Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Aktives Lernen von Fussballereignissen mit visueller Unterstützung

Lasse Merz

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Daniel Weiskopf
Betreuer/in:	Dipl.-Inf. Kuno Kurzhals, Robert Krüger, M.Sc.
Beginn am:	23. September 2016
Beendet am:	17. März 2017
CR-Nummer:	H.5.2, I.5.2

Kurzfassung

Videoanalyse gehört heutzutage zu den wichtigsten Mitteln im Sport, um das Spielverhalten von Mannschaften zu untersuchen. Dabei wird zum einen das Verhalten einzelner Spieler sowie die Taktik und Formation der Mannschaft anhand von statistischen Daten bewertet. Zum anderen wird versucht effektive Spielzüge und Verhaltensmuster zu identifizieren. Moderne Sensortechnik bietet die Möglichkeit, durch hochfrequente räumliche Bewegungsdaten der Spieler und des Balls die Videoanalyse zu erleichtern. Oftmals müssen interessante Ereignisse manuell annotiert werden, bevor die eigentliche Analyse stattfinden kann. In dieser Arbeit wird ein Konzept vorgestellt, das beim Finden von interessanten Events helfen soll. Ausgehend von ausgewählten Beispielszenen wird ein Klassifikationsmodell erstellt, mit welchem weitere ähnliche Szenen gefunden werden können. Für die optimale Modellbildung ist es möglich in jeder Phase Domainenwissen einzubringen. Dieses Konzept wurde anhand eines Fußballspiels implementiert und durch eine Evaluation mit verschiedenen Anwendungsfällen ausgewertet. Abschließend werden die Ergebnisse diskutiert.

Abstract

Video analysis is nowadays one of the most important means in sport to investigate the gaming behavior of teams. On the one hand, the behavior of individual players as well as the tactics and formation of the team are evaluated on the basis of statistical data. On the other hand, attempts are being made to identify effective moves and behavioral patterns. Modern sensor technology offers the possibility to facilitate the video analysis by high-frequency spatial movement data of the players and the ball. Interesting events have to be manually annotated before the actual analysis can take place. In this work a concept will be presented to find interesting events. Starting from selected sample scenes, a classification model is created with which further similar scenes can be found. For optimal modeling, it is possible to introduce domain knowledge in each phase. This concept was implemented on the basis of a soccer game and was evaluated with different use cases. Finally, the results are discussed.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Analysetools gehören im Fußball wie auch in fast allen Sportarten zum Standard [AS13]. Sie werden genutzt, um vergangene Spiele im Detail zu untersuchen. Dadurch versucht man Stärken und Schwächen des eigenen und der gegnerischen Teams oder gar einzelner Spieler herauszufinden. Die benötigten Daten werden dabei meistens durch Videoanalysetools berechnet. Diese nutzen die vielen Kameras, die heutzutage üblicherweise ein Spiel aufzeichnen, um Werte wie Beschleunigung, Geschwindigkeit und Position aller Spieler zu ermitteln. Daraus werden außerdem weiterführende Daten berechnet, wie zum Beispiel zurückgelegte Distanz, Ballbesitz, Passgenauigkeit oder auch Packing. Noch genauere Daten erhält man, wenn die Spieler GPS-Tracker direkt am Körper tragen, vor allem an den Beinen. Da diese Geräte aber hinderlich beim Spielen sein können, sind solche Datensätze selten und hauptsächlich von Trainingsspielen. Diese Daten liegen in einer sehr hohen Frequenz und Komplexität vor. Sowohl zeitliche und räumliche als auch Bewegungs- und Richtungsdaten sind vorhanden. Die Schwierigkeit liegt in der Verarbeitung dieser unterschiedlichen Daten durch effiziente Methoden, ohne wichtige Details zu verlieren.

Alle Profiteams beschäftigen einen oder gar mehrere spezielle Videoanalysten, die für die Vorbereitung auf Spiele und während den Spielen die Daten analysieren. Dabei werden sowohl einzelne Spieler beobachtet als auch die Taktik und Formation der Mannschaft untersucht. Auf Grundlage der daraus gewonnenen Erkenntnisse, können Vorschläge für die Trainingspläne, Taktik und die Mannschaftsaufstellung gemacht werden. Dabei muss meistens zuerst eine manuelle Analyse durchgeführt werden. Interessante Ereignisse wie Tore, Ecken, Fouls usw. müssen annotiert und ausgewertet werden. Die weltweit führende Software von STATS[1] und Opta [2] oder auch die in Deutschland angebotenen Services wie IMPIRE [3] und Fubalytics[4], bieten solche annotierten Videos zur Analyse an. Damit kann der Videoanalyst alle wichtigen Szenen und dazugehörenden Statistiken anschauen und auswerten.

Hier setzen Visual Analytics Ansätze an. Es wird versucht, möglichst viele Schritte automatisch oder semi-automatisch durchzuführen. Dabei soll der Nutzer die Möglichkeit haben, sein Domänenwissen einzubringen, um den automatischen Prozess zu verbessern. Die extrahierten Modelle sollen anschließend in verständliche visuelle Form gebracht werden. Zusätzlich sollen

[1] https://www.stats.com/
[2] http://www.optasports.com/
[3] http://www.bundesliga-datenbank.de/
[4] http://www.fubalytics.de/

interaktive Systeme es dem Nutzer erleichtern, einen Überblick über die Daten zu bekommen, diese zu analysieren und automatische Werkzeuge zu nutzen, um das gewünschten Ergebnis zu kommen.

Viele Ansätze existieren, um Analysten von Sportmannschaften zu unterstützen. Diese zielen hauptsächlich auf die übersichtliche Darstellung und Interaktion mit den Daten ab. Wenige Varianten bieten zusätzliche automatische oder semi-automatische Werkzeuge zur Identifizierung wichtiger Ereignisse an. Diese nutzen ein Klassifikationssystem, um verschiedene Ereignisse zu erkennen und einzuordnen. Die Klassifikation wird hierbei mit Hilfe von Algorithmen des maschinellen Lernens vorgenommen. In dieser Arbeit soll ein besonderer Fokus auf der interaktiven Exploration und Anpassung des benutzten Klassifikators liegen.

Aufgabenstellung

In dieser Arbeit soll untersucht werden, inwieweit ein Visual Analytics Ansatz einem Analysten dabei helfen kann, interessante Szenen aus einem Fußballspiel zu extrahieren. Als Grundlage steht ein Datensatz eines Fußballspiels mit GPS getrackten Daten und das dazugehörige Video zur Verfügung. Die erste Aufgabe ist es, aus den Daten geeignete Merkmale, sogenannte Features, zu extrahieren, die zur Beschreibung jeder Art von Spielszene wichtig sind. Anhand dieser Features und dem Video kann der Analyst ein Event definieren, das er untersuchen möchte. Aufgrund dieser Definition wird automatisch ein Modell berechnet, welches das Event beschreibt. Dieser automatische Prozess und das entstehende Modell sollen vom Nutzer angepasst und verändert werden können. Damit wird der Prozess und das Modell mit Hilfe von Expertenwissen optimiert. Die Herausforderung liegt darin, ein interaktives System zu entwickeln, das es dem Nutzer ermöglicht, dieses Ziel zu erreichen. Anschließend sollen durch Anwendung des optimierten Modells auf das gesamte Fußballspiel weitere Events gefunden werden, die dem definierten ähnlich sind. Diese können vom Nutzer inspiziert und validiert werden. Dadurch kann die Definition des Events verbessert werden. Indem dieser Prozess mehrmals iterativ durchgeführt wird, soll das generierte Modell möglichst genaue Ergebnisse liefern.

Gliederung

Diese Arbeit ist in acht Kapitel gegliedert. Auf die Einleitung folgen in Kapitel 2 einige Grundlagen, die in dieser Arbeit Anwendung finden. Verwandte Arbeiten werden in Kapitel 3 vorgestellt. Anschließend wird in Kapitel 4 das Konzept erläutert, das in dieser Arbeit entwickelt wurde. Danach wird in Kapitel 5 die Implementierung dieses Konzeptes erklärt. Eine Evaluation in Form von typischen Anwendungsfällen wird in Kapitel 6 beschrieben. Daraufhin folgt in Kapitel 7 eine Diskussion der Ergebnisse. Abschließend wird in Kapitel 8 die Arbeit zusammengefasst und ein Ausblick auf mögliche zukünftige Arbeiten gegeben.

2 Grundlagen

In diesem Kapitel werden einige wichtige Grundlagen erklärt. Dabei werden Techniken vorgestellt, die in dieser Arbeit verwendet werden. Diese sollen zum Verständnis der restlichen Arbeit beitragen.

2.1 Visual Analytics

Visual Analytics beschreibt einen Ansatz, um automatische Datenanalyse und interaktive Visualisierungen mit den Fähigkeiten des Menschen zu kombinieren [KMS+08]. Dabei soll die Fähigkeit, Trends und komplexe Muster zu erkennen, in die Datenanalyse einfließen, um letztendlich Wissen oder gar Entscheidungen aus den Daten abzuleiten. Visual Analytics ist interdisziplinär, da es Techniken aus unterschiedlichen Gebieten der Visualisierung, Datenanalyse und Mensch-Computer-Interaktion verbindet.

Heutzutage arbeiten wir mit riesigen Datenmengen [FB13; GR12]. Wir können alle Daten sammeln und abspeichern, um sie anschließend zu analysieren. Die Größe, Unterschiedlichkeit, Inkonsistenz und Komplexität dieser Daten stellt eine große Herausforderung dar. Die Aufgabe besteht darin, interessantes Wissen aus diesen Daten zu generieren.

Der Prozess von Visual Analytics ist in Abbildung 2.1 zu sehen. Am Anfang stehen die Rohdaten. Bevor diese in den Visual Analytics Prozess kommen, müssen sie in den meisten Fällen vorverarbeitet werden. Dazu gehören das Bereinigen, Normalisieren, Gruppieren und Vereinigen von Daten aus verschiedenen Quellen. Anschließend werden die Daten in ein einheitliches Format transformiert. Erst wenn diese Schritte abgeschlossen sind, kann die eigentliche Datenanalyse beginnen. Dabei kann der Nutzer entscheiden, ob er zuerst mit Hilfe von visuellen oder Data-Mining Methoden weiterarbeiten möchte. Je nachdem erhält er direkt eine Visualisierung der Daten oder er erzeugt zuerst ein Modell der Daten. Dieses Modell kann abhängig vom Anwendungsfall unterschiedlichste Formen annehmen. Nachdem ein Modell erstellt wurde, muss dieses validiert und bewertet werden. Dafür wird eine Visualisierung erzeugt, die es dem Nutzer ermöglicht, interaktiv mit den Daten zu arbeiten. Außerdem ist es dem Nutzer möglich, in die Modellgenerierung einzugreifen, indem er andere Parameter setzt oder eine andere Methode für die Modellgenerierung auswählt. Dadurch soll das Modell der Daten iterativ angepasst und verbessert werden. Dieser Wechsel zwischen automatischen Methoden zur Generierung der Modelle und der Anpassung und Verbesserung anhand einer visuellen Repräsentation durch einen Nutzer bildet den Kern des Visual Analytics Prozess. Durch diese

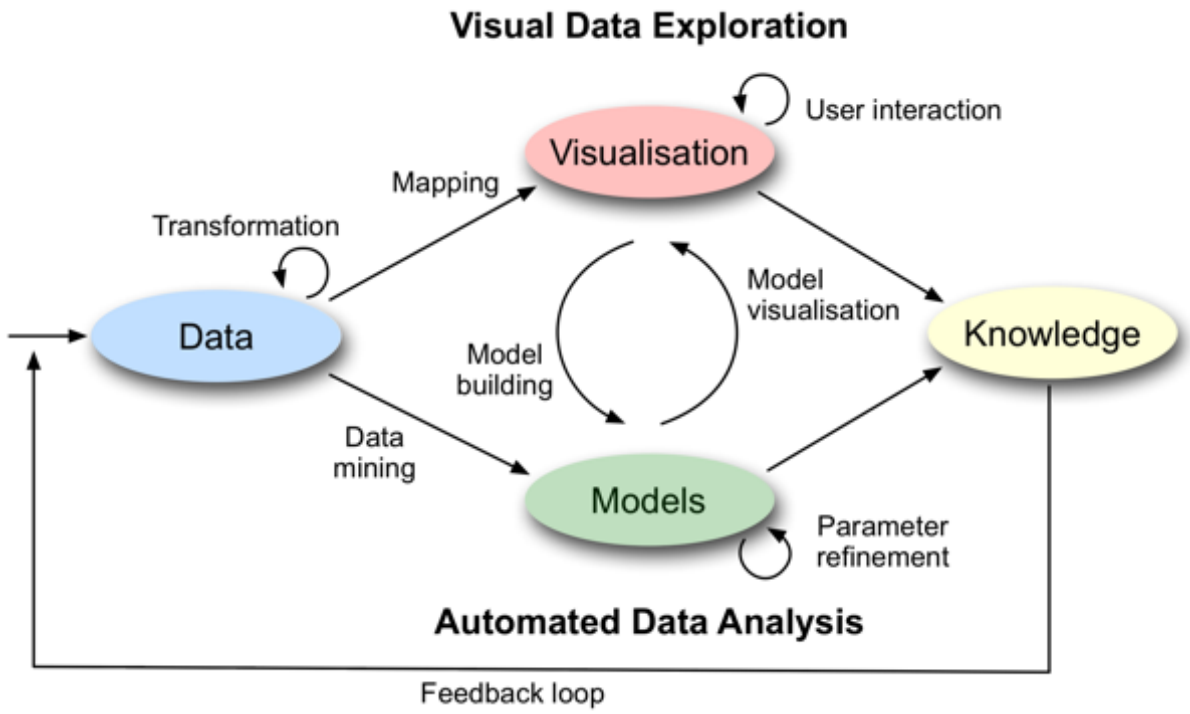


Abbildung 2.1: Visual Analytics Prozess [KKEM10]

kontinuierliche Iteration möchte man verhindern, dass Fehler oder Ausreißer zu schlechten Modellen der Daten führen. Die Integration des Nutzers in den gesamten Prozess soll hierbei die frühe Erkennung solcher fehlerhaften oder unpassenden Modelle sicherstellen. Wenn die Visualisierung vor dem Data-Mining ausgewählt wird, muss der Nutzer versuchen, Muster in dieser Visualisierung zu finden. Daraus kann eine Hypothese abgeleitet werden, die für den automatischen Prozess der Modellbildung genutzt werden kann.

Die interaktive Exploration der Daten durch den Nutzer ist essenziell für die Erzeugung eines optimalen Modells. Durch Funktionen wie Zoomen, Filtern und verschiedene Visualisierungsarten soll es dem Nutzer möglich sein, wichtige Informationen aus den Daten zu extrahieren. Dieses Wissen fließt anschließend in das Modell ein. Da wir in den meisten Fällen mit zu großen Datenmengen arbeiten, um alle Daten anzuzeigen, hat sich als Mantra für Visual Analytics etabliert: ”Analyse first, show the important, zoom/filter, analyse further, details on demand” [KMS+08].

2.2 Maschinelles Lernen

Lernen ist ein weitgreifender Begriff. Er beschreibt unter anderem das Erlernen neuen Wissens oder Fähigkeiten, das Lernen anhand einer Beschreibung oder Vorführung, Wissen in generellere Konzepte zu transformieren und neues Wissen durch ausprobieren und beobachten zu generieren. Maschinelles Lernen versucht diese Konzepte in einem Computer zu implementieren [CMM83]. Dadurch werden Programme entwickelt, die sich selbst verbessern, sich auf wechselnde Umgebungen anpassen und neues Wissen generieren können. Dabei lernt das Programm kontinuierlich auf Grund seiner Erfahrungen. Maschinelles Lernen ist stark mit Data-Mining verwandt, da im Data-Mining viele Algorithmen des maschinellen Lernens Anwendung finden, um neue Muster zu erkennen. Andere Anwendungsfelder liegen beim Filtern von Spam, Sprach-, Gesichts- und Schrifterkennung, bei der Erkennung von Krankheitsausbrüchen und der Robotersteuerung [Mit06].

Das maschinelle Lernen wird generell in zwei große Kategorien unterteilt: Das überwachte Lernen und das unüberwachte Lernen. Beim überwachten Lernen werden dem Programm Beispiele gegeben. Diese beinhalten sowohl Eingabedaten als auch das erwünschte Ergebnis. Das Programm soll daraus Regeln erlernen, die diese Input-Output-Abhängigkeit repräsentieren. In einem nächsten Schritt werden Daten mit nur den Eingabewerten übergeben und das Ergebnis wird anhand der gefundenen Regeln berechnet. Typischerweise wird diese Methode bei der Klassifikation verwendet, bei der Daten zu verschiedenen Klassen zugeordnet werden sollen. Dies wird zum Beispiel beim Filtern von Spam verwendet, wo das Programm anhand von Beispielen von Spam-Mails und normalen Mails erlernen soll, in Zukunft Spam-Mails zu erkennen. Eine Technik, mit der diese Klassifikation umgesetzt werden kann, ist der Entscheidungsbaum, der in dieser Arbeit verwendet und in Kapitel 2.3 erklärt wird.

Beim unüberwachten Lernen werden die Daten dem Algorithmus ganz ohne vorher bestimmte Klassen übergeben. Dieser erstellt anschließend von selbst ein Modell, das die Daten in verschiedene Muster oder Klassen einteilt oder eine einfachere Repräsentation der Daten erstellt. Diese Art von Algorithmen werden hauptsächlich im Data-Mining verwendet, da es dort gerade darum geht, neue Muster aus großen Datenmengen zu extrahieren. Ein Beispiel hierfür ist die sogenannte Clusteranalyse, bei der Datenmengen in Gruppen von ähnlichen Objekten aufgeteilt werden.

Es gibt noch weitere Kategorien des maschinellen Lernens, unter anderem das bestärkende Lernen (reinforcement learning) , bei der dem Programm eine Belohnungsfunktion gegeben wird. Das Programm versucht seine Aufgabe so zu lösen, dass diese Funktion maximiert wird, indem es anhand verschiedener Lösungsansätze die beste Möglichkeit erlernt.

In dieser Arbeit wird der Entscheidungsbaum als eine Technik des maschinellen Lernens genutzt, um ein Modell der Daten zu erstellen, wie es in Kapitel 2.1 mit dem Visual Analytics Prozess vorgestellt wurde. Der Algorithmus soll als Klassifikator von Fußballereignissen dienen. Das hierbei erlernte Modell kann in mehreren Iterationen angepasst, verbessert und individuell vom Nutzer gestaltet werden.

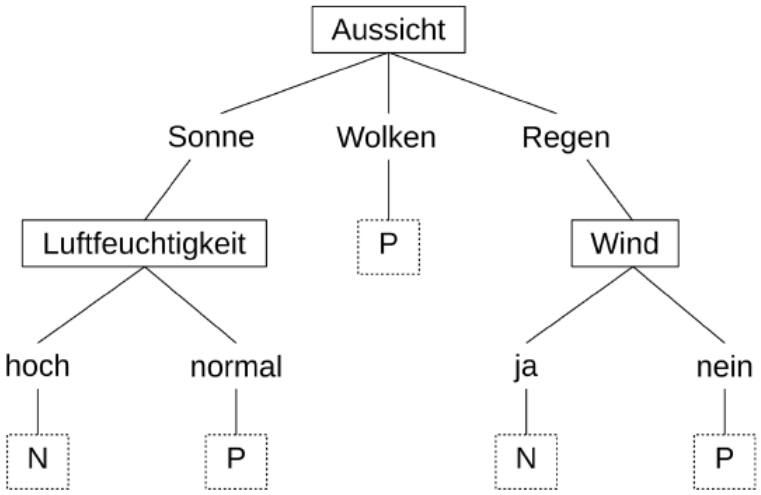


Abbildung 2.2: Beispiel für einen Entscheidungsbaum zur Klassifikation von Samstagvormittagen anhand des Wetters in positiv (P) und negativ (N) [Qui86].

