(fit.resid, dist='norm')
print(f"Anderson-Darling Test Statistic: {result.statistic:.4f}")
print(f"Critical Values: {result.critical_values}")
print(f"Significance Levels: {result.significance_level}")

# Box-Ljung Test
ljung_p_values = acorr_ljungbox(fit.resid, lags=[10], return_df=True)['lb_pvalue']
print(f"Box-Ljung p-value: {ljung_p_values.iloc[-1]:.4f}")

# Forecasting for the next 2 years
forecast = fit.forecast(steps=2)
conf_int = fit.get_forecast(steps=2).conf_int()

# Plot results
plt.figure(figsize=(10, 5))
plt.plot(tsdf["year"], tsdf["lap_time_diff2"], marker="o", label="Actual Data")
plt.plot(range(tsdf["year"].max() + 1, tsdf["year"].max() + 3), forecast, marker="o",
color="red", label="Forecast")
plt.fill_between(range(tsdf["year"].max() + 1, tsdf["year"].max() + 3), conf_int.iloc[:, 0],
conf_int.iloc[:, 1], color='pink', alpha=0.8)
plt.xlabel("Year")
plt.ylabel("Differenced Lap Time")
plt.title("ARIMA (1,0,1) Forecast for Albert Park")
plt.legend()
plt.show()

"""MONACO"""

# Keep only Monaco races
monaco_races = df[df["circuitRef"] == "monaco"]

# Convert fastest lap time to seconds
def time_to_seconds(time_str):
    try:
        m, s = map(float, time_str.split(":"))
        return m * 60 + s
    except:
```

```
        return None  # Handle missing or incorrect values

monaco_races["fastestLapTime_seconds"] =
monaco_races["fastestLapTime"].apply(time_to_seconds)

# Compute average fastest lap time per race (year)
tsdf = monaco_races.groupby("year")["fastestLapTime_seconds"].mean().reset_index()
tsdf = tsdf[(tsdf["year"] >= 2004) & (tsdf["year"] <= 2023)].copy()

# TS plot, ACF, PACF
fig, axes = plt.subplots(1, 3, figsize=(25, 8))
tsdf["fastestLapTime_seconds"].plot(title="Monaco Avg Fastest Lap Times Over Time",
ax=axes[0])
plot_acf(tsdf["fastestLapTime_seconds"], ax=axes[1])
plot_pacf(tsdf["fastestLapTime_seconds"], ax=axes[2])
plt.show()

# ADF Test
result = adfuller(tsdf["fastestLapTime_seconds"])
print(f"ADF Statistic: {result[0]}")
print(f"p-value: {result[1]}")

if result[1] < 0.05:
    print("Time series is stationary.")
else:
    print("Time series is NOT stationary.")

# 1st Diff
tsdf["lap_time_diff"] = tsdf["fastestLapTime_seconds"].diff()

# Drop the first NaN row
tsdf.dropna(inplace=True)

# ADF Test
result = adfuller(tsdf["lap_time_diff"])
print(f"ADF Statistic: {result[0]}")
print(f"p-value: {result[1]}")

if result[1] < 0.05:
    print("Time series is stationary.")
else:
    print("Time series is NOT stationary.")

# TS plot, ACF, PACF
fig, axes = plt.subplots(1, 3, figsize=(25, 8))
tsdf["lap_time_diff"].plot(title="Monaco 1st Diff Over Time", ax=axes[0])
plot_acf(tsdf["lap_time_diff"], ax=axes[1])
plot_pacf(tsdf["lap_time_diff"], ax=axes[2])
plt.show()

# Define ARIMA models to compare
```

```
models = [(1,0,0), (0,0,1), (1,0,1)]
results = []

for order in models:
    model = sm.tsa.ARIMA(tsdf["lap_time_diff"], order=order)
    fit = model.fit()

    # Number of parameters (p + d + q + 1 for variance)
    k = sum(order) + 1
    n = len(tsdf["lap_time_diff"])

    results.append({
        "Model": f"ARIMA{order}",
        "AIC": fit.aic,
        "BIC": fit.bic,
        "AICc": calculate_aicc(fit.aic, k, n)
    })

# Convert results to a DataFrame and display
Results = pd.DataFrame(results)
print(Results)

# Residual Testing
fit = sm.tsa.ARIMA(tsdf["lap_time_diff"], order=(1, 0, 0)).fit()
residuals = fit.resid

fig, axes = plt.subplots(1, 3, figsize=(15, 4))

# Q-Q plot
sm.qqplot(residuals, line="s", ax=axes[0])
axes[0].set_title("Q-Q Plot of Residuals")
# Residuals over time
axes[1].plot(residuals)
axes[1].set_title("Residuals Over Time")

# Autocorrelation plot
plot_acf(residuals, ax=axes[2])
axes[2].set_title("Residuals ACF plot")
plt.show()

# Residual Testing
# Residual Shapiro
stat, p_shapiro = shapiro(fit.resid)  # Residuals from Auto-ARIMA
print(f"Shapiro-Wilk p-value: {p_shapiro:.4f}")

# Res Anderson-Darling
result = anderson(fit.resid, dist='norm')
print(f"Anderson-Darling Test Statistic: {result.statistic:.4f}")
print(f"Critical Values: {result.critical_values}")
print(f"Significance Levels: {result.significance_level}")
```

