

Grundlagen des maschinellen Lernens

Nichtlineare Verfahren

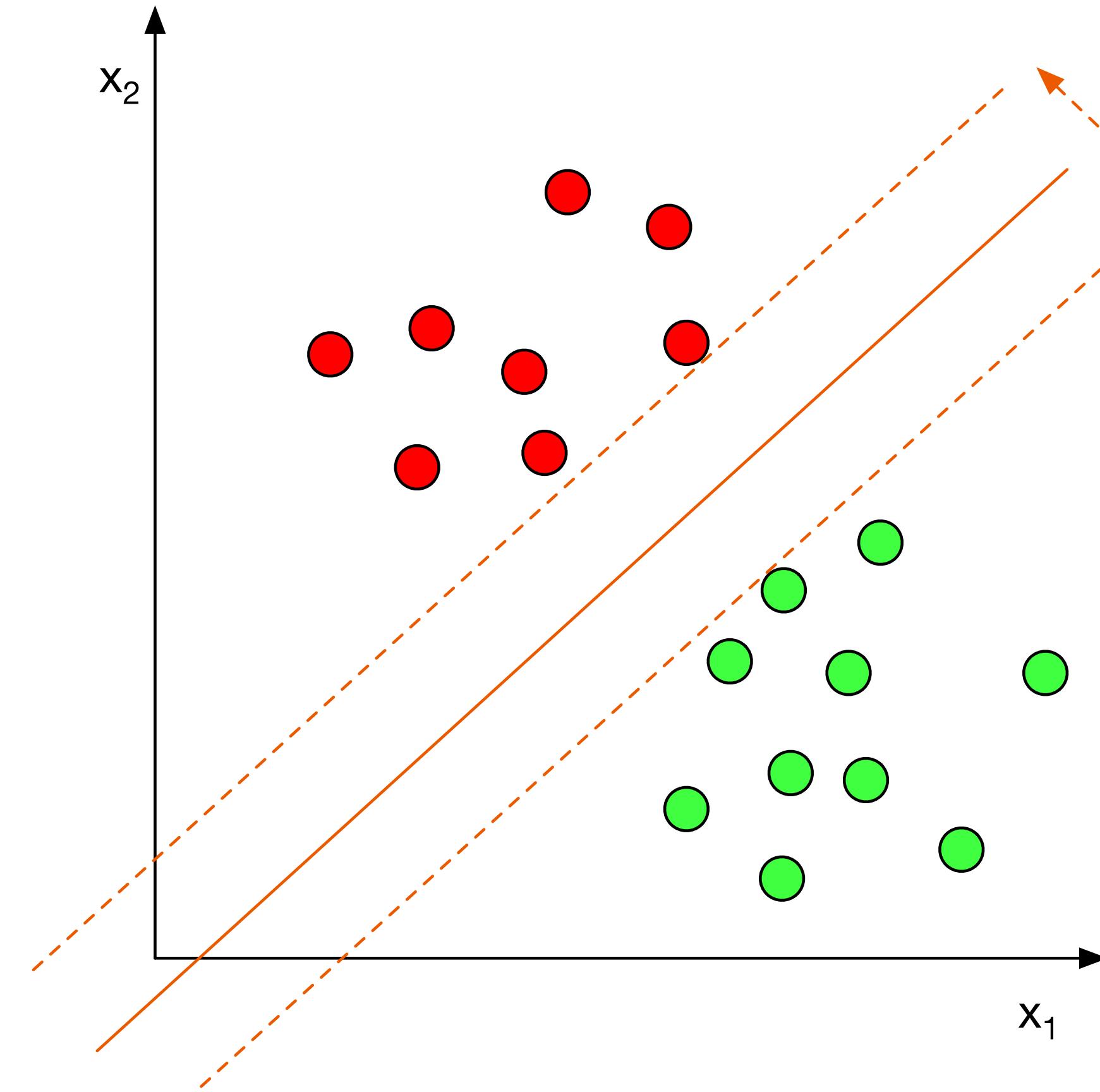
12.05.2020
Nils Schwenzfeier



1. SVM mit Kernel-Trick
2. Entscheidungsbäume
 - 2.1. Einführung
 - 2.2. Regressionsbäume
 - 2.3. Klassifikationsbäume
 - 2.4. Hyperparameter, Overfitting, Underfitting
 - 2.5. Instabilität von Entscheidungsbäumen
 - 2.6. Anhang

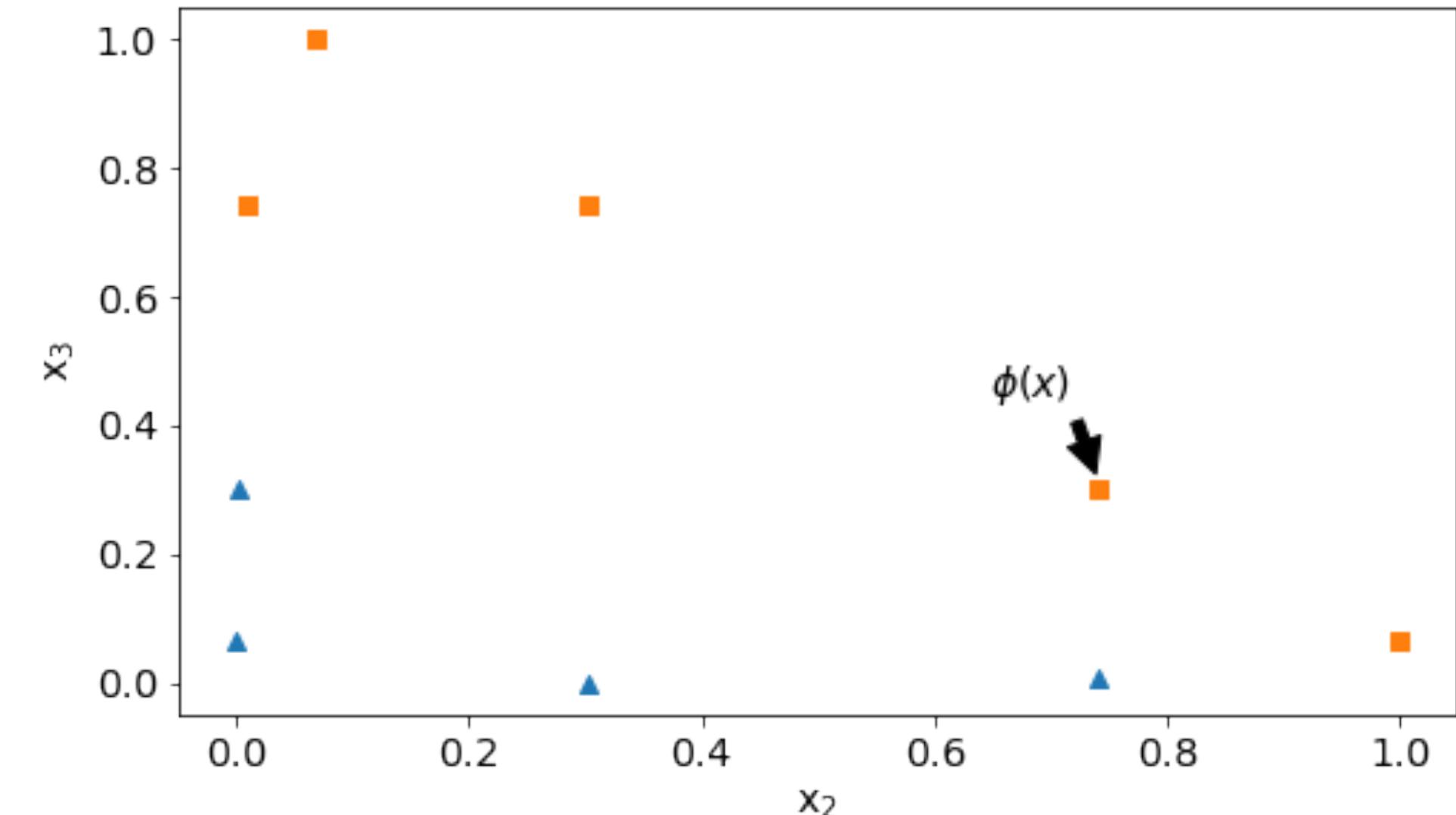
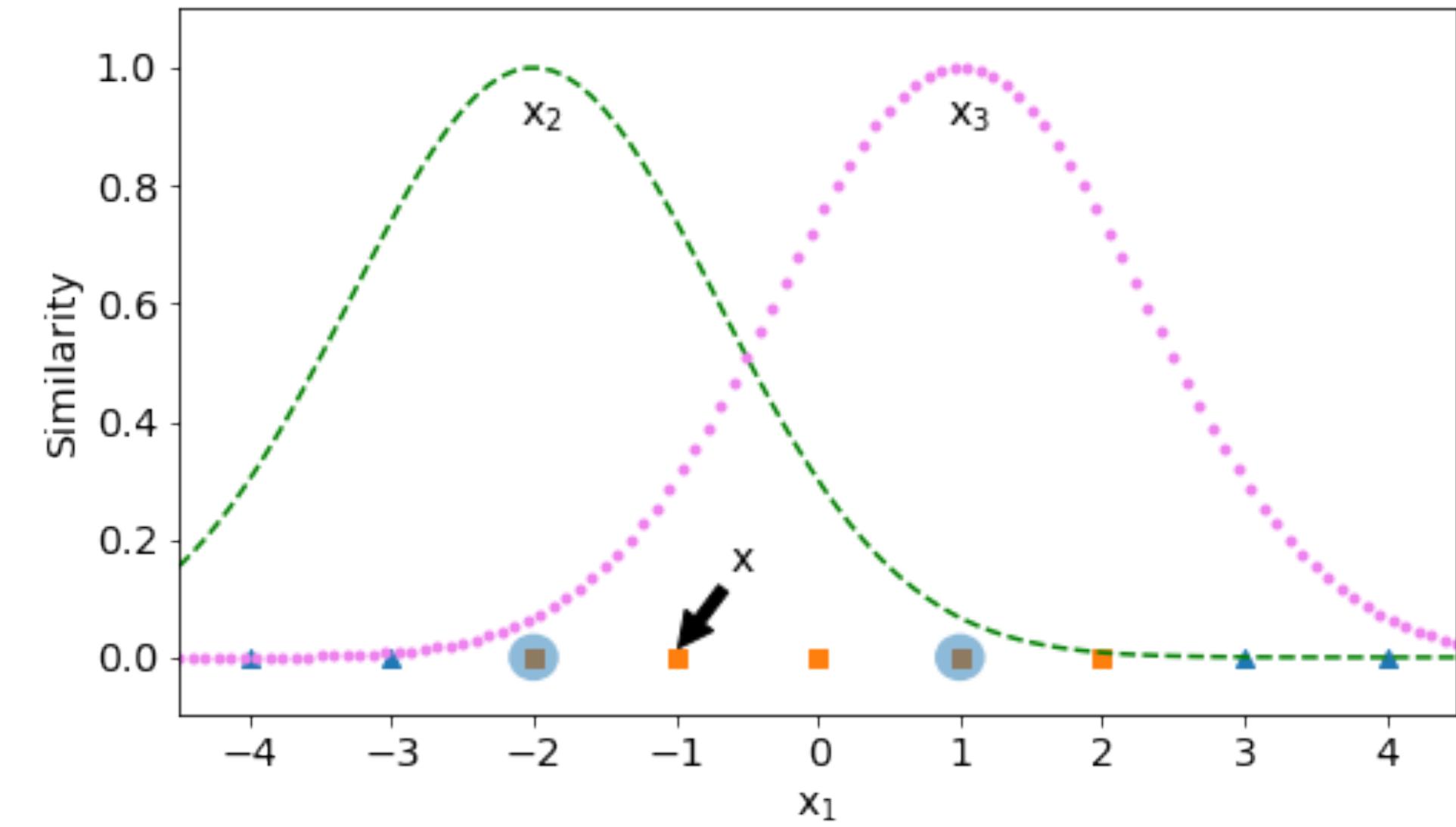
SVM mit Kernel-Trick

- Support Vector Machine zur binären Klassifikation
- Hard-margin vs Soft-margin
- Polynom-Trick zur nichtlinearen Klassifikation



- Erweiterung des Feature-Space
 - Polynom-Trick
 - Ähnlichkeits-Merkmale (Similarity Features)
 - wie ähnlich ist ein Datenpunkt zu einer bestimmten Landmarke (Referenzpunkt)
 - Bsp. einer Ähnlichkeits-Funktion (Similarity function): *Gaussian Radial Basis Function (RBF)*

