Projektbericht

in dem Modul

Web Mining

zu dem Thema:

Web-Scraping von Daten der ersten Bundesliga zur Vorhersage von Spielergebnissen

Vorgelegt im berufsbegleitenden Studiengang M.Sc. Data Science

von

Alfred Anselm
Matrikelnummer 30258459

Kevin Diec
Matrikelnummer 30245778

Luca Janas
Matrikelnummer 30277119

Prüfer: Prof. Dr. Christian Gawron
Im Sommersemester 2023

Eigenständigkeitserklärung

Ich erkläre hiermit, dass die vorgelegte Arbeit mein eigenes Werk ist. Alle direkt oder indirekt verwendeten Quellen sind als Referenzen angegeben. Die Arbeit wurde bisher nicht vor einem anderen Prüfungsausschuss vorgelegt und nicht veröffentlicht.

Mir ist bekannt, dass die Arbeit in digitaler Form auf die Verwendung unerlaubter Hilfsmittel überprüft werden kann, um festzustellen, ob die Arbeit als Ganzes oder darin enthaltene Teile als Plagiat zu werten sind. Für den Vergleich meiner Arbeit mit vorhandenen Quellen erkläre ich mich damit einverstanden, dass sie in eine Datenbank aufgenommen wird und dort auch nach der Prüfung verbleibt, um einen Vergleich mit künftigen eingereichten Arbeiten zu ermöglichen.

Münster, 30. September 2023.

Alfred Anselm **Kevin Diec** **Luca Janas**

I Inhaltsverzeichnis

II Abbildungsverzeichnis

III Tabellenverzeichnis

1 Projektbeschreibung

In dem hier vorgelegten Projektbericht werden Inhalte und der Aufbau des Projektes im Modul Web Mining im berufsbegleitenden M.Sc. Data Science an der Fachhochschule Südwestfalen beschrieben. Ziel des Projektes ist es, unter Verwendung von Daten, die auf der Webseite https://transfermarkt.de/ zur Verfügung gestellt werden, Spielergebnisse in der ersten deutschen Bundesliga vorherzusagen. Dazu werden historische Daten zu Spielen und Vereinen von transfermarkt.de durch Web-Scraping gesammelt und anschließend aufbereitet und analysiert, um einen Datensatz zu erstellen, der zur Vorhersage der Spielergebnisse verwendet werden kann. Dafür bietet Transfermarkt Informationen zu Fußballspielern, Vereinen, Marktwerten und Statistiken für verschiedene Ligen weltweit.

Das Projekt kann in folgende 5 Einzelteile aufgeteilt werden.

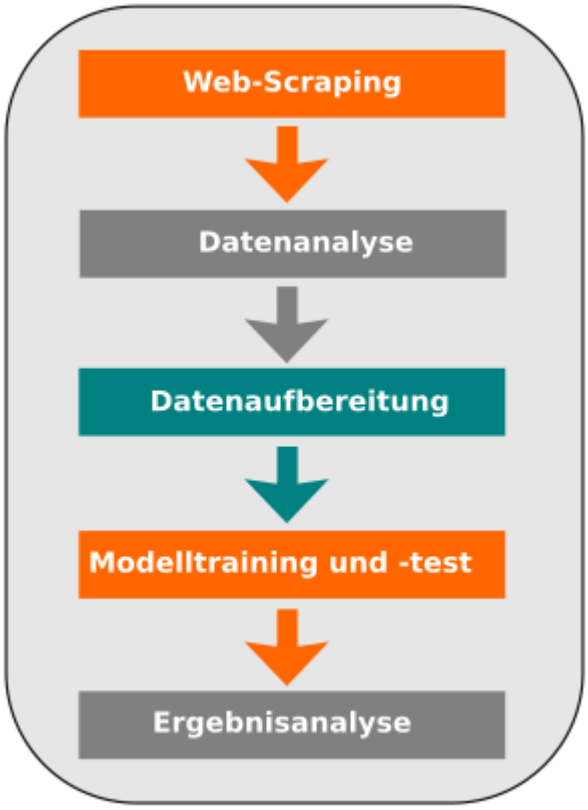

Abbildung 1 - Projektvorgehen

2 Web-Scraping

Für die Selektion der Daten zu den Mannschaften aus der ersten Bundesliga und den jeweiligen Spielergebnissen pro Spieltag und Saison wird das Verfahren des Web-Scrapings auf die Internetseite https://transfermarkt.de/ als Basis-URL angewendet. Für die jeweilige Datenselektion werden weitere URL-Pfade untersucht, die unter anderem Parameter beinhalten, welche die ausgewählte Saison und Spieltage enthalten. Über eine Loop-Funktion können alle Kombinationsmöglichkeiten daraus extrahiert werden.
Die Datenselektionen erfolgten in der Regel mit den Python-Bibliotheken *requests*, *BeautifulSoup* und *lxml.* Mit dem Modul *requests* ist es möglich, eine Anfrage an die vorgegebene URL zu senden und die HTML-Daten der Webseite auszulesen und zu extrahieren.
Um die extrahierten Daten aus dem HTML-Quellcode zu analysieren und zu filtern, wird das Modul *BeautifulSoup* verwendet. Es handelt sich dabei um eine Python-Bibliothek, die beim Web-Scraping eine wichtige Rolle spielt. Sie dient dazu, HTML-Dokumente zu parsen und sie in einer aufbereiteten Struktur darzustellen. In dieser Ausarbeitung wird *BeautifulSoup* in Verbindung mit der „lxml"-Bibliothek als Parser verwendet. Durch die Verwendung von *lxml* kann *BeautifulSoup* von den Funktionen und Vorteilen dieser Bibliothek profitieren, um die Analyse und Manipulation von Webseiteninhalten zu optimieren. Dabei bietet sie eine Reihe von Funktionen und Methoden an, um den Inhalt von Webseiten zu analysieren, die Datenstruktur zu verstehen und spezifische Elemente wie Tabellen, Überschriften, Links oder Absätze zu identifizieren.
Um die Daten weiterzuverarbeiten und zu analysieren, wird in gesonderten Fällen die Methode *etree* aus dem Modul *lxml* verwendet. Diese Methode ermöglicht es, die HTML-Struktur der Webseite genauer zu untersuchen und die Informationen gezielt auszuwählen. Insbesondere wird hierbei auf die Struktur der XPATH-Logik mit *etree* zurückgegriffen, um Schwierigkeiten bei der eindeutigen Identifizierung der Struktur zu überwinden. Das XPATH-Format erlaubt es, bestimmte Elemente in einem HTML-Dokument basierend auf ihrer Position und Hierarchie in einer Tabelle zu identifizieren und über eine Schleife über die Reihen und Spalten gezielt auszuwählen.
Für dieses Projekt wurden die folgenden Daten von Transfermarkt für die erste deutsche Bundesliga gescraped. Der Code dazu steht unter
web-mining/01_Crawler.ipynb at main · lucajanas/web-mining (github.com) zur Verfügung und ist so aufgebaut, dass auch die Daten anderer Ligen, wie der englischen Premier League, ohne Weiteres abgerufen und verarbeitet werden können.

Vereinsüberblick pro Saison

Transfermarkt bietet eine umfassende Übersicht über sämtliche Mannschaften der ersten Bundesliga pro Saison an. Hierbei ist zu beachten, dass sich die Zusammensetzung der Mannschaften in der Liga von Saison zu Saison ändert, da Teams auf- und absteigen können. Mithilfe der folgenden URL kann der Parameter für die jeweilige Saison übergeben werden:
https://www.transfermarkt.de/bundesliga/startseite/wettbewerb/L1/plus/?saison_id=<VALUE>.

In diesem Kontext kann <VALUE> durch die gewünschte Saison ersetzt werden. Beispielsweise kann für die Saison 2022/2023 der Wert 2022 für <VALUE> eingesetzt werden. Auf diese Weise kann die Seite, welche die Informationen zur Bundesliga-Saison 2022/2023 auf Transfermarkt präsentiert, aufgerufen werden. Diese Herangehensweise ermöglicht es, beim Web-Scraping systematisch mittels einer For-Schleife alle verfügbaren Saisons zu durchlaufen. Die Informationen, die daraus selektiert werden können, sind in einem Dataframe zusammengefasst und beinhalten die folgenden Spalteninformationen.

Tabellenspalte	**Erklärung**
CLUB_NAME	Vereinsname aus der ersten Bundesliga
PLAYERS_COUNT	Entspricht dem Kader (Die Anzahl der Spieler pro Verein innerhalb der Saison)
PLAYERS_AVG_AGE	Durchschnittliches Alter des Kaders
LEGIONARIES_COUNT	Anzahl der Legionäre (Spieler außerhalb seines Heimatlandes)
AVG_MARKET_VALUE	Durchschnittlicher Marktwert des Vereins zum Zeitpunkt der Saison
TOTAL_MARKET_VALUE	Gesamtmarktwert des Vereins zum Zeitpunkt der Saison
season	Das Jahr, in dem die Saison begann

Tabelle 1: Vereinsübersicht pro Saison

Normalerweise sind in jeder Bundesliga-Saison 18 Vereine vertreten. In den Jahren 1963 und 1964 gab es jedoch lediglich 16 Vereine, während es im Jahr 1991 sogar 20 Vereine pro Saison gab. Um diese besonderen Fälle in unserem Skript angemessen zu berücksichtigen, wurde eine IF-Bedingung eingeführt, die es ermöglicht, diese Ausnahmen korrekt abzubilden.
Es ist erwähnenswert, dass Marktwerte von Spielern erst seit der Saison 2002 erfasst und aufgelistet werden.

Transferdaten pro Saison

Über die folgende URL-Adresse
https://www.transfermarkt.de/bundesliga/transfers/wettbewerb/L1/plus/?saison_id=<SAISON>&s_w=&leihe=1&intern=0&intern=1

kann eine umfassende Untersuchung der Zu- und Abgänge innerhalb der Vereine innerhalb einer Saison durchgeführt werden. Hierbei könnten die entsprechenden Transfers auf Spielerbasis erfasst werden. Da diese detaillierte Erfassung auf Spielerbasis für unsere Analyse irrelevant und zu granular ist, wird sich auf aggregierte Informationen auf Vereinsebene konzentriert. In diesem Zusammenhang sind die nachfolgenden Informationen extrahiert worden.

Tabellenspalte	**Erklärung**
CLUB_NAME	Vereinsname aus der ersten Bundesliga
AVG_AGE_JOINING	Durchschnittsalter aller Zugänge in der Saison
AVG_AGE_LEAVING	Durchschnittsalter aller Abgänge in der Saison
TOTAL_VALUE_JOINING_MIO	Gesamtmarktwert der Zugänge in Mio.
TOTAL_VALUE_LEAVING_MIO	Gesamtmarktwert der Abgänge in Mio.
EXPENSES_JOINING_MIO	Ausgaben durch Zugänge in Mio.
REVENUE_LEAVING_MIO	Einnahmen durch Abgänge in Mio.
season	Das Jahr, in dem die Saison begann

Tabelle 2: Transferdaten pro Saison

Es ist zu beachten, dass die Angaben bezüglich der Marktwerte und finanziellen Transaktionen hier nicht in einer einheitlichen Währungseinheit („Mio. €“) präsentiert werden, sondern auch in Tausender Schritten („Tsd. €“) vorkommen können. Daher war eine Transformation notwendig, um sämtliche Angaben in einer einheitlichen Struktur darzustellen. Dies gewährleistet, dass die Daten für unsere Analysezwecke bereinigt und konsistent sind.
Es ist anzumerken, dass die Angaben zum Marktwert erst ab dem Jahr 2004 verfügbar sind.

