universität wien

DIPLOMARBEIT / DIPLOMA THESIS

Titel der Diplomarbeit / Title of the Diploma Thesis

„Vorhersage von Torschussevents im Fußball basierend auf vorangegangenen Events mit Hilfe von Klassifikationsalgorithmen“

verfasst von / submitted by

Kevin Kristiner

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Magister der Naturwissenschaften (Mag.rer.nat.)

Wien, 2018 / Vienna, 2018

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 190 482 406
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Lehramtsstudium UF Bewegung und Sport UF Mathematik
Betreut von / Supervisor:	Dipl.-Math. Dr. Michael Stöckl

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst habe und nur die ausgewiesenen Hilfsmittel verwendet habe. Diese Arbeit wurde daher weder an einer anderen Stelle eingereicht (z.B. für andere Lehrveranstaltungen) noch von anderen Personen (z.B. Arbeiten von anderen Personen aus dem Internet) vorgelegt.

Ort, Datum	Unterschrift

Danksagung

An dieser Stelle möchte ich mich recht herzlich bei meinen Eltern bedanken, die mich während meines gesamten Werdegangs unterstützt haben. Ihr habt mir diesen Studienabschluss ermöglicht und dafür bin ich euch sehr dankbar. Besonders möchte ich zudem meine Freundin Anna-Carina erwähnen, die mir stets Motivation und Kraft gab, um unter anderem diese Arbeit zu schreiben.

Mein Dank gilt auch Herrn Dr. Michael Stöckl für seine hervorragende Arbeit bei der Betreuung dieser Diplomarbeit. Er hatte dabei immer ein offenes Ohr für meine Anliegen und zeichnete sich durch seine freundliche Art gepaart mit fachlicher Kompetenz aus.

Des Weiteren möchte ich mich bei allen Personen bedanken, die mich im Laufe meines Studiums begleitet haben. Speziell erwähnen möchte ich dabei meine Freunde Andreas, Jakob & Nikolaus, welche mir durch ihre Fähigkeiten im Bereich der Informatik stets neue Denkansätze lieferten, wenn ich diese im Rahmen dieser Arbeit nötig hatte.

Kurzfassung

Diese Diplomarbeit beschäftigt sich mit der Untersuchung von Fußballdaten mittels maschinellem Lernen. Ausgehend von einigen Begriffsdefinitionen wurde anschließend der aktuelle Forschungsstand ermittelt. Im Zuge des Forschungsprozesses wurden insgesamt 1123 Fußballspiele der aktuellen Saison aus den Top-Fünf Ligen des europäischen Klubfußballs analysiert und hinsichtlich der zentralen Fragestellung der Arbeit, dem Vorhersagen von Tor-Events, bearbeitet. Dieses Ziel sollte durch Klassifizierungsalgorithmen erreicht werden, welche mithilfe der Open-Source-Bibliothek scikit-learn angewandt wurden. Der Vorverarbeitung der Daten wird ein eigenes Kapitel gewidmet und es wird dabei genau darauf eingegangen, welche Informationen für die Untersuchung genutzt wurden. Für ein besseres Verständnis wurden die wesentlichen Algorithmen, welche im Rahmen der Arbeit verwendet wurden, vorweg genauer ausgeführt. Abschließend wurden die Ergebnisse der Prognosen des Computers präsentiert, validiert und zudem auch optimiert.

Abstract

This diploma thesis is dealing with the analysis of soccer data employing Machine Learning. Based on definitions of terms relevant to soccer and the concept of Machine Learning, an overview on the state of research is given. In the process of research 1123 games of soccer from the current season of the top-five leagues in Europe's club football have been analysed. The main subject of this thesis is to predict goal events. This aim should be achieved by the use of scikit-learn, an open-source-library for Machine Learning. The explanation of the preprocessing of the data is dealt with in a separate chapter. The focus also lies upon how given information is used for the study. For a better understanding, the essential algorithms which are used in this thesis are explained before they are utilized. Finally, the outcome of the predictions is presented, validated and optimised.

Inhaltsverzeichnis

1 EINLEITUNG

> *"Die Menschen gehen zum Fußball, weil sie nicht wissen, wie's ausgeht."*
>
> (Sepp Herberger)

Seit Sepp Herberger (1897-1977), Trainer der Bundesrepublik Deutschland beim Weltmeistertitel 1954, diese Aussage tätigte, hat sich das Sportspiel Fußball rasant weiterentwickelt. Einhergehend mit der Entwicklung des Spiels selbst, sind im Laufe der Zeit auch eine Vielzahl an Methoden zur systematischen Wettkampfanalyse entstanden.

> *„Pelé hat keine Ahnung vom Fußball. Seine Analysen sind immer falsch. Wenn du gewinnen willst, höre zu, was Pelé sagt – und mache genau das Gegenteil."*
>
> (Luiz Felipe Scolari)

Eine so einfache Strategie für einen Gewinn, welche dieses Zitat des ehemaligen Trainers der brasilianischen Nationalmannschaft impliziert, wäre natürlich wünschenswert. Seit Jahrzehnten beschäftigen sich Experten mit Wettkampfanalysen. Diese sollen als Ergänzung zu sportmotorischen, psychologischen oder auch biomechanischen Testverfahren zusätzliche Erkenntnis liefern, um ein umfassenderes Verständnis für das Spiel selbst zu erlangen. Wie Nopp (2009, S. 24) feststellt, scheint ein Methodenmix verschiedener Ansätze aus der Wettkampfanalyse am vielversprechendsten zu sein. Freie Spielanalyse kombiniert mit schriftlich, graphisch, akustisch gebundenen sowie per Video und Computer erstellten Spielinformationen hält er für optimal.

Die vorliegende Arbeit beschäftigt sich mit der Auswertung großer Datensätze aus der aktuellen Fußballsaison (2017/18). Spielinformationen wurden also computerunterstützt analysiert. Dafür wurden jedoch nicht die Deskriptiv- oder Inferenzstatistik als Hilfsmittel herangezogen, sondern eine andere, moderne Technik der Datenanalyse angewandt, welcher sehr viel Potential für die Zukunft beigemessen wird: „Machine Learning“. Im folgenden Kapitel sollen dazu einige Grundbegriffe erklärt und definiert werden.

1.1 Begriffsexplikationen

1.1.1 Data Science & Machine Learning

Der Begriff Data Science ist relativ neu und wurde im Jahr 2002 vom International Council for Science anerkannt. Die mathematische Basis hinter Data Science ist hingegen schon einige Jahrhunderte alt und im Wesentlichen ein Verfahren der Analyse von Statistiken und Wahrscheinlichkeiten (vgl. Mueller & Massaron, 2016, S. 31).

Das Zeitalter der modernen Technologie ist durch die Unmengen an verfügbaren Daten gekennzeichnet. Um diese jedoch sinnvoll zu nutzen, braucht es geeignete Methoden. Machine Learning (kurz: ML) bietet dabei einen vielversprechenden Ansatz.

Raschke & Mirjalili (2018, S. 25) sehen Machine Learning als das spannendste Forschungsfeld der Informatik. Ziel des „maschinellen Lernens“ ist es, anhand vom Algorithmen Muster in Daten zu erkennen und dadurch Vorhersagen über zukünftige Ereignisse treffen zu können.

Durch diese selbstlernenden Algorithmen sollen also Daten in Wissen verwandelt werden. Auch in unserem Alltag spielen diese Verfahren eine zunehmend größere Rolle: E-Mail-Spamfilter, Spracherkennungssoftware oder auch bald selbstfahrende Autos sind nur einige Beispiele an Möglichkeiten, welche durch Machine Learning realisiert werden können (vgl. Raschke & Mirjalili, 2018, S. 26).

Shalev-Shwartz & Ben-David (2014, S. 4) unterscheiden zwischen überwachtem und unüberwachtem maschinellen Lernen. Beim überwachten Lernen soll der Computer bestimmte Kennzeichen vorhersagen, welche als Extrainformation im Testdatensatz bereits mitgelernt werden. Eine Unterkategorie des überwachten Lernens stellt die Klassifizierung dar. Es wird dabei davon ausgegangen, dass eine neue Beobachtung zu einer bestimmten Gruppe gehört (vgl. Mueller & Massaron, 2016, S. 239). Bei diesen Vorhersagen handelt es sich um diskrete Ausgänge. Im Gegensatz dazu liefert die Regression, eine weitere Unterkategorie des überwachten Lernens, stetige Ausgabewerte (vgl. Raschka & Mirjalili, 2018, S. 27). Beim unüberwachten Lernen gibt es keine Extrainformation, welche im Testdatensatz inkludiert ist. Das Ziel bei dieser Variante maschinellen Lernens ist es, die Daten sinnvoll zusammenzufassen und zu komprimieren. Eine typische Anwendung des unüberwachten Lernens ist das sogenannte Clustering (vgl. Shalev-Shwartz & Ben-David, 2014, S. 5).

Auch im Bereich der Sportspielanalyse wird Machine Learning als Ansatz verwendet, um neue Erkenntnisse zu gewinnen. Bevor jedoch auf diese eingegangen wird, müssen noch einige Grundbegriffe erläutert werden.

1.1.2 Sportspiel Fußball

Den Begriff Sportspiel und dessen Etablierung gibt es wohl schon über ein Jahrhundert. Die Definition und Beschreibung dieses Begriffes ist im Laufe der Zeit immer wieder adaptiert worden. Folgende Gegenstandsbestimmung stammt von Lames (1994, S. 15):

> „Sportspiele sind Sportarten mit international kodifiziertem Regelwerk, bei denen zwei Parteien […] in einen Interaktionsprozess eintreten, der dadurch zustande kommt, dass beide Parteien gleichzeitig ihr eigenes Spielziel anstreben und verhindern wollen, dass die gegnerische Partei ihr Spielziel erreicht."

Das Sportspiel, welches in dieser Arbeit behandelt wird, ist das wohl beliebteste weltweit: Fußball. Das oben angesprochene Spielziel im Fußball ist es, ein Tor zu erzielen, während gleichzeitig verhindert werden soll, dass das gegnerische Team dies tut. Simultan verfolgt das andere Team dasselbe Ziel, was eine typische Eigenschaft des Sportspiels darstellt (vgl. Lames & McGarry, 2007, S. 62).

Weitere Merkmale im Sportspiel, welche es zu bestimmen gilt, sind die Begriffe der Leistung und des Erfolgs.

Die sportliche Leistung und ihre Entwicklung sind nach Schnabel & Thieß (1993, S. 531) wesentlich durch den sportlichen Wettkampf bestimmt. Sie ist die wesentlichste Zeitgröße und damit auch das Hauptkriterium des sportlichen Trainings. Neben der sportlichen Leistung ist in Sportspielen auch die taktische Leistung ein Charakteristikum.

Der sportliche Erfolg, aufgefasst als der erzielte Rangplatz im Wettkampf, ist nach Carl eindeutig von der sportlichen Leistung zu unterscheiden (Carl in Schnabel & Thieß, 1993, S. 266). Mit sportlichem Erfolg kann jedoch auch das Erreichen einer angestrebten sportlichen Leistung verstanden werden (vgl. Schnabel & Thieß, 1993, S. 266).

Auch Lames (1994, S. 16) merkt kritisch an, dass für den Erfolg im Sport Relativierungen typisch sind. Die Basis dieser Relativierungen stellen ihm zufolge Informationen dar, welche insbesondere der systematischen Spielbeobachtung nicht zugänglich sind. Um eine Zugänglichkeit in der systematischen Spielbeobachtung zu erreichen, wird der Erfolg aus leistungsdiagnostischer Perspektive mit dem Erreichen des Spielziels identifiziert (vgl. Lames, 1994, S. 16). Die Wettkampfbeobachtung wird nach Röthig & Prohl (2003, S. 653) als deskriptives und analytisches Verfahren zur Wettkampfkontrolle und zur Ergänzung der Trainingskontrolle beschrieben. Eine standardisierte, merkmalsbezogene Wettkampfbeobachtung orientiert sich an den wissenschaftlichen Kriterien der Objektivität und Zuverlässigkeit. Durch sie sollen präzise Daten für eine nachfolgende Analyse des Wettkampfs generiert werden (vgl. Röthig & Prohl, 2003, S. 653).

1.1.3 Spielanalyse

Einhergehend mit der Entwicklung des Sportspiels selbst, reifte das Bedürfnis heran, Leistungen von Spielern greifbar zu machen. Zusätzlich zu taktischen Überlegungen entstanden auch erste Ansätze, welche heute unter dem Begriff der Spielanalyse zu verstehen sind. Die Anfänge wurden jedoch nicht im Fußball gemacht, sondern im Baseball. Bereits 1912 gab es die erste Veröffentlichung, in der ein Sportjournalist Spielstatistiken systematisch aufarbeitete (vgl. Memmert & Raabe, 2017, S. 24).

Im Fußball war der Brite Charles Reep der erste bekannte Spielanalyst. Im Jahre 1950 hielt er in einem Notizbuch das Spielgeschehen zwischen Swindon Town und den Bristol Rovers fest und entwickelte dabei ein eigenes Notationssystem (vgl. Memmert & Raabe, 2017, S. 28).

Mit fortlaufender Zeit wurden die Systeme zur Analyse von Sportspielen beständig weiterentwickelt. Nach diesem kurzen Einblick in die Anfänge der Sportspielanalyse wird nun ein Überblick über den gegenwärtigen Entwicklungsstand gegeben.

