Entscheidungsbäume und Ensemble-Methoden

ML für kleine und/oder tabellarische Daten

Jan Meischner

07. Januar 2026

Agenda

- Motivation: Warum Entscheidungsbäume?
- Ein einfaches Beispiel und Begriffsklärung
- Wie lernen wir einen Baum?
- Überanpassung und Baumgröße
- Motivation für Ensembles
- Random Forests
- (Optional) Bagging und Boosting
- (Optional) Ausblick: gradientenbasiertes Lernen von Bäumen
- Zusammenfassung

Motivation: Warum Entscheidungsbäume?

- **Interpretierbarkeit:**
  - Jeder Pfad von der Wurzel zum Blatt ist eine einfache Wenn-Dann-Regel.
  - Gut geeignet, um Entscheidungen zu erklären.

Motivation: Warum Entscheidungsbäume?

- **Interpretierbarkeit:**
  - Jeder Pfad von der Wurzel zum Blatt ist eine einfache Wenn-Dann-Regel.
  - Gut geeignet, um Entscheidungen zu erklären.
- **Praktisch für tabellarische Daten:**
  - Kategoriale und numerische Merkmale.
  - Kein Feature-Scaling, wenig Vorverarbeitung nötig.

Motivation: Warum Entscheidungsbäume?

- **Interpretierbarkeit:**
  - Jeder Pfad von der Wurzel zum Blatt ist eine einfache Wenn-Dann-Regel.
  - Gut geeignet, um Entscheidungen zu erklären.
- **Praktisch für tabellarische Daten:**
  - Kategoriale und numerische Merkmale.
  - Kein Feature-Scaling, wenig Vorverarbeitung nötig.
- **Nichtlinearität:**
  - Durch rekursive Splits können komplexe Entscheidungsgrenzen entstehen.

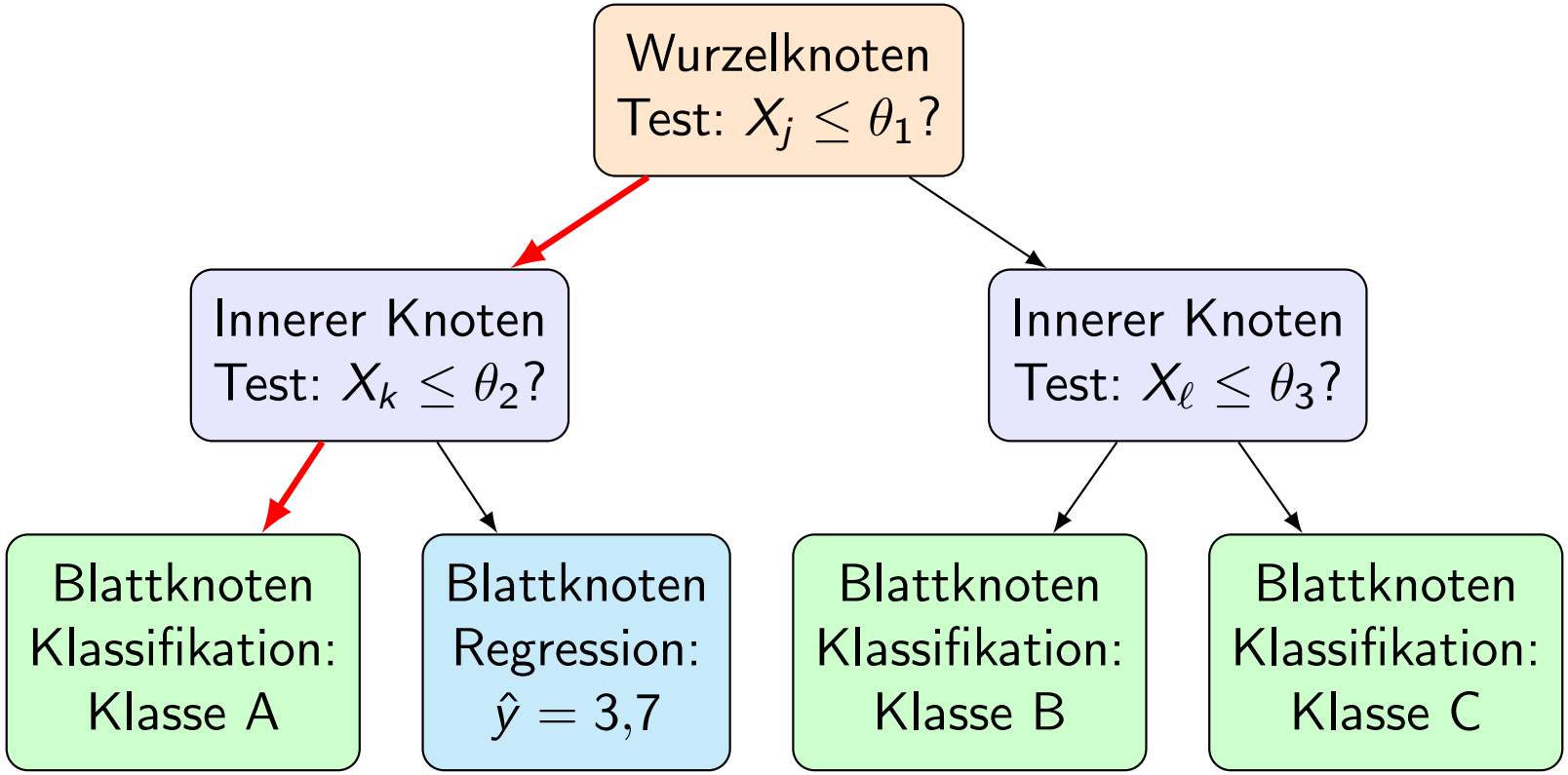
Motivation: Warum Entscheidungsbäume?

- **Interpretierbarkeit:**
  - Jeder Pfad von der Wurzel zum Blatt ist eine einfache Wenn-Dann-Regel.
  - Gut geeignet, um Entscheidungen zu erklären.
- **Praktisch für tabellarische Daten:**
  - Kategoriale und numerische Merkmale.
  - Kein Feature-Scaling, wenig Vorverarbeitung nötig.
- **Nichtlinearität:**
  - Durch rekursive Splits können komplexe Entscheidungsgrenzen entstehen.
- **Effizienz:**
  - Relativ schnelles Training auf kleinen bis mittleren Datensätzen.
  - Schnelle Vorhersage (Pfad entlanglaufen).

Motivation: Warum Entscheidungsbäume?