Spielinformationen zu allen Spieltagen pro Saison
Transfermarkt stellt die Ergebnisse der Spiele der ersten Bundesliga für pro Spieltag zur Verfügung, so z. B. unter Bundesliga - Spieltagsübersicht - 2. Spieltag 23/24 | Transfermarkt. Neben den Mannschaften, die gegeneinander spielen, und dem Spielergebnis stehen das Spieldatum, der Schiedsrichter der Partie sowie die Tipprunden-Tendenzen zur Verfügung. Die Tipprunden-Tendenzen stellen die Ergebnisse von Umfragen in der Transfermarkt-Community dar. Es stehen drei Prozentwerte zur Verfügung, die angeben, welche Wahrscheinlichkeit die Transfermarkt den folgenden Ergebnissen zuordnet: Heimsieg, Unentschieden oder Auswärtssieg.
Für das Scraping und Parsing der Informationen werden requests und BeautifulSoup verwendet. Requests wird dabei für das Ausführen des http-Requests verwendet, um den Inhalt der Transfermarkt-Seite abzufragen und BeautifulSoup für das Parsen des

Inhalts. Gezeigt wird dies im Folgenden am Beispiel der Heimmannschaften eines Spieltags, die ermittelt werden.
Nach Ausführung des http-Requests wird ein BeautifulSoup-Objekt erstellt. Nach Sichtung des HTML-Inhalts der abgerufenen Transfermarkt-Website steht fest, dass die benötigten Heimmannschaften in einer Tabelle mit dem HTML-Tag *td* und dem CSS-Klassennamen *„rechts hauptlink no-border-rechts hide-for-small spieltagsansicht-vereinsname“* stehen. Diese Tabellenelemente können über das erstellte BeautifulSoup-Objekt gefiltert werden. In diesem Fall müssen die tatsächlichen Vereinsnamen anschließend per regulärem Ausdruck aus den vorgefilterten Tabelleneinträgen ermittelt werden.
Auf diese Art und Weise werden die weiteren genannten Spieltagsdaten ebenfalls ermittelt. Letztendlich entsteht folgende Tabellenstruktur:

Tabellenspalte	**Erklärung**
{HOME, AWAY}_TEAM	Name der Heim- und Auswärtsmannschaftmannschaft
{HOME, AWAY}_GOALS	Anzahl der Tore der Heim- und Auswärtsmannschaft
PLACE_{HOME, AWAY}_TEAM	Platzierung der Heim- und Auswärtsmannschaft vor Beginn des Spieltags
REFEREE	Schiedsrichter der Partie
DATE	Datum, an dem das Spiel stattfand
WEEKDAY	Wochentag des Spiels
MONTH	Monat, in dem das Spiel stattfand
SEASON	Saison, in der das Spiel stattfand
MATCHDAY	Spieltag
WIN_PERC_{HOME, AWAY}	Tipprundentendenzen der Transfermarkt-Community zu „Sieg Heim- bzw. Auswärtsmannschaft“
REMIS_PERC	Tipprundentendenzen der Transfermarkt-Community zum Ergebnis „Unentschieden“

Tabelle 3 - Spieltagsdaten

Die Notation {HOME, AWAY} bedeutet, dass es das Attribut sowohl für die Heim- als auch für die Auswärtsmannschaft gibt. Die geschriebene Funktion zum Scrapen der Spieltagsdaten wird für je einen Spieltag einer Saison aufgerufen. Dies passiert in einer Schleife für alle benötigten Spieltage.

Finale Bundesliga-Tabelle pro Saison

Um den Zusammenhang zwischen Attributen wie dem Marktwert eines Vereins und seiner finalen Platzierung in der Bundesliga-Tabelle untersuchen zu können, werden die finale Bundesliga-Tabellen jeder Saison von Transfermarkt abgerufen. Eine Beispiel-Tabelle findet sich auf der folgenden Seite: Bundesliga - Tabelle | Transfermarkt Es werden die Tabellenelemente aus der CSS-Klasse *"no-border-links hauptlink"* benötigt. Diese werden über das erstellte BeautifulSoup ermittelt. Dies resultiert in folgendem Tabellenaufbau:

Tabellenspalte	**Erklärung**
SEASON	Saison
CLUB_NAME	Vereinsname
PLACE	Finale Platzierung am Ende der Saison

Tabelle 4 - Attribute der finalen Bundesligatabelle

Die erstellten Datensätze stellen die Grundlage für die nachfolgenden Schritte dar.

Wetterdaten

Um weitere Einflussfaktoren für die Analyse der Spielergebnisse zu berücksichtigen, wurden zusätzliche Wetterdaten herangezogen, welche durch Web-Scraping von der Internetseite https://www.wetterkontor.de ermittelt werden konnten. Auf dieser Plattform sind tägliche Wetterdaten ab dem 01.01.2011 verfügbar. Über die nachfolgende URL können tageweise alle Wetterinformation herausselektiert werden, wenn das Datum in der URL als Parameter im Format *yyyymmdd* eingepflegt wird:

`https://www.wetterkontor.de/de/wetter/deutschland/extremwerte.asp?id=<yyyymmdd>`

Dadurch stehen folgende Wetterdaten zur Verfügung:

Tabellenspalte	**Erklärung**
WEATHER_STATION	Standort der Wetterstation
MIN_TEMP_C	Minimale Temperatur in °C des Tages
MAX_TEMP_C	Maximale Temperatur in °C des Tages
SNOW_HEIGHT_cm	Schneehöhe in cm
RAIN_l/m2	Niederschlagsmenge l/m²
SUNSHINE_DURATION_h	Sonnenscheindauer in Stunden pro Tag
HOME_TEAM	Zu der Wetterstation naheliegender Standort der Heimmannschaft
MATCHDAY	Spieltag

Tabelle 5: Wetterdaten pro Tag pro Wetterstation

Es ist jedoch anzumerken, dass es mitunter schwierig ist, die genauen Standorte der Fußballvereine den verfügbaren Wetterstationen eindeutig zuzuordnen. Dies ergibt sich aus der Tatsache, dass nicht in allen Städten Wetterstationen vorhanden sind und einige Städte über mehrere Wetterstationen verfügen. Die Zuordnung erfolgte daher manuell unter Berücksichtigung des bestmöglichen Wissens und Gewissens. Darüber hinaus unterliegen die Wetterstationen im Laufe der Zeit Veränderungen, da neue Stationen hinzukommen und einige wieder entfernt worden sind.
Dies führt dazu, dass eine erhebliche Anzahl von Datensätzen an den verschiedenen Spieltagen verloren gehen, wie dies in der unten dargestellten Heatmap deutlich wird. Die Flächen, die in roter Farbe markiert sind, repräsentieren die fehlenden Werte. Es sollte beachtet werden, dass die Heatmap so strukturiert ist, dass jede Zeile einem Spieltag entspricht. Obwohl es herausfordernd sein kann, die Daten zeilenweise zu interpretieren, erweist sich die Heatmap als äußerst hilfreich, um einen umfassenden Überblick über die fehlenden Informationen zu erhalten.

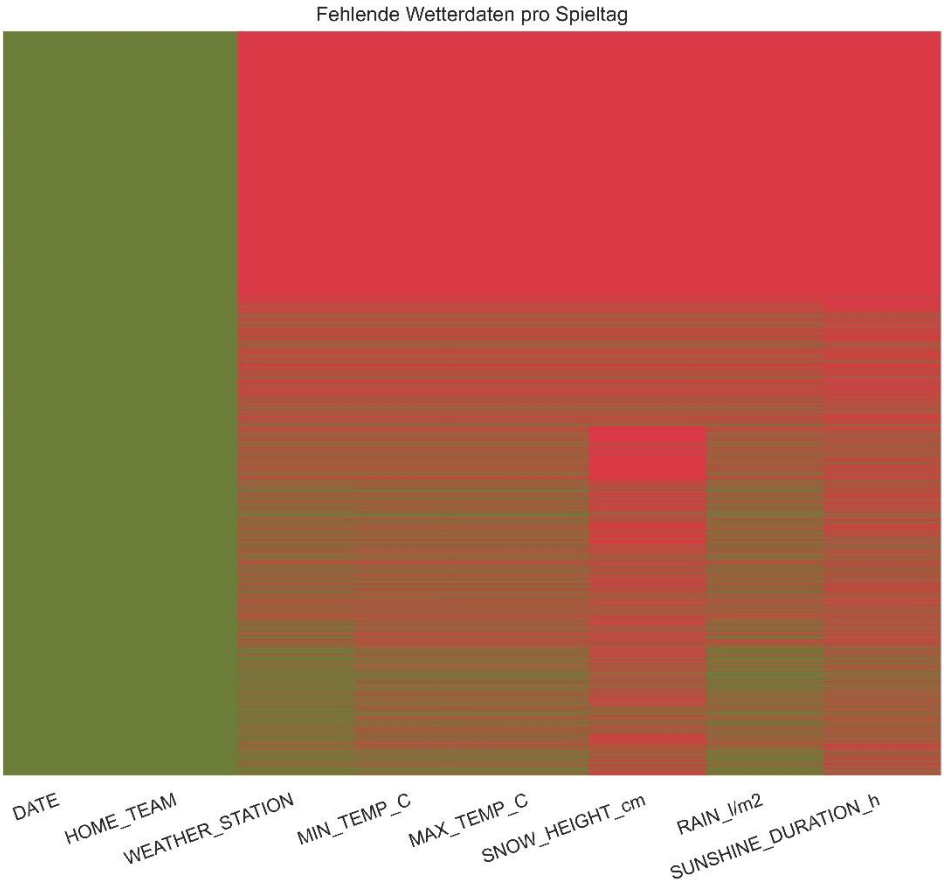

Abbildung 2: Heatmap über fehlende Wetterdaten pro Spieltag

In Zahlen ausgedrückt bedeutet dies, dass wir von 5814 Spieltagen nur 2251 Wetterstation zuordnen konnten, welche wiederum nur 1980 verwertbare Temperaturdaten, 1198 Schneedaten, 2170 Regendaten und 1425 Sonnenscheindaten aufweist.
Zusammengefasst ist dies aus der Python-Funktion df.info() zu entnehmen:

```
#   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   DATE                 5814 non-null   datetime64[ns]
 1   HOME_TEAM            5814 non-null   object
 2   WEATHER_STATION      2251 non-null   object
 3   MIN_TEMP_C           1980 non-null   object
 4   MAX_TEMP_C           1980 non-null   object
 5   SNOW_HEIGHT_cm       1193 non-null   object
 6   RAIN_l/m2            2170 non-null   object
 7   SUNSHINE_DURATION_h  1425 non-null   object
```

3 Datenaufbereitung und Datenanalyse

Das Ziel besteht darin, basierend auf einer Datenzeile das Ergebnis eines Spiels vorherzusagen. Die erstellten Datensätze weisen eine unterschiedliche Granularität auf. Sowohl die allgemeinen Vereinsdaten als auch die Transferdaten und die finale Bundesliga-Platzierung stehen pro Verein pro Saison zur Verfügung. Im Gegensatz dazu stehen die Spieltagsdaten pro Spieltag pro Saison zur Verfügung. Dieses Format stellt das Zielformat dar, da der Ausgang dieser Spiele geschätzt werden soll. Die Daten, die ausschließlich pro Saison zur Verfügung stehen, müssen demnach, um für die Schätzung des Spielausgangs verwendet werden zu können, auf die Spieltage der jeweils dazugehörigen Saison verteilt werden. Der Code dazu findet sich in web-mining/02_Merging.ipynb at main · lucajanas/web-mining (github.com)
Als erstes müssen die Vereinsnamen in dem Spieltagsdatensatz angepasst werden, da dort Kurzversionen der verschiedenen Vereinsnamen verwendet werden. Diese Kurzversionen werden auf die ausgeschriebenen Varianten gemapped, um im nächsten Schritt über einen Left-Join mit pandas jedem Spieltag die vereins- und transferbezogenen Informationen für Heim- und Auswärtsteam für die jeweilige Saison zuzuordnen. Der Join wird daher auf den Attributen Saison und Vereinsname durchgeführt.
Der erstellte Datensatz wird im Folgenden zur Visualisierung und Analyse der Spieltagsdaten verwendet. Der Code zur Erstellung der Abbildungen sowie die Abbildungen selbst sind unter web-mining/04_DataAnalysis.ipynb at main · lucajanas/web-mining (github.com) abgelegt.
Abbildung 2 zeigt die Bundesliga-Platzierung sowie den Marktwert einer Mannschaft dargestellt in einem Scatterplot.