2.3 Entscheidungsbaum

Unter einem Entscheidungsbaum versteht man eine Baumstruktur, die Regeln, ihre möglichen Entscheidungen und die daraus resultierenden Ergebnisse repräsentiert [BFSO84]. Entscheidungsbäume sind eine Technik der Klassifizierung und sie finden vor allem Anwendung im maschinellen Lernen. Sie können aber auch durch Experten manuell hergestellt werden. In dieser Arbeit soll eine Kombination von beiden Möglichkeiten zum Tragen kommen. Indem es dem Nutzer möglich ist, den automatisch erstellten Baum anzupassen, soll auch Expertenwissen in die Generierung des Entscheidungsbaums einfließen.

Jeder Knoten eines Entscheidungsbaums stellt eine Regel dar und jeder Pfad, der von diesem Knoten ausgeht, beschreibt eine Möglichkeit, diese Regel zu erfüllen. Wenn man anhand dieser Regeln durch den Baum gegangen ist, endet man an einem Blatt, welches eine vordefinierte Klasse repräsentiert. Generell arbeitet man mit Objekten, die verschiedene Attribute haben. Jede Regel testet dabei eines dieser Attribute und entscheidet auf Grund des Wertes, welcher Pfad genommen wird. Damit werden alle betrachteten Objekte anhand der Werte ihrer Attribute einer Klasse zugeordnet.

Ein Beispiel für einen Entscheidungsbaum sieht man in Abbildung 2.2. In diesem Beispiel wird das Wetter an Samstagvormittagen betrachtet und es soll entschieden werden , ob es positiv (P) oder negativ (N) ist. Damit sind die möglichen Klassen N und P, wobei jedes Objekt anhand seiner Attribute einer dieser beiden Klassen zugeordnet wird. Diese Attribute beschreiben in diesem Fall Wettereigenschaften wie Aussicht, Temperatur, Luftfeuchtigkeit und Wind. Jedes Attribut kann einen bestimmten Wert annehmen, zum Beispiel kann der Wert des Attributs Aussicht entweder Sonne, Wolken oder Regen sein. Ein Objekt o könnte wie folgt aussehen: *o = Aussicht: Sonne, Temperatur: heiß, Luftfeuchtigkeit: normal, Wind: nein.* Um o zu klassifizieren,

geht man von der Wurzel durch den Baum und entscheidet sich bei jedem Knoten anhand des Wertes des Attributs für einen Pfad, bis man zu einem Blatt gelangt. Bei der Aussicht geht man über den linken Pfad Sonne und anschließend über Luftfeuchtigkeit normal zu der Klasse P. Damit ist das Objekt o ein Beispiel für positives Wetter. Hier kann man sehen, dass häufig nur ein Teil der Attribute nötig ist, um ein Objekt zu klassifizieren.

Für die automatische Erstellung eines Entscheidungsbaums wird ein Trainingsdatensatz benötigt. Damit ist eine Menge an Objekten gemeint, die schon klassifiziert ist. Dabei werden Beispiele für alle möglichen Klassen benötigt. Anhand dieses Trainingsdatensatzes werden die Entscheidungsregeln erstellt, sodass die Trainingsdaten korrekt in ihre jeweiligen Klassen aufgeteilt werden. Beginnend bei der Wurzel wird ein Attribut ausgewählt, das die Trainingsdaten möglichst gut aufteilt. Anschließend werden für alle Werte dieses Attributes Pfade erstellt und der Trainingsdatensatz wird passend auf die Pfade verteilt. Bei jedem Pfad wird dieser Prozess mit der jeweiligen Teilmenge des Trainingsdatensatzes wiederholt. Sobald ein Pfad nur noch Datensätze einer Klasse beinhaltet, wird ein Blatt mit dieser Klasse erstellt. Das normale Vorgehen beinhaltet außerdem den Trainingsdatensatz aufzuteilen und zusätzlich einen Testdatensatz zu erstellen. Mit einer zufällig gewählten Teilmenge wird ein Entscheidungsbaum erstellt. Die Testdatensätze werden dann genutzt, um zu evaluieren, wie gut der Baum ist, indem man sie klassifiziert. Anschließend können die falsch klassifizierten Datensätze zum Trainingsdatensatz hinzugefügt werden, sodass der Baum erneut erstellt werden kann. In mehreren Iterationen kann somit der Entscheidungsbaum verbessert werden.

Die wichtigste Frage jedes Algorithmus zur Erstellung eines Entscheidungsbaums ist, welches Attribut in welchem Schritt gewählt wird, um den Datensatz aufzuteilen. Dazu wird der sogenannte Information Gain für jedes Attribut berechnet. Neben der unterschiedlichen Berechnung des Information Gain unterscheiden sich Algorithmen zur Erstellung von Entscheidungsbäumen darin, wie sie mit Ausnahmefällen, wie gleichen Objekten oder fehlenden Attributen, und zu großen Bäumen umgehen. Im folgenden Abschnitt wird der in dieser Arbeit gewählte Algorithmus zur Erstellung des Entscheidungsbaums erklärt.

2.3.1 C4.5 Algorithmus

Der C4.5 Algorithmus [Qui14] zur Erstellung eines Entscheidungsbaums ist ein Nachfolger des ID3 Algorithmus[Qui86], der als Grundlage für dieses Gebiet gilt. Beide Algorithmen wurden vom gleichen Autor entwickelt, der im C4.5 Algorithmus einige Verbesserungen eingeführt hat. Unter anderem wird eine neue Formel für den Information Gain benutzt. ID3 bevorzugt Attribute mit vielen Möglichkeiten. Falls wir zum Beispiel ein Attribut für die Steuernummer von Personen hätten, würde dieses genutzt, da es für jede Person eindeutig ist. Der resultierende Entscheidungsbaum, bestehend aus einem Knoten mit so vielen Pfaden wie es Personen gibt, wäre zwar korrekt, aber nicht sinnvoll oder hilfreich.

Der ID3 Algorithmus berechnet den Information Gain anhand zweier Größen. Dazu werden Formeln aus der Informationstheorie verwendet. Diese besagt, dass die Information, die in

einer Nachricht liegt, aus ihrer Wahrscheinlichkeit berechnet werden kann. Wenn man einen Datensatz D hat und aussagen möchten, dass ein Objekt d dieses Datensatzes zu einer Klasse C gehört, dann ist die Wahrscheinlichkeit dafür:

$$\frac{|\{d \in D | d \in C\}|}{|D|}$$

Die daraus gewonnene Information ist:

$$-\log_2(\frac{|\{d \in D | d \in C\}|}{|D|}) \text{ bits}$$

Um damit die erwartete Information für alle k vielen Klassen zu berechnen, wird über alle Klassen in Abhängigkeit ihrer Wahrscheinlichkeit summiert. Damit erhält man:

$$\text{info(D)} = -\sum_{n=1}^{k} \frac{|\{d \in D | d \in C\}|}{|D|} * \log_2(\frac{|\{d \in D | d \in C\}|}{|D|}) \text{ bits}$$

Anschließend wird betrachtet, was passiert, wenn der Datensatz anhand eines Attributes X aufgeteilt wird. Falls die Aufteilung durch X zu n vielen Möglichkeiten führt, kann die erwartete Information von D in Abhängigkeit von X durch die Summe dieser n Teilmengen berechnet werden:

$$\text{info}_x(D) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} * \text{info(D}_i)$$

Der Information Gain ergibt sich als:

$$\text{gain(X) = info(D) - info}_x(D)$$

Im C4.5 Algorithmus wird eine weitere Größe eingeführt, die sogenannte $split\ info$, die zu einer Art Normalisierung führt:

$$\text{split info(X)} = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} * \log_2(\frac{|D_i|}{|D|})$$

Diese berechnet die erwartete Information, die man erhält, wenn man den Datensatz durch das Attribut X aufteilt, unabhängig der möglichen Klassen.

Mit dieser Information wird die $gain\ ratio$ berechnet:

$$\text{gain ratio(X) = gain(X)/split info(X)}$$

Beim C4.5 Algorithmus wird der mit dieser Formel berechnete Wert als Maß für die Wahl der Attribute genutzt und löst somit den reinen Information Gain ab.

Diese Methode wird sowohl für kategorische als auch numerische Attribute benutzt. Bei numerischen wird als Ergebnis ein Splitpoint berechnet, wobei alle Datensätze in > und <= relativ zu diesem Splitpoint aufgeteilt werden. Dafür werden die Datensätze anhand dieses numerischen Attributs sortiert. Anschließend wird durch diese Liste iteriert und jeweils der Mittelwert zwischen dem gerade zu betrachtenden Wert und dem nächsten Wert berechnet und, der höchste Wert des Datensatzes, der nicht höher als der berechnete Wert ist, wird als Splitpoint definiert. Auf diese Weise wählt man als Splitpoint immer einen Wert, der im Datensatz vorkommt. Danach wird anhand dieses Splitpoints aufgeteilt und die Möglichkeit mit der höchsten $gain\ ratio$ gewählt. Außerdem besitzt der Algorithmus eine Schranke, die besagt, dass mindestens zwei der Teilmengen, die durch die Aufteilung entstehen, eine relevante Größe haben müssen. Diese relevante Größe wird standardmäßig auch auf 2 gesetzt. Dadurch sollen triviale Aufteilungen vermieden werden.

Diese Formel kann zudem leicht für Datensätze mit fehlenden Werten erweitert werden. Für diese wird bei der Berechnung von $gain(X)$ und $split\ info(X)$ eine zusätzliche Teilmenge mit ihrer Wahrscheinlichkeit erstellt. Bei der Frage, zu welcher tatsächlichen Teilmenge der Datensatz mit fehlendem Wert zugeordnet wird, nachdem die Aufteilung vollzogen wurde, entscheidet sich der C4.5 Algorithmus dafür, diesen Datensatz allen Teilmengen mit einem Gewicht relativ zu deren Wahrscheinlichkeit zuzuordnen. Beim Klassifizieren eines neuen Objekts wird der selbe Ansatz gewählt. Falls ein Knoten auf ein Attribut testet, das keinen Wert hat, werden alle Möglichkeiten ausprobiert. Am Ende wird die Klasse gewählt, die man am häufigsten erreicht hat.

Eine wichtige Technik des C4.5 Algorithmus ist das pruning. Nachdem der Entscheidungsbaum erstellt wurde, wird er anschließend, so weit es geht, gekürzt. Dazu geht man den Baum von unten nach oben durch und falls die erwartete Fehlerrate an einem inneren Knoten kleiner ist als an seinen Kindern, werden diese entweder durch einen Blattknoten ersetzt oder durch den besten Teilbaum. Dadurch können Blattknoten nicht mehr nur eine Klasse bezeichnen, sondern sie geben eine Verteilung über die Klassen an. Der Blattknoten wird mit der häufigsten Klasse bezeichnet, gibt aber zusätzlich an, wie viele der Trainingsdaten nicht zu dieser Klasse gehören. P(5/1) bedeutet, dass dieses Blatt die Klasse P repräsentiert, fünf der Trainingsdaten an diesem Blatt ankommen und davon ein Datum nicht zur Klasse P gehört.

Zusammenfassend beginnt der C4.5 Algorithmus anhand eines Trainingsdatensatzes von der Wurzel des Entscheidungsbaums an, diesen Trainingsdatensatz auf die Klassen aufzuteilen. In jedem Schritt wird das Attribut mit der höchsten $gain\ ratio$ ausgewählt und der Trainingsdatensatz wird auf die jeweiligen möglichen Werte dieses Attributes aufgeteilt. Dies wird solange rekursiv wiederholt, bis eine Teilmenge des Trainingsdatensatzes nur noch aus Daten einer Klasse besteht, für welche ein Blattknoten erstellt wird. Im anschließenden Pruning-Schritt wird der Entscheidungsbaum, soweit es die erwartete Errorrate minimiert, verkürzt, wodurch Knoten mit einer Verteilungsfunktion der Klassen entstehen.

Der C4.5 Algorithmus wird in dieser Arbeit verwendet, um aus ausgewählten Beispielszenen einen Entscheidungsbaum zu erzeugen. Anhand dieses Entscheidungsbaums werden alle anderen Szenen klassifiziert, wodurch ähnliche Szenen gefunden werden.

3 Verwandte Arbeiten

In diesem Kapitel werden verwandte Arbeiten aus den Themenbereichen des maschinellen Lernens, Visual Analytics und deren Anwendungen auf die Fußballanalyse vorgestellt.

3.1 Maschinelles Lernen und Visual Analytics

Ein wichtiger Aspekt des maschinellen Lernens ist es, die erzeugten Modelle einfach verständlich zu visualisieren. Außerdem ist es essenziell den Nutzer interaktiv in den Prozess der Modellbildung und Evaluation einzubinden. Liu et al. [LWLZ17] geben einen Überblick über Ansätze der Visual Analytics für die Analyse von Modellen des maschinellen Lernens. Eine zentrale Aufgabe ist die Wahl der geeigneten Features, die zur Modellerzeugung genutzt werden sollen. Mühlbacher und Piringer [MP13] stellen ein System zur interaktiven Auswahl von Features für die Regressionsanalyse anhand ihres Rankings und ihrer Abhängigkeiten vor. Weitere Ansätze für die interaktive Auswahl von wichtigen Features werden für die Vorhersage des Erfolgs von Kinofilmen benutzt [KBT+13; LKT+14]. In diesen Arbeiten wird dem Nutzer die Relevanz der Features, die Zusammenhänge und Abhängigkeiten der Features und die Auswirkungen der Wahl der Features auf das berechnete Modell in geeigneten Visualisierungen dargestellt.

Für das überwachte maschinelle Lernen ist außerdem die Wahl des Testdatensatzes besonders wichtig. Liao et al. [LCSM16] stellen ein System vor, das es Nutzern erleichtert Videos mit Labeln zu versehen. Eine Scatterplot Visualisierung mit einem konturbasierten Hintergrund der Videos ermöglicht es dem Nutzer Cluster von Videos zu erkennen und mit den passenden Labeln zu kennzeichnen.

3.2 Videoanalyse und automatische Eventerkennung

Die Analyse von Fußballspielen wird hauptsächlich anhand von Videos durchgeführt. Heutzutage wird jedes Spiel mit mehreren Kameras aufgenommen. Dabei ist es wichtig alle relevanten Informationen und Statistiken aus den Videos zu extrahieren. In einem weiteren Schritt erleichtert es die Analyse enorm, wenn Events automatisch oder semi-automatisch generiert werden können, da die manuelle Annotation von Videos sehr zeitaufwendig ist.

Einen Überblick über die automatische Eventerkennung in der Fußballanalyse anhand von Videos geben de Sousa et al. [SAM11]. Vorgestellt werden typische Ansätze für das Erkennen des Spielfelds, des Balls, der Spieler und wie daraus komplexere Events entstehen.

Rathod et al. [RN14] erweitern diesen Ansatz auf multimodale Methoden, um Events aus Videos zu extrahieren. Dabei werden sowohl Informationen direkt aus dem Video gewonnen als auch durch das dazugehörige Audio und gegebenenfalls eingeblendeten Text.

Weitere Ansätze zur Eventerkennung werden von Zawbaa et al. [ZEHK12] beschrieben, die ein System vorstellen, um automatische Zusammenfassungen aus Fußballvideos zu generieren. Mit Hilfe von mehreren Algorithmen des maschinellen Lernens wird das Video in verschiedene Shots eingeteilt, welche anschließend anhand von Features aus dem Video, dem dazugehörigen Ton und eingeblendetem Text in spannende und uninteressante Szenen Shots klassifiziert werden. Dabei wird außerdem Expertenwissen für die Wahl und Gewichtung der Features genutzt. Dieses Ziel wird auch von Assfalg et al. [ABD+02] anhand von Hidden-Markov-Modellen und Chen et al. [CSCZ04] mit Hilfe von Entscheidungsbäumen verfolgt.

3.3 Visual Analytics in der Sportanalyse

In der Sportanalyse geht es vor allem darum, dem Analysten hilfreiche Tools zu liefern, die Spiele seiner Mannschaft und von Gegnern analysieren zu können. Dabei ist es interessant eine Übersicht über das Spielgeschehen und einzelne Spieler zu erhalten. Weiterhin möchte man das Verhalten der Mannschaft in einzelnen Szenen untersuchen.

Shao et al. [SSN+16] stellen ein Tool vor, bei dem Nutzer Skizzen von interessanten Bewegungsmustern in ein Fußballfeld zeichnen können, woraufhin das System alle Situationen mit diesem Bewegungsmuster findet. Dabei können sowohl einzelne Muster und komplexere Spielsituationen als auch abstrakte Situationen gezeichnet werden. Durch geeignete Filter und Optionen kann der Nutzer seine Suche interaktiv verbessern.

Matchpad [LCP+12] ist eine Software für die interaktive Visualisierung von Footballspielen. Analysten können Spiel-, Team- oder Spielerevents annotieren, welche anschließend in einer übersichtlichen Visualisierung analysiert werden können. Verschiedene Events sowie dazugehörige Eigenschaften und Attribute werden wiedererkennbar mit Glyphen symbolisiert, was die Analyse einfacher macht.

Im Gegensatz zu Abschnitt 3.2 stellen Gudmundsson et al. [GW10] ein Tool vor, das anhand allein der Positionsdaten der Spieler Events extrahiert. Mit Hilfe von annotierten Events wird die Position des Balls ermittelt, um im nächsten Schritt einfache Events wie Ball im Feld oder im Aus und Ball berührt zu erkennen. Dadurch können anschließend komplexere Events wie Pässe, Ballbesitz und Torschüsse extrahiert werden, um im letzten Schritt Events wie Freistoß, Faul und Auswechslung zu erkennen. Aus diesen Daten können außerdem die Bewegungsmuster der Spieler berechnet und analysiert werden.

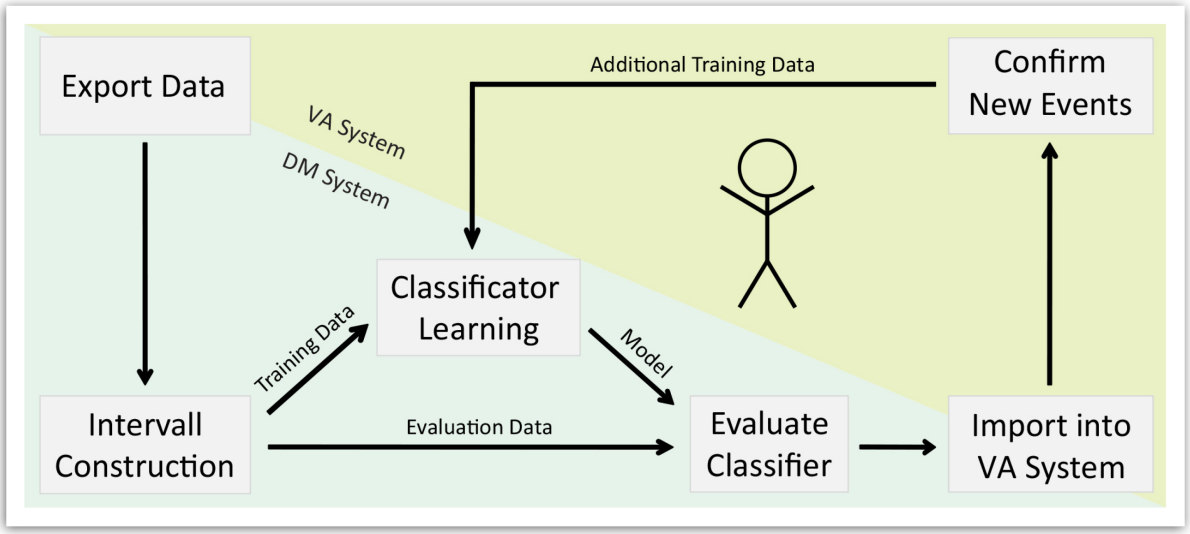


Abbildung 3.1: Von Sacha et al.[SSS+14] vorgeschlagener Workflow für die Klassifikation von interessanten Fußballszenen.