```
# Box-Ljung Test
ljung_p_values = acorr_ljungbox(fit.resid, lags=[10], return_df=True)['lb_pvalue']
print(f"Box-Ljung p-value: {ljung_p_values.iloc[-1]:.4f}")

# Forecasting for the next 2 years
forecast = fit.forecast(steps=2)
conf_int = fit.get_forecast(steps=2).conf_int()

# Plot results
plt.figure(figsize=(10, 5))
plt.plot(tsdf["year"], tsdf["lap_time_diff"], marker="o", label="Actual Data")
plt.plot(range(tsdf["year"].max() + 1, tsdf["year"].max() + 3), forecast, marker="o",
color="red", label="Forecast")
plt.fill_between(range(tsdf["year"].max() + 1, tsdf["year"].max() + 3), conf_int.iloc[:, 0],
conf_int.iloc[:, 1], color='pink', alpha=0.8)
plt.xlabel("Year")
plt.ylabel("Differenced Lap Time")
plt.title("ARIMA (1,0,0) Forecast for Monaco")
plt.legend()
plt.show()

"""Chapter 7: Machine Learning"""

"""Scaling"""

data.head()

# Scaling for PCA
# Separate the features (X) from the target (y)
X = data.drop(columns=['driver_name','positionOrder','race_outcome'])
y = data['race_outcome']

# Initialize the scaler
scaler = StandardScaler()

# Fit and transform the X variables to scale them
X_scaled = scaler.fit_transform(X)

# Convert the scaled data back to a DataFrame (optional, for readability)
X_scaled = pd.DataFrame(X_scaled, columns=X.columns)

print(X_scaled.head())

"""PCA Feature Selection"""

# Initialize the PCA model
pca = PCA()

# Fit the PCA model to the scaled data
pca.fit(X_scaled)
```

```
# Transform the data into the principal components
X_pca = pca.transform(X_scaled)

# Explained variance ratio - shows the proportion of variance captured by each principal component
print("Explained Variance Ratio by each component:")
print(pca.explained_variance_ratio_)

# Cumulative explained variance - shows how much variance is explained by the first N components
print("\nCumulative Explained Variance:")
print(pca.explained_variance_ratio_.cumsum())

# Plot the explained variance to see how many components to keep
plt.figure(figsize=(8, 6))
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1), pca.explained_variance_ratio_.cumsum(), marker='o', linestyle='--')
plt.title('Cumulative Explained Variance by Principal Components')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.grid(True)
plt.show()

# Explain 95% of the variance
n_components = (pca.explained_variance_ratio_.cumsum() <= 0.95).sum()

print(f"\nNumber of components explaining 95% of the variance: {n_components}")

# Reduce the dataset to the first 10 principal components
X_pca_reduced = X_pca[:, :10]

# Convert back to a DataFrame for readability
X_pca_reduced_df = pd.DataFrame(X_pca_reduced, columns=[f'PC{i+1}' for i in range(10)])

#Top 3 Contributing Features for Each Principal Component

# Get the PCA components (loadings) from the PCA object
loadings = pca.components_

# Create a DataFrame of loadings with original feature names as columns
loadings_df = pd.DataFrame(loadings, columns=X.columns)

# Get the absolute value of the loadings to determine feature importance
loadings_abs = loadings_df.abs()

# For each principal component, get the top contributing features
top_features = {}
for i in range(loadings_abs.shape[0]):
    top_feature_indices = loadings_abs.iloc[i].nlargest(3).index  # Get top 3 features for each component
    top_features[f'PC{i+1}'] = top_feature_indices.tolist()
```

```
# Display the most significant features for each principal component with pprint
print("Top 3 Contributing Features for Each Principal Component:")
pprint(top_features)

"""Splitting the Data"""

# Split data into features (X) and target (y)
X = X_pca_reduced_df
y = data['race_outcome']

# Split data into 80% train and 20% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1234)

"""Classification with Random Forest"""

# Create the model
model = RandomForestClassifier(class_weight='balanced', random_state=1234)

# Train the model on the training data
model.fit(X_train, y_train)

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

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Classification Report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Confusion Matrix
print("Confusion Matrix:")

# Plot the heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=['Class 0', 'Class 1', 'Class
2'], yticklabels=['Class 0', 'Class 1', 'Class 2'],cbar=False)

# Labels, title, and display
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

# Cross-Validation for generalization check
```

```
from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(model, X_train, y_train, cv=5)
print(f'Mean CV Accuracy: {cv_scores.mean():.2f}')

"""Classification with SVM"""

# Create the model
svm_model = SVC(kernel='rbf', C=10, gamma = 0.01, random_state=1234)  # RBF kernel works well for non-linear data

# Train the model
svm_model.fit(X_train, y_train)

# Make predictions
y_pred_svm = svm_model.predict(X_test)

# Evaluate performance
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print(f'SVM Accuracy: {accuracy_svm:.2f}')

# Classification Report
print("Classification Report:")
print(classification_report(y_test, y_pred_svm))

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred_svm)

# Confusion Matrix
print("Confusion Matrix:")

# Plot the heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=['Class 0', 'Class 1', 'Class 2'], yticklabels=['Class 0', 'Class 1', 'Class 2'],cbar=False)

# Labels, title, and display
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

# Perform cross-validation
cv_scores = cross_val_score(svm_model, X_train, y_train, cv=5)  # 5-fold cross-validation

# Print cross-validation results
print(f'Mean CV Accuracy: {cv_scores.mean():.2f}')

"""Clustering with K-Means"""

# Elbow Method to determine the optimal number of clusters
distortions = []
```

```
K_range = range(1, 11)  # You can change this range based on the expected number of clusters

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=1234)
    kmeans.fit(X)
    distortions.append(kmeans.inertia_)

# Plotting the Elbow Curve
plt.figure(figsize=(8, 6))
plt.plot(K_range, distortions, marker='o')
plt.title('Elbow Method For Optimal k')
plt.xlabel('Number of Clusters')
plt.ylabel('Distortion')
plt.show()

# Number of clusters (k)
optimal_k = 3

# Perform K-Means with the chosen k
kmeans = KMeans(n_clusters=optimal_k, random_state=1234)
kmeans.fit(X)

# Get the cluster labels
cluster_labels = kmeans.labels_

# Add the cluster labels to your data for further analysis (optional)
X['Cluster'] = cluster_labels

# Print the cluster centers (optional)
print("Cluster Centers:")
print(kmeans.cluster_centers_)

# Print the first few rows of data with cluster labels
print(X.head())

# Silhouette Score
score = silhouette_score(X, kmeans.labels_)
print(f"Silhouette Score: {score}")

# Calculate Adjusted Rand Index
true_labels = data['race_outcome']
ari = adjusted_rand_score(true_labels, kmeans.labels_)
print(f"Adjusted Rand Index: {ari}")

# Calinski-Harabasz
ch_score = calinski_harabasz_score(X, kmeans.labels_)
print(f"Calinski-Harabasz Index: {ch_score}")

# Visualizing the Clusters using PCA
pca = PCA(n_components=2)
```

```
X_pca = pca.fit_transform(X_scaled)

# Create a DataFrame with the PCA results and cluster labels
pca_df = pd.DataFrame(X_pca, columns=['PC1', 'PC2'])
pca_df['Cluster'] = cluster_labels

# Plotting the clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(data=pca_df, x='PC1', y='PC2', hue='Cluster', palette='viridis', s=100,
alpha=0.7, edgecolor='k')
plt.title('Clusters Visualized using PCA')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='Cluster')
plt.show()

"""Clustering with Hierarchical Clustering"""

X = X.iloc[:, :-1]

X

# Fit the AgglomerativeClustering model (Hierarchical Clustering)
hierarchical = AgglomerativeClustering(n_clusters=3, linkage='ward')
hierarchical_labels = hierarchical.fit_predict(X)

# Create the Dendrogram (Hierarchical Tree) to visualize clusters
linked = linkage(X, 'ward')

# Plot the dendrogram
plt.figure(figsize=(10, 7))
dendrogram(linked)
plt.title("Dendrogram for Hierarchical Clustering")
plt.xlabel('Sample Index')
plt.ylabel('Distance')
plt.show()

# Evaluate the clustering performance

# Silhouette Score
sil_score_hierarchical = silhouette_score(X, hierarchical_labels)

# Calculate Adjusted Rand Index
true_labels = data['race_outcome']
ari_hierarchical = adjusted_rand_score(true_labels, hierarchical_labels)

# Calinski-Harabasz
ch_score_hierarchical = calinski_harabasz_score(X, hierarchical_labels)

# Display the results
print(f"Silhouette Score for Hierarchical Clustering: {sil_score_hierarchical}")
```

```
print(f"Adjusted Rand Index for Hierarchical Clustering: {ari_hierarchical}")
print(f"Calinski-Harabasz Index for Hierarchical Clustering: {ch_score_hierarchical}")

# Visualizing the Clusters using PCA
pca = PCA(n_components=2)
X_pca_2d = pca.fit_transform(X)

# Create a DataFrame for the 2D PCA components and the cluster labels
df_pca = pd.DataFrame(X_pca_2d, columns=['PC1', 'PC2'])
df_pca['Cluster'] = hierarchical_labels

# Visualize the clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df_pca, x='PC1', y='PC2', hue='Cluster', palette='viridis', s=100,
alpha=0.7, edgecolor='k')
plt.title('Hierarchical Clustering Results with PCA Visualization', fontsize=16)
plt.xlabel('Principal Component 1', fontsize=12)
plt.ylabel('Principal Component 2', fontsize=12)
plt.legend(title='Cluster', loc='best', fontsize=12)
plt.show()

# Sample labels from KMeans and Agglomerative clustering
kmeans_labels = kmeans.labels_
agglo_labels = hierarchical_labels

# Create a DataFrame for comparison
df_clusters = pd.DataFrame({
   "KMeans": kmeans_labels,
   "Agglomerative": agglo_labels
})

# Check where KMeans and Agglomerative assigned the same cluster
matching_labels = (df_clusters["KMeans"] == df_clusters["Agglomerative"]).sum()

# Calculate the percentage of matching labels
accuracy = matching_labels / len(df_clusters) * 100
print(f"Percentage of matching labels: {accuracy:.2f}%")

### Bar Chart
plt.figure(figsize=(10, 5))
ax = sns.countplot(data=df_clusters.melt(var_name="Method", value_name="Cluster"),
              x="Cluster", hue="Method", palette="Set1")

# Add count labels on top of each bar
for p in ax.patches:
   ax.annotate(f'{int(p.get_height())}',
            (p.get_x() + p.get_width() / 2., p.get_height()),
            ha='center', va='bottom', fontsize=12, fontweight='bold')

plt.title("Cluster Distribution: KMeans vs Agglomerative")
plt.xlabel("Cluster")
```

```
plt.ylabel("Count")
plt.legend(title="Method")
plt.show()

### Confusion Matrix
conf_matrix = pd.crosstab(df_clusters["KMeans"], df_clusters["Agglomerative"])
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", linewidths=0.5)
plt.title("Confusion Matrix: KMeans vs Agglomerative")
plt.xlabel("Agglomerative Clusters")
plt.ylabel("KMeans Clusters")
plt.show()
```

A.2: Figures

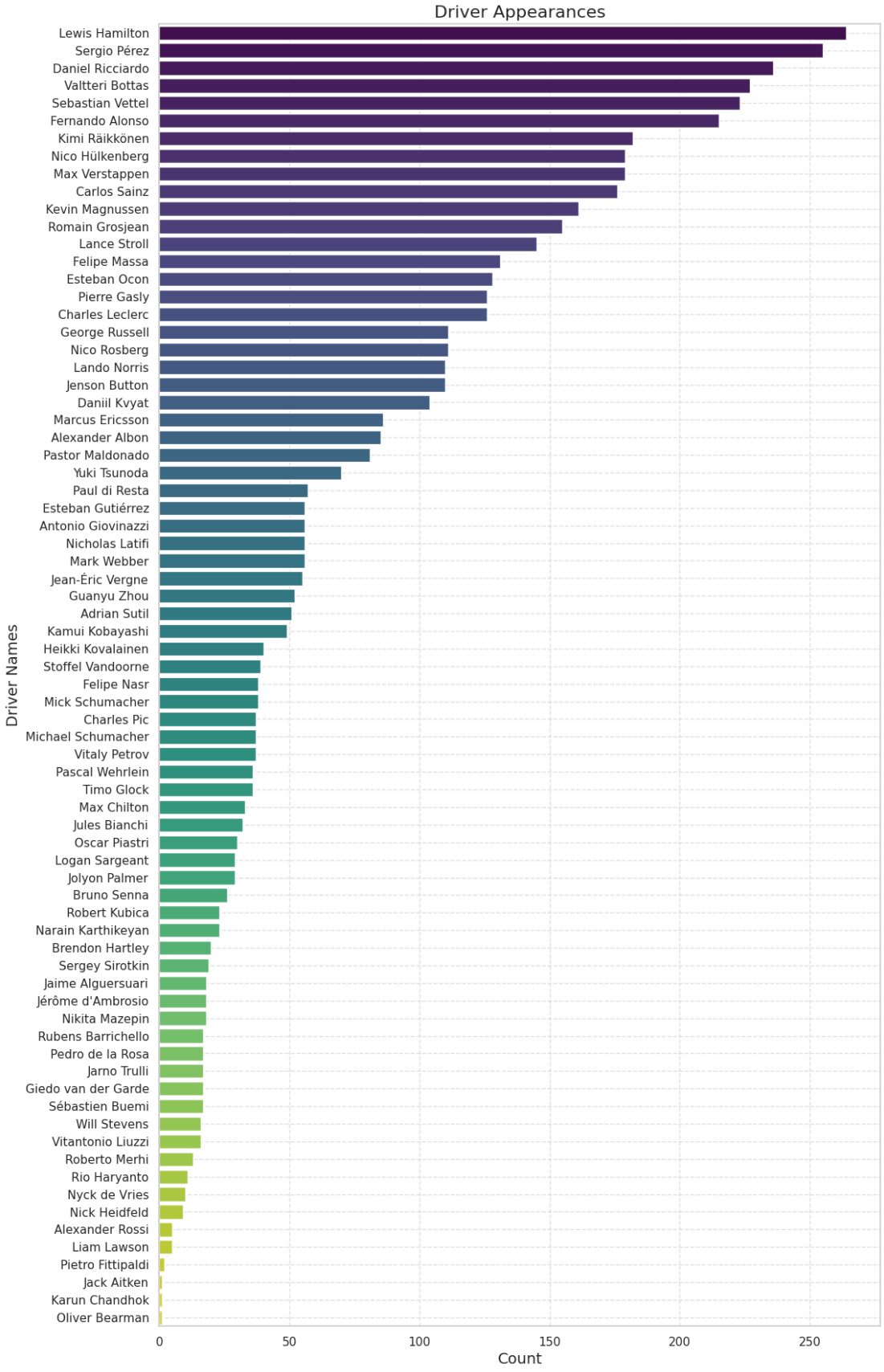

Figure A.1: Driver Appearances Bar Plot

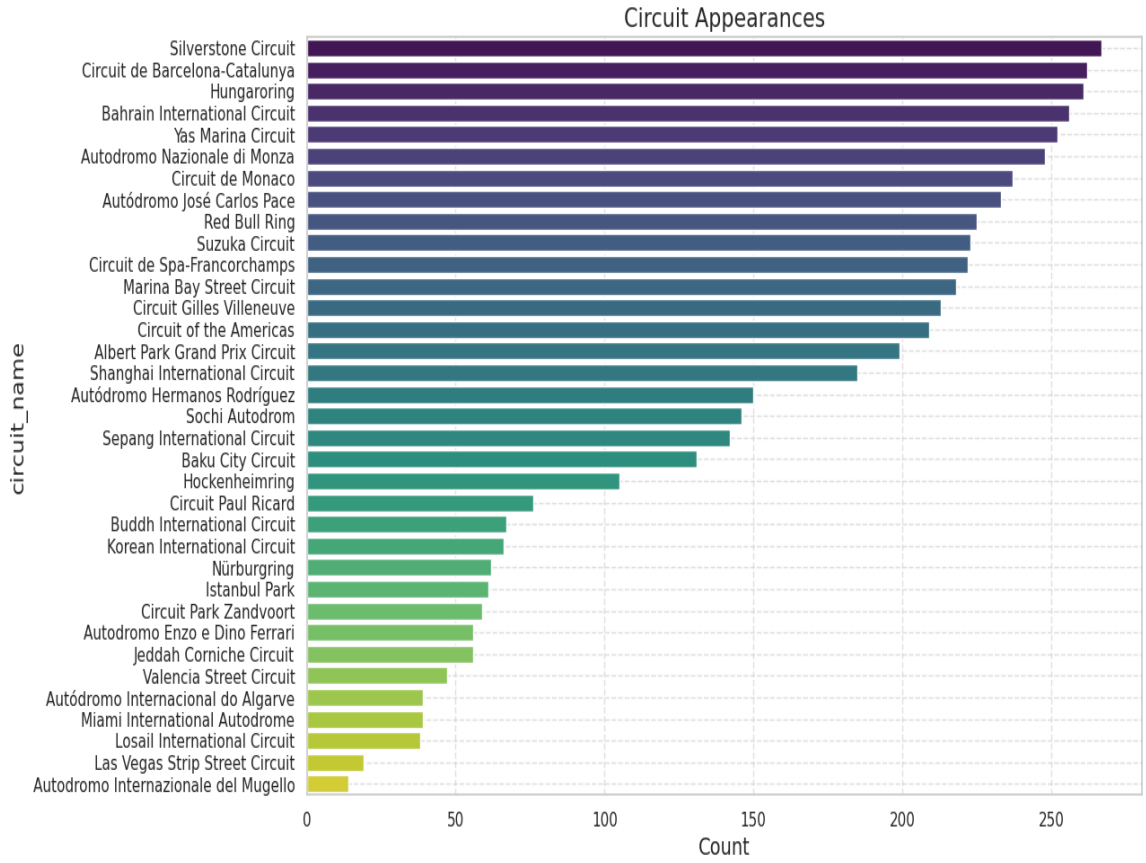

Figure A.2: Circuit Appearances Bar Plot

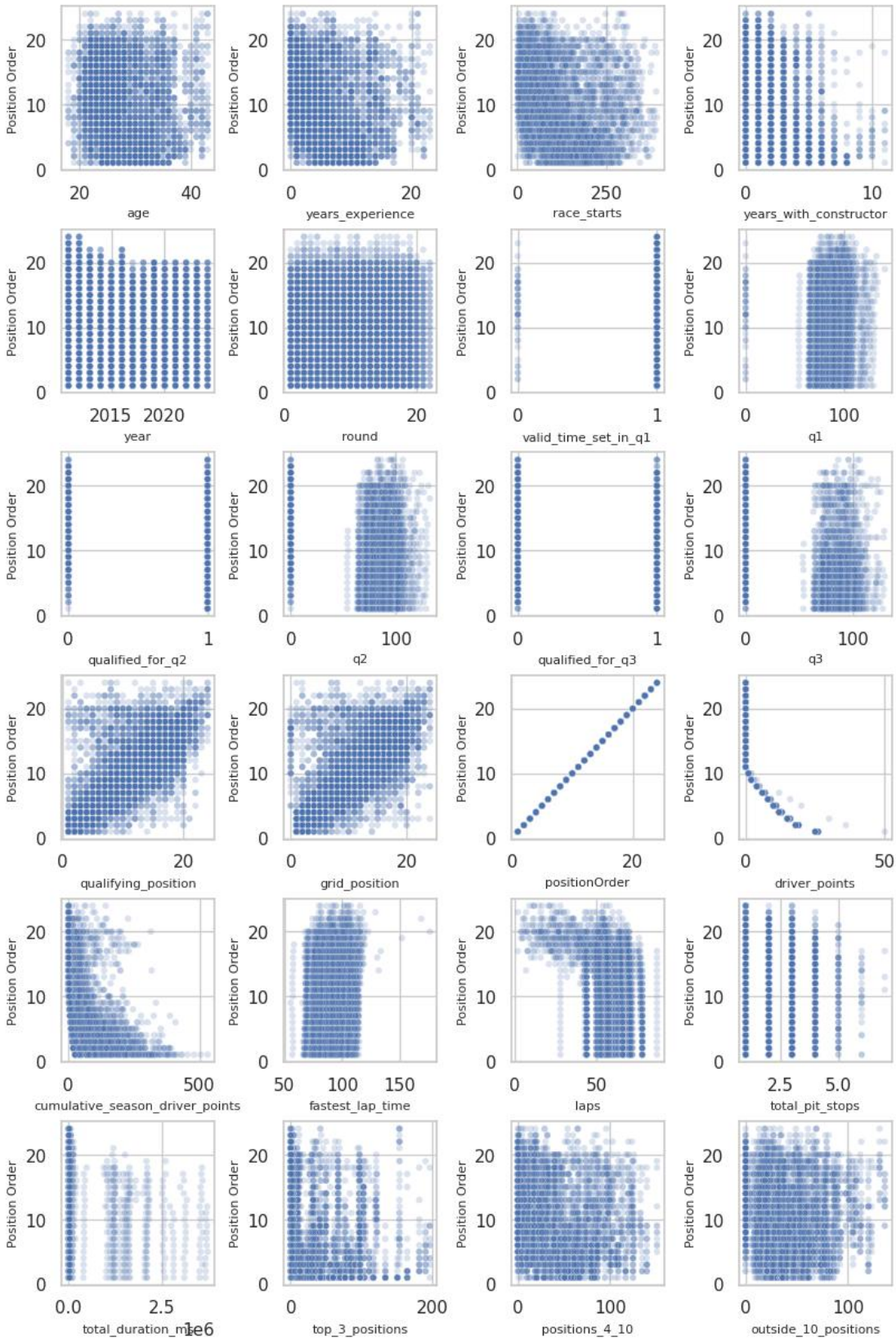

Figure A.3: All Non-Categorical Variables vs Position Order Scatterplots

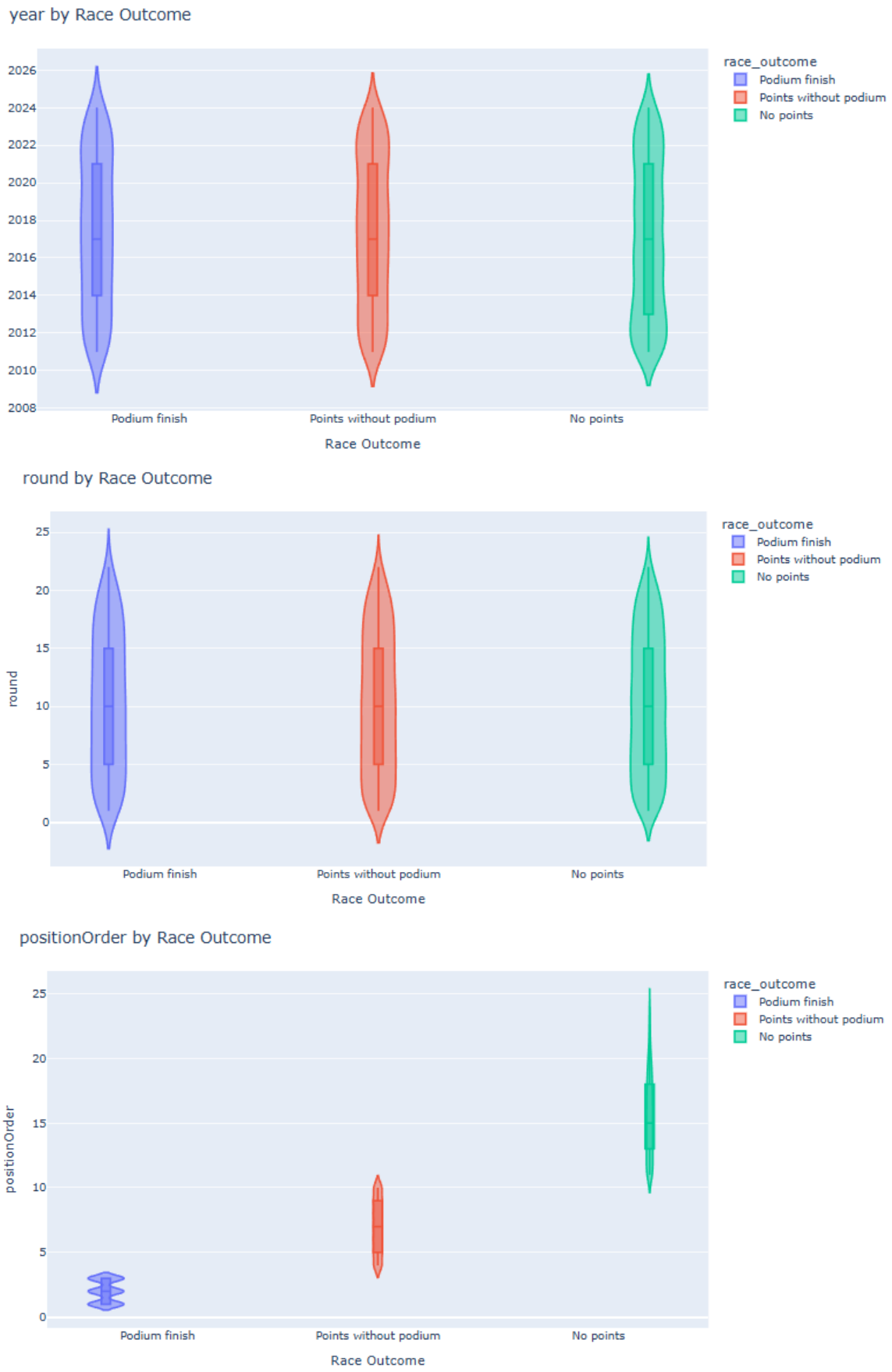

Figure A.4: Numeric Variables vs Race Outcome Violin plots (7)

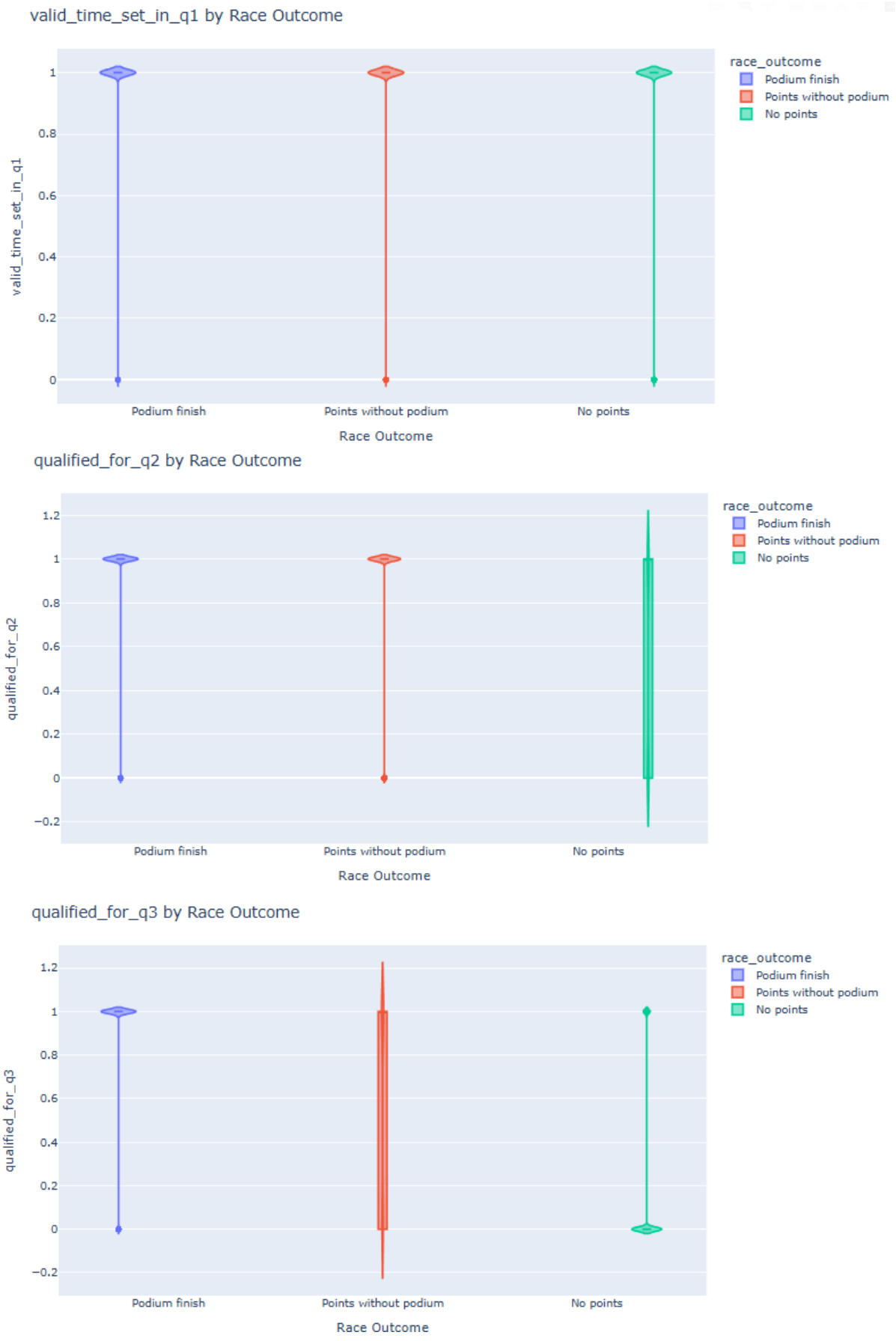

Figure A.5: Numeric Variables vs Race Outcome Violin plots (8)

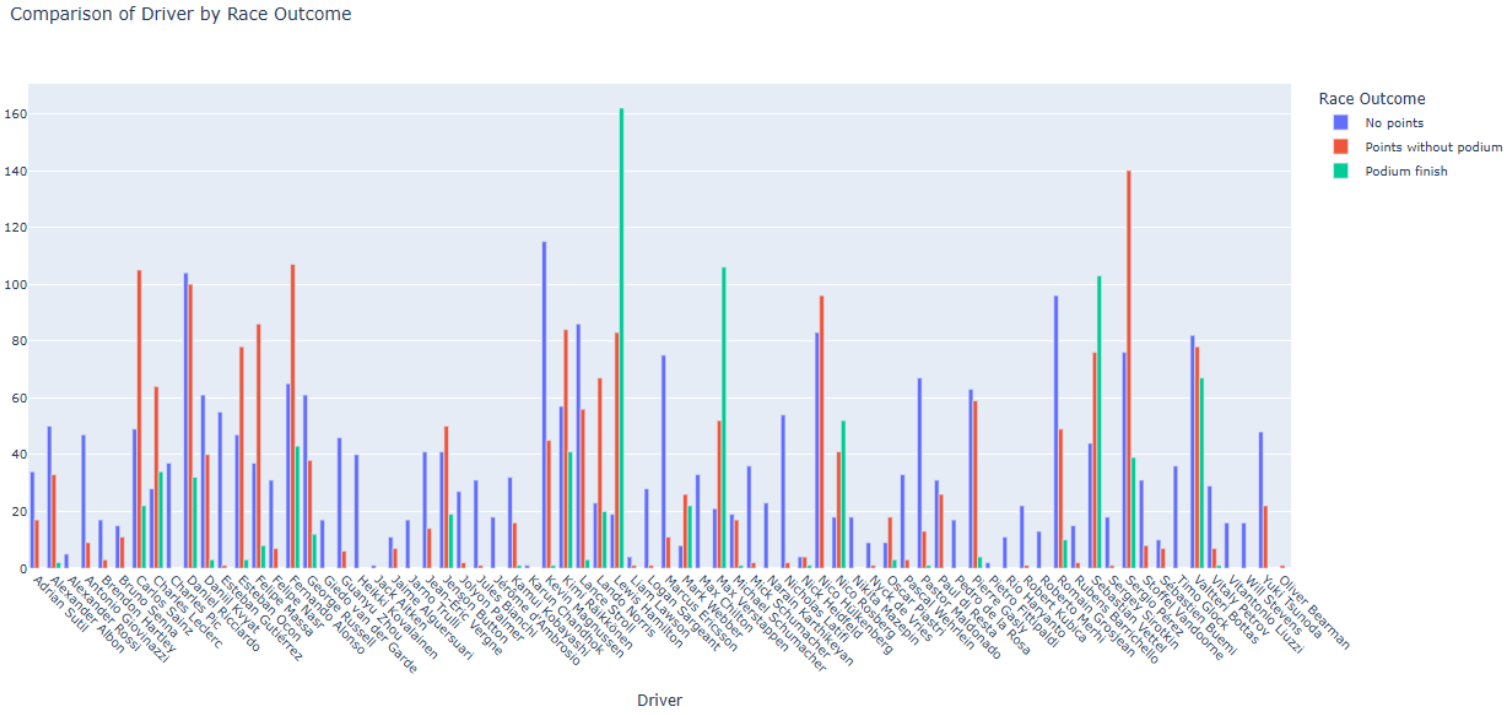

Figure A.6: All Drivers vs Race Outcome Bar Plot

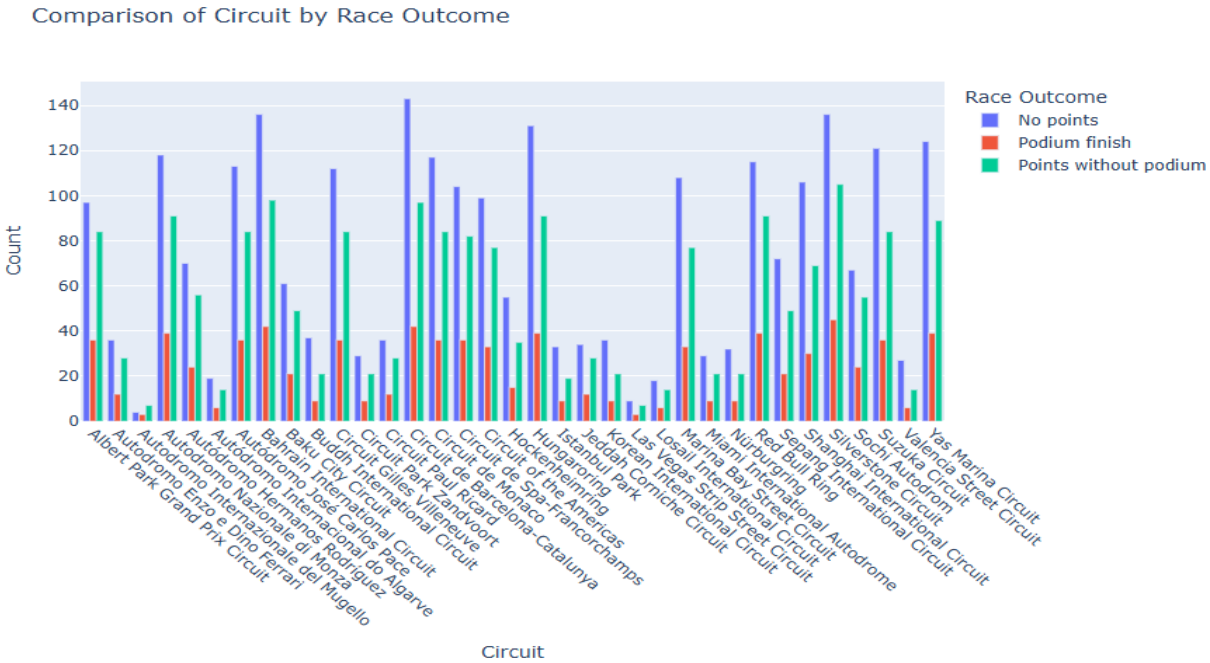

Figure A.7: Circuits vs Race Outcome Bar Plot