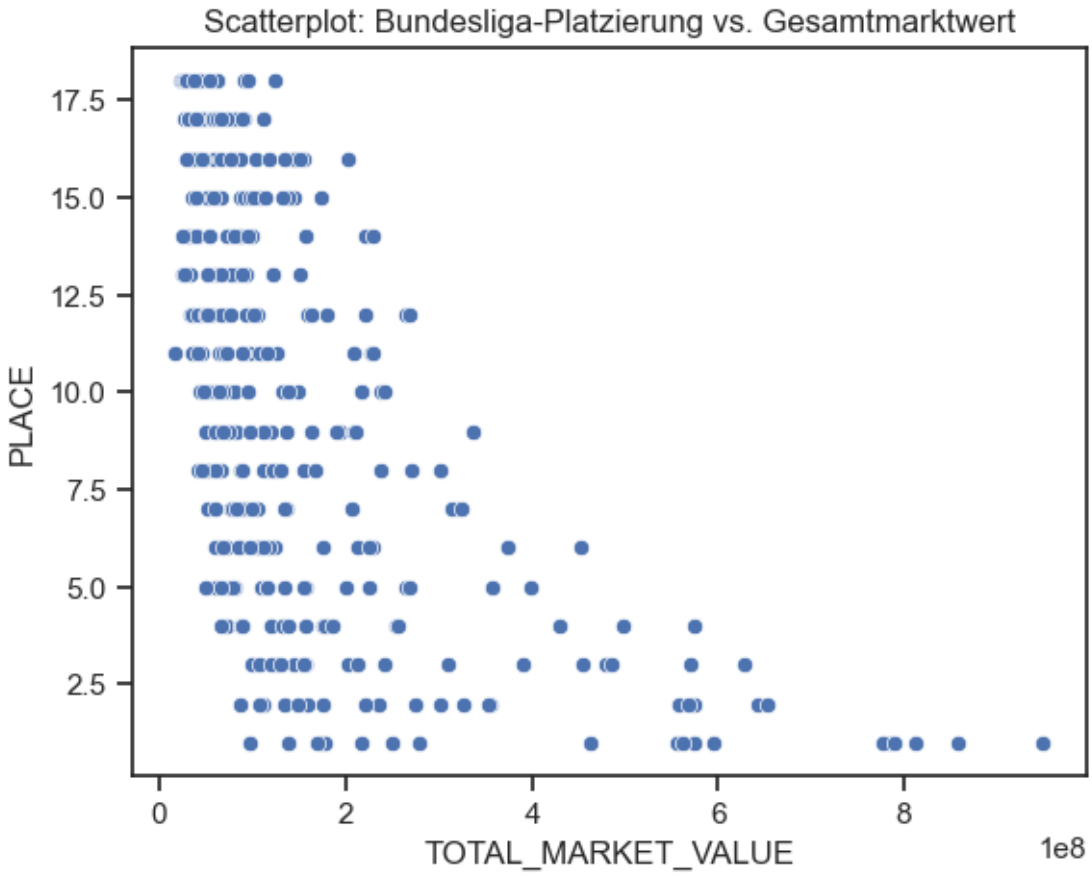

Abbildung 3 - Scatterplot: Bundesliga-Platzierung vs. Gesamtmarktwert

Abbildung 3 zeigt, wie zu vermuten war, dass Mannschaften mit einem höheren Marktwert tendenziell eine bessere Platzierung am Saison-Ende erreichen. Deswegen wird der Marktwert einer Mannschaft beim Training eines Klassifikationsmodells zur Vorhersage des Spielausgangs vermutlich ein wichtiges Feature darstellen.
Abbildung 4 zeigt die durchschnittlichen Einnahmen und Ausgaben in der Bundesliga pro Saison.

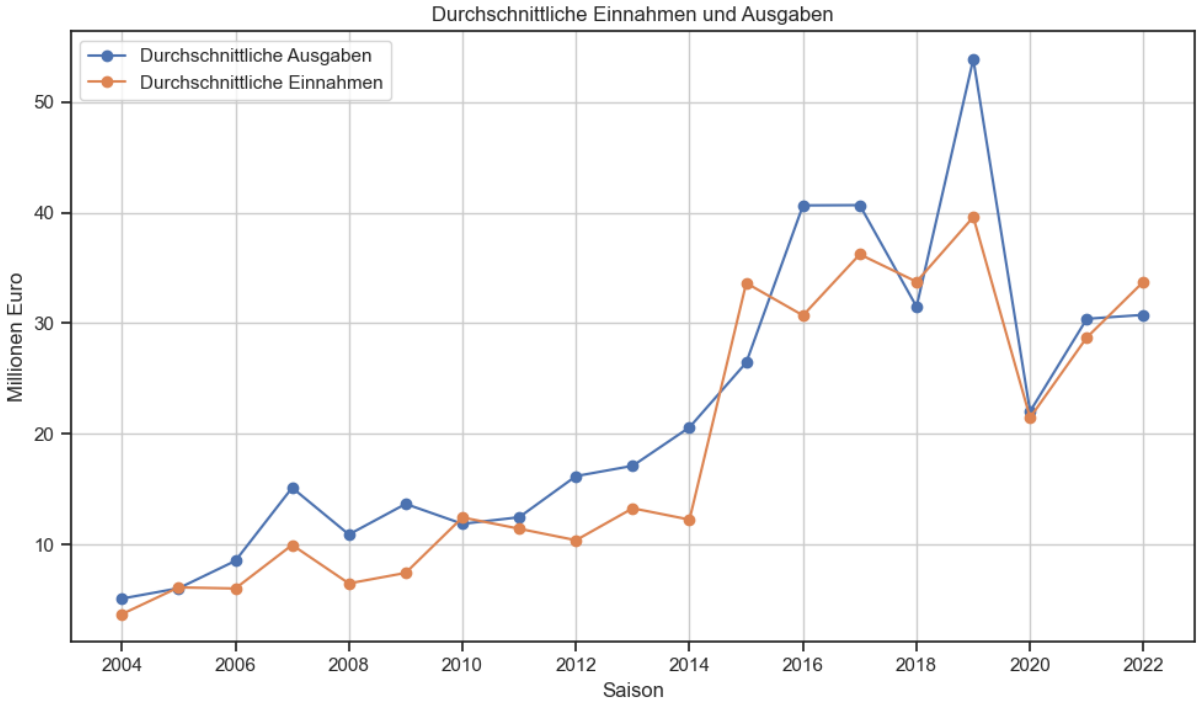

Abbildung 4 - Durchschnittliche Einnahmen und Ausgaben pro Saison

Abbildung 4 zeigt, dass die Ausgaben der Bundesliga für Spielereinkäufe üblicherweise über den Einnahmen durch Spielerverkäufe liegen, mit einigen Ausnahmen, wo die Einnahmen die Ausgaben leicht überstiegen, so z. B. im Jahr 2022. Da sich die Einnahmen und Ausgaben pro Verein zum Teil deutlich unterscheiden, werden diese Angaben ebenfalls als Input für das Klassifikationsmodell verwendet. Insgesamt steigen sowohl die Einnahmen als auch die Ausgaben von einem mittleren einstelligen Millionenbetrag im Jahr 2004 auf mittlere zweistellige Millionenbeträge bis hin zu Ausgaben von über 50 Millionen Euro im Jahr 2019. 2020 wurden deutlich niedrigere Einnahmen und Ausgaben verzeichnet, was auf die durch die Corona-Pandemie zu dem Zeitpunkt unsichere wirtschaftliche Situation vieler Vereine sowie des Gesamtprofifußballs zurückzuführen ist.
Abbildung 5 zeigt die Anzahl der verschiedenen Spielausgänge – Heimsieg, Auswärtssieg oder Unentschieden, pro Saison.

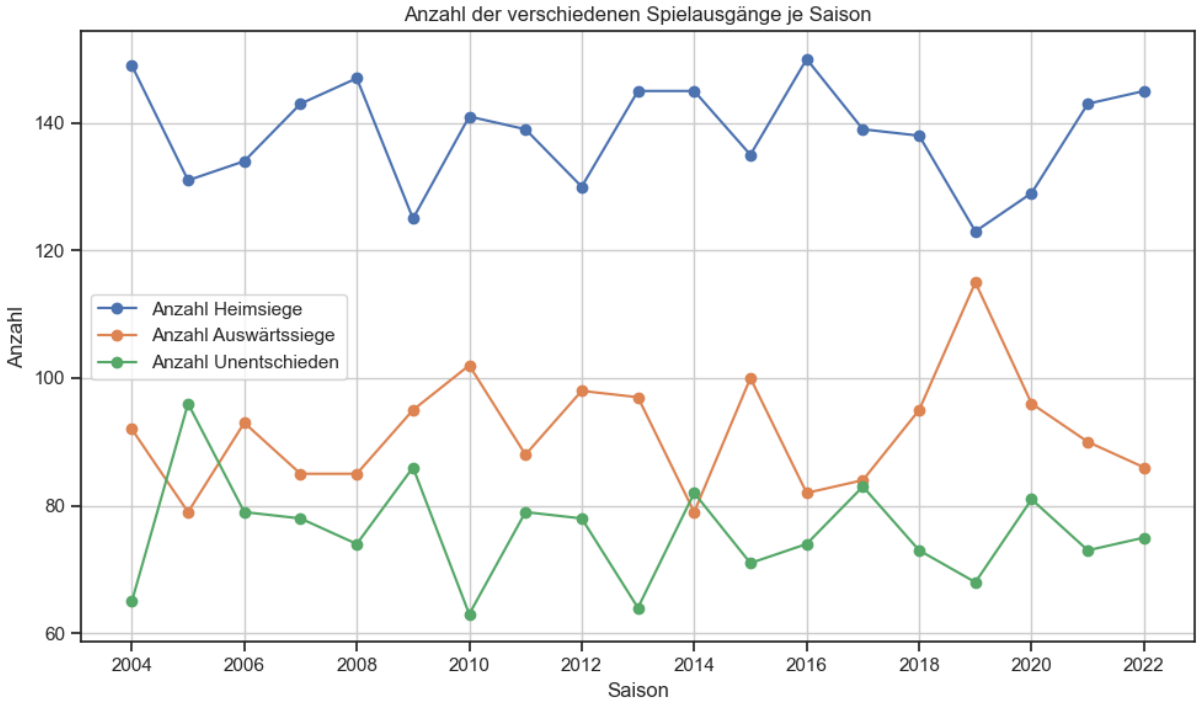

Abbildung 5 - Anzahl der verschiedenen Spielausgänge je Saison

Abbildung 5 lässt vermuten, dass der oft vermutete Heimvorteil tatsächlich zu existieren scheint. Die Anzahl der Heimsiege ist stets größer als die Anzahl der Auswärtssiege. Einzig im Jahr 2019 sind die Anzahl der Heimsiege sowie die Anzahl der Auswärtssiege beide rund um 120, da in dieser Saison unüblich viele Auswärtssiege und unüblich wenige Heimsiege auftraten. Üblicherweise treten zwischen 120 und 150 Heimsiege auf und zwischen 80 und 100 Auswärtssiege. Die Anzahl der Unentschieden schwankt zwischen 60 und 100, wobei in einem Großteil der Saisons zwischen 60 und 80 Unentschieden auftreten. Abbildung 4 lässt vermuten, dass die Unterscheidung zwischen Heim- und Auswärtsteam einen Einfluss auf die Genauigkeit des Klassifikationsmodells haben wird. Deswegen wird diese Unterscheidung in den Features vorgenommen.

Abbildung 6 zeigt die Confusion Matrix, die entsteht, wenn die Tipprundentendenzen der Transfermarkt-Community verwendet werden, um die Spielergebnisse vorherzusagen.

