

Besondere Lernleistung

Formel 1

Predictive Analytics

Von Eray Kayur
Datum: 12.03.2024
Tutor: Frau van Kessel
Fach: Praktische Informatik

Inhaltsverzeichnis

| | |
|--|----|
| 1. Einleitung | 1 |
| 1.1 Projektziel | 1 |
| 1.2 Projektumfeld | 1 |
| 1.3 Abgrenzung des Projektes | 1 |
| 2. Theoretische Grundlagen | 1 |
| 2.1 Überblick über die verwendeten Modelle | 1 |
| 2.1.1 Lineare Regression | 1 |
| 2.1.2 Multinomiale Logistische Regression | 2 |
| 2.1.3 Evaluationsmetriken für Regressionsmodelle | 3 |
| 2.1.4 Random Tree | 3 |
| 3. Planung | 5 |
| 3.1 Projektphasen | 5 |
| 3.2 Ressourcenplanung | 5 |
| 3.2.1 Personal | 5 |
| 3.2.2 Hardware | 5 |
| 3.2.3 Software | 5 |
| 3.3 Entwicklungsprozess | 5 |
| 4. Datenbeschaffung | 6 |
| 4.1 Recherche und Auswahl der Datenquelle | 6 |
| 4.2 Implementation Data Fetcher | 6 |
| 4.3 Überblick über die abgerufenen Daten | 7 |
| 4.3.1 Datenqualität | 7 |
| 4.3.2 Datenbankaufbau | 8 |
| 4.3.3 Daten Exploration | 9 |
| 5. Datenaufbereitung | 10 |
| 5.1 Feature Extraction | 10 |
| 5.2 Feature Engineering | 11 |
| 5.3 Erstellung von Trainings- und Testdatensätzen | 11 |
| 6. Modellierung | 12 |
| 6.1 Auswahl und Begründung der verwendeten Modelle | 12 |
| 6.2 Modell Implementierung | 12 |

| | |
|--|----|
| 6.3 Modelloptimierung | 12 |
| 6.4 Modellbewertung | 13 |
| 7. Evaluation | 14 |
| 7.1 Darstellung der Analyseergebnisse | 14 |
| 7.2 Analyse und Bewertung der Ergebnisse | 14 |
| 7.2.1 Lineare Regression | 14 |
| 7.2.2 Multinomiale Logistische Regression | 14 |
| 7.2.3 Random Tree | 15 |
| 7.3 Bewertung der Modelle | 16 |
| 7.4 Vergleich mit bestehenden Lösungen | 16 |
| 8. Diskussion und Ausblick | 17 |
| 8.1 Zusammenfassung der Ergebnisse | 17 |
| 8.2 Mögliche zukünftige Erweiterungen | 18 |
| 9. Fazit | 19 |
| 10. Anhang | 20 |
| 10.1 Quelltext | 20 |
| 10.1.1 DDL | 20 |
| 10.2.2 Test-, Trainingsdaten split mit Sklearn | 24 |
| 10.2.3 Grid Search Optimierung | 24 |
| 10.2.4 Random Search Optimierung | 24 |
| 10.2.5 Feature Importance | 25 |
| 10.2.6 Precision und Accuracy für Top 3, 5 und 10 | 25 |
| 10.2.7 MAE Berechnung | 26 |
| 10.2.8 MSE UND R2 | 26 |
| 10.2.9 Implementierung Modelle | 26 |
| 10.2.9.1 Lineare Regression | 26 |
| 10.2.9.2 Multinomiale Logistische Regression | 26 |
| 10.2.9.3 Random Tree | 26 |
| 10.3 Model Ergebnisse | 27 |
| 10.3.1 Lineare Regression | 27 |
| 10.3.2 Multinomiale logistische Regression: Variante 1 | 28 |
| 10.3.3 Multinomiale Logistische Regression: Variante 2 | 30 |
| 10.3.4 Random Tree | 32 |

| | |
|--|----|
| 10.4 Datenexploration | 34 |
| 10.5 Diagramme | 36 |
| 10.5.1 ERM | 36 |
| 10.5.2 Heatmaps und Veranschaulichungen | 37 |
| 10.5.2.1 Lineare Regression | 37 |
| 10.5.3.2 Multinomiale logistische Regression: Variante 1 | 38 |
| 10.5.3.3 Multinomiale logistische Regression: Variante 2 | 39 |
| 10.5.3.4 Random Forest | 40 |
| 10.6 Quellenverzeichnis | 41 |
| 10.7 Eidesstattliche Erklärung | 42 |

1. Einleitung

1.1 Projektziel

In meiner Freizeit beschäftige ich mich schon lange mit der Formel 1 und verfolge die Rennen regelmäßig. Die letzten Jahre hat die Formel 1 einen neuen Aufschwung erlebt mit neuen Sponsoren wie z. B. Oracle und Amazon Web Services damit einher gingen auch Verbesserungen in den Vorhersagen für die nächsten Rennen für die Zuschauer, aber auch Vorhersagen in welcher Position Fahrer nach einem Pit Stop sein werden. Seit dem hat mich dieses Thema ergriffen und ich wollte mal mich selber dran trauen die Formel 1 Rennergebnisse Vorherzusagen, sowie diese Vorhersagen immer bereit zu haben den Amazon Web Services stellt diese nicht bei jedem Rennen zu Verfügung, sondern erst ungefähr in der Mitte der Saison.

In diesem Projektrahmen möchte ich also das nächste Rennergebnis der Formel auf Basis von Verfahren des Maschinellen Lernen Vorhersagen.

1.2 Projektumfeld

Ich habe mein Projekt während meiner Freizeit Zuhause umgesetzt und didaktisch mir mit Büchern wie Introduction to statistical learning das Fachwissen angeeignet.

1.3 Abgrenzung des Projektes

Das Hauptziel dieses Projektes ist es nicht ein Machine Learning Modell zu haben welches die Rennen der neuen Formel 1 Ära vorhersagt und nur für die nächsten Rennen. Es wird zunehmend nur auf die Daten der letzten Formel 1 Ära eingegangen (Hybrid-Ära)

2. Theoretische Grundlagen

2.1 Überblick über die verwendeten Modelle

2.1.1 Lineare Regression

Die Lineare Regression[6] ist ein statistisches Verfahren, um den linearen Zusammenhang zwischen einer abhängigen Zielvariable y und einer oder mehreren unabhängigen Variablen x zu modellieren. Ziel ist es, eine Geradengleichung zu finden, die den Zusammenhang möglichst gut beschreibt und damit Vorhersagen für y bei gegebenen Werten für x erlaubt.

Die Gleichung:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

Dabei sind β_0 der y -Achsenabschnitt und β_1 bis β_n die Koeffizienten, die angeben, um wie viele Einheiten sich y ändert, wenn sich das jeweilige x um eine Einheit erhöht. Die Koeffizienten werden anhand der Trainingsdaten so bestimmt, dass die Summe der

quadratischen Abweichungen zwischen den vorhergesagten und tatsächlichen y -Werten minimal wird.

Die Lineare Regression eignet sich, wenn ein linearer Zusammenhang zwischen den Variablen plausibel ist und die Zielgröße metrisch skaliert ist. Vorteile sind die einfache Interpretierbarkeit und die Effizienz bei der Berechnung. Nachteile sind die Anfälligkeit gegenüber Ausreißern und die Unfähigkeit, nicht-lineare Zusammenhänge abzubilden.

2.1.2 Multinomiale Logistische Regression

Die logistische Regression[4] ist ein statistisches Verfahren zur Modellierung und Vorhersage kategorialer, meist binärer Zielvariablen (z.B. "Ja" oder "Nein", "Überlebt" oder "Nicht überlebt"). Sie wird häufig eingesetzt, wenn die Eintrittswahrscheinlichkeit eines Ereignisses in Abhängigkeit von verschiedenen Einflussfaktoren geschätzt werden soll.

Im Gegensatz zur linearen Regression, die eine kontinuierliche Zielgröße modelliert, verwendet die logistische Regression die logistische Funktion, um die Vorhersage auf einen Wertebereich zwischen 0 und 1 zu begrenzen. Dieser Wert kann dann als Wahrscheinlichkeit für das Eintreten der positiven Klasse interpretiert werden.

Die Gleichung der logistischen Funktion lautet:

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)}}$$

Dabei ist $p(x)$ die Wahrscheinlichkeit für die positive Klasse, x_1, \dots, x_n sind die Eingabevariablen und β_0, \dots, β_n die zu lernenden Koeffizienten. Durch Anpassen der Koeffizienten wird versucht, die beobachteten Daten bestmöglich zu erklären.

Zur Schätzung der Koeffizienten wird meist die Methode der Maximum-Likelihood-Schätzung verwendet. Dabei werden die Koeffizienten so gewählt, dass die Likelihood (Wahrscheinlichkeit) der beobachteten Daten unter dem Modell maximiert wird.

Die multinomiale logistische Regression ist eine Erweiterung der binären logistischen Regression für den Fall, dass die Zielvariable mehr als zwei Kategorien hat (z.B. "Rot", "Grün", "Blau"). Statt einer einzelnen Wahrscheinlichkeit werden nun für jede Kategorie (außer einer Referenzkategorie) separate Koeffizienten geschätzt. Die Wahrscheinlichkeit für jede Kategorie ergibt sich dann aus der Softmax-Funktion.

$$p(y = j | x) = \frac{e^{\beta_{j0} + \beta_{j1} x_1 + \dots + \beta_{jn} x_n}}{\sum_{k=1}^K e^{\beta_{k0} + \beta_{k1} x_1 + \dots + \beta_{kn} x_n}}$$

Hier ist $p(y = j | x)$ die Wahrscheinlichkeit, dass die Beobachtung zur Kategorie j gehört, gegeben die Eingabevariablen x . K ist die Anzahl der Kategorien und β_{jn} sind die Koeffizienten für Kategorie j und Eingabevariable n .

Somit sind auch die Vorteile einer logistischen Regression die leichte Interpretierbarkeit, der wenige Rechenaufwand und das der Einfluss der Parameter bzw. Features auf die Zielkategorie angegeben werden kann

2.1.3 Evaluationsmetriken für Regressionsmodelle

Um die Güte eines Regressionsmodells zu messen, werden verschiedene Metriken verwendet, die die Abweichung zwischen vorhergesagten und tatsächlichen Werten quantifizieren. In diesem Projekt werden der mittlere absolute Fehler (MAE), der mittlere quadratische Fehler (MSE) und das Bestimmtheitsmaß (R^2) betrachtet.

Der MAE ist definiert als der Durchschnitt der absoluten Differenzen zwischen Vorhersage und wahren Wert über alle Datenpunkte.

$$MAE = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i|$$

Dabei ist n die Anzahl der Datenpunkte, y der vorhergesagte Wert für den i -ten Datenpunkt und y_i der wahre Wert. Der MAE hat die gleiche Einheit wie die Zielgröße und ist leicht interpretierbar.

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2$$

Der MSE ist definiert als der Durchschnitt der quadrierten Differenzen zwischen Vorhersage und wahren Wert.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Durch die Quadrierung werden große Abweichungen stärker bestraft. Die Wurzel des MSE ergibt den RMSE (Root Mean Squared Error), der wieder die Einheit der Zielgröße hat.

Das Bestimmtheitsmaß R^2 gibt den Anteil der Varianz in den Daten an, der durch das Modell erzeugt wird.

Dabei ist \bar{y} der Mittelwert der wahren Werte. Ein R^2 von 1 bedeutet eine perfekte Anpassung, bei 0 erklärt das Modell die Daten nicht besser als eine Vorhersage mit dem Mittelwert. R^2 eignet sich zum Vergleich verschiedener Modelle auf den gleichen Daten.

2.1.4 Random Tree

Random Forest ist ein leistungsstarkes Ensemble-Lernverfahren, das auf der Kombination mehrerer Entscheidungsbäume basiert. Im Gegensatz zu einem einzelnen Entscheidungsbaum, der anfällig für Overfitting und hohe Varianz sein kann, erzielt der Random Forest durch die Aggregation der Vorhersagen vieler Bäume eine höhere Robustheit und Genauigkeit[5].

Das Grundprinzip des Random Forest besteht darin, eine große Anzahl von Entscheidungsbäumen auf zufällig ausgewählten Teilmengen der Trainingsdaten zu trainieren. Jeder Baum kann dabei mit einer Zufallsstichprobe der Datenpunkte und einer zufälligen Auswahl der Merkmale konstruiert werden. Dieser Prozess wird als "Bootstrap Aggregating" bezeichnet und führt dazu, dass die einzelnen Bäume unkorreliert sind und

unterschiedliche Aspekte der Daten erfassen und ist somit eine unsupervised Lernmethode.

Bei der Vorhersage für einen neuen Datenpunkt lässt der Random Forest jeden seiner Bäume eine unabhängige Prognose treffen. Im Falle einer Klassifikationsaufgabe wird die finale Vorhersage durch Mehrheitsentscheid bestimmt, d.h. die Klasse, die von den meisten Bäumen vorhergesagt wurde, wird als Ergebnis ausgegeben. Bei einer Regressionsaufgabe wird der Durchschnitt der Vorhersagen aller Bäume gebildet.

Die zufällige Auswahl von Datenpunkten und Merkmalen bei der Konstruktion der Bäume hat mehrere Vorteile. Zum einen reduziert sie die Korrelation zwischen den Bäumen und erhöht somit die Diversität des Ensembles. Dadurch werden Overfitting und Varianz verringert, da kein einzelner Baum die Vorhersage dominiert. Zum anderen ermöglicht die Zufallsauswahl eine effiziente Parallelisierung des Trainings, da die Bäume unabhängig voneinander erstellt werden können, welches sich aber erst bei Cuda Fähigen Karten stark bevorteilt.

Ein weiterer Vorteil des Random Forest ist seine Fähigkeit, die Wichtigkeit der Merkmale zu bewerten. Durch die Analyse, wie oft ein Merkmal für die Aufteilung in den Bäumen verwendet wird und wie stark es zur Verringerung der Unreinheit beiträgt, lässt sich ein Maß für die Relevanz jedes Merkmals ableiten. Diese Information kann wertvolle Einblicke in die zugrunde liegenden Zusammenhänge der Daten liefern und bei der Merkmalsauswahl helfen.