Im Vergleich dazu gibt es Systeme, die sich von den einzelnen Events lösen und direkt ganze Mannschaftsformationen analysieren [BLC+14; PGM13]. Bei diesen Ansätzen werden die Rollen der Spieler und die verschiedenen Formationen der Mannschaften ermittelt. Damit können diese Formationen analysiert und hinsichtlich ihres Erfolgs sowie ihrer Effektivität bewertet werden.

Ein System zur Analyse von Fußballspielen auf verschiedenen Leveln wird durch Sacha et al. [SSS+14] vorgestellt. Sowohl einzelne Spieler, mehrere Spieler als auch ganze Spielereignisse können analysiert werden. In einem ersten Schritt werden die Features eines einzelnen Spielers in einem definierten Zeitabschnitt berechnet. Diese werden mit Hilfe von Clustering-Verfahren in verschiedene Spielphasen eingeteilt. Die Ergebnisse dieses Verfahrens können in einer Übersicht angeschaut werden. Damit kann die Leistung eines Spielers analysiert und verglichen werden. Dieser Ansatz wird auf Mannschaftsformationen erweitert. Damit kann das Verhalten der gesamten Mannschaft analysiert werden. Eine dritte Möglichkeit wird in der Eventanalyse geboten. Hier können annotierte Events anhand der Features der beteiligten Spieler begutachtet oder zum Finden ähnlicher Szenen genutzt werden. Dazu wird der in Abbildung 3.1 zu sehende Workflow benutzt. Nachdem die Features berechnet und exportiert wurden, werden sie in Zeitintervalle eingeteilt und anschließend jedem zur Verfügung stehend Klassifikator übergeben. Die besten Klassifikatoren werden genutzt, um neue ähnliche Szenen zu finden, welche vom Nutzer verifiziert werden.

Auf diesem Ansatz bauen Stein et al. [SHJ+15] auf. Sie stellen ein System vor, mit dem interessante Szenen eines Fußballspiels analysiert werden können. Das Spiel wird in definierte Intervalle aufgeteilt. Anhand von ausgewählten Szenen werden diese Intervalle in interessante und uninteressante klassifiziert. Diese Intervalle werden anschließend zu Spielszenen erweitert,

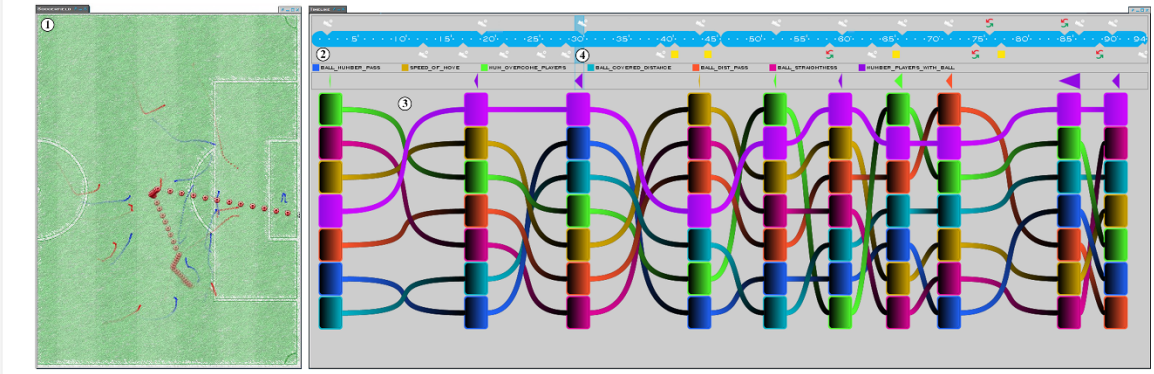


Abbildung 3.2: Benutzeroberfläche des Systems von Stein et al. [SHJ+15], bestehend aus dem Spielfeld (1), der Zeitachse (2) und dem Feature-Diagramm (3).

welche mit einem Ballgewinn anfangen und mit dem endgültigen Verlust des Balls aufhören. Die gefundenen Spielszenen können mit Hilfe von ausgewählten Features analysiert werden, um in einem nächsten Schritt ähnliche Szenen zu finden. Das System wird in Abbildung 3.2 gezeigt. Auf der linken Seite (1) sieht man die derzeit betrachtete Szene. Rechts sieht man ganz oben das Spiel anhand einer Zeitachse (2). Diese wird zur einfachen Navigation und Darstellung einiger annotierter Ereignisse genutzt. Die klassifizierten interessanten Szenen werden in einer Art Flussdiagramm angezeigt (3). Jede Szenen wird dabei durch eine nach Wichtigkeit sortierte Featureliste repräsentiert. Das wichtigste Feature der jeweiligen Szene wird durch einen dazugehörigen Pfeil veranschaulicht (4). Dadurch soll es dem Nutzer möglich sein, nicht nur zu erkennen, welche Szenen interessant sind, sondern auch warum. In diesem Beispiel ist das lilafarbene Feature in jeder Szene besonderes dominant weit oben. Der Analyst kann das Ranking interaktiv verändern, um sein Expertenwissen in die Suche nach ähnlichen Szenen einfließen zu lassen.

Der in dieser Arbeit vorgestellte Visual Analytics Ansatz kann hier eingeordnet werden. Zusätzlich zu dem in Abbildung 3.1 zu sehenden Workflow soll es dem Nutzer möglich sein in den Klassifikationsprozess durch geeignete Manipulation des Entscheidungsbaums direkt einzugreifen. Damit kann der Nutzer bestimmen, welche Features wichtig für Ereignis sind. Dies soll vor allem die Klassifikation von neuen Szenen unterstützen, welche anschließend automatisch gefunden werden. Außerdem wird für die Auswahl der Beispielszenen und für die Verifikation der gefundenen Szenen das Video des Spiels genutzt.

4 Konzept

Verschiedene Softwaretools erleichtern es, Fußballspiele zu analysieren. Neben Übersichten über unterschiedliche Statistiken des Fußballspiels geht es dabei immer mehr um die automatische oder semi-automatische Analyse des Spiels. In diesem Kapitel wird im Folgenden ein Konzept vorgestellt, das es dem Nutzer ermöglichen soll, durch die Hilfe eines automatischen Klassifikationprozesses, interessante Events eines Fußballspiels zu finden. Dabei hat der Nutzer die Option, aktiv in diesen automatischen Prozess einzugreifen.

Oftmals sind automatische Klassifikationsalgorithmen für den Nutzer schwer zu durchschauen. Man wird mit einem Ergebnis konfrontiert, ohne zu verstehen, wie es zustande kam. Damit verliert man potentielles Wissen, das der Nutzer zur Verbesserung der Klassifikation beisteuern kann. Nutzer besitzen oft Domainenwissen, das nicht in den Daten widergespiegelt wird. Damit können im Voraus mögliche Fehler, Ausreißer, Unterschiede und Gemeinsamkeiten in den Daten identifiziert werden. Dieses Wissen integriert in einen automatischen Algorithmus, kann zu optimierten Klassifikationsergebnissen führen. Dehsalb liegt der Schwerpunkt dieses Konzepts auf der Integration des Nutzers in den volltändigen Klassifikationsprozess.

Für die einfache Verständlichkeit des Klassifikators wurde der Entscheidungsbaum als Verfahren ausgewählt. Ein Entscheidungsbaum ist direkt ein visuelles Modell des Klassifikators, das dem Nutzer angezeigt werden kann. Dadurch entsteht keine Abstraktion zwischen dem berechneten und visuellen Modell. Außerdem ist er leicht zu interpretieren. Über die Knoten kann abgelesen werden, welche Features als wichtig gelten. Die Kanten geben zusätzlich Auskunft darüber, wie sich die verschiedenen Szenen in diesen Features unterscheiden. Weiterhin kann die Manipulation des Klassifikators im berechneten Modell umgesetzt werden. Der Entscheidungsbaum kann direkt verändert und angepasst werden. Damit bietet sich der Entscheidungsbaum für die Integration des Nutzers in ein Klassifikationsverfahren an.

Den generellen Ablauf für die Benutzung des Systems kann man in Abbildung 4.1 sehen. Als erstes werden die Daten des Fußballspiels in das System importiert. Als nächstes werden daraus relevante Features berechnet, die typischerweise eine Szenen eines Fußballspiels beschreiben. Im nächsten Schritt wird der Nutzer aktiv. Der nachfolgende Prozess wird für jede Art von Event durchgeführt, dass erkannt werden möchte. Dazu wird eine Kategorie für das zu erkennende Event erstellt. Anschließend wählt der Nutzer einige Beispielszenen für diese Kategorie aus dem Video des Fußballspiels aus. Anhand dieser Beispielszenen wird ein Klassifikator trainiert. Das Ergebnis dieses Klassifikators wird in Form eines Entscheidungsbaums visualisiert. Diesen Entscheidungsbaum kann der Nutzer beliebig verbessern, anpassen und verändern. Sobald dieser Schritt abgeschlossen ist, können alle restlichen Szenen des Spiels

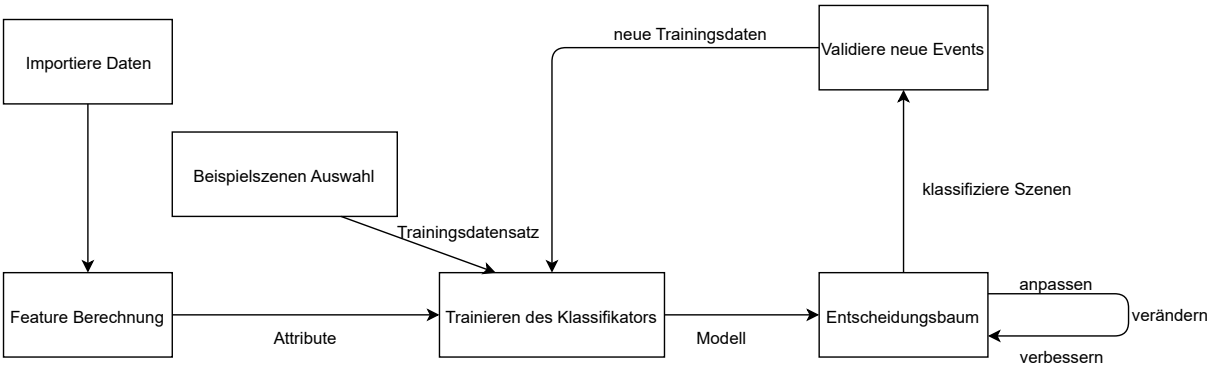


Abbildung 4.1: Ablaufdiagramm des Systems

anhand des Entscheidungsbaums klassifiziert werden. Die gefundenen Szenen kann der Nutzer analysieren und gegebenenfalls validieren. Diese Szenen werden daraufhin zum Trainingsdatensatz hinzugefügt und eine weitere Iteration mit einem verbesserten Entscheidungsbaum kann erstellt werden.

Dieses Grundkonzept zur Klassifikation von Events ist nicht nur auf Fußball beschränkt. Es kann auf jede andere Sportart und weitere Anwendungen, die Klassifikation benutzen, übertragen werden. Für diese Arbeit wurde ein Datensatz eines Fußballspiels zur Verfügung gestellt, welcher im Folgenden zusammen mit den einzelnen Aspekten des Konzepts vorgestellt wird.

4.1 Daten

In dieser Arbeit wird ein Datensatz verwendet, der für die DEBS (Distributed Event-Based Systems) 2013 Grand Challenge [MZJ13] zur Verfügung gestellt wurde. Jedes Jahr wird eine Challenge vorgestellt, die mit Hilfe eines Event basierenden Systems gelöst werden soll. Anschließend werden die besten Lösungen neben anderen Themen auf der internationalen DEBS Konferenz diskutiert.

Bei der DEBS 2013 Grand Challenge geht es um die Eventanalyse eines Fußballspiels in Echtzeit. Die Daten wurden während eines Fußballspiels in Nürnberg, Deutschland, erhoben. Dabei handelte es sich um ein Trainingsspiel zwischen zwei Mannschaften mit jeweils 8 Spielern. Das Spiel wurde auf einer Hälfte eines traditionellen Fußballfeldes gespielt und bestand aus zwei Halbzeiten mit jeweils 30 Minuten. Erhoben wurden die Daten mit Hilfe des RedFIR tracking System [GFW+11]. Dazu hatte jeder Spieler und der Schiedsrichter einen Sensor an beiden Beinen. Zusätzlich besaßen Torhüter einen Sensor an beiden Händen. Außerdem hatte der Ball einen integrierten Sensor. Die von den Sensoren gesendeten Daten wurden über zwölf im Stadion installierte Antennen empfangen, zentral verarbeitet und gespeichert. Die Sensoren der Spieler sendeten mit einer Frequenz von 200 Hz und die Sensoren der Bälle mit einer Frequenz von 2000 Hz. Damit wurden ca. 15000 Events pro Sekunde erzeugt.

Für ein Event wurden zwölf verschiedene Attribute aufgezeichnet, welche in Tabelle 4.1 betrachtet werden können.

Attribut	Erklärung
sid	ID des Sensors
ts	Timestamp bzw. Zeitpunkt des Events in picosekunden
x,y,z	Koordinaten des Sensors in mm
|v|	Geschwindigkeit in m/s
|a|	Beschleunigung in m/s^2
vx, vy, vz	richtungsbasierte Geschwindigkeit mit Normalisierungsfaktor 10.000
ax, ay, az	richtungsbasierte Beschleunigung mit Normalisierungsfaktor 10.000

Tabelle 4.1: Tabelle aller zur Verfügung stehender Attribute

Die x, y und z Koordinaten sind dabei abhängig vom Mittelpunkt (0,0,0) der sich im Mittelpunkt des vollständigen Fußballfeldes befindet. Die x und y Koordinaten sind sehr genau und weichen höchstens wenige Zentimeter ab. Die z Koordinaten sind mit der gleichen Genauigkeit aufgenommen, können aber aufgrund von fehlenden Antennen in der Vertikalen gegebenenfalls negative Werte annehmen.

Zusätzlich wurde eine Tabelle bereitgestellt, mit der die IDs der Sensoren den Spielern zugeordnet werden können, sowie Zeitpunkte für den Start und das Ende der beiden Halbzeiten. Außerdem wurden Videos der beiden Halbzeiten aufgenommen und zur Verfügung gestellt.

4.2 Visualisierung

Über die Benutzeroberfläche soll es dem Nutzer möglich sein, Kategorien für verschiedene interessante Events zu erstellen, den Kategorien ausgewählte Szenen zuzuordnen und anschließend den entstehenden Entscheidungsbaum anzuschauen. Deshalb wurde eine Visualisierung mit grundsätzlich drei Komponenten konzipiert. Die erste Komponente ist das Video des Spiels. Zum zweiten gibt es eine Liste aller erstellten Kategorien des Nutzers. Die dritte Komponente ist die sogenannte Timeline. Dort werden die ausgewählten Szenen angezeigt, zugeordnet zu ihren Kategorien. Eine vierte Komponente wird sichtbar, wenn man den Entscheidungsbaum einer Kategorie auswählt. Dieser wird in einem zusätzlichen Fenster angezeigt. Der Entscheidungsbaum würde anderweitig zu viel Platz vom Video oder von der Timeline wegnehmen. Dadurch wird außerdem der automatische Klassifikationsprozess optisch vom Rest der Anwendung getrennt.

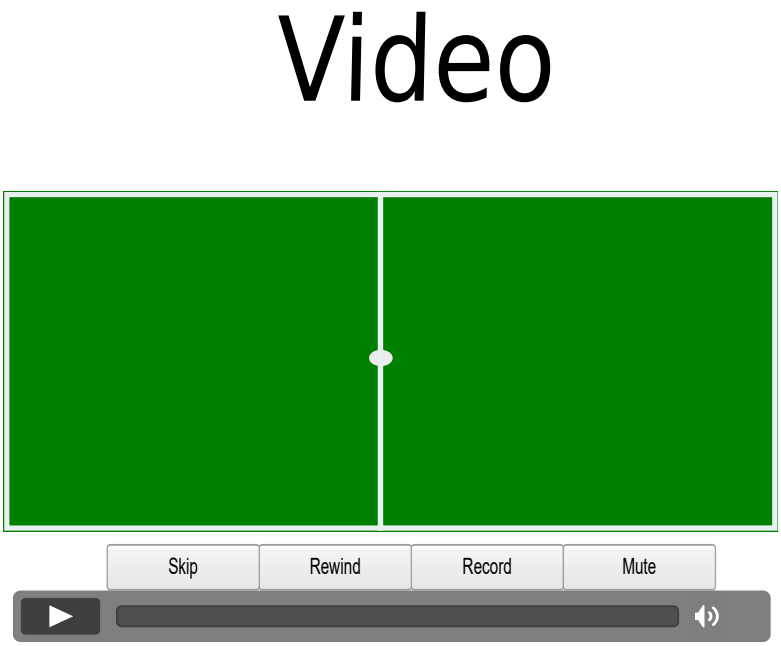


Abbildung 4.2: Konzept für den Videoplayer

4.2.1 Video

Das Video des Fußballspiels soll vom Nutzer zur Auswahl von Beispiel- und Gegenbeispielszenen verwendet werden. Das heißt, es wird ein Videoplayer benötigt, der folgende Funktionen zur Verfügung stellt:

- Spielen/Pausieren
- Vorspulen
- Zurückspulen
- Aufnehmen
- Stumm schalten
- Zu beliebigem Zeitpunkt springen

Diese Funktionen kann man skizzenhaft in Abbildung 4.2 sehen. Die Auswahl der Szenen soll dabei über die Aufnahme im Video funktionieren. Eine andere Möglichkeit wäre es, den Start- und Endzeitpunkt der ausgewählten Szenen über eine Maske eingeben zu lassen. Es ist aber um einiges einfacher und intuitiver, während dem Anschauen des Videos zu Beginn und Ende der Szene auf Aufnahme zu drücken. Durch Vor- und Zurückspulen kann man schnell die Umgebung einer Szene einschätzen. Man kann überprüfen, was vor und nach der Szene passiert, um das optimale Zeitfenster festzulegen. Zuletzt soll es über einen Zeitschieberegler möglich sein zu jedem Zeitpunkt des Videos zu springen. Dies ist die übliche Art und Weise aller bekannter Videoplayer, um diese Funktion umzusetzen.