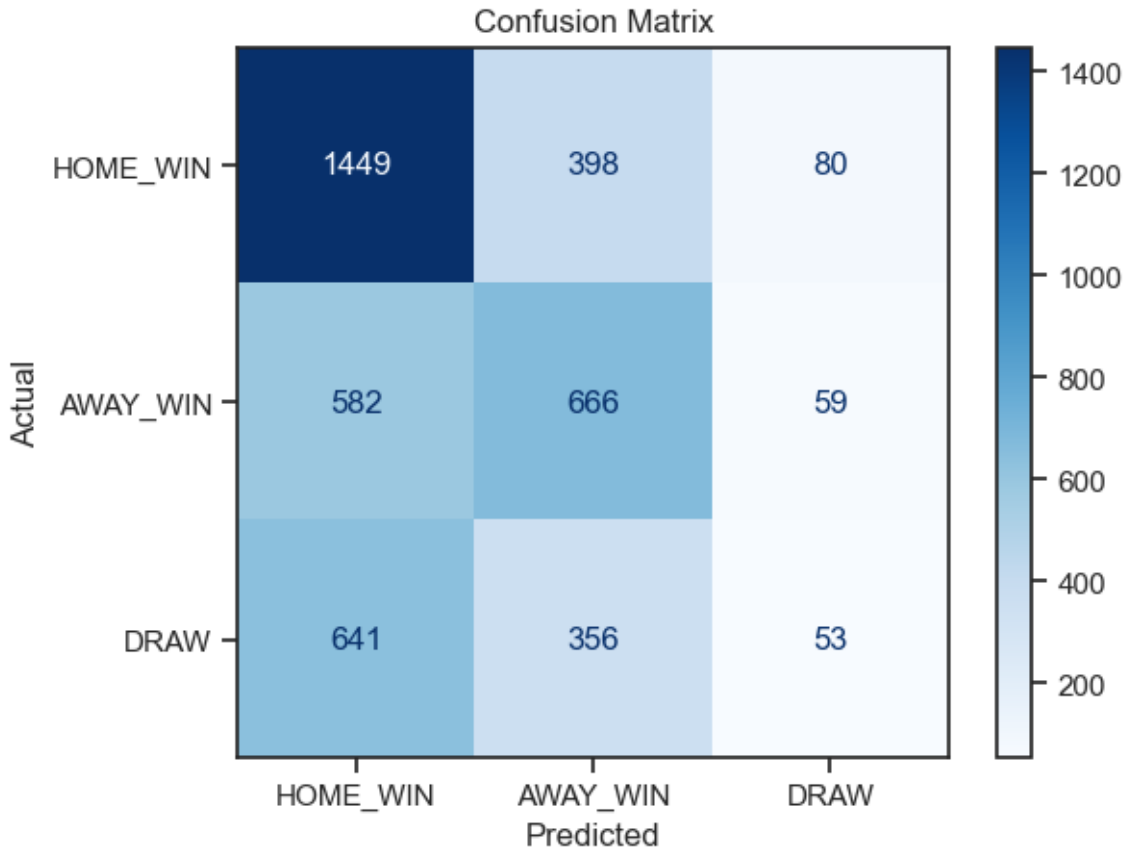

Abbildung 6 - Confusion Matrix basierend auf Tipprundentendenz

Abbildung 6 zeigt, dass Tipprundentendenzen allein keine verlässlichen Indikatoren zur Vorhersage des Spielergebnisses sind. Es werden 50,6% der Spielergebnisse richtig getippt. Während ca. 75% der Heimsiege richtig erkannt werden, werden lediglich ca. 51% der Auswärtssiege und 5% der Unentschieden richtig getippt. Abbildung 6 zeigt die Confusion Matrix, wenn ausschließlich Spiele berücksichtigt werden, bei denen die Transfermarkt-Community mit mindestens 90% auf den jeweiligen Spielausgang getippt hat. Das trifft auf 23% der ursprünglich im Datensatz enthaltenen Spiele zu. In diesen 23% sind keine Unentschieden enthalten, d. h. die Transfermarkt-Community setzt nie zu 90% oder mehr auf ein Unentschieden als Spielausgang. Das Filtern erhöht die Gesamtgenauigkeit auf circa 69%. Es werden 94,04% der Heimsiege erkannt, 64,54% der Auswärtssiege und kein einziges Unentschieden.

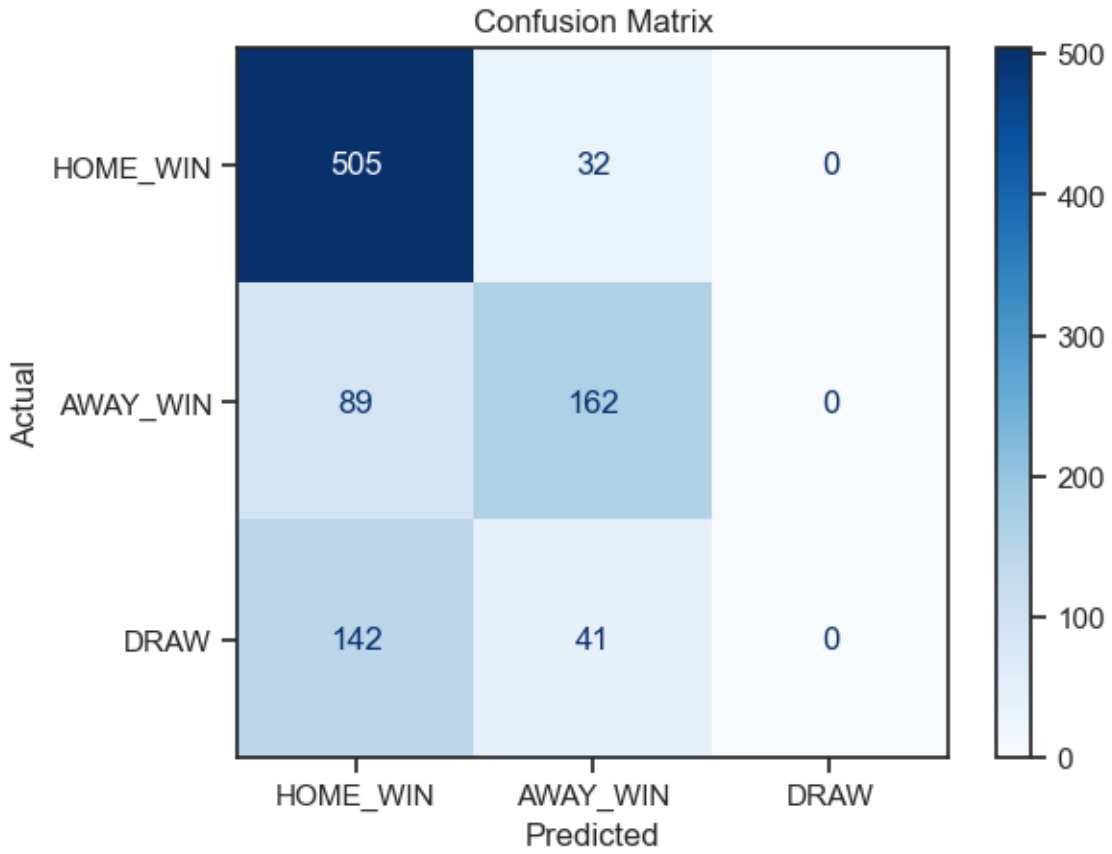

Abbildung 7 - Confusion Matrix basierend auf Tipprundentendenz mit 90%-Quote

Um zu untersuchen, welchen Einfluss der Schiedsrichter der Partie auf den Spielausgang für die einzelnen Vereine hat, werden die Siegquoten der Vereine für die einzelnen Schiedsrichter ermittelt. Abbildung 8 zeigt die Siegquoten pro Schiedsrichter für den FC Bayern München. Dabei wurden ausschließlich Schiedsrichter berücksichtigt, die mehr als 15 Spiele des FC Bayern München geleitet haben. Die durchschnittliche Siegesquote des FC Bayern München liegt bei ca. 85%. Auffällig ist, dass kein einziges Spiel, das von Bastian Dankert geleitet wurde, verloren wurde. Unter Sascha Stegemann konnten nur 65% der Spiele gewonnen werden. Auch wenn die Stichproben sicherlich nicht groß genug sind, um einen eindeutigen Zusammenhang zwischen Schiedsrichter und Spielausgang herzustellen und weitere Faktoren weitaus mehr Einfluss haben werden, wird der Schiedsrichter der Partie im Klassifikationsmodell berücksichtigt.

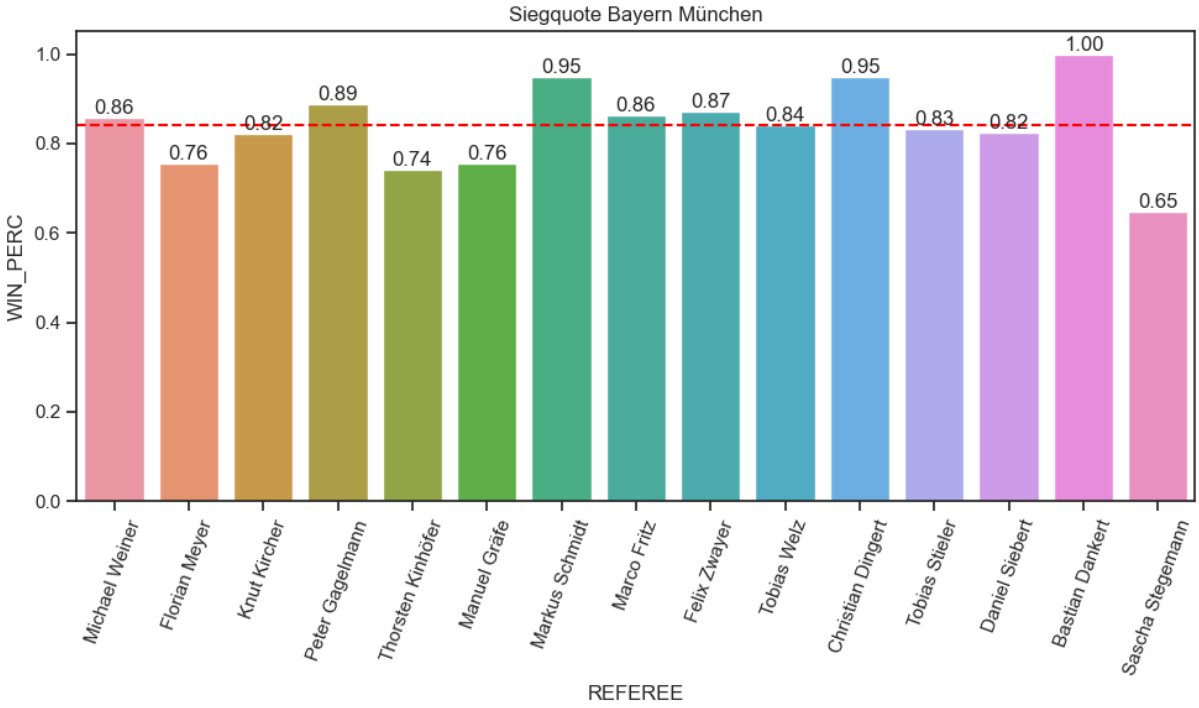

Abbildung 8 - Siegesquote des FC Bayern München pro Schiedsrichter

Abbildung 9 zeigt die Siegesquote des 1. FC Köln pro Schiedsrichter.