$$\phi_\gamma(x, l) = \exp(-\gamma \|x - l\|^2)$$

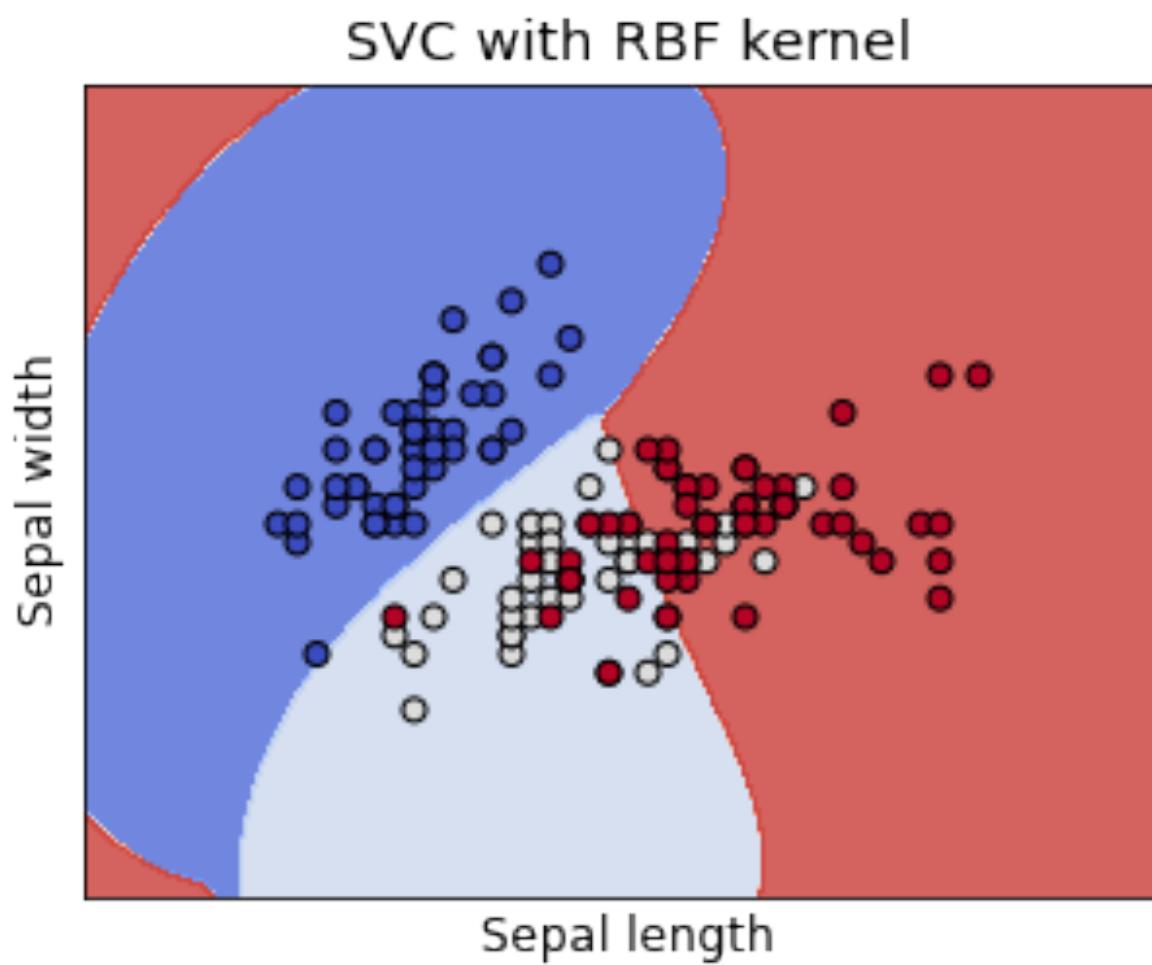
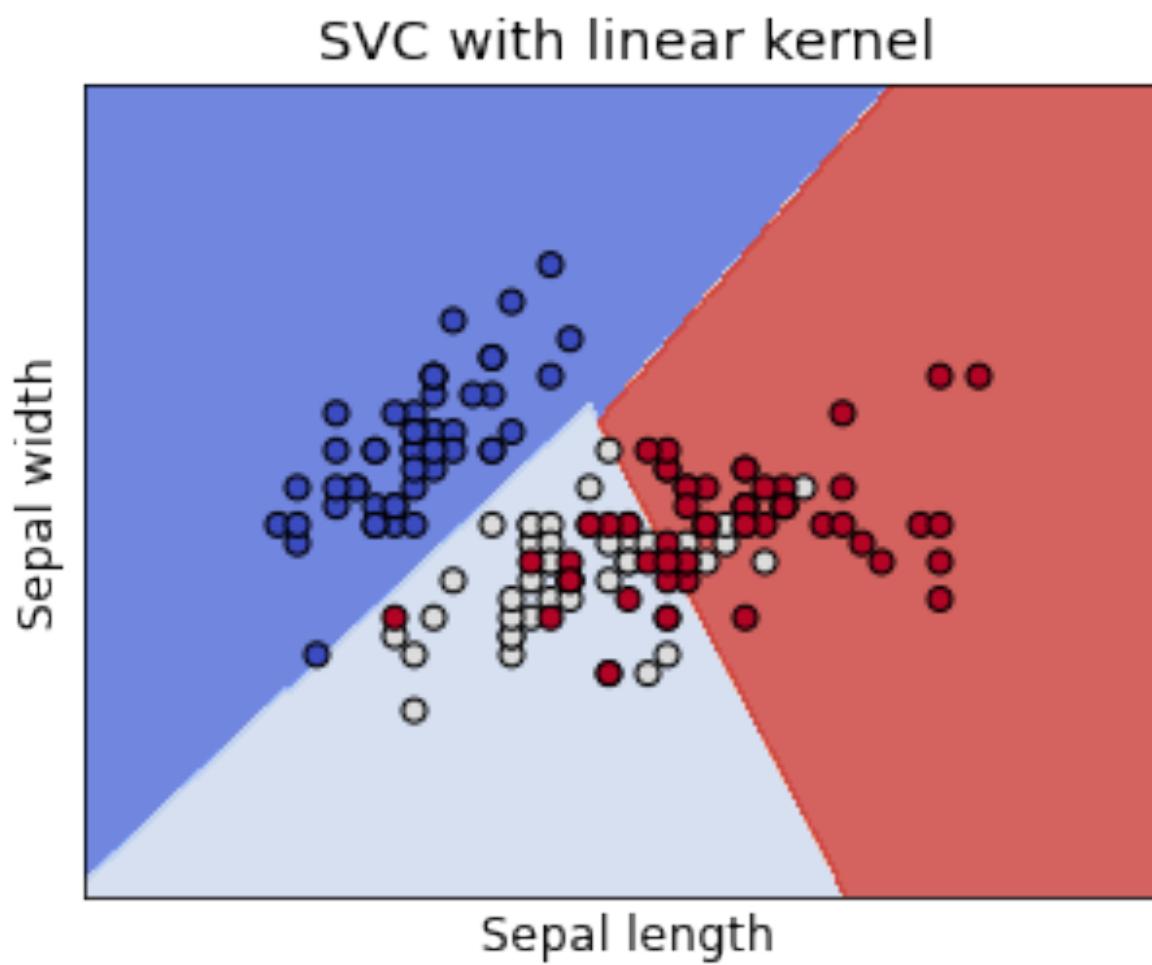


Nichtlineare Klassifikation

► SVM

se
UDE

- Wie bestimmen wir die Positionen der Landmarken?
- Wann wird die Verwendung des Polynom-Tricks/ der Similarity Features problematisch?



Gesucht wird die Lösung für das folgende beschränkte Optimierungsproblem:

$$\underset{\omega, b, \zeta}{\text{minimize}} \quad \frac{1}{2} \omega^T \cdot \omega + C \sum_{i=1}^m \zeta^{(i)}$$

$$\text{subject to} \quad t^{(i)} (\omega^T \cdot x^{(i)} + b) \geq 1 - \zeta^{(i)} \quad \text{and} \quad \zeta^{(i)} \geq 0 \quad \text{for } i=1, 2, \dots, m$$

Für ein bedingtes Optimierungsproblem (primal problem) kann ein verwandtes Problem (dual problem) mit anderen Eigenschaften formuliert werden. Die Lösung des Dual-Problem liefert meist eine untere Schranke für die Lösung des Primal-Problem. Unter bestimmten Bedingungen können jedoch auch beide Lösungen identisch sein.

Die SVM erfüllt diese Bedingungen, sodass entweder das Primal- oder das Dual-Problem gelöst werden kann. Die Dual-Problem-Formulierung der Zielfunktion der linearen SVM lautet:

$$\underset{\alpha}{\text{minimize}} \quad \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha^{(i)} \alpha^{(j)} t^{(i)} t^{(j)} x^{(i)T} \cdot x^{(j)} - \sum_{i=1}^m \alpha^{(i)}$$

subject to $\alpha^{(i)} \geq 0 \quad \text{for } i=1, 2, \dots, m$

Gesucht wird $\hat{\alpha}$, das die Gleichung minimiert. Wurde $\hat{\alpha}$ des Dual-Problems gefunden, können daraus ω und b des Primal-Problems abgeleitet werden.

Die Darstellung dient lediglich zum besseren Verständnis des Kernel-Tricks der SVM. Die Herleitung sprengt den Rahmen dieser Veranstaltung. Interessierte finden weiterführende Informationen in “Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media, 2019.”

Wir möchten den Polynom-Trick mit Grad 2 auf ein zweidimensionales Datenset anwenden. Die zugehörige Abbildung (mapping function) ϕ lautet

$$\phi(x) = \phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}.$$

Der resultierende Vektor ist nun dreidimensional. Wie sieht das Punkt-Produkt (dot product) von zwei Datenpunkten **a** und **b**, auf die wir den Polynom-Trick mit Grad 2 anwenden, aus?

Wie sieht das Punkt-Produkt (dot product) von zwei Datenpunkten \mathbf{a} und \mathbf{b} , auf die wir den Polynom-Trick mit Grad 2 anwenden, aus?

$$\begin{aligned}\phi(\mathbf{a})^T \cdot \phi(\mathbf{b}) &= \begin{pmatrix} a_1^2 \\ \sqrt{2}a_1a_2 \\ a_2^2 \end{pmatrix}^T \cdot \begin{pmatrix} b_1^2 \\ \sqrt{2}b_1b_2 \\ b_2^2 \end{pmatrix} = a_1^2b_1^2 + 2a_1b_1a_2b_2 + a_2^2b_2^2 \\ &= (a_1b_1 + a_2b_2)^2 = \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^T \cdot \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)^2 = (\mathbf{a}^T \cdot \mathbf{b})^2\end{aligned}$$

Das Punkt-Produkt der transformierten Vektoren entspricht dem Quadrat des Punkt-Produkts der originalen Vektoren!

$$\phi(\mathbf{a})^T \cdot \phi(\mathbf{b}) = (\mathbf{a}^T \cdot \mathbf{b})^2$$

Dual-Problem-Formulierung der Zielfunktion einer linearen SVM:

$$\underset{\alpha}{\text{minimize}} \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha^{(i)} \alpha^{(j)} t^{(i)} t^{(j)} x^{(i)^T} \cdot x^{(j)} - \sum_{i=1}^m \alpha^{(i)}$$

subject to $\alpha^{(i)} \geq 0$ for i=1, 2, ..., m

Transformation aller Datenpunkte mit der Funktion ϕ , wobei ϕ die Polynom-Transformation zweiten Grades darstellt:

$$x^{(i)^T} \cdot x^{(j)} \rightarrow \phi(x^{(i)})^T \cdot \phi(x^{(j)}) = (x^{(i)^T} \cdot x^{(j)})^2$$

Ein Kernel K ist eine Funktion, die das Punkt-Produkt $\phi(a)^T \cdot \phi(b)$ zwischen a und b berechnet ohne die Funktion ϕ berechnen zu müssen.

Dies ermöglicht die effiziente Berechnung für die nichtlineare Trennung mittels SVM, da die Transformation des Feature Space nicht explizit berechnet wird.

Häufig verwendete Kernel sind:

- Linear:
$$K(a, b) = a^T \cdot b$$

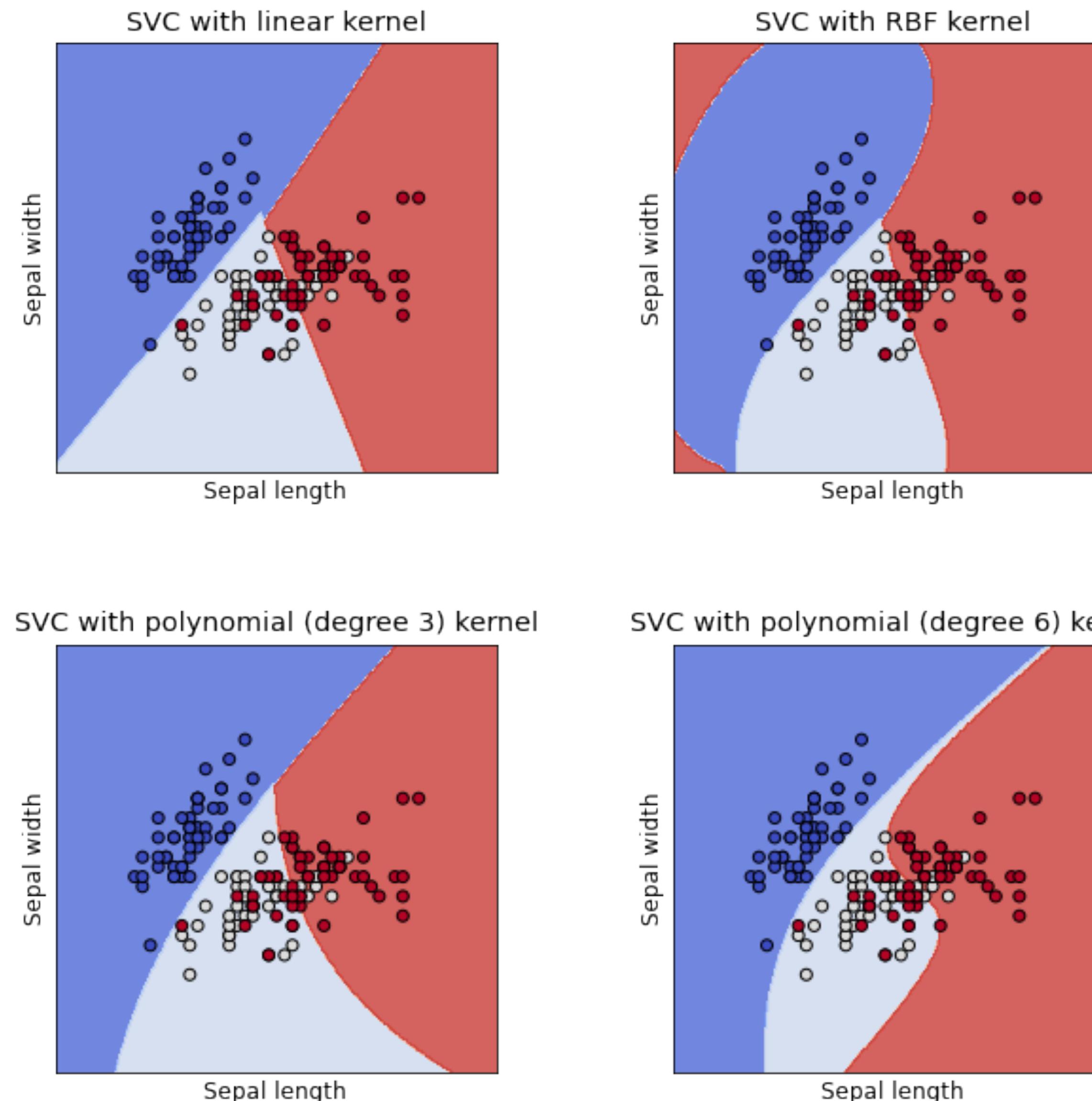
- Polynomial:
$$K(a, b) = (\gamma a^T \cdot b + r)^d$$

- Gaussian RBF:
$$K(a, b) = \exp(-\gamma \|a - b\|^2)$$

```
from sklearn.svm import SVC
```

```
svm_linear      = SVC(kernel='linear', C=1.0)
svm_polyomial  = SVC(kernel='poly', degree=2, C=1.0)
svm_rbf         = SVC(kernel='rbf', gamma=0.3, C=1.0)
```