- **Interpretierbarkeit:**
  - Jeder Pfad von der Wurzel zum Blatt ist eine einfache Wenn-Dann-Regel.
  - Gut geeignet, um Entscheidungen zu erklären.
- **Praktisch für tabellarische Daten:**
  - Kategoriale und numerische Merkmale.
  - Kein Feature-Scaling, wenig Vorverarbeitung nötig.
- **Nichtlinearität:**
  - Durch rekursive Splits können komplexe Entscheidungsgrenzen entstehen.
- **Effizienz:**
  - Relativ schnelles Training auf kleinen bis mittleren Datensätzen.
  - Schnelle Vorhersage (Pfad entlanglaufen).
- **Baustein für Ensembles:**
  - Random Forests, Gradient Boosting, neuere Tree-Ensembles.

Visualisierung & Begriffe



Instanz: $x = (x_1, \ldots, x_d)$ wandert von der Wurzel entlang eines Entscheidungspfads bis zum Blatt.
Merkmale: $X_1, \ldots, X_d$ sind die Eingangsvariablen, die in den Tests vorkommen.

Beispiel: Tennisspieler

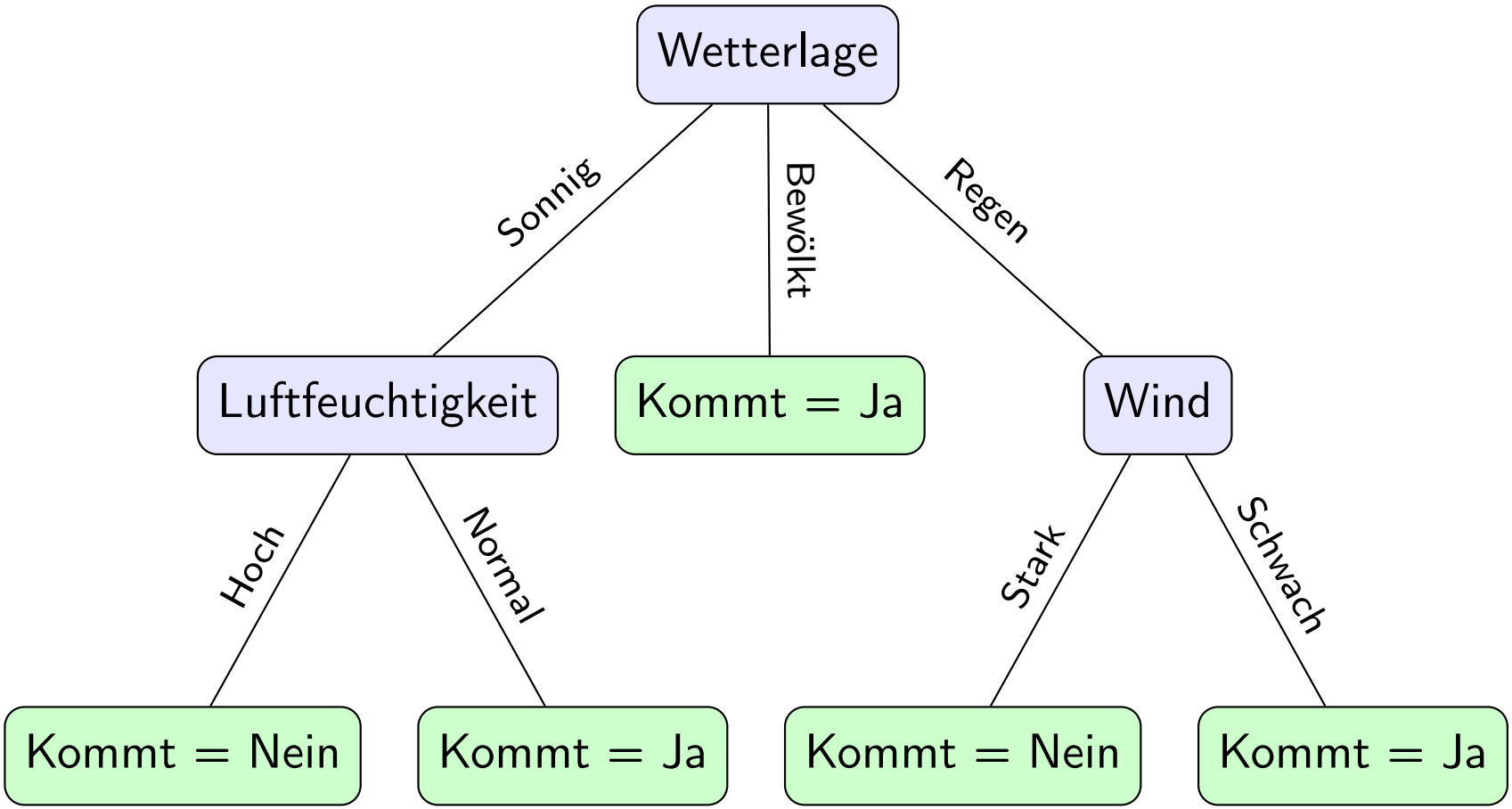
Nr	Wetterlage	Temperatur	Luftfeuchtigkeit	Wind	Kommt
1	Sonnig	Heiß	Hoch	Schwach	Nein
2	Sonnig	Heiß	Hoch	Stark	Nein
3	Bewölkt	Heiß	Hoch	Schwach	Ja
4	Regen	Mild	Hoch	Schwach	Ja
5	Regen	Kühl	Normal	Schwach	Ja
6	Regen	Kühl	Normal	Stark	Nein
7	Bewölkt	Kühl	Normal	Stark	Ja
8	Sonnig	Mild	Hoch	Schwach	Nein
9	Sonnig	Kühl	Normal	Schwach	Ja
10	Regen	Mild	Normal	Schwach	Ja
11	Sonnig	Mild	Normal	Stark	Ja
12	Bewölkt	Mild	Hoch	Stark	Ja
13	Bewölkt	Heiß	Normal	Schwach	Ja
14	Regen	Mild	Hoch	Stark	Nein

- **Fragestellung:** "Wir beobachten aus dem Fenster den Tennisplatz gegenüber und fragen uns: Unter welchen Wetterbedingungen kommt der Spieler auf den Platz?"