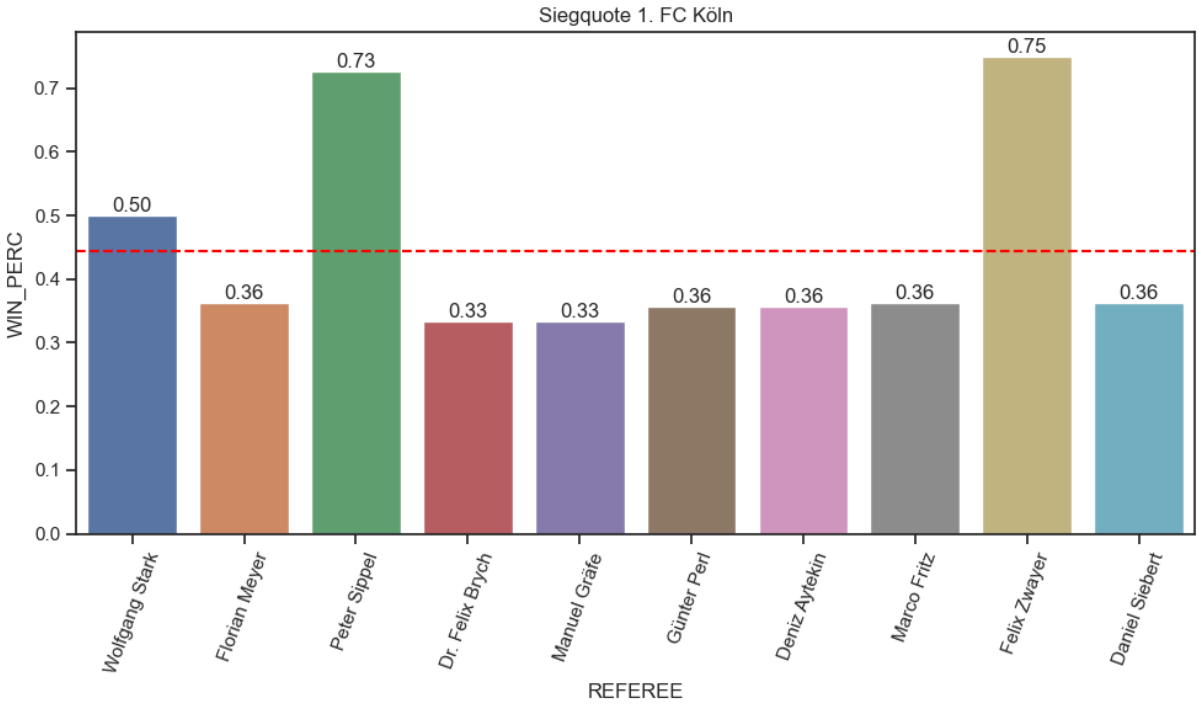

Abbildung 9 - Siegesquote des 1. FC Köln pro Schiedsrichter

Es wurden Schiedsrichter berücksichtigt, die mehr als zehn Partien des 1. FC Köln geleitet haben. Die durchschnittliche Siegesquote des 1. FC Köln unter diesen Schiedsrichtern liegt bei ca. 45%. Auffällig ist, dass mit 73 bzw. 75% Siegesquote

unter Peter Sippel bzw. Felix Zwayer deutlich mehr Spiele gewonnen werden als unter den anderen Schiedsrichtern.

4 Elo-Benchmark Model

Das Elo-Rating kommt ursprünglich vom Schachspiel und die Elo-Zahl gibt die Spielstärke der Schachspieler wieder. Für die Anwendung im Fußball wurden einige Modifizierungen für die Berechnung der Elo-Wertung eines Teams hinzugefügt. Auf der Seite http://clubelo.com/ können zu den meisten Teams die historischen Elo-Wertungen heruntergeladen werden. Sind für beide Teams die Elo-Wertungen für einen Spieltag verfügbar, lässt sich ein erwartetes Ergebnis von zwei Mannschaften mittels der folgenden Formel berechnen

$$W_e = \frac{1}{10^{-dr/400} + 1}$$

Der Parameter *dr* Ist der Punktabstand in der Elowertung zwischen Heim- und Auswärtsteam.
Aus der Perspektive der Heimmannschaft betrachtet, werden zu “dr” noch 100 Einheiten hinzuaddiert. Ein Erwartungswert von 1 deutet auf einen sicheren Sieg für die Heimmannschaft hin, ein Wert nahe zur 0 auf eine sichere Niederlage. Ein Wert, der nahe an 0.5 liegt, würde als unentschieden oder gleich stark gewertet werden.
Das für diesen Anwendungsfall entwickelte Elo-Modell besteht aus den folgenden drei Funktionen in der Datei *helpers.py*

elo_value(path, match_date, home_team, away_team)

Hier wird für die Heimmannschaft und die Auswärtsmannschaft ein vom Datum abhängiger Elo-Wert zurückgegeben.

elo_rating(path, match_date, home_team, away_team)

Anhand eines vom Datum abhängigen Elo-Wertes wird der Erwartungswert von zwei Mannschaften berechnet. Dabei wird eine Niederlage der Heimmannschaft prognostiziert, wenn der Elo-Wert in dem Bereich zwischen 0 - 0.49 liegt. Im Bereich >= 0.49 <= 0.51 wird unentschieden prognostiziert und im Bereich > 0.51 ein Sieg für die Heimmannschaft.

get_elo_forecast_results(X_test, y_test, label_mapping_TEAM, PATH_ELO_CLUBS, HOME_TEAM_WIN, AWAY_TEAM_WIN, DRAW, le_teams)

Die Funktion wertet die Prognosen der elo_rating() Funktion aus. Genauso wie in der Datei “06_Forecast.ipynb” mit anderen Modellen wird das Elo-Modell auf alle Spieldaten, die auch mit den anderen Prognose-Modellen prognostiziert werden, angewendet.

5 Modelltraining und -test

Drei unterschiedliche Ausgänge eines Spiels sind möglich. Das bedeutet, dass die Prognose eines Spielausgangs ein Klassifizierungsproblem mit drei Klassen darstellt.
Klasse 0: Auswärtsmannschaft gewinnt
Klasse 1: Unentschieden
Klasse 2: Heimmannschaft gewinnt

5.1 Modelle

Für die Prognose kommt das sklearn-Framework mit fünf Modellen zum Einsatz.

5.1.1 RandomForestClassifier

Das Klassifikationsmodell ist ein Ensemble-Lernverfahren. Ein "Random Forest" besteht aus einer Sammlung von Entscheidungsbäumen, die während des Trainings erstellt werden. Jeder Baum im Wald wird aus einer Bootstrap-Stichprobe der Trainingsdaten erstellt. Zusätzlich wird bei der Aufteilung jeder Knoten im Baum eine zufällige Auswahl von Merkmalen berücksichtigt.

5.1.2 KNeighborsClassifier

Der KNeighborsClassifier ist ein einfaches Klassifikationsmodell und basiert auf dem k-NN (k-Nearest Neighbors) Algorithmus. Die Idee ist, die k nächstgelegenen Nachbarn eines unbekannten Datenpunkts in einem vorhandenen Datensatz zu finden und anhand dieser Nachbarn eine Vorhersage für die Klasse des unbekannten Datenpunkts zu treffen.

5.1.3 MultinomialNB

Der MultinomialNB ist ein Klassifikationsmodell, welches den multinomialen Naive-Bayes-Algorithmus implementiert. Es geht von einer Multinomialverteilung, einer Verallgemeinerung der Binomialverteilung, aus und gibt die A-posteriori-Wahrscheinlichkeiten für jede Klasse an.

5.1.4 GaussianNB

Der GaussianNB-Algorithmus ist ebenfalls ein Naive-Bayes-Klassifikator. Dieser Naive-Bayes-Algorithmus geht davon aus, dass die Daten für jedes Merkmal (Feature) innerhalb jeder Klasse normalverteilt sind. Auch hier werden A-posteriori-Wahrscheinlichkeiten für jede der Klassen berechnet.

5.1.5 QuadraticDiscriminantAnalysis

Das Modell QuadraticDiscriminantAnalysis (QDA) ist eine Form der diskriminanten Analyse, die verwendet wird, um Beobachtungen in verschiedenen Klassen zu kategorisieren. Diskriminante Methoden versuchen eine Funktion oder "Diskriminante" zu finden, die am besten zwischen den Klassen unterscheidet, basierend auf den Merkmalen der Beobachtungen. Die allgemeine Idee besteht darin, die Diskriminante so zu gestalten, dass die Abstände zwischen Beobachtungen unterschiedlicher Klas-

sen so groß wie möglich und die Abstände zwischen Beobachtungen derselben Klasse so klein wie möglich sind.

5.2 Trainings- und Testdaten, Hyperparametersuche

In dem Notebook 06_Forecast.ipynb wird ein Datensatz ab dem Jahr 2004 mit 5814 Spielen verwendet. 20% des Datensatzes werden zum Testen und Evaluieren der Modelle genutzt. Die Testdaten haben eine Länge von 1163 Spielen und beginnen ab dem 2019-10-06. Der Bereich ab 2004 bis zum 2019-10-06 stellt die Trainingsdaten dar und diese werden für das Training und ein Teil davon für die Parametersuche verwendet.
Die meisten vorgestellten Algorithmen haben eine kleine Anzahl von Hyperparametern und der Aufwand für die Optimierung der Modelle ist deswegen überschaubar. Für die Suche der optimalen Hyperparameter wird die Bayes'sche Optimierung aus der Bibliothek “scikit-optimize” verwendet. Die Methode verwendet ein probabilistisches Modell, oft einen Gauß'schen Prozess, um die zu optimierende Funktion zu schätzen. Durch die Akquisitionsfunktion werden neue Evaluierungspunkte gewählt, wobei sowohl die Exploration unbekannter Bereiche als auch die Ausnutzung bekannter, vorteilhafter Bereiche berücksichtigt werden. Ziel der Methode ist die globale Optimierung der Funktion, um lokale Minima oder Maxima zu überwinden.

6 Ergebnisanalyse

Um die Ergebnisse zwischen den unterschiedlichen Modellen zu vergleichen, werden unterschiedliche Metriken für die Klassifikation herangezogen, die im Folgenden vorgestellt werden.

6.1 Metriken

Die Leistung aller Modelle, einschließlich des Elo-Modells, wird mit den gleichen Metriken gemessen. Dabei werden die Tests ausschließlich auf den Testdaten gemacht, mit denen keines der Modelle vorher Berührung hatte.

6.1.1 Accuracy

Die Genauigkeit gibt das Verhältnis der korrekt klassifizierten Beobachtungen zur Gesamtzahl der Beobachtungen an. Sollte aber vor allem dann, wenn die Verteilung der Klassen nicht ausgewogen ist, mit Vorsicht benutzt werden.

6.1.2 Balanced Accuracy

Die Metrik "Balanced Accuracy" ist eine Erweiterung der herkömmlichen Genauigkeit und wird insbesondere in Szenarien, wie in diesem Fall, mit unausgewogenen Klassenverteilungen verwendet.
Während die herkömmliche Genauigkeit nur das Verhältnis der korrekt klassifizierten Beobachtungen zur Gesamtzahl der Beobachtungen angibt, berücksichtigt die Ba-

lanced Accuracy die Leistung des Modells für jede Klasse einzeln und nimmt daraus den Durchschnitt.

6.1.3 Precision

Präzision ist das Verhältnis der korrekt identifizierten positiven Fälle zu allen vorhergesagten positiven Fällen, d. h. den korrekten und allen falschen Fällen, die vom Modell als "Positiv" vorhergesagt wurden. Diese Metrik ist besonders relevant, wenn ein falsches positiv möglichst vermieden werden muss.

6.1.4 Recall

Recall, der auch als Sensitivität bezeichnet wird, ist das Verhältnis der korrekt identifizierten positiven Fälle zu allen tatsächlichen positiven Fällen, d. h. die Summe der "Richtig Positiven" und der "Falsch Negativen". Die Metrik ist besonders relevant, wenn eine Klassifizierung eines “Falsch Negativen” möglichst vermieden werden muss.

6.1.5 F1-Score

Der F1-Score gilt als Kompromiss zwischen Precision und Recall und sollte besonders in unausgewogenen Klassenverteilungen und in Anwendungen in denen "Falsch Positiven” als auch “Falsch Negativen” hohe Kosten verursachen angewendet werden.

6.2 Analyse der Gesamtergebnisse

Als Erstes wird ein Überblick über die Verteilung der drei Klassen gegeben.

