DHBW
Duale Hochschule
Baden-Württemberg

Duale Hochschule Baden - Württemberg Mannheim

Seminararbeit

Entscheidungsbäume

Am Beispiel des ID3 Algorithmus

Studiengang Angewandte Informatik

Studienrichtung Informatik

Autor:	Martin Pretz
Matrikelnummern:	7060026
Kurs:	TINF18AI1
Bearbeitungszeitraum:	19.05.2021 - 10.06.2021

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Abstract

This paper covers a special type of machine learning used for solving classification problems, which is the decision tree. Therefore, at the beginning of this paper we will discuss in general what desicion trees are, how they work and how they can be created. During the main part of this paper, additionally the process of creating a desicion tree will be discussed in detail using the example of the ID3 algorithm. In the first place, this includes the explanation of basic concepts and the operating principle of the ID3 algorithm. In the second place this is followed by an exemplary creation of a decision tree under the use of a dedicated example data set. Afterwards this paper includes a personal implementation of the ID3 algorithm. While this paper also covers the potential challanges that my arise from using a desicion tree for a classification problem, it also covers the concept of pruning for improving desicion trees.

2 Einführung

Die Frage nach der korrekten Entscheidung bzw. die Herausforderung eine korrekte Vorhersage für ein Ereignis zu treffen beschäftigt sowohl Ökonomen als auch Wissenschaftler. Zum Beispiel stellt sich die Frage ob der Kurs eines bestimmten Wertpapiers in nächster Zeit steigen oder fallen wird oder ob eine bestimmte Behandlung für einen Patienten empfehlenswert ist. [1] Zu diesem Zweck kommt unter anderem künstliche Intelligenz zum Einsatz, konkret kommt hierbei besonders maschinelles Lernen im Form der Klassifikation zum Einsatz. [2] Dabei wird versucht eine Vorhersage anhand von vergangenen Erfahrungen bzw. Daten vorzunehmen, es wird also eine Größe unter Berücksichtigung von verschiedenen Variablen vorhergesagt. Dafür können verschiedene Methoden zum Einsatz kommen, wobei Entscheidungsbäume eine der bekanntesten Methode ist. Daher werden in dieser Arbeit Entscheidungsbäume näher beleuchtet.

3 Was sind Entscheidungsbäume?

3.1 Aufbau und Funtionsweise

Ein Entscheidungsbaum ist ein geordneter und gerichteter [3] Baum welcher aus Knoten und Kanten besteht. Dabei unterteilt man Konten in zwei verschiedene Typen. Zum einen gibt es die Entscheidungsknoten [4]. Dabei handelt es sich um eine Überprüfung einer Eigenschaft eines Datenobjektes [3] während eine Kante gerade das Ergebnis jener Überprüfung ist. Dabei gibt es den Wurzelknoten als besondere Ausprägung eins Entscheidungsknoten. Er unterscheidet sich von einem normalen Entscheidungsknoten nur in soweit, als dass der Wurzelknoten der Ursprung des gesammten Entscheidungsbaums ist. Zum anderen gibt es die Endknoten [4] welche üblicherweise als Blätter bezeichnet werden. Bei einem Blatt handel es sich um einen Enzustand welcher das Ergebnis bzw. die Klassifikation angibt. [3]

Möchte man nun ein Datenobjekt mit Hilfe eines zuvor erstellten Entscheidungsbaums klassifizieren, so folgt man beginnend mit dem Wurzelknoten dem Entscheidungsbaum abwärts. Dabei wird bei jedem Entscheidungsknoten eine Eigenschaft bzw. ein Attribut des Datenobjektes überprüft. Basierend auf dem Ergebnis dieser Überprüfung wird dann entlang einer der Kanten der nachfolgende Entscheidungsknoten ausgewählt. [3] Dies "[...] wird so lange fortgesetzt bis man ein Blatt erreicht"[5] und somit das Datenobjekt klassifiziert ist.

3.2 Erzeugung

Die Erzeugung eines Entscheidungsbaums erfolgt in der Regel nach dem Top-Down Prinzip, indem in jedem Prozessschrit das beste Attribut ausgewählt wird um daraus einen Knoten zu erzeugen und den Datensatz aufzuteilen. [6] Dabei nutzen die verschiedenen Algorithmen unterschiedliche Methoden um das beste Attribute zu ermitteln, z.B. den Informationsgewinn, den Gini-Index oder den Misclassification Error. [3] Nachdem der Datensatz aufgeteilt wurde, wird überprüft ob alle Datenobjekte einer Teilmenge klassifiziert sind. Falls dies der Falls ist wird ein Blatt mir der entsprechenden Klassifikation erstellt und der Prozess wird für die verbleibende Teilmenge wiederholt. [7]