Kernelized SVM Demo



Decision Trees

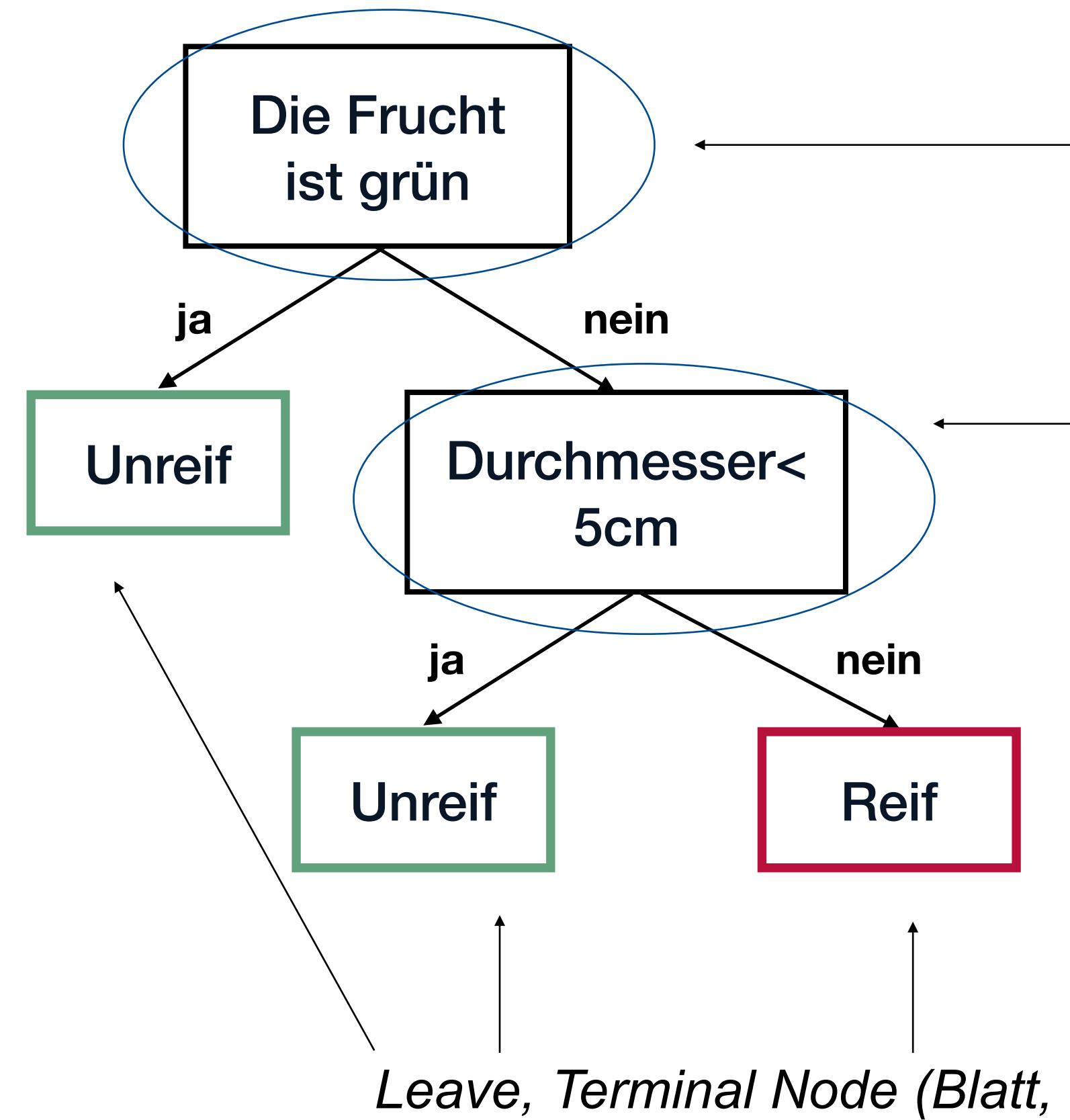
Einführung

- *Decision Trees (Entscheidungsbäume)*
 - sind Modelle des überwachten Lernens, die sowohl für Regression als auch Klassifikation eingesetzt werden können,
 - sind Modelle zur nichtlineare Regression und nichtlinearer Klassifikation,
 - können bei beliebig vielen Klassen und bei mehreren Zielvariablen (*Multi-Output*) eingesetzt werden,
 - sind oft einfach nachzuvollziehen und zu interpretieren (gehören zu *White-Box-Modellen*).
- Bäume, die zur Regression eingesetzt werden, werden als *Regression Trees (Regressionsbäume)*, und Bäume, die zur Klassifikation eingesetzt werden, als *Classification Trees (Klassifikationsbäume)* bezeichnet.

Decision Tree: Zwei simple Beispiele

- ▶ Decision Trees
- ▶ Einführung

Beispiel eines Klassifikationsbaums: Reife einer Frucht anhand von Farbe und Durchmesser vorhersagen

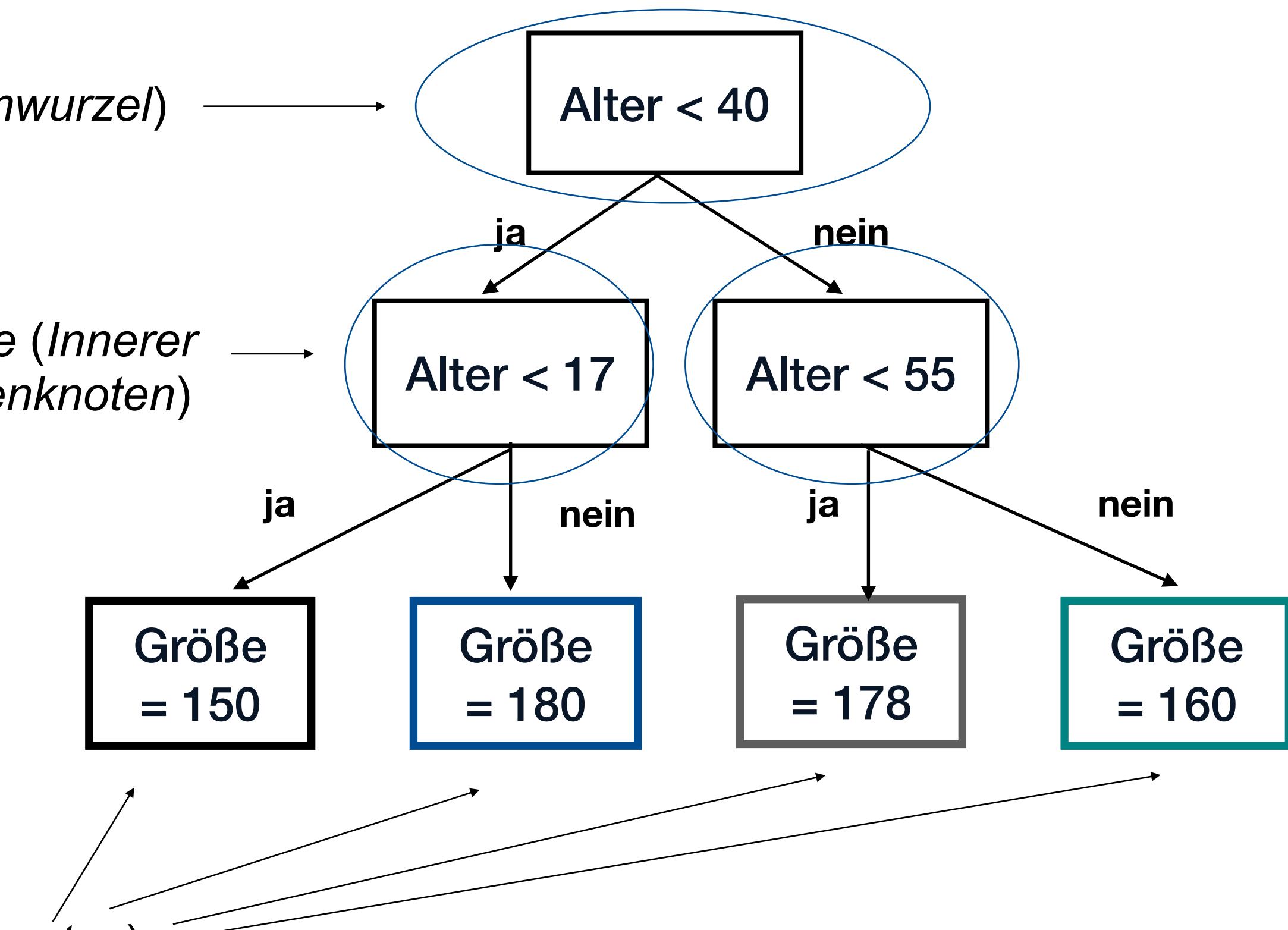


Root (Baumwurzel)

Internal Node (Innerer Knoten, Innenknoten)

Leave, Terminal Node (Blatt, Blattknoten, Endknoten)

Beispiel eines Regressionsbaums: Größe einer Person anhand von Alter vorhersagen



Lernen eines Entscheidungsbaums

- Decision Trees
- Einführung

- Während des Trainings wird ein Baum aus Fragen/Entscheidungen aufgebaut, wobei sich jede Frage auf ein Feature bezieht.
- Während des Trainings werden die Fragen (keine Features), die eine optimale Regression/Klassifikation erlauben, gelernt.
- Es werden binäre (ja/nein, d.h. jeder Innenknoten hat genau zwei Kinder) Fragen über die vorhandenen Features gelernt.
Hinweis: Manche Algorithmen erzeugen Bäume, deren Innenknoten mehr als zwei Kinder haben, diese sind kein Bestandteil der Vorlesung.
- *Depth (Tiefe/Baumtiefe)* entspricht der maximalen Anzahl an Fragen, die das Modell hintereinander stellen kann.
Hinweis: In der Literatur startet die Baumtiefe oft bei 0, Tiefe=0 entspricht Bäumen, die aus einer Baumwurzel, deren Kinder Blätter sind, bestehen. Hier gilt Tiefe=Anzahl der Fragen. D.h. für einen Baum bestehend aus einer Baumwurzel, deren Kinder Blätter sind, gilt Tiefe=1.
- Der Entscheidungsbaum splittet den Feature Space in disjunkte Regionen in Form von Hyperquadern, sodass jedes Sample in genau einem Hyperquader liegt. → Für jedes Sample sagt der Baum voraus, in welchem Hyperquader es liegt, bei Regression wird zudem ein numerischer Wert vorhergesagt.
- Ein Hyperquader ist
 - ein Intervall bei eindimensionalen Daten,
 - ein Rechteck bei zweidimensionalen Daten,
 - ein Quader bei dreidimensionalen Daten,
 - etwas nicht mehr sinnvoll darstellbares ab vierdimensionalen Daten.

- Die durch die Fragen aufgespannte Hyperplane (siehe F.34 der Vorlesung über lineare Verfahren) zwischen Hyperquadern heißt *Decision Boundary*.
- Was hat das mit Machine Learning zu tun? Wir wollen die optimalen Fragen und damit die optimale Aufteilung des Feature Space lernen!
- L sei die Anzahl der Hyperquader, R_1, \dots, R_L bezeichne die Hyperquader.
- $x^{(1)}, \dots, x^{(m)}$ bezeichne die Samples, y_1, \dots, y_m die Labels, $(x^{(1)}, y_1), \dots, (x^{(m)}, y_m)$ seien Datenpunkte, x_1, \dots, x_n bezeichne die Features.