Beispiel: Tennisspieler

Nr	Wetterlage	Temperatur	Luftfeuchtigkeit	Wind	Kommt
Sonnig → Split nach Luftfeuchtigkeit					
1	Sonnig	Heiß	Hoch	Schwach	Nein
2	Sonnig	Heiß	Hoch	Stark	Nein
8	Sonnig	Mild	Hoch	Schwach	Nein
11	Sonnig	Mild	Normal	Stark	Ja
9	Sonnig	Kühl	Normal	Schwach	Ja
Bewölkt → Blatt: Kommt = Ja					
3	Bewölkt	Heiß	Hoch	Schwach	Ja
12	Bewölkt	Mild	Hoch	Stark	Ja
13	Bewölkt	Heiß	Normal	Schwach	Ja
7	Bewölkt	Kühl	Normal	Stark	Ja
Regen → Split nach Wind					
6	Regen	Kühl	Normal	Stark	Nein
14	Regen	Mild	Hoch	Stark	Nein
4	Regen	Mild	Hoch	Schwach	Ja
5	Regen	Kühl	Normal	Schwach	Ja
10	Regen	Mild	Normal	Schwach	Ja

Beispiel: Tennisspieler



Vorhersage mit einem Entscheidungsbaum

Nr	Wetterlage	Temperatur	Luftfeuchtigkeit	Wind	Kommt
15	Sonnig	Heiß	Normal	Schwach	?

Allgemeiner Algorithmus:

- Eingabe: neue Instanz $x = (x_1, \ldots, x_d)$.
- Algorithmus:
  1. **Beginne an der Wurzel.**
  2. Prüfe den Test des aktuellen Knotens.
  3. Folge der passenden Kante.
  4. Wiederhole, bis du ein Blatt erreichst.
- Ausgabe: Label oder Wert im Blatt.

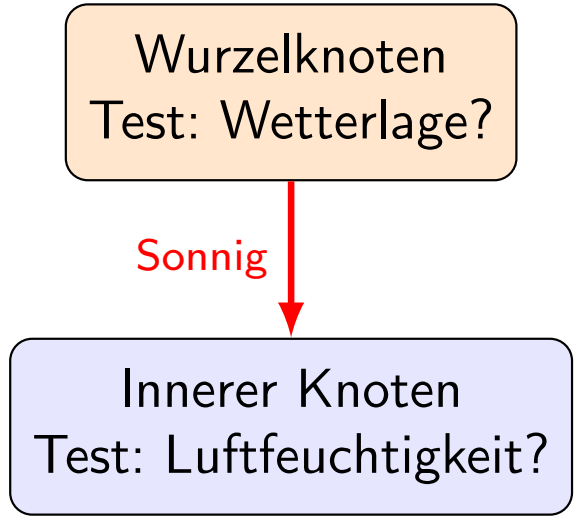
Wurzelknoten
Test: Wetterlage?

Vorhersage mit einem Entscheidungsbaum

Nr	Wetterlage	Temperatur	Luftfeuchtigkeit	Wind	Kommt
15	Sonnig	Heiß	Normal	Schwach	?

Allgemeiner Algorithmus:

- Eingabe: neue Instanz $x = (x_1, \ldots, x_d)$.
- Algorithmus:
  1. Beginne an der Wurzel.
  2. **Prüfe den Test des aktuellen Knotens.**
  3. **Folge der passenden Kante.**
  4. Wiederhole, bis du ein Blatt erreichst.
- Ausgabe: Label oder Wert im Blatt.

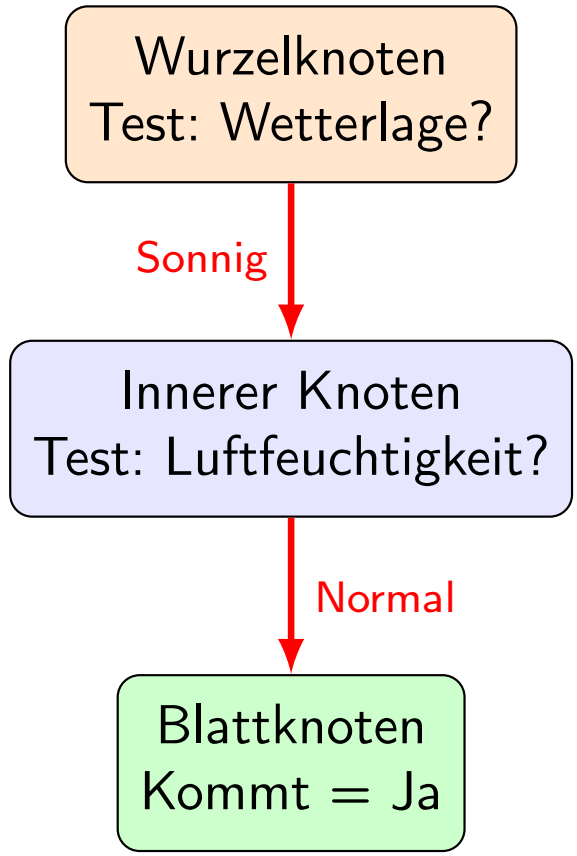


Vorhersage mit einem Entscheidungsbaum

Nr	Wetterlage	Temperatur	Luftfeuchtigkeit	Wind	Kommt
15	Sonnig	Heiß	Normal	Schwach	?

Allgemeiner Algorithmus:

- Eingabe: neue Instanz $x = (x_1, \ldots, x_d)$.
- Algorithmus:
  1. Beginne an der Wurzel.
  2. Prüfe den Test des aktuellen Knotens.
  3. Folge der passenden Kante.
  4. **Wiederhole, bis du ein Blatt erreichst.**



Ausgabe: $\hat{y}(x_{11})$: Kommt = Ja.

Wie lernt man einen Entscheidungsbaum?

Wir müssen uns über die folgenden Fragen Gedanken machen:

1. Wollen wir an inneren Knoten nur binäre- oder auch n-äre Splits erlauben?
2. Wie entscheiden wir, welches Merkmal wir zum Splitten verwenden?
3. Wann erzeugen wir einen Blattknoten und wie weisen wir das Klassenlabel zu?
4. Wie groß wollen wir unseren Baum wachsen lassen?

Wie splitten wir Entscheidungsbäume?

In der Praxis verwenden Entscheidungsbäume immer binäre Splits.

Gründe:

- **Effiziente Optimierung:** Binäre Splits erlauben eine saubere Suche nach der besten Schwelle ($X_j \leq \theta$). Mehrere Intervalle führen zu einem komplexen, kombinatorischen Optimierungsproblem.
- **Regularisierung:** Splits mit mehr als zwei Bereichen erzeugen schnell sehr kleine Teilmengen und verstärken Overfitting.
- **Interpretierbarkeit:** Tests wie „$X_j \leq \theta$?“ sind gut verständlich; n-äre Knoten werden unübersichtlich.
- **Praxis:** Gängige Baum-Algorithmen nutzen für numerische Merkmale ausschließlich binäre Splits (CART, ID3, C4.5, Random Forests, XGBoost, LightGBM).

Fazit: Für numerische Merkmale ist der binäre Split der robuste, interpretierbare und optimierbare Standard.

Wie lernt man einen Entscheidungsbaum?