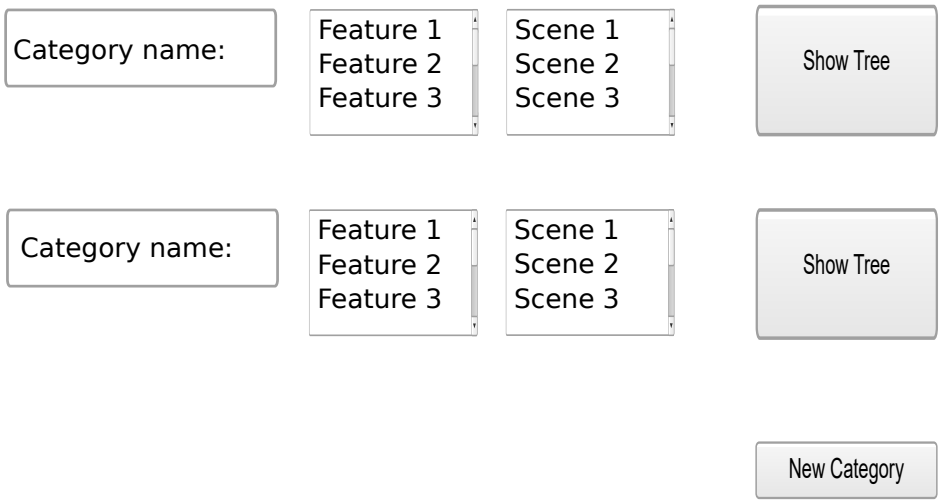


Abbildung 4.3: Konzept für Kategorien, die aus einem Namen und Listen der ausgewählten Features und Szenen bestehen.

4.2.2 Kategorie

Eine Kategorie symbolisiert eine Art von Event, die der Nutzer spezifizieren und finden möchte. Das können zum Beispiel Tore, Ecken oder Freistöße sein. Dem Nutzer soll es möglich sein

- beliebig viele Kategorien zu erstellen.
- ausgewählte Szenen zu verwalten.
- passende Features zu selektieren.

Diese Kategorien können beispielhaft in Abbildung 4.3 angeschaut werden, wobei jede Kategorie über ihren Namen identifiziert wird. Im Video aufgenommene Szenen sollen einer Kategorie zugeordnet werden. Diese Szenen können am einfachsten über eine Liste verwaltet werden. Eine andere Möglichkeit wäre es, die Kategorien in einer Tabelle zu speichern, wodurch leichter zu vergleichen wäre, welche Szenen und Features zwischen Kategorien gleich und unterschiedlich sind. In diesem Fall wurden Listen genommen, da der Vergleich der Szenen in der Timeline geschieht, welche im nächsten Abschnitt vorgestellt wird. Es soll außerdem jederzeit möglich sein, Szenen hinzuzufügen oder zu löschen. Weiterhin ist es wünschenswert, dass man sie später erneut anschauen kann, ohne wieder das gesamte Video zu durchsuchen.

Über die Kategorien soll es zudem möglich sein, die dazugehörigen Entscheidungsbäume berechnen und anzeigen zu lassen. Deshalb ist als weiteres Element eine Übersicht der vorhandenen Features notwendig. Gefragt ist eine Analyse, mit welchen Mitteln Events spezifiziert werden können. Auch muss der Nutzer die Entscheidungsfreiheit haben, anhand welcher dieser Features der Entscheidungsbaum erstellt werden soll. Damit kann der Nutzer Features von vornherein ausschließen, die aus seiner Sicht unpassend für seine Art von Event sind.

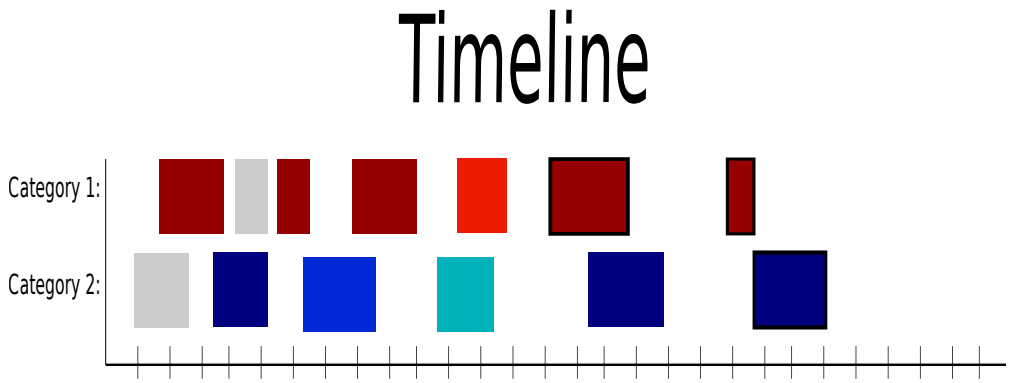


Abbildung 4.4: Konzept für die Timeline, die aus verschiedenen Arten von Szenen der Kategorien gebildet ist.

4.2.3 Timeline

Die Timeline dient als Übersicht über alle ausgewählten oder klassifizierten Szenen. Mit Hilfe der Timeline soll der Nutzer

- einen zeitlichen Überblick über die Szenen einer Kategorie erhalten.
- die Szenen von verschiedenen Kategorien unterscheiden und vergleichen können.
- Szenen anschauen und löschen können.
- die Konfidenz von Szenen bestimmen können.
- Eine Übersicht über das Ergebnis der Klassifikation in Form von markierten Szenen erhalten.

Dafür bietet sich ein einfacher Graph an, bei dem auf der y-Achse die Kategorien und auf der x-Achse die Zeit des Spiels abgebildet sind. Ein Konzept für diesen Graph kann in Abbildung 4.4 betrachtet werden. Fügt der Nutzer einer Kategorie eine neue Szene hinzu, so wird diese auch in der Timeline der Kategorie zugeordnet. Dadurch kann der Nutzer Szenen innerhalb einer Kategorie und gegenüber anderen Kategorien vergleichen.

Als nächstes soll der Nutzer ausdrücken können, wie gut die ausgewählte Szene zu seiner Kategorie gehört. Es könnte sinnvoll sein, wenn man zum Beispiel Tore analysieren möchte, auch Torschüsse hinzuzunehmen, damit mehr Beispiele vorhanden sind. In diesem Fall können diese Szenen mit einer geringeren Konfidenz gekennzeichnet werden. Insbesondere Gegenbeispiele zum Gesuchten sollen besonderes gekennzeichnet werden. Diese sind wichtig, da man für eine gute Klassifikation beides braucht, Beispiele und geeignete Gegenbeispiele. In dieser Arbeit werden diese unterschiedlichen Arten der Konfidenz mit Farben zu kodiert. Neben Farben wäre auch Form, Stil oder Größe für die Kodierung der Konfidenz möglich gewesen. Farben sind hierbei aber am einfachsten zu unterscheiden. Außerdem können mit verschiedenen Farben zusätzlich die Kategorien visuell unterschieden werden.

Eine besondere Rolle spielen die klassifizierten Szenen. Nachdem ein geeigneter Entscheidungsbaum erstellt wurde, werden die nicht ausgewählten Szenen des Spiels durch ihn klassifiziert. Dabei interessieren den Nutzer alle Szenen, die zusätzlich zu seiner Kategorie passen sollen. Diese Szenen müssen sich visuell von den selbst ausgewählten unterscheiden. Dazu gibt es verschiedene Möglichkeiten, wie weitere Farben, Größe, Kontur, Stil oder ein spezielles Icon. Für dieses Konzept wird ein markanter schwarzer Rand verwendet. Dies ist eine einfache, aber trotzdem effektive Möglichkeit, die zwei verschiedenen Arten von Szenen zu unterscheiden. Klassifizierte Szenen sollen anschließend vom Nutzer validiert werden. Deshalb soll es auch in der Timeline möglich sein Szenen zum Anschauen auszuwählen und zu löschen.

Als letztes ist die Interaktion mit der Timeline wichtig. Am Anfang möchte man einen Überblick über alle Kategorien und Szenen bekommen. Mit vielen Kategorien und Szenen kann dies aber schnell unübersichtlich werden. Deshalb soll es möglich sein einzelne Bereiche der Timeline auszuwählen, um sie genauer anzuschauen. Außerdem soll man mit Rein- und Rauszoomen die Tiefe der Details bestimmen können.

4.2.4 Entscheidungsbaum

Der Entscheidungsbaum dient als visuelles Modell für die Charakterisierung der ausgewählten Szenen. An ihm lässt sich erkennen, durch welche Features sich die ausgewählten Szenen von den Gegenbeispielen unterscheiden. Außerdem wird deutlich, welche Features das zu findende Event ausmachen. Weiterhin kann man die Wichtigkeit der verschiedenen Features erkennen. Das oberste Feature, die Wurzel des Baums, teilt die Szenen am besten in passende und nicht passende auf. Features, die sich weiter unten befinden, sind am besten, um die dortige Teilmenge der Szenen aufzuteilen.

Der wichtigste Unterschied dieser Arbeit im Vergleich zu anderen Arbeiten dieses Themengebiets ist es, dass der Nutzer die Möglichkeit hat, diesen Entscheidungsbaum nach seinen Vorstellungen anzupassen. Damit greift er direkt in den Klassifikationsprozess ein. Der automatisch erzeugte Entscheidungsbaum ist der optimale Entscheidungsbaum, der auf Grund alleine des aus den ausgewählten Szenen extrahierten Wissens erstellt werden kann. Der Nutzer kann mit der Manipulation des Entscheidungsbaums sein Expertenwissen in den Klassifikationsprozess einbringen. Für die Gestaltung des Entscheidungsbaums stehen dem Nutzer, wie man in Abbildung 4.5 durch drei Pfeile eingezeichnet sehen kann, drei Funktionen zur Verfügung. Er kann :

- an einem Knoten ein anderes Feature auswählen.
- an einer Kante einen anderen Wert für den Splitpunkt setzen.
- einen neuen Knoten in den Baum einfügen.

Mit diesen Möglichkeiten soll der Nutzer den Entscheidungsbaum nach seinen Vorstellungen gestalten können. Dadurch kann der Nutzer sein Wissen einbringen oder verschiedene Versionen ausprobieren und die Ergebnisse vergleichen.

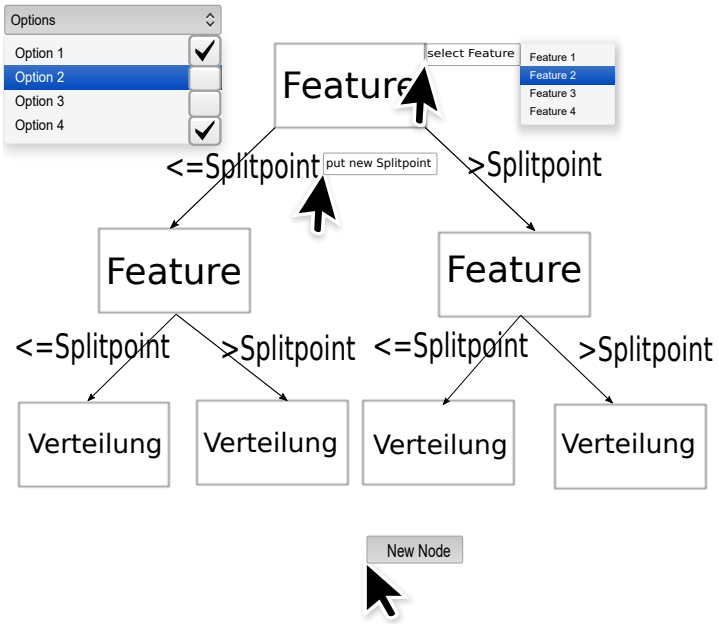


Abbildung 4.5: Konzept des Entscheidungsbaums mit Optionen und den verschiedenen Manipulationsmöglichkeiten.

Indem der Nutzer ein anderes Feature für einen Knoten auswählt, kann er seine Einschätzung bezüglich der wichtigsten Features zum Ausdruck bringen. Dadurch wird der Entscheidungsbaum anhand des neu ausgewählten Features aufgeteilt. Durch die Wahl eines eigenen Splitpoints kann außerdem diese Aufteilung beeinflusst werden. Als letztes soll durch die Erstellung eigener Knoten jede Freiheit in der Erweiterung des Entscheidungsbaums gegeben sein.

Bei allen diesen Veränderungen muss man aber bedenken, dass man vom optimalen Baum abweicht. Dies kann dazu führen, dass nutzlose Verzweigungen und Blätter entstehen. Damit sind Verzweigungen und dazugehörige Blätter gemeint, denen keine Szene zugeordnet sind, also auf dessen Pfad keine Teilmenge des Trainingsdatensatzes ankommt. Deshalb soll es dem Nutzer möglich sein gegebenenfalls Teile des Baums nach den Veränderungen automatisch anpassen zu lassen. Falls man mit dem Ergebnis nicht zufrieden ist, soll es außerdem möglich sein den Entscheidungsbaum wieder zurückzusetzen, um nochmal von vorne anzufangen.

Nachdem man den Entscheidungsbaum nach seinen Wünschen angepasst hat, kann man die restlichen Szenen klassifizieren. Dabei sollen dem Nutzer verschiedene Optionen zur Verfügung stehen. Er soll zum Beispiel die Möglichkeit haben die eine Halbzeit als Trainingsdatensatz zu verwenden und den entstandenen Entscheidungsbaum für die Klassifiaktion der anderen Halbzeit zu nutzen. Außerdem kann der Nutzer alle klassifizierten Szenen automatisch wieder entfernen lassen. Damit lassen sich die Ergebnisse verschiedener Entscheidungsbäume einfacher vergleichen.

5 Implementierung

Dieses Kapitel widmet sich der konkreten Umsetzung des entwickelten Konzepts. Dazu wird die für diese Arbeit implementierte Anwendung vorgestellt. Die Implementierung der verschiedenen Komponenten und die dafür verwendeten Technologien und Bibliotheken werden beschrieben. Das resultierende System ist eine Desktop Anwendung, welche in Java geschrieben wurde.

5.1 Benutzeroberfläche

Die Benutzeroberfläche des Programms ist in Abbildung 5.1 zu sehen. Sie besteht aus drei Teilen. Oben links (1) befindet sich die Liste der Kategorien, die der Nutzer erstellt hat. Oben rechts (2) ist das Video zu sehen und unten ist die Timeline (3) mit den eingetragenen Szenen.

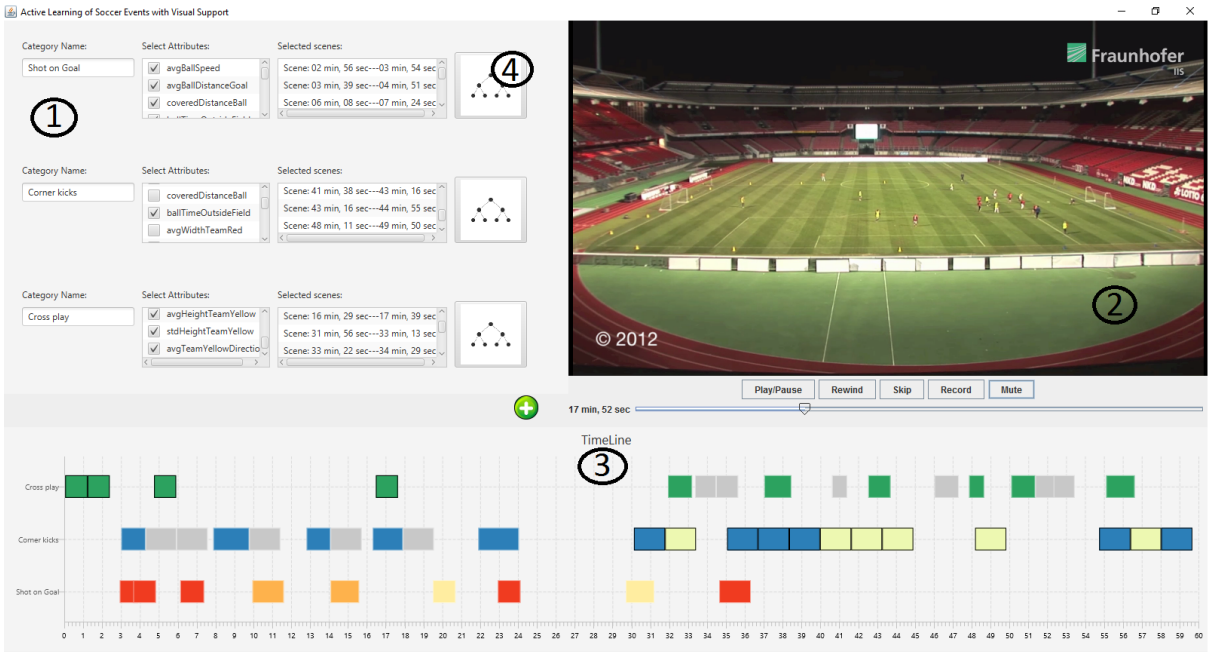


Abbildung 5.1: Benutzeroberfläche mit den vom Nutzer erstellten Kategorien (1), dem Video (2) und der Timeline (3).

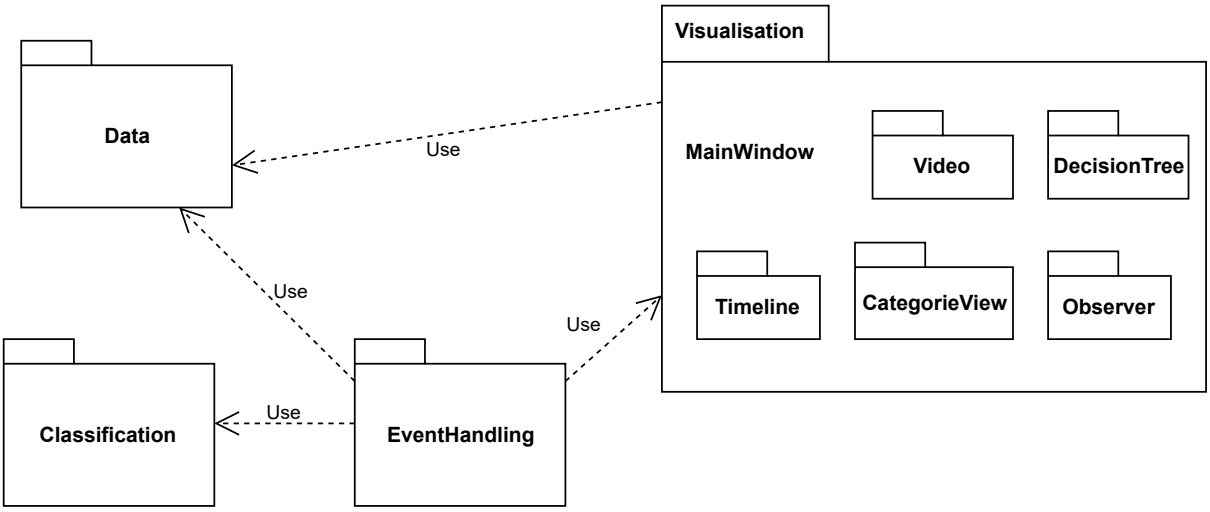


Abbildung 5.2: Paketdiagramm des entwickelten Systems

Ein zusätzliches Fenster öffnet sich, sobald man auf das Baum-Symbol (4) einer Kategorie klickt. In diesem wird der zur Kategorie gehörende Entscheidungsbaum angezeigt.

5.2 Systemstruktur

Das System ist nach dem Model-View-Controller-Prinzip aufgebaut. Dieses Architekturmuster wird genutzt, um die Daten von ihrer Präsentation und Steuerung zu trennen. Im Model sind nur die Daten, in dieser Arbeit die Eventdaten der DEBS 2013 Grand Challenge. Diese werden im View präsentiert. Dies geschieht hier durch das Video, die Features und die ausgewählten Szenen. Der Controller ist für die Zusammenarbeit dieser beiden Komponenten zuständig. Interaktionen des Benutzers im View führen zu Änderungen im Model, wodurch der View angepasst wird.

Die Paketstruktur des entwickelten Systems kann in Abbildung 5.2 betrachtet werden. Das Paket Data entspricht hierbei dem Model, das Visualisation-Paket dem View und das EventHandling-Paket dem Controller. Im Folgenden werden diese Komponenten im Einzelnen vorgestellt.