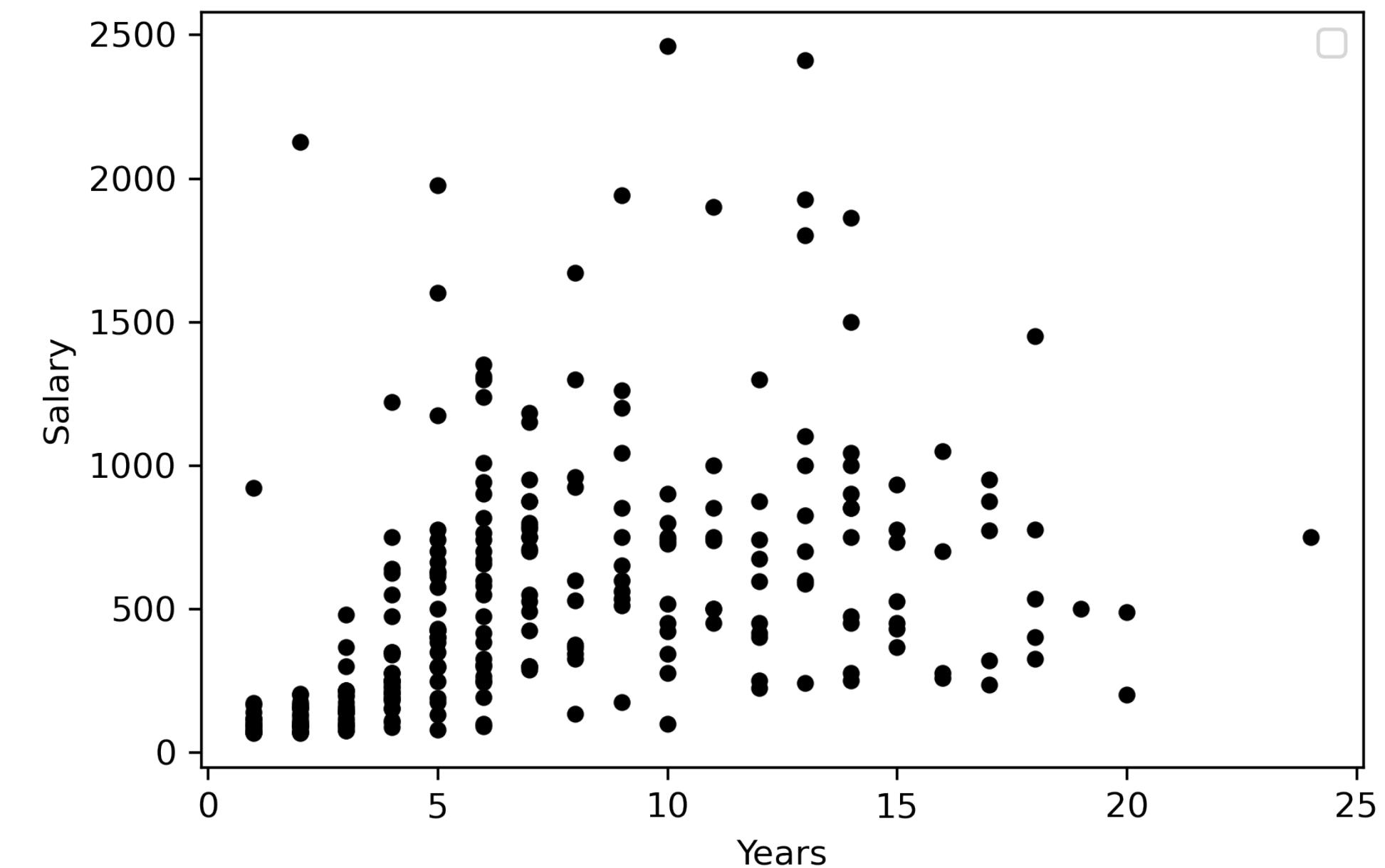
Decision Trees

Regressionsbäume

Beispiel: Hitters

- ▶ Decision Trees
- ▶ Regressionsbäume

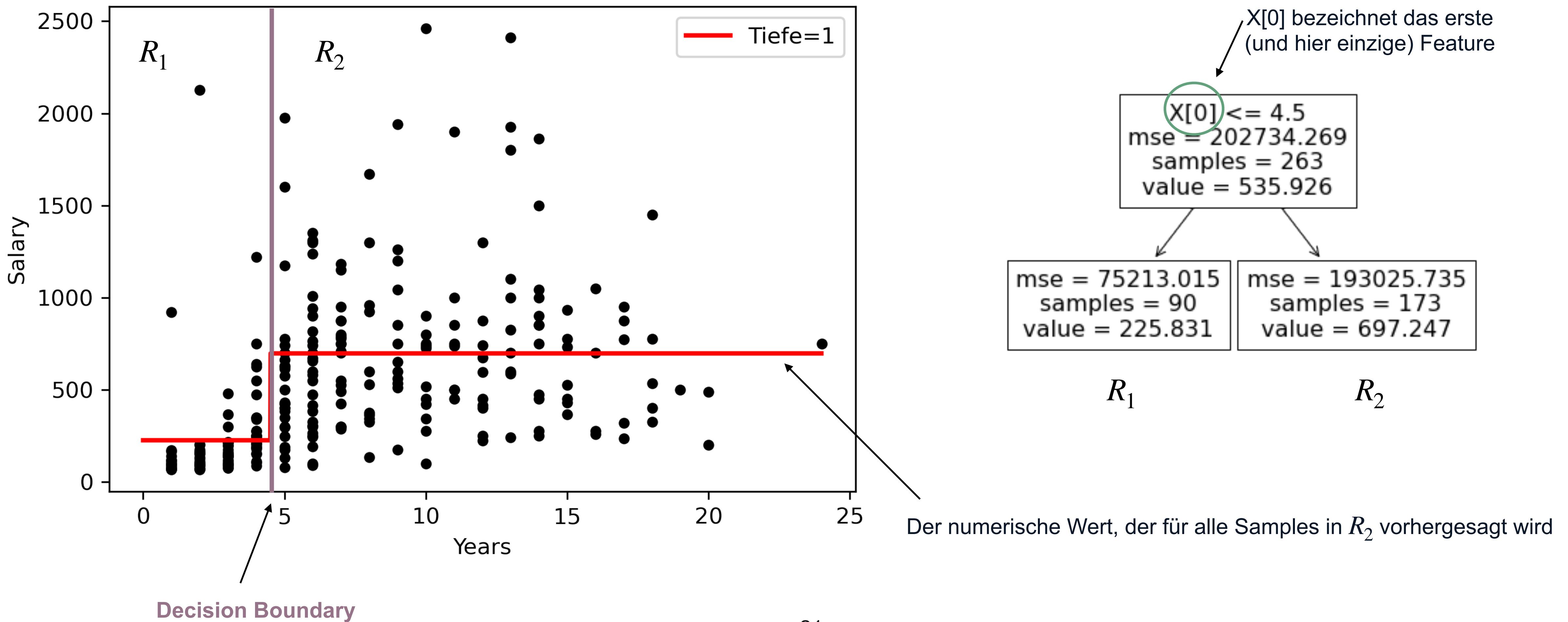
- Beispiel Hitters (Major League Baseball Data from the 1986 and 1987 seasons), den Datensatz findet man bei [kaggle.com](https://www.kaggle.com): Wir wollen anhand der Anzahl der Jahre, die ein Spieler gespielt hat, sein Gehalt voraussagen.
→ Anzahl der Jahre ist das Feature, Gehalt ist das Label.
- Es ist klar, dass eine Gerade (bspw. resultierend aus der linearen Regression) die Datenpunkte sehr schlecht abbilden würde.
- Wären die Datenpunkte so verteilt, dass sich diese durch ein Polynom vom Grad größer oder gleich 2 abbilden ließen, so ließe sich der Polynomtrick anwenden, um daraus ein lineares Problem zu machen.
- Die Datenpunkte sind hier aber eher „wild“ im Feature Space verteilt und wir können das Problem daher mit den bisher kennengelernten Verfahren und Tricks nicht lösen.
- Bis jetzt haben wir nur solche Daten betrachtet, in denen jedes Sample genau ein Label hat: Für alle bisherigen Beispiele gab es keine Datenpunkte mit $(x^{(i)}, y_i), (x^{(i)}, y_j), j \neq i$. Die links abgebildeten Daten zeigen, dass es für jedes (bis auf das ganz rechts) Sample mindestens zwei Labels gibt: Zu jeder Anzahl an Jahren gibt es mindestens zwei verschiedene Gehälter.



Beispiel: Hitters

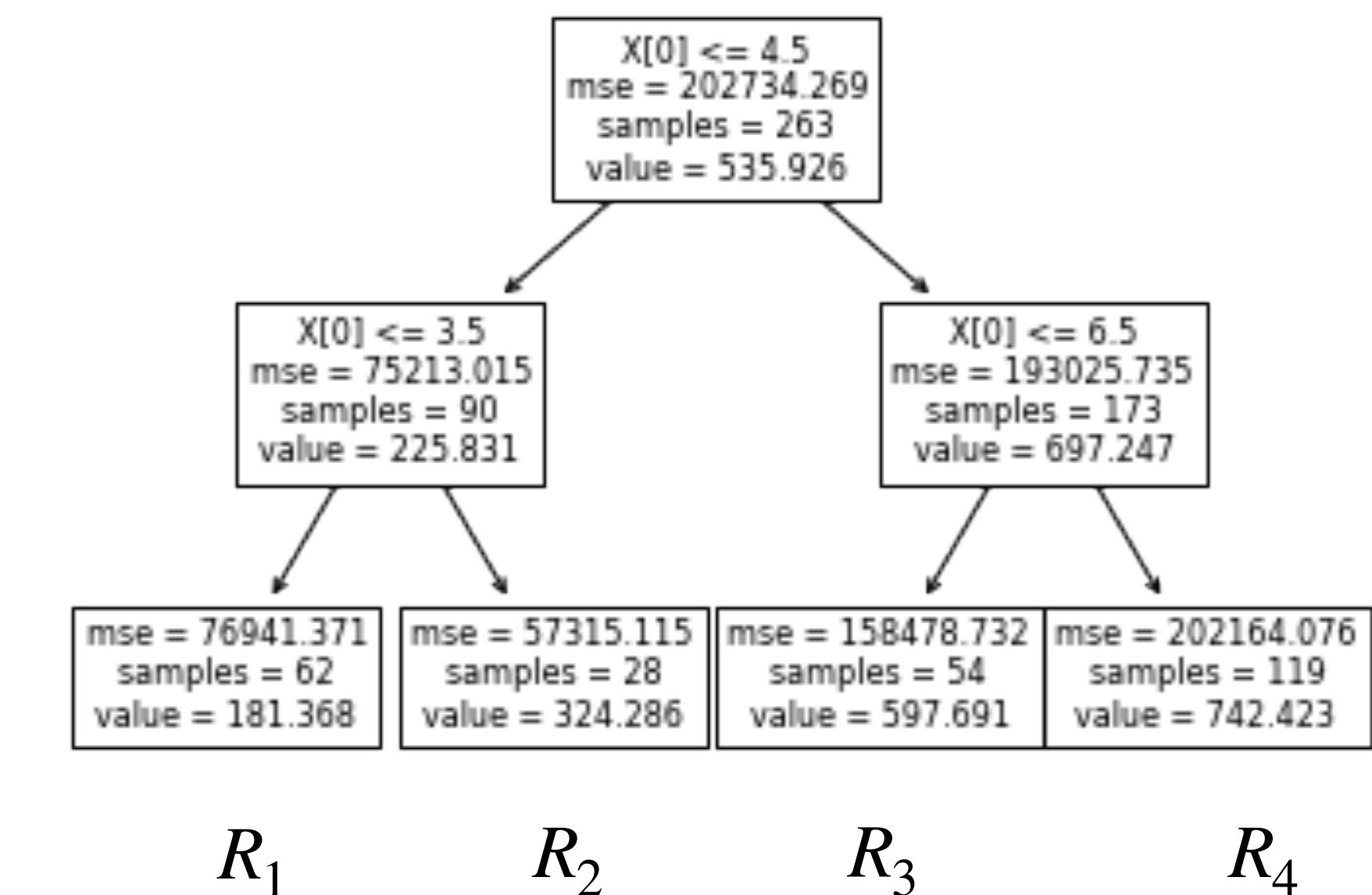
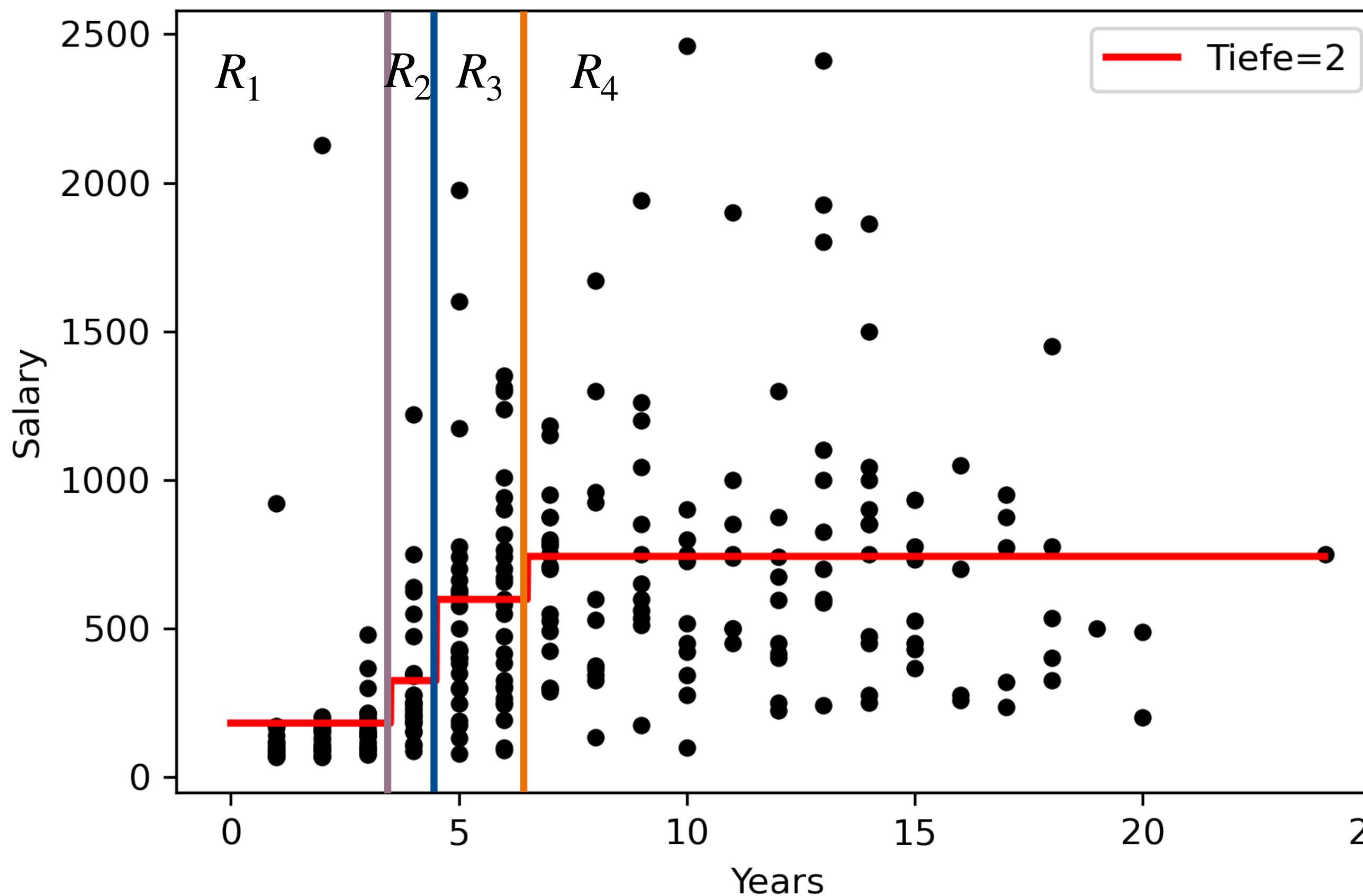
- ▶ Decision Trees
- ▶ Regressionsbäume

- Für alle Regressionsbäume gilt allgemein: Der numerische Wert, der in jedem Hyperquader vorhergesagt wird, ist der Durchschnitt der Labels über alle Samples in R_l , Erklärung folgt später.



