Mathematische Grundlagen III

Maschinelles Lernen II: Klassifikation mit Entscheidungsbäumen

Prof Dr. Matthew Crocker

Universität des Saarlandes

9. Juli 2015

Einleitung

Letzte Vorlesung haben wir den Naive Bayes Klassifikator kennengelernt, heute behandeln wir Entscheidungsbäume.

Maschinelle Lernverfahren in diesem Kurs:

- **Entscheidungsbäume**
- Naive Bayes Klassifikation
- Clustering

Inhaltsverzeichnis

Inhaltsverzeichnis

Maschinelles Lernen: Wiederholung

- Künstliche Generierung von Wissen aus Erfahrung
- Erkennung komplexer Muster und Regelmäßigkeiten in vorhandenen Daten
- Ziel: Verallgemeinerung (Generalisierung)
  - Über das Nachschlagen bereits gesehener Beispiele hinausgehen
  - Beurteilung unbekannter Daten

Beispiel

Erinnern Sie sich an unseren Golf-Club Manager, der entscheiden möchte, wann er seinen Angestellten frei geben kann und wann er zusätzlich Studenten als Aushilfe einstellen kann.

Er macht zwei Wochen Notizen zu Wetter und Spielverhalten.

Beispieldatensatz

Wir wollen lernen, bei welchen Wetterkonditionen gespielt wird.

outlook	temp.	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Wiederholung: Naive Bayes

Ein Naive Bayes Klassifizierer würde

- alle diese Wetterbeobachtungen als voneinander unabhängig betrachten
- für jedes Feature die Wahrscheinlichkeit für "Leute spielen" vs. "Leute spielen nicht" aus den Daten abschätzen, sowie die allgemeine Wahrscheinlichkeit dafür dass die Leute spielen (a-Priori-Wahrscheinlichkeit).
- Dann würden wir die Wahrscheinlichkeiten für alle Attribute gegeben "Spielen" multiplizieren, und ebenso für alle Attribute gegeben "nicht Spielen".
- Letztendlich würden wir uns für "Spielen" oder "Nicht spielen" entscheiden, je nachdem welche Kategorie die höhere a-posteriori Wahrscheinlichkeit hat.

Regeln lernen

Regeln wie

sonnig	&	$\neg$ schwühl	$\rightarrow$	spielen
$\neg$ sonnig	&	schwühl	$\rightarrow$	spielen
sonnig	&	schwühl	$\rightarrow$	$\neg$ spielen

kann man damit aber *nicht* lernen (wegen der Unabhängigkeitsannahme).

Mit Entscheidungsbäumen können wir genau solche Konjunktionen von Attributen und Werten lernen, um eine Instanz einer Kategorie zuzuordnen.

Inhaltsverzeichnis

Wann sind Entscheidungsbäume gut geeignet?

- Instanzen sind als Attribut-Wert Paare repräsentiert (d.h. jede Instanz hat Attribute (Bewölkung, Luftfeuchtigkeit...) mit diskreten Werten (sonnig, bewölkt...)
- Wir wollen Kategorien lernen (keine kontinuierliche Funktion)
- Konjunktion und/oder Disjunktion benötigt um Daten zu beschreiben
- Trainingsdaten können Fehler enthalten oder unvollständig bzgl. der Attribute einer Instanz sein. (Entscheidungsbäume sind robust in Bezug auf kleine Unregelmäßigkeiten.)

Entscheidungsbäume

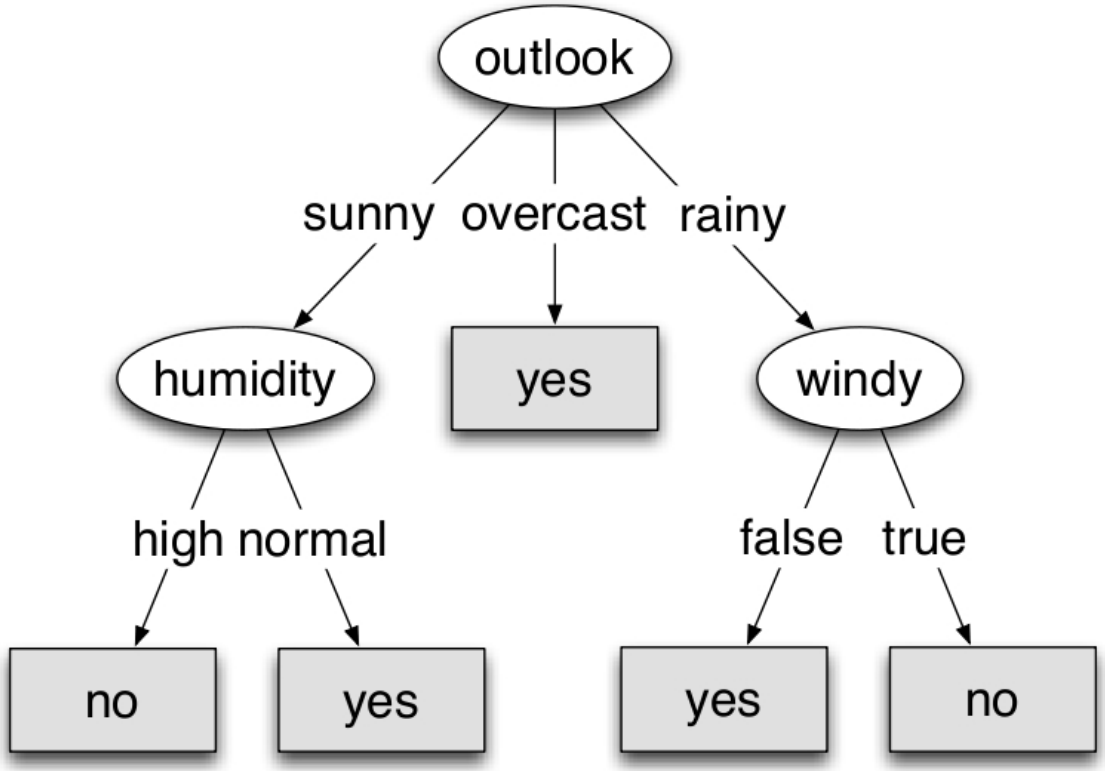
Allgemein

Ein von einem Experten erstellter Baum, um aufeinanderfolgende, hierarchische Entscheidungen zu veranschaulichen