5.3 Datenverarbeitung

Der bereitgestellte Datensatz umfasst 4,1 GB und fast 50.000 Events. Wenn man versucht, diese komplette Menge an Daten in ein System zu laden, stoßen schon Computer mit 8 GB RAM-Speicher an ihre Grenzen. Deshalb wurden in einem ersten Schritt alle Events auf eine Frequenz von 100 Hz reduziert.

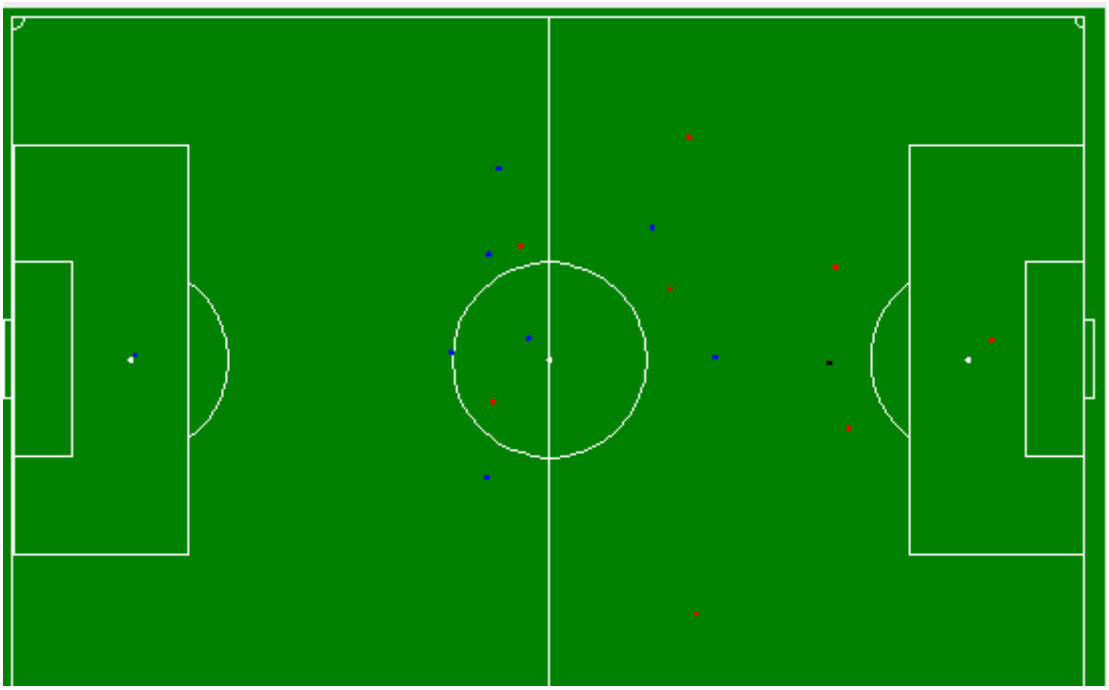


Abbildung 5.3: Repräsentation des Fußballspiels in 2D; Spieler werden durch Punkte repräsentiert; Die beiden Mannschaften durch verschiedene Farben

Um zu verifizieren, dass die Daten trotzdem die benötigte Genauigkeit haben, wurde ein kleines Programm geschrieben, mit dem man das Fußballspiel in 2D anschauen kann. Dazu wurde eine Liste mit allen Positionen der Spieler und des Balls erstellt. Diese wird anschließend durchgegangen und alle Punkte auf dem Feld werden aktualisiert. Auch mit einer Frequenz von nur noch 100 Hz wird das dadurch entstehende Spiel flüssig angezeigt. Ein Bild dieser Anwendung ist in Abbildung 5.3 zu sehen.

Anschließend wurden alle Events anhand ihrer SensorenID den dazugehörigen Spielern zugeordnet. Dabei wurden die Spieler zusätzlich ihrem jeweiligen Team zugeordnet. Diese Liste an Spielern, mit den dazugehörigen Events, wurde dann in einer Binärdatei gespeichert, welche bequem bei jedem Start des Programms eingelesen werden kann. Außerdem werden die Metadaten über die Abgrenzung des Spielfeldes sowie die Start- und Endzeitpunkte der beiden Halbzeiten eingelesen und gespeichert.

5.4 Features

Nachdem die Daten in das System importiert wurden, wurden daraus Features berechnet. Damit sind Eigenschaften gemeint, die eine Szene eines Fußballspiels beschreiben. Wie man in Tabelle 5.1 sieht, wurden 14 Features implementiert.

Abkürzung	Erklärung	Einheit
avgBallSpeed	Durchschnittliche Geschwindigkeit des Balls	km/s
avgBallBistanceGoal	Durchschnittliche Distanz des Balls zum Tor	m
coveredDistanceBall	Zurückgelegte Strecke des Balls	m
ballTimeOutsideField	Zeit, die der Ball außerhalb des Spielfeldes war	s
avgWidthTeamRed	Durchschnittliche Breite des roten Teams	m
stdWidthTeamRed	Standardabweichung der Breite des roten Teams	m
avgHeightTeamRed	Durchschnittliche Höhe des roten Teams	m
stdHeightTeamRed	Standardabweichung der Höhe des roten Teams	m
avgTeamRedDirection	Durchschnittlich zurückgelegte Distanz des roten Teams	m
avgWidthTeamYellow	Durchschnittliche Breite des gelben Teams	m
stdWidthTeamYellow	Standardabweichung der Breite des gelben Teams	m
avgHeightTeamYellow	Durchschnittliche Höhe des gelben Teams	m
stdHeightTeamYellow	Standardabweichung der Höhe des gelben Teams	m
avgTeamYellowDirection	Durchschnittlich zurückgelegte Distanz des gelben Teams	m

Tabelle 5.1: Tabelle der implementierten Features

Diese Features werden für jede ausgewählte und zu klassifizierende Szene berechnet. Dazu müssen als erstes alle Events aus den Daten extrahiert werden, die sich im Zeitraum der betrachteten Szenen befinden. Für die Features des Balls werden daraufhin alle Events von Bällen, die sich außerhalb des Spielfelds befinden, ausgeschlossen. Für das genutzte Fußballspiel wurden vier verschiedene Bälle verwendet, weshalb immer derjenige identifiziert werden musste, der gerade im Spiel ist. Die übriggebliebenen Events wurden zur Berechnung der durchschnittlichen Geschwindigkeit, der durchschnittlichen Distanz zum Tor und der zurückgelegten Strecke benutzt. Für die durchschnittliche Distanz zum Tor wurde immer die Distanz zu beiden Toren berechnet und am Ende die geringere der beiden verwendet. Außerdem wurden alle Zeitabschnitte der Szenen, in der kein Ball im Feld ist, zusammengerechnet, was die Zeit des Balls außerhalb des Feldes ergibt.

Die Features der beiden Mannschaften wurden gleich berechnet. Für die durchschnittliche Breite einer Mannschaft wurden die Positionen aller Spieler, außer dem Torwart, genommen und die Distanz zwischen den beiden am weitesten in x-Richtung auseinander stehenden Spieler berechnet. Dasselbe wurde für die durchschnittliche Höhe der Mannschaft mit den am weitesten auseinander stehenden Spieler in y-Richtung gemacht. Gleichzeitig wurde die Standardabweichung dieser beiden Werte berechnet. Die durchschnittlich zurückgelegte Distanz einer Mannschaft wurde ermittelt, indem über die durchschnittlich zurückgelegte Strecke in x-Richtung jedes Spielers summiert wurde.

Das wichtigste Objekt im Fußball ist der Ball. Ereignisse im Fußball finden immer dort statt, wo sich der Ball befindet. Deshalb sind die implementierten Ballfeatures wichtig, um jede Art von

Event zu beschreiben. Geschwindigkeit und Distanz zum Tor sind interessant für Angriffsszenen. Die Zeit, in der sich der Ball im Aus befindet, ist vor allem für Standardsituationen wichtig. Die zurückgelegte Strecke kann außerdem für längere Spielzüge bedeutungsvoll sein. Mit den Mannschaftsfeatures kann man herausfinden, wie eine Mannschaft aufgestellt ist. Damit kann unterschieden werden, ob sie weiträumig verteilt ist oder ob alle Spieler auf engem Raum sind. Außerdem wird erkannt, ob das Spiel in die Breite gezogen wird und ob bevorzugt über die Mitte oder Außen gespielt wird. Die durchschnittlich zurückgelegte Distanz einer Mannschaft soll zudem ausdrücken, welches Bewegungsverhalten eine Mannschaft zeigt, also ob sie sich mehr im Angriff oder in der Verteidigung befindet.

5.5 Visualisierung

Das Visualisierungspaket beinhaltet alle in der Benutzeroberfläche zu sehenden Komponenten. Diese sind Bestandteil des MainWindows (Abbildung 5.2). Das MainWindow steuert zusätzlich die Kommunikation zwischen den verschiedenen Views. Einige Funktionen haben Auswirkungen auf mehrere Views, wie zum Beispiel das Aufnehmen einer Szene, welche sowohl in der Timeline als auch in der Kategorie gespeichert wird. Für diese Kommunikation wird das Vermittler-Muster [GJHV11] genutzt. Dieses Entwurfsmuster dient dazu verschiedene Objekte, die die gleichen Daten repräsentieren, zu aktualisieren, falls sich eines davon ändert. Dazu gibt es einen zentralen Vermittler, der bei allen Objekten als Beobachter angemeldet ist. Dieser wird informiert, wenn sich etwas an diesem Objekt verändert. In dieser Arbeit ist das MainWindow dieser Vermittler. Es wird informiert, sobald sich etwas in einem View verändert oder eine Funktion ausgeführt wird. Beim Informieren des Vermittlers wird zusätzlich die Änderung, in Form eines Updates, mit übergeben. Damit werden die unterschiedlichen Views immer synchron gehalten. Da die angebotenen Funktionen nicht nur Änderungen in den Views, sondern zusätzlich auch in den dahinterliegenden Modellen des Klassifikators und der Events haben, wird dieses zentral verwaltete Muster verwendet.

5.5.1 Video

Das Abspielen des Videos wurde mit Hilfe von vlcj[1] umgesetzt. Dies ist ein Framework, um den VLC Media Player in Java AWT und Swing Komponenten umzusetzen. Es unterstützt jegliche Videoformate und bietet alle Funktionen, die auch der VLC Media Player zur Verfügung stellt.

Zum Datensatz gehören neben den Events auch die Videos der ersten und zweiten Halbzeit. Zur einfacheren Handhabung wurden diese zuerst zu einem Video zusammengefügt. Anschließend wurde dieses Video mit dem vlcj Framework in eine Java Swing Komponente eingebettet.

[1] http://capricasoftware.co.uk//projects/vlcj

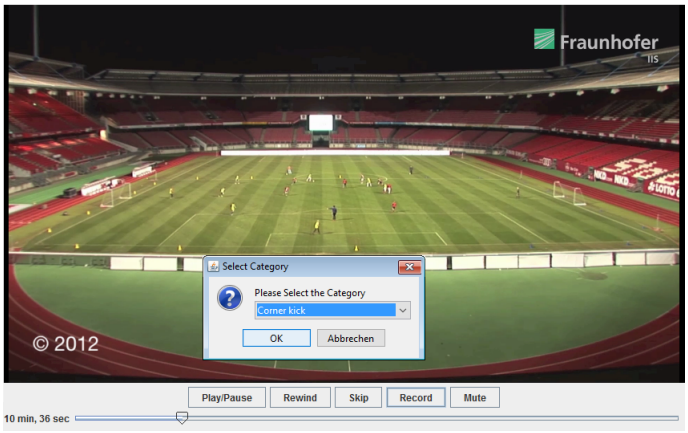


Abbildung 5.4: Zuordnen einer aufgenommenen Szene zu einer Kategorie

Zusätzlich wurden alle benötigten Funktionen des Videoplayers implementiert. Dies beinhaltet das Abspielen, Pausieren, Vor- und Zurückspulen sowie Stummschalten des Videos. Für das Aufnehmen der Szenen gibt es außerdem einen Record-Button. Während man das Video anschaut, können Szenen aufgenommen werden, indem man entweder auf den Record-Button drückt oder die R-Taste auf der Tastatur klickt. Mit dem ersten Klick wird die Aufnahme gestartet und mit einem zweiten Klick wieder beendet. Anschließend muss man in einem Dialog auswählen, zu welcher Kategorie diese Szene gehört, was man in Abbildung 5.4 sehen kann. Als letztes wurde zusätzlich ein Zeitschieberegler implementiert, mit dem man jederzeit an eine beliebige Stelle im Video springen kann.

5.5.2 CategorieView

Der CategorieView besteht aus einer Liste von Kategorien. Dieser kann in Abbildung 5.5 nochmal genauer betrachtet werden.

Eine neue Kategorie wird vom Nutzer durch einen Klick auf das + Icon[2] erzeugt. Jede Kategorie kann durch einen Rechtsklick und die Wahl von ”delete Category” wieder gelöscht werden. Eine Kategorie wird durch ihren Namen identifiziert, der jederzeit vom Nutzer angepasst werden kann. Außerdem beinhaltet eine Kategorie die Liste aller aufgenommenen und klassifizierten Szenen. Mit einem Rechtsklick auf eine Szene kann man diese erneut anschauen oder löschen. Außerdem ist es möglich alle Szenen, die automatisch klassifiziert wurden, zu löschen. Damit muss man nicht jede einzelne durchgehen und manuell anpassen. Es bleibt dem Nutzer überlassen, welche und wie viele er validiert.

[2]Urheber:Hopstarter; http://www.iconarchive.com/show/button-icons-by-hopstarter/Button-Add-icon.html; Lizenz: CC Attribution-Noncommercial-No Derivate 4.0

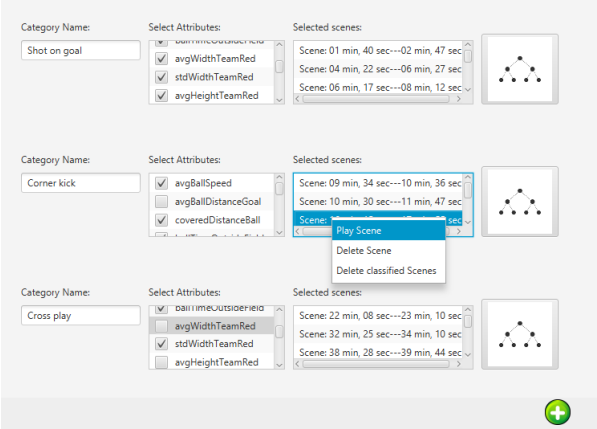


Abbildung 5.5: Auswahlmöglichkeiten von Funktionen zur Szenenverwaltung

In einer zweiten Liste werden alle zur Verfügung stehenden Features angezeigt. Dabei kann man durch Comboboxen auswählen, welche zur Erzeugung des Entscheidungsbaums genutzt werden sollen. Damit ist es möglich, einige Features von vornherein auszuschließen.

Das letzte Element, dass mit einer Kategorie assoziiert ist, ist der Entscheidungsbaum. Dieser kann erzeugt werden, indem man auf den Button mit dem Baum-Symbol klickt. Durch diesen Klick öffnet sich ein neues Fenster, in dem der berechnete Entscheidungsbaum (siehe Kapitel 5.6), ausgehend von den ausgewählten Beispielszenen, angezeigt wird.

Alle erstellten Kategorien, mit den dazugehörigen Attributen, werden beim Beenden der Anwendung gespeichert, um sie beim nächsten Öffnen laden zu können.

5.5.3 Timeline

Die Timeline wird durch ein x-y-Diagramm realisiert. Jede Kategorie wird durch ihren Namen auf der y-Achse repräsentiert. Außerdem wird jede Kategorie mit einer eigenen Farbe identifiziert. Dadurch soll die Unterscheidung der verschiedenen Kategorien einfacher sein. Eine Szene wird durch ein Rechteck, in der Farbe der Kategorie, repräsentiert, dessen Breite die Länge der Szene widerspiegelt. Die Konfidenz einer Szene, also wie gut die Szene zum gesuchten Event passt, wird durch ihren Farbton ausgedrückt. Dabei bedeutet der dunkelste Farbton, dass die Konfidenz am höchsten ist. Wie man zum Beispiel in Abbildung 5.6 sehen kann, ist für die Kategorie ”Shot on Goal” rot die Farbe für die höchste Konfidenz, orange für die zweithöchste und ein helles orange für die niedrigste. Diese Einstellung kann durch Rechtsklick auf eine Szene und anschließend durch die Wahl von ”Select Konfidenz” geändert werden. Neben den drei Stufen für die Konfidenz einer Szene gibt es eine vierte Möglichkeit, um Gegenbeispiele zu kennzeichnen. Gegenbeispiel werden einheitlich in grau angezeigt.

Die Validation der klassifizierten Szenen, markiert durch einen schwarzen Rand, geschieht gleichermaßen durch das ”Select Konfidenz” Menü. Dabei werden auch klassifizierte Szenen,

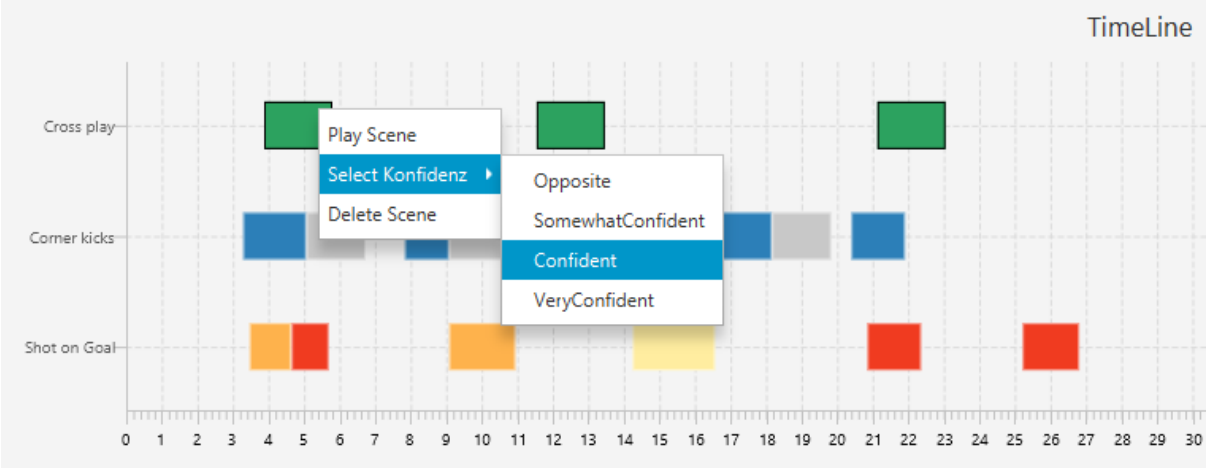
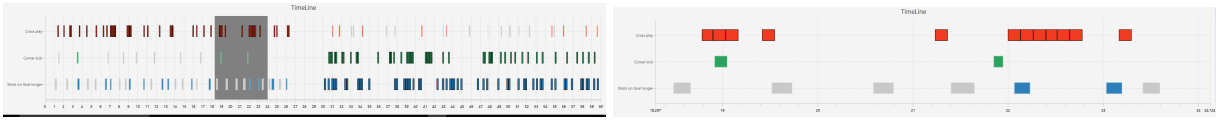


Abbildung 5.6: Ausschnitt der Timeline mit Funktionenmenü der Szenen

nach der Konfidenz des Algorithmus in die Klassifikation, mit dem gleichen Farbschema markiert. Wurde eine Szene einmal validiert, wird der schwarze Rand entfernt und diese Szenen wird wie jede selbst ausgewählte Szene behandelt.