2 FORSCHUNGSSTAND

In diesem Kapitel werden gängige technologische Ansätze und Methoden der Sportspielanalyse vorgestellt. Um einen Überblick über die technologischen Entwicklungen zu erlangen, haben Memmert & Raabe die Spielanalyse in vier Kategorien eingeteilt (siehe Abbildung 1):

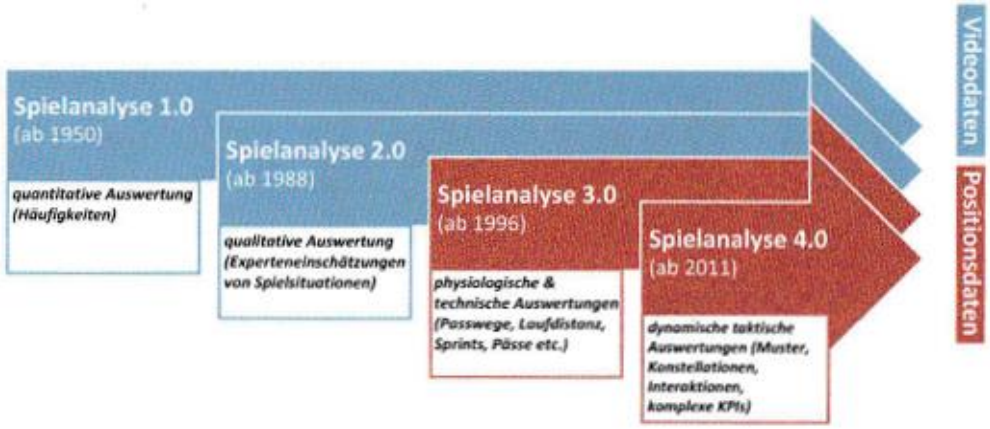

Abbildung 1: Kategorisierung der Spielanalyse (Memmert & Raabe, 2017, S. 8)

Die ab 1950 stattfindende Spielanalyse 1.0 beinhaltet die quantitative Auswertung von Häufigkeiten. Ab 1988 ist von Spielanalyse 2.0 die Rede, wobei qualitative Auswertungen von Experten durchgeführt werden. Die Spielanalyse 3.0, welche sich nicht mehr auf Videodaten stützt, sondern anhand von Positionsdaten durchgeführt wird, umfasst Auswertungen zu Passwegen, Laufdistanzen etc. Kennzeichen für die erst ab 2011 aufkommende Spielanalyse 4.0 sind dynamische taktische Auswertungen. Verschiedene Muster und Konstellationen werden verarbeitet, und komplexe Zusammenhänge sollen dabei sichtbar gemacht werden (vgl. Memmert & Raabe, 2017).

2.1 Methoden der Datenerfassung

Ein wichtiger Faktor der Spielanalyse ist die Erfassung der Daten. Wie schon in Kapitel 1.1.3 erwähnt, waren Handnotationssysteme die erste Methode, wie Fußballdaten erfasst wurden. Diese bildeten jahrelang den Standard, doch dieser Prozess war vor allem enorm zeitaufwendig. Mehrere Stunden dauerte es oft, bis alle Aktionen vollständig notiert wurden. Im Verlauf der 1990er – Jahre änderte sich dies und es fand die fast vollständige Digitalisierung der Spielanalyse statt (vgl. Memmert & Raabe, 2017, S. 32).

Lames (1994, S. 130) bezeichnet die Datenerfassung als „Flaschenhals“ der systematischen Spielbeobachtung. Zu dieser Zeit war ihm zufolge die Online-Erfassung von Daten während des Spiels zwar höchst wünschenswert, wies damals jedoch nicht die notwendige Informationsdichte und -zuverlässigkeit auf. Bereits im Jahr 1987 entwickelten Church und Hughes sogenannte Concept Keyboards. Diese speziellen Tastaturen hatten jede Taste mit einer Spielaktion (Pass, Schuss, etc.) belegt, was im Vergleich zum mühsamen Eintippen eine rasante Beschleunigung lieferte (vgl. Memmert & Raabe, 2017, S. 33). Zusätzlich erwähnt Lames (1994, S. 131) auch die computergestützte Spracherkennung, welche ebenfalls zur Erleichterung der Datenerfassung beitragen sollte. Diese Spracherkennungssysteme kamen schlussendlich auch auf den Markt und ermöglichten eine noch schnellere Eingabe der Daten. In vordefiniertem Code wurde in ein Mikrofon gesprochen und zusätzlich auf einem Bildschirm der Ort der Aktion auf einem virtuellen Spielfeld eingegeben (vgl. Memmert & Raabe, 2017, S. 33).

Die Einbindung verschiedenster Videoquellen war eine zusätzliche Voraussetzung für die Datenerfassung. Anfangs noch über VHS-Rekorder, wurde die Videoanalyse auch aufgrund der immer schneller arbeitenden Prozessoren vollständig auf den Computer verlagert (vgl. Memmert & Raabe, 2017).

2.1.1 Positionstracking

Wie schon in der Übersicht vorweggenommen, ist die Spielanalyse rein anhand von Videodaten nicht der technologische Schlusspunkt. Immer häufiger wird auf Positionsdaten zurückgegriffen, welche nicht nur aufgrund des Einsatzes modernster technologischer Geräte, sondern auch aufgrund neuester Methoden der Bildverarbeitung erfasst werden können. Nicht nur die Position des Balles ist dabei von Interesse, auch die Standorte der 22 Akteure am Feld werden über den gesamten Spielverlauf mitverfolgt (vgl. Memmert & Raabe, 2017).

Baca (in Memmert & Raabe, 2017) unterscheidet zwischen verschiedenen Systemen der Positionserfassung. Er nennt dabei die auf GPS basierenden Systeme, die videobasierenden Systeme unter Verwendung von Methoden der Bildverarbeitung und als Drittes radar- bzw. mikrowellen- basierende Systeme.

Durch diese Positionsdatenerfassung entsteht eine sehr große Menge an zusätzlichen Daten. Kratz (2017, S. 5) bestätigt diese Behauptung und erwähnt, dass während der neunzig Minuten eines Bundesligaspiels bis zu sechzig Millionen Daten gesammelt werden.

Fast alle Profiklubs nützen das auf Navigationssatellitensystemen basierende GPS-Tracking auch in ihrem Trainingsalltag. Diese Methode der Leistungs-überwachung wird über einen Transponder erfasst, welcher meist an einem Brustgurt angebracht ist und von den Spielern getragen wird (vgl. Memmert & Raabe, 2017).

Bei den videobasierten Systemen sind an den Spielern keine elektronischen Hilfsmittel notwendig. Durch moderne Verfahren der Bildverarbeitung werden die Spieler im Kamerabild identifiziert und deren Laufwege zudem über verschiedene Perspektiven nachvollzogen. Gewisse, unübersichtliche Situationen können dabei jedoch die Erkennung der Spieler erschweren, was einen Nachteil der videobasierten Systeme darstellt (vgl. Memmert & Raabe, 2017).

Für die radar- bzw. mikrowellenbasierenden Systeme werden wie beim GPS-System Transponder benötigt, um eine Ortung durchzuführen. Rund um das Spielfeld befinden sich fest installierte Empfängereinheiten, welche regelmäßig Kontakt mit den Transpondern an den Athleten aufnehmen. Durch die gleichmäßige Ausbreitung elektromagnetischer Wellen können so die genauen Positionen berechnet und mit kaum merklicher Verzögerung direkt an Analysten weitergegeben werden (vgl. Memmert & Raabe, 2017, S. 67).

Obwohl man sich von der Realisierung einer Echtzeit-Positionsdatenerfassung große Fortschritte versprechen kann, merkt Lames (2008, S. 306) an, dass die wesentlichen Aufgaben des Coachings alleine damit noch kaum Unterstützung erfahren. Sie helfen nicht dabei, Stärken und Schwächen der Spieler zu identifizieren und liefern ebenso keine Anregungen für eine Spielstrategie. Lames (2008, S. 306) betont, dass Aussagen auf abstrakteren Ebenen als lediglich der Positionsebene benötigt werden.

Abbildung 2: Ebenen einer computergestützten Spielanalyse (Lames, 2008, S. 306)

Wie in der Abbildung 2 ersichtlich, sollen zunächst die Aktionen auf dem Platz erfasst werden. Wichtige Analysekonzepte wie Ballbesitz, Pass, Dribbling oder Torschuss können nach Lames (2008, S. 306) Gegenstand von Aktionsprofilen werden. Anhand der Aktionen auf dem Feld kann man dann im Weiteren auf Situationen zurückschließen.

Lames (2008, S. 306) versteht darunter taktische Konfigurationen wie z.B. Konter, Positionsspiel, Wandpass oder Gegenangriff. Auf einem noch höheren Abstraktionsniveau wird die Taktik analysiert. Man kann hierbei auch Analysen durchführen, die das Ziel haben, das taktische System des Gegners in Offensive und Defensive zu durchschauen, was Lames (2008, S. 306) als einen wertvollen Beitrag zum Coaching sieht.

Zuoberst findet sich die Strategieebene, die direkt auf den für den Coach relevanten Sachverhalten operiert. Als Beispiele für diese Abstraktionsebene nennt Lames (2008, S. 306) die Stärken und Schwächen der Spieler in physischer, technischer und taktischer Hinsicht, bevorzugte Aktionsweisen von Spielern und Mannschaften, die Übereinstimmung zwischen Spielstrategie und gezeigtem Verhalten des eigenen und des gegnerischen Teams und Änderungen in der Taktik nach kritischen Ereignissen im Spiel wie z.B. Herausstellungen, Toren oder Auswechslungen.

Nach diesem Einblick in die Techniken der Wettkampfanalyse wird speziell auf jenen Ansatz eingegangen, welcher im Forschungsprozess dieser Arbeit verwendet wurde.

2.2 Machine Learning im Sportspiel

Machine Learning wird in den verschiedensten Bereichen der Wissenschaft eingesetzt (siehe Kap. 1.1.1). Auch in der Sportspielanalyse gibt es bereits einige Ansätze, welche mittels Machine Learning realisiert werden.

Kumar (2013, S. 1) stellt fest, dass Machine Learning in Sportarten wie Baseball oder Basketball bereits angewandt wird. Er merkt auch an, dass die Verwendung im Fußball noch limitiert ist und es herauszufinden gilt, ob Machine Learning auch dort geeignet ist, um aufschlussreiche Resultate zu erzielen. Er beschäftigt sich in seiner Forschungsarbeit mit Spielerratings und versucht Kriterien zu finden, anhand welcher Experten die Athleten bewerten.

Die Bewertungen der einzelnen Spieler werden von ihm zu einem Teamrating zusammengefasst (vgl. Kap 3.1.2) und anhand dieser das entsprechende Spielergebnis vorhergesagt. Eine Prognosegenauigkeit von 90% lässt ihn darauf schließen, dass die Leistungsbewertungen sehr stark vom Ergebnis abhängen. Die Richtigkeit einer Vorhersage für das folgende Spiel eines Teams lag dagegen bei nur 53%. Diesen Wert führt der Autor vor allem auf eine Unvorhersehbarkeit eines Unentschiedens zurück (vgl. Kumar, 2013).

Auch zu erwähnen ist der Versuch von Liti et. al (2017, S. 229) die Ergebnisse von Fußballspielen vorherzusagen. Konkret wurden zwei Halbsaisonen der italienischen Serie A (Frühjahr 2014-15 & Herbst 2015-16) analysiert, wobei speziell die Spiele verwendet wurden, welche zur Halbzeit einen ausgeglichenen Spielstand aufwiesen. Anhand statistischer Informationen aus der ersten Halbzeit, sollte der Computer voraussagen, wie das Spiel zu Ende gehen würde. Liti et. al (2017, S. 233) verwendeten dafür einige Klassifizierungsalgorithmen, welche auch in dieser Arbeit ihre Anwendung finden.

Einen weiteren Ansatz des maschinellen Lernens stellen sogenannte Bayes‘sche Netze dar. Joseph et al. (2006) haben ein solches Bayes’sches Netz verwendet und mit anderen Techniken des Machine Learnings verglichen. Dieses Netzwerk wurde nur anhand weniger Merkmale trainiert und obwohl die Autoren dies als ein sehr vereinfachtes Modell des gesamten Spiels sehen, hat es beeindruckende Ergebnisse geliefert (vgl. Joseph et al., 2006, S. 9).

Auch Constantinou et al. (2012, S. 322) bestätigen die Vorhersagegenauigkeit von Bayes’schen Netzwerken und behaupten dadurch auch Buchmacher und deren Quoten damit „besiegen“ zu können.

Nazim et al. (2017, S. 6) haben für die Vorhersage von Ausgängen der englischen Premier League mithilfe von Bayes'schen Netzwerken eine Prognosegenauigkeit von durchschnittlich über 75% erreicht. Die Autoren erhoffen sich, dadurch einen Maßstab für die zukünftige Forschung in diesem Bereich gesetzt zu haben.

Ob dies der Fall ist, wird man in naher Zukunft beobachten können. Zweifelsohne wird dagegen das nachfolgende Kapitel bestätigen, dass der Fokus dieser Arbeit nicht direkt auf die Prognose von Spielergebnissen gerichtet ist.

2.3 Ziel dieser Arbeit

Ziel dieser Arbeit ist es, mithilfe einer Vielzahl an Modellen des Machine Learnings, Tor-Events in einem Fußballspiel vorhersagen. Ein Computer, welcher mit Hilfe verschiedener Algorithmen einen Datensatz erlernt, soll also anhand der ihm überlieferten Informationen vorhersagen, ob es wahrscheinlich ist, dass in weiterer Folge ein Schuss auf das Tor oder ein Torerfolg passiert. Die vom Computer getroffene Prognose soll rein auf den zur Verfügung stehenden Eventinformationen desselben Spiels basieren. Diese werden dabei von der Maschine dynamisch eingelesen und in weiterer Folge in Form von verschiedenen Modellen antrainiert. Im nächsten Kapitel wird darauf eingegangen, wie die Daten im Laufe des Forschungsprozesses aufbereitet wurden, um eine Prognose mittels maschinellen Lernens (ML) durchzuführen.