Beispiel: Hitters

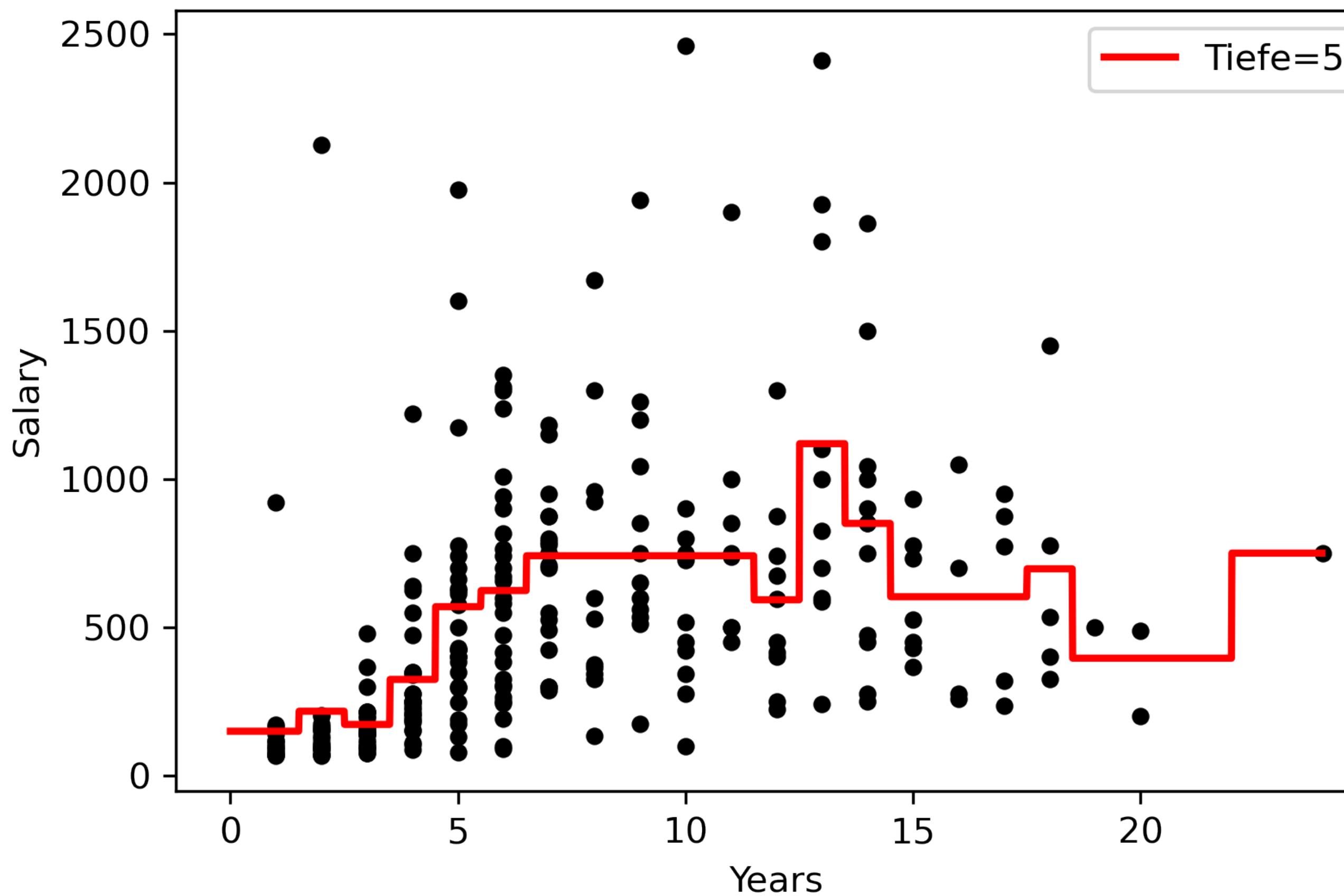
- ▶ Decision Trees
- ▶ Regressionsbäume



Beispiel: Hitters

- Decision Trees
- Regressionsbäume

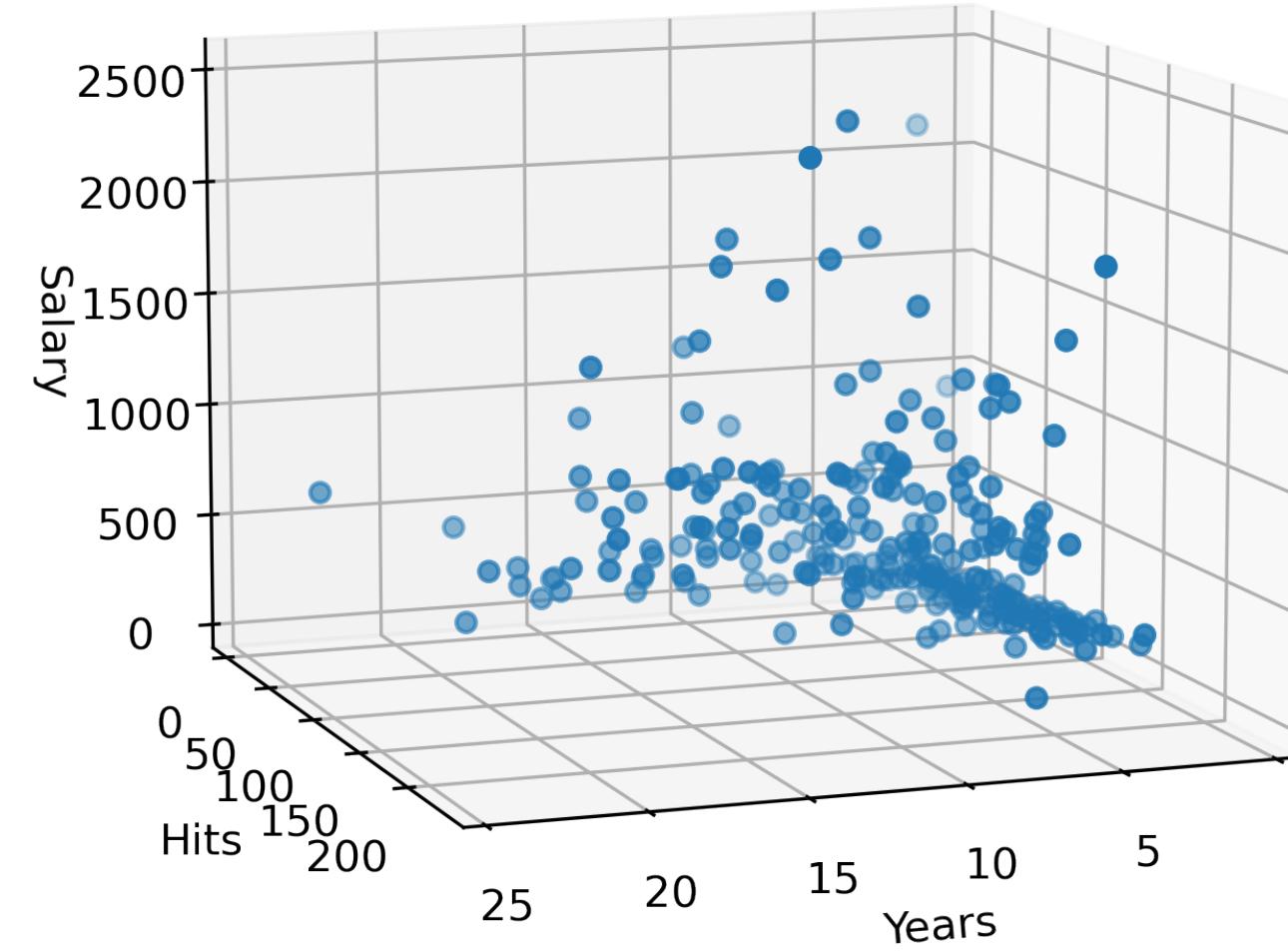
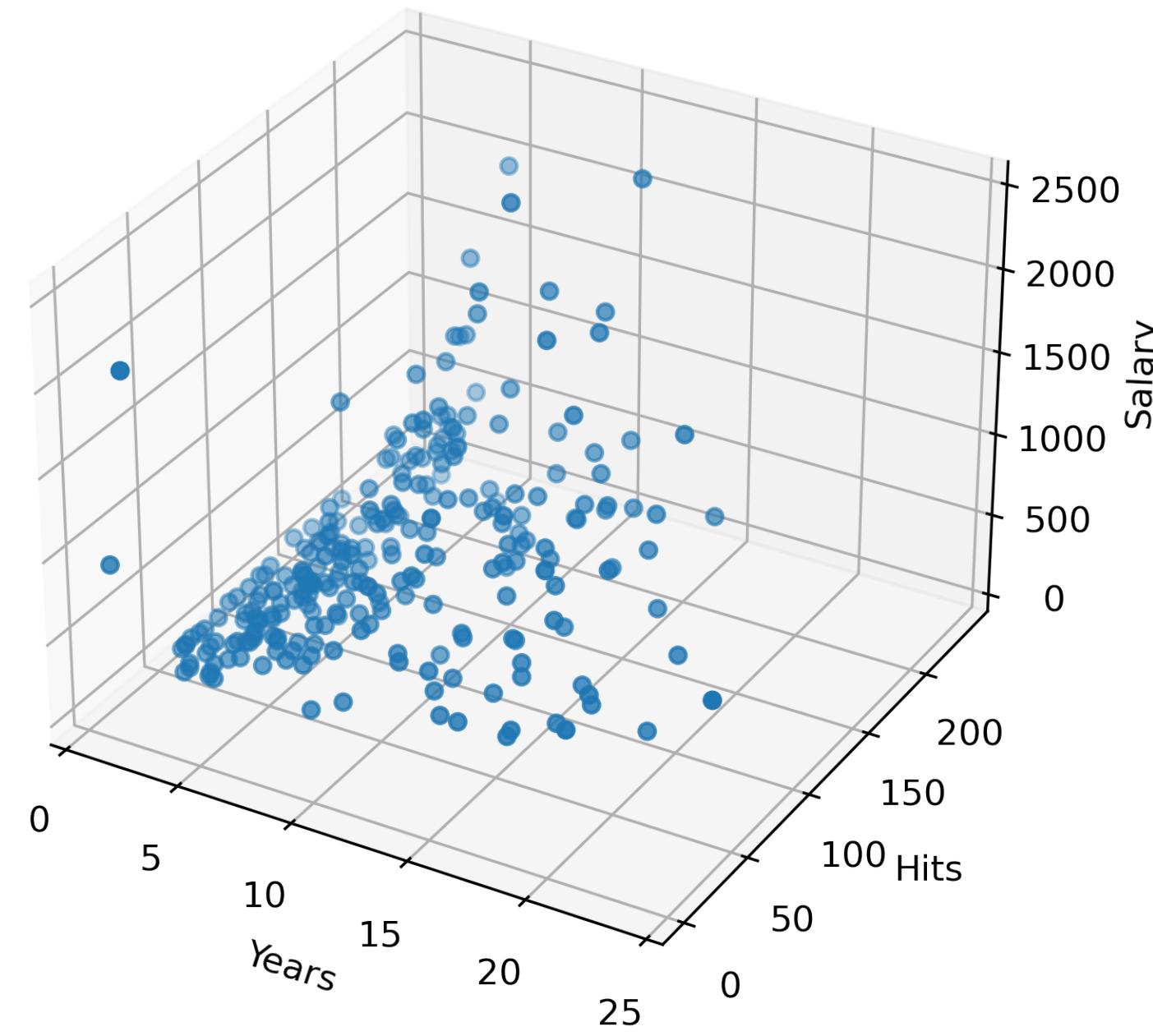
- Bereits bei einer Baumtiefe von 5 entsteht ein komplizierter Baum, der sich zu stark an die Daten, insbesondere an Outlier, anpasst (die Datenpunkte mit Salary > 1500 könnten als Outlier betrachtet werden) und den Feature Space in 14 Rechtecke (zähle Blätter im Baum links) splittet.



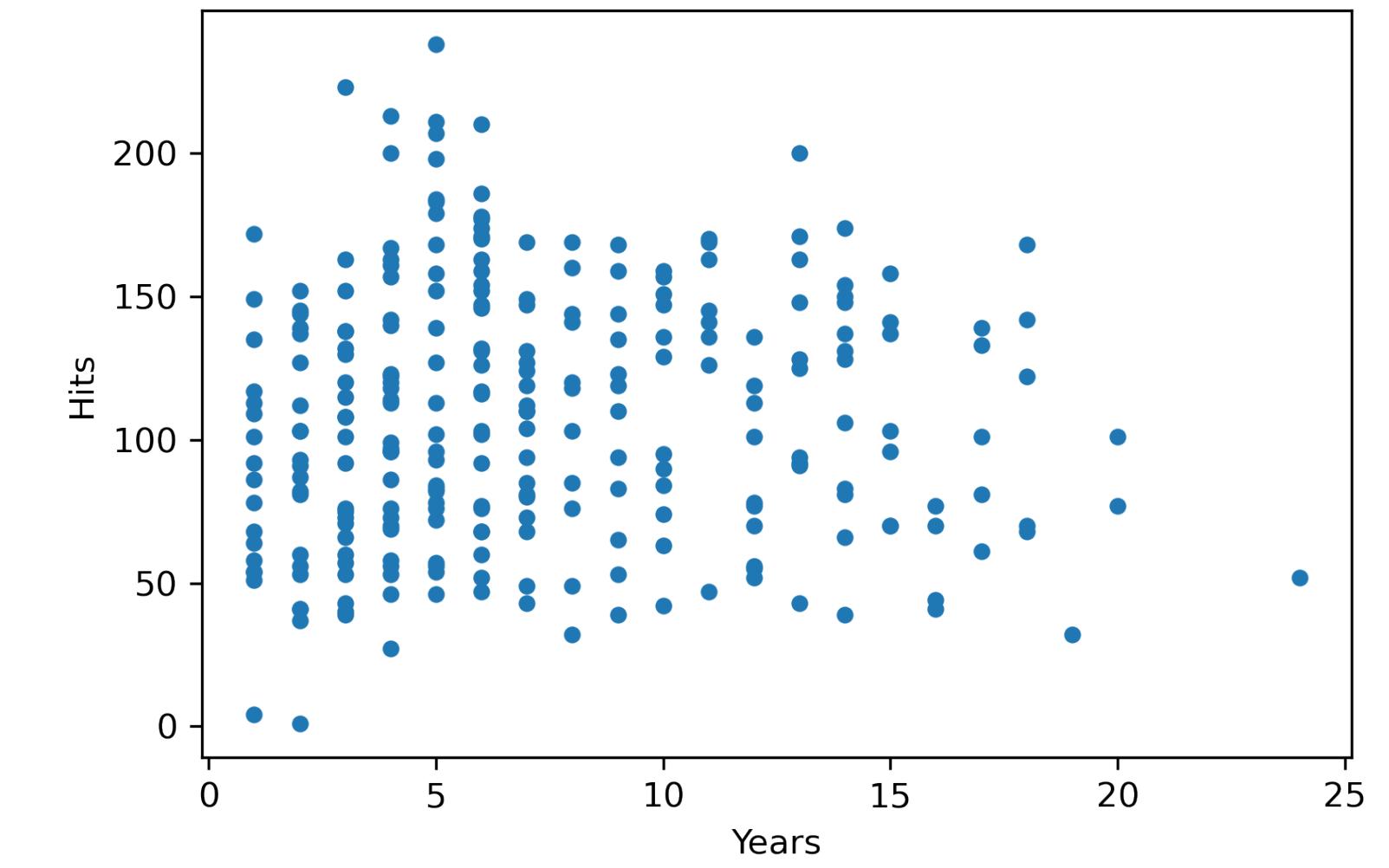
Beispiel: Hitters in 3D

- ▶ Decision Trees
- ▶ Regressionsbäume

- Wir wollen anhand der Anzahl der Jahre, die ein Spieler gespielt hat, und zusätzlich der Anzahl der erzielten Körbe sein Gehalt voraussagen.
→ Anzahl der Jahre und Anzahl der erzielten Körbe sind nun die Features, Gehalt ist das Label.
- Mit einem weiteren Feature sind die Datenpunkte weiterhin so verteilt (was anhand verschiedener Perspektiven ersichtlich ist), dass uns die bisherigen Verfahren und Tricks auch hier bei der Lösung des Problems nicht helfen.