Um die Interaktion mit den Szenen zu erleichtern, können Bereiche der Timeline ausgewählt und vergrößert werden. Mit gedrückter linker Maustaste wird ein Rechteck über den ausgewählten Bereich gezogen, in den nach dem Loslassen hineingezoomt wird. Dies kann man in Abbildung 5.7 sehen. Rein- und wieder herauszoomen kann man zusätzlich durch Drehen des Mausrads.



(a) Markierung eines Bereichs der Timeline **(b)** Ausgewählter Bereich der Timeline

Abbildung 5.7: Ein Bereich der Timeline kann vergrößert werden, indem man (mit der linken Maustaste gedrückt) ein Rechteck über dem gewünschten Bereich erstellt

5.6 Entscheidungsbaum

Der Entscheidungsbaum besteht aus zwei Teilen. Zum einen hat man die visuelle Repräsentation des Baums, zum anderen die automatische Berechnung desselben im Hintergrund. Beide Teile beeinflussen sich dabei gegenseitig. Zuerst wird der Entscheidungsbaum automatisch berechnet und das Ergebnis angezeigt. Anschließend kann der Nutzer den Baum durch interaktion verändern. Diese Veränderungen müssen in den Klassifikationsalgorithmus übertragen werden.

5.6.1 Klassifikationsalgorithmus

Der Entscheidungsbaum wird im Hintergrund durch Weka[3] umgesetzt. Das heißt, die ausgewählten Szenen werden in Trainingsdaten übersetzt und ein in Weka vorhandener Algorithmus nutzt diese für die Erstellung des Entscheidungsbaums. Weka beinhaltet eine Sammlung aller gängiger Algorithmen des maschinellen Lernens. Neben einer Desktop Anwendung bietet Weka auch eine Java Bibliothek an. In dieser Arbeit wurde die Java Implementierung des C 4.5 Algorithmus von Weka benutzt, um den Entscheidungsbaum zu berechnen. Für die Datenrepräsentation arbeitet Weka grundsätzlich mit dem .arff[4] Dateiformat. Ein Datensatz wird durch eine Relation gekennzeichnet. Weiterhin müssen alle Attribute mit ihrem Typ im Voraus definiert sein. Dabei muss ein Klassenattribut ausgewählt werden, hinsichtlich welchem alle Datensätze klassifiziert werden sollen. Anschließend werden die Datensätze durch Angabe der Werte für die verschiedenen Attribute eingegeben. In Java wird dieses Format mit sogenannten Instanzen repräsentiert.

Bevor der Entscheidungsbaum erstellt werden kann, müssen alle ausgewählten Szenen dementsprechend in Instanzen umgewandelt werden. Dafür werden alle Features für die Szene berechnet und zu einer Instanz zusammengefasst. Falls ein Feature nicht berechnet werden konnte, zum Beispiel auf Grund fehlender Sensordaten, wird es als fehlend mit einem ”?” angegeben. Falls der Nutzer nicht selbständig Gegenbeispiele für seine Kategorie ausgewählt hat, werden diese automatisch berechnet. Dazu wird die durchschnittliche Szenenlänge der ausgewählten Szenen genommen und bisher nicht ausgewählte Szenen dieser Länge als Gegenbeispiele definiert. Dabei werden Szenen bevorzugt, die zwischen ausgewählten Szenen liegen. Es kann angenommen werden, dass der Nutzer diese bewusst nicht als Beispielszenen ausgewählt hat. In diesem Fall werden so viele Gegenbeispiele automatisch erzeugt, wie es auch ausgewählte positive Beispiele gibt.

Nachdem Instanzen für alle benötigten Szenen erstellt wurden, kann der Algorithmus angewandt werden. Dazu bietet Weka verschiedene Optionen an. Unter anderem kann man auswählen, ob und welcher Pruning-Prozess genutzt werden soll, was für Splits benutzt werden sollen und in welcher Art der Information Gain berechnet werden soll. In dieser Arbeit wurde kein Pruning benutzt, da die Entscheidungsbäume bei kleinen Trainingsdatensätzen auch klein ausfallen. Außerhalb dieser Optionen bietet Weka keine Möglichkeiten an, in den Klassifikationsprozess einzugreifen. Auch den resultierenden Entscheidungsbaum kann man nicht manuell verändern. Um die gewünschten Interaktionen des Nutzers mit dem Entscheidungsbaum zu ermöglichen, musste deshalb die Weka Bibliothek angepasst werden.

Für die spätere Klassifikation der übrigen Szenen wird dasselbe Prinzip wie für die Gegenbeispiele verwendet. Alle nicht ausgewählten Szenen werden anhand der durchschnittlichen Szenenlänge identifiziert und anschließend dem Entscheidungsbaum zur Klassifikation übergeben. Als Ergebnis erhält man für jede mögliche Klasse eine Zahl zwischen 0 und 1. Damit

[3] http://www.cs.waikato.ac.nz/ml/weka/
[4] http://weka.wikispaces.com/ARFF

wird die Wahrscheinlichkeit ausgedrückt, mit der die Instanz zu der jeweiligen Klasse gehört. Die Konfidenz der klassifizierten Szenen wird nach diesem Ergebnis bestimmt. Mit 100% zugeordnete Szenen werden mit der höchsten Konfidenz markiert, ab 75% mit der zweithöchsten und ab 50% mit dem dritten Konfidenzlevel.

5.6.2 Anpassung der Weka Bibliothek

Der in Weka implementierte Algorithmus berechnet viele Eigenschaften des Baums erst im Verlauf der Durchführung. So werden die Art des Splitpoints und die Teilmengen des Trainingsdatensatzes an den jeweiligen Knoten erst bei der rekursiven Erstellung des Baums definiert. Damit man den Entscheidungsbaum manuell anpassen kann, wurden deshalb einige Methoden der Weka Bibliothek umgeschrieben, sodass man öffentlich auf sie zugreifen kann. Außerdem wurden Getter und Setter Methoden für Attribute eingefügt, die bisher keine hatten.

Um Knoten und Kanten des Entscheidungsbaums zu verändern oder neu hinzuzufügen, muss man sie als erstes identifizieren können. Dafür wurden sowohl die Knoten in der visuellen Repräsentation des Baums als auch in Weka mit einer einheitlichen ID versehen. Nachdem man den zu verändernden Knoten damit gefunden hat, wird in dessen lokalem Klassifikationsmodell das Feature bzw. der Splitpoint neu gesetzt. Anschließend muss die Aufteilung der Trainingsdaten in diesem Knoten anhand der Veränderung neu berechnet werden. Dies muss schließlich rekursiv in allen Unterteilbäumen des Knotens angepasst werden.

Wenn der Nutzer ein anderes Feature oder Splitpoint wählt, kann es vorkommen, dass der vom Algorithmus berechnete Information Gain unterhalb der unteren Schranke liegt. Um die Entscheidungsfreiheit des Nutzers zu gewährleisten, wurde deshalb eine Bedingung eingefügt, die bestimmt, dass jede Veränderung des Nutzers übernommen wird.

5.6.3 Visuelle Repräsentation

Nach der automatischen Erstellung des Entscheidungsbaums wird dieser dem Nutzer angezeigt. In dieser Arbeit wird dafür JGraphX[5] verwendet. JGraphX ist eine Bibliothek, mit der man Graphen in Java Swing Komponenten umsetzen kann. Ein Beispiel für diese Umsetzung ist in Abbildung 5.8 zu sehen.

In einem Knoten steht das dort verwendete Feature. Die Kanten werden durch die jeweiligen Splitpoints gekennzeichnet. Ein Blatt repräsentiert eine Klasse. Dabei bezeichnet ”true” die Klasse von Szenen, die erkannt werden möchte. Auf der anderen Seite beschreibt ”false” alle Szenen, die nicht zu den ausgewählten Beispielszenen passen. Die dazugehörigen Zahlen beschreiben, wie viele Szenen des Trainingsdatensatzes an diesem Blatt enden. Falls zwei

[5] https://github.com/jgraph/jgraphx

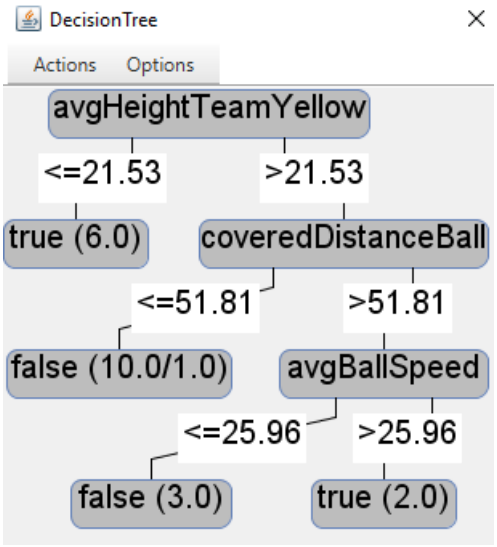
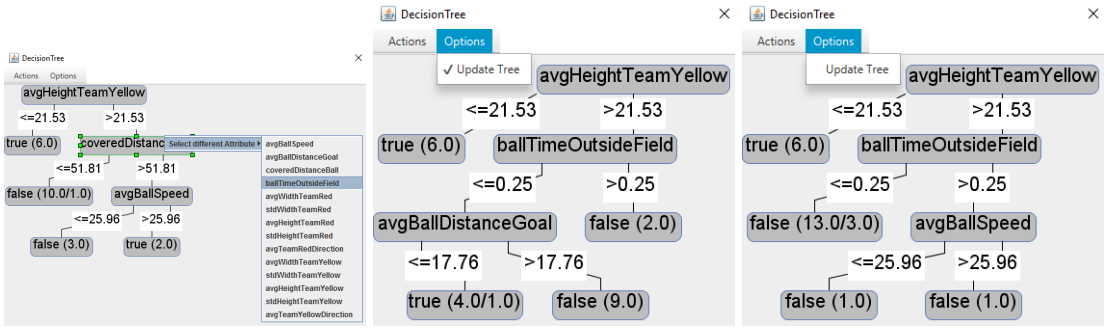


Abbildung 5.8: Beispiel eines erstellten Entscheidungsbaums

Zahlen der Form (X/Y) vorhanden sind bedeutet X, wie viele Szenen an diesem Blatt ankommen, und Y, wie viele Szenen an diesem Blatt nicht zur spezifizierten Klasse gehören.

Ein anderes Feature wählt man aus, indem man mit Rechtsklick auf einen Knoten klickt und in der Liste von Features das gewünschte selektiert. Wählt man an einem Knoten ein anderes Feature aus, wird der Entscheidungsbaum an dieser Stelle anhand des ausgewählten Features aufgeteilt. Der optimale Splitpoint für die Aufteilung wird dabei automatisch berechnet. Für die Aufteilung kann man außerdem auswählen, ob der Teilbaum unter dem veränderten Knoten erhalten bleiben soll oder mit dem ausgewählten Feature neu berechnet werden soll. Diese Wahl kann über den Menüpunkt ”Options” durchgeführt werden, wobei ein Häkchen die zweite Option bedeutet. Falls die Option ausgewählt wird, dass der Teilbaum erhalten werden soll, werden die Szenen anhand des berechneten Splitpoints aufgeteilt. Falls sich in dem Teilbaum noch weitere Knoten befinden, die die Teilmengen an Szenen aufteilen, bleiben diese erhalten. Einzig die Aufteilung wird neu berechnet. Dies kann dazu führen, dass an manchen Blättern des Baums keine Szenen eingeordnet werden können. Damit sind diese Blätter im Grunde nutzlos, da sie in keiner Weise zur Klassifizierung beitragen. Wenn die Option für eine neue Berechnung des Teilbaums unter dem ausgewählten Knoten selektiert wird, wird dieser Teilbaum automatisch neu berechnet. Dadurch wird der Entscheidungsbaum unter Einbeziehung des veränderten Features optimal berechnet. Trotzdem kann es möglich sein, dass Blattknoten ohne Funktion entstehen, wenn es keinen guten Splitpoint für das ausgewählt Feature gibt. Letztendlich hat der Nutzer durch diese Optionen die Freiheit, den Baum zu gestalten, wie er möchte. Er kann entweder alles selbst bestimmen oder Teile wieder automatisch berechnen lassen. Diese Unterschiede können in Abbildung 5.9 betrachtet werden. In Abbildung 5.9b sieht man, wie der Teilbaum unter dem veränderten Knoten automatisch angepasst wird. Dagegen wird in Abbildung 5.9c nichts an diesem Teilbaum, außer der Verteilung, verändert. Dadurch



(a) Auswahl eines anderen Features durch Rechtsklick auf einen Knoten

(b) Veränderter Baum mit Option "updateTree"

(c) Veränderter Baum ohne Option "updateTree"

Abbildung 5.9: Auswahl eines anderen Features und die Ergebnisse je nach Optionswahl

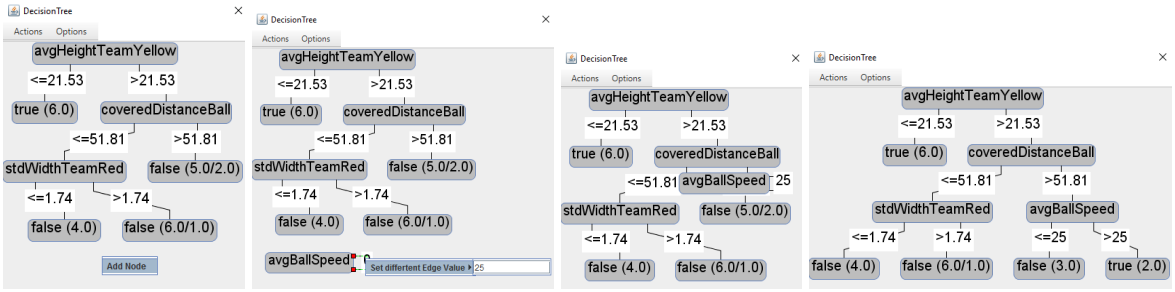
entstehen unter einem Knoten zwei Blätter mit den gleichen Klassen, weshalb dieser Knoten durch die Wahl des neuen Features überflüssig geworden ist.

Einen anderen Splitpoint definiert man analog zum anderen Feature, indem man auf einer Kante mit der rechten Maustaste klickt und den neuen Wert eingibt. Bestimmt man an einer Kante einen neuen Splitpoint, dann werden die Szenen an dem dazugehörigen Startknoten anhand dieses Splitpoints aufgeteilt. Dabei ist es egal, an welcher Kante man einen neuen Wert eingibt, da die Aufteilung immer in größer und kleiner gleich gemacht wird. Auch hier wird der darunterliegende Teilbaum je nach ausgewählter Option entweder beibehalten oder neu berechnet. Dazu werden die Szenen am Vaterknoten der Kante neu anhand des definierten Splitpoints aufgeteilt. Hier besteht eine noch höhere Gefahr, Blattknoten zu kreieren, denen keine Szene zugeordnet werden kann. Falls zum Beispiel alle Szenen einen höheren Wert als der bestimmte Splitpoint haben, wird die Kante mit kleiner gleich praktisch überflüssig. Einen neuen Splitpoint zu setzen ist ein größerer Eingriff, als ein anderes Feature auszuwählen, da in letzterem Fall zumindest der Splitpoint automatisch berechnet wird.

Die letzte Option ist es, einen neuen Knoten einzufügen, was man in Abbildung 5.10 sieht. Dazu benutzt man die rechte Maustaste irgendwo im Fenster, außerhalb des Entscheidungsbaums, und klickt auf "Add Node". Anschließend kann das Feature und der Splitpoint definiert werden. Daraufhin wird der neue Knoten mit drag & drop in den Entscheidungsbaum gezogen. Der Knoten muss über der gewünschten Kante losgelassen werden. Dadurch wir der Knoten genau an dieser Stelle in den Entscheidungsbaum eingefügt und der Teilbaum unter dem eingefügten Knoten wird dementsprechend angepasst.

Jederzeit hat man die Möglichkeit, den Baum wieder zurückzusetzen. Indem man unter "Actions/reset Tree" auswählt, wird wieder der automatisch berechnete Entscheidungsbaum angezeigt.

Sobald man mit dem Entscheidungsbaum zufrieden ist, kann man ihn anwenden. Dazu wählt man unter "Actions/classify" aus. Dabei hat man die Möglichkeit, jeweils nur eine Halbzeit



(a) Erstellung eines neuen Knotens

(b) Setzen des Splitpoints des Knotens

(c) Einfügen des Knotens mit drag & drop

(d) Entscheidungsbaum mit eingefügtem Knoten

Abbildung 5.10: Erstellen und einfügen eines neuen Knotens in den Entscheidungsbaum

oder das gesamte Spiel zu klassifizieren. Damit werden alle bisher nicht verwendeten Szenen anhand des Entscheidungsbaums klassifiziert. Das Ergebnis wird anschließend in der Timeline angezeigt und kann analysiert werden. Als letztes ist es über den Menüpunkt ”Actions” möglich, die klassifizierten Szenen wieder automatisch zu löschen.

6 Evaluation

In diesem Kapitel wird das implementierte System evaluiert. Nach dem Durchgehen einiger typischer Anwendungsfälle werden die entstehenden Ergebnisse sowohl qualitativ als auch quantitativ ausgewertet. Dazu werden Kategorien für interessante Events eines Fußballspiels mit Hilfe von Beispielszenen erstellt. Der dadurch entstehende Entscheidungsbaum und die damit gefundenen klassifizierten Szenen werden ausgewertet. Zunächst wurde das Fußballspiel angeschaut und alle relevanten Szenen manuell aufgezeichnet. Für die Auswertung müssen dann die klassifizierten Szenen mit den aufgezeichneten Szenen verglichen werden. Dabei wird grundsätzlich unterschieden, ob die klassifizierte Szene einer aufgezeichneten voll entspricht, überhaupt nicht entspricht oder zumindest eine ähnliche Szene repräsentiert. Wenn zwei oder drei klassifizierte Szenen direkt aufeinander folgen, werden sie als eine Szene betrachtet und bewertet. Sobald es mehr sind, werden diese Blöcke an Szenen wiederum aufgeteilt.

6.1 Torschüsse

Zu den wichtigsten Events eines Fußballspiels gehören Torschüsse und daraus entstehende Tore. Torschüsse sind sowohl für eine Zusammenfassung des Spiels als auch für die Analyse der Spieler und Mannschaften interessant. Dabei kann man untersuchen, wie solche Torszenen entstehen und welche Spieler beteiligt sind. Daraufhin können erkannte Muster trainiert werden, sowohl für das Angriffsspiel als auch die Verteidigung.

Für die DEBS 2013 Grand Challenge wurden alle Torschüsse des Spiels manuell aufgezeichnet. Diese Daten wurden genutzt, um Beispielszenen aus der ersten Hälfte des Spiels auszuwählen und anschließend die vom System gefundenen Szenen in der zweiten Hälfte auszuwerten.

In der Durchführung wurde als erstes eine neue Kategorie erstellt. Anschließend wurden die Torschussszenen der ersten Halbzeit durchgegangen, im Video aufgenommen und der Kategorie zugeordnet. Danach wurden Gegenbeispiele ausgewählt. Mit Hilfe des resultierenden Entscheidungsbaums wurde die zweite Halbzeit klassifiziert. Die dadurch gefundenen Szenen wurden mit den aufgezeichneten Daten verglichen und bewertet.