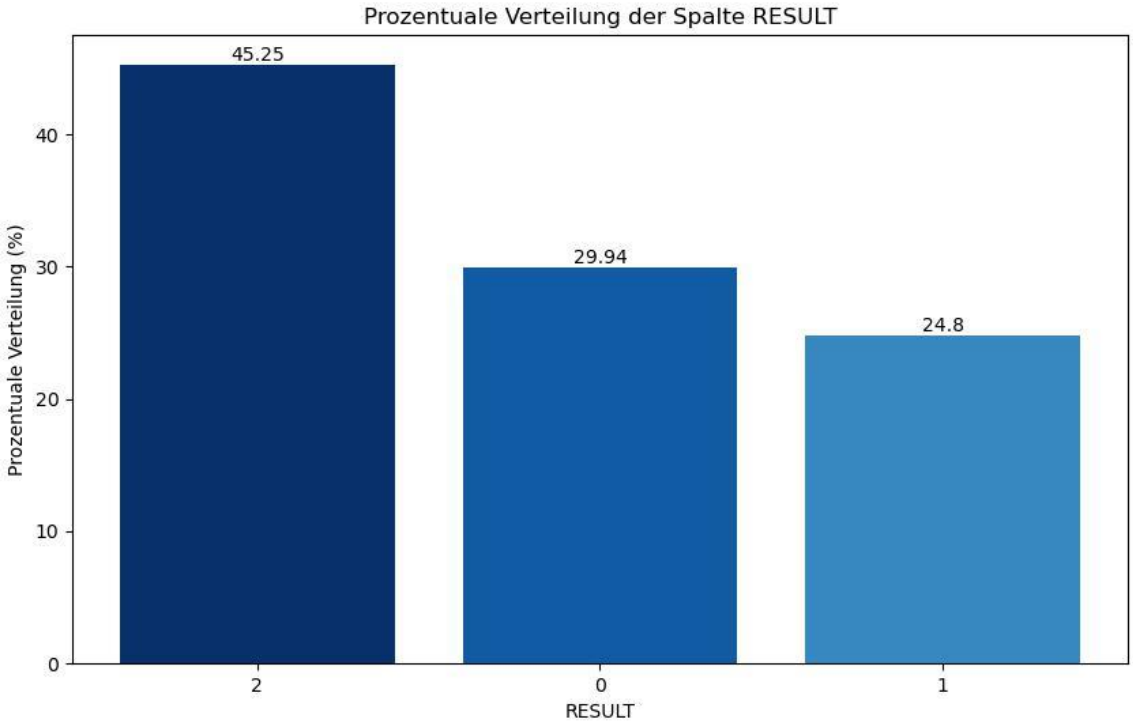

Abbildung 10 - Verteilung der Spalte RESULT

Wie in dem oberen Bild zu sehen, ist die Klasse 2 fast doppelt so oft vertreten wie die Klasse 1. Wird demnach immer auf einen Sieg der Heimmannschaft gesetzt, ganz unabhängig davon, welche Mannschaften gegeneinander spielen, wird eine Genauigkeit von 45,25% erreicht.

6.2.1 Total Accuracy

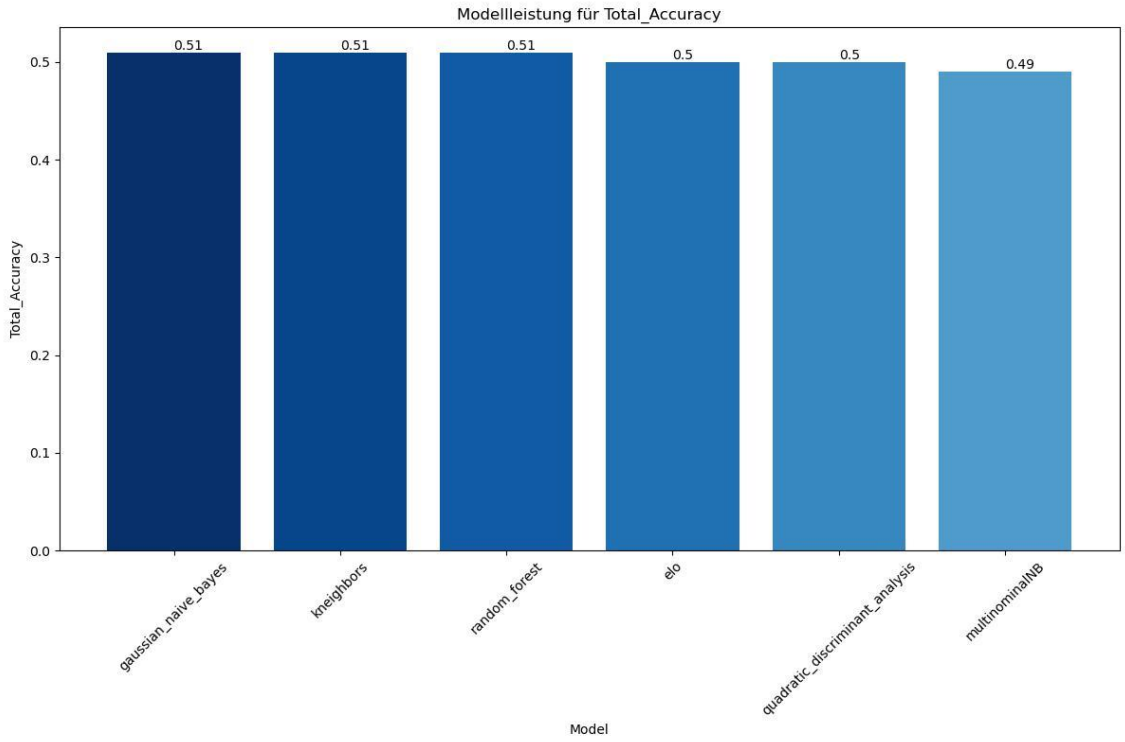

Abbildung 11 - Modelleistung für "Total Accuracy"

Das obere Bild zeigt die Gesamtgenauigkeit über alle Teams zusammengefasst an. Alle Modelle liegen in einem ähnlichen Bereich. Wobei die Modelle GaussianNB, KNeighborsClassifier und RandomForestClassifier mit 0.51 Genauigkeit leicht über den anderen Modellen liegen. Wie schon zuvor erwähnt, sollte die Verteilung der Klassen im Blick gehalten werden und wenn diese unausgewogen sind, sollten weitere Metriken hinzugezogen werden.

6.2.2 Total Balanced Accuracy

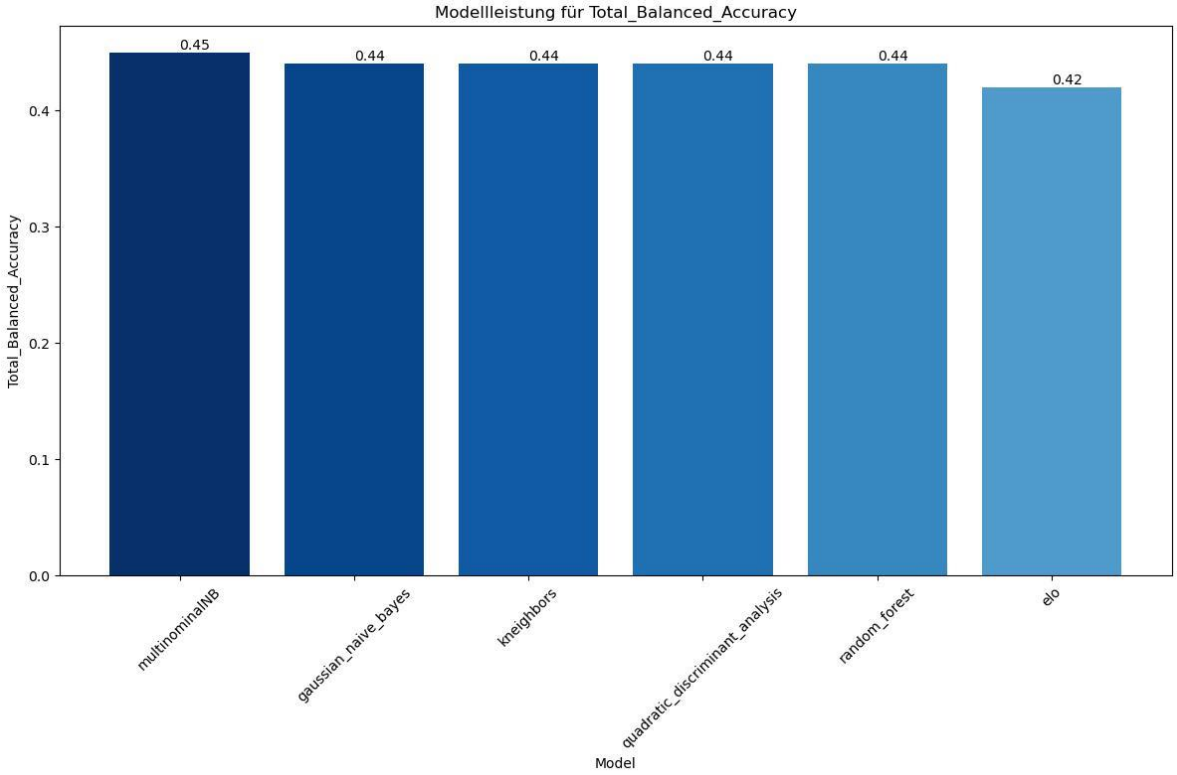

Abbildung 12 - Modelleistung für "Total Balanced Accuracy"

Die Balanced Accuracy fällt erwartungsgemäß geringer aus als die normale Accuracy. Fast alle Modelle liegen in einem ähnlichen Bereich von 0.44 - 0.45. Das Elo-Modell bildet mit 0.42 das Schlusslicht. Bei der unausgewogenen Verteilung der Ergebnisse ist diese Metrik aussagekräftiger als die normale Accuracy.

6.3 Analyse der Mannschaftsergebnisse

Im Folgenden werden einige Ergebnisse pro Mannschaft und Modell analysiert.

6.3.1 Höchste Balanced Accuracy für Heimmannschaft

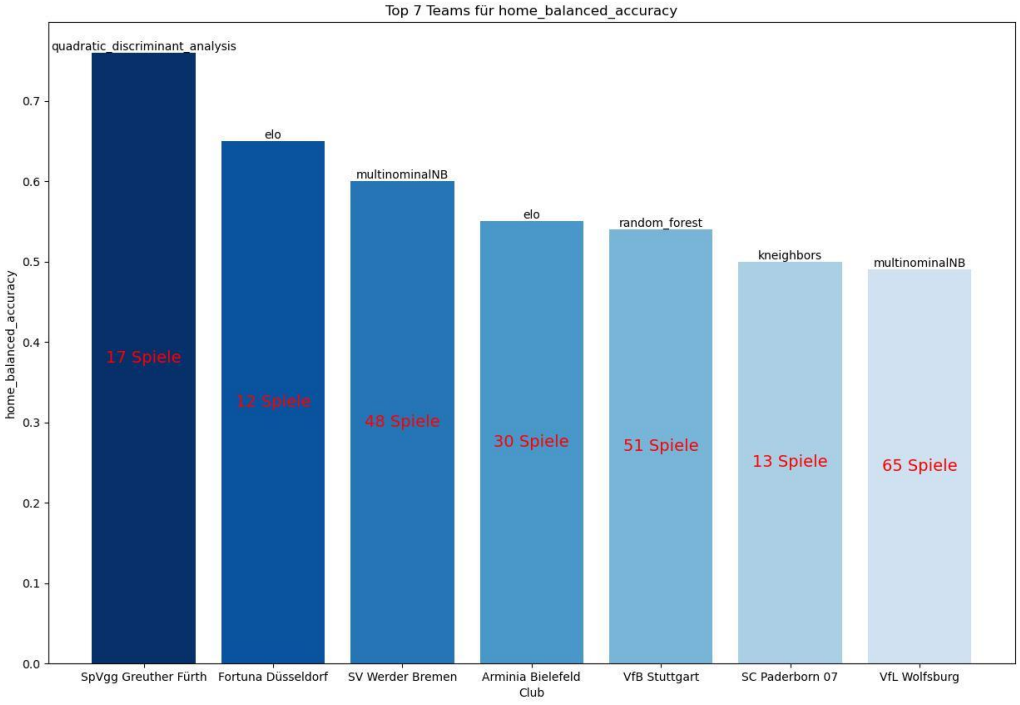

Abbildung 13 - Top 7 Teams für "Home Balanced Accuracy"

Das obere Diagramm zeigt die beste “Balanced Accuracy” in absteigender Reihenfolge für die Prognose der Ergebnisse, wenn das jeweilige Team zu Hause gespielt hat. Die Anzahl der Heimspiele ist für das jeweilige Team im dazugehörigen Balken abgebildet. Das Team “SpVgg Greuther Fürth” hat eine sehr hohe ausbalancierte Genauigkeit von über 0.7. Aber bei einer Anzahl von nur 17 Heimspielen ist es noch nicht sehr aussagekräftig. Beim SV Werder Bremen kann eine ausbalancierte Genauigkeit von knapp 0.6 beobachtet werden und mit fast 50 Heimspielen ist die Aussagekraft wesentlich besser. Es ist zu erkennen, dass sowohl das Elo-Model als auch der multinomiale Naive-Bayes-Algorithmus zwei Mal in den Top sieben in dieser Kategorie vorkommen.