3 METHODE

3.1 Datenmaterial

Das Datenmaterial, welches in dieser Arbeit behandelt wird, wurde von der Firma Sportradar zur Verfügung gestellt. Sportradar ist eine der größten internationalen Firmen der Sportdatenerfassung. Auf der Website dieser Firma wird angegeben, dass über 1900 Mitarbeiter an über 30 Standorten weltweit für das Unternehmen arbeiten (abgerufen am 20.4.2018 von sportradar.com). Konkret wurden Fußballdaten der aktuellen Saison (2017/18) aus den fünf Top-Ligen in Europa (Deutschland, England, Frankreich, Italien, Spanien) via API Sandbox vom Unternehmen bereitgestellt. Jedes Spiel war darin als eigene XML-Datei vorhanden und deren Eventinformationen sind in diesen Dokumenten anhand einer „Timeline“ chronologisch gelistet. Die Daten wurden manuell erfasst und nach einem einheitlichen System erhoben. Insgesamt wurden die Daten von 1123 Fußballspielen mithilfe der Programmiersprache Python ausgewertet und analysiert. Die Informationen zu den Spielen basieren wie schon erwähnt auf reinen Eventinfos, welche wie folgt dokumentiert wurden.

3.1.1 Events

Die nachfolgende Tabelle 1 enthält die relevanten Events, welche von Sportradar für alle Matches dokumentiert wurden.

Tabelle 1: Eventbeschreibung

corner_kick	**Eckball**
free_kick	Freistoß
goal_kick	Abstoß
red_card	Platzverweis
offside	Abseitsposition
throw_in	Einwurf
shot_off_target	Schuss vorbei am Tor
shot_on_target	Schuss auf das Tor
score_change	Torerfolg

3.1.2 Kodierung

Jedem Event wurde eine Zahl zugeordnet, welche bei Auftreten dieses Events dem Computer übermittelt wurde. Zusätzlich zu der Information, welchen Typ ein erfasstes Event besitzt, sind sowohl das ausführende Team sowie zugehörige Koordinaten (x, y) gegeben, welche die jeweils aktuelle Position am Spielfeld beschreiben. Diese zusätzlichen Informationen wurden unter anderem für die Kodierung der Freistöße genutzt. Link et. al (2016) untersuchten im Artikel zur Topographie von Freistößen im Fußball jene Freistöße, welche eine Distanz von weniger als 35 Metern zum gegnerischen Tor aufwiesen. Auch bei der Datenkodierung wurden die Freistöße in die Kategorien > bzw. < 35m aufgeteilt, da davon ausgegangen wird, dass für diese zwei Kategorien unterschiedliche Erfolgswahrscheinlichkeiten für einen daraus resultierenden Torschuss beziehungsweise einen Torerfolg vorliegen. Des Weiteren gab es zusätzlich zu den Events die Information eines Ratings für alle teilnehmenden Spieler, welches deren Leistung mit einem Wert von 0-10 bewertete.

Diese Spielerbewertungen wurden ebenso in die Datenanalyse eingebaut. Um jedoch nicht aus dieser Information Vorhersagen über das aktuelle Spiel zu verfälschen, wurden die vergangenen zwei Spiele eines Teams herangezogen und der Mittelwert

$$\overline{x} := \frac{1}{n}(x_1 + x_2 + \ldots + x_n) = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Formel 1: Arithmetisches Mittel

der einzelnen Spielerratings berechnet. So wurde aus den Bewertungen der Spieler ein Teamrating errechnet, welches eine Art „Formfaktor“ aus den zwei vergangenen Matches als Zusatzinformation liefern soll.

3.2 Programmiersprache

Um die Forschung durchzuführen, wurde wie bereits kurz erwähnt die Programmiersprache Python benutzt. Es wurde mit der Version 3.6.4 gearbeitet. Auf den Aufbau der Programmiersprache wird hier allerdings nicht weiter eingegangen. Ein kurzer Einblick in zwei wichtige Python-Bibliotheken, welche im Forschungsprozess verwendet wurden, soll jedoch nicht vorenthalten werden.

3.2.1 Pandas

Bei Pandas handelt es sich in diesem Fall nicht um Tiere, sondern um eine Python-Bibliothek. Aufgebaut ist dieses Modul um eine Datenstruktur namens DataFrame. Vereinfacht gesagt, handelt es sich bei einem Pandas DataFrame um eine Tabelle, welche einem Excel-Tabellenblatt nicht unähnlich ist (vgl. Heydt, 2015, S. 127). Die Bibliothek bietet eine große Bandbreite zum Modifizieren und Verarbeiten dieser Tabellen (vgl. Guido & Müller, 2017, S. 37). Im Verlauf der Forschungsarbeit war unter anderem die Verfügbarkeit von Einleseprozeduren für kommaseparierte Dateien (CSV) mithilfe von Pandas wichtig, da die zur Verfügung gestellten Daten als solche in Python umkodiert wurden. Zusätzlich wurden mithilfe des Pandas-Moduls die vorhin erwähnten Teamratings errechnet.

3.2.2 Scikit-Learn

Scikit-Learn ist nach Grus (2016, S. 315) die vermutlich beliebteste und verständlichste Open-Source-Bibliothek für Machine Learning. Das Modul verfügt nach Pedregosa et. al (2011) über eine große Breite der modernsten Algorithmen für Machine Learning und ist auf der Homepage *http://scikit-learn.sourceforge.net* frei verfügbar. In weiterer Folge wird nun auf die angesprochenen Lernalgorithmen eingegangen, welche auch als Klassifizierer bezeichnet werden.

3.3 Klassifizierer

Die Problemstellung der Arbeit wurde mit der Methode des überwachten Lernens bearbeitet. Es handelt sich dabei um Klassifizierungsprobleme, bei welchen Beobachtungen bestimmten Gruppen zugeordnet werden (vgl. Kap. 1.1.1).

In Abbildung 3 werden vier verschiedene Lernalgorithmen dargestellt, welche verschiedene Möglichkeiten zur Klassifikation von Daten aufzeigen.

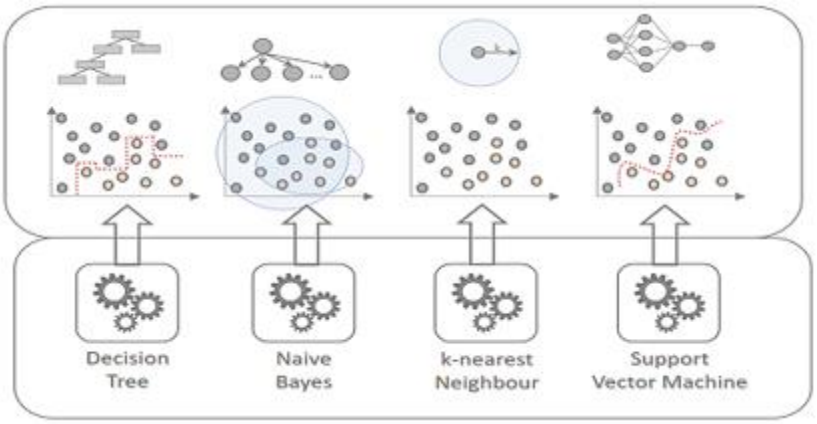

Abbildung 3: Klassifizierungsalgorithmen (*https://data-science-blog.com/blog/2017/12/03/ensemble-learning/)*

Jeder dieser Lernalgorithmen besitzt gewisse Eigenheiten und geht von bestimmten Annahmen aus. Da es keinen Klassifizierer gibt, der für alle denkbaren Szenarien geeignet wäre, empfehlen Raschka & Mirjalili (2018, S. 73) für die Praxis, die Leistung verschiedener Lernalgorithmen miteinander zu vergleichen und das beste Modell auszuwählen.

Nachfolgend werden die Lernalgorithmen, welche im Laufe des Arbeitsflusses für die Klassifizierungsprobleme verwendet wurden, genauer beschrieben.

3.3.1 Naive Bayes

Dieser Algorithmus baut auf ein Theorem der Wahrscheinlichkeitsrechnung auf, welches bereits im 18. Jahrhundert von Thomas Bayes formuliert wurde (vgl. Mueller & Massaron, 2016, S. 329).

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Formel 2: Bayes Theorem

Beim Bestimmen der Wahrscheinlichkeit eines Ereignisses neigt man zum Glauben, dass dies in jeder Situation angewandt werden kann. Die Anfangswahrscheinlichkeit kann sich jedoch anhand von Aussagen ändern. Als Beispiel nennen Mueller & Massaron (vgl. 2016, S. 330) die Bestimmung der Wahrscheinlichkeit, dass eine Person eine Frau ist. Man könnte meinen, dass die Anfangswahrscheinlichkeit bei ca. 50% liegt. Wäre die Aussage, ob eine Person weiblich oder männlich ist, jedoch von beispielsweise der Haarlänge der Person abhängig, würde sich die Anfangswahrscheinlichkeit drastisch ändern. Mit Hilfe der Bayes-Regel kann man diese sogenannte bedingte Wahrscheinlichkeit ausrechnen.

Der Naive Bayes-Klassifizierer nutzt diese, um die Anfangswahrscheinlichkeit einer Vorhersage zu ändern. Er profitiert dabei von allen verfügbaren Aussagen der Daten und berechnet mithilfe dieser Formel seine Prognosen. Trotz der simplen Annahmen erwähnen Mueller & Massaron (2016, S. 331), dass viele Studien über gute Leistungen dieses Klassifizierers berichten.

Zhang (2004, S. 2-3) versucht eine Erklärung dafür zu finden, warum dieser Algorithmus so gute Ergebnisse liefert. Er vermutet, dass trotz der starken Abhängigkeiten durch das Bayes-Theorem, welche den grundsätzlichen Annahmen unabhängiger Variablen widersprechen, diese sich auf den gesamten Datensatz jedoch mehr oder weniger ausgleichen und somit gegenseitig aufgehoben werden.

Im Python Modul Scikit-Learn werden drei verschiedene Naive Bayes-Klassen mitgeliefert. Jeder dieser Ansätze eignet sich unterschiedlich gut für verschiedene Verteilungen der Daten. Der GaussianNB-Klassifizierer geht beispielsweise von einer Normalverteilung aller Merkmale aus (vgl. Mueller & Massaron, 2016).

3.3.2 Nearest Neighbours

Dieses ist eines der einfachsten Vorhersagemodelle und stellt keine mathematischen Annahmen auf (vgl. Grus, 2017, S. 161). Ein Maß für die Distanz und die Annahme, dass näher beieinanderliegende Punkte ähnlich sind, sind die einzigen zwei Voraussetzungen dafür. Das „Nächste-Nachbarn"-Modell wird vermutlich keine Hilfe sein, um Ursachen eines betrachteten Phänomens zu verstehen. Wenn man als Beispiel das Stimmverhalten bei einer Wahl nehmen würde, würde das Modell dieses rein anhand des Stimmverhaltens der Nachbarn vorhersagen und nicht anhand von alternativen Informationen wie Einkommen oder Familienstand (vgl. Grus, 2017, S. 162).

Ein Nachteil, welcher durch den Nearest Neighbours-Klassifizierer auftreten kann, ist das sogenannte „Overfitting". Wenn beispielsweise K=1 und somit immer nur der nächste Nachbar betrachtet wird, lernt der Algorithmus bei vielen Daten oft Zusammenhänge, welche nur Zufall oder das Rauschen der Daten (auch „noise" genannt) sein können. Eine Balance zwischen Generalisierung und sturem Auswendiglernen ist ein häufiges Problem bei vielen Algorithmen. Um eine solche Überanpassung zu reduzieren, kann an vielen Modellen eine Regularisierung verwendet werden (vgl. Hackeling, S. 20).

3.3.3 Logistic Regression

Die lineare Regression hat eine lange Geschichte in der Statistik und ist für die Abschätzung von Werten grundsätzlich gut geeignet. Für die Vorhersage von Klassen einer Beobachtung ist das Ergebnis jedoch meist enttäuschend. Um korrekt zu klassifizieren wird ein besser angepasstes Maß als ein Kontinuum von numerischen Schätzungen benötigt. Diese numerischen Schätzungen einer linearen Regression können allerdings in eine Wahrscheinlichkeit transformiert werden, welche eine Klassifizierung besser beschreiben kann (vgl. Mueller & Massaron, 2016, S. 327).

Bei diesem Verfahren wird die logistische Funktion verwendet, welche in Abbildung 4 dargestellt wird:

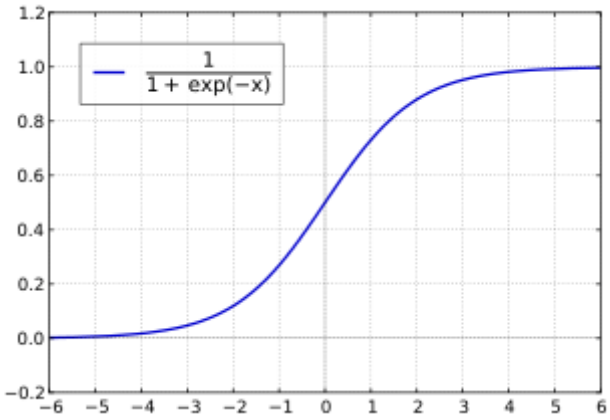

Abbildung 4: Logistische Funktion (*http://statistik-dresden.de/archives/629)*

Wie man anhand des Graphen der Funktion sehen kann, bedeutet ein positiver Eingabewert einen Funktionswert > 0.5. Je größer dieser Eingabewert wird, desto näher rückt der Funktionswert an 1 heran. Analog gilt dies auch für einen negativen Eingabewert – je niedriger dieser ist, desto näher rückt der Funktionswert an 0 heran. Dies und auch die Eigenschaft der Ableitungsfunktion macht sich dieses Modell zu Nutze (vgl. Grus, 2017, S. 204). Ein Vorteil der logistischen Regression ist es, dass die Implementierung durch die relative Einfachheit des Modells leichtfällt. Durch die Maximierung der bedingten Wahrscheinlichkeiten der Trainingsdaten ist dieses Modell jedoch anfällig für Ausreißer in den Daten (vgl. Raschka & Mirjalili, 2018, S. 100).

3.3.4 Support Vector Machines

Eine Support Vector Machine, auch kurz SVM genannt, liefert bei praktischen Klassifizierungsaufgaben häufig sehr ähnliche Ergebnisse wie das Modell der logistischen Regression (vgl. Raschka & Mirjalili, 2018). Bei einer SVM sind vor allem die Punkte in der Nähe der Entscheidungsgrenze ausschlaggebend. Dies wird auch in Abbildung 5 ersichtlich gemacht.

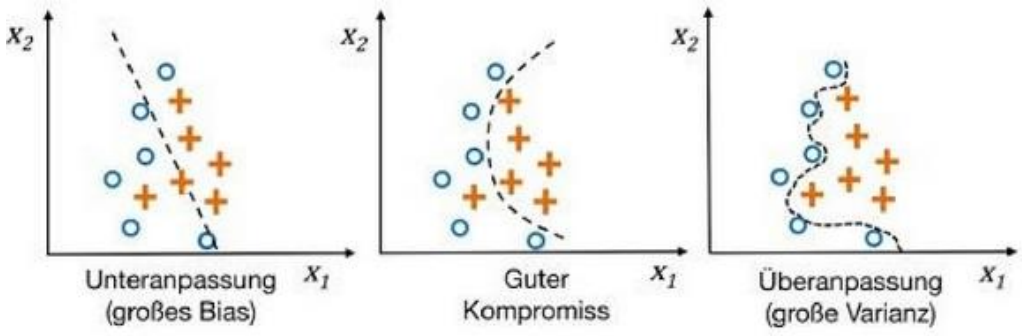

Abbildung 5: Entscheidungsgrenze (Raschka & Mirjalili, 2018, S. 94)

Wie die Grafik zeigt, kann die Entscheidungsgrenze einer SVM verschiedene Formen annehmen. Mueller & Massaron (2016, S. 370) bezeichnen SVM als eine komplexe maschinelle Lerntechnik. SVMs sind auch in höherdimensionalen Räumen noch effektiv. Ein wichtiger Parameter einer SVM ist ihr Kernel. Raschka (2016) meint, dass man im Normalfall zu Beginn der Forschungsarbeit wenig über den Datensatz selbst Bescheid weiß und daher auch nicht einschätzen kann, welcher Kernel sich am besten für das zu bearbeitende Problem eignet. Er empfiehlt für die Praxis mit dem linearen Ansatz zu beginnen und stückweise aufwärts in komplexere, höherdimensionale Entscheidungsräume vorzudringen (vgl. Raschka, 2016).

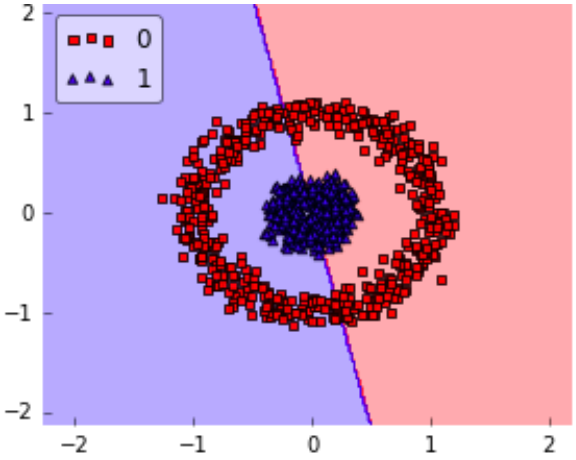

Abbildung 6: Unpassende Anwendung eines linearen Kerns (Raschka, 2016)

Wie Abbildung 6 verdeutlicht, kann ein linearer Kern auch die falsche Wahl für ein Klassifizierungsproblem darstellen. Ein RBF-Kernel wäre in diesem Fall besser geeignet. Raschka (2016) erklärt, dass dieser nichtlineare Kombinationen der Datenmerkmale erzeugt, sodass die ursprünglichen Merkmale auf einen höherdimensionalen Raum abgebildet werden (Abb. 7).

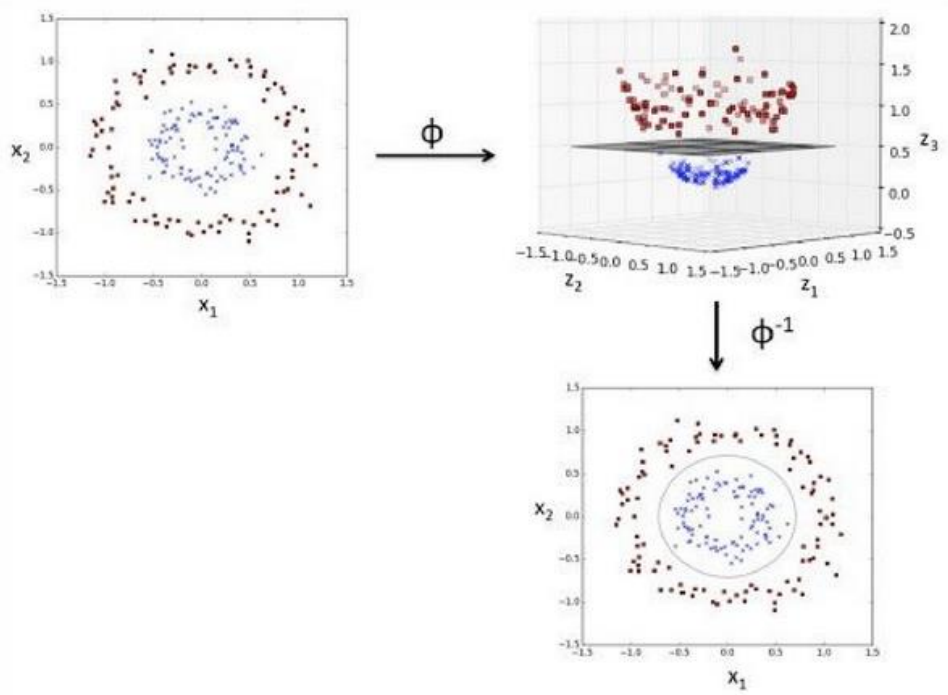

Abbildung 7: Kernel-Trick (Raschka & Marjalili, 2018, S. 103)

3.3.5 Neuronale Netze

Grus (2017, S. 225) vergleicht die Arbeitsweise eines künstlichen neuronalen Netzwerks mit der eines Gehirns. Er versteht darunter eine Ansammlung verdrahteter Neuronen, welche als Eingabe die jeweilige Ausgabe anderer Neuronen erhalten. Ein Perzeptron, bei dem ein einzelnes Neuron mit n binären Eingaben nachgebildet wird, ist die einfachste Form eines neuronalen Netzwerks. Wenn die gewichtete Summe der Eingaben größer oder gleich null ist, „feuert" das Perzeptron (vgl. Grus, 2017).

Für Aufgaben, welche für ein einzelnes Perzeptron nicht mehr lösbar sind, müssen komplexere neuronale Netzwerke angewandt werden. Ein Feed-forward-Netz besteht aus klar abgegrenzten Schichten von Neuronen, von denen jede mit der nächsten verbunden ist. Das Netzwerk besteht aus einer Eingabeschicht, einer oder mehreren verborgenen Schichten und einer Ausgabeschicht, welche in der Abbildung 8 von links nach rechts dargestellt werden.

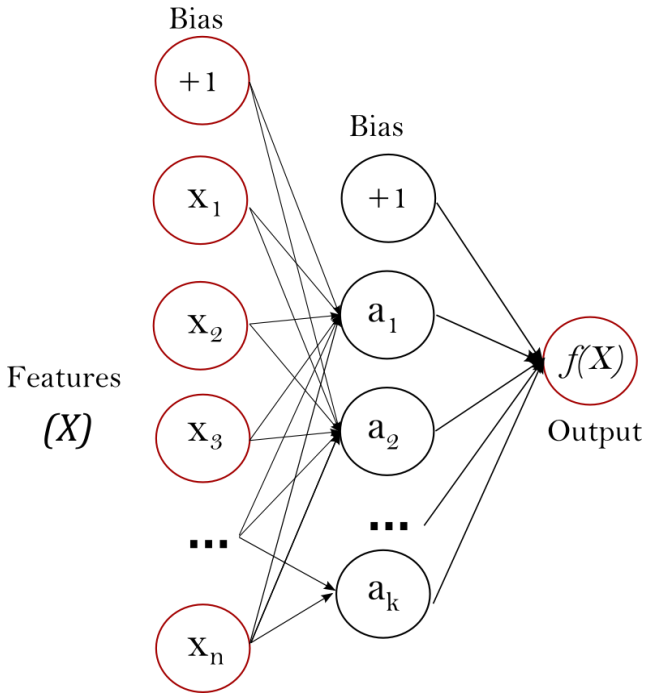

Abbildung 8: Neuronales Netzwerk

(http://scikit-learn.org/stable/modules/neural_networks_supervised.html)

Zusätzlich wird jedem Nicht-Eingabe-Neuron ein Bias-Wert und eine Gewichtung angehängt (vgl. Grus, 2017). Ein Neuron kann also einfach als Liste von Gewichten ausgedrückt werden. Das gesamte neuronale Netz lässt sich somit als Liste von Schichten beschreiben, wobei jede Schicht wiederum eine Liste der enthaltenen Neuronen bildet (Grus, 2017, S. 228).

3.3.6 Random Forest

Die Grundstrukturen dieses Klassifizierers bilden Entscheidungsbäume. Diese verwenden eine Baumstruktur, um mögliche Entscheidungspfade und ein Ergebnis für jeden Pfad zu speichern (Grus, 2017, S. 213).

Die Idee dabei ist es, den Datensatz durch spezifische Regeln, basierend auf den Werten seiner Merkmale, in kleinere Teile aufzuspalten (Mueller & Massaron, 2016, S. 388). Vorteile von Entscheidungsbäume sind unter anderem, dass sie leicht zu verstehen und zu interpretieren sind. Ein Beispiel dafür soll die Abbildung 9 liefern.

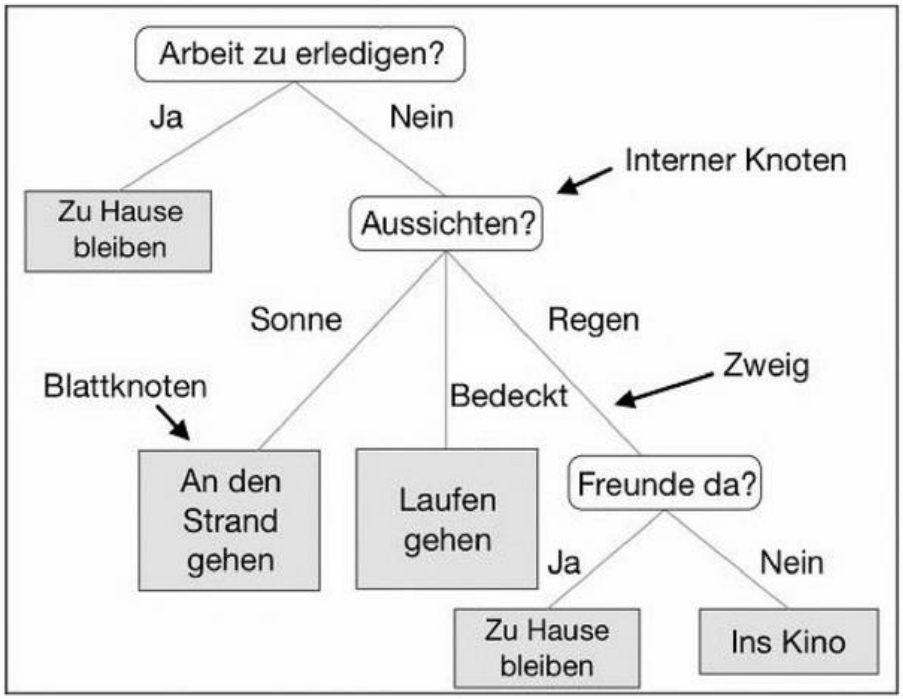

Abbildung 9: Beispiel eines Entscheidungsbaumes (Raschka & Mirjalili, 2018, S. 107)

Ein Entscheidungsbaum alleine bringt jedoch den Nachteil mit sich, dass er wie auch andere Klassifizierungsalgorithmen zu einer Überanpassung neigt. Der Random Forest besteht somit aus mehreren Entscheidungsbäumen, welche schlussendlich zusammen über die Klassifikation abstimmen. Raschka & Mirjalili (2018, S. 116) erwähnen auch, dass man sich einen Random Forest als ein Ensemble von Entscheidungsbäumen vorstellen kann. Mehrere Entscheidungsbäume, welche meist große Varianz aufweisen, werden zu einem stabileren Modell kombiniert. Dieses ist besser für die Verallgemeinerung geeignet und weniger anfällig für Überanpassung.

Durch die sogenannte „Bootstrapping"-Methode werden die Entscheidungsbäume mit unterschiedlichen Daten konstruiert, was einen zusätzlichen Vorteil beim Testen außerhalb der Stichprobe mit sich bringt (vgl. Grus, 2017, S. 223).

Random Forests haben nach Raschka & Mirjalili (2018, S. 116) im vergangenen Jahrzehnt bei Anwendungen des Machine Learnings aufgrund ihrer guten Klassifizierungsfähigkeit, Skalierbarkeit und ihrer einfachen Handhabung sehr an Popularität gewonnen.

3.4 Arbeitsfluss

Im Laufe des Forschungsprozesses wurden die Daten auf verschiedene Arten aufbereitet. In weiterer Folge wurden die in Kapitel 3.3 erwähnten Algorithmen darauf angewandt. Die nächsten Kapitel beschreiben, wie die Daten zerlegt wurden, damit sie im Python-Modul scikit-learn sinnvoll verwendet werden konnten.

3.4.1 Kumulierte Daten

In dieser Variante, bei welcher die Daten unter anderem als Voraussetzung für den „Multinomialen Naive Bayes“- Klassifizierer kumuliert aufbereitet wurden, wurde eine bestimmte Anzahl an Events festgelegt, welche als Ereignisketten ausgegeben wurden. Die Länge dieser Ereignisketten bestand aus zwischen fünf und fünfzehn solcher Events. Alle Events außer die Zielevents „shot_on_target“ bzw. „score_change“ wurden dabei berücksichtigt. Für jede Eventkette wurde eine zusätzliche, boolesche Variable definiert. Diese Variable hatte den Wert 0, falls kein Zielevent zwischen den anderen gezählten Events auftrat, und den Wert 1, falls sich zumindest ein Zielevent innerhalb der Eventkette ereignete. Für jede Kette an Events wurde zusätzlich noch eine Variable definiert, welche das Zeitintervall, in welchem die Events stattgefunden haben, als Zusatzinformation enthielt. Durch die unterschiedliche Länge dieser sogenannten Eventketten ergaben sich daraus logischerweise auch unterschiedlich große Datensätze. Bei der Untersuchung dieser Ketten mit je fünfzehn Ereignissen waren es n=7643 solcher Ketten, bei der Eventzusammenfassung zu je fünf Stück waren es n=22989. Die sogenannte „Null Accuracy“ war bei den Kettenlängen von 5 bzw. 15 jeweils sehr hoch (>65%). Das bedeutet, dass bei Kettenlänge 5 viel öfter das Ereignis „0“, also kein Torevent auftrat, jedoch bei einer Kettenlänge von 15 viel öfter das Ereignis „1“. Sollte der Computer also bei jeder Entscheidung einfach die häufiger vorkommende Klasse auswählen, würde er in diesem Fall schon eine Vorhersagegenauigkeit von über 65% erzielen. Es stellte sich in weiterer Folge heraus, dass eine Kettenlänge von neun Events die geringste „Null Accuracy“ mit ca. 50,8% aufwies (n=13077). Auf dieses Phänomen wird in Kapitel 4.1 noch einmal eingegangen.

3.4.2 Zerlegung der Daten in Zustände

Eine weitere Überlegung, wie die Daten zur Analyse aufbereitet werden konnten, war, diese in Zustände zu zerlegen. Es wurde dabei zwischen vier verschiedenen Zuständen unterschieden: Team A hat Ballbesitz, Team B hat Ballbesitz, Schuss auf das Tor und Torerfolg. Der Zusammenhang der Zustände wird durch folgendes Diagramm erläutert:

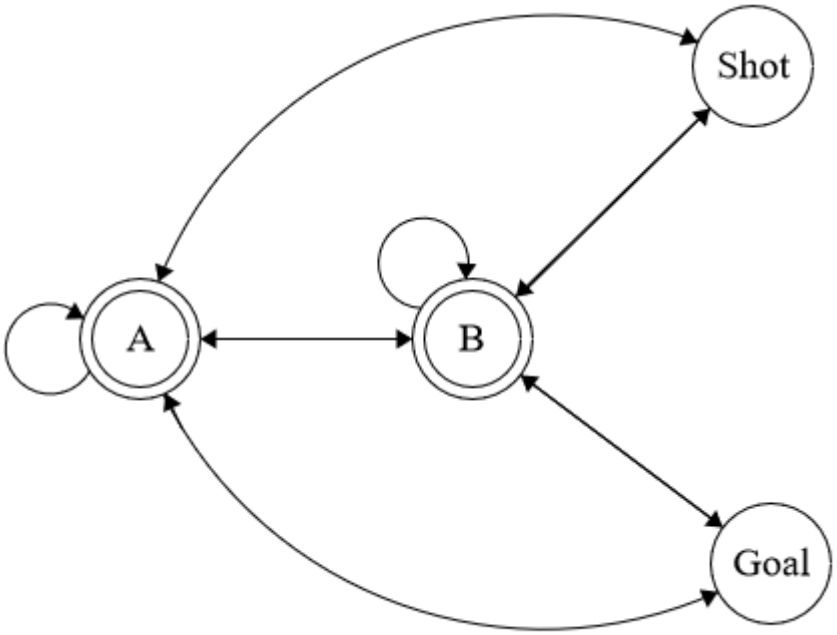

Abbildung 10: Diagramm der Zustände und deren möglichen Übergänge

Jedes einzelne Event eines Spiels wurde in der Folge als einer dieser vier Zustände aufgefasst und verfügte über einen Vektor mit zusätzlichen Informationen. Diese Informationen beinhalteten den aktuellen Spielstand, die aktuelle Spielzeit (in Minuten), ob im Laufe des Spiels ein Platzverweis stattgefunden hatte, die Spielstärke des Teams aus den vergangenen drei Spielen, x/y-Koordinaten des Balles, welches Event gerade stattfand und das aktuelle Team, welches dieses Event ausführte. Aus diesen acht Informationen sollte der Zustand für das nächste Event vorhergesagt werden. Es handelt sich dabei also um ein Klassifikationsproblem mit vier Klassen. Insgesamt wurden n=27726 Zustände für die Spiele der aktuellen Serie A-Saison analysiert.

3.4.4 Evaluierung & Optimierung der Algorithmen

Eine der gängigsten Methoden, um Ergebnisse im Machine Learning zu verifizieren, ist die Kreuzvalidierung. Bei der Kreuzvalidierung werden die Resultate einer maschinellen Lernhypothese getestet. Dabei werden die Trainings- und Testdaten auf verschiedene Arten zusammengesetzt (vgl. Mueller & Massaron, 2016, S. 252).

Mueller & Massaron (2016, S. 345) ergänzen: „Wenn Testsätze aufgrund der Stichprobenauswahl instabile Ergebnisse liefern, lautet die Lösung, eine bestimmte Anzahl an Testsätzen systematisch zu testen und dann den Durchschnitt der Ergebnisse zu bilden.“ Dabei handelt es sich um einen statistischen Ansatz, welcher auch die Basis der Kreuzvalidierung bildet.

Eine Kreuzvalidierung wird folgendermaßen durchgeführt: Die Daten werden gleichmäßig in Teilmengen („folds“) getrennt, welche jeweils einmal als Testdatensatz verwendet werden. Der restliche Datensatz dient wiederum zum Trainieren des Algorithmus. Dieser Prozess wird für jede Teilmenge am Datensatz ausgeführt und der Durchschnitt sowie die Standardabweichung der erreichten Teilmengentests werden dabei errechnet (vgl. Mueller & Massaron, 2016). Eine Kreuzvalidierung mit zehn Teilmengen wird in der 11 dargestellt.

Abbildung 11: Kreuzvalidierung (Bunker & Thabtah, 2017, S. 6)

Ein weiterer Aspekt, welchen es noch zu behandelt gibt, ist die Optimierung der Parameter. Das sogenannte Tuning kann mithilfe einer Rastersuche durchgeführt werden. Es handelt sich dabei um eine zeitaufwendige Technik, welche allerdings eine gute Möglichkeit bietet, eine maschinelle Lernanwendung zu optimieren (vgl. Mueller & Massaron, 2016).

3.4.5 Weiteres Vorgehen

Nach der Vorverarbeitung der Daten, welche nach Koehrsen (2017) meist 80% der investierten Zeit einer Datenanalyse in Anspruch nimmt, wurden die Lernalgorithmen an den vorbereiteten Datensätzen angewandt. In Abbildung 12 wird nochmals eine Übersicht über den gesamten Entwicklungsprozess eines Machine Learning Problems gegeben.

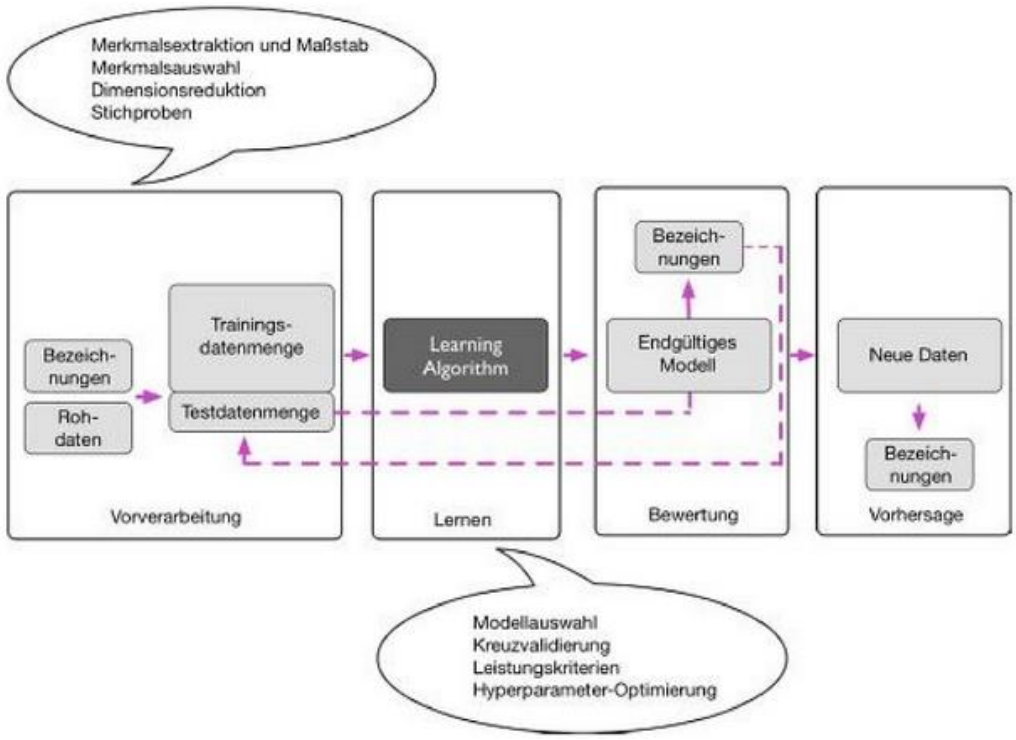

Abbildung 12: Entwicklung eines Systems im Machine Learning (Raschka & Mirjalili, 2018, S. 35)

4 ERGEBNISSE

In diesem Kapitel werden die Resultate der Klassifizierungsalgorithmen behandelt. Diese werden wie schon im vorherigen Kapitel nach Aufbereitungsmethode der Daten unterteilt.

4.1 Kumulierte Daten

Zuerst musste für den kumulierten Datensatz untersucht werden, welche Länge der sogenannten Eventketten sinnvoll für eine Untersuchung ist. Eine Ausgewogenheit der zu klassifizierenden Merkmale ist anzustreben, da sonst eine hohe „Null Accuracy“ gegeben ist. Dieses Phänomen wird anhand der verwendeten Daten illustriert. Die in Abbildung 13 angeführte „Confusion Matrix“ zeigt, wie ein Machine Learning-Algorithmus auf einen Datensatz mit hoher „Null Accuracy“ reagiert.

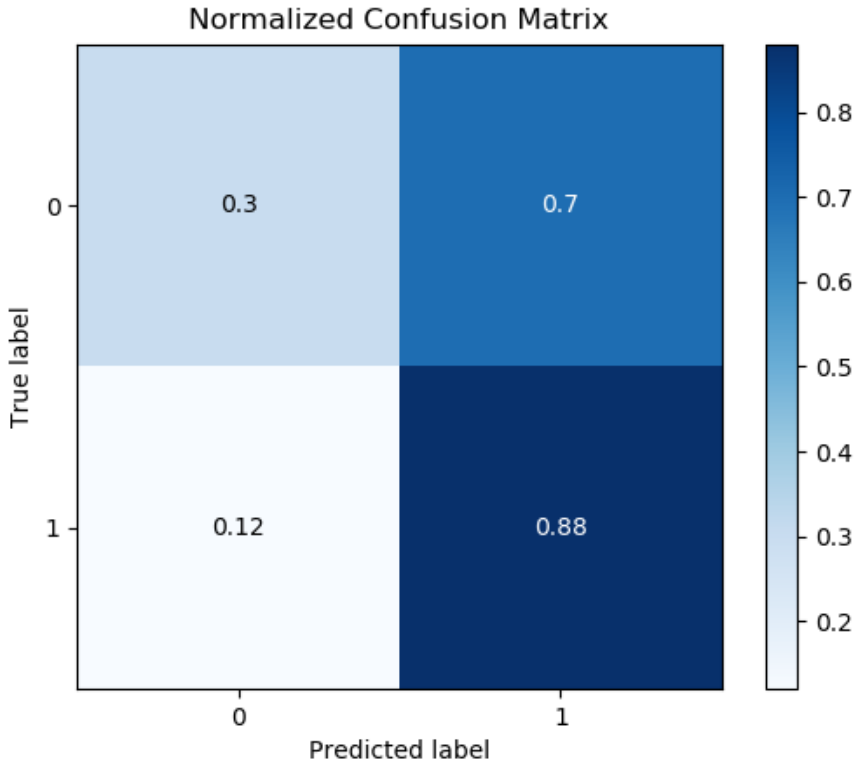

Abbildung 13: Wahrheitsmatrix (Confusion Matrix) für MNB mit Eventketten der Länge 15 (NA „1“ = 66.4%)

In diesem Fall war die zu vorhersagende Kategorie „1“ für ca. 2/3 des Datensatzes die richtige Vorhersage. Der Algorithmus wählte also sehr oft die Kategorie „1“ aufgrund des Ungleichgewichts im Datensatz. Die Kategorie „0“ wurde dagegen nur zu 30% richtig erkannt und in 70% der Fälle, in welchen Kategorie „0“ richtig gewesen wäre, wurde fälschlicherweise Kategorie „1“ vorhergesagt.

Als Gegenbeispiel wird in Abbildung 14 eine weitere Wahrheitsmatrix für eine andere Eventkettenlänge betrachtet.

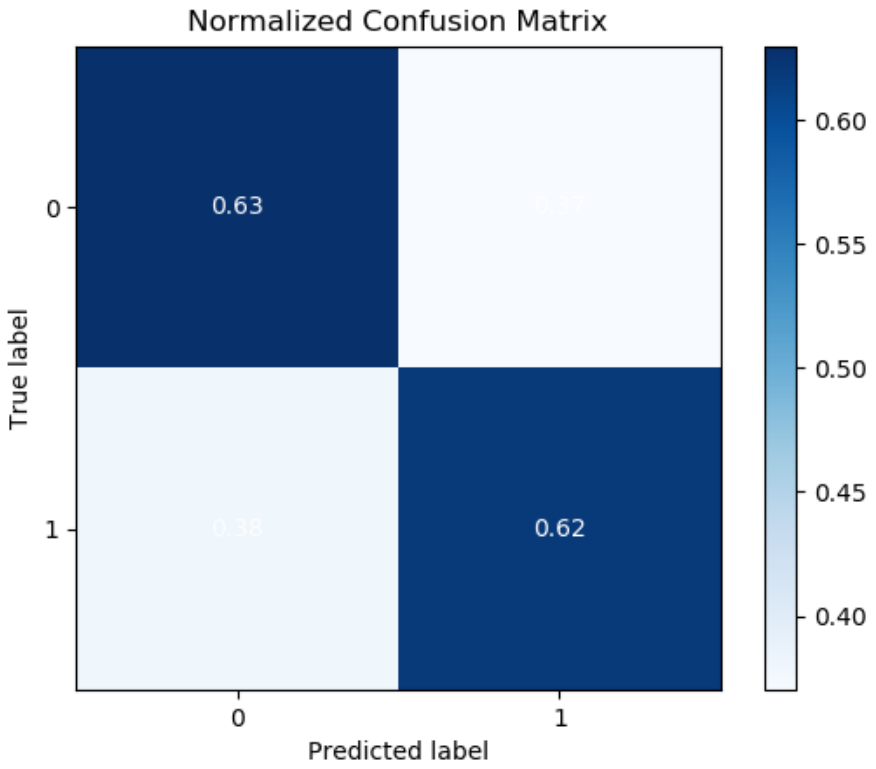

Abbildung 14: Wahrheitsmatrix für MNB bei Eventkettenlänge = 9 (NA=50,8%)

Im Vergleich zur vorigen „Confusion Matrix“ handelt es sich hier um eine Untersuchung, in der die beiden zu vorhersagenden Kategorien „0“ und „1“ nahezu gleich oft vorgekommen sind. Der Algorithmus kann hier also nicht mehr auf das häufigere Vorkommen einer Kategorie zurückgreifen und muss die Kategorie aus den Zusammenhängen der anderen Variablen erkennen. So war zwar die Vorhersagegenauigkeit niedriger, jedoch war die prozentuale Fehlklassifikation für eine der Kategorien ebenso geringer (vgl. Brownlee, 2017).

In der folgenden Abbildung wird die Prognosegenauigkeit für den „Nearest Neighbour“-Klassifizierer visualisiert. Für diesen Algorithmus wurden also zwischen einem und 33 „Nachbarn“ betrachtet, um schlussendlich Kategorie „0“ oder Kategorie „1“ zu prognostizieren.

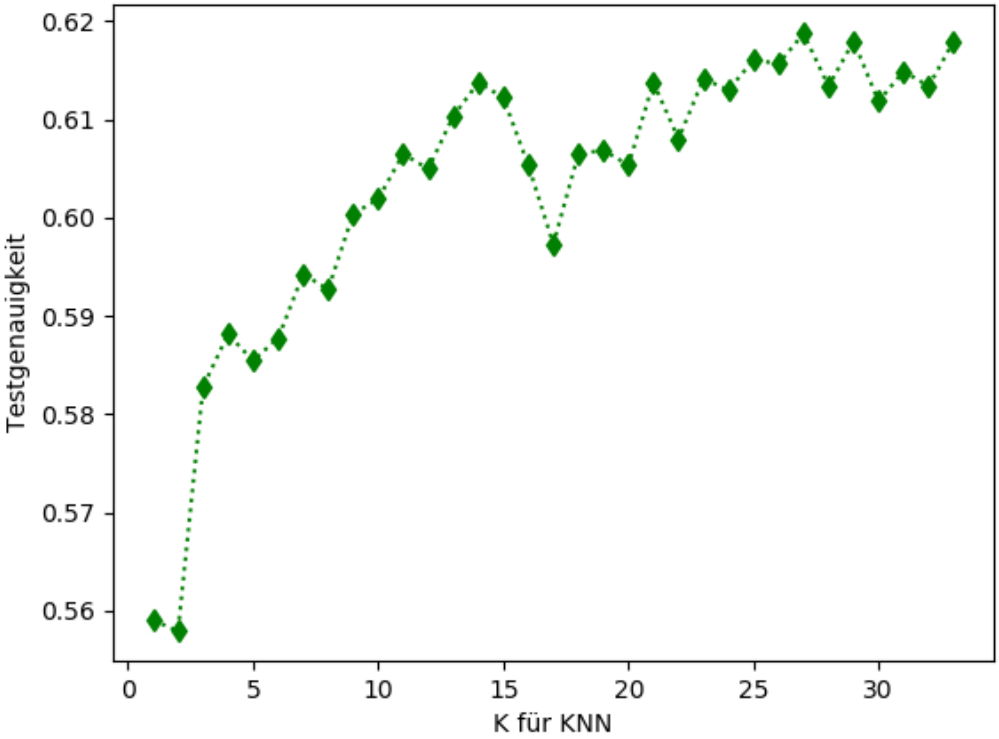

Abbildung 15: Prognosegenauigkeit für den "Nearest Neighbour“-Classifier

Wie man in der Abbildung erkennen kann, hat der Algorithmus mit K=27 die höchste Genauigkeit erreicht. Dieser lag bei knapp 0.62, was jedoch immer noch ein niedrigerer Wert ist, als andere Klassifizierer erreicht haben. Die Prognosegenauigkeiten für die weiteren Algorithmen werden in Tabelle 2 angeführt.

Tabelle 2: Accuracy Score (9 Events pro Kette)

Klassifizierer	TTS_1	TTS_2	TTS_3	TTS_mean
Multinomial NB	0.6321	0.6337	0.6386	0.6348
Neural Network	0.6363	0.6505	0.6359	0.6409
SVM (RBF Kernel)	0.6211	0.6328	0.6294	0.6278
SVM (Linear Kernel)	0.6325	0.6474	0.6413	0.6404
Random Forest Classifier	0.6107	0.6019	0.6069	0.6065
Logistic Regression	0.6421	0.6489	0.6417	0.6442

TTS steht hierbei für „Train Test Split“. Das bedeutet, der gesamte Datensatz wurde in Trainingsdaten und Testdaten aufgeteilt. Das Verhältnis dieser Unterteilung wurde mit 4:1 gewählt. 80% des Datensatzes wurden also zum Anlernen der Algorithmen und 20% des Datensatzes zur Testung deren Prognosegenauigkeit verwendet (vgl. Garreta & Moncecchi, 2013, S. 11).

Für diese drei Testläufe erzielte der Klassifizierer „Logistic Regression“ mit 64,42% die durchschnittlich höchste Prognosegenauigkeit. Dieses lineare Modell, welches trotz seines irreführenden Namens eher zur Klassifikation als zur Regressionsanalyse vorgesehen ist (vgl. Pedregrosa, 2011), hat das untersuchte Klassifizierungsproblem am besten gelöst, jedoch nicht mit einem signifikanten Unterschied zu den anderen verwendeten Lernalgorithmen.

4.2 Daten als Zustände

Auch das Zustandsmodell wurde mithilfe der Klassifizierungsalgorithmen untersucht. Wie in Kapitel 3.3.2 erwähnt, wurden die untersuchten Spiele in vier Zustände eingeteilt. Diese basierten auf folgenden Ereignissen: Ballbesitz Team A, Ballbesitz Team B, Torschuss & Torerfolg. Für dieses Klassifikations-problem mit vier Kategorien erzielten die Klassifizierer die in Tabelle 3 angegebenen Ergebnisse:

Tabelle 3: Accuracy Score (Zustandsmodell)

Klassifizierer	TTS_1	TTS_2	TTS_3	TTS_mean
Neural Network	0.5110	0.4823	0.5096	0.5010
SVM (RBF Kernel)	0.5027	0.4950	0.4849	0.4942
Random Forest Classifier	0.7845	0.7903	0.7739	0.7829
Logistic Regression	0.5279	0.5352	0.5335	0.5322

Das neuronale Netzwerk, die Support Vector Machine mit RBF-Kernel und die logistische Regression brachten es dabei auf eine Vorsagegenauigkeit von ca. 50%. Der Random Forest-Klassifizierer weist eine signifikant höhere Prognosegenauigkeit als alle anderen Algorithmen auf. Im Schnitt konnte dieser Algorithmus bei drei Versuchen über 78% die richtige Kategorie prognostizieren.

In weiterer Folge wird noch einmal auf die Struktur eines Random Forests eingegangen. In Abbildung 16 wird ein Ast dargestellt, welcher dem Algorithmus zur Entscheidungsfindung dient. Der gesamte Baum besteht aus mehreren solchen Ästen und wird von oben nach unten gelesen.

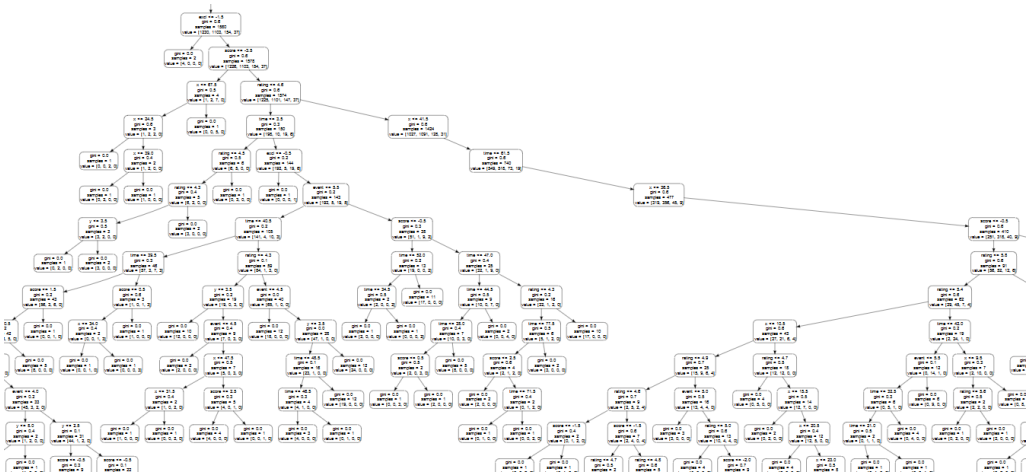

Abbildung 16: Ein Ast eines Random Forests

Um die Struktur eines Random Forests noch besser zu verstehen, wird in Abbildung 17 ein abermals kleinerer Abschnitt eines Asts betrachtet.

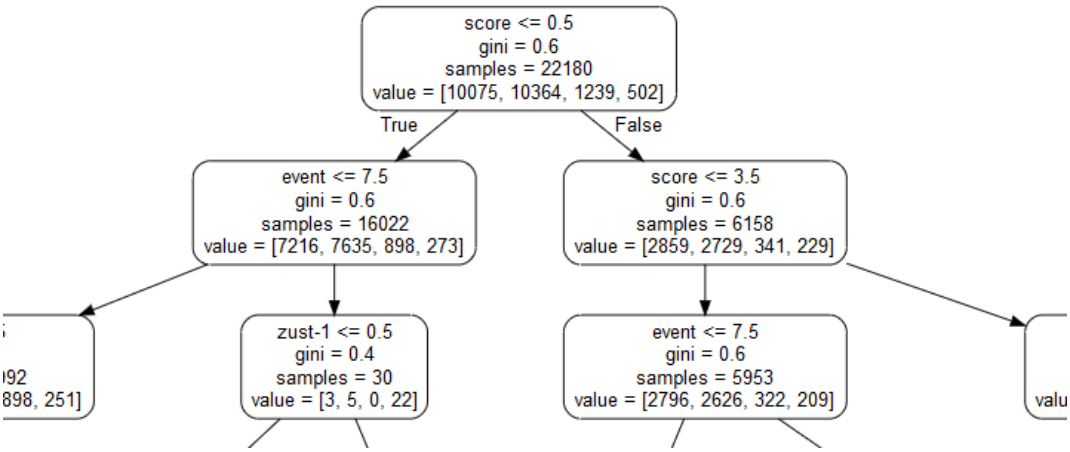

Abbildung 17: Detailansicht Random Forest

An der „Wurzel“ befinden sich insgesamt 22180 Zustandsdaten, welche klassifiziert werden sollen. Ist deren „score“-Wert <= 0.5, so wird dem „True“-Pfad links entlang gefolgt. Ist diese Behauptung jedoch falsch, so wird dem „False“-Pfad gefolgt und es wären nur mehr 6158 Zustandsdaten zu kategorisieren. Dieser Vorgang wird so lange wiederholt, bis der Algorithmus an den „Blättern“ angekommen ist und damit eine Klassifizierung vollzogen hat (vgl. Koehrsen, 2017).

4.4 Evaluierung & Optimierung der Algorithmen

Die Kreuzvalidierung wurde am Datensatz dieser Arbeit mithilfe des KFold-Iterators durchgeführt. Dieser Iterator wird in scikit-Learn bereitgestellt und teilt den Datensatz in k verschiedene Teilmengen ein (vgl. Mueller & Massaron, 2016, S. 346).

Wie auch in der Abbildung 11 (siehe Kapitel 3.4.4) wurde der für diese Arbeit verwendete Datensatz in zehn Teilmengen eingeteilt. Die Kreuzvalidierung für den Random Forest-Klassifizierer wurde außerdem nicht nur anhand des „Serie A"-Datensatzes, sondern auch für die weiteren vier untersuchten Ligen durchgeführt. Die erreichten Prognosewahrscheinlichkeiten wurden dabei in Tabelle 4 aufgelistet:

Tabelle 4: Ergebnisse der Kreuzvalidierung

Liga	Zustandsanzahl	Prognosewahrscheinlichkeit
Serie A	n = 27728	0.7647
Ligue 1	n = 30738	0.8452
Premier League	n = 19946	0.7745
Bundesliga	n = 23806	0.8473
Primera División	n = 25740	0.8184

Die Prognosewahrscheinlichkeiten des Random Forest-Klassifizierers waren für die weiteren vier Ligen sogar noch höher als jene für den „Serie A"-Datensatz. Nach dieser Validierung der Ergebnisse des Random Forest-Klassifizierers wird nun zusätzlich versucht, eine Optimierung der Parameter für diesen Algorithmus durchzuführen.

Random Forests haben nach Raschka & Mirjalili (2018, S. 117) im Allgemeinen den Vorteil, dass man sich wenig Gedanken über die Auswahl der Hyperparameter machen muss. Einzig die Anzahl der Entscheidungsbäume für den Random Forest wird in der Praxis manuell gewählt. Raschka & Mirjalili (2018, S. 117) stellen fest, dass der Random Forest-Klassifizierer typischerweise umso zuverlässiger arbeitet, aus je mehr Entscheidungsbäumen er besteht. Dies resultiert ihnen zufolge allerdings in einer erhöhten erforderlichen Rechenleistung und kann auch das Ausmaß der Überanpassung erhöhen.

Mit den Zustandsdaten wurde bezüglich des Random Forest-Algorithmus eine Rastersuche durchgeführt, um die verschiedenen Prognosewahrscheinlichkeiten anhand der Anzahl der Entscheidungsbäume vergleichen zu können. Dabei wurde die Anzahl der Entscheidungsbäume wie gerade erwähnt variiert und deren erreichte Prognosewahrscheinlichkeit berechnet. Diese Genauigkeit für die unterschiedliche Anzahl an Entscheidungsbäumen wurde jeweils mittels Kreuzvalidierung errechnet. In Abbildung 18 wird das Ergebnis dieser Rastersuche visualisiert.

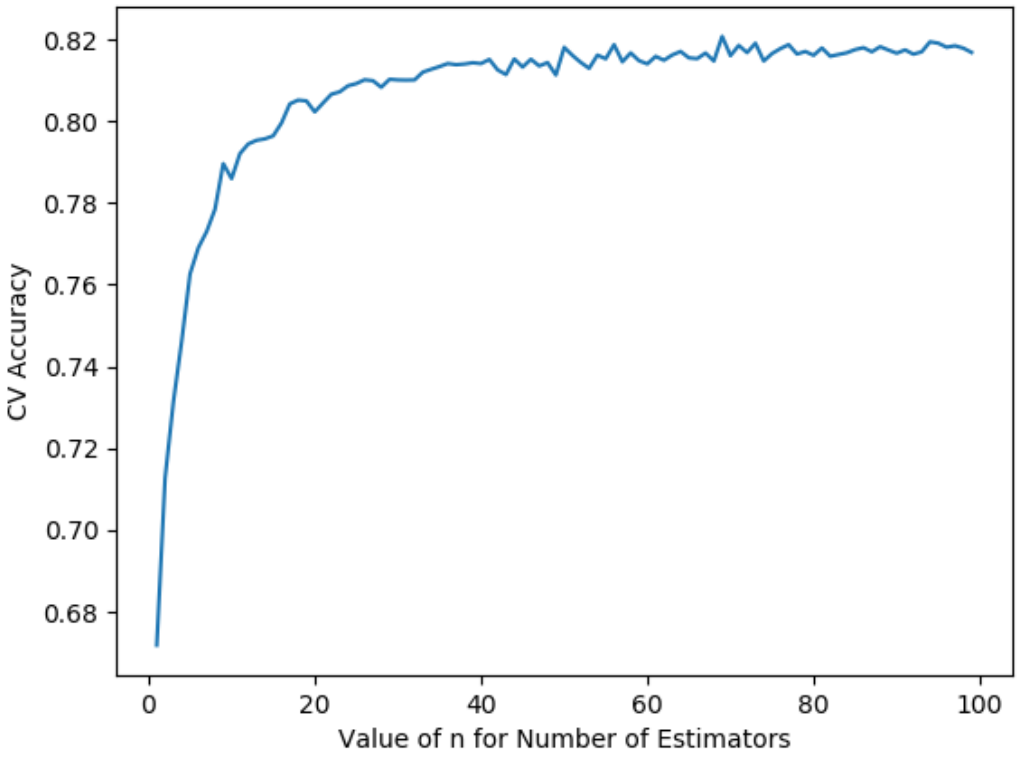

Abbildung 18: Ergebnisse der Rastersuche

Wie bereits erwähnt stellen Raschka & Mirjalili (2018, S. 117) fest, dass die Prognosegenauigkeit mit steigender Anzahl der Entscheidungsbäume ebenso steigt. Auch die hier durchgeführte Rastersuche kann diese Behauptung bestätigen. Ab einer Anzahl von ca. zwanzig Entscheidungsbäumen lag die Prognosegenauigkeit über 80% und variierte nur mehr leicht. Den besten Prognosewert mit knapp über 82% erreichte der Random Forest-Klassifizierer mit einer Anzahl von 69 Entscheidungsbäumen (siehe Abbildung 18).

Zusätzlich konnte auch die Wichtigkeit der Merkmale für eine Klassifizierung mithilfe der „Feature_importances“-Funktion ermittelt werden. Die Variable „Teamrating“ wies dabei den höchsten „Importance“-Faktor auf. Der Random Forest-Klassifizierer verwendete also häufig dieses Merkmal als Indikator, um eine Klassifizierung durchzuführen.

5 RESÜMEE UND AUSBLICK

Im letzten Kapitel dieser Arbeit erfolgt eine kurze Zusammenfassung der dargelegten Ergebnisse. Diese werden ebenso diskutiert und anderen Forschungsarbeiten des Themenfelds „Machine Learning im Sportspiel“ gegenübergestellt.

Zusammenfassend lässt sich sagen, dass im Laufe des Forschungsprozesses die Daten auf verschiedene Arten aufbereitet wurden. Für diese Arbeit wurden dabei zwei Modelle ausgewählt, welche zum Ziel hatten, eine Prognose von Torschussevents zu treffen.

Einerseits wurden die Daten kumuliert betrachtet. Diese wurden dabei in Ketten von Events zusammengefasst, wobei eine dieser Ketten für eine sinnvolle Klassifizierungsanwendung aus neun Events bestand. Dabei sollte der Computer prognostizieren, ob während dieser Spanne an Events ein Torschuss beziehungsweise ein Tor fällt. Die akkuratesten Klassifizierungs-algorithmen erreichten bei der Modellierung mit kumulierten Daten eine Prognosegenauigkeit von ca. 64%.

Um dieses Ergebnis zu interpretieren, sollte man es nicht direkt mit den Ausgängen anderer Forschungsarbeiten vergleichen. Grund dafür ist, dass in den bisherigen Arbeiten für das Sportspiel Fußball nämlich immer der Spielausgang prognostiziert und nicht wie in dieser Arbeit speziell die Torschussevents vorhergesagt werden sollten.

Trotzdem wird versucht, dieses Ergebnis in Relation zu anderen Arbeiten zu betrachten. Nazim et al. (2017, S. 7) erwähnen eine Prognosegenauigkeit von über 75% für die Vorhersage von Spielausgängen im Fußball. Dabei handelt es sich um ein Klassifizierungsproblem bezüglich drei Klassen (Heimsieg, Unentschieden, Auswärtssieg).

Führt man sich vor Augen, dass die Klassifizierung am kumulierten Datensatz nur 64% Prognosegenauigkeit für die Einteilung in zwei verschiedene Klassen erreicht hat, was grundsätzlich eine leichtere Aufgabe als eine Klassifizierung bezüglich drei Klassen darstellen sollte, so ist dieses Ergebnis nicht sehr zufriedenstellend.

Eine deutlich höhere Prognosewahrscheinlichkeit wurde bei der zweiten Aufbereitungsmethode der Daten erreicht. Bei der Modellierung der Daten als Zustände sollten ebenfalls Informationen über das Torschussverhalten erlangt werden. Es handelt sich dabei um ein Klassifizierungsproblem, wobei der Computer die Aufgabe hatte, sich zwischen vier Kategorien zu entscheiden und die richtige davon vorherzusagen.

Einige der angewandten Klassifizierungsalgorithmen konnten dabei eine Prognosegenauigkeit um die 50%-Marke erreichen. Der Random Forest-Klassifizierer dagegen erreichte eine vielversprechend hohe Prognose-genauigkeit. Diese lag mittels Kreuzvalidierung für die Spiele der französischen und deutschen Liga sogar bei fast 85%.

Mittels der „Feature_importances“-Funktion, welche in scikit-learn zur Verfügung steht, konnte das Rating als wichtigster Faktor für die Klassifizierung des Random Forests ausgemacht werden. Nicht nur in dieser Arbeit, sondern auch in anderen Forschungsarbeiten wie von Joseph et al. oder von Pettersson & Nyquist (2017, S. 15) spielen die Leistungsparameter der Teams beziehungsweise der Einzelspieler eine wichtige Rolle im Vorhersageprozess.

Die in dieser Arbeit erreichten Ergebnisse mit einer Prognosegenauigkeit von über 80% scheinen auf den ersten Blick enorm, müssen jedoch relativiert werden. Wenn man die Wahrheitsmatrizen für dieses Modell betrachtet, erkennt man, dass auch dieses Modell für die angestrebte Vorhersage von Torschüssen und Toren unbrauchbar ist. Die Zustände Tor bzw. Torschuss kamen in den untersuchten Spielen deutlich seltener vor als die beiden anderen Zustände des Modells, welche zu klassifizieren waren (Aktion Team A oder Aktion Team B). Der Algorithmus prognostizierte also hauptsächlich, welches Team beim nächsten Event im Besitz des Balles sein würde.

Die hohe Prognosewahrscheinlichkeit wurde vom Algorithmus also erreicht, weil er eine sehr genaue Vorhersage liefern konnte, welches Team beim nächsten Event am Ball sein würde. Diese Genauigkeit kann man als einen Erfolg werten, was allerdings nicht die Intention hinter dem Modell war.

Die Prognosegenauigkeit für einen Torschuss beziehungsweise einen Torerfolg lag im besten Fall bei ca. 15% (vgl. Confusion Matrix im Anhang). Dieser niedrige Wert kann unter anderem durch das Ungleichgewicht der zu prognostizierenden Klassen erklärt werden (vgl. Brownlee, 2017).

Das Ziel, Tor-Events vorherzusagen, konnte also nicht zufriedenstellend umgesetzt werden. Dabei muss auch die Schwierigkeit erwähnt werden, welche sich ergibt, wenn ein Sportspiel durch ein Modell abgebildet werden soll. Lames (1994, S. 35) erwähnt in diesem Zusammenhang, dass oft bewusst nur Teilaspekte des Spielgeschehens betrachtet werden.

Wenn man sich die Events, welche im Datensatz dieser Arbeit betrachtet und dokumentiert wurden, jedoch genauer ansieht, handelt es sich dabei eigentlich um Spielunterbrechungen. Ausgenommen vom Event „shot_on_target“, welches zu vorhersagen war, ist bei jedem einzelnen Event das Spiel selbst gestoppt (auch das Event „shot_off_target“ resultiert in einem Abstoß, also in einer Spielunterbrechung).

Die zu Verfügung stehenden Daten sagen also nahezu nichts über den eigentlichen Spielfluss aus. Zusätzliche Events wie zum Beispiel Zweikämpfe, Pässe, Ballverluste oder Dribblings wären weitere nützliche Informationen, um das Spielgeschehen realgetreuer zu modellieren.

Auch die moderne Erfassung von Positionsdaten (vgl. Kap 2.1.1) hätte genutzt werden können, um neben den Koordinaten der Ballevents auch die Positionen der Spieler als Information beizufügen.

Dies würde jedoch eine weitere Schwierigkeit mit sich bringen. Wie der Leiter der Abteilung für künstliche Intelligenz der Firma STATS erklärt, gibt es 3,5 Milliarden Permutationen, wie zehn Feldspieler im Fußball zueinander positioniert sein können. Wenn man auch die Gegenspieler miteinbeziehen würde, wären es mehr Möglichkeiten als Atome im Universum (vgl. Lucey in Woodie, 2017). Er betont jedoch auch, dass Teams, welche Big Data-Analysen nicht nützen, einen Nachteil haben. So wird von ihm auch der Ansatz des „Ghostings“ erwähnt, welcher Teams einen direkten Vorteil bringen soll. Dabei wird ein unüberwachtes, neuronales Netzwerk genutzt, um die Bewegungen von Spielern in bestimmten Situationen vorherzusagen (vgl. Le et al., 2017).

Auch Lames (2008, S. 306) hat keine Zweifel, dass die Informatik das Potenzial hat, das Coaching auf eine wesentlich breitere und tiefere Art und Weise zu unterstützen, als dies bisher geschehen ist. Man darf auf alle Fälle gespannt sein, was die Zukunft der Forschung auf diesem Gebiet betrifft.

Literaturverzeichnis

Bunker, R. P. & Thabtah, F. (2017). A machine learning framework for sport result prediction. Applied Computing and Informatics. ISSN 2210-8327. https://doi.org/10.1016/j.aci.2017.09.005

Brownlee, J. (2017). Classification Accuracy is Not Enough: More Performance Measures You Can Use. Zugriff am 7.5.2018 unter https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/

Constantinou, A. C., Fenton, N. E., & Neil, M. (2012). *pi-football: A Bayesian network model for forecasting Association Football match outcomes.* Knowledge-Based Systems, 36, 322-339. http://dx.doi.org/10.1016/j.knosys.2012.07.008

Garreta, R. & Moncecchi, G. (2013). Learning scikit-learn: machine learning in Python. Birmingham. Packt Publishing.

Guido, S. & Müller, A. (2017). Einführung in Machine Learning mit Python: Praxiswissen Data Science. O'Reilly Verlag. ISBN: 3960101120.

Grus, J. (2016). Einführung in Data Science – Grundprinzipien der Datenanalyse mit Python. Deutsche Übersetzung von Kristian Rother. ISBN: 9783960090212. 1. Auflage. O'Reilly. Dpunkt.verlag GmbH.

Hackeling, G. (2014). Mastering Machine Learning with scikit-learn. Birmingham. Packt Publishing.

Heydt, M. (2015). Learning pandas: get to grips with pandas - a versatile and high-performance Python library for data manipulation, analysis, and discovery. Birmingham, UK: Packt Publishing.

Joseph, A., Fenton, N. E., and Neil, M. (2006). Predicting football results using Bayesian nets and other machine learning techniques. Knowledge-Based Systems, 19(7), 544-553.

Koehrsen, W. (2017). Random Forest in Python - A Practical End-to-End Machine Learning Example. https://towardsdatascience.com/random-forest-in-python-24d0893d51c0. (Zuletzt aufgerufen am: 24.4.2018)

Kratz, T. (2017). Analyse und Interpretation von Big Data im Fußball. Zugriff am 2.5.2018 unter https://tinyurl.com/ychb2so5

Kumar, G. (2013). Machine Learning for Soccer Analytics. Thesis for: Master of Science in Artificial Intelligence. University College Dublin. 10.13140/RG.2.1.4628.3761.

Lames, M. (1994). Systematische Spielbeobachtung. Münster: Philippka.

Lames, M. (2008). Informatische Unterstützung des Coachings im Hochleistungssport. Informatik_Spektrum. S. 301-307.

Lames, M. & McGarry, T. (2007). On the search for reliable performance indicators in game sports. International Journal of Performance Analysis in Sport. Vol .7(1). 62-79.

Le, H. M., Carr, P., Yue, Y. & Lucey, P. (2017). Data-Driven Ghosting using Deep Imitation Learning. MIT-Sloan Sports Analytics Conference. Zugriff am 6.5.2018 unter https://s3-us-west-1.amazonaws.com/disneyresearch/wp-content/uploads/20170228130457/Data-Driven-Ghosting-using-Deep-Imitation-Learning-Paper1.pdf

Link, D., Kolbinger, 0., Weber, H. & Stöckl M. (2016). A topography of free kicks in soccer. Journal of sports sciences. ISSN: 0264-0414. 34:24, 2312-2320, DOI: 10.1080/02640414.2016.1232487.

Liti, C., Piccialli, V. & Sciandrone, M. (2017). Predicting soccer match outcome using machine learning algorithms. MathSport International 2017 Conference Proceedings. 229-237.

Memmert, D. & Raabe, D. (2017). Revolution im Profifußball - Mit Big Data zur Spielanalyse 4.0. Springer Verlag. Berlin, Heidelberg.

Mueller, J. P. & Massaron, L. (2016). Data Science mit Python für Dummies. Weinheim: WILEY, Wiley-VCH Verlag GmbH & Co. KGaA. 1. Auflage.

Nazim, R., Aida, M., Faiz, A. Y. & Ruhaya, A. A. (2017). Predicting Football Matches Results using Bayesian Networks for English Premier League (EPL). https://doi.org/10.1088/1757-899X/226/1/012099

Nopp, S. (2009). Systematische Spielanalyse im Fußballsport: Ein wichtiger Bestandteil der Wissenschaft und der Praxis. In *Forschung Innovation Technologie: das F.I.T.-Wissenschaftsmagazin der Deutschen Sporthochschule Köln, 14*(1), S. 24-28. Köln.

Pedregosa, F. *et al. (2011)*. Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830.

Pettersson, D. & Nyquist, R. (2017). Football Match Prediction using Deep Learning. Recurrent Neural Network Applications. Master's Thesis in Computer Science – algorithms, languages and logic. Gothenburg, Sweden. Zugriff am 10.5.2018 unter publications.lib.chalmers.se/records/fulltext/250411/250411.pdf

Polamuri, S. (2017). Visualize decision tree in python with graphviz. Zugriff am 24.4.2018 unter http://dataaspirant.com/2017/04/21/visualize-decision-tree-python-graphviz/

Raschka, S. (2016). How to select Support Vector Machine Kernels. Michigan State University. Zugriff am 3.5.2018 unter https://www.kdnuggets.com/2016/06/select-support-vector-machine-kernels.html

Raschka, S. & Mirjalili, V. (2018). Machine Learning mit Python und Scikit-Learn und TensorFlow: Das umfassende Praxis-Handbuch für Data Science, Deep Learning und Predictive Analytics. Mitp Verlag. 2. Auflage.

Röthig. P. & Prohl, R. (Hrsg.). (2003). Sportwissenschaftliches Lexikon. Hofmann Verlag. 7. Auflage.

Schnabel, G. & Thieß, G. (1993). Lexikon Sportwissenschaft: Leistung - Training - Wettkampf. Berlin: Sportverl. 1. Auflage.

Shalev-Shwartz, S. & Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press.

Woodie, A. (2017). Deep Learning Is About to Revolutionize Sports Analytics. Here's How. Zugriff am 7.5.2018 unter https://www.datanami.com/2017/05/26/deep-learning-revolutionize-sports-analytics-heres/

Zhang, H. (2004). The Optimality of Naive Bayes. Faculty of Computer Science University of New Brunswick. Zugriff am 25.4.2018 unter http://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

Anhang

Beispiel der Event-Timeline in XML-Struktur:

```
- <timeline>
      <event id="386924015" type="match_started" time="2018-02-05T19:48:07+00:00"/>
      <event id="386924011" type="period_start" time="2018-02-05T19:48:07+00:00"
          period_type="regular_period" period_name="1st half" period="1"/>
      <event id="386924195" type="throw_in" y="3" x="24" team="away" time="2018-02-
          05T19:48:44+00:00" period_type="regular_period" period="1" match_time="1"/>
      <event id="386924503" type="throw_in" y="2" x="32" team="away" time="2018-02-
          05T19:49:32+00:00" period_type="regular_period" period="1" match_time="2"/>
    - <event id="386924629" type="offside" y="10" x="32" team="away" time="2018-02-
      05T19:49:54+00:00" period_type="regular_period" period="1" match_time="2">
          <player id="sr:player:11724" name="Rigoni, Luca"/>
      </event>
      <event id="386925347" type="free_kick" team="home" time="2018-02-05T19:51:51+00:00"
          period_type="regular_period" period="1" match_time="2"/>
      <event id="386925225" type="throw_in" y="4" x="62" team="home" time="2018-02-
          05T19:51:28+00:00" period_type="regular_period" period="1" match_time="4"/>
      <event id="386925399" type="throw_in" y="4" x="77" team="home" time="2018-02-
          05T19:52:00+00:00" period_type="regular_period" period="1" match_time="4"/>
      <event id="386925703" type="throw_in" y="96" x="61" team="home" time="2018-02-
          05T19:52:58+00:00" period_type="regular_period" period="1" match_time="5"/>
      <event id="386926263" type="throw_in" y="3" x="16" team="away" time="2018-02-
          05T19:54:46+00:00" period_type="regular_period" period="1" match_time="7"/>
      <event id="386926439" type="shot_saved" team="home" time="2018-02-05T19:55:11+00:00"
          period_type="regular_period" period="1" match_time="8"/>
    - <event id="386926435" type="shot_on_target" y="46" x="10" team="away" time="2018-02-
      05T19:55:11+00:00" period_type="regular_period" period="1" match_time="8">
          <player id="sr:player:2695" name="Pandev, Goran"/>
      </event>
      <event id="386926779" type="throw_in" y="3" x="78" team="away" time="2018-02-
          05T19:56:07+00:00" period_type="regular_period" period="1" match_time="8"/>
```

Beispiel-Code in Python:

Auslese der xml-Dateien:

```
from bs4 import BeautifulSoup
import requests
import glob


objl = open("matchid.txt", "r").readlines()
i=1

for lines in objl:

    matchid = objl[i]

    print("Lese MatchID: ",matchid[9:17])


    url_2 = str(matchid[:17])+"/timeline.xml?api_key=966b5wtbazdg3ybc9dbb93eg"
    response = requests.get("https://api.sportradar.us/soccer-xt3/eu/en/matches/"+str(url_2))
    soup = BeautifulSoup(response.content, "html.parser")

    xmls = str(soup)
    id_m=i
    liste = [1]

    i=i+1

    while len(listemop) < 2:

        liste.append(0)
        f = open(str(id_m)+"testmatch.xml", "w")
        f.write(xmls)
        f.close()

objl.close()
```

Aufbereitung der Daten:

```
import xml.etree.cElementTree as ET
import os
import glob
import numpy as np
from random import *
from statistics import mean

xml_path = glob.glob("*.xml")
i=0

for doc in xml_path:

        i=i+1

        print ("Lese - ", {i} , os.path.basename(doc))

        class Variablen:
                global freekick
                freekick = 0

                global eventscount
                eventscount = 0

                global gefree
                gefree = 0

                global eafree
                eafree = 0

                global unfree
                unfree = 0

                global corner
                corner = 0
```

```
tree = ET.ElementTree(file=doc)

root = tree.getroot()

for d in root.iter('sport_event'):
        date = d.get('scheduled')
        dateb = (" '"+date+"'")

for c in root.iter('team'):
    teamid = c.get('id')
    teamnumber.append(teamid)

teamh = (" '"+teamnumber[0]+"'")


teamb = (" '"+teamnumber[1]+"'")


for b in range(leng):

        if dateb == data.iat[b,1]:

                for r in range(leng):

                        if datahome.iat[r, 0] == teamh:

                                try:
                                        ratingh = float((datahome.iat[r+1, 1]+datahome.iat[r+2, 1]))/2
                                        ratingh = format(ratingh, '.2f')
                                except:
                                        ratingh = 0
                        r=r+1

                for s in range(leng):

                        if dataaway.iat[s, 0] == teamb:


                                try:
                                        ratinga = float((dataaway.iat[s+1, 1]+dataaway.iat[s+2, 1]))/2
                                        ratinga = format(ratinga, '.2f')
                                except:
                                        ratinga = 0
                        s=s+1

        b=b+1


#print(ratingh, ratinga)

if not ratingh == 0 and not ratinga == 0:

        for child in root.iter('event'):
           info = child.get('type')
           info2 = child.get('team')
           info3 = child.get('x')
            #info4 = child.get('id')
           info5 = child.get('y')
           info6 = child.get('period')
           info7 = child.get('match_time')


           if info5 == None:
                    info5=50

           if info3 == None:
                    info3=50

           if not (info == "score_change" or info == "shot_on_target"):

                if info == "free_kick":
                    freekick=freekick+1

                    if info2 =='home':
```

Verwendung von Algorithmen:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier as KN
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn import neural_network
from sklearn.linear_model import LogisticRegression as LR
from sklearn import metrics
from sklearn.cross_validation import train_test_split as tts
from sklearn.grid_search import GridSearchCV
from sklearn.tree import export_graphviz
import scikitplot as skplt
import numpy
import pydot
import os


data = pd.read_csv("zust_buli.txt")
data.columns = ["un", "score", "time", "excl", "rating", "x", "y", "event", "zust-1",
feature_cols = ["score", "time", "excl", "rating", "x", "y", "event", "zust-1"]
#feature_cols = ["rating", "excl"]
#feature_cols = ["score", "time", "excl", "x", "y", "event", "zust-1"]


X = data[feature_cols]
y = data["zust"]

print("Anzahl an untersuchten Eventketten: ", len(y))
print(y.value_counts())

neur = neural_network.MLPClassifier([100, 50, 25])
rf = RandomForestClassifier(n_estimators = 100)

y_pred = neur.fit(X, y).predict(X)
print("Accuracy Score (NN_1):", metrics.accuracy_score(y, y_pred))

y_pred2 = rf.fit(X,y).predict(X)

print("Accuracy Score (RFC)100:", metrics.accuracy_score(y, y_pred2))
```

Confusion Matrix für den Random Forest-Klassifizierer beim Zustandsmodell:

Label 0 = Team A hat Ballbesitz
Label 1 = Team B hat Ballbesitz
Label 3 = Torschuss
Label 4 = Tor