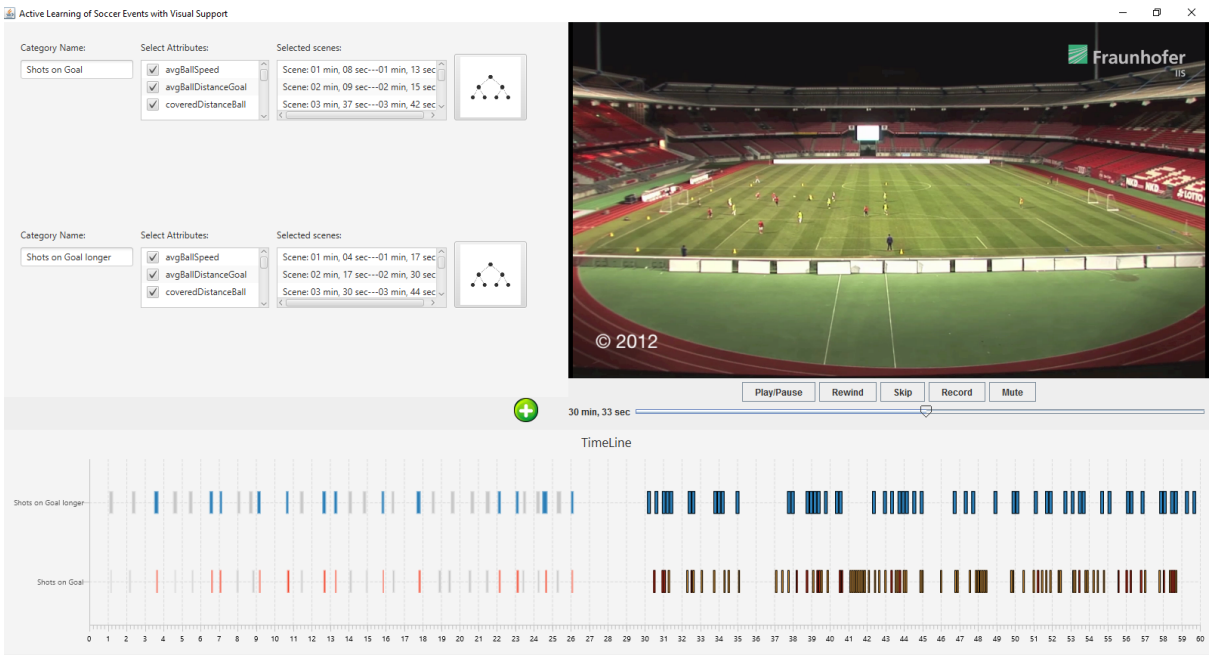
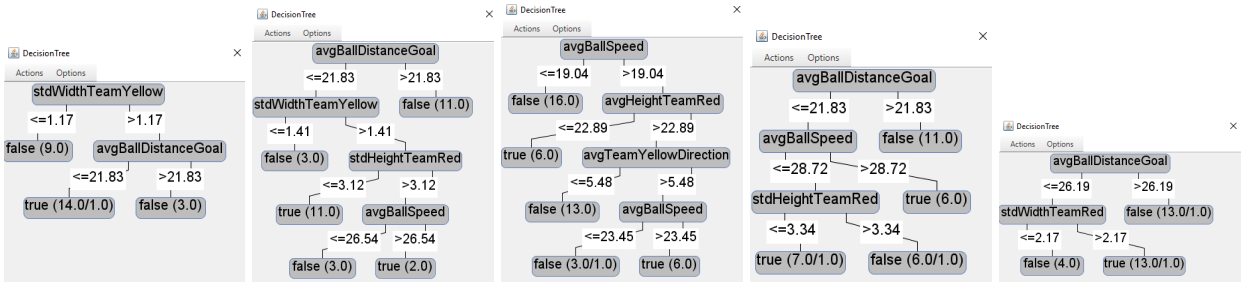


Abbildung 6.1: Ausgewählte und klassifizierte Szenen für die Kategorien ”Shot on Goal” (Methode 2) und ”Shot on Goal longer” (Methode 5)

Dieser Prozess wurde mit fünf verschiedenen Methoden durchgeführt:

1. Automatische Gegenbeispiele
2. Selbst ausgewählte Gegenbeispiele
3. Klassifizierte Gegenbeispiele
4. Angepasster Entscheidungsbaum
5. Längere Beispielszenen

Für alle Methoden wurden die gleichen 13 Beispielszenen, mit einer durchschnittlichen Länge von acht Sekunden, für Torschüsse aus der ersten Halbzeit ausgewählt. In der ersten Methode wurden Gegenbeispiele aber automatisch vom System berechnet. Das heißt, das System nimmt Szenen, die bisher nicht ausgewählt wurden und deklariert sie als Gegenbeispiel, bis es gleich viele positive wie negative Beispiele gibt. Deswegen wurden 13 hintereinander auftretende Szenen, die nach der ersten ausgewählten Szene kommen, als Gegenbeispiele definiert. Für die zweite Methode dagegen wurden auch die Gegenbeispiele manuell ausgesucht, um ein möglichst großes Spektrum an möglichen Szenen abzudecken, die man nicht finden möchte. Dabei wurde darauf geachtet, ungefähr gleich lange Szenen auszuwählen. Die dritte Methode benutzt zusätzlich die falsch klassifizierten Szenen der zweiten Methode. Die Szenen, die bei der Auswertung der klassifizierten Szenen der zweiten Methode keine Torschüsse repräsentierten, wurden der Kategorie als Gegenbeispiele hinzugefügt. Die vierte Methode benutzt die gleichen Szenen der zweiten Methode. Dafür wurde der entstehende Entscheidungsbaum



(a) Entscheidungsbaum nach Methode 1 **(b)** Entscheidungsbaum nach Methode 2 **(c)** Entscheidungsbaum nach Methode 3 **(d)** Entscheidungsbaum nach Methode 4 **(e)** Entscheidungsbaum nach Methode 5

Abbildung 6.2: Entscheidungsbäume, die für das Erkennen von Torschüssen erstellt wurden

			Richtig erkannt			Falsch erkannt			
Methode	**#gesucht**	**#gefunden**	**#**	**%gefunden**	**%gesucht**	**#**	**%gefunden**	**fast richtig erkannt**	**Torraumszenen**
1	24	56	17	0.30	0.71	39	0.70	4	12
2	24	42	17	0.40	0.71	25	0.60	2	10
3	24	38	13	0.34	0.54	25	0.66	3	6
4	24	58	18	0.31	0.75	40	0.69	2	13
5	24	52	19	0.37	0.79	33	0.63	0	11

Tabelle 6.1: Ergebnisse der ausgewerteten klassifizierten Szenen durch verschiedenen Methoden für die Erkennung von Torschüssen (# = Anzahl)

individuell angepasst. Dazu wurde das Attribut ”avgBallSpeed” ausgewählt, um zusammen mit ”avgDistanceGoal” eine Torszene zu beschreiben. Eine geringe Distanz zum Tor und eine hohe Geschwindigkeit des Balls sind aussagekräftig für einen Torschuss. In der fünften Methode wurden als Beispiele zwar grundsätzlich die gleichen Szenen wie für Methode zwei genommen, aber von diesen wurden jeweils längere Aufnahmen verwendet. Das heißt, anstatt durchschnittlich acht Sekunden, sind alle aufgenommenen Szenen durchschnittlich zwölf Sekunden lang. Die entstehenden ausgesuchten und klassifizierten Szenen sieht man beispielhaft für die zweite und fünfte Methode in Abbildung 6.1. Dabei sieht man direkt den Unterschied in der Länge der ausgewählten Szenen. Außerdem kann man erkennen, dass die klassifizierten Szenen zum Großteil die gleichen Bereiche abdecken.

Für alle Methoden wurde anschließend der Entscheidungsbaum erstellt. Diese verschiedenen Entscheidungsbäume sieht man in Abbildung 6.2. Es fällt auf, dass die Distanz zum Tor in allen Bäumen eine wichtige Rolle spielt, außer in Abbildung 6.2c. Für eine Torschussszene muss diese immer unter 26.19 Metern liegen. Auch die Geschwindigkeit des Balls ist meistens wichtig. Bei den Mannschaftsfeatures dagegen werden viele verschiedene genutzt, um die Szenen zu definieren.

Die entstandenen Entscheidungsbäume wurden dann genutzt, um alle Szenen der zweiten Halbzeit zu klassifizieren. Die klassifizierten Szenen wurden angeschaut, bewertet und mit den aufgezeichneten Torschussszenen der DEBS 2013 Grand Challenge verglichen. In der zweiten Halbzeit wurden 24 Torschüsse aufgezeichnet. Diese sollten vom System erkannt werden. Dazu wurde als erstes überprüft, ob die betrachtete Szene einer aufgezeichneten Szene entspricht. Je nachdem wurde sie den richtig oder falsch erkannten Szenen zugeordnet. Falsch erkannte Szenen wurden zusätzlich darauf überprüft, ob sie fast richtigen Szenen entsprechen, das heißt nur um wenige Sekunden zeitlich versetzt zu einer aufgezeichneten Szene sind, oder zumindest Torraumszenen zeigen. Zusätzlich wurde berechnet, wie viel Prozent der gesuchten Szenen richtig erkannt werden. Außerdem wurde das Verhältnis zwischen richtig und falsch erkannten Szenen zu den insgesamt gefundenen Szenen berechnet.

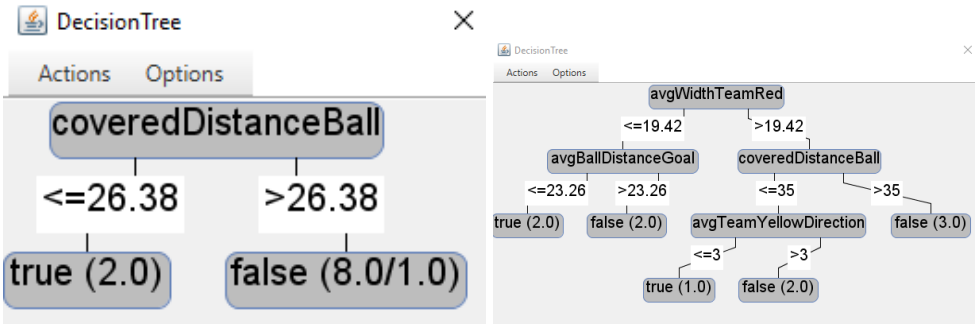
Die Ergebnisse dieser Auswertung sieht man in Tabelle 6.1. An dieser Tabelle kann man ablesen, dass alle Methoden, außer der dritten, über 70% der aufgezeichneten Szenen gefunden haben. Bei einigen Methoden würde dieser Wert auf über 80% steigen, wenn man die Szenen hinzunimmt, die sich nur um wenige Sekunden von gesuchten Szenen unterscheiden. Auf der anderen Seite kann man erkennen, dass zwischen 60% und 70% der insgesamt gefundenen Szenen keine Torschüsse sind. Auf der einen Seite ist das System damit erfolgreich dabei, das Gewünschte zu finden, auf der anderen Seite wird aber zusätzlich vieles Ungewünschte erkannt. Dabei ist es überraschend, dass mit Methode drei, wo die falschen klassifizierten Szenen zum Trainingsdatensatz hinzugefügt wurden, das Ergebnis verschlechtert wurde. Bei allen Methoden waren von den falsch identifizierten Szenen zumindest ca. ein Drittel Torraumszenen. Auch diese können für eine Zusammenfassung des Spiels oder die Analyse der Mannschaften genutzt werden.

6.2 Ecken

Standardsituationen sind immer spannende Momente. Sowohl für Zuschauer als auch für Trainer. Einige Teams trainieren intensiv das Verhalten bei Standardsituationen, um effektiver zu verteidigen und anzugreifen. Eine Art von Standardsituation sind Ecken. Diese sind am einfachsten zu identifizieren und zu analysieren, da sie immer von genau einem von vier Punkten auf dem Spielfeld ausgeführt werden.

Für diesen Anwendungsfall wurde das Fußballspiel selbständig annotiert. Anschließend wurden die Ecken aus der ersten Halbzeit als Beispielszenen verwendet, um alle Ecken aus der zweiten Halbzeit zu erkennen. In der ersten Halbzeit gab es drei Szenen mit Ecken, welche genutzt wurden, um fünf Ecken in der zweiten Halbzeit zu finden. Als Gegenbeispiele wurden verschiedenste Spielszenen, die keine Ecken sind, ausgewählt.

Den Entscheidungsbaum, der damit automatisch erstellt wird, sieht man in Abbildung 6.3a. Durch die wenigen zur Verfügung stehenden Beispiele fällt dieser sehr simpel aus. Deshalb wird in diesem Fall Expertenwissen eingebracht, um diesen Entscheidungsbaum anzupassen.



(a) Automatisch erstellter Entscheidungsbaum

(b) Selbst erstellter Entscheidungsbaum

Abbildung 6.3: Entscheidungsbäume, die für das Erkennen von Ecken erstellt wurden

Methode	#gesucht	#gefunden	richtig erkannt	falsch erkannt
automatisch	5	25	1	24
angepasst	5	31	4	27

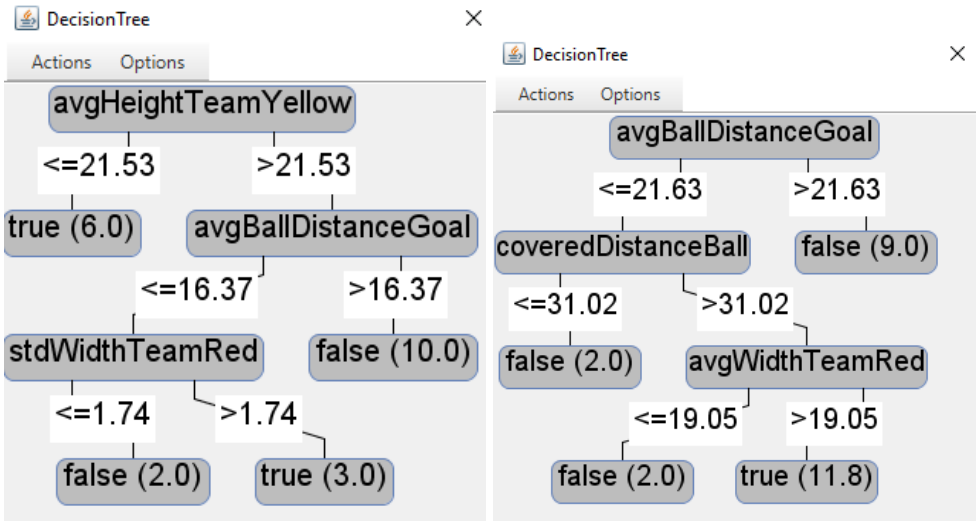
Tabelle 6.2: Ergebnisse der ausgewerteten klassifizierten Szenen für die Erkennung von Ecken

Bei Ecken halten sich beide Mannschaften in einem sehr kleinen Bereich auf. Deshalb ist die Breite der Mannschaften gering. Außerdem ist der Ball von Anfang bis Ende der Szene nah am Tor. Als letztes bewegen sich alle Spieler nicht sonderlich viel während Eckstößen. Darum wurden diese Features ausgewählt, um eine Ecke differenzierter, aber genauer zu beschreiben, was man in Abbildung 6.3b sieht.

Beide Entscheidungsbäume wurden anschließend verwendet, um die zweite Halbzeit des Spiels zu klassifizieren. Die Auswertung dieser Klassifikation kann in Tabelle 6.2 angeschaut werden. Es ist direkt zu erkennen, dass der angepasste Entscheidungsbaum deutlich besser für das Erkennen von Ecken ist. Außerdem gilt für beide Methoden, dass sie sehr viele zusätzliche Szenen identifizieren, die keine Ecken sind. Dies liegt hauptsächlich daran, dass die gemessene Distanz, die der Ball zurückgelegt hat, nur gemessen wird, solange der Ball im Spiel ist. Dadurch haben, neben Ecken, vor allem auch Einwürfe sehr geringe Werte für dieses Feature.

6.3 Flankenspiel

Für Trainer ist es sehr interessant, effektive Methoden zu finden, um viele Torschüsse und daraus entstehende Tore zu forcieren. Eine Taktik ist dabei, auf Flanken zu setzen. Typischerweise bedeutet das, dass ein Spieler mit dem Ball entlang der Grundlinie Richtung Eckfahne läuft und den Ball, meist hoch in den Torraum schlägt, worauf ein Mitspieler den Ball Richtung Tor



(a) Automatisch erstellter Entscheidungsbaum

(b) Durch zusätzliche Beispielszenen erstellter Entscheidungsbaum

Abbildung 6.4: Entscheidungsbäume, die für das Erkennen von Flanken erstellt wurden

befördert. Das heißt, man versucht bevorzugt über außen zu spielen, um anschließend gute Kopfballspieler vor dem Tor zu erreichen.

Für die Durchführung dieses Anwendungsfalls wurden als erstes alle Flanken des Fußballspiels identifiziert und aufgeschrieben. Daraufhin wurden alle Flanken der zweiten Halbzeit mit Hilfe des Systems aufgenommen und einer Kategorie zugeordnet. Außerdem wurden Gegenbeispiele ausgewählt, die Szenen symbolisieren, die keine Flanken sind. Der aus diesen Daten entstehende Entscheidungsbaum ist in Abbildung 6.4a zu sehen.

Wenn man Flanken nur im typischen Sinne versteht, ist man relativ limitiert. Einen Trainer könnten auch hohe Bälle aus dem Mittelfeld in den Strafraum und andere Spielzüge über außen interessieren. Diese Varianten würden immer noch die gewünschte Taktik repräsentieren. Deshalb wurde zum Vergleich eine zweite Version der Kategorie erstellt. Dafür wurden die gleichen Szenen wie zuvor genommen, aber ein paar neue hinzugefügt. Die neuen Szenen sind dabei Beispiele für Flanken aus dem Mittelfeld und andere Spielzüge über außen. Da der Fokus trotzdem auf Flanken liegen soll, wurden diese neuen Szenen mit einer geringeren Konfidenz bewertet. Damit wird ausgedrückt, dass diese Szenen in die Richtung des Gesuchten gehen, aber ihm nicht ganz entsprechen. Den resultierenden Entscheidungsbaum sieht man in Abbildung 6.4b.

Die Auswertung der klassifizierten Szenen durch die beiden Entscheidungsbäume ist in Tabelle 6.3 abgebildet. Es ist direkt zu sehen, dass der zweite Ansatz um einiges effektiver ist. Er hat sowohl mehr richtige Szenen erkannt als auch weniger falsche. Beide Methoden bringen aber zusätzliche ähnliche Szenen zum Vorschein. Damit sind Szenen gemeint, die nicht direkt Flanken sind, aber eine ähnliche Struktur haben. Dazu wurden gescheiterte Flankenversuche,

				Falsch erkannt	
Methode	**#gesucht**	**#gefunden**	**richtig erkannt**	**Anzahl**	**ähnliche Szenen**
nur Flanken	6	30	1	29	4
alle Varianten	6	24	3	21	4

Tabelle 6.3: Ergebnisse der ausgewerteten klassifizierten Szenen für die Erkennung von Flanken

Flanken aus dem Halbfeld und andere Angriffszüge über außen gezählt. Auch diese Szenen beinhalten hilfreiche Informationen, aus denen gelernt werden kann. Letzendlich ist offensichtlich, dass beide Methoden nicht optimal für das Finden von Flanken sind. Sowohl die Erfolgsrate, also auch die Anzahl der falsch gefundenen Szenen, haben Verbesserungspotenzial.

7 Diskussion

In diesem Kapitel werden die Ergebnisse aus der Evaluation diskutiert. Dabei wird auf Probleme und mögliche Verbesserungen eingegangen.