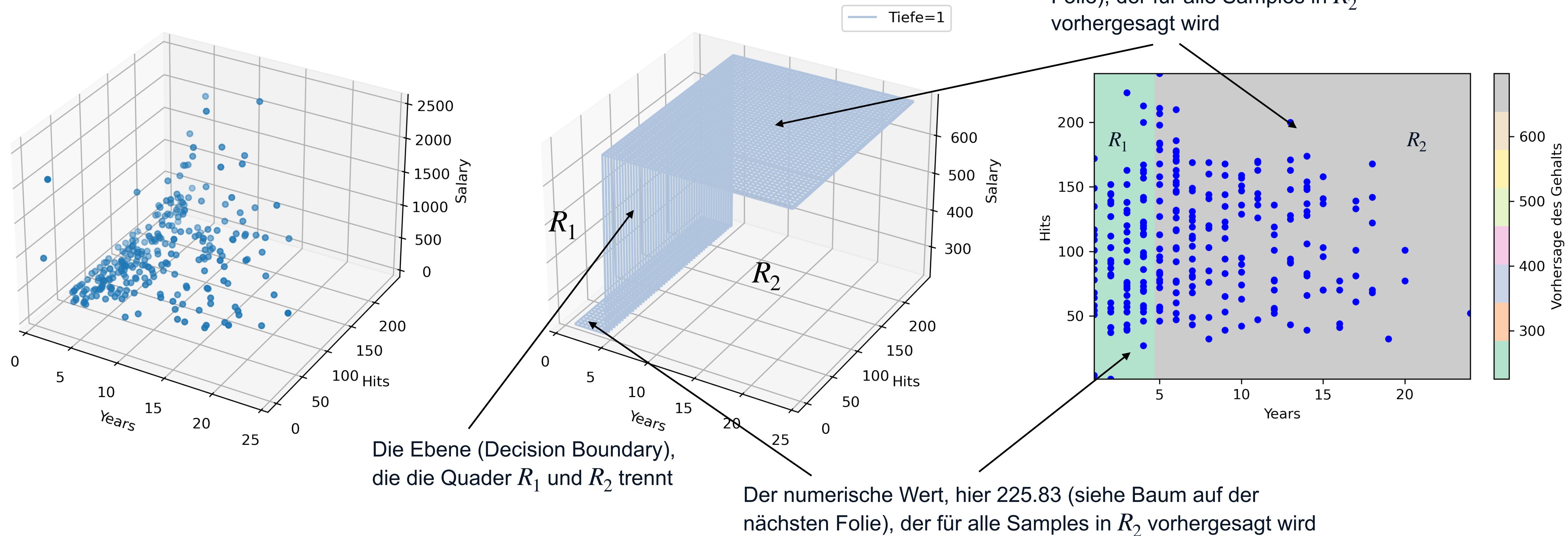


2D Ansicht von „oben“, xy-Ebene



Beispiel: Hitters in 3D

- ▶ Decision Trees
- ▶ Regressionsbäume

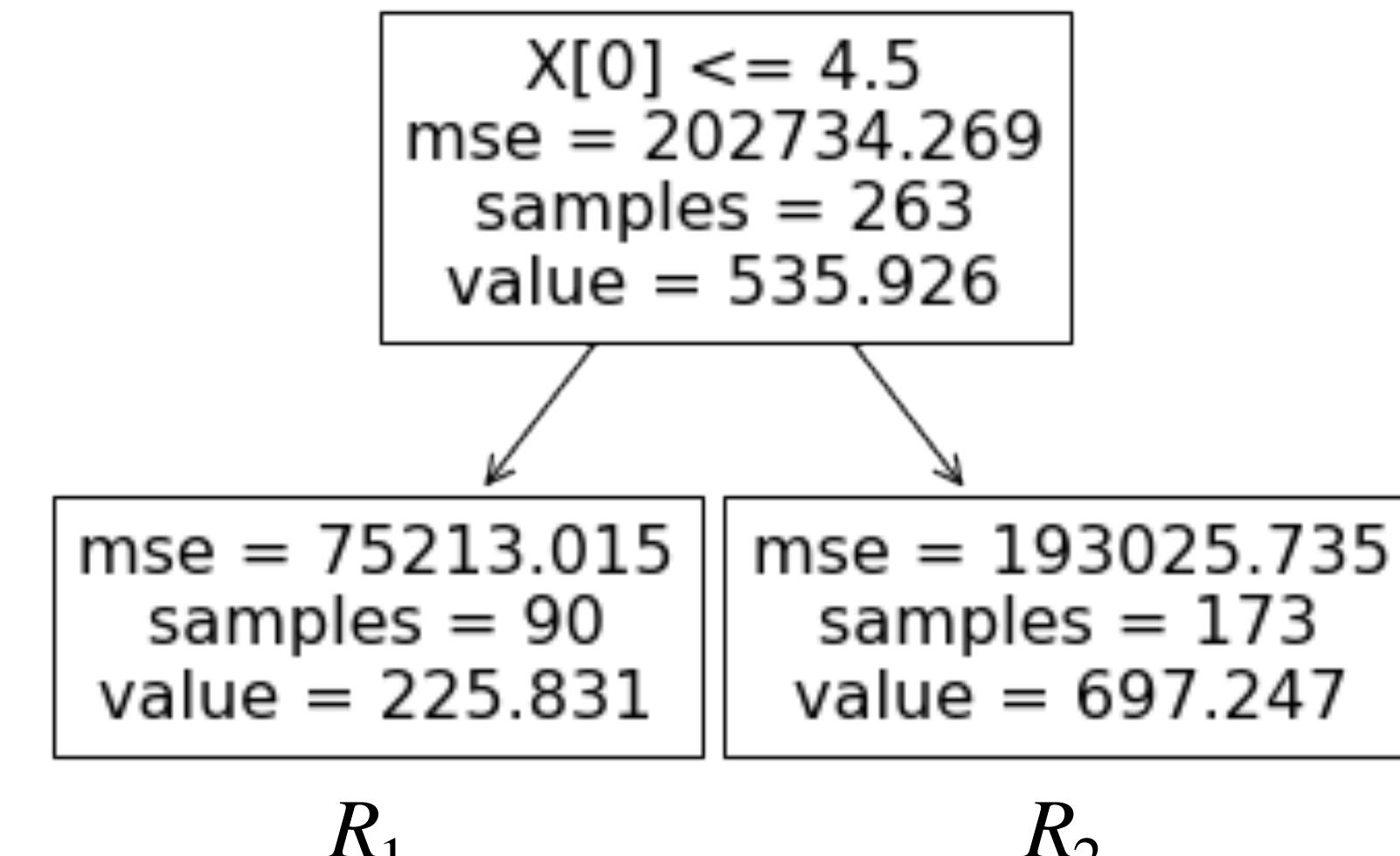


Beispiel: Hitters in 3D

- ▶ Decision Trees
- ▶ Regressionsbäume

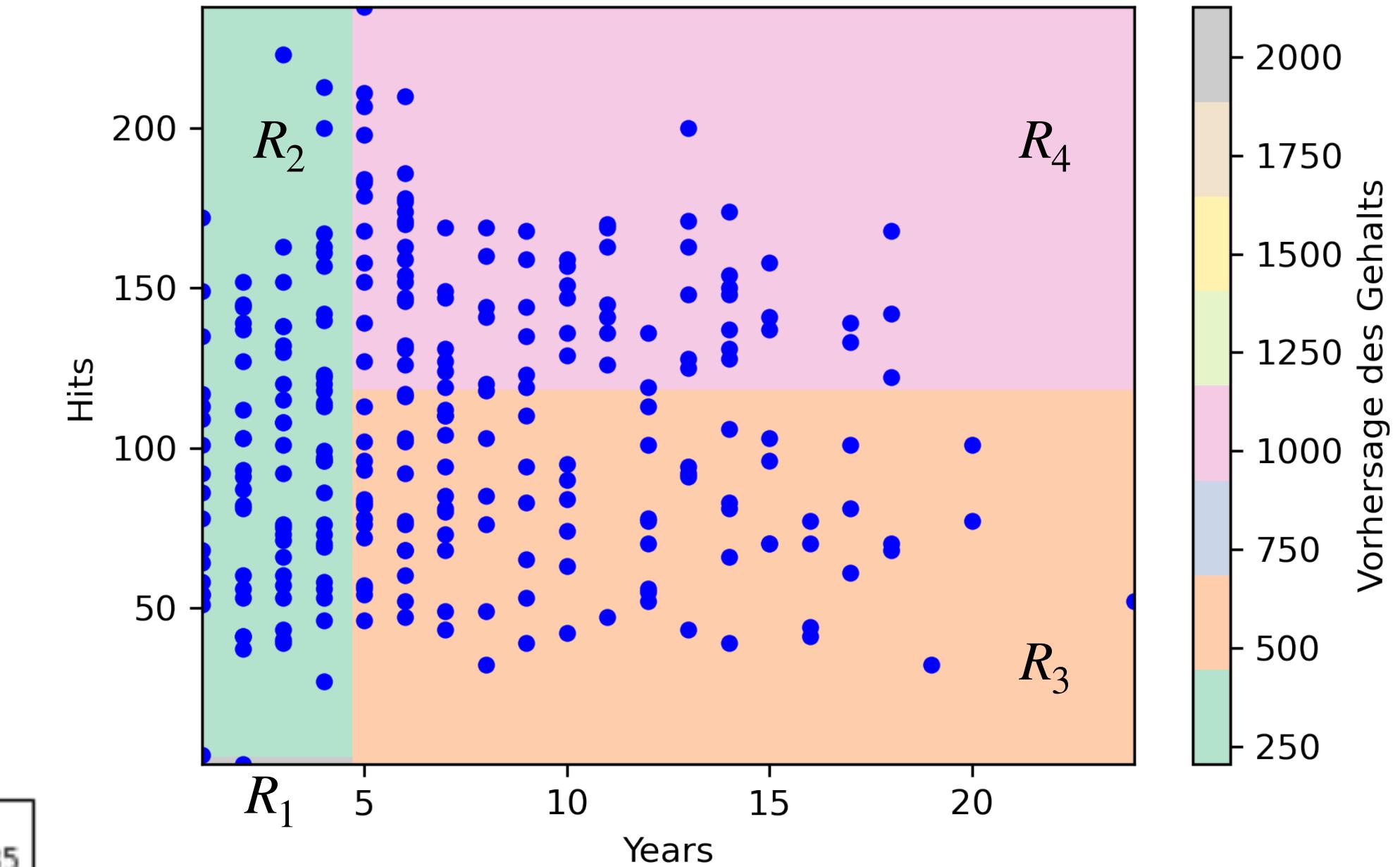
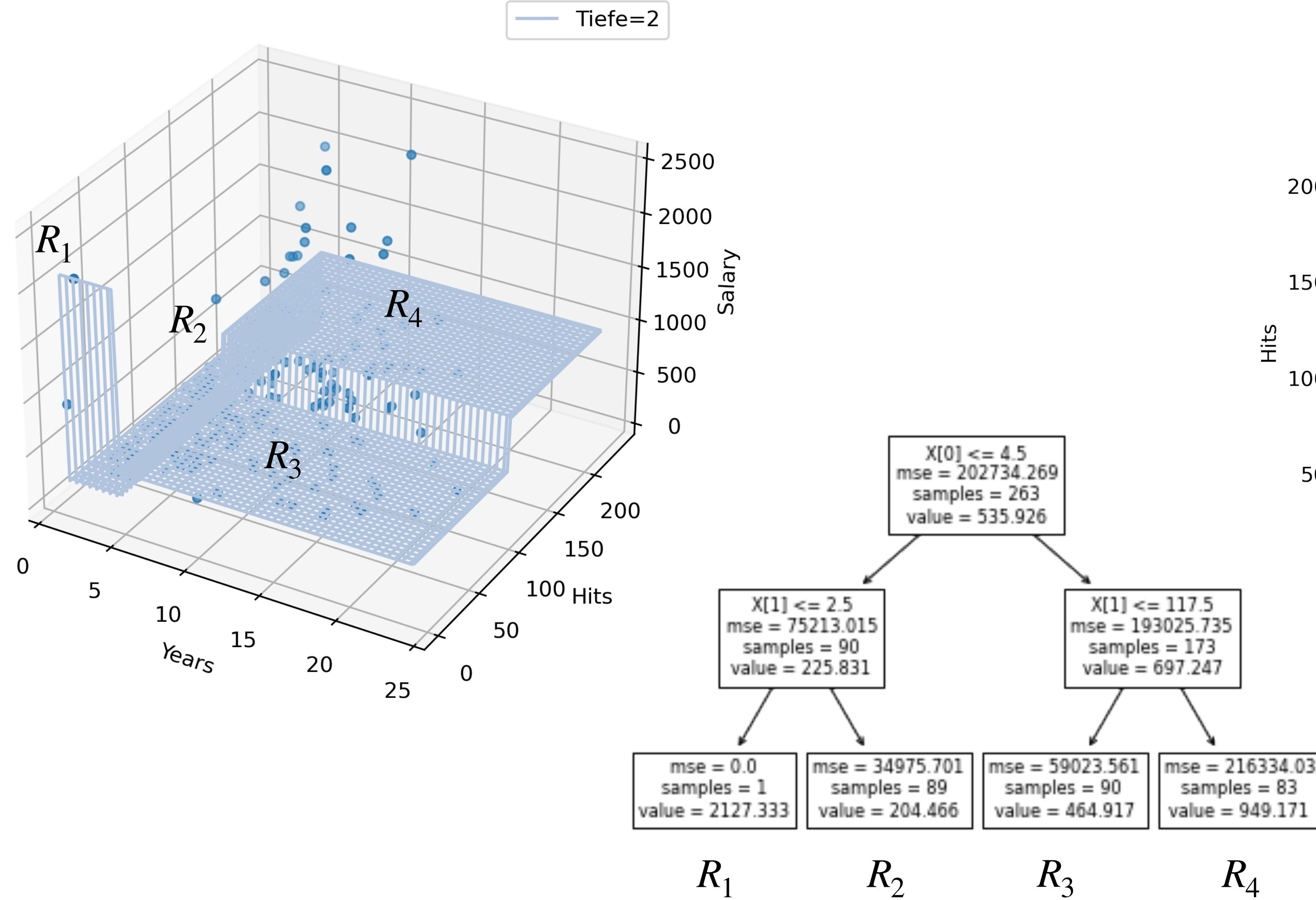
- Im Gegensatz zu 2D liegt der numerische Wert, der in jedem Quader vorhergesagt wird, nicht auf einer konstanten Gerade parallel zur x -Achse, sondern auf einer konstanten Ebene parallel zur xy -Ebene.
- Für den resultierenden Regressionsbaum für Hitters in 3D mit Tiefe = 1 spielt das Feature Hits keine Rolle, der Feature Space wird nur anhand des Features Years ($X[0] \leq 4.5$, vgl. Baum auf der letzten Folie) in R_1 und R_2 geteilt.
- Zur Erinnerung: Das Splitting entsteht in jedem Knoten anhand nur eines Features.