Data Mining

Ein auf Basis verfügbarer Daten erstelltes Modell, das benutzt werden kann, um Voraussagen über neue Daten zu machen

Baum für den Wetter-Datensatz



- Nicht-terminale Knoten testen ein Attribut
- Baumzweige entsprechen den Werten, die ein Attribut annehmen kann
- Blätter weisen Instanzen einer Kategorie zu

ID3 Algorithmus: Intuition

- Bestimmen, welches Attribut die Daten am besten klassifiziert
- Dieses Attribut als Wurzel des Baums verwenden
- Einen Zweig für jeden Wert erstellen, den das Attribut annehmen kann
- Dieses Prozess für jeden Baumzweig wiederholen, jeweils mit der Untermenge der zu klassifizierenden Daten
- Rekursiv weitermachen, bis die Klassifikation unter allen Blättern eindeutig ist

Auswahl des klassifizierenden Attribut

Frage: Wie wird das beste Attribut bestimmt?

Antwort: Es wird das Attribut gewählt, das

. . . die meiste Information beiträgt

. . . die Entropie im Datensatz am meisten reduziert

Auswahl des klassifizierenden Attribut

Frage: Wie wird das beste Attribut bestimmt?

Antwort: Es wird das Attribut gewählt, das

- ...die meiste Information beiträgt
- ...die Entropie im Datensatz am meisten reduziert

Informationsgewinn

Für jedes Attribut A:

$$Gain(S, A) = H(S) - \sum_{v \in A} \frac{|S_v|}{|S|} H(S_v)$$

S	:	Datensatz
$H(S)$	:	Entropie im Datensatz oder Untermenge davon
S_v	:	Untermenge von S für die A den Wert v hat
$\|S\|$	:	Mächtigkeit von S
$\|S_v\|$	:	Mächtigkeit von S_v

Der Informationsgewinn $Gain(S, A)$ ist die Reduzierung der Entropie, die man dadurch erwartet, dass man den Wert von Attribut A kennt

Entropie im Datensatz

$$H(S) = -\sum_{c=1}^{|C|} p_c \log_2 p_c$$

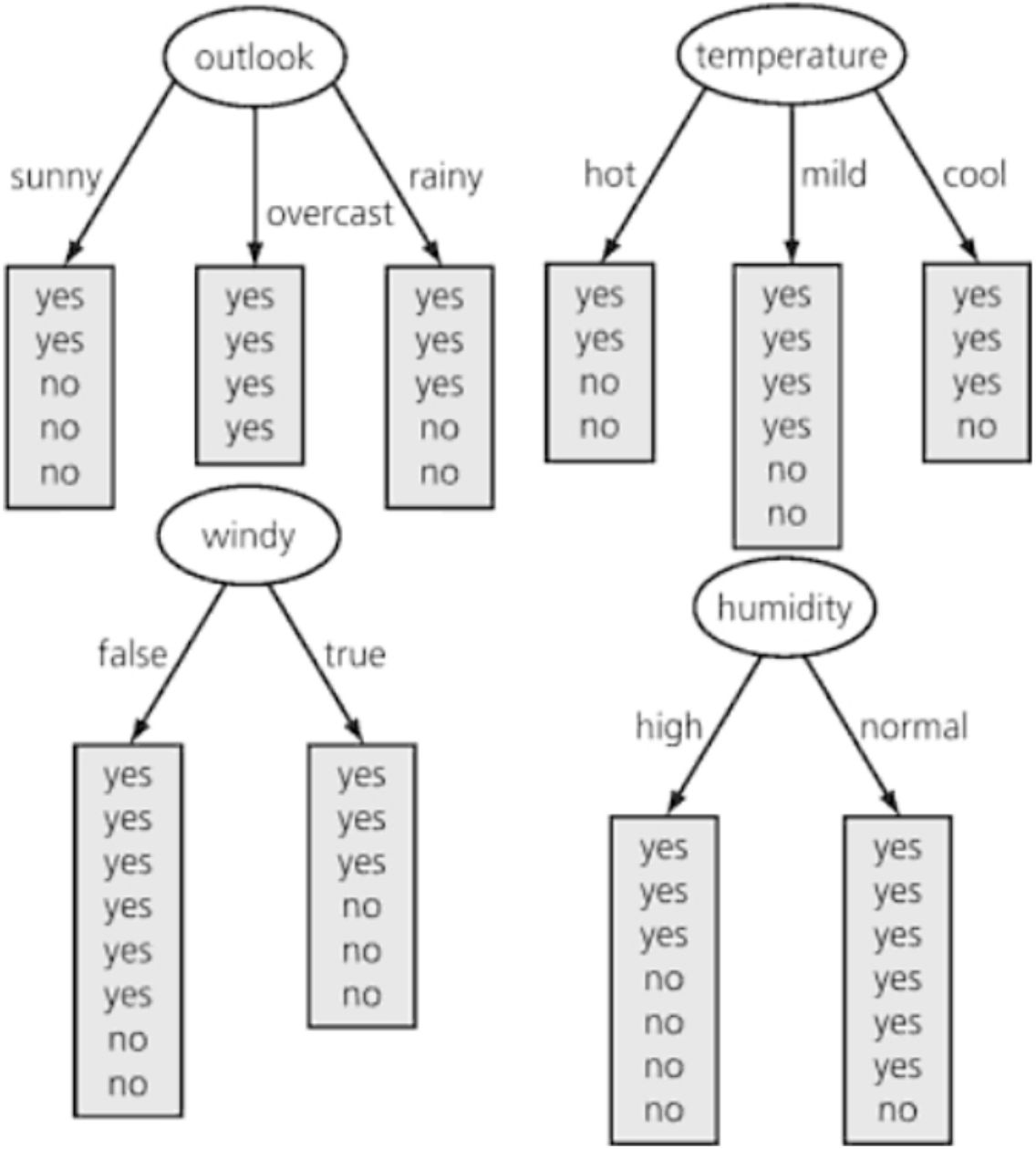
S: Datensatz

$|C|$: Anzahl Kategorien

p_c: Anteil der Instanzen in S, die Kategorie k angehören

Informationstheoretische Interpretation

- $H(S)$: Anzahl Bits, die benötigt werden, um ein beliebiges Element aus S zu kodieren
- $H(S)$ ist gleich null, wenn alle Instanzen von S der selben Kategorie angehören
- Bei einer binären Klassifikation ist $H(S)$ gleich eins, wenn S gleich viele Instanzen aus jeder Klasse enthält


outlook
sunny
overcast
rainy
yes yes no no no
yes yes yes yes
yes yes yes no no
windy
false
true
yes yes yes yes yes no no
yes yes yes no no no
temperature
hot
mild
cool
yes yes no no
yes yes yes yes no no
yes yes yes no
humidity
high
normal
yes yes yes no no no no
yes yes yes yes yes no

Beispiel für den Wetterdatensatz

$$Gain(S, A) = H(S) - \sum_{v \in A} \frac{|S_v|}{|S|} H(S_v)$$

- Erst die Gesamtentropie berechnen
  14 Instanzen, 5 × *"play=no"*, 9 × *"play=yes"*:
  $$H(S) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940286$$
- Dann für jedes Attribut den Datensatz so unterteilen, dass alle Instanzen für das Attribut A den gleichen Wert haben, und für jeden solchen Teildatensatz die Entropie berechnen
  $A : windy = false$: 8 Instanzen, 6 × *"play=yes"* + 2× *"play=no"*
  $$H(S_{false}) = -\frac{6}{8} \log_2 \frac{6}{8} - \frac{2}{8} \log_2 \frac{2}{8} = 0.8112781$$
  $A : windy = true$: 6 Instanzen, 3 × *"play=yes"*, 3 × *"play=no"*:
  $$H(S_{true}) = -\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6} = 1$$

Beispiel für den Wetterdatensatz

$$Gain(S, A) = H(S) - \sum_{v \in A} \frac{|S_v|}{|S|} H(S_v)$$

- Erst die Gesamtentropie berechnen

  14 Instanzen, 5 × *"play=no"*, 9 × *"play=yes"*:

$$H(S) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940286$$

- Dann für jedes Attribut den Datensatz so unterteilen, dass alle Instanzen für das Attribut A den gleichen Wert haben, und für jeden solchen Teildatensatz die Entropie berechnen

  $A : windy = false$: 8 Instanzen, 6 × *"play=yes"* + 2× *"play=no"*

$$H(S_{false}) = -\frac{6}{8} \log_2 \frac{6}{8} - \frac{2}{8} \log_2 \frac{2}{8} = 0.8112781$$

  $A : windy = true$: 6 Instanzen, 3 × *"play=yes"*, 3 × *"play=no"*:

$$H(S_{true}) = -\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6} = 1$$

Beispiel für den Wetterdatensatz

$$Gain(S, A) = H(S) - \sum_{v \in A} \frac{|S_v|}{|S|} H(S_v)$$

- Erst die Gesamtentropie berechnen

  14 Instanzen, 5 × *"play=no"*, 9 × *"play=yes"*:

$$H(S) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940286$$

- Dann für jedes Attribut den Datensatz so unterteilen, dass alle Instanzen für das Attribut A den gleichen Wert haben, und für jeden solchen Teildatensatz die Entropie berechnen

  A : *windy* = *false*: 8 Instanzen, 6 × *"play=yes"* + 2× *"play=no"*

$$H(S_{false}) = -\frac{6}{8} \log_2 \frac{6}{8} - \frac{2}{8} \log_2 \frac{2}{8} = 0.8112781$$

  A : *windy* = *true*: 6 Instanzen, 3 × *"play=yes"*, 3 × *"play=no"*:

$$H(S_{true}) = -\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6} = 1$$

Beispiel für den Wetterdatensatz

$$Gain(S, A) = H(S) - \sum_{v \in A} \frac{|S_v|}{|S|} H(S_v)$$

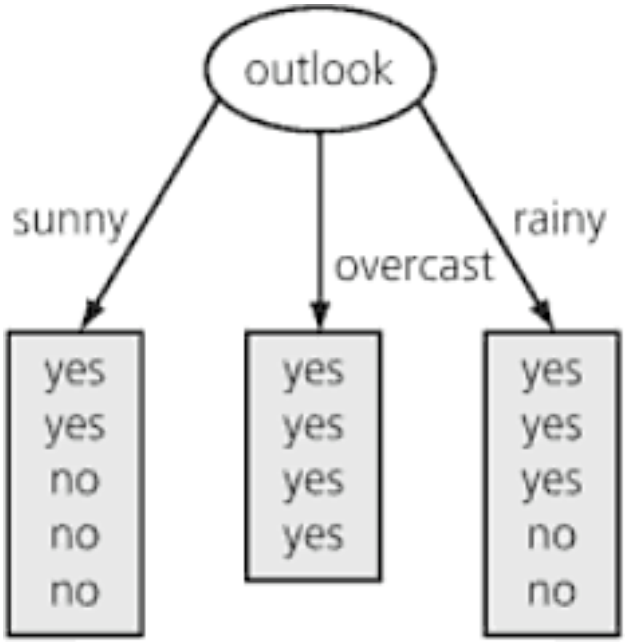
- Daraus und aus der Gesamtentropie setzt sich $Gain(S, windy)$ zusammen:

$$\begin{aligned} Gain(S, windy) &= H(S) - \frac{|S_{false}|}{|S|} H(S_{false}) - \frac{|S_{true}|}{|S|} H(S_{true}) \\ &= 0.940286 - \frac{8}{14} * 0.8112781 - \frac{6}{14} * 1 = 0.04812703 \end{aligned}$$

Beispiel für den Wetterdatensatz

- Ähnlich werden berechnet:

  $Gain(S, windy) = 0.048$
  $Gain(S, outlook) = 0.247$
  $Gain(S, temperature) = 0.029$
  $Gain(S, humidity) = 0.152$

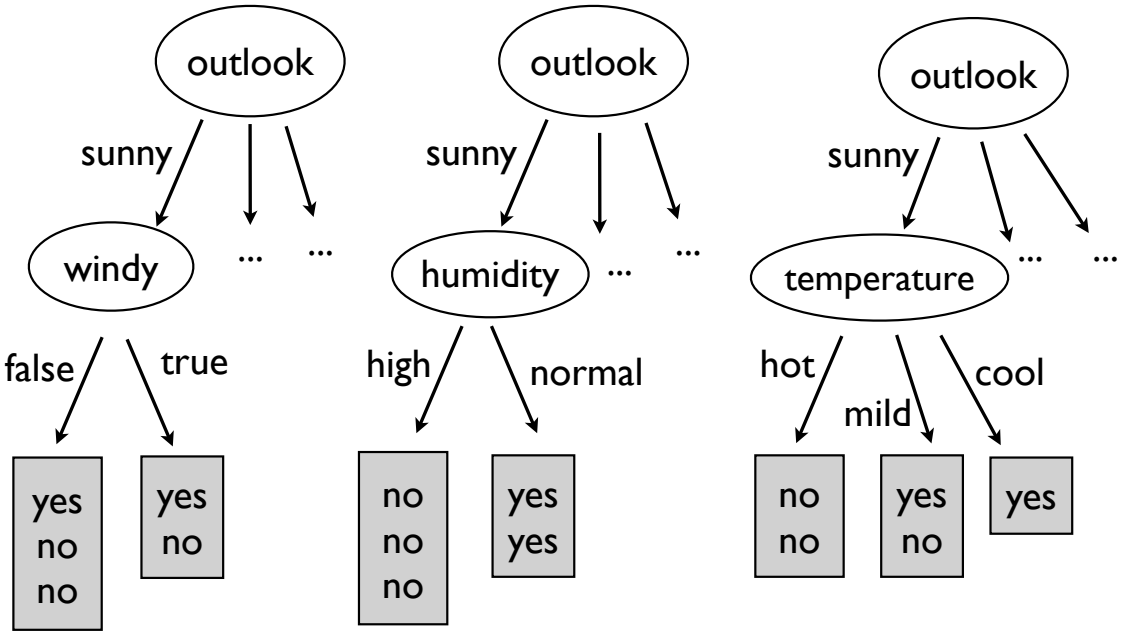


- *Outlook* erzielt den größten Informationsgewinn, daher wird es als Baumwurzel gewählt

- Nächster Schritt:
  Rekursion für jeden Attributwert, z. B. *sunny*,
  5 Instanzen, 2 Mal "*play* = *yes*", 3 Mal "*play* = *no*"

Entscheidungsbäume

Beispiel für den Wetterdatensatz: Rekursion

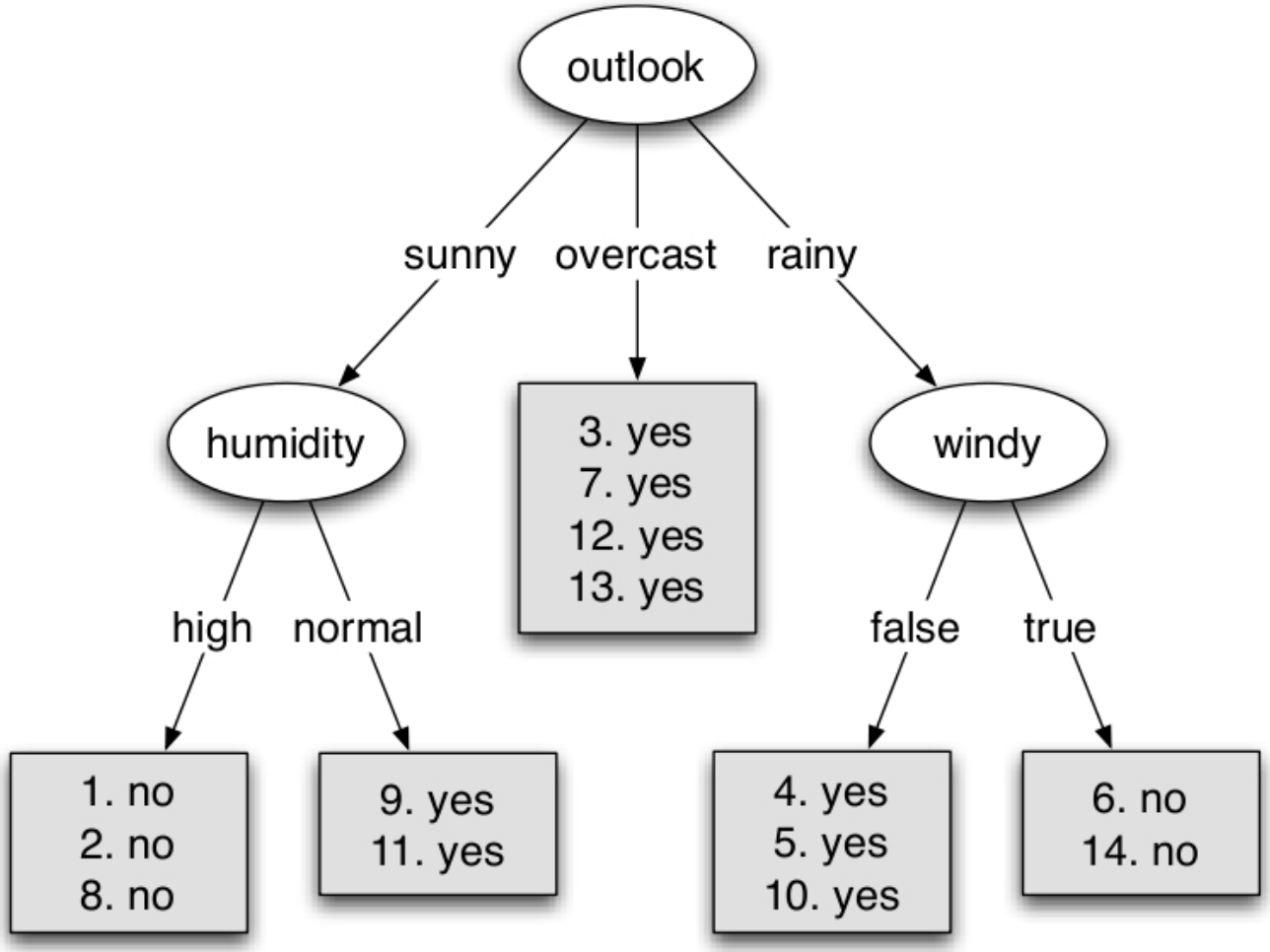


$Gain(S, windy) = 0.020$ $Gain(S, temperature) = 0.571$

$Gain(S, humidity) = 0.971$

Humidity erzielt den größten Informationsgewinn, daher wird es als nächsten Knoten unter "*outlook* = *sunny*" eingesetzt

Baum für den Wetter-Datensatz



Alle Instanzen unter einem Blatt werden gleich klassifiziert (Entropie $= 0$)

Beachte: Attribut *temperature* wurde nicht benutzt.

Entscheidungsbäume

Anmerkung

- Der Pfad zwischen Wurzel und Blatt stellt eine Konjunktion dar
- Jeder Attribut darf auf dem Pfad einmal vorkommen
- Ein Attribut kann in verschiedenen Unterbäume mehrmals vorkommen

Umwandlung in Entscheidungsregeln

```
IF (outlook==sunny) ∧ (humidity==high)
THEN play=no
IF (outlook==sunny) ∧ (humidity==normal)
THEN play=yes
...
```

Entscheidungsbäume

Interpretation

- Wenn das Wetter bewölkt ist, spielen Leute immer
- Es gibt Leute, die sogar bei Regen spielen, aber nicht wenn der Wind weht
- Wenn die Sonne scheint, spielen Leute gerne, außer wenn die Luftfeuchtigkeit hoch ist

Schlussfolgerungen:

- Der Manager kann an Tagen, an denen Regen und starken Wind vorausgesagt wird, seinem Personal frei geben
- An sonnigen Tagen, an denen es Luft gibt, kann er sich überlegen, ob er zusätzlich ein paar Studenten einstellt

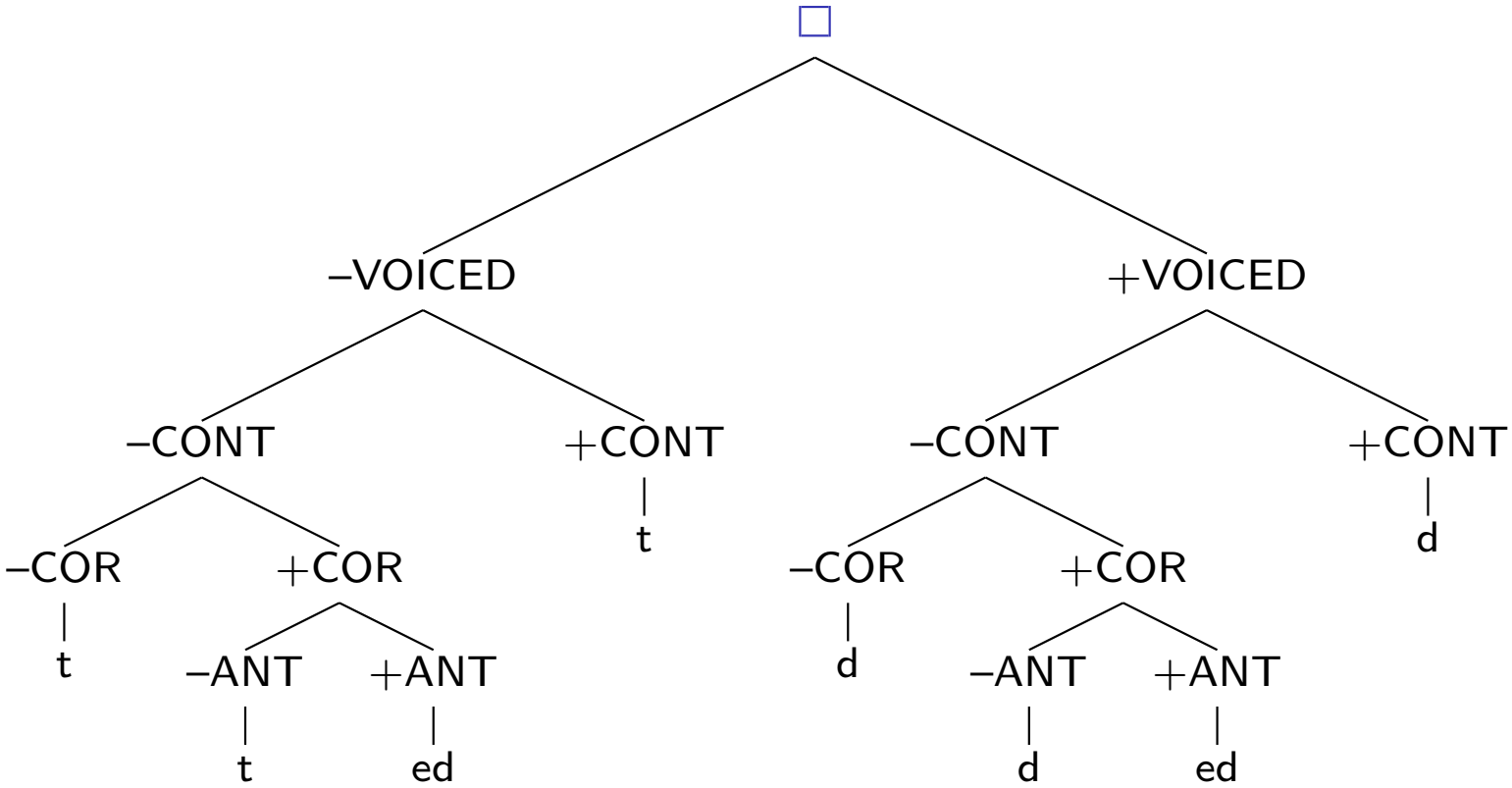
Entscheidungsbäume

Eigenschaften des ID3 Algorithmus

- Die Suche ist unvollständig, d. h. nicht alle möglichen Bäume werden in Betracht gezogen
- Es ist möglich, dass der "beste" (kleinste) Baum nicht gefunden wird (z. B. wenn es einen insgesamt kleineren Baum gibt, der eine andere Wurzel hat, als die, die anfangs den höchsten Informationsgewinn bietet)

Entscheidungsbäume

Beispiel für nicht optimale Struktur (past tense Endungen Englisch):



CONT = continuant; COR = coronal; ANT = anterior

Entscheidungsbäume

Eigenschaften des ID3 Algorithmus

- Die Suche ist unvollständig, d. h. nicht alle möglichen Bäume werden in Betracht gezogen
- Es ist möglich, dass der "beste" (kleinste) Baum nicht gefunden wird (z. B. wenn es einen insgesamt kleineren Baum gibt, der eine andere Wurzel hat, als die, die anfangs den höchsten Informationsgewinn bietet)
- Attribute werden aufgrund des Informationsgewinns über die ganze Datenmenge ausgewählt. Vorteil: robust bzgl. Unregelmäßigkeiten und Fehlern. Nachteil: nicht inkrementell.
- Allgemein werden flache Bäume bevorzugt

Occam's Razor: Einfachste Hypothese die die Daten erklären kann bevorzugen.

- Warum ist es gut, flache Bäume zu bevorzugen?
- Occam's razor: kurze Hypothesen sind langen Hypothesen vorzuziehen.
- Begründung: es gibt weniger kurze Hypothesen als lange, daher ist das Risiko geringer, eine kurze Hypothese zu finden, die die Daten nur aus Zufall erklärt.

Überanpassung (Overfit) vermeiden

- Der ID3 Algorithmus läßt den Baum so lange weiter wachsen, bis die Trainingsdaten so gut wie möglich erklärt werden
  (d.h. für Trainingsdaten ohne Inkonsistenzen wird der Baum so groß bis er alle Beispiele aus den Trainingsdaten perfekt erklärt.)
- Problematisch wenn es fehlerhafte Beobachtungen in den Daten gibt, oder wenn die Beobachtungsmenge zu klein ist.

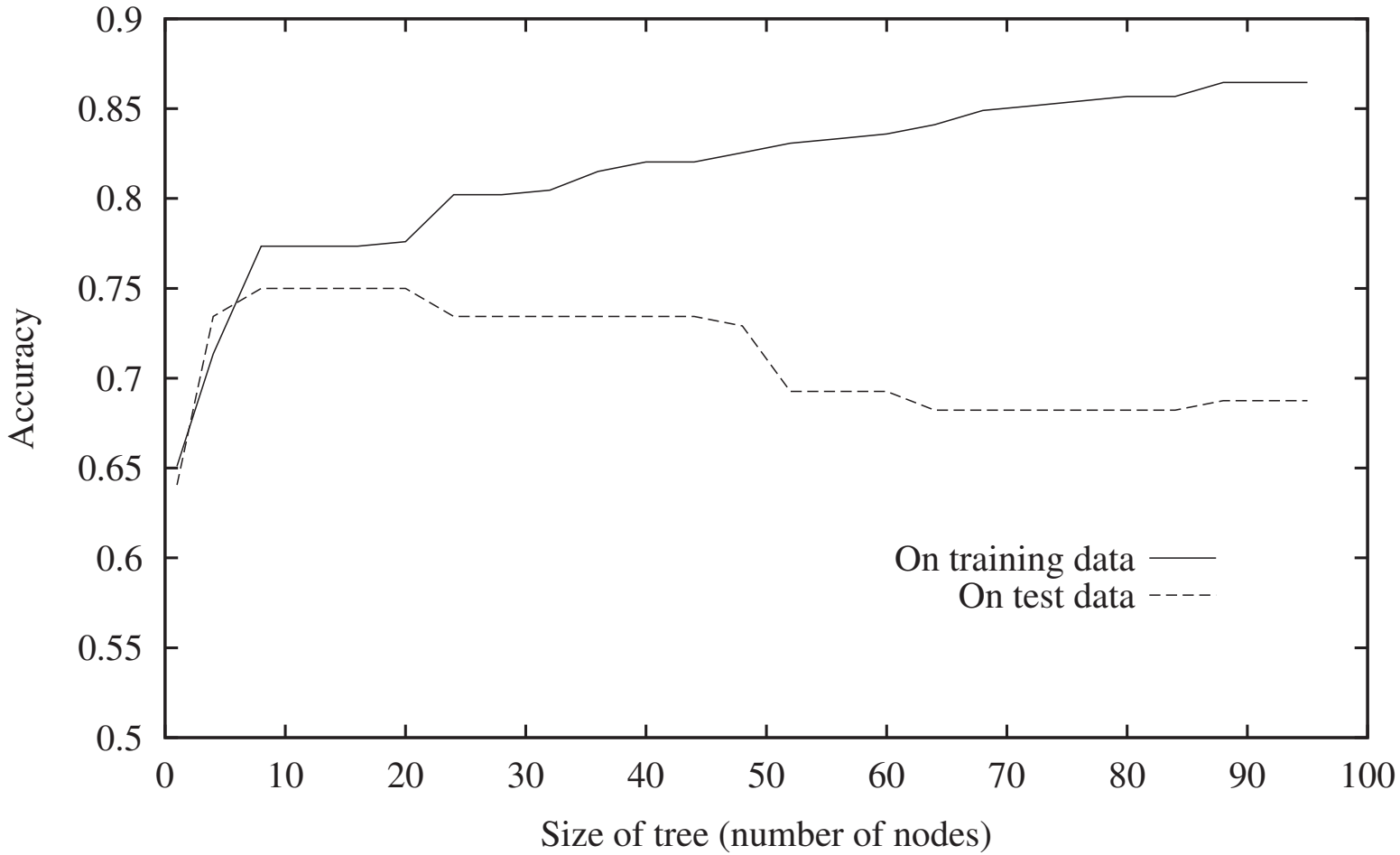
Beispiel

Alle unsere Beobachtungen besagen bislang, dass die Leute spielen wenn es sonnig ist und die Luftfeuchtigkeit nicht zu hoch ist. Wenn wir ein neues Beispiel hinzufügen das dem widerspricht

outlook	temp.	humidity	windy	play
sunny	hot	normal	true	no

wird der Entscheidungsbaum einen neuen Knoten unter Luftfeuchtigkeit hinzufügen und wir werden (falls dies keine generelle Regel ist) später eine höhere Fehlerrate auf Testdaten haben.

Überangepasster Entscheidungsbaum



(Mitchell 1997, p.67)

Baum beschneiden (pruning)

Es gibt zwei Ansätze um Überanpassung zu vermeiden:

- Aufhören den Baum weiter zu bauen bevor er die Trainingsdaten komplett klassifizieren kann
  Problem: Wie erkennen wir, wann genau wir aufhören sollen?
- Erst den ganzen Baum erstellen, und dann nachträglich beschneiden.

Nachträgliches Beschneiden ist sehr viel verbreiteter.

Baum beschneiden (pruning)

Vorgehen für nachträgliches Beschneiden:

- Die Trainingsdaten werden in Trainings- und Validierungsdaten aufgeteilt.
- Validierungsdaten werden benutzt um zu entscheiden wie weit der Baum beschnitten wird.
- Idee: Die Validierungsdaten werden normalerweise nicht die gleichen Fehler und zufälligen Muster enthalten wie die Trainingsdaten.
- Häufige Aufteilung der Daten in 2/3 Training und 1/3 Validierungsdaten.

Baum beschneiden

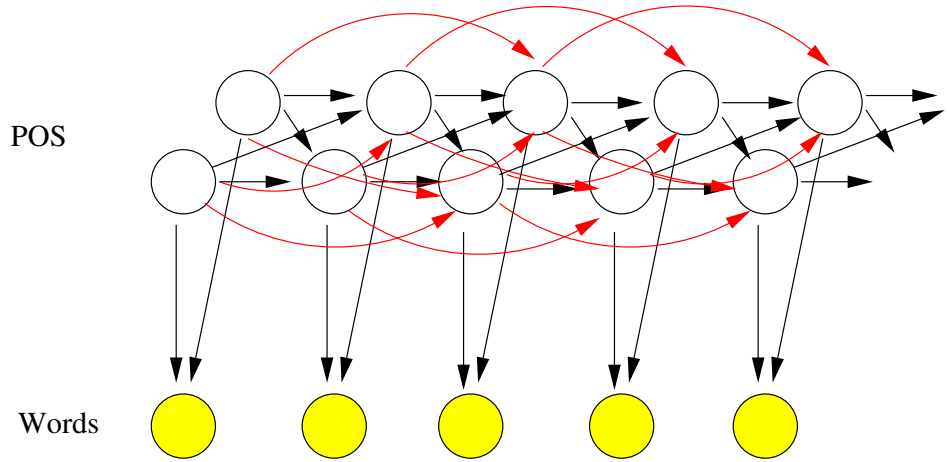
- ein Blatt nach dem anderen abschneiden, immer das am wenigsten hilfreiche Blatt zuerst
- wie “hilfreich” ein Blatt ist, kann z.B. durch die Menge an Evidenz für das Blatt bewertet werden (wie viele Instanzen hilft dieses Blatt richig zu kategorisieren?)
- Nach jedem Abschneiden eines Blattes vergleichen wir, wie gut der Entscheidungsbaum jetzt die Validierungsdaten erklären kann.
- Wir hören auf, herumzuschnippeln wenn die Ergebnisse auf den Validierungsdaten schlechter werden (das ist ein Zeichen, dass wir wertvolle Information wegschneiden).

Inhaltsverzeichnis

Erinnerung: POS tagging mit HMM (Schmid 1994)

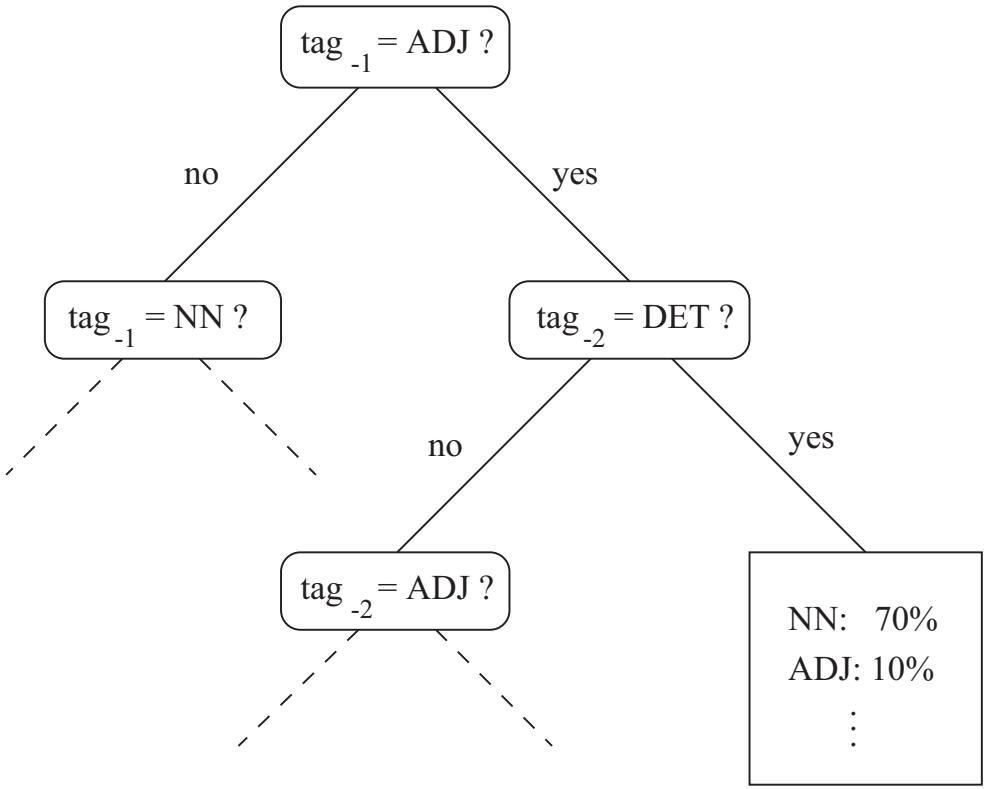
- Wir hatten uns vor einiger Zeit Part-of-Speech tagging mit Hidden Markov Modellen angesehen.
- Dabei kam die Frage auf, wie viel Kontext man braucht um ein POS tag bestimmen zu können.
- Für manche Fälle braucht man mehr als 1 oder 2 Wörter Kontext.
- Wenn man noch längere Kontexte abschätzt, bekommt man aber Probleme mit spärlichen Daten.
- Daher die Idee: flexible Kontextlänge je nach POS-tag, bzw., automatisch die informativsten Teile des Kontexts zu wählen.

2^{nd} order HMM
$P(POS_i|POS_{i-2}, POS_{i-1})$



Entscheidungsbaum beim TreeTagger

- Definiere einen maximalen Kontext (e.g. n=4)
- Mögliche Attribute sind alle möglichen POS tags an allen Positionen im Kontext
- Mögliche Werte eines Attributs: ja oder nein



Entscheidungsbaum beim TreeTagger

- Definiere einen maximalen Kontext (e.g. n=4)
- Mögliche Attribute sind alle möglichen POS tags an allen Positionen im Kontext
- Mögliche Werte eines Attributs: ja oder nein

Warum nicht einfach "POS-tag?" als Frage und die POS tags als Werte?

- Dann würde sich bei jeder Frage die Datenmenge in so viele Teile zerteilen, wie es POS tags gibt.
- Problem: viele leere Mengen würden generiert, Vorhersagen für diese unmöglich

Pruning beim TreeTagger

- Der TreeTagger baut erst einen großen Baum
- Dann wird beschnitten nach dem Kriterium des gewichteten Informationsgewinns

$$G = f(C)Gain(C, A)$$

f(C) = Anzahl von Trigrammen im relevanten Teil des Trainingssets
C = Trainingsdaten
A = Attribut

- Falls der gewichtete Informationsgewinn unter einen Grenzwert fällt, wird ein Blatt weggeschnitten.

Zusammenfassung

- Entscheidungsbäume kann man anwenden für Klassifizierungsprobleme bei denen die Attribute diskrete Werte haben.
- Der ID3 Algorithmus lernt den Entscheidungsbaum von der Wurzel an, basieren auf Informationsgewinn (berechnet durch Entropie).
- Dadurch lernt ID3 generell relative kleine Bäume (aber nicht garantiert optimal).
- Überanpassung kann passieren, wenn fehlerhafte Beobachtungen in den Trainingsdaten sind, oder das Trainingsset zu klein ist.
- Um Überanpassung zu vermeiden, kann der Baum beschnitten werden.
- Aufteilung: Trainingsdaten und Validierungsdaten.