- Gegeben: Trainingsmenge $S = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$.
- Ziel: Finde Baumstruktur und Blattvorhersagen, die y aus x möglichst gut vorhersagen.
- Klassischer Ansatz: Top-down, greedy, rekursiv.
- Idee:
  - Starte mit allen Instanzen in der Wurzel.
  - Wähle den **besten Split** (Merkmal + Schwelle).
  - Teile in Teilmengen und wiederhole im jeweiligen Teilbaum.

CART: Grundschema (informell)

CART = *Classification And Regression Trees* (Breiman et al.)

Rekursiver Algorithmus

Gegeben Teilmenge S der Trainingsdaten:

1. Wenn alle $y^{(i)}$ in S die gleiche Klasse haben:
   - Erzeuge ein Blatt mit dieser Klasse.
2. Sonst:
   - Prüfe Stopkriterien (z. B. maximale Tiefe, minimale Knotengröße).
   - Falls stoppen: Blatt mit Mehrheitsklasse in S.
   - Falls nicht: Wähle Merkmal und Split, die die **Impurity** am stärksten verringern.
   - Teile S in S_{links} und S_{rechts} (binärer Split).
   - Rufe Algorithmus rekursiv auf S_{links} und S_{rechts} auf.

Impurity: Unreinheit eines Knotens

Für Klassifikation sei p_k der Anteil der Klasse k in einem Knoten.

Gini-Impurity (CART):

$$G(S) = \sum_k p_k(1 - p_k) = 1 - \sum_k p_k^2.$$

- $G(S) = 0$ bei reiner Klasse (ein $p_k = 1$, alle anderen 0).
- Maximal, wenn Klassen gleich vertreten sind.

Entropie (ID3/C4.5):

$$H(S) = -\sum_k p_k \log_2 p_k.$$

- $H(S) = 0$ bei reiner Klasse.
- Maximale Unreinheit bei Gleichverteilung der Klassen.

Beide Maße verhalten sich ähnlich; CART nutzt Gini aus Effizienzgründen.

Gini-Impurity als Varianz eines Klassenindikators

Idee: Wir betrachten für jede Klasse einen Indikator

$$Y_k = \begin{cases} 1 & \text{wenn Beispiel zur Klasse } k \text{ gehört} \\ 0 & \text{sonst} \end{cases}$$

mit Wahrscheinlichkeit p_k = Anteil der Klassen k im Knoten.

Die Varianz dieses Indikators ist:

$$\begin{aligned} \mathrm{Var}(Y) &= \mathbb{E}\big[(Y - \mathbb{E}[Y])^2\big] \\ &= \mathbb{E}[Y^2] - \big(\mathbb{E}[Y]\big)^2 \\ &= p - p^2 = p(1-p) \end{aligned}$$

Die Gini-Impurity ist die Summe dieser Varianzen über alle Klassen:

$$G = \sum_{k=1}^{K} p_k(1-p_k) \quad = \quad 1 - \sum_{k=1}^{K} p_k^2.$$

Beispiel: Gini-Impurity berechnen

Knoten mit 9 Instanzen:

- 6 mal Klasse `ja`, 3 mal Klasse `nein`.

$$
\begin{aligned}
p_{\text{ja}} = \frac{6}{9}, \quad p_{\text{nein}} &= \frac{3}{9} \\
G(S) &= 1 - \left(p_{\text{ja}}^2 + p_{\text{nein}}^2\right) \\
&= 1 - \left(\frac{4}{9} + \frac{1}{9}\right) \\
&= 1 - \frac{5}{9} = \frac{4}{9} \approx 0,44
\end{aligned}
$$

Interpretation:

- $G = 0$: reine Knoten (keine Streuung).
- G groß: durchmischte Klassen $\rightarrow$ hohe Unsicherheit.

Split-Kriterium: Impurity-Reduktion

Gegeben ein Knoten S und ein möglicher Split in S_1 und S_2:

$$\Delta G = G(S) - \left(\frac{|S_1|}{|S|} G(S_1) + \frac{|S_2|}{|S|} G(S_2) \right).$$

- Wir wählen den Split mit **größter** Impurity-Reduktion ΔG.
- Analog mit Entropie als $H(S)$ statt $G(S)$.

Gini-Reduktion für Merkmal *Wetterlage*

Daten (Gesamtmenge S):

Nr	Wetterlage	Temperatur	Luftfeuchtigkeit	Wind	Kommt
1	Sonnig	Heiß	Hoch	Schwach	Nein
2	Sonnig	Heiß	Hoch	Stark	Nein
3	Bewölkt	Heiß	Hoch	Schwach	Ja
4	Regen	Mild	Hoch	Schwach	Ja
5	Regen	Kühl	Normal	Schwach	Ja
6	Regen	Kühl	Normal	Stark	Nein
7	Bewölkt	Kühl	Normal	Stark	Ja
8	Sonnig	Mild	Hoch	Schwach	Nein
9	Sonnig	Kühl	Normal	Schwach	Ja
10	Regen	Mild	Normal	Schwach	Ja
11	Sonnig	Mild	Normal	Stark	Ja
12	Bewölkt	Mild	Hoch	Stark	Ja
13	Bewölkt	Heiß	Normal	Schwach	Ja
14	Regen	Mild	Hoch	Stark	Nein

Gini-Reduktion für Merkmal *Wetterlage*

Klassenverteilung in S:

$$|S| = 14, \quad 9 \text{ Ja}, \ 5 \text{ Nein}.$$

Gini am Wurzelknoten:

$$G(S) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 1 - \frac{81}{196} - \frac{25}{196} = \frac{90}{196} \approx 0{,}46.$$

Frage: Wie stark reduziert ein Split nach *Wetterlage* diese Unreinheit?

Split-Kandidat *Wetterlage = Bewölkt*

Binärer Split (CART): $\boxed{\textit{Wetterlage} \in \{\text{Bewölkt}\}\ ?}$

- Linke Teilmenge $S_L = \{\text{Bewölkt}\}$: $|S_L| = 4,\ 4\,\text{Ja},\ 0\,\text{Nein}$

$$G(S_L) = 0.$$

- Rechte Teilmenge $S_R = \{\text{Sonnig, Regen}\}$: $|S_R| = 10,\ 5\,\text{Ja},\ 5\,\text{Nein}$

$$G(S_R) = 1 - \left(\frac{5}{10}\right)^2 - \left(\frac{5}{10}\right)^2 = 0{,}5.$$

Gewichtete Gini nach dem Split:

$$G_{\text{nach}} = \frac{4}{14} \cdot 0 + \frac{10}{14} \cdot 0{,}5 = \frac{5}{14} \approx 0{,}357.$$

Gini-Reduktion:

$$\Delta G = G(S) - G_{\text{nach}} = \frac{90}{196} - \frac{5}{14} = \frac{20}{196} \approx 0{,}102.$$

Split-Kandidat *Wetterlage = Sonnig*

Binärer Split (CART): $\boxed{\textit{Wetterlage} \in \{\text{Sonnig}\}\ ?}$

- $S_L = \{\text{Sonnig}\}$: $|S_L| = 5,\ 2\,\text{Ja},\ 3\,\text{Nein}$

$$G(S_L) = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = \frac{12}{25} = 0{,}48.$$

- $S_R = \{\text{Bewölkt, Regen}\}$: $|S_R| = 9,\ 7\,\text{Ja},\ 2\,\text{Nein}$

$$G(S_R) = 1 - \left(\frac{7}{9}\right)^2 - \left(\frac{2}{9}\right)^2 = \frac{28}{81} \approx 0{,}346.$$

Gewichtete Gini nach dem Split:

$$G_{\text{nach}} = \frac{5}{14} \cdot \frac{12}{25} + \frac{9}{14} \cdot \frac{28}{81} = \frac{124}{315} \approx 0{,}394.$$

Gini-Reduktion:

$$\Delta G = \frac{90}{196} - \frac{124}{315} = \frac{289}{4410} \approx 0{,}066.$$

Split-Kandidat *Wetterlage = Regen*

Binärer Split (CART): $\boxed{\textit{Wetterlage} \in \{\text{Regen}\}\ ?}$

- $S_L = \{\text{Regen}\}$: $|S_L| = 5,\ 3\,\text{Ja},\ 2\,\text{Nein}$

$$G(S_L) = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = \frac{12}{25} = 0{,}48.$$

- $S_R = \{\text{Sonnig, Bewölkt}\}$: $|S_R| = 9,\ 6\,\text{Ja},\ 3\,\text{Nein}$

$$G(S_R) = 1 - \left(\frac{6}{9}\right)^2 - \left(\frac{3}{9}\right)^2 = \frac{4}{9} \approx 0{,}444.$$

Gewichtete Gini nach dem Split:

$$G_{\text{nach}} = \frac{5}{14} \cdot \frac{12}{25} + \frac{9}{14} \cdot \frac{4}{9} = \frac{16}{35} \approx 0{,}457.$$

Gini-Reduktion:

$$\Delta G = \frac{90}{196} - \frac{16}{35} = \frac{1}{490} \approx 0{,}002.$$

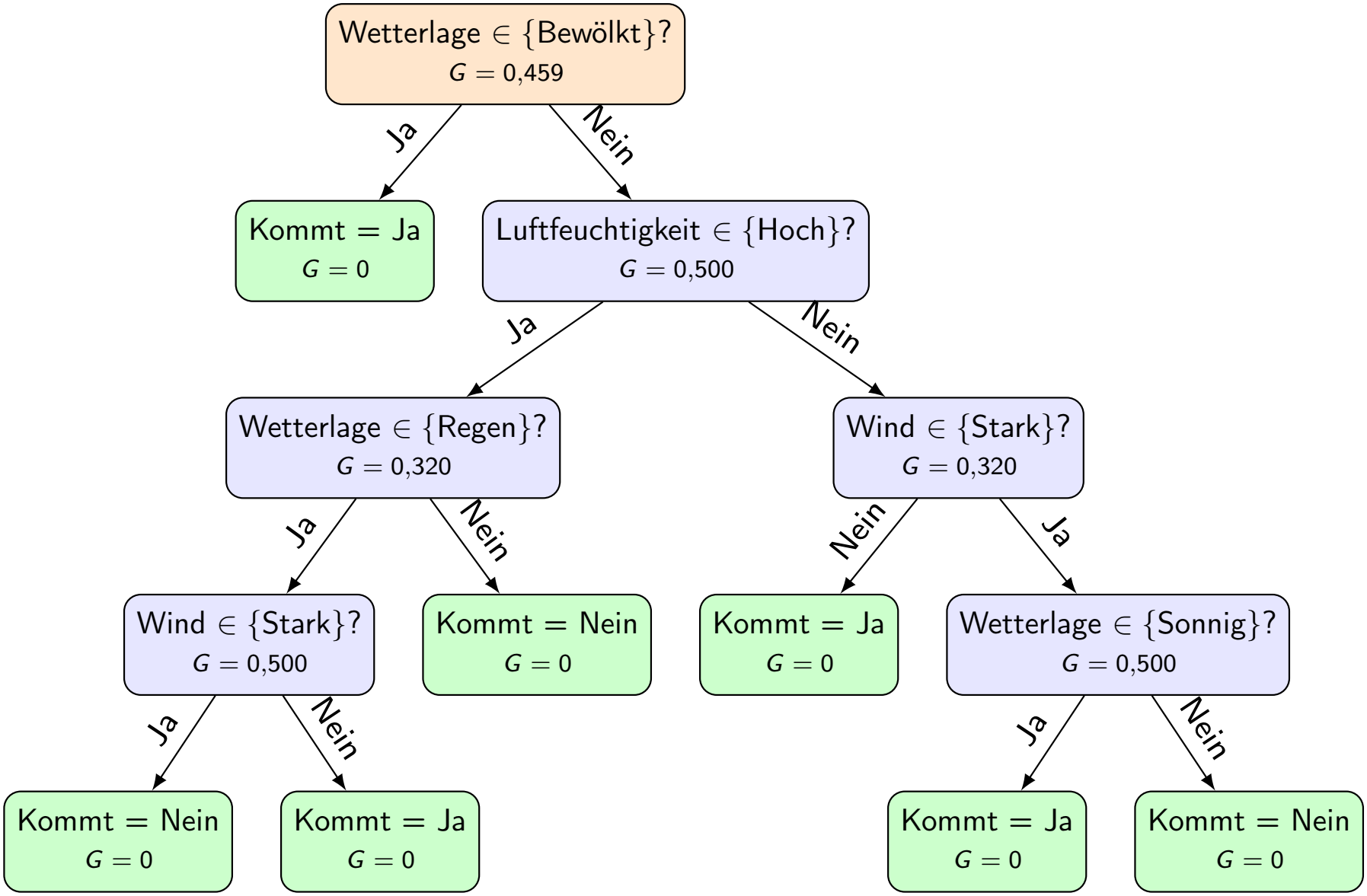
Alle Kandidaten für den ersten Split

$$G(S) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = \frac{45}{98} \approx 0{,}459$$

Feature	**Test (Wert)**	ΔG
Wetterlage	$\in \{\text{Bewölkt}\}$?	**0,1020**
Wetterlage	$\in \{\text{Sonnig}\}$?	0,0655
Wetterlage	$\in \{\text{Regen}\}$?	0,0020
Temperatur	$\in \{\text{Heiß}\}$?	0,0163
Temperatur	$\in \{\text{Kühl}\}$?	0,0092
Temperatur	$\in \{\text{Mild}\}$?	0,0009
Luftfeuchtigkeit	$\in \{\text{Hoch}\}$?	0,0918
Luftfeuchtigkeit	$\in \{\text{Normal}\}$?	0,0918
Wind	$\in \{\text{Schwach}\}$?	0,0306
Wind	$\in \{\text{Stark}\}$?	0,0306

Bester erster Split: $\boxed{\text{Wetterlage} \in \{\text{Bewölkt}\}?}$

Resultierender Entscheidungsbaum



Blattknoten und Labels

- **Klassifikation:**
  - Häufig: Blattlabel = Mehrheitsklasse im Knoten.
  - Option: speichere Klassenverteilung und nutze Wahrscheinlichkeiten.

Blattknoten und Labels

- **Klassifikation:**
  - Häufig: Blattlabel = Mehrheitsklasse im Knoten.
  - Option: speichere Klassenverteilung und nutze Wahrscheinlichkeiten.
- **Regression:**
  - Blattwert = Mittelwert der Zielwerte im Knoten.

Blattknoten und Labels

- **Klassifikation:**
  - Häufig: Blattlabel = Mehrheitsklasse im Knoten.
  - Option: speichere Klassenverteilung und nutze Wahrscheinlichkeiten.
- **Regression:**
  - Blattwert = Mittelwert der Zielwerte im Knoten.
- **Stopkriterien (Pre-Pruning):**
  - Maximale Tiefe erreicht.
  - Weniger als n_{min} Instanzen im Knoten.
  - Keine sinnvolle Impurity-Reduktion mehr.

Überanpassung und Pruning

- Wenn wir den Baum immer weiter wachsen lassen:
  - Jeder Knoten kann am Ende sehr wenige Instanzen enthalten.
  - Der Baum passt sich stark an Zufälligkeiten im Trainingsset an.

Überanpassung und Pruning

- Wenn wir den Baum immer weiter wachsen lassen:
  - Jeder Knoten kann am Ende sehr wenige Instanzen enthalten.
  - Der Baum passt sich stark an Zufälligkeiten im Trainingsset an.
- **Pre-Pruning:**
  - Wachstum frühzeitig stoppen.
  - Gefahr: zu früh gestoppt $\Rightarrow$ zu hoher Bias.

Überanpassung und Pruning

- Wenn wir den Baum immer weiter wachsen lassen:
  - Jeder Knoten kann am Ende sehr wenige Instanzen enthalten.
  - Der Baum passt sich stark an Zufälligkeiten im Trainingsset an.
- **Pre-Pruning:**
  - Wachstum frühzeitig stoppen.
  - Gefahr: zu früh gestoppt $\Rightarrow$ zu hoher Bias.
- **Post-Pruning:**
  - Baum zunächst groß wachsen lassen.
  - Dann Teilbäume abschneiden, die die Validierungsleistung nicht verbessern.

Überanpassung und Pruning

- Wenn wir den Baum immer weiter wachsen lassen:
  - Jeder Knoten kann am Ende sehr wenige Instanzen enthalten.
  - Der Baum passt sich stark an Zufälligkeiten im Trainingsset an.
- **Pre-Pruning:**
  - Wachstum frühzeitig stoppen.
  - Gefahr: zu früh gestoppt $\Rightarrow$ zu hoher Bias.
- **Post-Pruning:**
  - Baum zunächst groß wachsen lassen.
  - Dann Teilbäume abschneiden, die die Validierungsleistung nicht verbessern.
- Entscheidungsbäume sind tendenziell **hochvariante** Modelle.
  - Kleine Änderungen in den Daten führen zu anderen Bäumen.

Zwischenfazit: Entscheidungsbäume

Stärken

- Interpretierbar.
- Flexibel für viele Datentypen.
- Relativ einfach zu implementieren.
- Grundlage vieler leistungsfähiger Ensembles.

Schwächen

- Neigen zum Overfitting.
- Hohe Varianz.
- Einzelbaum oft nicht State-of-the-Art in Genauigkeit.

Motivation für Ensembles

- Idee: Statt **einen** Baum zu trainieren, trainieren wir **viele** Bäume.
- Analogie: Viele unterschiedliche Meinungen gemittelt $\Rightarrow$ robustere Entscheidung.
- Ziel:
  - Varianz reduzieren.
  - Generalisierung verbessern.

Motivation für Ensembles

- Idee: Statt **einen** Baum zu trainieren, trainieren wir **viele** Bäume.
- Analogie: Viele unterschiedliche Meinungen gemittelt $\Rightarrow$ robustere Entscheidung.
- Ziel:
  - Varianz reduzieren.
  - Generalisierung verbessern.
- Zwei wichtige Grundideen:
  1. **Bagging** (Bootstrap Aggregating)
  2. **Boosting** (sequentielle Fehlerkorrektur)
- Random Forests sind ein Spezialfall von Bagging mit Entscheidungsbäumen.

Bagging: Bootstrap Aggregating

- Erzeuge B Bootstrap-Samples:
  - Ziehe n Instanzen mit Zurücklegen aus dem Trainingsset.
  - Einige Instanzen werden mehrfach gezogen, andere gar nicht.

Bagging: Bootstrap Aggregating

- Erzeuge B Bootstrap-Samples:
  - Ziehe n Instanzen mit Zurücklegen aus dem Trainingsset.
  - Einige Instanzen werden mehrfach gezogen, andere gar nicht.
- Trainiere auf jedem Sample einen eigenen Entscheidungsbaum (oft wenig oder kein Pruning).
- Für die Vorhersage:
  - Klassifikation: Mehrheitsvotum der Bäume.
  - Regression: Mittelwert der Vorhersagen.

Bagging: Bootstrap Aggregating

- Erzeuge B Bootstrap-Samples:
  - Ziehe n Instanzen mit Zurücklegen aus dem Trainingsset.
  - Einige Instanzen werden mehrfach gezogen, andere gar nicht.
- Trainiere auf jedem Sample einen eigenen Entscheidungsbaum (oft wenig oder kein Pruning).
- Für die Vorhersage:
  - Klassifikation: Mehrheitsvotum der Bäume.
  - Regression: Mittelwert der Vorhersagen.
- Effekt:
  - Reduktion der Varianz.
  - Robustheit gegenüber Ausreißern und Rauschen.

Random Forest: Bagging + Feature-Sampling

- Random Forest = Bagging mit Entscheidungsbäumen plus zusätzlicher Randomisierung:
  - Für jeden Baum: Bootstrap-Sample der Daten.
  - An jedem Split: Nur eine zufällige Teilmenge von Merkmalen wird betrachtet.

Random Forest: Bagging + Feature-Sampling

- Random Forest = Bagging mit Entscheidungsbäumen plus zusätzlicher Randomisierung:
  - Für jeden Baum: Bootstrap-Sample der Daten.
  - An jedem Split: Nur eine zufällige Teilmenge von Merkmalen wird betrachtet.
- Vorteil:
  - Bäume werden stärker voneinander verschieden.
  - Geringere Korrelation der Bäume.
  - Bessere Ensemble-Wirkung.

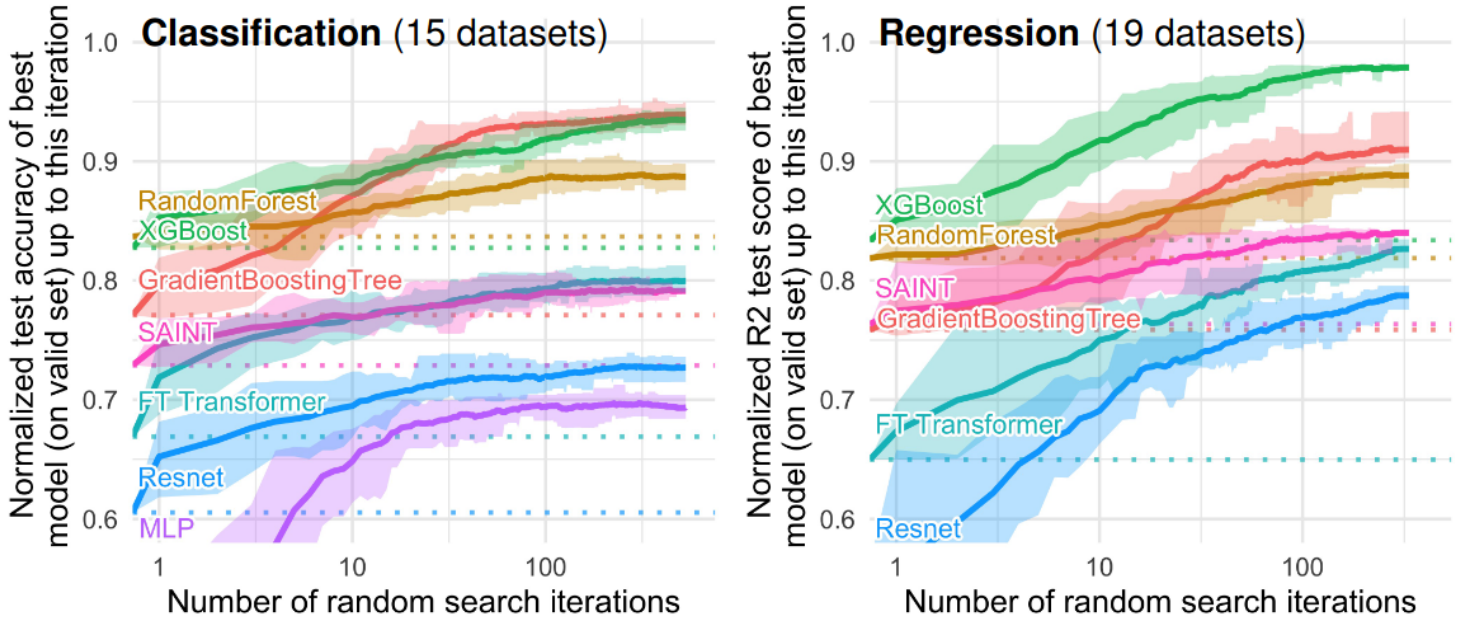
Random Forest: Bagging + Feature-Sampling

- Random Forest = Bagging mit Entscheidungsbäumen plus zusätzlicher Randomisierung:
  - Für jeden Baum: Bootstrap-Sample der Daten.
  - An jedem Split: Nur eine zufällige Teilmenge von Merkmalen wird betrachtet.
- Vorteil:
  - Bäume werden stärker voneinander verschieden.
  - Geringere Korrelation der Bäume.
  - Bessere Ensemble-Wirkung.
- In vielen praktischen Anwendungen:
  - Sehr gute Performance auf tabellarischen Daten.
  - Wenig Hyperparameter, robust.

Random Forests

Weitere Modelle, die auf ähnlichen Ideen basieren und oft noch höhere Genauigkeit erreichen

- Gradient-boosted Trees (XGBoost)
- CatBoost

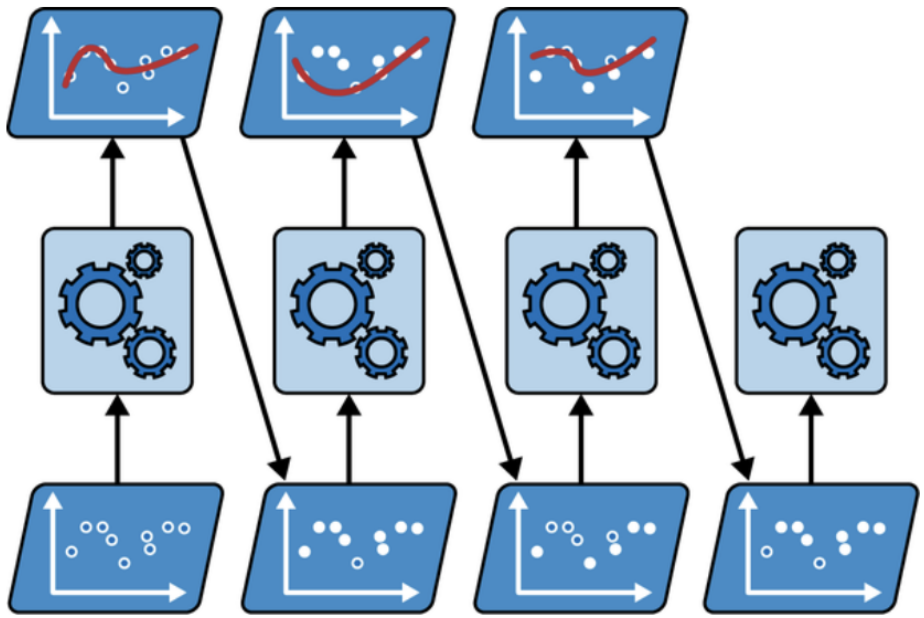


(Optional) Boosting: Schwache Lerner stark machen

- Basisidee:
  - Trainiere Modelle sequentiell.
  - Jedes neue Modell fokussiert die Fehler der bisherigen Modelle.

AdaBoost

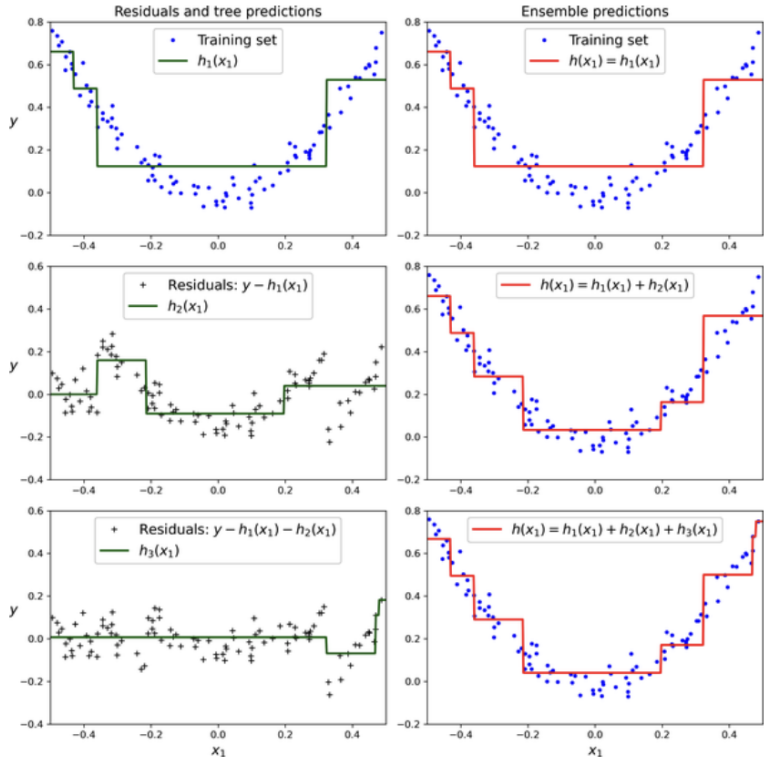
- Anfang: Alle Instanzen gleich gewichtet.
- Nach jedem Baum: Erhöhe die Gewichte falsch klassifizierter Instanzen.
- Kombiniere die Bäume mit Gewichten.



Quelle: Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, 3rd Edition, O'Reilly Media, 2022.

Gradient Boosting (z. B. XGBoost)

- Betrachte Fehler als Residuen.
- Jeder neue Baum approximiert einen Schritt in Richtung Gradient der Loss-Funktion.



Quelle: Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, 3rd Edition, O'Reilly Media, 2022.

(Optional) Ausblick: Gradient-basierte Baum-Ensembles

- Klassische Entscheidungsbäume sind nicht-differenzierbar:
  - Splits sind harte Entscheidungen.
  - Training basiert auf heuristischer Suche (greedy).

(Optional) Ausblick: Gradient-basierte Baum-Ensembles

- Klassische Entscheidungsbäume sind nicht-differenzierbar:
  - Splits sind harte Entscheidungen.
  - Training basiert auf heuristischer Suche (greedy).
- Neuere Ansätze:
  - Relaxieren die harten Entscheidungen zu „weichen“ Splits.
  - Ermöglichen Training mit Gradientenverfahren (Backpropagation).

(Optional) Ausblick: Gradient-basierte Baum-Ensembles

- Klassische Entscheidungsbäume sind nicht-differenzierbar:
  - Splits sind harte Entscheidungen.
  - Training basiert auf heuristischer Suche (greedy).
- Neuere Ansätze:
  - Relaxieren die harten Entscheidungen zu „weichen“ Splits.
  - Ermöglichen Training mit Gradientenverfahren (Backpropagation).
- Beispiel: GRANDE (Gradient-Based Decision Tree Ensembles for Tabular Data)
  - Bäume werden als parametrisiertes, differenzierbares Modell formuliert.
  - Parameter (z. B. Schwellwerte) werden durch Gradientenabstieg optimiert.
  - Ziel: die Stärken von Bäumen (für tabellarische Daten) mit der Optimierbarkeit neuronaler Netze kombinieren.
- Botschaft: Auch bei Entscheidungsbäumen ist die Forschung noch sehr aktiv.

Zusammenfassung

- Entscheidungsbäume:
  - Intuitiv, interpretierbar, universell auf tabellarischen Daten einsetzbar.
  - Lernen über rekursive Splits mit Impurity-Reduktion (z. B. Gini).

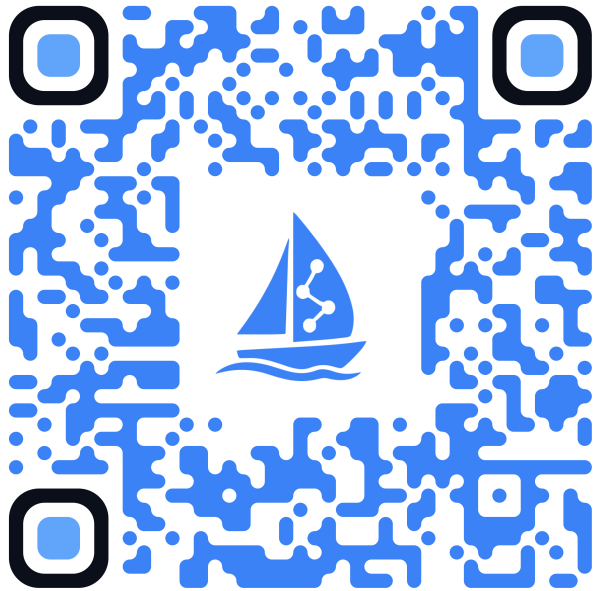
Zusammenfassung

- Entscheidungsbäume:
  - Intuitiv, interpretierbar, universell auf tabellarischen Daten einsetzbar.
  - Lernen über rekursive Splits mit Impurity-Reduktion (z. B. Gini).
- Hauptproblem: Overfitting und hohe Varianz.
- Lösung: Ensembles wie Random Forests und Boosting.
- Für viele praktische tabellarische Probleme:
  - Baum-Ensembles sind nach wie vor sehr starke Baselines.

Zusammenfassung

- Entscheidungsbäume:
  - Intuitiv, interpretierbar, universell auf tabellarischen Daten einsetzbar.
  - Lernen über rekursive Splits mit Impurity-Reduktion (z. B. Gini).
- Hauptproblem: Overfitting und hohe Varianz.
- Lösung: Ensembles wie Random Forests und Boosting.
- Für viele praktische tabellarische Probleme:
  - Baum-Ensembles sind nach wie vor sehr starke Baselines.
- Offenes Feld: gradientenbasierte Tree-Ensembles, bessere Interpretierbarkeit, faire und robuste Modelle.

Begleittext zur Vorlesung



https://jmeischner.com/writing/decision-trees-and-ensemble-methods