Der resultierende Regressionsbaum bei Tiefe=1, der den Feature Space in zwei Quader, wie auf der letzten Folie dargestellt, splittet



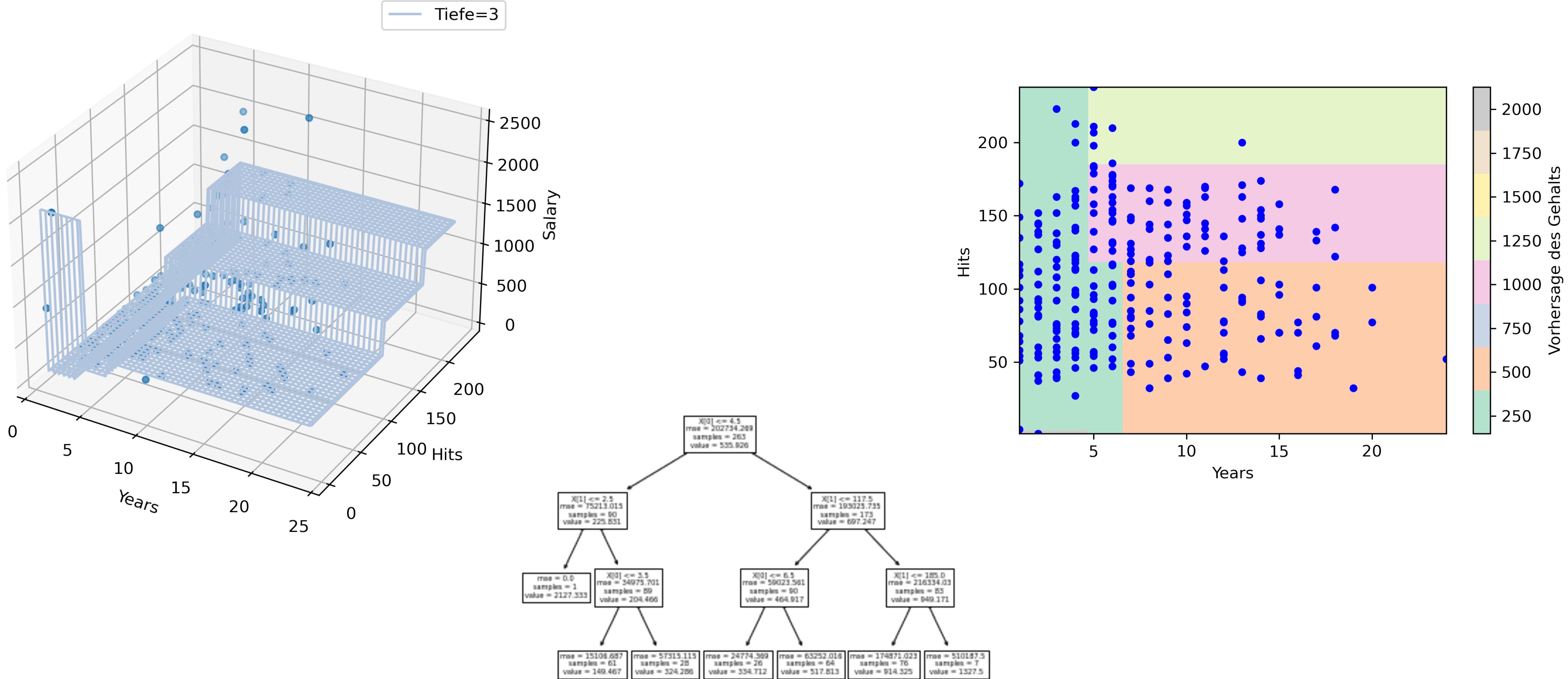
Beispiel: Hitters in 3D

- ▶ Decision Trees
- ▶ Regressionsbäume



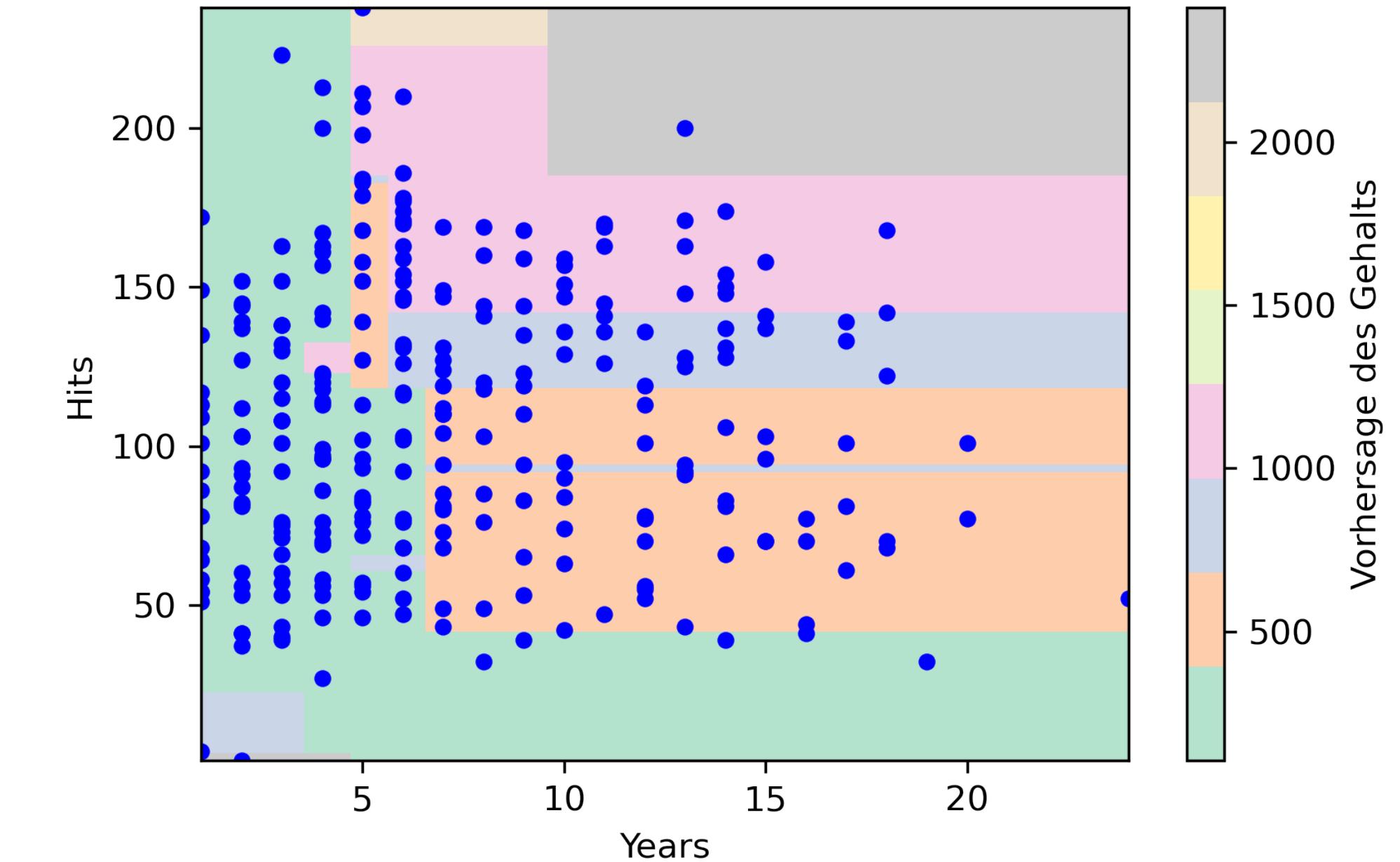
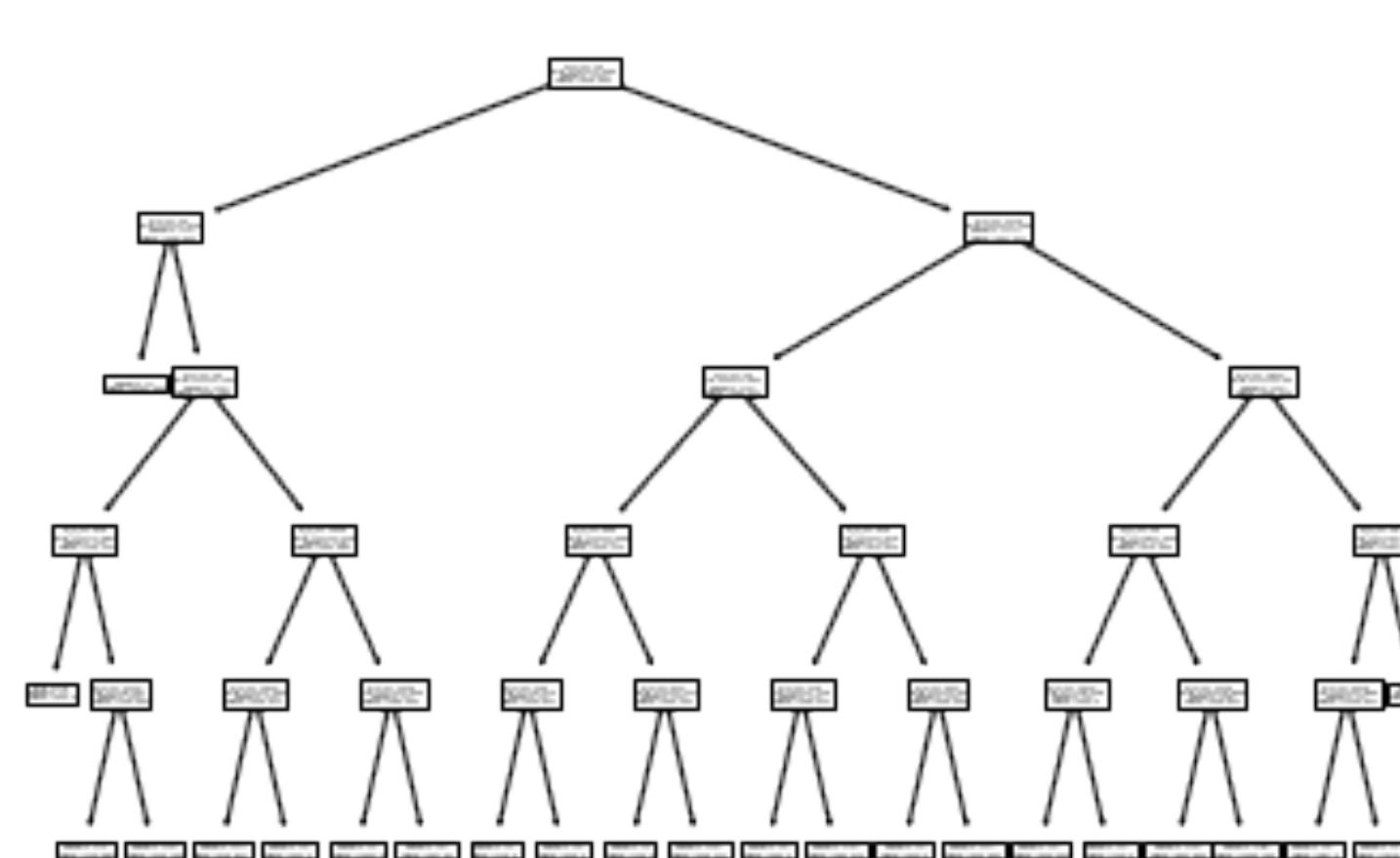
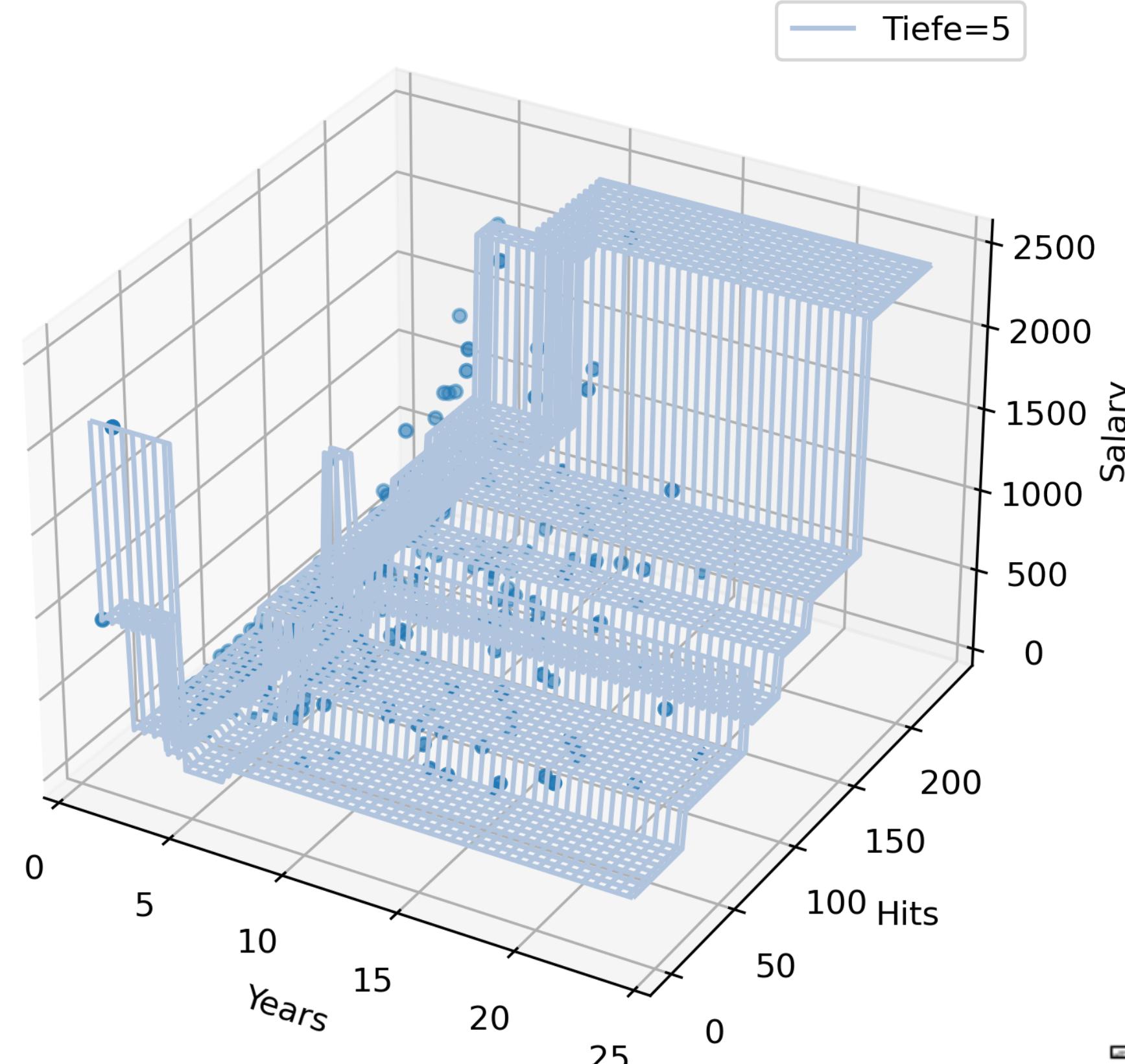
Beispiel: Hitters in 3D

- ▶ Decision Trees
- ▶ Regressionsbäume



Beispiel: Hitters in 3D

- ▶ Decision Trees
- ▶ Regressionsbäume



Beispiel: Hitters in 3D

- Decision Trees
- Regressionsbäume

- Ähnlich wie in 2D passt sich der Baum schnell an die Daten an.
- In diesem Beispiel scheint der Baum mit Tiefe = 3 die Daten am besten abzubilden.
- Hier fällt auf, dass bei Hinzufügen einer weiteren Dimension die Bäume größer und komplizierter werden können.
→ Je mehr Dimensionen, desto mehr Möglichkeiten und Kombinationen existieren, um den Feature Space zu splitten.