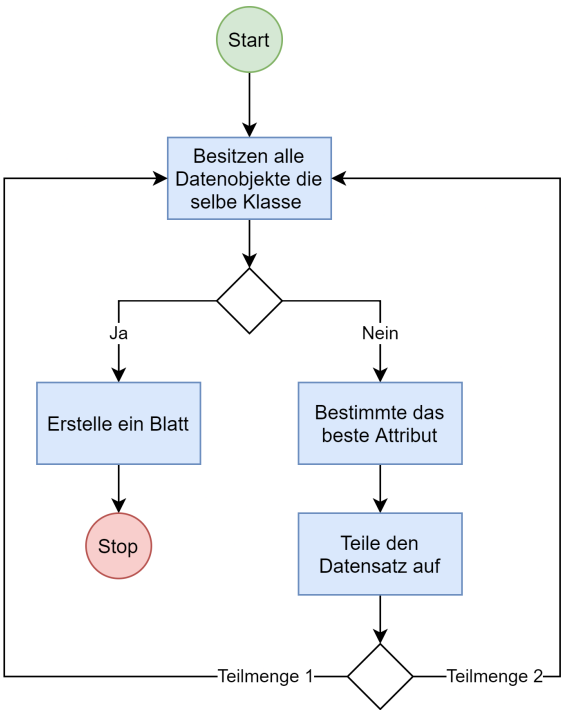

Abbildung 3.1: Vereinfachte konzeptionelle Darstellung der Erzeugung eines Entscheidungsbaums [7]

Es ist zu beachten dass die Erstellung eines garantiert optimalen Entscheidungsbaums ein NP-vollständiges Problem ist. [8] Daher nutzen viele Entscheidungsbaum-Algorithmen einen gierigen Ansatz, was bedeutet dass nur nach der lokal optimalen Entscheidung für jeden Knoten gesucht wird. [3]

Zur Erzeugung eines Entscheidungsbaums wird ein Trainingsdatensatz mit Objekten verwendet, für die bereits eine Klassifikation vorliegt. [5] Dabei ist zu beachten dass es abhängig vom gewählten Trainingsdatensatz zu sogenanntem *Overfitting* kommen kann. Dies ist zum Beispiel der Fall, wenn ein Datensatz viele Attribute aber nur wenige Datenobjekte besitzt. [9] Entscheidungsbäume die unter *Overfitting* leiden, zeichnen sich dadurch aus, dass sie überangepasst (also gut an den Trainingsdatensatz angepasst) sind und nur schlecht generallisieren. Das bedeutet dass unbekannte reale Daten, die vom Trainingsdatensatz abweichen falsch klassifiziert werden. [3]

4 Der ID3 Algorithmus

Bei ID3 (Iterative Dichotomiser 3) handelt es sich um einen Algorithmus zur Erstellung eines Entscheidungsbaumes welcher von Ross Quinlan entwickelt wurde. [2]

4.1 Grundlagen

Der ID3-Algorithmus macht sich zwei Konzepte der Informationstheorie zu nutze. Es handelt sich zum einen um die Entropie und zum anderen um den Informationsgewinn. Beide Konzepte werden im nachfolgenden erläutert. Im Anschhluss daran wird die eigentliche Funktionsweise des ID3 Algorithmus erläutert.

4.1.1 Entropie

In der Informationstheorie wird mit der Entropie H die Sicherheit bzw. Unsicherheit einer Variablen X angegeben. Dementsprechend ist x_i eine mögliche Ausprägung der Variablen X und $P(x_i)$ die Wahrscheinlichkeit mit der die Variable X die Ausprägung x_i hat. [10]

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_b P(x_i)$$

Abbildung 4.1: Definition der Entropie nach Shennon[10]

Beispiel: Sei D ein Datensatz in dem das Attribut X mit den möglichen Ausprägungen x_1, x_2 und x_3 vorkommt. Weiterhin gelte, dass x_1 neun mal, x_2 drei mal und x_3 fünf mal in X vorhanden ist. Zur Bestimmung der Entropie von X egribt sich die nachfolgende Berechnung. Für den ID3-Algorithmus wird üblicherweise der Logarithmus zur Basis b=*2* verwendet. [11]

$$
\begin{aligned}
H(X) &= -\sum_{i=1}^{3} P(x_i) \log_2 P(x_i) \\
&= -(P(x_1) \log_2 P(x_1) + P(x_2) \log_2 P(x_2) + P(x_3) \log_2 P(x_3)) \\
&= -\frac{9}{17} \cdot \log_2 \left(\frac{9}{17}\right) - \frac{3}{17} \cdot \log_2 \left(\frac{3}{17}\right) - \frac{5}{17} \cdot \log_2 \left(\frac{5}{17}\right) \\
&\approx 0,485755 + 0,441618 + 0,519275 \\
&\approx \underline{\underline{1,446648}}
\end{aligned}
$$

Abbildung 4.2: Exmeplarische Berechung der Entropie

4.1.2 Informationsgewinn

In der Informationstheorie beschreibt der Informationsgewinn IG das Maß an Informationen das über eine Zufallsvariablen X durch Beobachtung einer anderen Zufallsvariablen Y gewonnen werden kann. [3] Konkret ergibt sich der Informationsgewinn aus der Differenz der Entropie $H(X)$ und der bedingten Entropie $H(X|Y)$. [3, 12, 2]

$$IG(X,Y) = H(X) - H(X|Y) = H(X) - \sum_{y \in Y} P(y) H(X|Y = y)$$

Abbildung 4.3: Allgemeine Definition des Informationsgewinns [3, 13]

Beispiel: Sei S ein Datensatz mit den in Tabelle 4.1 dargestellten Werten. Außerdem seien A, B, C und T Attribute von S mit den möglichen Ausprägungen *True* und *False*. Sei weiterhin das Attribut T das Zielattribut gegen das der Informationsgewinn der übrigen Attribute ermittelt werden soll.

ID	Attribut A	Attribut B	Attribut C	Attribut T
1	True	True	True	False
2	True	False	True	True
3	False	False	True	True
4	False	True	True	False
5	False	True	False	True

Tabelle 4.1: Beispiel Datensatz S zur Berechnung des Informationsgewinns [12]

Im nachfolgenden wird der Informationsgewinn für das Attribut A berechnet. Dabei wird zu erst die Entropie des Datensatz S bestimmt, welche durch die Entropie des Zielattributes T charakterisiert wird. Es gilt also $H(S) = H(T)$.

$$\begin{aligned} H(S) = H(T) &= -\sum_{i=1}^{2} P(x_i) \log_2 P(x_i) \\ &= -(P(\text{True}) \log_2 P(\text{True}) + P(\text{False}) \log_2 P(\text{False})) \\ &= -\frac{3}{5} \cdot \log_2 \left(\frac{3}{5}\right) - \frac{2}{5} \cdot \log_2 \left(\frac{2}{5}\right) \\ &\approx 0,970951 \end{aligned}$$

Im nächsten Schritt wird die bedingte Entropie für das Attribut A berechnet.

$$\begin{aligned} H(T|A) &= \sum_{a \in A} P(a) \cdot H(T|A = a) \\ &= P(True) \cdot H(T|A = True) + P(False) \cdot H(T|A = False) \\ &= \frac{2}{5} \cdot \left(-\frac{1}{2} \cdot \log_2 \left(\frac{1}{2}\right) - \frac{1}{2} \cdot \log_2 \left(\frac{1}{2}\right)\right) + \frac{3}{5} \cdot \left(-\frac{1}{3} \cdot \log_2 \left(\frac{1}{3}\right) - \frac{2}{3} \cdot \log_2 \left(\frac{2}{3}\right)\right) \\ &= \frac{2}{5} \cdot 1 + \frac{3}{5} \cdot 0,918296 \\ &\approx 0.950978 \end{aligned}$$

Nachdem nun sowohl die Entropie als auch die bedingte Entropie berechnet sind kann final der Informationsgewinn bestimmt werden.

$$\begin{aligned} IG(T, A) &= H(T) - H(T|A) \\ &= 0,970951 - 0.950978 \\ &= \underline{\underline{0,019973}} \end{aligned}$$

Analog können die Informationsgewinne für die Attribute B und C berechnet werden. Diese liegen bei $IG(T, B) = \textit{0,419973}$ und bei $IG(T, C) = \textit{0,170951}$.

4.1.3 Funktionsweise

Der ID3 Algorithmus startet mit einem Datensatz S mit Objekten $O_1, ..., O_n$ welche die Attribute $A_1, ..., A_k$ und die Klassifizierung C besitzen. [11, 2] Die Werteausprägungen der Attribute sind dabei normalerweise endlich und diskret. [14]
Zu Beginn muss ein (Wurzel-) Knoten bestimmt werden. Dazu wird der Informationsgewinn $IG(S)$ für jedes Attribut $A_1, ..., A_k$ berechnet. Das Attribut mit dem höchsten Informationsgewinnungswert A_H wird dann als Knoten gewählt. Dabei gilt dass A_H die möglichen Ausprägungen $a_1, ..., a_v$ haben kann. Basierend auf A_H wird S in $S_1, ..., S_v$ Teilmengen zerlegt wobei die Teilmenge S_i die Objekte aus S beinhaltet deren Wert in A_H die Ausprägungen a_i hat. Anders ausgedrückt gilt $S_i = \{S|A_H = a_i\}$. Für jede Teilmenge S_i wird dann rekursiv der vorher beschriebene Prozess für die verleibenden Attribute $\{A_1, ..., A_k\} \setminus A_H$ durchlaufen wodurch entsprechende Teilbäume erzeugt werden. [2] Dabei wird die Rekursion beendet sofern,

1. alle Objekte aus S_i die gleiche Ausprägung c_i der Klassifizierung C besitzen. In diesem Fall wird der zu diesem Zeitpunkt ausgewählte Knoten zu einem Blatt mit der Ausprägung c_i. [2, 9, 15]

2. bereits alle Attribute $A_1, ..., A_k$ für den Entscheidungsbaum verwendet worden sind, es also keine Attribute mehr gibt mittels derer ein Objekt aus S_i klassifiziert werden könnte, aber nicht alle Objekte der selben Klassifizierung c_i zugeordnet werden können. In diesem Fall wird der aktuelle Knoten zu einem Blatt, welchem der Modalwert von C zugeordnet wird.[2, 9, 15]

3. die Teilmenge S_i keine Objekte mehr beinhaltet, sie also leer ist. Dies tritt ein wenn sich in der übergeordneten Menge keine Objekte befinden welche die Ausprägung a_i des aktuell ausgewählten Attributes A_H besitzen. In diesem Fall wird ein Blatt erzeugt welchem der Modalwert von C aus der übergeordneten Menge zugeordnet wird. [9, 15]

4.2 Anwendungsbeispiel

In diesem Kapitel wird die Erzeugung eines Entscheidungsbaumes mit Hilfe des ID3 Algorithmus anhand eines Beispiels eräütert. Zu diesem Zweck wurde ein Datensatz S ausgewählt welcher in Tabelle 4.2 dargestellt ist. [11]. Dieser Datensatz beinhaltet die Attribute *WEATHER*, *TEMP*, *HUMIDITY* und *WIND* sowie ein Klassifizierung *PLAY*. Hierbei ist die Aufgabe den ID3 Allgorithmus zu nutzen um einen Entscheidungsbaum zu erstellen, welcher vorhersagt ob das Wetter ermöglicht dass Fußball gespielt werden kann.

DAY	WEATHER	TEMP	HUMIDITY	WIND	PLAY
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Cloudy	Hot	High	Weak	Yes
4	Rainy	Medium	High	Weak	Yes
5	Rainy	Cold	Normal	Weak	Yes
6	Rainy	Cold	Normal	Strong	No
7	Cloudy	Cold	Normal	Strong	Yes
8	Sunny	Medium	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
10	Rainy	Medium	Normal	Weak	Yes
11	Sunny	Medium	Normal	Strong	Yes
12	Cloudy	Medium	High	Strong	Yes
13	Cloudy	Hot	Normal	Weak	Yes
14	Rainy	Medium	High	Strong	No

Tabelle 4.2: Beispiel Datensatz S

Als erstes werden die Informationsgewinne für die Attribute aus dem Datensatz S berechnet. Dabei ergibt sich folgende Übersicht.

$$\begin{aligned} IG(S,\, WEATHER) &= 0.246749 \\ IG(S,\, TEMP) &= 0.029222 \\ IG(S,\, HUMIDITY) &= 0.151835 \\ IG(S,\, WIND) &= 0.048127 \end{aligned}$$

Da das Attribut *WEATHER* den höchsten Informationsgewinn hat wird es als Knoten bzw. Wurzelknoten gewählt. *WEATHER* besitzt die drei möglichen Ausprägungen *Sunny*, *Cloudy* und *Rainy* für die nun im jeweiligen Iterationschritt eine Teilmenge von S erzeugt wird. Daraus ergibt sich zunächst für die Ausprägung *Sunny* folgende Teilmenge $S_1 = S_{WEATHER=Sunny}$.

DAY	WEATHER	TEMP	HUMIDITY	WIND	PLAY
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Medium	High	Weak	No
9	Sunny	Cold	Normal	Weak	Yes
11	Sunny	Medium	Normal	Strong	Yes

Tabelle 4.3: Teilmenge S_1

Für S_1 wird nun rekursiv ein Teilbaum erzeugt wobei die Teilmenge keine der Abbruchbedingungen erfüllt, sodass man nun mit der Berechnung der Informationsgewinne für die Attribute aus S_1 fortfahren kann. Dabei ergeben sich folgende Werte.

$$\begin{aligned} IG(S,\, TEMP) &= 0.570951 \\ IG(S,\, HUMIDITY) &= 0.970951 \\ IG(S,\, WIND) &= 0.019973 \end{aligned}$$

Nun wird das Attribut *HUMIDITY* als Knoten gewählt da es nun den höchsten Informationsgewinn besitzt. Dabei hat *HUMIDITY* die zwei mögliche Ausprägungen *High* und *Normal*. Somit ergeben sich die Teilmengen $S_{1_{HUMIDITY=High}}$ und $S_{1_{HUMIDITY=Normal}}$ aus S_1.

DAY	WEATHER	TEMP	HUMIDITY	WIND	PLAY
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Medium	High	Weak	No

Tabelle 4.4: Teilmenge $S_{1_{HUMIDITY=High}}$

DAY	WEATHER	TEMP	HUMIDITY	WIND	PLAY
9	Sunny	Cold	Normal	Weak	Yes
11	Sunny	Medium	Normal	Strong	Yes

Tabelle 4.5: Teilmenge $S_{1_{HUMIDITY=Normal}}$

Betrachtet man die oben dargestellten Teilmengen fällt sofort auf dass alle Objekte dieser Teilmengen jeweils die selbe Klassifizierung besitzen. Für $S_{1_{HUMIDITY=High}}$ nämlich die Klassifizierung *No* und für $S_{1_{HUMIDITY=Normal}}$ die Klassifizierung *Yes*. Daher wird für beide Ausprägungen des Knotens *HUMIDITY* ein Blatt erstellt welches jeweils

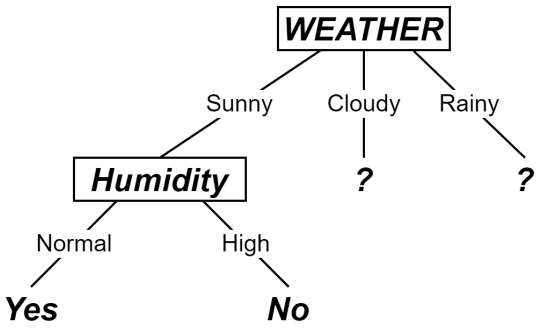

Abbildung 4.4: Vorläufiger Teilbaum nachdem *HUMIDITY* klassifiziert wurde

die entsprechende Klassifizierung zugewiesen bekommt. Daraus ergibt sich folgender vorläufiger Teilbaum.

Da nun die Ausprägungen des Attributes *HUMIDITY* klassifiziert worden sind fahren wir mit der zweiten Ausprägung von *WEATHER* fort, nämlich *Rainy*. Für diese wird die Teilmenge $S_2 = S_{WEATHER=Rainy}$ erstellt.

DAY	WEATHER	TEMP	HUMIDITY	WIND	PLAY
4	Rainy	Medium	High	Weak	Yes
5	Rainy	Cold	Normal	Weak	Yes
6	Rainy	Cold	Normal	Strong	No
10	Rainy	Medium	Normal	Weak	Yes
14	Rainy	Medium	High	Strong	No

Tabelle 4.6: Teilmenge S_2

Für S_2 wird nun erneut der Informationsgewinn der verbleibenden Attribute berechnet. Zu diesem Zeitpunkt bleiben nur noch die Attribute *TEMP* und *WIND* übrig, da die anderen Attribute bereits verwendet worden sind. Es ergeben sich folgende Werte für den Informationsgewinn.

$$\begin{aligned} IG(S,\ TEMP) &= 0.019973 \\ IG(S,\ WIND) &= 0.970951 \end{aligned}$$

Basierend auf den Informationsgewinnen wird *WIND* als Knoten gewählt. Dieses Attribut besitzt die zwei Ausprägungen *Weak* und *Stong*. Somit ergeben sich die Teilmengen $S_{2_{WIND=Weak}}$ und $S_{2_{WIND=Strong}}$ aus S_2.

DAY	WEATHER	TEMP	HUMIDITY	WIND	PLAY
4	Rainy	Medium	High	Weak	Yes
5	Rainy	Cold	Normal	Weak	Yes
10	Rainy	Medium	Normal	Weak	Yes

Tabelle 4.7: Teilmenge $S_{2_{WIND=Weak}}$

DAY	WEATHER	TEMP	HUMIDITY	WIND	PLAY
6	Rainy	Cold	Normal	Strong	No
14	Rainy	Medium	High	Strong	No

Tabelle 4.8: Teilmenge $S_{2_{WIND=Strong}}$

Bei beiden Teilmengen fällt auf dass alle Objekte aus diesen Teilmengen die selbe Klassifizierung aufweisen. An dieser Stelle wird analog zum Vorgehen beim Attribut *HUMIDITY* ein Blatt für die Ausprägungen von *WIND* erstellt. Darus ergibt sich nun folgender Teilbaum.

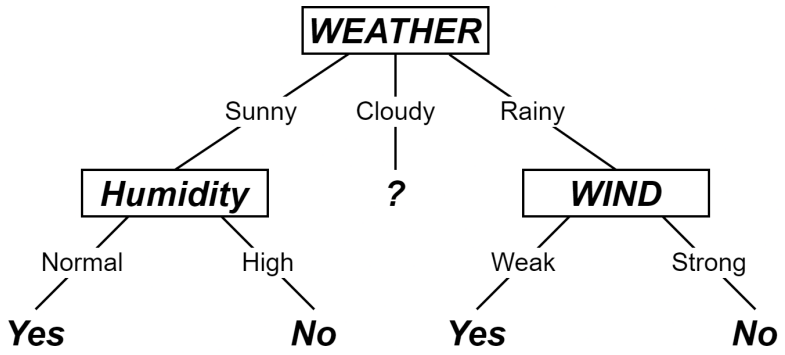

Abbildung 4.5: Vorläufiger Teilbaum nachdem *WIND* klassifiziert wurde

Nachdem nun die Klassifizierung für das Attribut *WIND* und somit auch für die Ausprägung *Rainy* abgeschlossen ist, bleibt nur noch die Ausprägung *Cloudy* zur Klassifizierung übrig. Basierend darauf ergibt sich folgende Teilmenge $S_3 = S_{WEATHER=CLoudy}$.

Bei der Betrachtung von S_3 fällt auf dass alle beinhalteten Objekte die selbe Klassifizierung *Yes* besitzen. Aus diesem Grund wird analog zu der Vorgehensweise bei *WIND* und *HUMIDITY* ein Blatt mit der entsprechenden Klassifizierung erstellt. Somit wird der ID3 Algorithmus beendet und es entsteht der folgende finale Entscheidungsbaum.

DAY	WEATHER	TEMP	HUMIDITY	WIND	PLAY
3	Cloudy	Hot	High	Weak	Yes
7	Cloudy	Cold	Normal	Strong	Yes
12	Cloudy	Medium	High	Strong	Yes
13	Cloudy	Hot	Normal	Weak	Yes

Tabelle 4.9: Teilmenge S_3

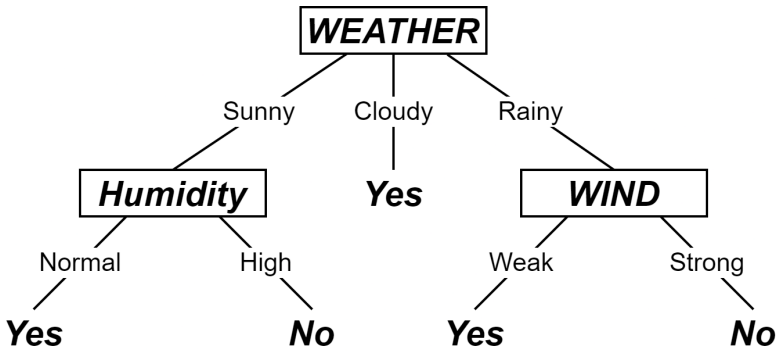

Abbildung 4.6: Finaler Entscheidungsbaum

Aus dem oben dargestellten Entscheidungsbaum ergeben sich folgende Regeln für das Spielen.

1. Wenn das Wetter sonnig ist und die Humidität normal, dann kann Fußball gespielt werden. Wenn die Humidität allerdings hoch ist, dann kann kein Fußball gespielt werden. [11]
2. Wenn das Wetter bewölkt ist, dann kann Fußball gespielt werden. [11]
3. Wenn das Wetter regnerisch ist und der Wind schwach, dann kann Fußball gespielt werden. Falls aber der Wind stark ist, dann kann kein Fußball gespielt werden. [11]

4.3 Implementation

Diese persönliche Implementation besteht im wesentlichen aus vier Funktionen. Zum einen aus Nebenfunktionen wie der Berechnung des Informationsgewinns, der Entropie und des Modalwerts, zum anderen aus der Hauptfunktion bzw. dem eigentlichen ID3 Algorithmus.

4.3.1 Berechnung der Entropie

Die hier vorliegende Implementation zur Berechnung der Entropie erwartet als Eingabeparameter ein Attribut eines Datensatzes. Im Anschluss daran wird über die Zuweisung in Zeile 3 die Anzahl aller Ausprägungne des Attributes ermittelt. Dabei kommt die Bibliotheksfunktion `np.unique` zum Einsatz. Diese Funktion gibt zwei Listen zurück. Die erste Liste beinhaltet dabei alle möglichen Ausprägungen des Attributes während die zweite Liste die Anzahl eben jener Ausprägungen beinhaltet. Im Anschluss daran wird über die Länge der Liste `count` iteriert. Dabei wird in jedem Durchlauf die Wahrscheinlichkeit einer Ausprägung des Attributes berechnet. Außerdem wird die Entropie partiell für die vorliegende Ausprägung berechnet und mit dem vorherigen partiellen Entropiewert addiert.

```
1         def entropy(attribute):
2             entropy = 0.0
3             values, count = np.unique(attribute, return_counts=True)
4             for index in range(len(values)):
5                 probablility = count[index] / sum(count)
6                 entropy += (-probablility * np.log2(probablility))
7             return entropy
```

Abbildung 4.7: Funktion zu Berechnung der Entropie eines Attributes[9, 11]

4.3.2 Berechnung des Informationsgewinns

Die vorliegende Implementation des Informationsgewinns erwartet drei Eingabeparameter. Diese sind der Datensatz, ein Attribut dieses Datensatzes und das Zielattribute gegen welches der Informationsgewinns bestimmt werden soll. Zu Beginn werden unter Zuhilfenahme der Funktion `np.unique` alle möglichen Ausprägungen des Attributes sowie deren Anzahl bestimmt.
Im Hauptteil dieser Funktion wird über die möglichen Ausprägungen `values` des Attribute iteriert. Dabei wird in Zeile 5 zunächst eine Teilmenge `subdata` des ursprünglichen Datensatzes gebildet (vgl. Kapitel 4.1.2). Im Anschluss daran wird die

Wahrscheinlichkeit der Ausprägung `value`, sowie die Entropie der Teilmenge `subdata` berechnet. Dabei werden die Entropieen der verschiedenen Ausprägungen addiert. Im letzen Schritt wird der Informationsgewinn aus der Differenz der Entropie des Zielattributes und der bedingten Entropie berechnet (vgl. Kapitel 4.1.2).

```
1         def information_gain(data, attribute, target_attribute):
2             values, count = np.unique(data[attribute], return_counts=True)
3             entropy_val = 0.0
4             for index, value in enumerate(values):
5                 subdata = data[data[attribute] == value][target_attribute]
6                 probablility = count[index] / sum(count)
7                 entropy_val += ( probablility * entropy(subdata) )
8             target_entropy = entropy(data[target_attribute])
9             information_gain = target_entropy - entropy_val
10            return information_gain
```

Abbildung 4.8: Funktion zur Berechnung des Informationsgewinns [9, 11]

4.3.3 Berechnung des Modalwertes

Die Funktion `modal` erwartet eine Liste von Elementen als Eingabeparameter. Zu Beginn der Funktion wird zunächst mit Hilfe der Funktion `np.unique` die Menge aller vorkommenden Werte sowie deren Anzahl in der Anfangsliste bestimmt. Im Anschluss daran wird aus beiden Informationen ein Dictionary erstellt. Im letzten Schritt wird mittels einer Lambda Funktion der Schlüssel des Dictionary ermittelt dessen Wert am höchsten ist.

```
1         def modal(attribute):
2             values, count = np.unique(attribute, return_counts=True)
3             total = dict(zip(values, count))
4             return max(total, key=lambda k: total[k])
```

Abbildung 4.9: Berechnung des Modalwertes [16]

4.3.4 ID3 - Hauptfunktion

Die Hauptfunktion erwartet als Eingabeparameter den Ursprungsdatensatz, das Zielattribut und eine Liste aller möglichen Attribute des Datensatzes. Bei dem Zielattribut handelt es sich um die Klassifizierung.
Zu Beginn der Funktion werden zwei Abbruchbedingungen geprüft (vgl. Kapitel 4.1).

Als erstes wird untersucht, ob die sich in dem Datensatz `data_set` auschließlich Objekte mit der gleichen Klassifizierung befinden. Wenn dem so ist, wird genau diese Klassifizierung zurück gegeben. Als zweites wird untersucht, ob bereits alle Attribute verwendet worden sind. Sofern dies zutrifft wird der Modalwert des Zielattributes zurück gegeben.
Im Anschluss daran wird mit der Funktion `calculate_all_IG` der Informationsgewinn für jedes Attribut berechnet, wobei das Attribut mit dem höchsten Wert (`best_attribute`) als (Wurzel-) Knoten gewählt wird. Im Anschluss daran wird in Zeile 10 `best_attribute` aus der Menge aller Attribute entfernt.
Nun beginnt der iterative Teil der Funktion. Dabei wird über alle Ausprägungen des Attributes `best_attribute` eine Teilmenge `subset` des ursprünglichen Datensatzes erstellt. Danach wird die dritte Abbruchbedingung für die Rekursion geprüft, nämlich ab die Teilmenge `subset` leer ist. In diesem Fall wird ein Blatt erstellt welchem der Modalwert `modal_value` des Zielattributes zugeordnet wird. Wenn allerdings die Teilmenge `subset` nicht leer ist, so wird in Zeile 18 rekursiv ein Teilbaum erstellt. Dieser wird in Zeile 19 der Ausprägung `value` des Attributes `best_attribute` zugewiesen.

```
1         def ID3(data_set, target_attribute, attributes):
2             if len(np.unique(data_set[target_attribute])) <= 1:
3                 return np.unique(data_set[target_attribute])[0]
4             if len(attributes) <= 1:
5                 return modal(data_set[target_attribute])
6
7             IG = calculate_all_IG(data_set, target_attribute, attributes)
8             best_attribute = max(IG, key=lambda k: IG[k])
9             tree = {best_attribute: {}}
10            attributes = [x for x in attributes if x != best_attribute]
11
12            for value in np.unique(data_set[best_attribute]):
13                subset = data_set[data_set[best_attribute] == value]
14                if subset.empty:
15                    modal_value = modal(data_set[target_attribute])
16                    tree[best_attribute][value] = modal_value
17                else:
18                    subtree = ID3(subset, target_attribute, attributes)
19                    tree[best_attribute][value] = subtree
20            return tree
```

Abbildung 4.10: Hauptfunktion des ID3 Algorithmus [16, 15, 9]

5 Pruning

Beim Pruning handelt es sich um eine Methode zur Bearbeitung eines Entscheidungsbaums. Das Ziel ist dabei die Größe eines Baums zu reduzieren um so dem die Auswirkungen des *Overfittings* zu minimieren. Die Prädiktive Qualität des Baums soll allerdings nicht unter Pruning leiden. [17, 3] Man unterscheidet dabei zwsichen dem *Pre-Pruning* und dem *Post-Pruning*. [3]

5.1 Pre-Pruning

Beim *Pre-Pruning* erfolgt die Reduktion des Entscheidungsbaums noch bevor dieser fertiggestellt ist, bzw. während dieser erzeugt wird. Dabei soll verhindert werden dass unnötige Entscheidungsknoten und Teilbäume erzeugt werden, weshalb *Pre-Pruning* einsetzt, wenn der Datensatz geteilt wird. Um Festzustellen ob der Datensatz geteilt werden sollte können verschiedene Kriterien zum Einsatz kommen. Zum Beispiel könnte das Kriterium des minimalen Informationsgewinns genutzt werden, bei welchem der Datensatz nur dann geteilt wird wenn der Wert des Informationsgewinn über einem bestimmten Mindestwert liegt. [3] Allerdings unterliegen alle *Pre-Pruning* Methoden dem *Horizont Effekt*. [17]
Dabei handelt es sich um ein Problem welches dazu führt dass eventuell ein Teilbaum erzeugt wird der nicht wünschenswert ist. Dies liegt daran, dass beim *Pre-Pruning* nur der aktuelle Entscheidungsknoten betrachtet wird und wir nicht über diesen hinaus schauen können. [18, 17] Der *Horizont Effekt* lässt sich anhand des folgenden Beispiels veranschaulichen.
Nehmen wir an dass wir uns gerade innerhalb eines Entscheidungsbaums an einem Entscheidungsknoten befinden und bestimmen wollen wie der Trainingsdatensatz aufgeteilt werden soll. Dabei nutzen wir das Kriterium des minimalen Informationsgewinns. Also berechnen wir die Informationsgewinne der verschiedenen Attribute A, B und C. Wir stellen dabei fest, dass der Informationsgewinn der Attribute A und C unterhalb des von uns festgelegten Mindestwertes liegt. Aus diesem Grund wählen wir das Attribut B für den Entscheidungsknoten aus. Was zu diesem Zeitpunkt allerdings nicht klar ist, ist dass nach dem Entscheidungsknoten mit dem Attribut A, zwei weitere Entscheidungsknoten mit den Attributen A_1 und A_2 gefolgt wären, welche den Datensatz besser klassifiziert hätten als das Attribut B.

5.2 Post-Pruning

Die Reduktion der Baumgröße erfolgt beim *Post-Pruning* nachdem der Entscheidungsbaum fertiggestellt wurde. Dabei ist die grundlegende Idee, dass Entscheidungsknoten bzw. Teilbäume zu Blättern umgewandelt werden. Erreicht werden kann dies durch verschiedene Methoden, welche sich in die zwei Kategorien *Bottom-Up Post-Pruning* und *Top-Down Post-Pruning* einteilen lassen können. [17]
Wie der Name es bereits vermuten lässt, handelt es sich bei der *Bottom-Up* um eine Pruning Vorgehensweise welche bei dem tiefsten Blatt des Entscheidungsbaums startet. Sofern sich zwei Blätter auf der gleichen Ebene befinden, wird das Blatt gewählt welches sich am weitesten Links befindet. [7] Von diesem Blatt dem Baum aufwärts folgend, wird die Relevanz jedes Entscheidungsknoten für die vorliegende Klassifikation bestimmt. Falls ein Entscheidungsknoten als irrelevant bewertet wird, so wird dieser durch ein Blatt ersetzt. Einer der bekanntesten und einfachsten *Bottom-Up Post-Pruning* Algorithmen ist der *Reduced-Error-Pruning* Algorithmus, welcher einem Blatt den Modalwert der Klassifikation zuweist. [7]
Dem gegenüber steht die Vorgehensweise des *Top-Down Post-Prunings* Prunings. Diese Form des *Post-Prunings* startet an dem Wurzelknoten und arbeitet sich abwärts zu den Blättern vor. Auch hier wird analog zum *Bottom-Up* überprüft ob ein Entscheidungsknoten relevant für die Klassifikation ist. Problematisch ist dabei, dass durch das entfernen einens Entscheidungsknoten unter Umständen ein kompletter Teilbaum entfernt wird, obwohl unklar ist obwohl unklar ist ob dieser von Relevanz für die Klassifikation ist. [17, 7]

6 Zusammenfassung

6.1 Vorteile

Entscheidungsbäume bieten diverse Vorteile gegenüber anderen Methoden. Einer der offensichtlichsten Vorteile ist die einfache Verständlichkeit und Interpretierbarkeit von Entscheidungsbäumen. [3] Sie können visualisiert werden und ermöglichen es somit auch Laien mit ihnen zu Arbeiten. [9] Außerdem können Entscheidungsbäume aus Datensätzen erstellt werden die nicht normalisiert sind und sowohl numerische als auch mit kategoriale Attribute enthalten. Dabei können Entscheidungsbäume auch mit sehr großen Datensätzen arbeiten. [9]

6.2 Nachteile

Aber natürlich besitzen Entscheidungsbäume auch Einschränkungen. So sind sie sind relativ inkonsistent, da bereits kleine Änderungen innerhalb des Trainingsdatensatzes zu weitreichenden Veränderungen des Entscheidungsbaums führen können. [9]
Besonders bei Entscheidungsbäumen kann es zu Overfitting kommen wodurch reale Daten falsch klassifiziert werden. Allerdings kann diesem Problem mittels Pruning entgegengewirkt werden. [3]
Wenn Datensätze verwendet werden, die kategorialen Attribute mit einer Vielzahl von möglichen Ausprägungen besitzen, ist der Informationsgewinn in Entscheidungsbäumen zugunsten dieser Attribute ausgelegt.[9]

Literatur

[1] J. Quinlan, *Decision Trees and Decisionmaking*, 1990.

[2] ——, *Induction of Decision Trees*, 1986.

[3] M. Schinck, *Data Mining Vorlesung 2: Classification 1, Einführung, Validierung und Decision Trees*, 2021.

[4] P. S. B. Kamiński M. Jakubczyk, *A framework for sensitivity analysis of decision trees*, 2017.

[5] *Entscheidungsbaum – Wikipedia*, `https://de.wikipedia.org/wiki/Entscheidungsbaum`, (Accessed on 05/20/2021).

[6] O. M. L. Rokach, *Top-down induction of decision trees classifiers - a survey*, 2005.

[7] „Decision Tree Algorithm explained p.4 – Decision Tree Pruning - SEBASTIAN MANTEY.“ (), Adresse: `https://www.sebastian-mantey.com/theory-blog/decision-tree-algorithm-explained-p4-decision-tree-pruning`.

[8] R. L. R. Laurent Hyafil, *Constructing Optimal Binary Decision Trees is NP-complete*, 1976.

[9] *Machine Learning with Python: Decision Trees in Python*, `https://www.python-course.eu/Decision_Trees.php`, (Accessed on 05/21/2021).

[10] C. Shannon, *A Mathematical Theory of Communication*, 1948.

[11] S. V. Rupali Bhardwaj, *Implementation of ID3 Algorithm*, CSE, Bahra Univerity India, 2013.

[12] *Information gain in decision trees - Wikipedia*, (Accessed on 05/27/2021). Adresse: `https://en.wikipedia.org/wiki/Information_gain_in_decision_trees`.

[13] J. A. T. Thomas M. Cover, *Elements of Information Theory*, 1991.

[14] K. Jearanaitanakij, *Classifying Continuous Data Set by ID3 Algorithm*, 2005.

[15] „ID3 algorithm - Wikipedia.“ (), Adresse: `https://en.wikipedia.org/wiki/ID3_algorithm`.

[16] *Python max()*, `https://www.programiz.com/python-programming/methods/built-in/max`, (Accessed on 05/21/2021).

[17] „Decision tree pruning - Wikipedia.“ (), Adresse: `https://en.wikipedia.org/wiki/Decision_tree_pruning`.

[18] H. J. Berliner. „Some Necessary Conditions for a Master Chess Program.“ (1973).