Durch die verschiedenen Anwendungsfälle wird deutlich, dass die Qualität der klassifizierten Szenen von den ausgewählten Beispielszenen, Gegenbeispielen und dem Entscheidungsbaum abhängt. Für ein gutes Ergebnis ist es nicht nur wichtig die Szenen auszuwählen, die man erkennen möchte, sondern auch geeignete Gegenbeispiele. Viele Szenen sind vom Aufbau her recht ähnlich. Zum Beispiel ist sowohl bei der Ecke als auch beim Einwurf der Ball zu Beginn der Szene außerhalb des Spielfeldes. Dadurch werden außerdem die anderen Ballfeatures, durchschnittliche Distanz zum Tor und Geschwindigkeit, beeinflusst, da sie nur berechnet werden, wenn der Ball im Spielfeld ist. Weiterhin wurde deutlich, dass es vorteilhaft sein kann, wenn man den Entscheidungsbaum manuell anpasst. Damit kann man wichtige Features eines Events einbringen, die bisher vom Algorithmus nicht beachtet wurden. Aus den Ergebnissen kann als weitere Erkenntnis geschlossen werden, dass die Auswahl von zusätzlichen Szenen, die nicht perfekt passen, sinnvoll sein kann. Auch ähnliche Szenen helfen, das gesuchte Event besser zu definieren und können weniger gewichtet werden. Als letztes hilft es oft, längere Szenen auszuwählen. Das System klassifiziert Szenen mit gleicher Länge wie die ausgewählten Szenen, wodurch weniger Szenen klassifiziert werden und damit auch weniger falsche.

7.1 Probleme

Das größte Problem sind die verhältnismäßig vielen falsch erkannten Szenen. Auch wenn die meisten Methoden über 70% der gesuchten Szenen richtig identifiziert, stehen dem viel mehr falsch erkannte Szenen gegenüber. Von den insgesamt gefundenen Szenen sind 60-87% falsch erkannt. Das heißt für einen Analysten, dass frühestens jede dritte Szene interessant für ihn ist. Dies liegt hauptsächlich daran, dass es schwer ist, Events genau zu definieren. Vor allem ist es schwierig, sie von anderen ähnlichen Szenen abzugrenzen.

Dazu tragen zum Großteil die bisherigen Features bei. Fast alle Features haben als Resultat Durchschnittswerte. Das ist problematisch, da verschiedene Szenen ähnliche Werte erhalten können. Je nach dem, wie die Szene ausgewählt wird, können wichtige Informationen wieder negiert werden. Wenn man zum Beispiel eine Torschussszene auswählt, kann man davon ausgehen, dass die Geschwindigkeit des Balls vergleichsweise hoch ist. Falls der Ball aber vom Torwart gehalten wurde und dieser ihn für einige Sekunden behält, ist die Geschwindigkeit im

Durchschnitt nicht hoch. Das führt dazu, dass Szenen von den Features her ähnlich erscheinen, aber in Wirklichkeit völlig unterschiedlich sind. Oftmals können Szenen gerade durch zwei Extreme definiert werden. Zum Beispiel werden Freistöße dadurch charakterisiert, dass der Ball zu Beginn an einem Punkt still liegt und anschließend in Richtung Tor geschossen wird. Diese Eigenschaften können mit Durchschnittswerten nicht ausgedrückt werden.

Ein weiteres Problem ist es, geeignete Features auszuwählen. Ohne die klassifizierten Szenen anzuschauen, ist es schwierig, den berechneten Entscheidungsbaum zu evaluieren. Es gibt keine Möglichkeit, im Voraus ein Gefühl dafür zu bekommen, wie gut der Entscheidungsbaum ist und welche Features möglicherweise wichtig sein könnten.

Eine weiterer Aspekt, der problematisch ist, sind die Szenenlängen. Bisher wird für die Wahl von automatischen Gegenbeispielen und zu klassifizierenden Szenen die Durchschnittslänge der ausgewählten Szenen genommen. Außerdem wird das Spiel vor vorne bis hinten durchgegangen und in Szenen dieser Länge aufgeteilt. Wie man in der Evaluation sehen kann, führt dies dazu, dass Szenen teilweise kurz vor einem gesuchten Event aufhören oder kurz danach anfangen.

7.2 Verbesserungen

Eine Verbesserung wären mehr Datensätze. Für das in dieser Arbeit implementierte System wurde nur ein Fußballspiel als Grundlage genommen. Das limitiert die zur Verfügung stehenden Szenen sehr. Zum Beispiel gibt es genau einen direkten Freistoß im gesamten Spiel, was es fast unmöglich macht, einen geeigneten Entscheidungsbaum für Freistöße zu erstellen.

7.2.1 Features

Eine Möglichkeit, Szenen genauer zu beschreiben, wären mehr und aussagekräftigere Features. Bisher werden Ball und Spieler nur separat betrachtet. Weitere Features, die eine Kombination von beidem berechnen, könnten dabei helfen, bessere Entscheidungsbäume zu erstellen. Dazu könnten Ballbesitz, Anzahl der Pässe, Anzahl der verschiedenen Spieler, die am Ball waren, oder auch Packing gehören.

Schwieriger ist es, zeitabhängige Features zu definieren. Eine Möglichkeit wäre es, Features durch Vektoren zu repräsentieren. Das heißt, eine Szene wird in mehrere Abschnitte aufgeteilt, für die alle Features berechnet werden, und die Szene wird durch einen Vektor dieser Abschnitte repräsentiert.

Außerdem könnte für die Auswahl der Features ein Ranking derselben helfen. Eine Möglichkeit wäre es, die Features nach ihrem Information Gain zu sortieren.

Weiterhin könnte es hilfreich sein, die Werte der verschiedenen Features für ausgewählte Szenen zu sehen. Dafür könnte man die Werte der Features über die Zeit in einem Graphen anzeigen. Damit wäre es möglich, diese Werte von verschiedenen Szenen zu vergleichen,

um Unterschiede und Gemeinsamkeiten zu identifizieren. Dieses Wissen könnte zum einen für die Wahl von Beispielen und Gegenbeispielen und zum anderen für die Anpassung des Entscheidungsbaums genutzt werden.

7.2.2 Entscheidungsbaum

Der Entscheidungsbaum wird nur anhand des Trainingsdatensatzes erstellt. Für ein besseres Ergebnis wäre es sinnvoll, diesen Trainingsdatensatz aufzuteilen, um einen Testdatensatz zu erhalten. Nach dem Erstellen wird der Entscheidungsbaum durch diesen Testdatensatz evaluiert. Anschließend können falsch klassifizierte Szenen dem Trainingsdatensatz hinzugefügt werden und der Prozess wird wiederholt. Dabei würden zusätzliche interessante Informationen, wie verschiedene Errorraten, Komplexitäten des Baums und Information Gains berechnet werden. Mit diesen Informationen könnten Veränderungen des Entscheidungsbaums direkt evaluiert werden. Für diese Methoden wären mehr Datensätze wichtig. Wenn der Trainingsdatensatz sehr klein ist, ist es schwierig, diese Methoden effizient einzusetzen.

Weiterhin könnte die Länge der klassifizierten Szenen variabler entschieden werden. Falls eine Szene als gesucht identifiziert wird, könnten automatisch verschiedene Längen dieser Szene ausprobiert werden, um herauszufinden, welche davon am besten passt.

Außerdem könnte es sinnvoll sein, einen Entscheidungsbaum ganz ohne Trainingsdaten zu erstellen. Ein Nutzer könnte rein durch sein Domainenwissen einen Entscheidungsbaum aus den gegebenen Features erstellen. Damit wäre es nicht nötig, das Fußballspiel anzuschauen, um Beispielszenen zu finden.

Ein weiterer Ansatz sind Entscheidungswälder. Anstatt mit nur einem Entscheidungsbaum zu arbeiten, benutzt man mehrere, die in einem Mehrheitsentscheid klassifizieren. Dafür wird der Trainingsdatensatz aufgeteilt und für jeden Teil wird ein eigener Entscheidungsbaum erzeugt. Neben besseren Klassifikationsergebnissen bietet dieser Ansatz weitere Möglichkeiten für Interaktion. Dem Nutzer wäre es möglich, mit mehreren Entscheidungsbäumen für ein Event zu arbeiten, diese zu vergleichen und zu evaluieren. Dabei könnte er einige Entscheidungsbäume anpassen und hätte den direkten Vergleich mit den automatisch erstellten.

7.2.3 Visualisierung

Die Benutzeroberfläche bietet Potenzial, um den Nutzer in seinen Aufgaben zu unterstützen. Es könnte einfacher sein, das Video direkt über die Timeline zu steuern. Dadurch hätte der Nutzer die Möglichkeit, seine Szenen beim Betrachten des Videos zu verändern.

Eine weitere Verbesserung könnte es sein, die Kategorien besser vergleichen zu können. Es könnte interessant sein zu sehen, welche Features in welchen Kategorien wichtig sind. Weiterhin könnte es helfen, wenn für jede Kategorie eine kleine Version des Entscheidungsbaums direkt angezeigt wird, die mit jeder hinzugefügten Szene angepasst wird.

Als letztes kann die visuelle Repräsentation des Entscheidungsbaums verständlicher gestaltet werden. Ein interessanter Ansatz wäre es, die Kanten anhand den dort passenden Szenen anzupassen, damit direkt identifiziert werden kann, welche Äste des Baums besonders wichtig sind. Außerdem wäre es verständlicher, wenn man gleiche Features in gleicher Weise markiert.

8 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Konzept zur interaktiven Erstellung eines Klassifikators für die Erkennung von interessanten Events in Fußballspielen vorgestellt. Dazu wurden zuerst einige wichtige Grundlagen erklärt. Anschließend wurde das entwickelte Konzept in Zusammenhang mit existierende Ansätze gestellt.

Daraufhin wurde das Konzept, bestehend aus vier Komponenten, präsentiert. Mit Hilfe des Videos des Fußballspiels können Beispielszenen und Gegenbeispiele aufgenommen werden und einer Kategorie zugeordnet werden. Eine Kategorie steht für das zu erkennende Event. Alle Szenen werden durch vordefinierte Features repräsentiert. Mit diesen Features wird anschließend ein Klassifikator, in Form eines Entscheidungsbaums, trainiert. Dabei hat der Nutzer die Möglichkeit, Szenen unterschiedlich zu gewichten und Features im Voraus auszuschließen. Der entstehende Entscheidungsbaum kann außerdem beliebig verändert werden. Für Knoten können andere Features ausgewählt werden, für Kanten andere Splitpoints gesetzt werden und weitere Knoten können hinzugefügt werden. Der Entscheidungsbaum kann daraufhin genutzt werden, um weitere Szenen des Fußballspiels zu klassifizieren. Das Ergebnis dieser Klassifikation wird in einer Timeline angezeigt und die gefundenen Szenen können über das Video validiert werden. Dieses Konzept wurde basierend auf einem Datensatz mit GPS-getrackten Sensordaten von Ball und Spielern und dem dazugehörigen Video umgesetzt.

Abschließend wurde das implementierte System durch verschiedene Anwendungsfälle evaluiert. Dabei wurde versucht, Torschüsse, Ecken und Flanken des Fußballspiels zu finden, in dem eine Halbzeit als Testdatensatz für den Entscheidungsbaum diente, um die andere Halbzeit zu klassifizieren. Auch wenn viele Ansätze in der Lage waren, die meisten der gesuchten Szenen zu identifizieren, standen diesem Ergebnis eine viel größere Anzahl an falsch erkannten Szenen gegenüber. Diese Ergebnisse wurden diskutiert und verschiedene Verbesserungen vorgestellt.

Ausblick

Eine Ausweitung des vorgelegten Konzepts könnte durch die Anwendung auf mehrere Datensätze erfolgen. Es wäre interessant und hilfreich, Szenen aus verschiedenen Fußballspielen zu klassifizieren.

Darüber hinaus gibt es viele Möglichkeiten, weitere Features zu implementieren, die das Spezifizieren von Ereignissen verbessern.

Außerdem kann die Erstellung des Entscheidungsbaums verbessert werden. Zum einen wäre ein automatisches Testen des Entscheidungsbaums sinnvoll. Zum anderen könnte es interessant sein, vom Nutzer komplett manuell erstellte Entscheidungsbäume zu verwenden und mit automatisch erstellten zu vergleichen. Darüber hinaus könnte das Konzept auf Entscheidungswälder ausgeweitet werden. Dies würde weitere Möglichkeiten der Interaktion und Klassifikation bieten.

Neben den vorgestellten Verbesserungen kann das entwickelte Konzept auch für andere Anwendungen verwendet werden. Sowohl andere Sportarten als auch generell Anwendungen, die interaktive Klassifikationsalgorithmen nutzen, können von diesem Konzept profitieren. Entscheidungsbäume sind eine einfach verständliche Art, Klassifikatoren zu repräsentieren. Die in dieser Arbeit vorgestellten Konzepte können dabei helfen, diese sonst automatisch erstellten Klassifikatoren durch die Integration des Nutzers in den Prozess zu verbessern.

Literaturverzeichnis

[ABD+02] J. Assfalg, M. Bertini, A. Del Bimbo, W. Nunziati, P. Pala. „Soccer highlights detection and recognition using HMMs." In: *ICME (1)*. 2002, S. 825–828 (zitiert auf S. 22).

[AS13] C. Anderson, D. Sally. *The numbers game: why everything you know about football is wrong*. Penguin UK, 2013 (zitiert auf S. 11).

[BFSO84] L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen. *Classification and regression trees*. CRC press, 1984 (zitiert auf S. 16).

[BLC+14] A. Bialkowski, P. Lucey, P. Carr, Y. Yue, S. Sridharan, I. Matthews. „Large-scale analysis of soccer matches using spatiotemporal tracking data". In: *2014 IEEE International Conference on Data Mining*. IEEE. 2014, S. 725–730 (zitiert auf S. 23).

[CMM83] J. G. Carbonell, R. S. Michalski, T. M. Mitchell. „An overview of machine learning". In: *Machine learning*. Springer, 1983, S. 3–23 (zitiert auf S. 15).

[CSCZ04] S.-C. Chen, M.-L. Shyu, M. Chen, C. Zhang. „A decision tree-based multimodal data mining framework for soccer goal detection". In: *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*. Bd. 1. IEEE. 2004, S. 265–268 (zitiert auf S. 22).

[FB13] W. Fan, A. Bifet. „Mining big data: current status, and forecast to the future". In: *ACM sIGKDD Explorations Newsletter* 14.2 (2013), S. 1–5 (zitiert auf S. 13).

[GFW+11] T. von der Grün, N. Franke, D. Wolf, N. Witt, A. Eidloth. „A real-time tracking system for football match and training analysis". In: *Microelectronic systems*. Springer, 2011, S. 199–212 (zitiert auf S. 26).

[GJHV11] E. Gamma, R. Johnson, R. Helm, J. Vlissides. *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Pearson Deutschland GmbH, 2011 (zitiert auf S. 37).

[GR12] J. Gantz, D. Reinsel. „The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east". In: *IDC iView: IDC Analyze the future* 2007.2012 (2012), S. 1–16 (zitiert auf S. 13).

[GW10] J. Gudmundsson, T. Wolle. „Towards automated football analysis: Algorithms and data structures". In: *Proc. 10th Australasian Conf. on mathematics and computers in sport*. Citeseer. 2010 (zitiert auf S. 22).

[KBT+13] R. Krüger, H. Bosch, D. Thom, E. Püttmann, Q. Han, S. Koch, F. Heimerl, T. Ertl. „Prolix-visual prediction analysis for box office success“. In: *IEEE Conference on Visual Analytics Science and Technology*. 2013 (zitiert auf S. 21).

[KKEM10] D. Keim, J. Kohlhammer, G. Ellis, F. Mansmann. *Mastering the information age solving problems with visual analytics*. Eurographics Association, 2010 (zitiert auf S. 14).

[KMS+08] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, H. Ziegler. „Visual analytics: Scope and challenges“. In: *Visual data mining*. Springer, 2008, S. 76–90 (zitiert auf S. 13, 14).

[LCP+12] P. A. Legg, D. H. Chung, M. L. Parry, M. W. Jones, R. Long, I. W. Griffiths, M. Chen. „MatchPad: Interactive Glyph-Based Visualization for Real-Time Sports Performance Analysis“. In: *Computer Graphics Forum*. Bd. 31. 3pt4. Wiley Online Library. 2012, S. 1255–1264 (zitiert auf S. 22).

[LCSM16] H. Liao, L. Chen, Y. Song, H. Ming. „Visualization-Based Active Learning for Video Annotation“. In: *IEEE Transactions on Multimedia* 18.11 (2016), S. 2196–2205 (zitiert auf S. 21).

[LKT+14] Y. Lu, R. Krüger, D. Thom, F. Wang, S. Koch, T. Ertl, R. Maciejewski. „Integrating predictive analytics and social media“. In: *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on*. IEEE. 2014, S. 193–202 (zitiert auf S. 21).

[LWLZ17] S. Liu, X. Wang, M. Liu, J. Zhu. „Towards Better Analysis of Machine Learning Models: A Visual Analytics Perspective“. In: *arXiv preprint arXiv:1702.01226* (2017) (zitiert auf S. 21).

[Mit06] T. M. Mitchell. *The discipline of machine learning*. Bd. 9. Carnegie Mellon University, School of Computer Science, Machine Learning Department, 2006 (zitiert auf S. 15).

[MP13] T. Mühlbacher, H. Piringer. „A partition-based framework for building and validating regression models“. In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), S. 1962–1971 (zitiert auf S. 21).

[MZJ13] C. Mutschler, H. Ziekow, Z. Jerzak. „The DEBS 2013 grand challenge“. In: *Proceedings of the 7th ACM international conference on Distributed event-based systems*. ACM. 2013, S. 289–294 (zitiert auf S. 26).

[PGM13] J. Perl, A. Grunz, D. Memmert. „Tactics analysis in soccer–An advanced approach“. In: *International Journal of Computer Science in Sport* 12.1 (2013), S. 33–44 (zitiert auf S. 23).

[Qui14] J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014 (zitiert auf S. 17).

[Qui86] J. R. Quinlan. „Induction of decision trees“. In: *Machine learning* 1.1 (1986), S. 81–106 (zitiert auf S. 16, 17).

[RN14] M. G. I. Rathod, M. D. A. Nikam. „Review on event retrieval in soccer video“. In: *Int. J. Comput. Sci. Inf. Technol* 5 (2014), S. 5601–5605 (zitiert auf S. 22).

[SAM11] S. F. de Sousa, A. d. A. Araújo, D. Menotti. „An overview of automatic event detection in soccer matches“. In: *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*. IEEE. 2011, S. 31–38 (zitiert auf S. 22).

[SHJ+15] M. Stein, J. Häußler, D. Jäckle, H. Janetzko, T. Schreck, D. A. Keim. „Visual Soccer Analytics: Understanding the Characteristics of Collective Team Movement Based on Feature-Driven Analysis and Abstraction“. In: *ISPRS International Journal of Geo-Information* 4.4 (2015), S. 2159–2184 (zitiert auf S. 23, 24).

[SSN+16] L. Shao, D. Sacha, B. Neldner, M. Stein, T. Schreck. „Visual-Interactive Search for Soccer Trajectories to Identify Interesting Game Situations“. In: *Electronic Imaging* 2016.1 (2016), S. 1–10 (zitiert auf S. 22).

[SSS+14] D. Sacha, M. Stein, T. Schreck, D. A. Keim, O. Deussen et al. „Feature-driven visual analytics of soccer data“. In: *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on*. IEEE. 2014, S. 13–22 (zitiert auf S. 23).

[ZEHK12] H. M. Zawbaa, N. El-Bendary, A. E. Hassanien, T. Kim. „Event detection based approach for soccer video summarization using machine learning“. In: *Int J Multimed Ubiquitous Eng* 7.2 (2012), S. 63–80 (zitiert auf S. 22).

Alle URLs wurden zuletzt am 15. 03. 2017 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift