

Einführung in die Künstliche Intelligenz

WS 23/24

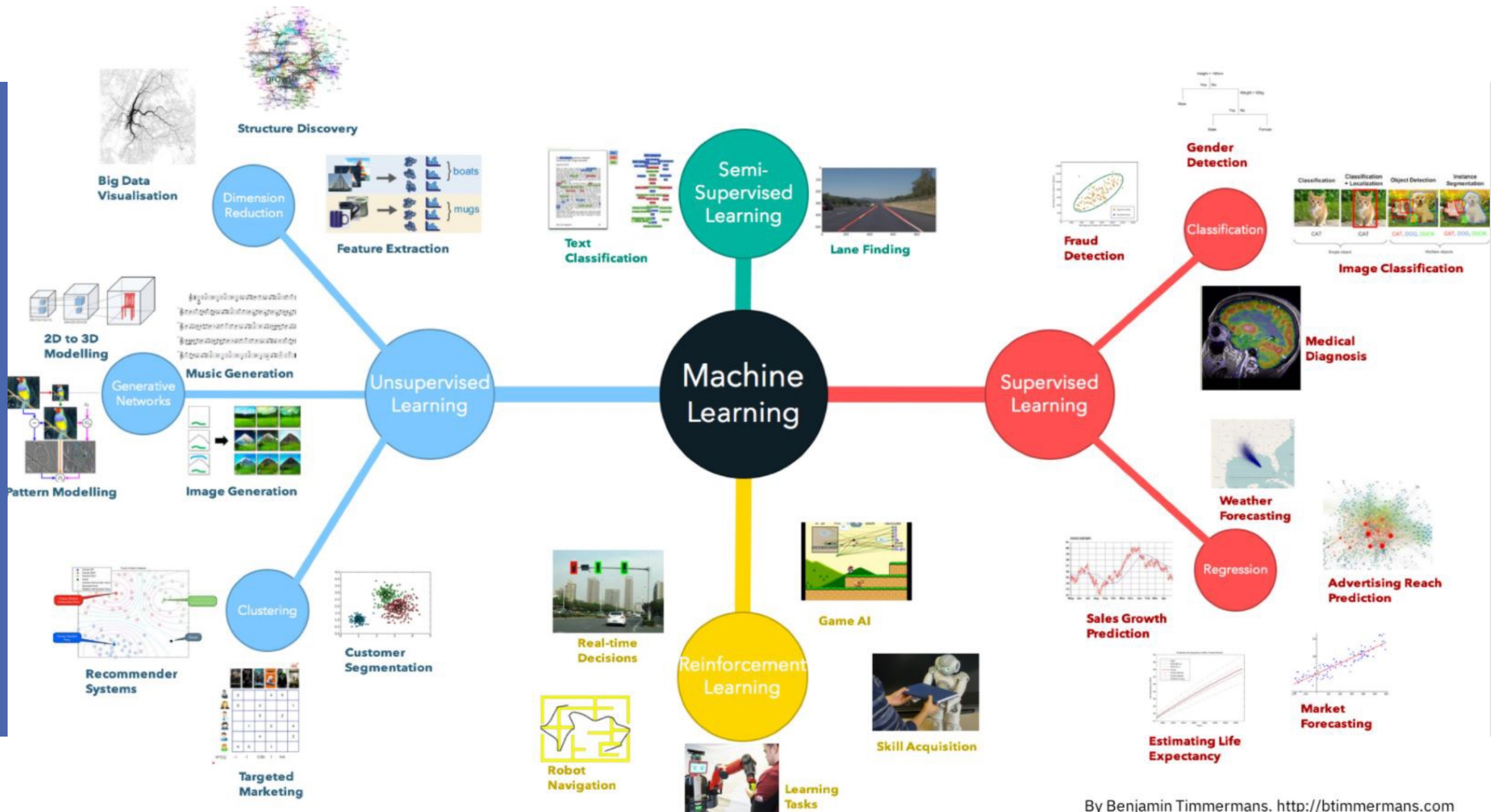
Prof. Dr. Annina
Neumann

Vorlesung 5: Thema und Lernziele

- Überblick Decision Trees
- Praxis-Notebook mit Python scikit-Learn
- Einbettung in den intelligenten Agenten

Lernen

- Bisher haben wir unserem Agenten Informationen bereitgestellt und auch Heuristiken vorgegeben
- Ein wichtiges Merkmal von KI-Systemen ist es jedoch, dass sie in der Lage sind selber aus Daten Muster zu lernen und ihre Entscheidungen damit zu verbessern
- Mit Maschinellern Lernen geben wir dem Agenten die Fähigkeit eigenständig aus Daten zu lernen und sein Handeln somit intelligenter zu machen



Decision Trees

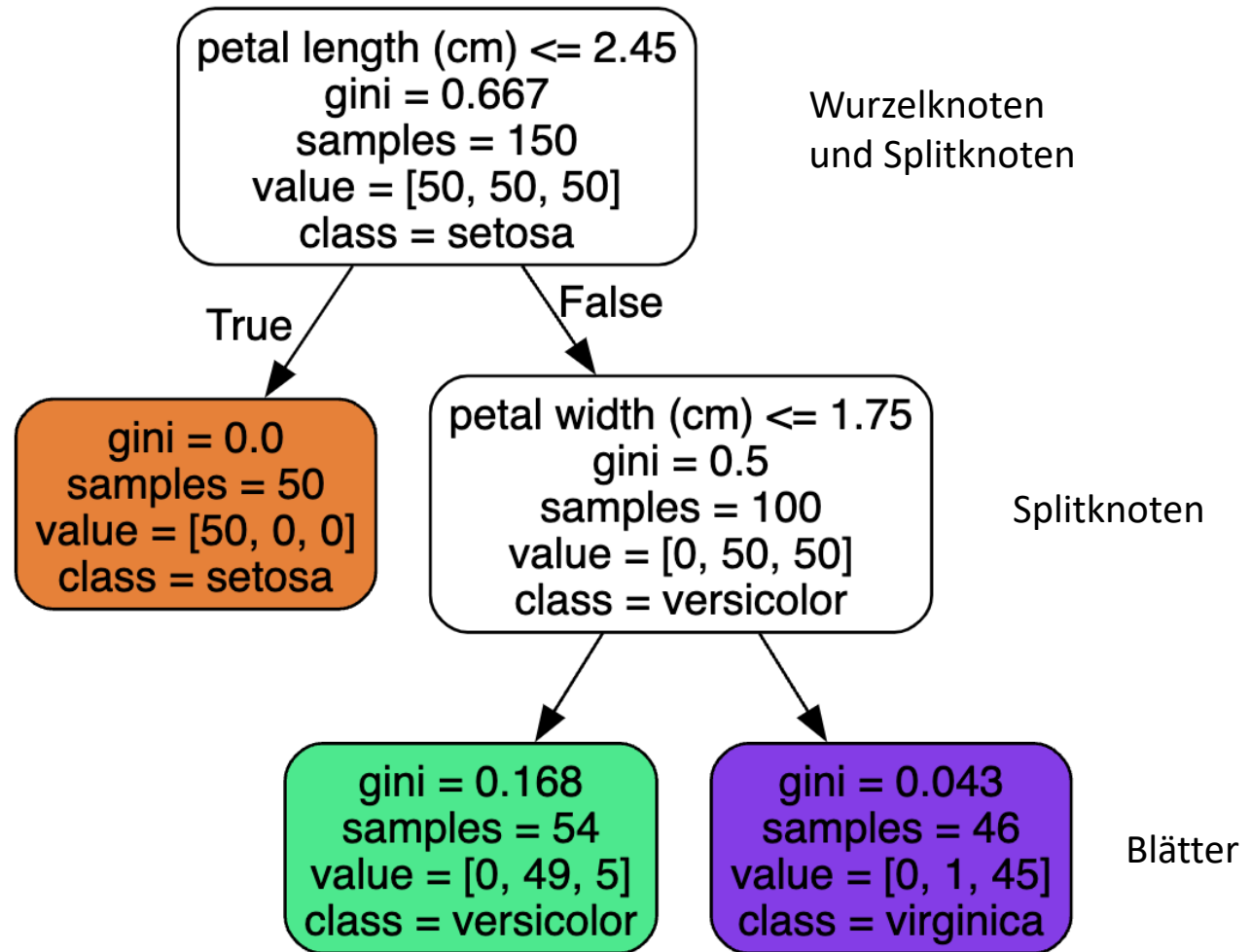
Die heutige Veranstaltung basiert auf Kapitel 6
Entscheidungsbäume aus

- Géron, Aurélien. Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme. O'Reilly.

Decision Trees

- Entscheidungsbäume sind sehr flexible Machine-Learning-Algorithmen, die sich sowohl für Klassifikations- als auch für Regressionsaufgaben eignen
- Die zugrundeliegenden Algorithmen sind sehr mächtig, es lassen sich komplexe Datensätze damit fitten
- Sie sind auch die funktionelle Komponente von Random Forests, die zu den mächtigsten heute verfügbaren Machine-Learning-Algorithmen gehören

Aufbau eines Entscheidungsbaums



Gini-Score

- Das Attribut gini misst die Gini-Unreinheit eines Knotens: Ein Knoten gilt als »rein« (gini=0), wenn sämtliche enthaltenen Datenpunkte der gleichen Kategorie angehören
- Formel zur Berechnung:

Formel 6-1: Gini-Unreinheit

$$G_i = 1 - \sum_{k=1}^n p_{i,k}^2$$

mit

G_i : Gini-Unreinheit für den i . Knoten.

$p_{i,k}$: Anteil von Instanzen der Kategorie k an den Datenpunkten im Knoten i .

- Z.B. Gini-Unreinheit für den linken Knoten bei depth 2:

$$1 - (0/54)^2 - (49/54)^2 - (5/54)^2 \approx 0,168.$$

Entropie

- Alternatives Maß für die Unreinheit
- Der aus der Thermodynamik stammende Begriff Entropie beschreibt Unordnung auf molekularer Ebene: Die Entropie nähert sich null, wenn Moleküle unbeweglich und wohlgeordnet sind.

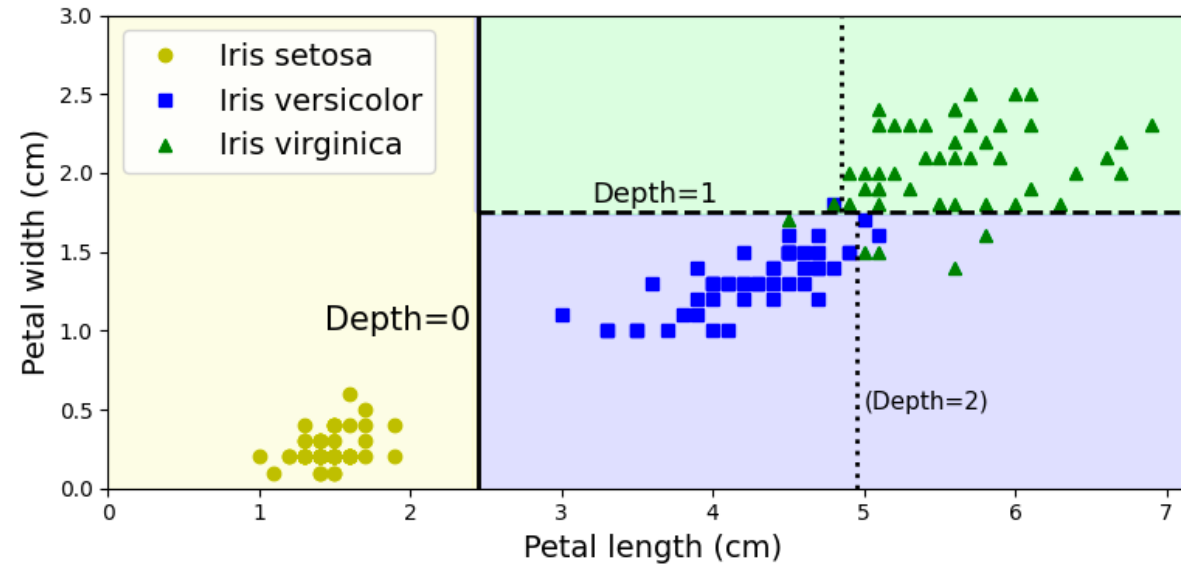
Formel 6-3: Entropie

$$H_i = - \sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^n p_{i,k} \log_2(p_{i,k})$$

- Im Machine Learning wird die Entropie oft als Maß für die Unreinheit eingesetzt: Die Entropie einer Menge ist null, wenn sie nur aus Datenpunkten einer Kategorie besteht.
- Z.B. Entropie für den linken Knoten bei depth 2:

$$-\frac{49}{54} \log_2\left(\frac{49}{54}\right) - \frac{5}{54} \log_2\left(\frac{5}{54}\right) \approx 0,445$$

Ziel: Homogene Blattknoten



- Bei Entscheidungsbäumen wird oft versucht, die Daten so aufzuteilen, dass der Gini-Index minimiert wird, was auf die Reinheit der Blattknoten abzielt. Ein niedriger Gini-Index bedeutet, dass die Blattknoten homogener sind, was die Trennschärfe des Entscheidungsbaums verbessert.

Decision Trees zum Wahrscheinlichkeiten schätzen

- Ein Entscheidungsbaum kann auch die Wahrscheinlichkeit für die Zugehörigkeit eines Datenpunkts zu einer Kategorie k abschätzen
- Zuerst schreitet das Verfahren den Baum ab, um das Blatt für diesen Datenpunkt zu finden, und gibt dann den Anteil der Trainingsdatenpunkte der Kategorie k in diesem Knoten zurück.

Beispiel- algorithmus: CART

Grundidee

- Der Algorithmus teilt die Trainingsdaten zunächst anhand eines Merkmals k und eines Schwellenwerts t_k in zwei Untermengen auf (z.B. »petal length 2,45 cm«).
- Hierbei sucht der Algorithmus nach dem Paar (k, t_k) , das die *reinsten* (nach deren Größe gewichteten) Untermengen hervorbringt.
- Dabei versucht der Algorithmus, die folgende Kostenfunktion zu minimieren:

$$J(k, t_k) = \frac{m_{\text{links}}}{m} G_{\text{links}} + \frac{m_{\text{rechts}}}{m} G_{\text{rechts}}$$

wobei $\begin{cases} G_{\text{links/rechts}} & \text{die Unreinheit der linken/rechten Untermenge misst,} \\ m_{\text{links/rechts}} & \text{die Anzahl Datenpunkte in der linken/rechten Untermenge ist.} \end{cases}$

- Dies setzt sich rekursiv fort, bis die (durch den Hyperparameter `max_depth` angegebene) maximale Tiefe erreicht ist oder keine Aufteilung gefunden werden kann, die die Unreinheit weiter reduziert

CART (Classification and Regression Tree)

- Der CART-Algorithmus ist sehr verbreitet und wird auch von Scikit-Learn verwendet
- Er erzeugt ausschließlich Binärbäume (innere Knoten haben stets zwei Kinder / Fragen lassen sich nur mit Ja oder Nein beantworten)
- Es existieren aber auch andere Algorithmen wie ID3, die Entscheidungsbäume erzeugen, deren Knoten mehr als zwei Kinder haben können

CART (Classification and Regression Tree)

- Gehört zu den Greedy-Algorithmen:
 - Sucht gierig nach einer optimalen Aufteilung auf der höchsten möglichen Ebene und setzt diesen Vorgang auf jeder Stufe fort
 - Es wird nicht geprüft, ob eine Aufteilung einige Schritte weiter zur höchsten möglichen Unreinheit führt
 - Ein Greedy-Algorithmus liefert oft eine recht gute Lösung, diese muss aber nicht die bestmögliche sein.
- Finden des optimalen Baums gehört zu den NP-vollständigen Problemen: Erfordert eine Zeit von $O(\exp(m))$, wodurch das Problem selbst für eher kleine Datensätze praktisch unlösbar wird
=> Wir müssen uns mit einer »annehmbar guten« Lösung zufriedengeben
- Wichtige Hyperparameter, die Abbruchbedingungen steuern: `max_depth`, `min_samples_split`, `min_samples_leaf`, `min_weight_fraction_leaf` und `max_leaf_nodes`.

A* Search

- Die Greedy-Best-First-Suche ist eine Form der Best-First-Suche, die **zuerst den Knoten mit dem niedrigsten $h(n)$ -Wert der Greedy-Best-First-Suche erweitert** – den Knoten, der dem Ziel am nächsten zu sein scheint – mit der Begründung, dass dies wahrscheinlich ist schnell zu einer Lösung führen
- Der gebräuchlichste informierte Suchalgorithmus ist die A*-Suche, eine Best-First-Suche, die die Bewertungsfunktion $f(n) = g(n) + h(n)$ verwendet, mit
 - $g(n)$ die Pfadkosten vom Anfangszustand zum Knoten n
 - $h(n)$ die geschätzten Kosten des kürzesten Pfades von n zu einem Zielzustand

=> $f(n)$ = geschätzte Kosten des besten Pfades, der weitergeht von n zum Ziel

Exkurs: Greedy Algorithmus

- Ein "greedy" Algorithmus ist eine Art von Algorithmus, der in jedem Schritt die beste unmittelbare Entscheidung trifft, um ein Problem zu lösen, ohne Rücksicht auf mögliche langfristige Auswirkungen oder alternative Lösungen.
- Ein "greedy" Algorithmus wählt also die lokal optimale Option in der Hoffnung, dass dies zu einer global optimalen Lösung führt.
- Greedy-Algorithmen sind in der Regel einfach und schnell, können jedoch in einigen Fällen suboptimale Ergebnisse liefern, da sie nicht notwendigerweise den besten langfristigen Pfad verfolgen.
- Die Auswahl der besten Option in jedem Schritt hängt stark von der spezifischen Anwendung und dem Problem ab, das gelöst werden soll.

Overfitting- Risiko

- Entscheidungsbäume treffen sehr wenige Annahmen über die Trainingsdaten (im Gegensatz zu beispielsweise linearen Modellen, die annehmen, dass sich die Daten linear verhalten)
- Solch ein Modell wird auch als parameterfreies Modell bezeichnet, bei welchem sich die Struktur des Baums sehr genau an die Trainingsdaten anpasst
- Das führt höchstwahrscheinlich zu Overfitting der Trainingsdaten
- Somit müssen die Freiheitsgrade eines Entscheidungsbaums beim Trainieren eingeschränkt werden (Regularisierung)

Wichtige Hyperparameter zur Regularisierung von Decision Trees

- `max_features`: Maximale Anzahl der Merkmale, die beim Aufteilen eines Knoten berücksichtigt werden
- `max_leaf_nodes`: Maximale Anzahl Blätter
- `min_samples_split`: Minimale Anzahl an Datenpunkten, die ein Knoten aufweisen muss, damit er aufgeteilt werden kann
- `min_samples_leaf`: Minimale Anzahl an Datenpunkten, die ein Blatt haben muss
- `min_weight_fraction_leaf`: Wie `min_samples_leaf`, aber als Anteil der gesamten gewichteten Datenpunkte.

Ein Erhöhen der `min_*`-Hyperparameter und ein Senken der `max_*`-Hyperparameter regularisiert das Modell

Regularisierung durch Pruning

Alternative Regularisierung

- Andere Algorithmen trainieren Entscheidungsbäume zunächst ohne Einschränkungen, entfernen aber anschließend überflüssige Knoten (Pruning).
- Ein Knoten wird als überflüssig angesehen, wenn seine Kinder ausschließlich Blätter sind und der durch ihn erbrachte Zugewinn an Reinheit nicht statistisch signifikant ist.

Exkurs/ Wiederholung ML-Basics

Wie erkennt man Overfitting?

Regression Trees

- Entscheidungsbäume können auch Regressionsaufgaben bewältigen.
- Der vorhergesagte Wert entspricht dem durchschnittlichen Zielwert der Datenpunkte in diesem Abschnitt.
- Der Algorithmus teilt jeden Abschnitt so auf, dass möglichst viele Trainingsdatenpunkte so nah wie möglich am vorhergesagten Wert liegen.

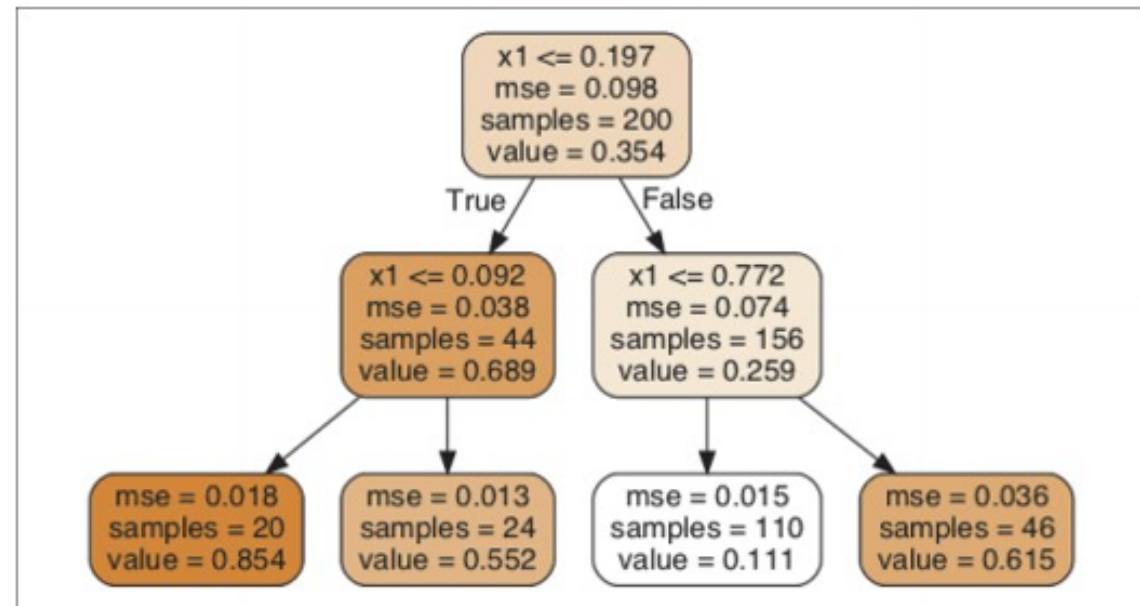


Abbildung 6-4: Ein Entscheidungsbaum zur Regression

Regression Trees mit CART

- Der CART-Algorithmus funktioniert wie weiter oben beschrieben. Der einzige Unterschied besteht darin, dass der Trainingsdatensatz so aufgeteilt wird, dass der MSE statt der Unreinheit minimiert wird.
- Die vom Algorithmus minimierte Kostenfunktion lautet:

Formel 6-4: CART-Kostenfunktion für die Regression

$$J(k, t_k) = \frac{m_{\text{links}}}{m} \text{MSE}_{\text{links}} + \frac{m_{\text{rechts}}}{m} \text{MSE}_{\text{rechts}} \quad \text{mit} \quad \begin{cases} \text{MSE}_{\text{Knoten}} = \frac{\sum_{i \in \text{Knoten}} (\hat{y}_{\text{Knoten}} - y^{(i)})^2}{m_{\text{Knoten}}} \\ \hat{y}_{\text{Knoten}} = \frac{\sum_{i \in \text{Knoten}} y^{(i)}}{m_{\text{Knoten}}} \end{cases}$$

Wie bei
Klassifikations-
aufgaben sind auch
Entscheidungsbäume
für
Regressionsaufgaben
anfällig für Overfitting

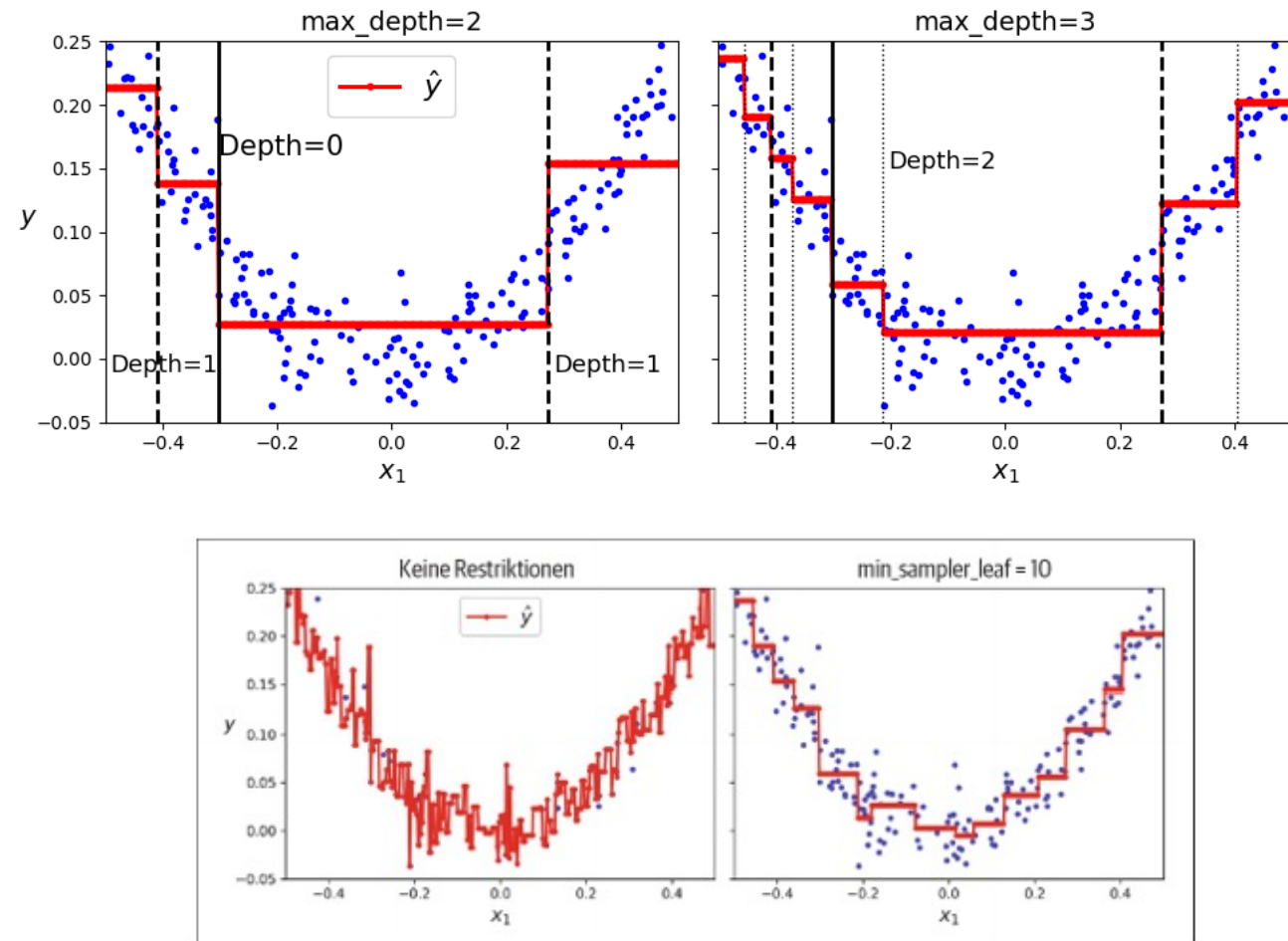
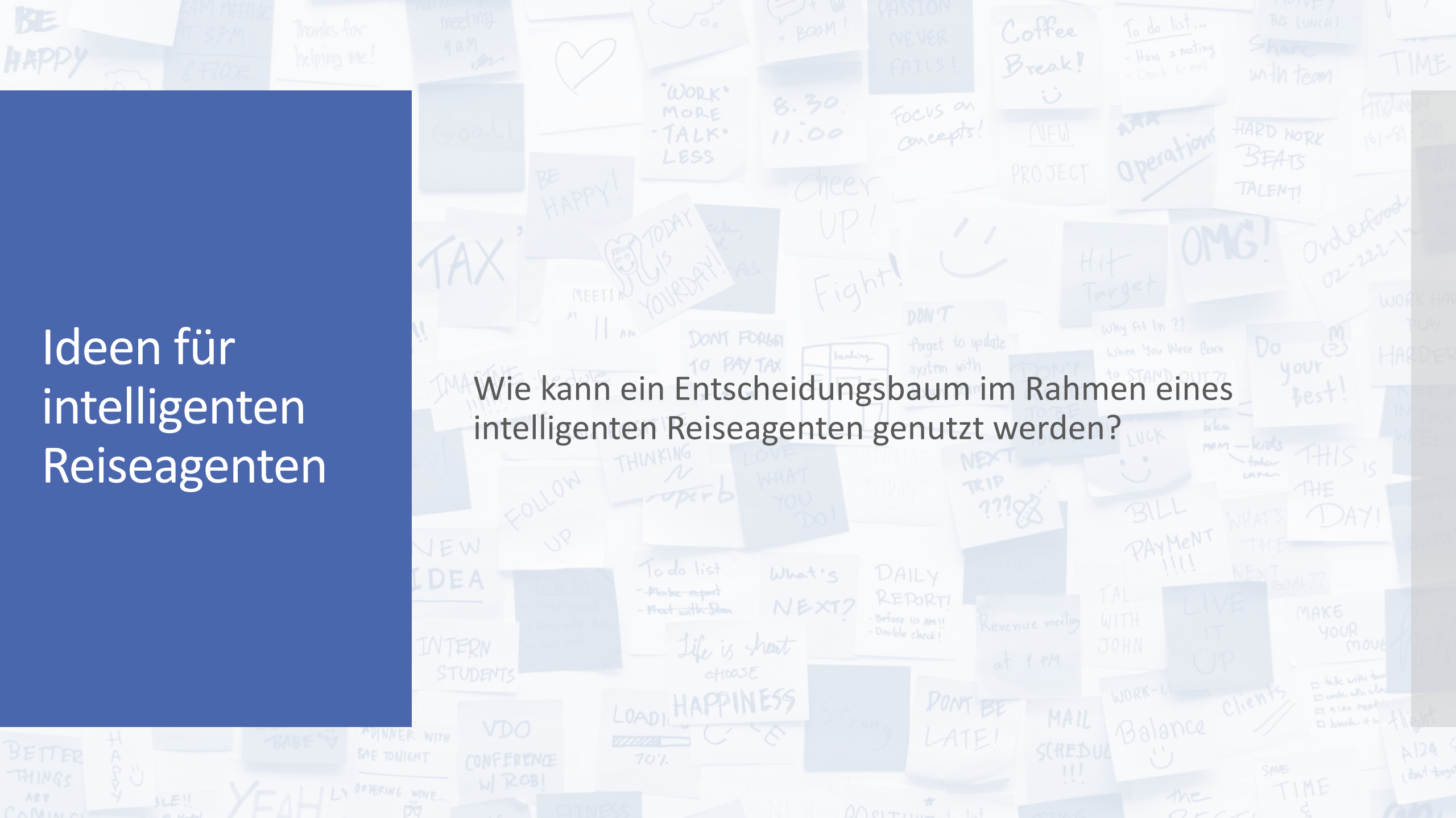


Abbildung 6-6: Vorhersagen eines unregulierten Regressionsbaums (links) und eines regulierten Baums (rechts)

Entscheidungsbäume Vor- und Nachteile

- Einer der vielen Vorzüge von Entscheidungsbäumen ist, dass sie sehr wenig Vorbereitung der Daten erfordern. Insbesondere ist keinerlei Skalierung oder Zentrierung von Merkmalen notwendig.
- Auch sind Entscheidungsbäume sehr transparent und intuitiv erfassbar
- Ein Nachteil ist die hohe Overfitting Gefahr
- Hauptproblem bei Entscheidungsbäumen ist, dass sie eine ziemlich große Varianz besitzen: Kleine Änderungen an den Hyperparametern oder den Daten können zu sehr unterschiedlichen Modellen führen
- Der von Scikit-Learn verwendete Trainingsalgorithmus arbeitet stochastisch (die bei jedem Knoten zu evaluierenden Merkmale werden zufällig ausgewählt). Daher können sogar mit den gleichen Trainingsdaten sehr unterschiedliche Modelle erhalten werden
- Alternative: Mitteln der Vorhersagen über viele Bäume (verrignert die Varianz).

=> **Ensemble** aus Bäumen (z.B. Random Forest) ist eine der leistungsfähigsten Arten von heutzutage verfügbaren Modellen



Ideen für intelligenten Reiseagenten

Wie kann ein Entscheidungsbaum im Rahmen eines intelligenten Reiseagenten genutzt werden?

Aufgabe

Laden Sie das zum Kapitel gehörende Notebook https://github.com/ageron/handson-ml3/blob/main/06_decision_trees.ipynb herunter und arbeiten Sie es durch.
Führen Sie hierzu alle Zellen nacheinander aus und kommentieren im Code, was in der Zelle passiert

Übungen aus Buch (Antworten im Notebook)

1. Was ist die ungefähre Tiefe eines mit 1 Million Datenpunkten (ohne Restriktionen) trainierten Entscheidungsbaums?
2. Ist die Gini-Unreinheit eines Knotens im Allgemeinen geringer oder größer als die seines Elternteils? Ist sie im Allgemeinen kleiner/größer oder immer kleiner/größer?
3. Sollte man versuchen, `max_depth` zu senken, wenn ein Entscheidungsbaum einen Trainingsdatensatz overfittet?
4. Sollte man versuchen, die Eingabemerkmale zu skalieren, wenn ein Entscheidungsbaum die Trainingsdaten underfittet?
5. Wenn es eine Stunde dauert, einen Entscheidungsbaum mit 1 Million Datenpunkten zutrainieren, wie lange etwa wird das Trainieren eines weiteren Baums mit 10 Millionen Datenpunkten dauern? Hinweis: Denken Sie an die Berechnungskomplexität des CART-Algorithmus.
6. Wenn es eine Stunde dauert, einen Entscheidungsbaum mit einem gegebenen Trainingsdatensatz zu trainieren, wie lange dauert es dann ungefähr, wenn sich die Anzahl der Merkmale verdoppelt?
7. Trainieren und optimieren Sie einen Entscheidungsbaum für den Datensatz `moons` anhand dieser Schritte.
 - a. Erzeugen Sie einen `moons`-Datensatz mit `make_moons(n_samples=10000, noise=0.4)`.
 - b. Teilen Sie ihn mit `train_test_split()` in einen Trainings- und einen Testdatensatz auf.
 - c. Verwenden Sie die Gittersuche mit Kreuzvalidierung (mithilfe der Klasse `GridSearchCV`), um gute Einstellungen für die Hyperparameter eines `DecisionTreeClassifier` zu finden. Hinweis: Probieren Sie unterschiedliche Werte für `max_leaf_nodes`.
 - d. Trainieren Sie den Baum mit den vollständigen Trainingsdaten und bestimmen Sie die Qualität Ihres Modells auf den Testdaten. Sie sollten eine Genauigkeit zwischen 85% und 87% erhalten.