3. Planung

3.1 Projektphasen

| Projektphase | Geplante Zeit |
|-------------------|---------------|
| Definition | 1 h |
| Planung | 2h |
| Datenbeschaffung | 16h |
| Modellentwicklung | 20h |
| Modellbewertung | 4h |

3.2 Ressourcenplanung

3.2.1 Personal

Das Projekt hat mich selbst als Personal sowie in etwa 1h Zeit von meiner betreuenden Lehrkraft in Anspruch genommen.

3.2.2 Hardware

Für das Projekt wurde mein Laptop das Apple Macbook Pro 15 2019 mit einer Radeon Pro 555 X Grafikkarte und einem i7 mit 6 Kernen in Anspruch genommen.

3.2.3 Software

Für die Programmierung habe ich die Programmiersprache Python 3.12 genutzt in Kombination mit der Entwicklungsumgebung Pycharm. Das Datenbankmanagementsystem was ich für das Projekt genutzt habe ist PostgreSQL. Als Adapter zwischen python und der Datenbank habe ich die psycopg2 framework genutzt.

Diese Python-Bibliotheken kommen zum Einsatz:

- Pandas für die Datenaufbereitung und -analyse
- Scikit-learn für Machine Learning Algorithmen
- Matplotlib und Seaborn für die Datenvisualisierung
- Jupyter Notebook für Data Exploration

3.3 Entwicklungsprozess

Als Entwicklungsprozess habe ich ein Kanban genutzt, welches ich auf dem Organisationstool Notion erstellt habe. Hier habe ich regelmäßig Ideen zum Vorhersagen der Rennergebnisse notiert sowie meinen Fortschritt verfolgt.

4. Datenbeschaffung

4.1 Recherche und Auswahl der Datenquelle

Für die Datenbeschaffung habe ich zunächst nach einer möglichst vollständigen Datenquelle Ausschau gehalten. Zunächst habe ich es in Erwägung gezogen eine Formel-1-Datenbank im CSV-Format auf Kaggle zu nehmen, jedoch fiel mir auf, dass es Inkonsistenzen bei Rennen gibt, die verschoben worden sind, und dass die Renndatenbank alle 2 Monate aktualisiert wird, was derzeit kein Problem darstellt, aber für spätere Erweiterungen ein Problem sein kann.

Als zweites bin ich auf die Ergast API gestoßen und darauf folgend auf den Pyergast Pandas-Wrapper, welcher für mich zunächst wie eine vollständige Lösung aussah, wo ich sogar keine Datenbank bräuchte.

Jedoch hat sich herausgestellt, dass Pyergast nicht auf meinem Macbook läuft, da es Urllib3-Library-Funktionen nutzt, die von macOS sowie Linux nicht unterstützt werden. Somit habe ich die betreffenden Codeabschnitte des Frameworks überarbeitet, sodass sie mit der Urllib2-Library funktionieren. Somit habe ich zunächst die Funktionalitäten des Wrappers getestet und kam zum Entschluss, dass dieser nicht alle Funktionen unterstützt, die die Ergast API zur Verfügung stellt, und somit bin ich zum Entschluss gekommen, eine eigene Datenbank zu erstellen und diese mit der ErgastAPI zu befüllen.

Die ErgastAPI[2] ist ein Community-Projekt eines Formel-1-Fans, welcher konsistente Daten zu allen Formel-1-Rennen, Fahrern, Teams, Ergebnissen und Qualifying-Zeiten, die seit Beginn der Weltmeisterschaft 1950 bis zur aktuellen Saison 2023 stattfanden, bereitstellt.

Die Aktualität der Daten ist sehr hoch, da Updates meistens 3 Stunden nach dem Rennen aktualisiert werden und die Daten im JSON-Format und in einer eigenen Subdomain-Query-Syntax abgerufen werden können, zum Beispiel <http://ergast.com/api/f1/2012>, um den Rennkalender der Formel 1 für das Jahr 2012 zu erhalten.

Der einzige Nachteil dieser Datenbank ist, dass keine Informationen zur Streckenhöhe gespeichert beziehungsweise abgerufen werden können. Jedoch ist diese Information meiner Meinung nach wichtig, da Formel-1-Teams in bestimmten Rennhöhen besser performen als sonst. So war 2021 zu beobachten, dass der Formel-1-Wagen von Mercedes auf Rennstrecken wie Zandvoort, welches besonders tief liegt, besser performt als sonst.

Um diese Information zu erhalten, habe ich die Open-Elevation API[3] genutzt, die kostenlos Informationen zur Höhe anhand der Longitude und Latitude bereitstellt. Hierfür muss man ähnlich wie bei Ergast die Longitude und Latitude in eine Subdomain packen und man erhält als Rückgabe eine JSON-Datei mit der Höhe.

4.2 Implementation Data Fetcher

Die Klasse DataFetcher beinhaltet die Methoden zum Abrufen und Speicher der Daten aus der ErgastAPI und Open-Elevation API. Die Grundlegende Funktionsweise ist das von

der API eine subdomain query durchgeführt wird und diese eine json Datei zurückgibt. Diese Daten werden aus der Json Datei entnommen und in der Datenbank gespeichert.

Um das vorgehen zu verdeutlichen erkläre ich es exemplarisch an der Methode `fetchLaptimes()` die für das Abrufen und Speichern der Rundenzeiten jedes Fahrers in jedem Rennen zuständig ist und welche eine relativ komplexe Hierarchische verschachtlung (Jahr > Rennen > Runde > Fahrer) vorweist.

In der Methode wird zuerst eine SQL-Abfrage ausgeführt, um die verfügbaren Jahre aus der Tabelle `Seasons` aufsteigend sortiert abzurufen. Dann wird für jedes Jahr die maximale Rundenzahl der Rennen bestimmt, indem die Funktion `SELECT MAX(round)` auf die Tabelle `racess` angewendet wird, gefiltert nach dem jeweiligen Jahr.

Anschließend durchläuft eine verschachtelte Schleife alle Rennen des Jahres von Runde 1 bis zur maximalen Rundenzahl. Für jede Runde wird die entsprechende JSON-Datei mit den Rundenzeiten von der Ergast API abgerufen. Die URL dafür setzt sich zusammen aus der Basis-URL "<http://ergast.com/api/f1/>", dem Jahr, der Rundennummer und dem Zusatz `"/laps.json?limit=2000"`.

Die empfangenen JSON-Daten werden wieder mit `json.load()` in ein Python-Objekt konvertiert. Wenn das Rennen in den Daten vorhanden ist (d.h. wenn die Länge von `data["MRData"]["RaceTable"]["Races"]` größer als 0 ist), wird über die Runden (`lap`) des Rennens iteriert. Für jede Runde wird die Rundennummer (`lap["number"]`) extrahiert. Dann folgt eine weitere Schleife über die Rundenzeiten (`time`) jedes Fahrers in der aktuellen Runde. Hier werden die `raceld` anhand von Rundennummer und Jahr sowie die `driverId` anhand der `driverId` aus dem JSON mit SQL-SELECT-Abfragen aus den Tabellen `racess` bzw. `drivers` abgerufen.

Mit den erhaltenen IDs und den weiteren Informationen aus dem JSON (`Position`, `Zeit`) wird ein SQL-INSERT-Statement erstellt, um die Rundenzeit in die Tabelle `laptimes` einzufügen. Das Statement wird mit `self.cur.execute()` ausgeführt, bei Erfolg wird die Transaktion mit `self.conn.commit()` bestätigt und der Zähler `count1` erhöht, bei einem Fehler wird die Transaktion zurückgerollt, der Fehler protokolliert und der Zähler `count2` erhöht.

Dieser Prozess wird für alle Runden, Fahrer und Jahre wiederholt. Am Ende gibt die Methode wieder die Anzahl der erfolgreich hinzugefügten und fehlgeschlagenen Einfügungen aus. Die Fehlschläge werden durch ein try-catch block gehandelt um die Datenintegrität zu behalten.

4.3 Überblick über die abgerufenen Daten

Die von der Ergast API und Open-Elevation-API abgerufenen Daten werden in einer PostgreSQL-Datenbank gespeichert. Die Struktur der Datenbank spiegelt die Entitäten und Beziehungen der Formel-1-Daten wider. Zudem bietet die persistente Speicherung den Vorteil, dass man mit den Daten ohne die Anbindung zu der API arbeiten kann und die API vor unnötigen mehrmaligen Anfragen schont, da diese immer noch auf Kosten von Fans betrieben wird.

4.3.1 Datenqualität

Die Daten liegen, wie schon zuvor erwähnt, zunächst nach dem Abrufen von den beiden APIs als JSON-Datei vor. Nach einer Überprüfung konnte ich feststellen, dass die für mein

Modell benötigten Daten maximal bis zum Jahr 2014 vollständig sind. Bis zum Jahr 1996 sind keine Rennrundenergebnisse verfügbar.

4.3.2 Datenbankaufbau

Das vorliegende ERM (Entity-Relationship-Modell) bildet die Struktur der Formel-1-Datenbank ab. Es besteht aus den Entitätstypen Circuits, Status, Constructors, Drivers, Seasons, Races, Qualifying, Results, SprintResults, ConstructorStandings, DriverStandings, LapTimes und PitStops.

Der Entitätstyp Circuits repräsentiert die Rennstrecken und hat die Attribute circuitId (identifizierend), name, location, country, lat, lng, alt und url.

Der Entitätstyp Status enthält mögliche Statuswerte für Rennergebnisse mit den Attributen statusId (identifizierend) und status.

Constructors steht für die Rennställe und hat die Attribute constructorId (identifizierend), name, nationality und url.

Der Entitätstyp Drivers repräsentiert die Fahrer mit den Attributen driverId (identifizierend), driverRef, forename, surname, dob, nationality und url. Jeder Fahrer (Drivers) fährt in mehreren Rennen (Races), wobei an einem Rennen mehrere Fahrer teilnehmen (n:m-Beziehung). Deshalb werden die Ergebnisse über die Assoziationstabellen Results, Qualifying und SprintResults abgebildet.

In Seasons werden die Jahre mit Saisons gespeichert, mit den Attributen year (identifizierend) und url.

Der Entitätstyp Races enthält die einzelnen Rennen mit den Attributen raceId (identifizierend), year, round, circuitId (Fremdschlüssel zu Circuits), name, date und url. Jedes Rennen findet auf genau einer Rennstrecke statt (1:n-Beziehung von Races zu Circuits).

Die Rennergebnisse sind in Results mit den Attributen resultId (identifizierend), raceId und driverId (Fremdschlüssel zu Races und Drivers), constructorId (Fremdschlüssel zu Constructors), grid, position, positionText, points, laps, time, fastestLap, rank, fastestLapTime, fastestLapSpeed und statusId (Fremdschlüssel zu Status) gespeichert.

Qualifying enthält die Qualifikationsergebnisse analog zu Results, mit den Attributen qualifyId (identifizierend), raceId, driverId, constructorId, number, position und den Rundenzeiten q1, q2, q3.

SprintResults speichert die Ergebnisse der Sprintrennen mit den Attributen sprintResultId (identifizierend), raceId, driverId, constructorId und weiteren Attributen ähnlich zu Results.

ConstructorStandings bildet die WM-Stände der Konstrukteure zu jedem Rennen ab, mit den Attributen constructorStandingsId (identifizierend), raceId, constructorId, points, position, positionText und wins.

Die Fahrerwertung steht analog dazu in DriverStandings mit driverStandingsId (identifizierend), raceId, driverId, points, position, positionText und wins.

LapTimes enthält die Rundenzeiten jedes Fahrers in jedem Rennen mit den Attributen raceld, driverId, lap (zusammen identifizierend), position und time.

Schließlich speichert PitStops die Boxenstopps mit raceld, driverId, stop (zusammen identifizierend), lap, time und duration.

Die Beziehungen zwischen den Entitätstypen sind über Fremdschlüssel abgebildet. Die n:m-Beziehungen zwischen Drivers, Races und Constructors werden über die Assoziationstabellen Results, Qualifying und SprintResults aufgelöst.

Im Anhang befindet sich das zugehörige DDL (10.1.1) und ERM(10.6.1).

4.3.3 Daten Exploration

Aus der Datenexploration lassen sich interessante Erkenntnisse über die Formel 1 gewinnen. Es zeigt sich, dass Teams oft Siegestränen erleben und über längere Zeiträume dominant sein können. Die Anzahl der Rennen pro Saison hat sich im Laufe der Zeit verändert, ebenso wie das Qualifyingformat, das mehrfach angepasst wurde.

Auch Skandale und Regelverstöße haben Einfluss auf die Leistung der Teams genommen. 2007 wurde McLaren im Zuge des Spygate-Skandals auf den letzten Platz verbannt. 2020 verbot die FIA das Dual-Axis-Steering-System von Mercedes, was sich negativ auf deren Leistung im Folgejahr auswirkte. Seit 2020 gibt es zudem eine Budgetobergrenze für die Teams sowie gravierende Änderungen im Bereich der Aerodynamik.

2010 erfolgte eine bedeutende technische Umstellung von V8-Motoren auf Hybrid-V6-Turbomotoren. Gleichzeitig wurde das DRS (Drag Reduction System) eingeführt, das Überholmanöver erleichterte. Um den Einstieg von Audi in die Formel 1 zu vereinfachen, wurde bis 2026 ein Entwicklungsstopp für Motoren verhängt.

Um die Vorhersagbarkeit der Platzierung besser einzuschätzen habe ich zusätzlich für jeden Fahrer einen Graphen geplottet mit den Runden der F1 Saison 2023 auf der X-Achse und der Position auf der Y-Achse. Die Datenpunkte die auf den Graph eingetragen worden sind haben eine Lineare Regression der Daten abgebildet und zeigen ob eine Abhängigkeit herrscht. Am Beispiel von dem Graph von Max Verstappen kann man einen Trend sehr deutlich erkennen im Vergleich dazu kann man bei dem Graph von George Russel keine Abhängigkeit erkennen.

5. Datenaufbereitung

5.1 Feature Extraction

Bei der Feature Extraction werden relevante Informationen direkt aus den Rohdaten extrahiert, ohne weitere Transformationen durchzuführen. Hierzu zählen beispielsweise die Startposition eines Fahrers in einem bestimmten Rennen, die Anzahl der Boxenstopps im vorherigen Jahr auf derselben Strecke oder die Rundenzeiten des Fahrers im letzten Rennen. Diese Merkmale liefern direkte Einblicke in die Leistung und das Verhalten von Fahrern und Konstrukteuren.

Fahrer-Features:

- Vollständiger Name des Fahrers
- Alter des Fahrers
- Anzahl der Jahre in der Formel 1
- Anzahl der Starts aus der ersten Startreihe
- Anzahl der Rennsiege in der Karriere
- Anzahl der Saisonsiege
- Anzahl der gestarteten Rennen
- Anzahl der beendeten Rennen
- Anzahl der Pole-Positions
- Anzahl der gewonnenen Fahrer-Meisterschaften
- Fahrer-Gesamtrang im Vorjahr
- Aktueller Fahrer-Gesamtrang
- Qualifying-Position des Fahrers im aktuellen Rennen
- Rennposition des Fahrers im aktuellen Rennen
- Ergebnisse des vorherigen Rennens (Endposition, Qualifying-Position, gewonnene Positionen)
- Ergebnisse des vorherigen Jahres auf derselben Strecke (Endposition, Qualifying-Position, gewonnene Positionen)
- Anzahl der Boxenstopps im Rennen des Vorjahres

Konstrukteur-Features:

- Name des Konstrukteurs
- Anzahl der gewonnenen Konstrukteurs-Meisterschaften
- Anzahl der Rennsiege
- Anzahl der Saisonsiege
- Konstrukteurs-Gesamtrang im Vorjahr
- Aktueller Konstrukteurs-Gesamtrang
- Qualifying-Position des besten Teamkollegen im aktuellen Rennen
- Gesamtzahl der Ausfälle

Strecken-Features:

- Name der Strecke
- Rennrunde
- Rennjahr

5.2 Feature Engineering

Fahrer-Features:

- Durchschnittlich gewonnene/verlorene Positionen in den letzten X Rennen
- Korrelation zwischen Qualifying-Position und Rennposition über die Karriere hinweg
- Zeitunterschied zum Sieger im letzten Rennen, berechnet als Prozentsatz
- Durchschnittliche Rundenzeit im letzten Rennen
- Konsistenz der Rundenzeiten im letzten Rennen, berechnet als Standardabweichung
- Geschwindigkeitsrang im letzten Rennen, basierend auf der durchschnittlichen Geschwindigkeit

Konstrukteur-Features:

- Anzahl der gewonnenen Meisterschaften in den letzten X Jahren
- Maximal gewonnene und verlorene Positionen in den letzten X Rennen
- Durchschnittlich gewonnene/verlorene Positionen in den letzten X Rennen
- Geschwindigkeitsunterschied zum schnellsten Konstrukteur im letzten Rennen, berechnet als Prozentsatz der schnellsten Runde
- Anzahl der Ausfälle in den letzten X Rennen
- Maximale, minimale und durchschnittliche Geschwindigkeit im letzten Rennen, basierend auf den schnellsten Runden
- Geschwindigkeitsrang im letzten Rennen, basierend auf der durchschnittlichen Geschwindigkeit

Strecken-Features:

- Durchschnittliche Anzahl der Überholmanöver pro Rennen auf der Strecke

5.3 Erstellung von Trainings- und Testdatensätzen

Für die Erstellung von Trainings- und Testdatensätzen habe ich die Daten bis zum Jahre 2021 benutzt, da die Regelveränderung im Jahre 2021 maßgebend waren für die folgenden Jahre und für die bisherigen Platzhirsche in der Formel 1 wie Mercedes und Red Bull.

Die Aufteilung von Trainings- und Testdatensätze habe ich im Verhältnis von 0,2 und zufällig bestimmt. Der Grund für die zufällige Bestimmung und in diesem Sachverhalt eigentlich komischen Entscheidung da eine chronologische sortiert Aufteilung mehr sinn ergebe ist es ein Overfitting zu vermeiden. Mir stehen relativ wenige Datensätze zur Verfügung die eine wirkliche Abhängigkeit mit dem nächsten Rennergebniss ausdrücken können und so Sorge präventiv für die Vermeidung des Overfittes.

Für die Implementierung habe ich die `train_and_test_split()` Methode der Sklearn Bibliothek genutzt (siehe Anhang 10.2.3)

6. Modellierung

6.1 Auswahl und Begründung der verwendeten Modelle

Die lineare Regression gehört zu den grundlegenden Modellen in der Statistik und im maschinellen Lernen. Obwohl die Vorhersage von Rennplatzierungen eigentlich ein Klassifikationsproblem darstellt, bei dem ein Modell zur Vorhersage kategorialer Variablen erforderlich ist, bietet die lineare Regression den Vorteil der leichten Interpretierbarkeit. Im Vergleich zu komplexeren Modellen lassen sich die Ergebnisse einer linearen Regression sehr einfach nachvollziehen und analysieren, was wiederum die Auswahl geeigneter weiterführender Modelle erleichtert. Darüber hinaus ermöglicht die lineare Regression einen guten Einblick in die generelle Vorhersagbarkeit der Zielgröße anhand der verfügbaren Merkmale. Da die Merkmale in der linearen Regression jedoch nur als kontinuierliche Variablen berücksichtigt werden können, ist das Modell nicht in der Lage, komplexere Zusammenhänge abzubilden.

Die multinomiale logistische Regression wird für Klassifikationsaufgaben verwendet, in meinem Beispiel, ob ein Fahrer Erster, Zweiter, Dritter, ... wird. Im Vergleich zur linearen Regression bietet sie somit den Vorteil, dass man als Vorhersage keinen stetigen Wert zurückbekommt, sondern einen diskreten. Das Modell kann im Vergleich zu einem Random-Forest-Modell auch deutlich schneller aufgebaut werden und somit auch zum Feature-Testing besser benutzt werden.

Aufgrund dessen habe ich zwei Modell implementiert die sich in den jeweiligen Features unterscheiden um ein Vergleich zu ziehen.

Das Random Forest Model wird als Gegenmodell zu der multinominalen logistischen Regression eingesetzt. Als Ensemble-Methode, die viele Entscheidungsbäume kombiniert, ist er in der Lage, komplexe nichtlineare Zusammenhänge und Wechselwirkungen zwischen den Eingabevariablen zu erfassen. Zudem ist er robust gegenüber Ausreißern und Overfitting, welche bei meiner kleinen Datenmenge provoziert wird.

6.2 Modell Implementierung

Für die Implementation wurden die fertigen Klassen bzw. Methoden von der Sklearn Bibliothek genutzt (siehe Anhang 10.2.9)

6.3 Modelloptimierung

Die Güte der Modelle hängt neben der grundsätzlichen Eignung des Modelltyps auch von der richtigen Einstellung der Hyperparameter ab. Für jedes Modell werden daher verschiedene Hyperparameter-Kombinationen evaluiert, um die optimale Konfiguration zu finden.

Bei der Linearen Regression gibt es nur wenige Hyperparameter, wie z.B. die Regularisierungsmethode (L1, L2) und die Stärke der Regularisierung. Diese werden mittels Grid Search optimiert (siehe Anhang 10.2.3), d.h. es werden alle sinnvollen Kombinationen ausprobiert und anhand der Validierungsmetriken bewertet und das beste davon genommen.

Die Logistische Regression verfügt über die gleichen Hyperparameter wie die Lineare Regression.

Beim Random Forest gibt es im Vergleich zu den anderen Modellen deutlich mehr Hyperparameter die einzustellen sind. So können Die Anzahl der Bäume im Ensemble (`n_estimators`), Die maximale Tiefe der Bäume (`max_depth`), Die minimale Anzahl von Datenpunkten, die ein Blatt enthalten muss (`min_samples_leaf`), Die Anzahl der Features, die bei jedem Split berücksichtigt werden (`max_features`) und ob eine Stichprobenziehung (`bootstrap`) aus oder an sein sollen. Aufgrund der vielen Kombinationsmöglichkeiten wird hier ein Random Search (siehe Anhang 10.2.4) durchgeführt, bei der die Hyperparameter-Werte zufällig aus vorgegebenen Bereichen gezogen werden. Die besten Einstellungen werden dann übernommen.

6.4 Modellbewertung

Die trainierten Modelle werden anhand Metriken aus den Testdaten bewertet und verglichen. Alle Modelle geben die Feature Wichtigkeit aus (siehe Anhang 10.2.5), die das Modellergebnis besser nachvollziehen lässt. Zudem wird abseits der Precision und Accuracy für das gesamte Modell eine Unterteilung für die Top 10 Top 5 und Top 3 Fahrer wahrgenommen um Stärken und Schwächen der Modelle weiter zu vergleichen(siehe Anhang 10.2.6).

Für die regressiven Modelle wird der mittlere absolute Fehler (MAE) , und für das lineare regressive zusätzlich noch der mittlere quadratische Fehler (MSE) und das Bestimmtheitsmaß (R^2) herangezogen (siehe Anhang 10.2.7 und 10.2.8). Der MAE gibt an, um wie viele Punkte die Vorhersage im Mittel daneben liegt. Der MSE bestraft große Abweichungen überproportional stark. Das R^2 misst den Anteil der erklärten Varianz in den Daten.

Alle Modelle geben zusätzlich eine Vorhersage über das Rennergebnis von Abu Dhabi im Jahr 2023 aus.

7. Evaluation

7.1 Darstellung der Analyseergebnisse

Die Ergebnisse der Modelle auf den Testdaten werden sowohl tabellarisch anhand der Evaluationsmetriken als auch grafisch aufbereitet.

Für die Lineare Regression werden die vorhergesagten Punktzahlen gegen die tatsächlichen Punktzahlen in einem Streudiagramm abgetragen. Zusätzlich wird ein Histogramm der Residuen (Differenzen zwischen Vorhersage und wahrem Wert) erstellt, um systematische Verzerrungen zu erkennen.

Für die multinomial logistische Regression und dem Random Forest Modell wird eine Heat Map geplottet um einen besseren Überblick über die Vorhersage Accuracy und Precision zu erhalten.

Zur Interpretation der Modelle werden die Feature Importances betrachtet, die angeben, welche Eingabevariablen den größten Einfluss auf die Vorhersage haben.

7.2 Analyse und Bewertung der Ergebnisse

7.2.1 Lineare Regression

Die Ergebnisse (siehe Anhang 10.3.1 & 10.5.2.1) zeigen, dass die Lineare Regression erwartungsgemäß nur einen geringen Teil der Varianz in den Punktzahlen erklären kann ($R^2=0,42$). Die Vorhersagen weichen im Mittel um 3,40 Punkte von den wahren Werten ab (MAE), das bedeutet, dass die Fahrer Position mit einer Abweichung von 3,29 vorhergesagt wird. Der Mean Squared Error beträgt 18,76, welches sehr hoch ist, aber gemessen daran, dass z.B. Max Verstappen in der Saison 2023 entweder stark gewonnen oder verloren hat, ist nicht verwunderlich.

Precision und Accuracy für Top 10, Top 5 und Top 3: Die Precision und Accuracy für die Top-Positionen sind hoch, insbesondere für die Top 5 und Top 3, wo die Precision 1,00 beträgt. Dies bedeutet, dass das Modell die Fahrer in den Top-Positionen genau vorhersagen kann. Die Accuracy ist ebenfalls hoch, was darauf hindeutet, dass das Modell die meisten Top-Fahrer korrekt identifiziert.

Die Vorhersageergebnisse für das letzte Rennen der Saison sind aufgrund der undiskreten Ergebnisse nur sehr schwer, dennoch kann man erkennen, dass das Modell bei zum Beispiel Max Verstappen, welcher bis dato 21 Rennen gewonnen hat.

In der Feature Importance kann man erkennen, dass Rennerfahrung, die aktuelle Position in der Fahrerweltmeisterschaft, Qualifying Position, Season Siege und die Weltmeistertitel den meisten Einfluss auf das Modell hatten.

7.2.2 Multinomiale Logistische Regression

Um die Vorhersage zu verbessern habe zwei multinomial logistische Regression Modelle implementiert, das erste hat die gleichen Features wie die der linearen Regression, also nur die des Fahrers, das zweite zusätzlich die des Konstrukteurs und der Strecke.

Das erste Modell (siehe Ergebnisse im Anhang 10.3.2 & 10.5.3.2) erreicht eine Model Accuracy von 0,11 bei der Vorhersage von den Fahrerpositionen. Bei den Top 3, 5 und

10 Positionen liegt die Accuracy bei ungefähr 0,8 und einer Precision von 0,8 für die Top 10 , 0,72 für die Top 5 und 0,55 für die Top 3. Im Sachverhalt bedeutet die Precision die richtige Vorhersage von Fahrern die tatsächlich die Top platze belegt haben und die Accuracy zusätzlich die Fahrer die richtig nicht eingeordnet worden sind.

Die Vorhersage Ergebnisse für das letzte Rennen waren nur in einem Fall (Max Verstappen Platz 1 Vorhergesagt und Platz 1 im rennen) korrekt, jedoch war die grobe Einordnung in den meisten Fällen korrekt z.B wurde Kevin Magnussen und Guanyu Zhou korrekt auf die letzten Plätze eingeordnet sowie Lewis Hamilton und Lance Stroll auf das Mittelfeld.

In der Feature Importance kann man erkenne das die aktuelle Position in der Fahrerweltmeisterschaft , Qualifying Positon, Season Siege, Karriere Siege und die Weltmeistertitel den meisten Einfluss auf das Modell hatten. 10 von meinen Features hatten einen sehr geringen Einfluss auf das Modell.

Das zweite Modell (siehe Ergebnisse im Anhang 10.3.3 & 10.5.3.3) erreicht ein Model Accuracy von nur 0,09 und ist somit das schlechtere von beiden. Precision und Accuracy ist für die Top 3, 5, und 10 Positionen jeweils um etwa 0,1 gesunken. In diesem Modell hatte die derzeit Position des Fahrers in der Weltmeisterschaft, Season Siege, Qualifying Position, Konstrukteur Rennsieg und die Anzahl der Konstrukteurtitel den größten Einfluss auf das Modell übernommen.

7.2.3 Random Tree

Der Random Forest liefert im Vergleich zu den anderen Modellen die besten Ergebnisse (siehe Anhang 10.3.4 & 10.5.3.4), wenn auch nur mit einer leichten Verbesserung gegenüber den logistischen Regressionsmodellen. Mit einer Model Accuracy von 0,14 und einer erhöhung der Precision- und Accuracy-Werten für die Top-Positionen kann der Random Forest die Reihenfolge der Fahrer etwas genauer vorhersagen als die anderen Modelle.

Die Precision für die Top 10 liegt bei 0,80, was bedeutet, dass 80% der vom Modell als Top 10 klassifizierten Fahrer tatsächlich in den Top 10 landeten. Die Accuracy für die Top 10 beträgt 0,78, was darauf hindeutet, dass das Modell 78% der Fahrer korrekt als entweder in den Top 10 oder außerhalb der Top 10 eingestuft hat. Für die Top 5 und Top 3 sind die Precision-Werte niedriger (0,67 bzw. 0,52), während der Accuracy-Werte für die Top 5 mit 0,84 höher ist und bei den Top 3 mit 0,84 ebenso. Dies deutet darauf hin, dass das Modell bei der Vorhersage der Fahrer in den Top 5 und Top 3 weniger präzise ist, aber insgesamt gut darin ist, die Fahrer als entweder in diesen Gruppen oder außerhalb davon zu klassifizieren.

Bei der Feature Importance zeigt sich, dass die Qualifying-Position, die aktuelle Position in der Fahrerwertung und die Zeitdifferenz zur Pole-Position im Qualifying den größten Einfluss auf das Modell haben. Dies unterstreicht die Bedeutung der Qualifying-Leistung und der aktuellen Form für die Vorhersage der Rennergebnisse. Auch Faktoren wie die letztjährige Position in der Fahrerwertung, die aktuelle Position des Teams in der Konstrukteurswertung und die durchschnittliche Platzierung in den letzten 5 Rennen spielen eine wichtige Rolle.

Die Vorhersage der Ergebnisse des letzten Rennens zeigt, dass das Modell die Platzierung einiger Fahrer, wie z.B. Max Verstappen auf Platz 1, genau vorhersagen

konnte. Allerdings gibt es auch einige Diskrepanzen, wie z.B. bei Carlos Sainz, der vom Modell auf Platz 6 vorhergesagt wurde, aber tatsächlich auf Platz 18 landete.

7.3 Bewertung der Modelle

Die Vorhersagequalität der Modelle ist von schlecht bis gut zu bewerten.

Die lineare Regression kann man praktisch nicht als Vorhersagemodell gebrauchen, da der R-Squared nicht als Vorhersagemodell zu gebrauchen, aber damit war schon zu rechnen, da lineare Regressionen nicht für komplexe und teils diskrete Daten der Formel 1 nicht gemacht sind.

Jedoch kann man bei diesem Modell gut erkennen, dass es Fahrer gibt die sehr konstante Rennergebnisse liefern wie zum Beispiel Max Verstappen in der 1. Position.

Die multinomiale logistische Regression mit den Fahrer- Features war mit einer MAE (3.59) ähnlich schlecht, jedoch bietet sie im Vergleich zur linearen Regression ein Rennergebnis aus welches diskreten und die platze von 1-20 jeweils einmal enthält. Zudem fällt hier schon auf das Modell overfitted worden ist. Daniel Riccardo ein Fahrer welcher in der Rennsaison 2023 in einem mittelguten Team fährt und in der Saison keine Platzierung im Vorderenbereich geschafft hat wird eine Platzierung als zweites Vorhergesagt, da er einer der älteren Fahrern mit der entsprechenden Erfahrung.

Das zweite multinomial logistische Modell welches zusätzlich noch Konstrukteursfeatures und Streckenfeatures liefert mit einer niedrigeren Accuracy(0.09) und MAE(4.06) ein noch schlechteres Ergebnis. Die Schlussfolgerung das mehr features immer einer besseren Vorhersage gleichen wird widerlegt und es wird belegt das die multinomial logistische Regression kein geeignetes Modell für die komplexe Datenstruktur von den Formel 1 Daten ist.

Das letzte Modell , das Random Forest Modell, bietet mit einer MAE von 3.31 und Modell Accuracy von 0.15 ein wesentlich besseres Ergebnis als die Modelle zuvor.

Abhängigkeiten werden wesentlich besser erkannt und die zusätzlichen features wirken sich positiv auf das Ergebnis aus. Die Rennergebniss Vorhersage ist zwar immer noch nicht genau zu treffend, jedoch liegt eine niedrige Differenz vor.

Zusammenfassend kann man sagen das das letzte Modell gut geeignet ist eine Formel 1 Vorhersage zu machen oder zumindest als Indikator für eine eigene Vorhersage zu erstellen. Die niedrige Accuracy des Modells ist nicht verwunderlich, da wichtige Daten fehlen die Positionsentscheidend sind wie zum Beispiel Wetterdaten, Fahrer Fitness, Team Budget und Strafen.

7.4 Vergleich mit bestehenden Lösungen

Um die Leistungsfähigkeit der entwickelten Modelle einzuordnen habe ich mein Modell mit derzeit bestehenden Lösungen verglichen. Eines der bestehenden Lösungen ist die Website f1-predictor.com . Die Website wird von einem Fan gehostet und nutzt auch die Data Science um eine Vorhersage zu treffen, aber verwendet ein anderes Modell dafür und zwar LightGBM welches sich Gradient Boosting zur nutze macht. Informationen über die verwendeten Daten und Metrik werden leider nicht geteilt, aber die Vorhergesagten Rennergebnisse lassen Sicht noch abrufen.

Eine der Unterschiede die sich direkt rausstellt ist, das genauso wie bei meinem Modell ein Rennergebnis ausgegeben wird und keine Wahrscheinlichkeiten für die Position eines Fahrers.

Zudem wird zusätzlich noch das Qualifying Vorhergesagt.

Am Beispiel von der Vorhersage des Rennergebnisses von Abu Dhabi [1] konnte sein Modell 7 Vorhersagen treffen die richtig sind meines im Vergleich nur 3. Da das Rennen von Abu Dhabi das letzte Rennen habe ich den Vergleich auch noch mal mit dem Anfangsrennen in Qatar gezogen, hier ist unser Modell mit jeweils 2 Richtigen Vorhersagen gleich auf.

Basierend auf der ähnlichen Präzision und der Tatsache das die freiverfügbare Datenmenge überschaubar ist, das hier auch die gleichen Daten benutzt worden sind. Derzeit würde aufgrund der beiden Vergleiche mein Model etwas schlechter einordnen, da das letzte Rennen nicht so viele Treffer hatte und mein Modell keine Vorhersage über das Qualifying Ergebnis trifft. Zusätzlich muss man aber auch in betracht ziehen, dass laut der Information auf der Website das das Projekt stillgelegt wird und keine Vorhersagen mehr gemacht werden, was meine Lösung derzeit die einzige für mich zugängliche macht.

Erwähnenswert ist noch das es fortgeschrittene in der Rennsaison es im Qualifying Vorhersagen über das Rennergebnis des Wochenendes gibt, jedoch wird dieses nicht immer gemacht und auch nur für einzelne Fahrer.

8. Diskussion und Ausblick

8.1 Zusammenfassung der Ergebnisse

Die entwickelten Machine Learning Modelle zur Vorhersage von Formel 1 Rennergebnissen liefern vielversprechende Ergebnisse, zeigen jedoch auch Limitationen auf, die Raum für zukünftige Verbesserungen bieten.

Die lineare Regression erwies sich als ungeeignet für die komplexen, nicht-linearen Zusammenhänge in den Formel 1 Daten. Die multinomiale logistische Regression erzielte bessere Ergebnisse, insbesondere bei der Vorhersage von Top-Platzierungen. Allerdings lassen die niedrigen Accuracy-Werte darauf schließen, dass das Modell Schwierigkeiten hat, die genaue Reihenfolge der Fahrer vorherzusagen. Der Random Forest lieferte die besten Ergebnisse mit einer leichten Verbesserung gegenüber der logistischen Regression. Dennoch bleibt die Gesamtgenauigkeit der Vorhersagen begrenzt.

Die Analyse der Feature Importance zeigte, dass Faktoren wie Qualifying-Ergebnisse, aktuelle Fahrer- und Konstrukteurswertung sowie historische Leistungsdaten den größten Einfluss auf die Vorhersagen haben. Gleichzeitig fehlen in den Modellen jedoch wichtige Informationen wie Wetterdaten, Fahrerfitness, Teambudgets und Strafversetzungen, die sich entscheidend auf das Rennergebnis auswirken können.

Insgesamt demonstriert das Projekt die Wichtigkeit der Richtigen Modellauswahl und Features für die Vorhersage. Die Vorhersagen der entwickelten Modelle und spezifisch der des Random Forest Modells bieten eine gute Grundlage um Schlüsse für eine eigene Vorhersage zu treffen. Nachwirkend kann man auch sagen, dass eine sehr präzise Vorhersage des Rennergebnisses basierend auf vorherigen Rennergebnissen nicht möglich ist.

8.2 Mögliche zukünftige Erweiterungen

Um die Vorhersagequalität weiter zu verbessern, sollten zukünftig zusätzliche Datenquellen integriert werden. Dazu gehören detaillierte Streckendaten, Reifenstrategien, Team-Budget und Wetterdaten. Auch eine Berücksichtigung qualitativer Faktoren wie Medienberichte und Expertenmeinungen könnte wertvolle Erkenntnisse liefern. Zudem könnte man das Modell um das Cross-Validation Verfahren erweitert werden um noch mehr aus den Daten rauszuholen, da in diesem Verfahren mehrere Test mit unterschiedlichen Datenaufteilungen gemacht werden und danach der durchschnitt ausgerechnet wird.

Neben den verwendeten klassischen Modellen könnte somit auch die Verwendung von Deep Learning Methoden eine Verbesserung der Vorhersage bringen bzw. eine neue Einsicht über die Daten bringen. Das Training von Neuronalen Netzwerken erfordert jedoch eine besonders starke Graphikkarte und speziell eine von Nvidia, um die CUDA-Technologie zur Modellierung nutzen zu können. Der Einsatz von Deep Learning Architekturen wie Convolutional Neural Networks (CNNs) für die Analyse von Strecken- und Wetterdaten oder Long Short-Term Memory (LSTM) Netzwerke für die Modellierung von Zeitreihen könnte die Vorhersagegenauigkeit weiter steigern. Allerdings ist dabei zu beachten, dass diese Ansätze auch deutlich mehr Trainingsdaten und Rechenressourcen erfordern als klassische Machine Learning Verfahren.

Derzeit benötigt es eine manuelle Veränderung am Code, um eine Vorhersage für ein bestimmtes Rennen zu erhalten, was zeitaufwendig ist und das Projekt unzugänglich für Dritte macht. Zukünftig wäre der Ausbau des Projektes zu einer Dienstleistung in Form einer Website oder mobilen App sinnvoll, um die Vorhersagen einem breiteren Publikum zur Verfügung zu stellen. Über eine benutzerfreundliche Oberfläche könnten Interessierte eigene Prognosen erstellen, historische Renndaten analysieren und die Auswirkungen verschiedener Szenarien simulieren. Zusätzlich ließe sich ein API-Zugang einrichten, der es Entwicklern erlaubt, die Vorhersagefunktionalitäten in eigene Anwendungen wie Formel-1-Spiele zu integrieren.

Des weiteren wäre eine Erweiterung des Modells auf Vorhersagen über das Qualifying eine sinnvolle Ergänzung und würde somit auch eine vollständige Alternative zu der alternative formula1-predictor.com bieten.

Schließlich könnte das Vorhersagemodell auch auf andere Rennserien wie die Formel E, übertragen werden. Dafür müssten die Modelle entsprechend angepasst und mit serienspezifischen Daten trainiert werden. Ein Vergleich der Prognosequalität über verschiedene Rennserien hinweg könnte zudem Erkenntnisse über die Gemeinsamkeiten und Unterschiede der Erfolgsfaktoren im Motorsport liefern.

9. Fazit

Das vorliegende Projekt demonstriert wie Machine Learning Methoden genutzt werden können, um Formel 1 Rennergebnisse vorherzusagen. Gleichzeitig zeigt es aber auch die Notwendigkeit einer kontinuierlichen Evaluation und Verbesserung der Modelle, um die Vorhersagequalität zu optimieren.

Für die Modellierung kamen drei verschiedene Ansätze zum Einsatz, die Lineare Regression, Multinomiale Logistische Regression und Random Forest. Dabei erwies sich die Lineare Regression aufgrund der komplexen, nicht-linearen Zusammenhänge in den Formel 1 Daten als ungeeignet. Die Multinomiale Logistische Regression erzielte bessere Ergebnisse, insbesondere bei der Vorhersage von Top-Platzierungen, hatte jedoch Schwierigkeiten, die genaue Reihenfolge der Fahrer vorherzusagen. Der Random Forest lieferte die besten Resultate, wenngleich die Gesamtgenauigkeit der Prognosen noch Raum für Verbesserungen lässt.

Es wurde deutlich, dass die Modelle von zusätzlichen Informationen wie Wetterdaten, Fahrerfitness, Teambudgets und Strafversetzungen profitieren würden, um noch präzisere Vorhersagen zu treffen. Das Fehlen dieser Daten schränkt die Effektivität des Projekts als Vorhersagemodell ein.

Im Nachhinein wäre eine alternative gewesen anstatt ein drittes Modell zu implementieren mehr Zeit in die Datenbeschaffung zu stecken um mehr Features für die Vorhersage zu haben, jedoch wäre aufgrund der Limitierung der multinominalen logistischen Regression komplexe Strukturen abzubilden an die Grenze gestoßen.

Mit meiner Arbeit konnte ich zudem beweisen das Domänenwissen für einen Data Scientist essenziell ist um aus den begrenzten Daten eine hohe Feature Extraktion zu haben, aber auch abzuwägen welche Information wirklich entscheidend sind und nicht zufällig eine Abhängigkeit darstellen, was bei der relativ geringen Datenmenge provoziert wird.

Die Ergebnisse belegen, dass eine Vorhersage von Rennergebnissen der Formel 1 basierend auf historische Daten grundsätzlich möglich ist, offenbaren aber auch die vielen verschiedenen Einflussfaktoren wofür es keine oder bzw. keine freien Verfügbaren Daten gibt.

Die Faszination und Unberechenbarkeit der Formel 1 bleibt somit erhalten und genau das macht den Reiz dieses Sports aus.

Das Projekt war für mich das erste in dieser Art und hat mir eine Menge Freude bereitet. Die Arbeit hat mir das wissenschaftliche Arbeiten näher gebracht.. Dabei habe ich wertvolle Erfahrungen im Umgang mit begrenzten Datenmengen und der Bewältigung von Klassifikationsproblemen gesammelt. Durch die Implementierung und den Vergleich von drei verschiedenen Modelltypen konnte ich ein tieferes Verständnis für die Stärken und Schwächen unterschiedlicher Ansätze entwickeln. Insbesondere habe ich gelernt, Modelle anhand verschiedener Metriken zu evaluieren und zu unterscheiden, was ein gutes von einem schlechten Modell ausmacht. Dabei wurde mir auch bewusst, wie wichtig es ist, über die Standardmetriken hinauszublicken und eigene zu definieren.

10. Anhang

10.1 Quelltext

10.1.1 DDL

```
DROP TABLE IF EXISTS circuits;
CREATE TABLE circuits
(
    circuitId SERIAL PRIMARY KEY,
    -- circuitRef VARCHAR(255) NOT NULL DEFAULT '',
    name    VARCHAR(255) NOT NULL DEFAULT '',
    location VARCHAR(255),
    country VARCHAR(255),
    lat     FLOAT,
    lng     FLOAT,
    alt     INT,
    url     VARCHAR(255) NOT NULL DEFAULT '',
    UNIQUE (url)
);

DROP TABLE IF EXISTS status;
CREATE TABLE status
(
    statusId SERIAL PRIMARY KEY,
    status VARCHAR(255) NOT NULL DEFAULT '',
    UNIQUE (status)
);

DROP TABLE IF EXISTS constructors;
CREATE TABLE constructors
(
    constructorId SERIAL PRIMARY KEY,
    -- constructorRef VARCHAR(255) NOT NULL DEFAULT '',
    name          VARCHAR(255) NOT NULL DEFAULT '',
    nationality VARCHAR(255),
    url           VARCHAR(255) NOT NULL DEFAULT '',
    UNIQUE (url, name)
);

DROP TABLE IF EXISTS drivers;
CREATE TABLE drivers
(
    driverId SERIAL PRIMARY KEY,
    driverRef VARCHAR(255) NOT NULL DEFAULT '',
    -- number INT,
    --code VARCHAR(3),
```



```

    forename VARCHAR(255) NOT NULL DEFAULT '',
    surname  VARCHAR(255) NOT NULL DEFAULT '',
    dob      DATE,
    nationality VARCHAR(255),
    url      VARCHAR(255) NOT NULL DEFAULT '',
    UNIQUE (url, forename, surname)
);

```

```

DROP TABLE IF EXISTS seasons;
CREATE TABLE seasons
(
    year INT PRIMARY KEY,
    url VARCHAR(255) NOT NULL DEFAULT '',
    UNIQUE (url)
);

```

```

DROP TABLE IF EXISTS races;
CREATE TABLE races
(
    raceId SERIAL PRIMARY KEY,
    year INT NOT NULL DEFAULT '0',
    round INT NOT NULL DEFAULT '0',
    circuitId INT NOT NULL DEFAULT '0',
    name VARCHAR(255) NOT NULL DEFAULT '',
    date DATE NOT NULL DEFAULT '1970-01-01', -- Updated default date
    url VARCHAR(255),
    FOREIGN KEY (circuitId) REFERENCES circuits (circuitId),
    Foreign KEY (year) REFERENCES seasons (year),
    UNIQUE (url, name)
);

```

```

DROP TABLE IF EXISTS qualifying;
CREATE TABLE qualifying
(
    qualifyId SERIAL PRIMARY KEY,
    raceId INT NOT NULL DEFAULT '0',
    driverId INT NOT NULL DEFAULT '0',
    constructorId INT NOT NULL DEFAULT '0',
    number INT NOT NULL DEFAULT '0',
    position INT,
    q1 TIME(3),
    q2 TIME(3),
    q3 TIME(3),
    UNIQUE (raceId, driverId, constructorId),
    FOREIGN KEY (constructorId) REFERENCES constructors (constructorId),
    FOREIGN KEY (driverId) REFERENCES drivers (driverId),
    FOREIGN KEY (raceId) references races (raceId),

```

```

    UNIQUE (qualifyId, raceId, driverId)
);

```

DROP TABLE IF EXISTS results;

CREATE TABLE results

```

(
    resultId      SERIAL PRIMARY KEY,
    raceId        INT          NOT NULL DEFAULT '0',
    driverId       INT          NOT NULL DEFAULT '0',
    constructorId INT          NOT NULL DEFAULT '0',
    number         INT,
    grid           INT          NOT NULL DEFAULT '0',
    position       INT,
    positionText   VARCHAR(255) NOT NULL DEFAULT '',
    -- positionOrder INT NOT NULL DEFAULT '0',
    points         FLOAT        NOT NULL DEFAULT '0',
    laps           INT          NOT NULL DEFAULT '0',
    time           varchar(250),
    -- milliseconds INT,
    fastestLap     INT,
    rank           INT          DEFAULT '0',
    fastestLapTime TIME(3),
    fastestLapSpeed VARCHAR(255),
    statusId       INT          NOT NULL DEFAULT '0',
    foreign key (statusId) REFERENCES status (statusId),
    FOREIGN KEY (raceId) references races (raceId),
    FOREIGN KEY (driverId) REFERENCES drivers (driverId),
    FOREIGN KEY (constructorId) REFERENCES constructors (constructorId),
    UNIQUE (raceId, driverId, constructorId)
);

```

DROP TABLE IF EXISTS sprintResults;

CREATE TABLE sprintResults

```

(
    sprintResultId SERIAL PRIMARY KEY,
    raceId          INT          NOT NULL DEFAULT '0',
    driverId         INT          NOT NULL DEFAULT '0',
    constructorId    INT          NOT NULL DEFAULT '0',
    number           INT          NOT NULL DEFAULT '0',
    grid            INT          NOT NULL DEFAULT '0',
    position         INT,
    positionText     VARCHAR(255) NOT NULL DEFAULT '',
    -- positionOrder INT NOT NULL DEFAULT '0',
    points           FLOAT        NOT NULL DEFAULT '0',
    laps            INT          NOT NULL DEFAULT '0',
    time            VARCHAR(255),
    -- milliseconds INT,
    fastestLap       INT,

```

```

fastestLapTime VARCHAR(255),
statusId      INT      NOT NULL DEFAULT '0',
FOREIGN KEY (raceId) REFERENCES races (raceId),
FOREIGN KEY (driverId) REFERENCES drivers (driverId),
FOREIGN KEY (constructorId) REFERENCES constructors (constructorId),
foreign key (statusId) references status (statusId),
UNIQUE (driverId, sprintResultId)
);

```

DROP TABLE IF EXISTS constructorStandings;

CREATE TABLE constructorStandings

```

(
  constructorStandingsId SERIAL PRIMARY KEY,
  raceId                INT  NOT NULL DEFAULT '0',
  constructorId         INT  NOT NULL DEFAULT '0',
  points                FLOAT NOT NULL DEFAULT '0',
  position              INT,
  positionText          VARCHAR(255),
  wins                  INT  NOT NULL DEFAULT '0',
  FOREIGN KEY (raceId) REFERENCES races (raceId),
  FOREIGN KEY (constructorId) REFERENCES constructors (constructorId),
  Unique (raceId, constructorId)
);

```

DROP TABLE IF EXISTS driverStandings;

CREATE TABLE driverStandings

```

(
  driverStandingsId SERIAL PRIMARY KEY,
  raceId            INT  NOT NULL DEFAULT '0',
  driverId          INT  NOT NULL DEFAULT '0',
  points            FLOAT NOT NULL DEFAULT '0',
  position          INT,
  positionText      VARCHAR(255),
  wins              INT  NOT NULL DEFAULT '0',
  FOREIGN KEY (raceId) REFERENCES races (raceId),
  FOREIGN KEY (driverId) REFERENCES drivers (driverId),
  unique (raceId, driverId)
);

```

DROP TABLE IF EXISTS lapTimes;

CREATE TABLE lapTimes

```

(
  raceId INT NOT NULL,
  driverId INT NOT NULL,
  lap INT NOT NULL,
  position INT,
  time Time(3),
  -- milliseconds INT,

```

```

PRIMARY KEY (raceId, driverId, lap),
FOREIGN KEY (raceId) REFERENCES races (raceId),
FOREIGN KEY (driverId) REFERENCES drivers (driverId),
unique (raceId, driverId, lap)
);

```

```

DROP TABLE IF EXISTS pitStops;
CREATE TABLE pitStops
(
    raceId INT NOT NULL,
    driverId INT NOT NULL,
    stop INT NOT NULL,
    lap INT NOT NULL,
    time TIME NOT NULL,
    duration VARCHAR(255),
-- milliseconds INT,
PRIMARY KEY (raceId, driverId, stop),
FOREIGN KEY (raceId) REFERENCES races (raceId),
FOREIGN KEY (driverId) REFERENCES drivers (driverId),
unique (raceId, driverId, stop)
);

```

10.2.2 Test-, Trainingsdaten split mit Sklearn

```

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

```

10.2.3 Grid Search Optimierung

```

model = Ridge()

param_grid = {'alpha': [0.1, 1.0, 10.0]}

grid_search = GridSearchCV(model, param_grid, cv=5,
scoring='neg_mean_absolute_error')

grid_search.fit(X_train, y_train)

best_model = grid_search.best_estimator_

```

10.2.4 Random Search Optimierung

```

param_distributions = {
    'n_estimators': randint(50, 500),
    'max_depth': [None] + list(range(5, 50, 5)),
    'min_samples_split': randint(2, 20),
    'min_samples_leaf': randint(1, 10),

```

```

    'max_features': ['sqrt', 'log2', None],
    'bootstrap': [True, False]
}

rf = RandomForestClassifier(random_state=random_state)

random_search = RandomizedSearchCV(estimator=rf,
param_distributions=param_distributions, n_iter=100, cv=5,
n_jobs=-1, verbose=2, random_state=random_state)

random_search.fit(X_train, y_train)

best_model = random_search.best_estimator_

```

10.2.5 Feature Importance

```

feature_importance = pd.DataFrame({'Feature': X.columns,
'Importance': best_model.coef_})

feature_importance = feature_importance.sort_values('Importance',
ascending=False)

print("\nFeature Importance:")

print(feature_importance)

```

10.2.6 Precision und Accuracy für Top 3, 5 und 10

```

y_test_top10 = y_test.apply(lambda x: 1 if x <= 10 else 0)
y_pred_top10 = pd.Series(y_pred).apply(lambda x: 1 if x <= 10 else
0)

precision_top10 = precision_score(y_test_top10, y_pred_top10)
accuracy_top10 = accuracy_score(y_test_top10, y_pred_top10)
print(f"Precision for Top 10: {precision_top10:.2f}")
print(f"Accuracy for Top 10: {accuracy_top10:.2f}")

y_test_top5 = y_test.apply(lambda x: 1 if x <= 5 else 0)
y_pred_top5 = pd.Series(y_pred).apply(lambda x: 1 if x <= 5 else
0)

precision_top5 = precision_score(y_test_top5, y_pred_top5)
accuracy_top5 = accuracy_score(y_test_top5, y_pred_top5)

```

```

print(f"Precision for Top 5: {precision_top5:.2f}")
print(f"Accuracy for Top 5: {accuracy_top5:.2f}")

y_test_top3 = y_test.apply(lambda x: 1 if x <= 3 else 0)
y_pred_top3 = pd.Series(y_pred).apply(lambda x: 1 if x <= 3 else 0)

precision_top3 = precision_score(y_test_top3, y_pred_top3)
accuracy_top3 = accuracy_score(y_test_top3, y_pred_top3)
print(f"Precision for Top 3: {precision_top3:.2f}")
print(f"Accuracy for Top 3: {accuracy_top3:.2f}")

```

10.2.7 MAE Berechnung

```
mae = mean_absolute_error(y_test, y_pred)
```

10.2.8 MSE UND R²

```

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

```

10.2.9 Implementierung Modelle

10.2.9.1 Lineare Regression

```

model = Ridge() //Klasse Lineare Regression
grid_search = GridSearchCV(model, param_grid, cv=5,
scoring='neg_mean_absolute_error')
grid_search.fit(X_train, y_train) //fit

```

10.2.9.2 Multinomiale Logistische Regression

```

model = LogisticRegression(multi_class='multinomial',
solver='lbfgs', max_iter=10000000)
model.fit(X_train, y_train)

```

10.2.9.3 Random Tree

```
rf = RandomForestClassifier(random_state=random_state)
```

```

random_search = RandomizedSearchCV(estimator=rf,
param_distributions=param_distributions, n_iter=100, cv=5,
n_jobs=-1, verbose=2, random_state=random_state)

random_search.fit(X_train, y_train)

```

10.3 Model Ergebnisse

10.3.1 Lineare Regression

File - LiR

```

2 Mean Absolute Error (MAE): 3.40
3 Mean Squared Error: 18.76
4 R-squared: 0.42
5 Precision for Top 10: 0.87
6 Accuracy for Top 10: 0.78
7 Precision for Top 5: 0.91
8 Accuracy for Top 5: 0.78
9 Precision for Top 3: 1.00
10 Accuracy for Top 3: 0.86
11
12 Feature Importance:
13     Feature Importance
14 18     QualifyingPosition  0.312204
15 11     CurrentStandings  0.299722
16 0      Age  0.155215
17 8      PolePositions  0.084878
18 6      RacesStarted  0.080405
19 14     PrevYearRacePitStops  0.005583
20 13     TimeDiffToWinnerLastRace  0.002953
21 16     AvgLapTimeConsistencyLastRace  0.001349
22 2      QualifyingTimeDiffToPole  0.000062
23 17     SpeedRankLastRace  0.000000
24 15     AvgLapTimeLastRace  -0.002322
25 4      CareerWins  -0.008052
26 10     LastYearStandings  -0.010770
27 3      FrontRowStarts  -0.012791
28 9      DriverChampionships  -0.048007
29 7      RacesFinished  -0.090169
30 1      YearsInF1  -0.190492
31 12     QualifyingRaceCorrelation  -0.191586
32 5      SeasonWins  -0.249244
33 Predicted Last Race Results:
34     Driver PredictedPosition ActualPosition
35 18 (Max, Verstappen)  -0.821218 1
36 6  (Charles, Leclerc)  7.334724 2
37 13 (George, Russell)  7.622025 3
38 10 (Sergio, Pérez)  6.162626 4
39 8  (Lando, Norris)  6.609717 5
40 11 (Oscar, Piastri)  7.579117 6
41 1  (Fernando, Alonso)  7.818790 7
42 17 (Yuki, Tsunoda)  10.070483 8
43 4  (Lewis, Hamilton)  7.074847 9
44 16 (Lance, Stroll)  11.161274 10
45 12 (Daniel, Ricciardo)  14.324793 11
46 9  (Esteban, Ocon)  11.768911 12
47 3  (Pierre, Gasly)  10.188200 13
48 0  (Alexander, Albon)  12.668529 14
49 5  (Nico, Hülkenberg)  12.382573 15
50 15 (Logan, Sargeant)  17.190947 16
51 19 (Guanyu, Zhou)  15.906105 17
52 14 (Carlos, Sainz)  12.091521 18
53 2  (Valtteri, Bottas)  14.878141 19
54 7  (Kevin, Magnussen)  16.015312 20
55

```

10.3.2 Multinomiale logistische Regression: Variante 1

File - MNLR_1

| | | | | | |
|----|---------------------------------|---------------------------|------------|----------|------------|
| 10 | Model Accuracy: 0.11 | | | | |
| 11 | Mean Absolute Error (MAE): 3.59 | | | | |
| 12 | | precision | recall | f1-score | support |
| 13 | | | | | |
| 14 | 1 | 0.67 | 1.00 | 0.80 | 6 |
| 15 | 2 | 0.23 | 0.30 | 0.26 | 10 |
| 16 | 3 | 0.14 | 0.08 | 0.11 | 12 |
| 17 | 4 | 0.00 | 0.00 | 0.00 | 6 |
| 18 | 5 | 0.27 | 0.23 | 0.25 | 13 |
| 19 | 6 | 0.00 | 0.00 | 0.00 | 8 |
| 20 | 7 | 0.25 | 0.08 | 0.12 | 13 |
| 21 | 8 | 0.00 | 0.00 | 0.00 | 8 |
| 22 | 9 | 0.00 | 0.00 | 0.00 | 10 |
| 23 | 10 | 0.00 | 0.00 | 0.00 | 7 |
| 24 | 11 | 0.00 | 0.00 | 0.00 | 8 |
| 25 | 12 | 0.17 | 0.12 | 0.14 | 8 |
| 26 | 13 | 0.11 | 0.10 | 0.11 | 10 |
| 27 | 14 | 0.06 | 0.20 | 0.09 | 5 |
| 28 | 15 | 0.00 | 0.00 | 0.00 | 4 |
| 29 | 16 | 0.14 | 0.10 | 0.12 | 10 |
| 30 | 17 | 0.50 | 0.08 | 0.13 | 13 |
| 31 | 18 | 0.00 | 0.00 | 0.00 | 7 |
| 32 | 19 | 0.00 | 0.00 | 0.00 | 5 |
| 33 | 20 | 0.00 | 0.00 | 0.00 | 9 |
| 34 | | | | | |
| 35 | accuracy | | | 0.11 | 172 |
| 36 | macro avg | 0.13 | 0.11 | 0.11 | 172 |
| 37 | weighted avg | 0.15 | 0.11 | 0.11 | 172 |
| 38 | | | | | |
| 39 | Precision for Top 10: 0.80 | | | | |
| 40 | Accuracy for Top 10: 0.76 | | | | |
| 41 | Precision for Top 5: 0.72 | | | | |
| 42 | Accuracy for Top 5: 0.87 | | | | |
| 43 | Precision for Top 3: 0.55 | | | | |
| 44 | Accuracy for Top 3: 0.85 | | | | |
| 45 | Feature Importance: | | | | |
| 46 | | Feature | Importance | Absolute | Importance |
| 47 | 11 | CurrentStandings | -0.703175 | | 0.703175 |
| 48 | 18 | QualifyingPosition | -0.653957 | | 0.653957 |
| 49 | 4 | CareerWins | 0.381311 | | 0.381311 |
| 50 | 5 | SeasonWins | 0.318259 | | 0.318259 |
| 51 | 9 | DriverChampionships | -0.301615 | | 0.301615 |
| 52 | 14 | PrevYearRacePitStops | 0.229740 | | 0.229740 |
| 53 | 0 | Age | 0.179822 | | 0.179822 |
| 54 | 1 | YearsInF1 | 0.160255 | | 0.160255 |
| 55 | 10 | LastYearStandings | 0.116212 | | 0.116212 |
| 56 | 7 | RacesFinished | -0.054642 | | 0.054642 |
| 57 | 6 | RacesStarted | 0.028406 | | 0.028406 |
| 58 | 3 | FrontRowStarts | -0.024584 | | 0.024584 |
| 59 | 8 | PolePositions | 0.016480 | | 0.016480 |
| 60 | 13 | TimeDiffToWinnerLastRace | -0.012844 | | 0.012844 |
| 61 | 17 | SpeedRankLastRace | 0.009290 | | 0.009290 |
| 62 | 15 | AvgLapTimeLastRace | 0.008563 | | 0.008563 |
| 63 | 12 | QualifyingRaceCorrelation | 0.001714 | | 0.001714 |

Page 1 of 2


```

64 16 AvgLapTimeConsistencyLastRace -0.001355      0.001355
65 2   QualifyingTimeDiffToPole 0.000328      0.000328
66 Predicted Last Race Results:
67   PredictedPosition ActualPosition      Driver Probability
68 18           1           1 (Max, Verstappen) 0.944202
69 6            3           2 (Charles, Leclerc) 0.291186
70 13           16          3 (George, Russell) 0.151464
71 10            4           4 (Sergio, Pérez) 0.272280
72 8             14          5 (Lando, Norris) 0.159039
73 11            11          6 (Oscar, Piastri) 0.194409
74 1             8           7 (Fernando, Alonso) 0.230039
75 17            19          8 (Yuki, Tsunoda) 0.137389
76 4             5           9 (Lewis, Hamilton) 0.244318
77 16            7           10 (Lance, Stroll) 0.233819
78 12            2           11 (Daniel, Ricciardo) 0.298564
79 9             20          12 (Esteban, Ocon) 0.113098
80 3             17          13 (Pierre, Gasly) 0.147663
81 0             18          14 (Alexander, Albon) 0.143129
82 5             10          15 (Nico, Hülkenberg) 0.196286
83 15            13          16 (Logan, Sargeant) 0.180422
84 19            15          17 (Guanyu, Zhou) 0.156731
85 14             6           18 (Carlos, Sainz) 0.234377
86 2             9           19 (Valtteri, Bottas) 0.201778
87 7             12          20 (Kevin, Magnussen) 0.187470
88
89 Process finished with exit code 0

```

10.3.3 Multinomiale Logistische Regression: Variante 2

File - MNLR_2

| | | | | | |
|----|---------------------------------|----------------------------------|------------|---------------------|---------|
| 10 | Model Accuracy: 0.09 | | | | |
| 11 | Mean Absolute Error (MAE): 4.06 | | | | |
| 12 | | precision | recall | f1-score | support |
| 13 | | | | | |
| 14 | 1 | 0.67 | 1.00 | 0.80 | 6 |
| 15 | 2 | 0.18 | 0.20 | 0.19 | 10 |
| 16 | 3 | 0.09 | 0.08 | 0.09 | 12 |
| 17 | 4 | 0.00 | 0.00 | 0.00 | 6 |
| 18 | 5 | 0.22 | 0.15 | 0.18 | 13 |
| 19 | 6 | 0.07 | 0.12 | 0.09 | 8 |
| 20 | 7 | 0.00 | 0.00 | 0.00 | 13 |
| 21 | 8 | 0.00 | 0.00 | 0.00 | 8 |
| 22 | 9 | 0.00 | 0.00 | 0.00 | 10 |
| 23 | 10 | 0.00 | 0.00 | 0.00 | 7 |
| 24 | 11 | 0.00 | 0.00 | 0.00 | 8 |
| 25 | 12 | 0.00 | 0.00 | 0.00 | 8 |
| 26 | 13 | 0.00 | 0.00 | 0.00 | 10 |
| 27 | 14 | 0.07 | 0.20 | 0.11 | 5 |
| 28 | 15 | 0.00 | 0.00 | 0.00 | 4 |
| 29 | 16 | 0.33 | 0.20 | 0.25 | 10 |
| 30 | 17 | 0.00 | 0.00 | 0.00 | 13 |
| 31 | 18 | 0.00 | 0.00 | 0.00 | 7 |
| 32 | 19 | 0.00 | 0.00 | 0.00 | 5 |
| 33 | 20 | 0.08 | 0.11 | 0.10 | 9 |
| 34 | | | | | |
| 35 | accuracy | | | 0.09 | 172 |
| 36 | macro avg | 0.09 | 0.10 | 0.09 | 172 |
| 37 | weighted avg | 0.09 | 0.09 | 0.09 | 172 |
| 38 | | | | | |
| 39 | Precision for Top 10: 0.79 | | | | |
| 40 | Accuracy for Top 10: 0.73 | | | | |
| 41 | Precision for Top 5: 0.72 | | | | |
| 42 | Accuracy for Top 5: 0.84 | | | | |
| 43 | Precision for Top 3: 0.61 | | | | |
| 44 | Accuracy for Top 3: 0.88 | | | | |
| 45 | Feature Importance: | | | | |
| 46 | | Feature | Importance | Absolute Importance | |
| 47 | 18 | QualifyingPosition | -0.572155 | 0.572155 | |
| 48 | 11 | CurrentStandings | -0.379382 | 0.379382 | |
| 49 | 5 | SeasonWins | 0.331720 | 0.331720 | |
| 50 | 20 | ConstructorRaceWins | 0.318921 | 0.318921 | |
| 51 | 10 | LastYearStandings | 0.263346 | 0.263346 | |
| 52 | 19 | ConstructorChampionshipsWon | -0.254341 | 0.254341 | |
| 53 | 14 | PrevYearRacePitStops | 0.206595 | 0.206595 | |
| 54 | 23 | ConstructorStandingsLastYear | -0.205894 | 0.205894 | |
| 55 | 27 | MaxPositionsLostLast5Races | -0.189740 | 0.189740 | |
| 56 | 4 | CareerWins | 0.183050 | 0.183050 | |
| 57 | 29 | BestFinishLast5Races | -0.165820 | 0.165820 | |
| 58 | 34 | ConstructorRetirementsLast5Races | -0.161478 | 0.161478 | |
| 59 | 31 | AvgFinishLast5Races | -0.146825 | 0.146825 | |
| 60 | 9 | DriverChampionships | -0.114321 | 0.114321 | |
| 61 | 24 | CurrentConstructorStandings | -0.093709 | 0.093709 | |
| 62 | 30 | WorstFinishLast5Races | 0.083535 | 0.083535 | |
| 63 | 28 | AvgPositionsGainedLostLast5Races | -0.081543 | 0.081543 | |

Page 1 of 2

```

64 1 YearsInF1 0.080792 0.080792
65 26 MaxPositionsGainedLast5Races 0.072046 0.072046
66 22 ConstructorChampionshipsLast5Years 0.063428 0.063428
67 3 FrontRowStarts -0.061632 0.061632
68 7 RacesFinished -0.055172 0.055172
69 8 PolePositions 0.047462 0.047462
70 32 ConstructorSpeedDiffToFastestLastRace -0.044119 0.044119
71 6 RacesStarted 0.039079 0.039079
72 25 MaxTeamMateQualifyingPosition 0.033464 0.033464
73 21 ConstructorSeasonWins -0.022542 0.022542
74 36 MinSpeedLastRace 0.016711 0.016711
75 0 Age 0.013651 0.013651
76 33 ConstructorRetirements -0.011176 0.011176
77 13 TimeDiffToWinnerLastRace -0.009452 0.009452
78 15 AvgLapTimeLastRace -0.008842 0.008842
79 37 AvgSpeedLastRace 0.006567 0.006567
80 35 MaxSpeedLastRace -0.003576 0.003576
81 12 QualifyingRaceCorrelation -0.003409 0.003409
82 17 SpeedRankLastRace 0.000595 0.000595
83 38 ConstructorSpeedRankLastRace 0.000595 0.000595
84 2 QualifyingTimeDiffToPole -0.000189 0.000189
85 16 AvgLapTimeConsistencyLastRace -0.000118 0.000118
86 Predicted Last Race Results:
87 PredictedPosition ActualPosition Driver Probability
88 18 1 (Max, Verstappen) 0.909776
89 6 13 (Charles, Leclerc) 0.220065
90 13 6 (George, Russell) 0.292585
91 10 4 (Sergio, Pérez) 0.434207
92 8 2 (Lando, Norris) 0.488617
93 11 12 (Oscar, Piastri) 0.226731
94 1 9 (Fernando, Alonso) 0.282930
95 17 18 (Yuki, Tsunoda) 0.185709
96 4 7 (Lewis, Hamilton) 0.288164
97 16 15 (Lance, Stroll) 0.201702
98 12 16 (Daniel, Ricciardo) 0.195944
99 9 8 (Esteban, Ocon) 0.287543
100 3 20 (Pierre, Gasly) 0.177082
101 0 5 (Alexander, Albon) 0.304989
102 5 10 (Nico, Hülkenberg) 0.242798
103 15 17 (Logan, Sargeant) 0.192959
104 19 11 (Guanyu, Zhou) 0.229578
105 14 19 (Carlos, Sainz) 0.181344
106 2 14 (Valtteri, Bottas) 0.203783
107 7 3 (Kevin, Magnussen) 0.445114
108
109 Process finished with exit code 0
110

```

10.3.4 Random Tree

File - RT

```

503 Best parameters: {'bootstrap': False, 'max_depth': 5, 'max_features': 'log2', 'min_sam
504 Best score: 0.15115836242462713
505 Model Accuracy: 0.10
506 Mean Absolute Error (MAE): 3.31
507     precision    recall  f1-score   support
508
509      1       0.75       1.00       0.86         6
510      2       0.08       0.10       0.09        10
511      3       0.00       0.00       0.00        12
512      4       0.00       0.00       0.00         6
513      5       0.00       0.00       0.00        13
514      6       0.00       0.00       0.00         8
515      7       0.00       0.00       0.00        13
516      8       0.14       0.12       0.13         8
517      9       0.10       0.10       0.10        10
518     10       0.10       0.14       0.12         7
519     11       0.00       0.00       0.00         8
520     12       0.00       0.00       0.00         8
521     13       0.12       0.10       0.11        10
522     14       0.07       0.40       0.12         5
523     15       0.08       0.25       0.12         4
524     16       0.29       0.40       0.33        10
525     17       0.00       0.00       0.00        13
526     18       0.00       0.00       0.00         7
527     19       1.00       0.00       0.00         5
528     20       1.00       0.00       0.00         9
529
530     accuracy                0.10       172
531     macro avg       0.19       0.13       0.10       172
532     weighted avg    0.16       0.10       0.08       172
533
534 Precision for Top 10: 0.80
535 Accuracy for Top 10: 0.78
536 Precision for Top 5: 0.67
537 Accuracy for Top 5: 0.84
538 Precision for Top 3: 0.52
539 Accuracy for Top 3: 0.84
540 Feature Importance:
541     Feature Importance
542 11      CurrentStandings 0.129857
543 10      LastYearStandings 0.086270
544 2       QualifyingTimeDiffToPole 0.072963
545 18      QualifyingPosition 0.071051
546 5       SeasonWins 0.060349
547 24      CurrentConstructorStandings 0.058535
548 22      ConstructorChampionshipsLast5Years 0.054782
549 23      ConstructorStandingsLastYear 0.036847
550 31      AvgFinishLast5Races 0.027440
551 21      ConstructorSeasonWins 0.025314
552 13      TimeDiffToWinnerLastRace 0.023574
553 30      WorstFinishLast5Races 0.023120
554 25      MaxTeammateQualifyingPosition 0.021217
555 20      ConstructorRaceWins 0.019725
556 3       FrontRowStarts 0.018515

```

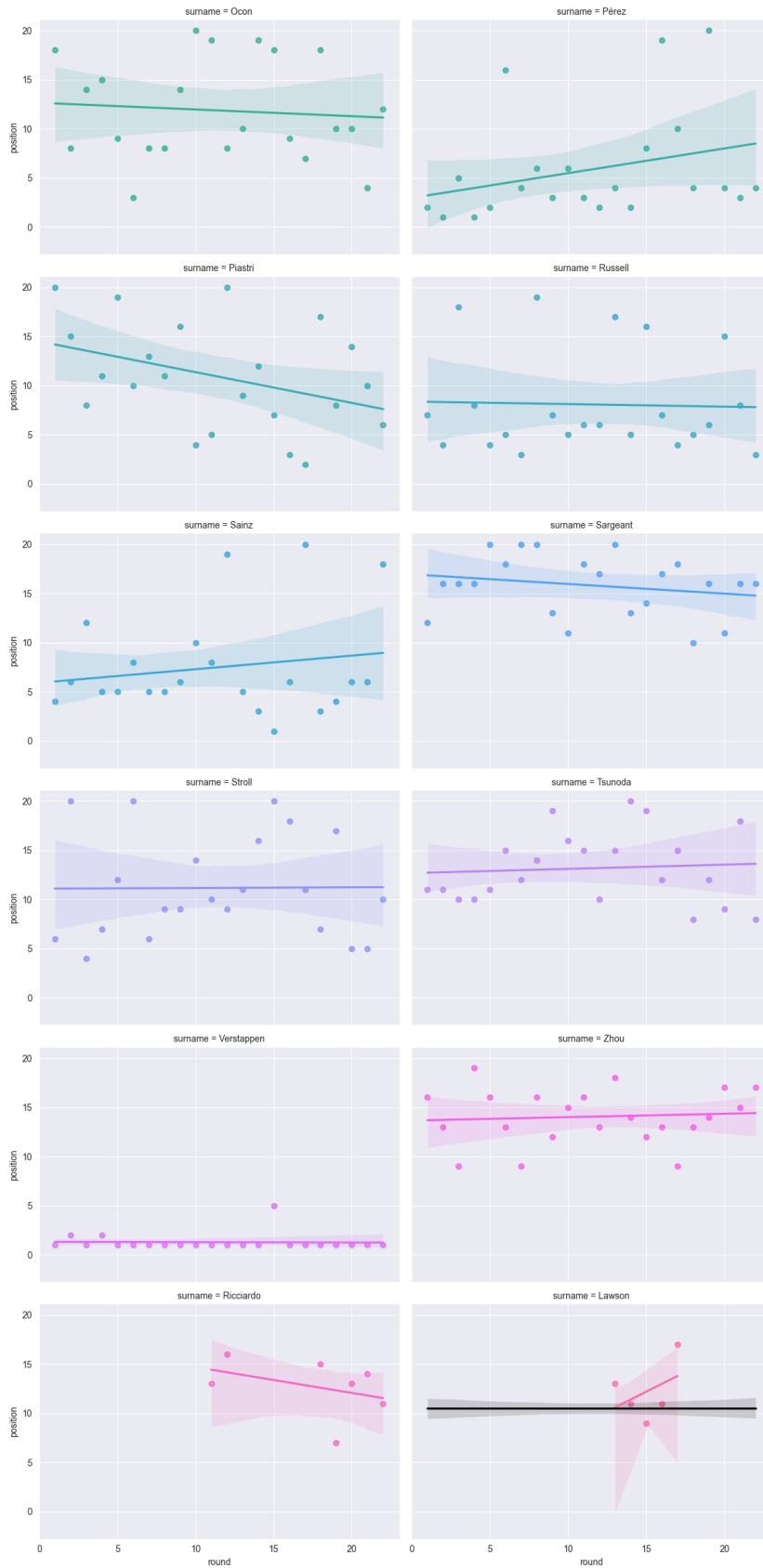
Page 1 of 2

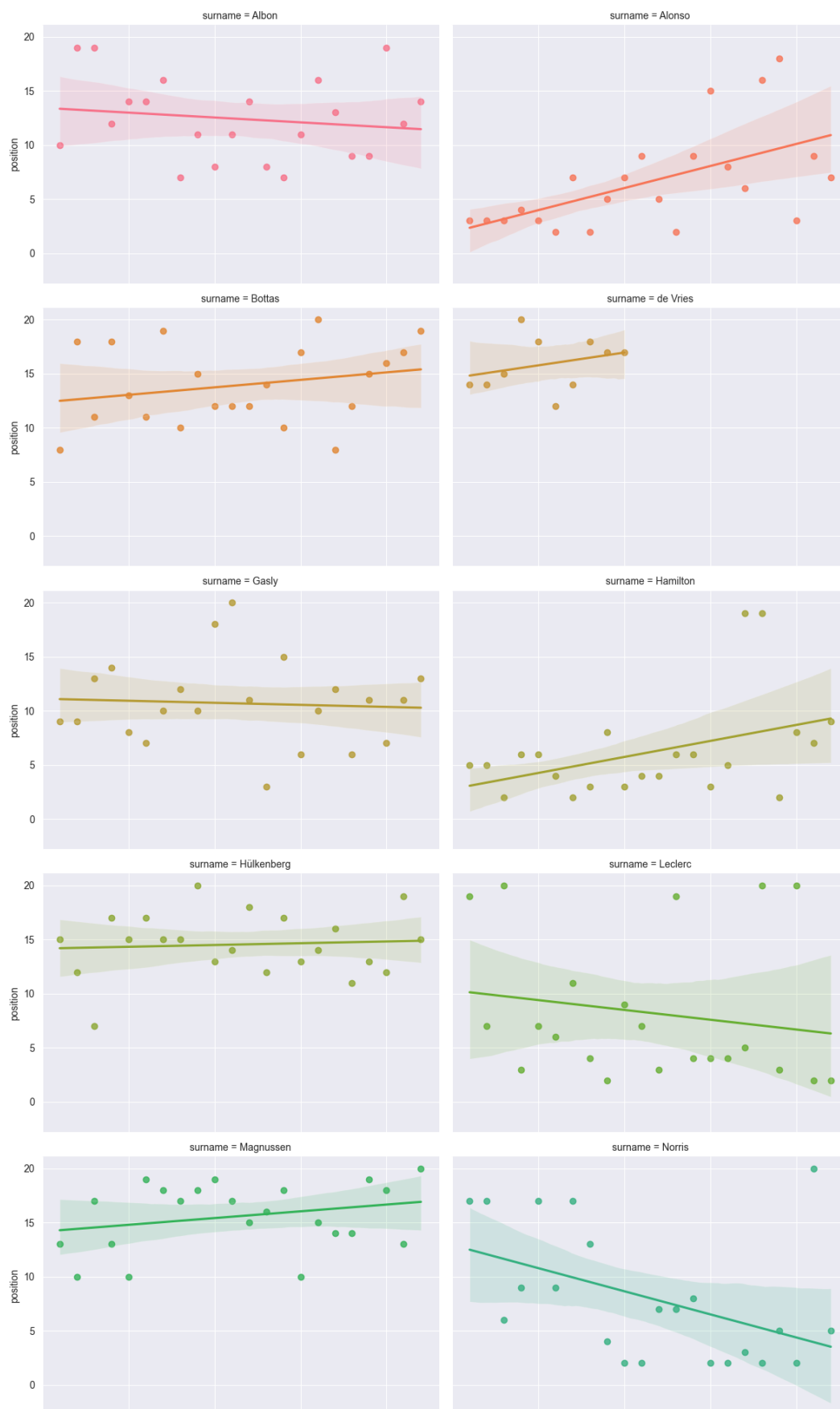
```

557 12      QualifyingRaceCorrelation  0.018397
558 28      AvgPositionsGainedLostLast5Races  0.018338
559 4        CareerWins  0.018074
560 36      MinSpeedLastRace  0.016693
561 19      ConstructorChampionshipsWon  0.016151
562 37      AvgSpeedLastRace  0.014812
563 0        Age  0.013680
564 32      ConstructorSpeedDiffToFastestLastRace  0.013584
565 16      AvgLapTimeConsistencyLastRace  0.013192
566 33      ConstructorRetirements  0.013124
567 8        PolePositions  0.012962
568 15      AvgLapTimeLastRace  0.012307
569 35      MaxSpeedLastRace  0.012255
570 7        RacesFinished  0.011351
571 26      MaxPositionsGainedLast5Races  0.010983
572 1        YearsInF1  0.010707
573 27      MaxPositionsLostLast5Races  0.008768
574 34      ConstructorRetirementsLast5Races  0.008244
575 6        RacesStarted  0.007665
576 9        DriverChampionships  0.007559
577 29      BestFinishLast5Races  0.006957
578 14      PrevYearRacePitStops  0.004641
579 17      SpeedRankLastRace  0.000000
580 38      ConstructorSpeedRankLastRace  0.000000
581 Predicted Last Race Results:
582 PredictedPosition ActualPosition      Driver Probability
583 18      1      1 (Max, Verstappen)  0.713190
584 6      2      2 (Charles, Leclerc)  0.202148
585 10     3      4 (Sergio, Pérez)  0.189668
586 8      4      5 (Lando, Norris)  0.168059
587 13     5      3 (George, Russell)  0.156604
588 14     6      18 (Carlos, Sainz)  0.156541
589 1      7      7 (Fernando, Alonso)  0.152041
590 4      8      9 (Lewis, Hamilton)  0.139719
591 15     9      16 (Logan, Sargeant)  0.131953
592 5      10     15 (Nico, Hülkenberg)  0.123318
593 3      11     13 (Pierre, Gasly)  0.114407
594 2      12     19 (Valtteri, Bottas)  0.114201
595 19     13     17 (Guanyu, Zhou)  0.113653
596 7      14     20 (Kevin, Magnussen)  0.111422
597 9      15     12 (Esteban, Ocon)  0.109996
598 0      16     14 (Alexander, Albon)  0.106070
599 12     17     11 (Daniel, Ricciardo)  0.101450
600 11     18     6 (Oscar, Piastri)  0.100816
601 16     19     10 (Lance, Stroll)  0.096437
602 17     20     8 (Yuki, Tsunoda)  0.082593
603 4.6
604
605 Process finished with exit code 0
606

```

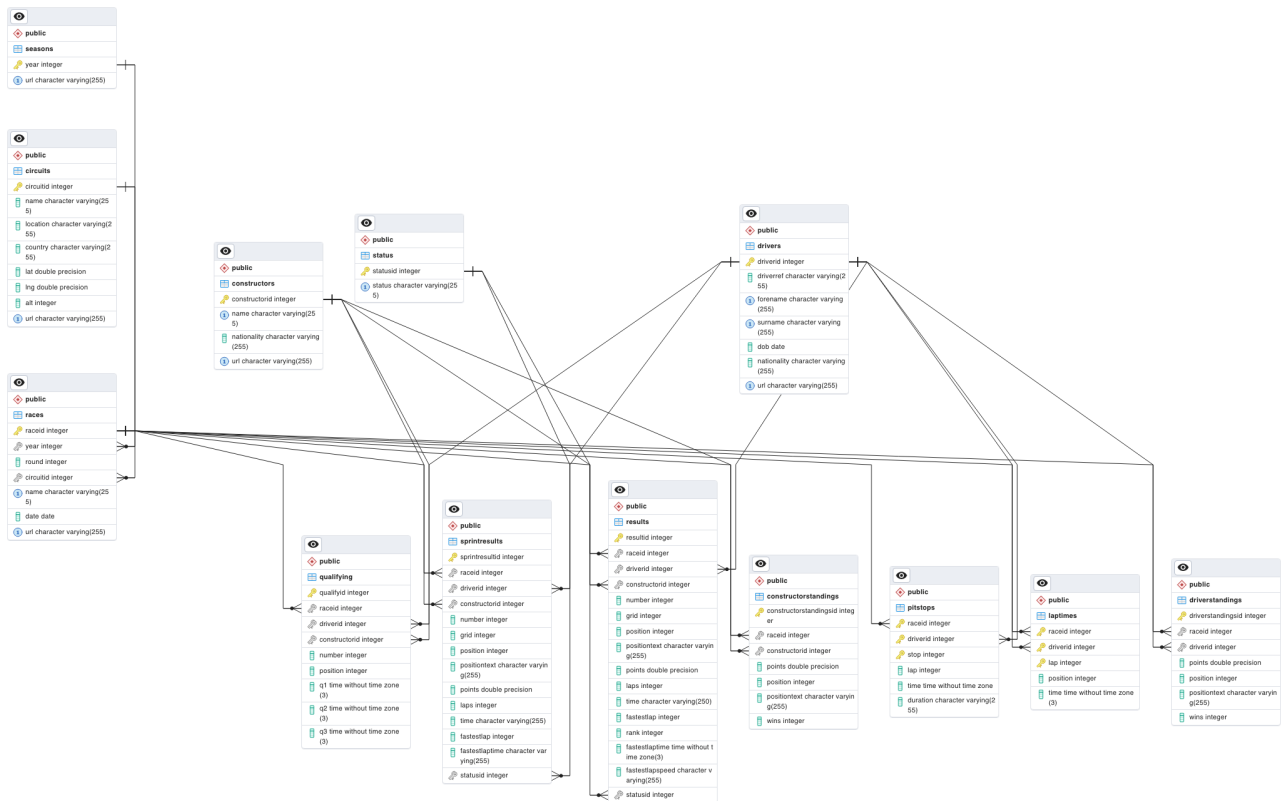
10.4 Datenexploration





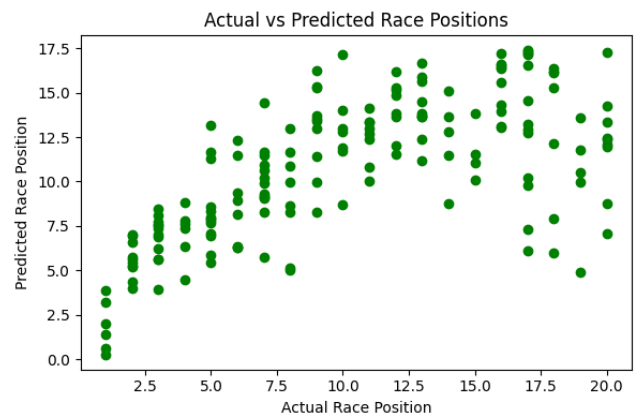
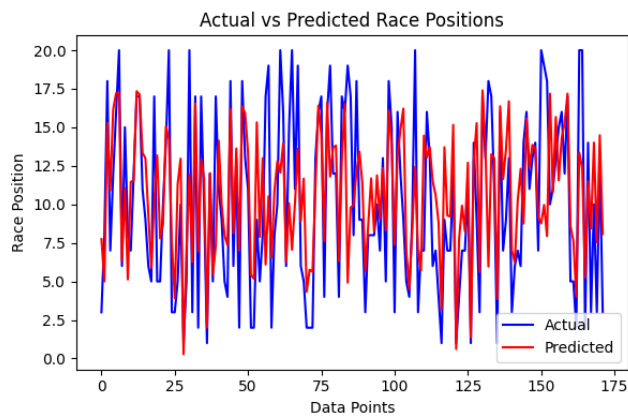
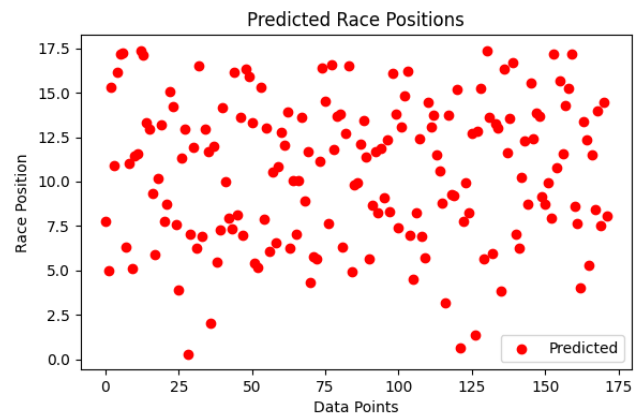
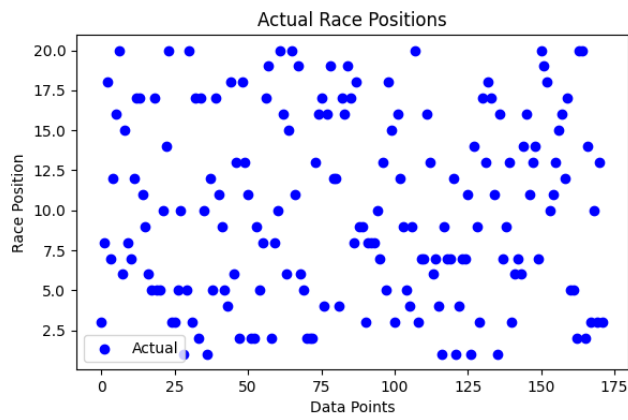
10.5 Diagramme

10.5.1 ERM

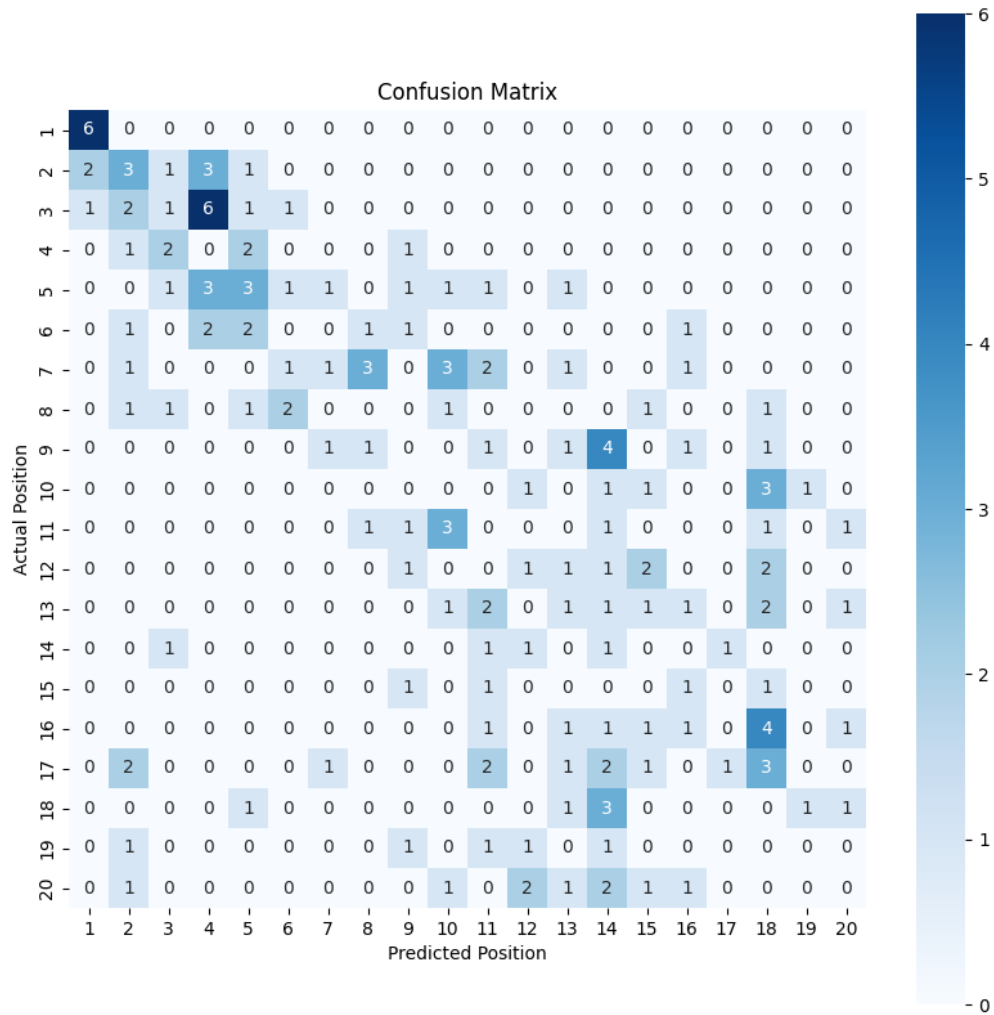


10.5.2 Heatmaps und Veranschaulichungen

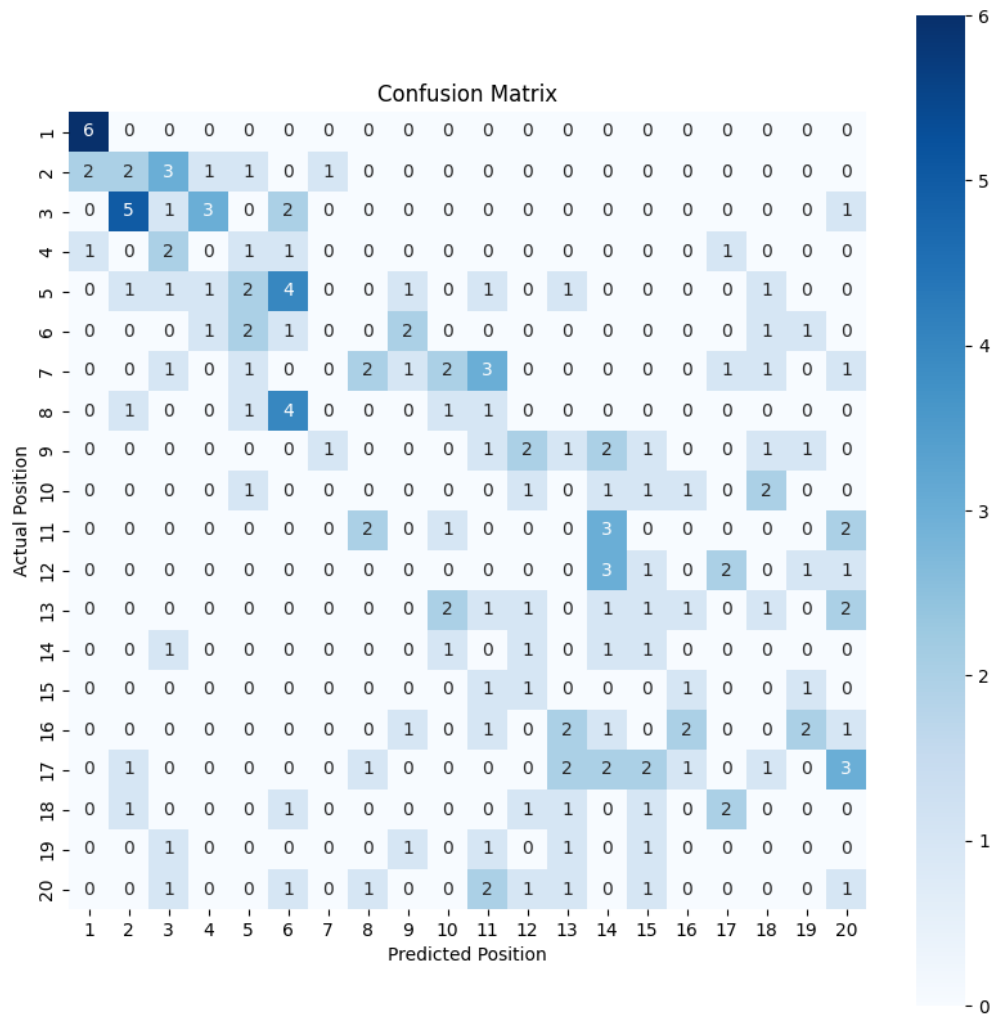
10.5.2.1 Lineare Regression



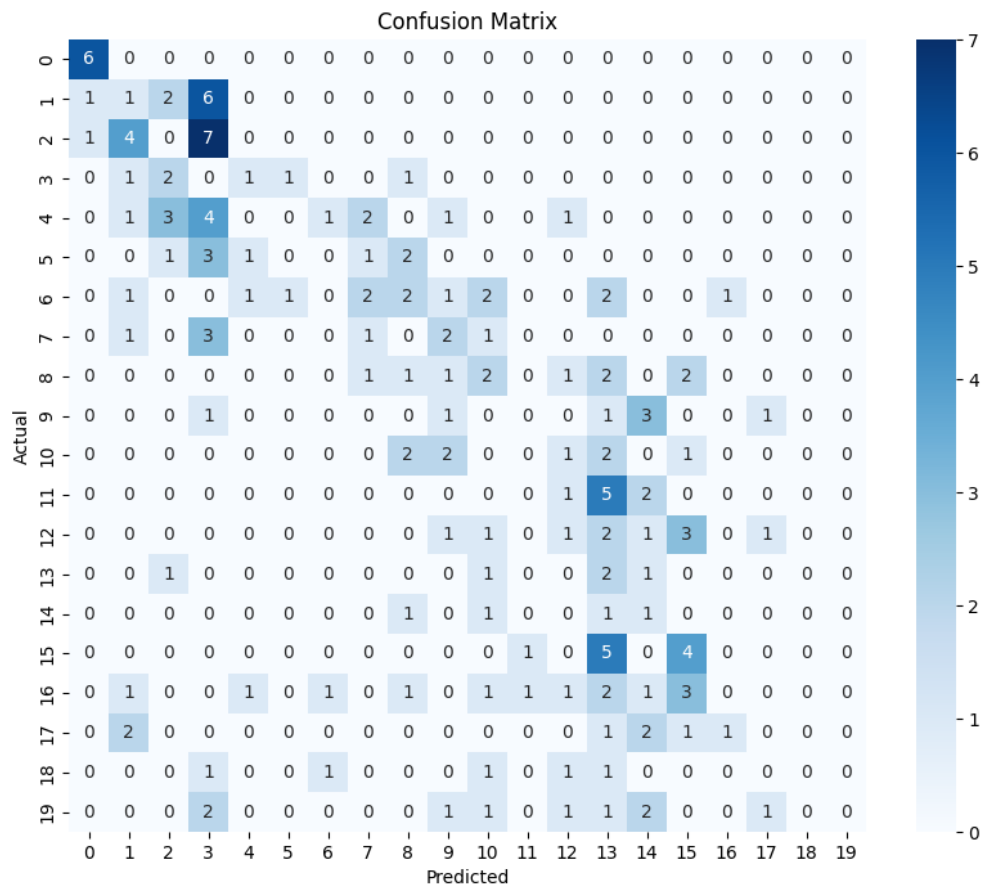
10.5.3.2 Multinomiale logistische Regression: Variante 1



10.5.3.3 Multinomiale logistische Regression: Variante 2



10.5.3.4 Random Forest



10.6 Quellenverzeichnis

(Letzter Zugriff 01.03.2024),

[1] <https://www.f1-predictor.com/abu-dhabi-gp-2023/>

(Letzter Zugriff 01.03.2024),

[2] <http://ergast.com/mrd/>

(Letzter Zugriff 01.03.2024),

[3] <https://open-elevation.com/>

(Letzter Zugriff 01.03.2024),

[4]<https://hastie.su.domains/Papers/ESLII.pdf> (Seite 144)

(Letzter Zugriff 01.03.2024),

[5]<https://hastie.su.domains/Papers/ESLII.pdf> (Seite 577-579)

(Letzter Zugriff 01.03.2024),

[5]<https://hastie.su.domains/Papers/ESLII.pdf> (Seite 577-579)


Letzter Zugriff 01.03.2024),

[6]<https://hastie.su.domains/Papers/ESLII.pdf> (Seite 10-13)

10.7 Eidesstattliche Erklärung

Hiermit versichern ich, Eray Kayur, dass ich die vorliegende Dokumentation mit dem Titel „Formel 1 Predictive Analytics“ selbstständig verfasst, keine anderen als die angegebenen Hilfsmittel verwendet und noch an keiner anderen Stelle vorgelegt habe.

Die Stellen der Dokumentation, die im Wortlaut oder im wesentlichen Inhalt aus anderen Werken entnommen wurden, sind mit genauer Quellenangabe kenntlich gemacht.

 , Aßlar am 12.03.2024

Unterschrift, Ort und Datum