6.3.2 Höchste Balanced Accuracy für Auswärtsmannschaft

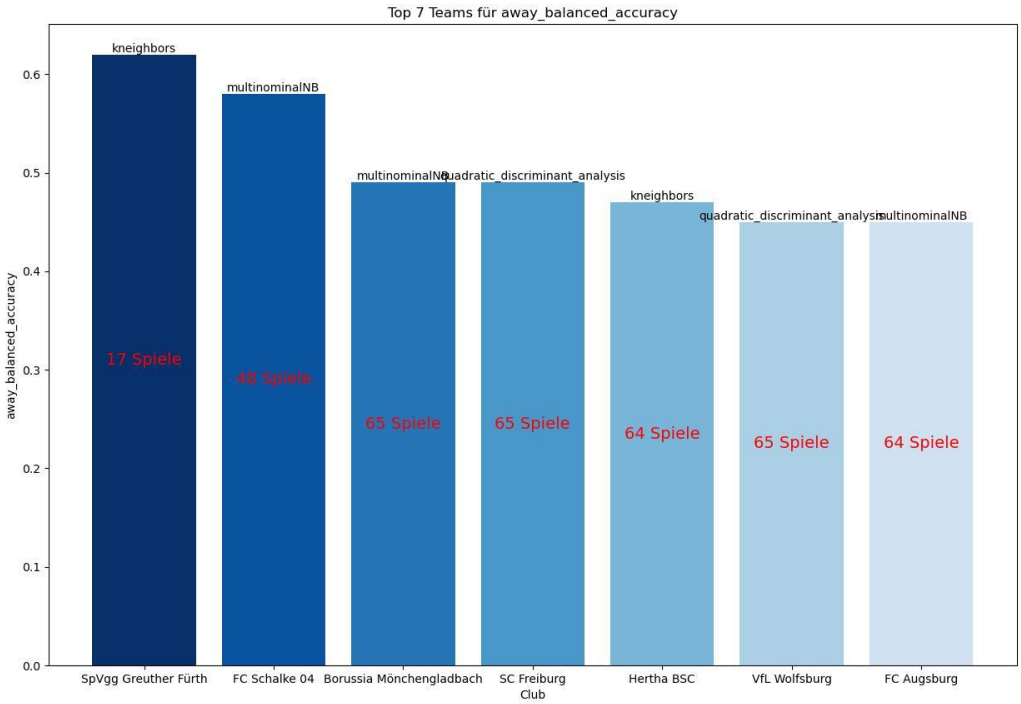

Abbildung 14 - Top 7 Teams für "Away Balanced Accuracy"

Bei der Kategorie “away balanced accuracy” geht es um die ausbalancierte Genauigkeit, wenn das jeweilige Team die Gastmannschaft darstellt. Das scheint etwas schwieriger zu sein als für die Heimmannschaft. Nur bei zwei Teams kann ein Wert über 0.5 beobachtet werden. Wobei das Team “SpVgg Greuther Fürth” wieder auf dem 1. Platz liegt. Die Auswärtsspiele des FC Schalke 04 können mit einer ausbalancierten Genauigkeit von 0.58 relativ gut prognostiziert werden im Vergleich zu den anderen Teams. Bei einer Anzahl von 48 Auswärtsspielen ist das Ergebnis auch aussagekräftig. Das Elo-Modell schafft es bei der Kategorie nicht unter die Top 7.

6.3.3 Höchster F1-Score: Heimsieg

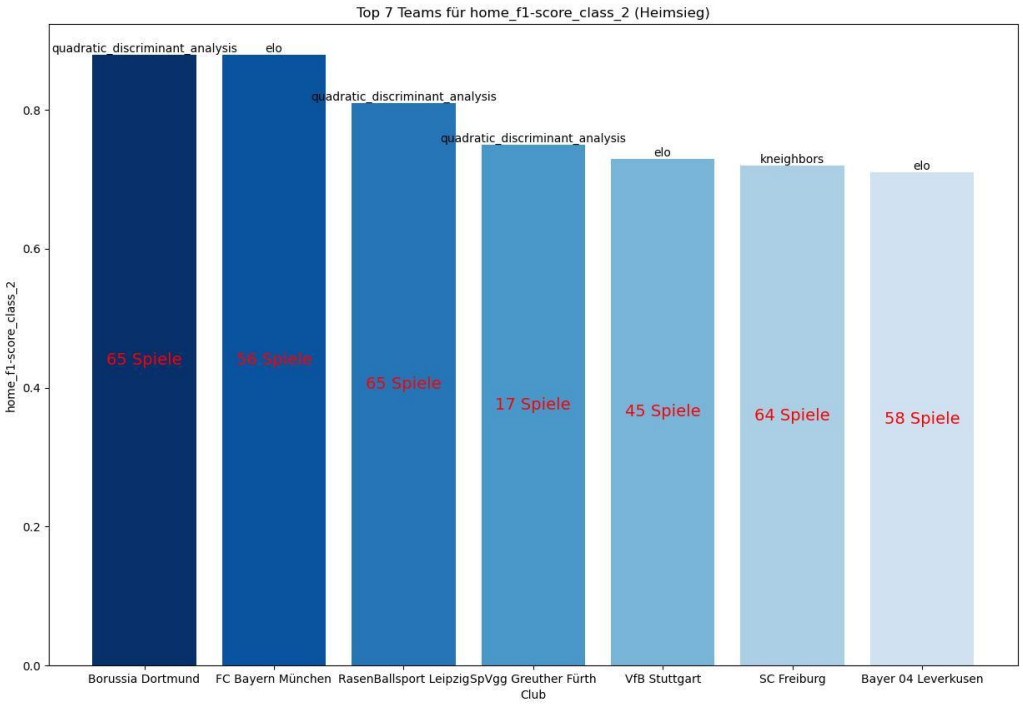

Abbildung 15 - Top 7 Teams für " F1 score class 2" (Heimsieg)

Die Metrik F1-Score kann als guter Kompromiss zwischen den Metriken “Recall und Precision” betrachtet werden. Bei der Kategorie für die Prognose der Klasse 2 (Heimmannschaft gewinn) sind im oberen Bild die besten sieben Ergebnisse zu sehen. Mit einem F1-Score von ca. 0.85 führen die Teams “Borussia Dortmund” und “FC Bayern” die Liste an. Unter den Top Plätzen ist das Modell mit der diskriminanten Analyse und das Elo- Modell mehrfach vertreten.

6.3.4 Höchster F1-Score: Auswärtssieg

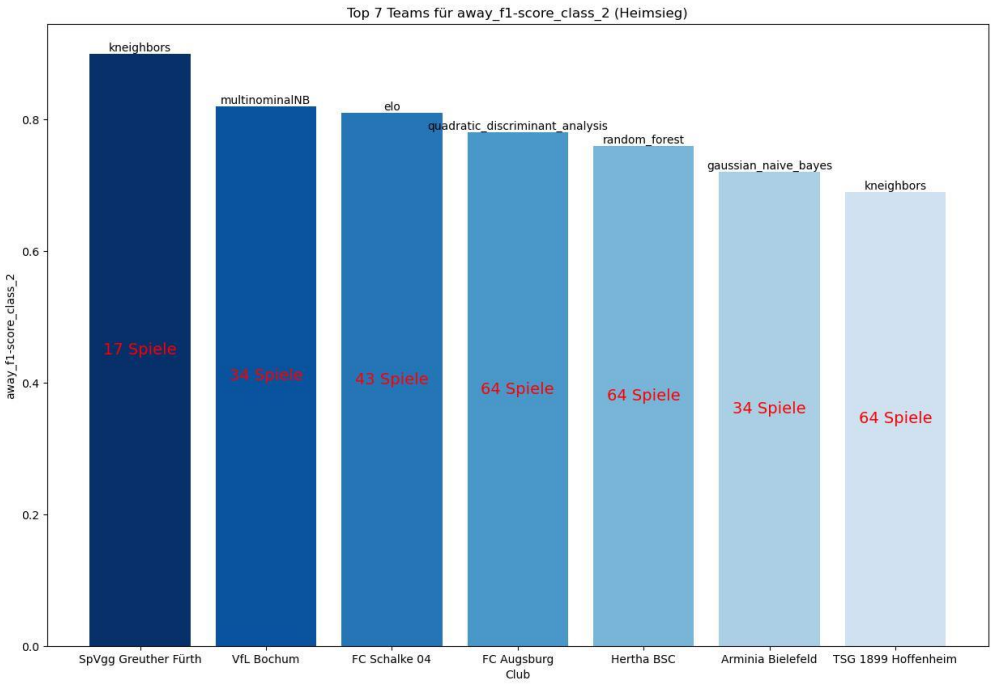

Abbildung 16 - Top 7 Teams für "Away F1 score class 2" (Heimsieg)

Bei der Kategorie geht es um den F1-Score für den Heimsieg, wenn die oben gezeigten Teams die Gastmannschaft waren. Das bedeutet aus der Sicht der oberen Teams eine Auswärtsniederlage. Wie schon des Öfteren führt das Team “SpVgg Greuther Fürth” die Liste mit einem Wert von ca. 0.85 an. Anscheinend können für das Team die Auswärtsniederlagen verhältnismäßig gut prognostiziert werden. Was die Modelle angeht, so kann ein dominierendes Modell nicht ausgemacht werden, denn unter den Top sieben Plätze sind alle Modelle vertreten.

6.4 Feature Importance

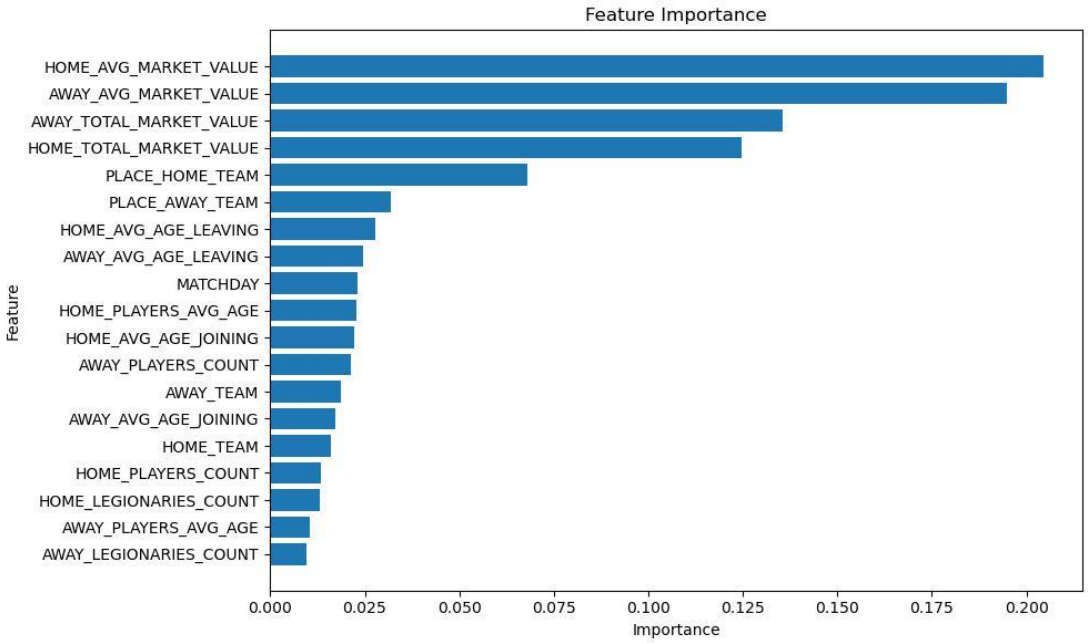

Abbildung 17 - Feature Importance

Im oberen Diagramm sind die in absteigender Wichtigkeit sortierten Merkmale vom Modell “Random Forest” aufgelistet. Die Wichtigkeit kann hier als die normalisierte Gesamtmenge betrachtet werden, daran gemessen, dass dieses Features zur Verbesserung des Modells beigetragen hat. Wie zu sehen ist, spielt der Marktwert der Mannschaften hier die größte Rolle. Ein weiteres wichtiges Merkmal ist die Tabellenposition der Heimmannschaft “PLACE_HOME_TEAM”. Die anderen Merkmale spielen eine untergeordnete Rolle.

7 Hypothesentests

7.1 Prognosemodelle vs. Zufallsmodell

Unsere Zielsetzung besteht darin, statistische Hypothesentests zu konzipieren, um die statistische Signifikanz der Leistung unserer Modelle im Vergleich zum reinen Zufall oder einer reinen Rategenauigkeit bei der Vorhersage von Spielergebnissen zu ermitteln. Um dies erfolgreich durchzuführen, ist es notwendig, im Voraus Informationen über die Apriori-Wahrscheinlichkeit zu sammeln, die uns Aufschluss darüber gibt, wie die Verteilung der Spielergebnisse aussieht. Konkret interessiert uns der Anteil, der angibt, wie oft die Heimmannschaft gewinnt, wie oft es zu einem Unentschieden kommt und wie oft die Heimmannschaft verliert. Dies erfordert eine eingehende Analyse der historischen Verteilung von Spielausgängen, wobei wir auf einen Datensatz von insgesamt 5814 Spielen zurückgreifen.

Heimmannschaft gewinnt	Unentschieden	Heimmannschaft verliert
45%	25%	30%

Im Rahmen dieses Projekts können mithilfe von Python eine gewichtete, pseudo-zufällige Auswahl der Spielergebnisse durchgeführt werden. Die Gewichtung orientiert sich an der oben erzeugten Verteilung. Anschließend werden diese Ergebnisse mit den Prognosemodellen verglichen.
Für die Analyse von Unterschieden zwischen zwei Stichproben bietet sich der McNemar-Test als geeignetes Verfahren an. Dieser Test eignet sich insbesondere für binäre Ausgabewerte, wie korrekte Vorhersagen gegenüber falschen Vorhersagen.
Die Nullhypthose zu diesem Test lautet:

H_0: Es gibt keinen Unterschied zwischen Stichprobe1 (Prognosemodell) und Stichprobe2 (zufälliges Spielergebnis)

Bevor wir den McNemar Test auswerten, ist es hilfreich vorab eine Kontingenztafel aufzustellen, die die Anzahl der richtig prognostizierten und richtig zufällig erzeugten Spielergebnisse gegenüber der abweichenden Spielergebnisvorhersagen stellt. Relevant für den Test sind allerdings nur die abweichenden Ergebnisse zwischen dem Zufallsmodell und dem Prognosemodell (b und c)

		Zufälliges Raten	
		Richtig	Falsch
Prognosemodell	Richtig	a	b
	Falsch	c	d

Auf dieser Basis können wir die Teststatistik χ^2 errechnen und dem kritischen Wert der χ^2-Verteilung gegenüberstellen.
Die Teststatistik für die Anwendung des McNemar-Tests ergibt sich ausgehend von der Kontingenztafel durch den Wert $\chi^2 = \frac{(b-c)^2}{b+c}$.
Demgegenüber wird der kritische Wert der χ^2-Verteilung mit 1 Freiheitsgrad und dem 95%-Quantil $\chi^2_{1;0,95}$ = 3.84 gestellt, wenn von einem Signifikanzniveau von 5% ausgegangen wird. Der Wert von 3.84 lässt sich von der Tabelle der χ^2-Verteilung ablesen.
Die Nullhypothese wird abgelehnt, falls $\chi^2 > \chi^2_{1;0,95}$ und die Alternativhypothese wird angenommen, dass das Prognosemodell vom Zufallsmodell statistisch signifikant abweicht.

Ergebnisse der Auswertung:

Random Forest vs. Zufallsmodell:

		Zufälliges Raten	
		Richtig	Falsch
Prognosemodell	Richtig	252	337
	Falsch	186	388

$$43{,}597 = \chi^2 > \chi^2_{1;0{,}95} = 3{,}84$$

Resultat: Die Nullhypothese wird abgelehnt.

K-Neighbor vs. Zufallsmodell:

		Zufälliges Raten	
		Richtig	Falsch
Prognosemodell	Richtig	253	338
	Falsch	185	387

$$44{,}759 = \chi^2 > \chi^2_{1;0{,}95} = 3{,}84$$

Resultat: Die Nullhypothese wird abgelehnt.

Multinominal Naive Bayes vs. Zufallsmodell:

		Zufälliges Raten	
		Richtig	Falsch
Prognosemodell	Richtig	233	332
	Falsch	205	393

$$30{,}035 = \chi^2 > \chi^2_{1;0{,}95} = 3{,}84$$

Resultat: Die Nullhypothese wird abgelehnt.

Gaussian Naive Bayes vs. Zufallsmodell:

		Zufälliges Raten	
		Richtig	Falsch
Prognosemodell	Richtig	250	339
	Falsch	188	386

$$43{,}266 = \chi^2 > \chi^2_{1;0{,}95} = 3{,}84$$

Resultat: Die Nullhypothese wird abgelehnt.

Quadratische Diskriminanzanalyse vs. Zufallsmodell:

		Zufälliges Raten	
		Richtig	Falsch
Prognosemodell	Richtig	244	340
	Falsch	194	385

$$39{,}917 = \chi^2 > \chi^2_{1;0{,}95} = 3{,}84$$

Resultat: Die Nullhypothese wird abgelehnt.

In allen untersuchten Fällen wurden demnach die Nullhypothesen abgelehnt. Das heißt, die Prognosemodelle unterscheiden sich statistisch signifikant von dem Zufallsmodell. Da der Test selbst keine Aussage darüber trifft, wie stark und in welche Richtung sich die Abweichungen bewegen, kann die Kontingenztafel zur Orientierung verwendet werden, um festzustellen, dass stets das Prognosemodell mehr richtige Vorhersagen getroffen hat, während das Zufallsmodell falsch lag, als umgekehrt. Der b-Wert ist stets größer als der c-Wert. Es kann daher angenommen werden, dass die hier untersuchten Prognosemodelle stets signifikant besser sind als das Zufallsmodell.

7.2 Signifikante Einflüsse auf das Spielergebnis

Ferner wurde untersucht, welche Variablen einen signifikanten Einfluss auf das Spielergebnis „RESULTS“ bzw. auf die Anzahl der gefallenen Tore bei der Heimmannschaft haben.
Hierzu wurden zu allen Variablen Hypothesen Tests erstellt, die wir im Einzelnen in drei Kategorien unterteilen.

1.) Hat eine nominale Variable einen Einfluss auf ein nominales Spielergebnis („RESULTS“)? Anders formuliert: Gibt es eine stochastische Abhängigkeit zwischen zwei nominalen Variablen?
Um diese Fragestellung zu beantworten, bietet es sich an ein χ^2-Test anzuwenden.
Geprüft wird hier die Nullhypothese:
H_0: Die beiden Variablen sind stochastisch unabhängig.

2.) Hat eine metrische Variable einen linearen Zusammenhang zu einer weiteren metrischen Variable („HOME_GOALS“ oder „GOALS_DIFFERENCE“ [Diffe-

renz der Anzahl der Tore zwischen Heimmannschaft und Auswärtsmannschaft]).
Um diese Fragstellung zu beantworten, bietet es sich an ein Pearson Korrelationstest anzuwenden.
Geprüft wird hier die Nullhypothese:
H_0: Es gibt keinen linearen Zusammenhang zwischen den beiden Variablen.

3.) Hat eine metrische Variable einen Einfluss auf ein nominales Spielergebnis („RESULTS“)? Erneut wird an dieser Stelle die stochastische Abhängigkeit geprüft.
Um diese Fragestellung zu beantworten, bietet es sich an eine Varianzanalyse (ANOVA) anzuwenden.
Geprüft wird hier die Nullhypothese:
H_0: Die beiden Variablen sind stochastisch unabhängig.

Alle signifikanten Ergebnisse sind in GitHub unter dem Link
web-mining/11_CorrelationAnalysis.ipynb at main · lucajanas/web-mining (github.com)
festgehalten.

8 Lessons learned

Während der Ausarbeitung dieser Projektarbeit haben wir uns mit dem Sammeln und Analysieren von Daten aus der ersten Bundesliga im Fußball beschäftigt. Dann haben wir die Daten in Tabellen organisiert und versucht, vorherzusagen, ob ein Fußballspiel zu einem Heimsieg, einem Unentschieden oder einem Auswärtssieg führen wird.

Wir haben während der Ausarbeitung festgestellt, dass die Qualität der gescrapten Daten von größter Bedeutung ist. Wenn die Daten Fehler enthalten oder ungenau sind, können unsere Vorhersagen falsch sein. Deshalb müssen wir sicherstellen, dass die Daten, die wir sammeln, wirklich verlässlich sind. Dabei haben wir gelernt, wie wichtig es ist, zuverlässige Datenquellen zu verwenden und die gescrapten Daten sorgfältig zu bereinigen und zu überprüfen. So ist uns zum Beispiel nur zufällig aufgefallen, dass sich die Angabe der Transfereinnahmen und -ausgaben nicht nur in Millionenangaben, sondern auch in Tausenderangaben vorliegen.

Bei Webscraping-Aufgaben ist es ebenso unerlässlich, robusten Code zu schreiben, der mit Fehlern und Änderungen auf der Website möglichst dynamisch umgehen kann. Es ist wichtig, dass unser Programm damit umgehen kann und dass wir es

überwachen. Zum Beispiel gab es Ausnahmen über die Anzahl der Vereine pro Saison. Der Code sollte daher möglichst viele mögliche Sonderfälle abfangen können.
Gleichzeitig muss ein Monitoring oder ein Warnhinweis implementiert werden, wenn das Skript auf einen Fehler läuft. Dies kann des Öfteren schon mal vorkommen, da die HTML-Seiten und Quellcodes seitens der Betreiber auch mal umgestaltet und umprogrammiert werden. So kann man schnellstmöglich auf Veränderungen reagieren und seinen Code anpassen.
Wir haben verstanden, dass Webscraping nicht nur technische, sondern auch ethische und rechtliche Überlegungen erfordert. Die Einhaltung der Website-Richtlinien und die Beachtung der geltenden Gesetze und Vorschriften sind von größter Bedeutung.
Wir haben festgestellt, dass es diverse Möglichkeiten beim Webscraping gibt, dieselben Daten an gleicher Stelle zu selektieren. Hier bietet Python mit BeautifulSoup, die Anbindung mit Selenium, mit Scrapy oder mittels XPATH verschiedene Kombinationsmöglichkeiten in der Anwendung, um an das selbe Ziel zu gelangen.
Im Großen und Ganzen können wir sagen, dass die Erfahrungen und Erkenntnisse uns nicht nur bei dieser Ausarbeitung, sondern auch für zukünftige Projekte im Bereich Web-Mining und Datenanalyse wertvolle Einsichten verschafft hat. Sie verdeutlichen, dass der Prozess von der Datenerfassung bis zur Ergebniskommunikation sorgfältige Planung, ständige Anpassung und die Berücksichtigung ethischer und rechtlicher Aspekte erfordert.

9 Aufgabenaufteilung

Kevin Diec: Scraping und Aufbereitung von Vereins-, Transfer- und Wetterdaten sowie Zufallsbenchmark durch Hypothesentests

Luca Janas: Scraping von Spieltagsdaten und Bundesligatabellen, Mergen der gescrapten Datensätze, Datenanalyse- und Visualisierung

Alfred Anselm: Elo Benchmark, Feature Engineering, Modelltraining und -evaluation