Modellierung eines Regressionsbaums

- ▶ Decision Trees
- ▶ Regressionsbäume

- Jedes Sample soll zu genau einem Hyperquader gehören: $x \in R_l$ für genau ein $l \in \{1, \dots, L\}$, $x \notin R_k \forall k \neq l$.
→ Das kann durch eine Indikatorfunktion dargestellt werden: $I(x \in R_l) := \begin{cases} 1, & x \in R_l \\ 0, & \text{sonst} \end{cases}$
- Zur Erinnerung: Für alle Samples im Hyperquader R_l wird derselbe numerische Wert vorhergesagt.
Für jedes Sample x wird ein numerischer Wert vorhergesagt, die Funktion ist somit $f(x) = \sum_{l=1}^L y_{R_l} I(x \in R_l)$.
→ Für jedes Sample x gilt somit $f(x) = y_{R_l}$ für genau ein $l \in \{1, \dots, L\}$.
- Welcher Wert wird im jeweiligen Hyperquader R_l vorhergesagt? Die Kostenfunktion ist der MSE, der minimiert werden soll:
$$\min_{y_{R_l}} \left\{ \sum_{x^{(i)} \in R_l} (y_i - f(x^{(i)}))^2 = \sum_{x^{(i)} \in R_l} (y_i - y_{R_l})^2 \right\}$$
$$\Rightarrow y_{R_l}^* = \frac{1}{|\{x^{(i)} \in R_l\}|} \sum_{x^{(i)} \in R_l} y_i \quad \text{mit } |\{x^{(i)} \in R_l\}| = \#\text{Samples in } R_l$$

→ Der Wert $y_{R_l}^*$, der für Samples in R_l vorhergesagt wird, ist somit der Durchschnitt der Labels über alle Samples in R_l .

Optimierungsproblem eines Regressionsbaums

- Decision Trees
- Regressionsbäume

- Das Optimierungsproblem ist also: Finde eine Partition (L Hyperquader), die den MSE minimiert:

$$\min_{L, y_{R_1}, \dots, y_{R_L}} \left\{ \sum_{l=1}^L \sum_{x^{(i)} \in R_l} (y_i - f(x^{(i)}))^2 = \sum_{l=1}^L \sum_{x^{(i)} \in R_l} (y_i - y_{R_l})^2 \right\}$$

- Dieses Optimierungsproblem lässt sich nicht effizient lösen.
- Stattdessen wird ein *Top-Down Greedy* Ansatz verwendet, das sogenannte *Recursive Binary Splitting* (*rekursives binäres Teilen*), und zwar konkret der Algorithmus *CART* (Classification And Regression Tree). Hinweis: scikit-learn verwendet CART zum Training von Entscheidungsbäumen.
- Top-Down Ansatz: Wir starten oben mit allen Samples, die zunächst zu einem einzigen Hyperquader gehören, und splitten den Feature Space sukzessiv, sodass bei jedem Split zwei neue Knoten (Innenknoten und/oder Blätter) unterhalb des bis dahin aufgebauten Baums entstehen.
- Greedy (gierig): Jeder Split ist der für diesen Schritt beste, d.h. es wird kein Split gesucht, der später zu einem besseren Baum führen kann.

1. R_0 sei der Hyperquader mit den darin enthaltenen Datenpunkten $(x^{(i)}, y_i), i \in \{1, \dots, m\}$,
 $j \in \{1, \dots, n\}$ bezeichne den Index eines Features und s einen Schwellwert.
2. Definiere zwei Halbräume: $R_1(j, s) = \{x \mid x_j \leq s\}$ und $R_2(j, s) = \{x \mid x_j > s\}$
Finde j und s , die die folgende Kostenfunktion (gewichteter MSE) minimieren:

$$C(j, s) := \frac{m_1}{m} \sum_{x^{(i)} \in R_1(j, s)} (y_i - y_{R_1}^*)^2 + \frac{m_2}{m} \sum_{x^{(i)} \in R_2(j, s)} (y_i - y_{R_2}^*)^2,$$

$$\text{mit } y_{R_1}^* = \frac{1}{m_1} \sum_{x^{(i)} \in R_1(j, s)} y_i, \quad y_{R_2}^* = \frac{1}{m_2} \sum_{x^{(i)} \in R_2(j, s)} y_i, \quad m_1 = |x^{(i)} \in R_1(j, s)|, \quad m_2 = |x^{(i)} \in R_2(j, s)|.$$

j und s können insbesondere bei einer geringen Anzahl von Features recht schnell gefunden werden.

3. Kehre zu Schritt 1 zurück, setze jeweils $R_0 = R_1$ und $R_0 = R_2$ und führe jeweils Schritt 2 aus. Fahre solange mit dem Splitting fort, bis ein Abbruchkriterium erreicht ist oder kein Splitting mehr möglich ist.

Decision Trees

Klassifikationsbäume

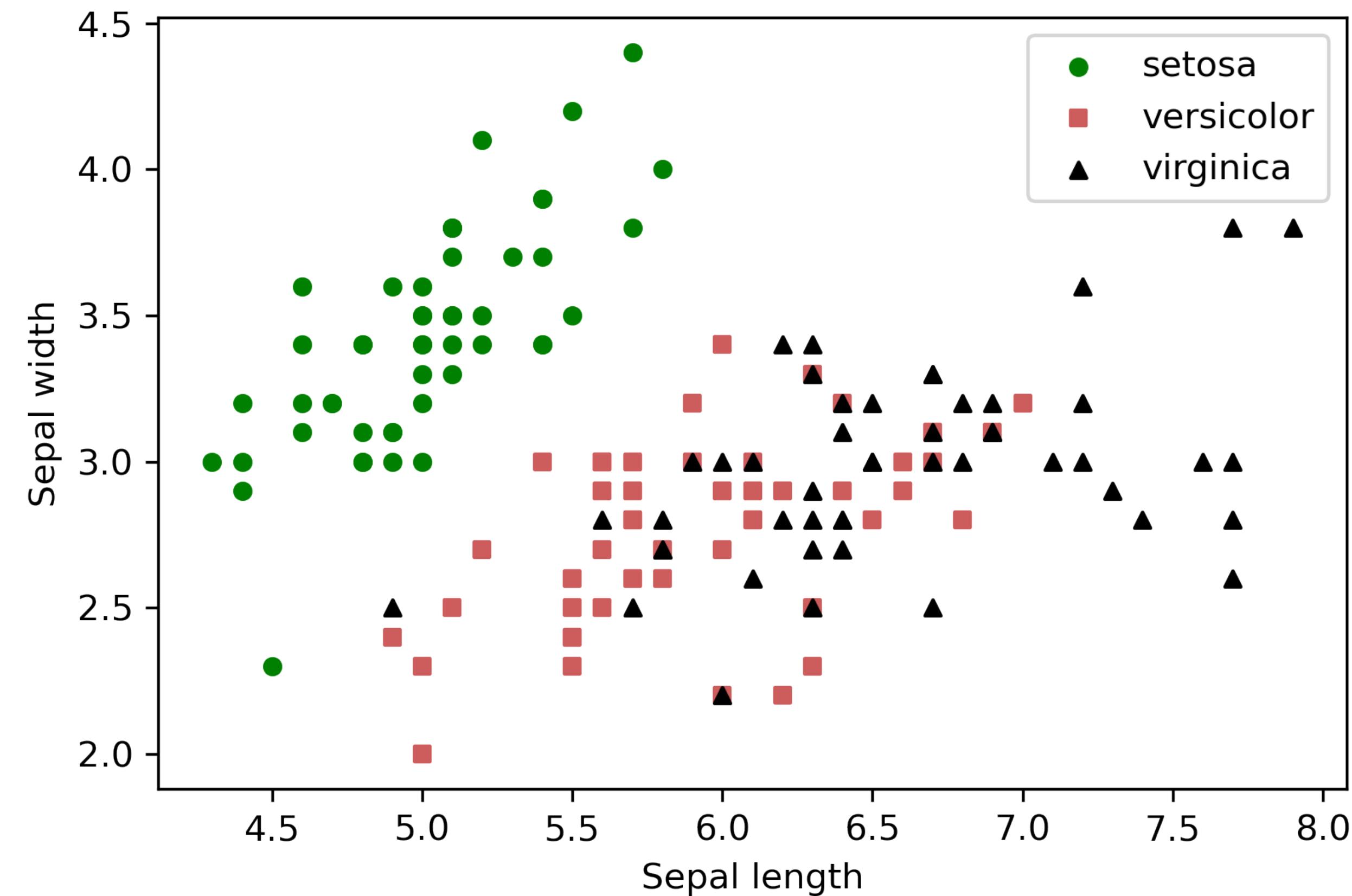
Beispiel: Iris

► Decision Trees

► Klassifikationsbäume

Iris Datensatz

- Wir wollen drei verschiedene Pflanzenarten aus der Familie der Schwertlilien (Iris) mithilfe eines Entscheidungsbaums klassifizieren.
- Die Features, die betrachtet werden, sind Kelchblattlänge und -Breite.



Beispiel: Iris

- ▶ Decision Trees
- ▶ Klassifikationsbäume

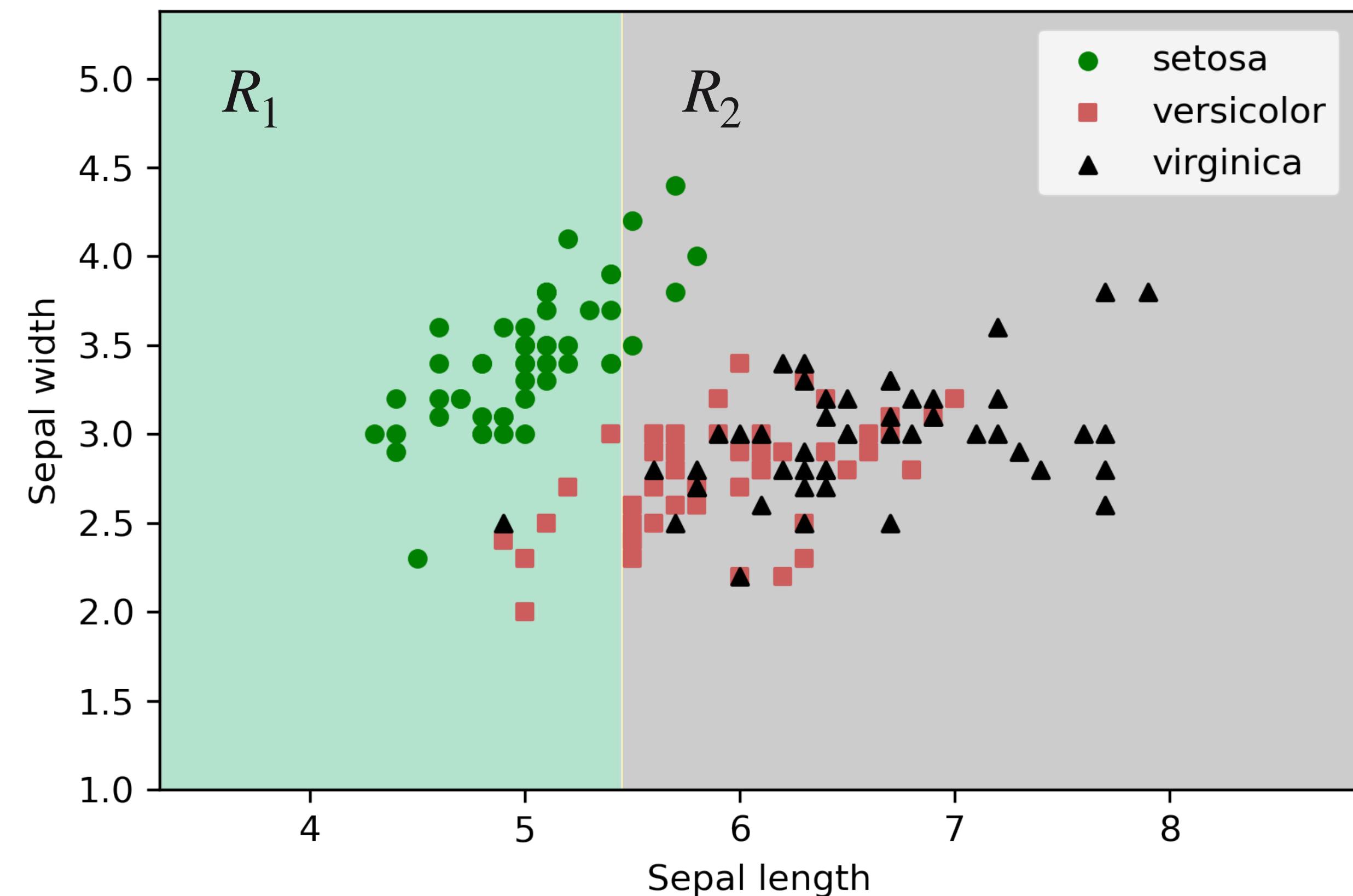
- Im Feature Space wird die Klassifikation des Entscheidungsbaums durch die gestellten Fragen sichtbar.
- Abbildung des Baums und Feature Space für Tiefe = 1

X[0] bezeichnet das erste Feature, hier Sepal length

$X[0] \leq 5.45$
gini = 0.667
samples = 150
value = [50, 50, 50]

gini = 0.237
samples = 52
value = [45, 6, 1]

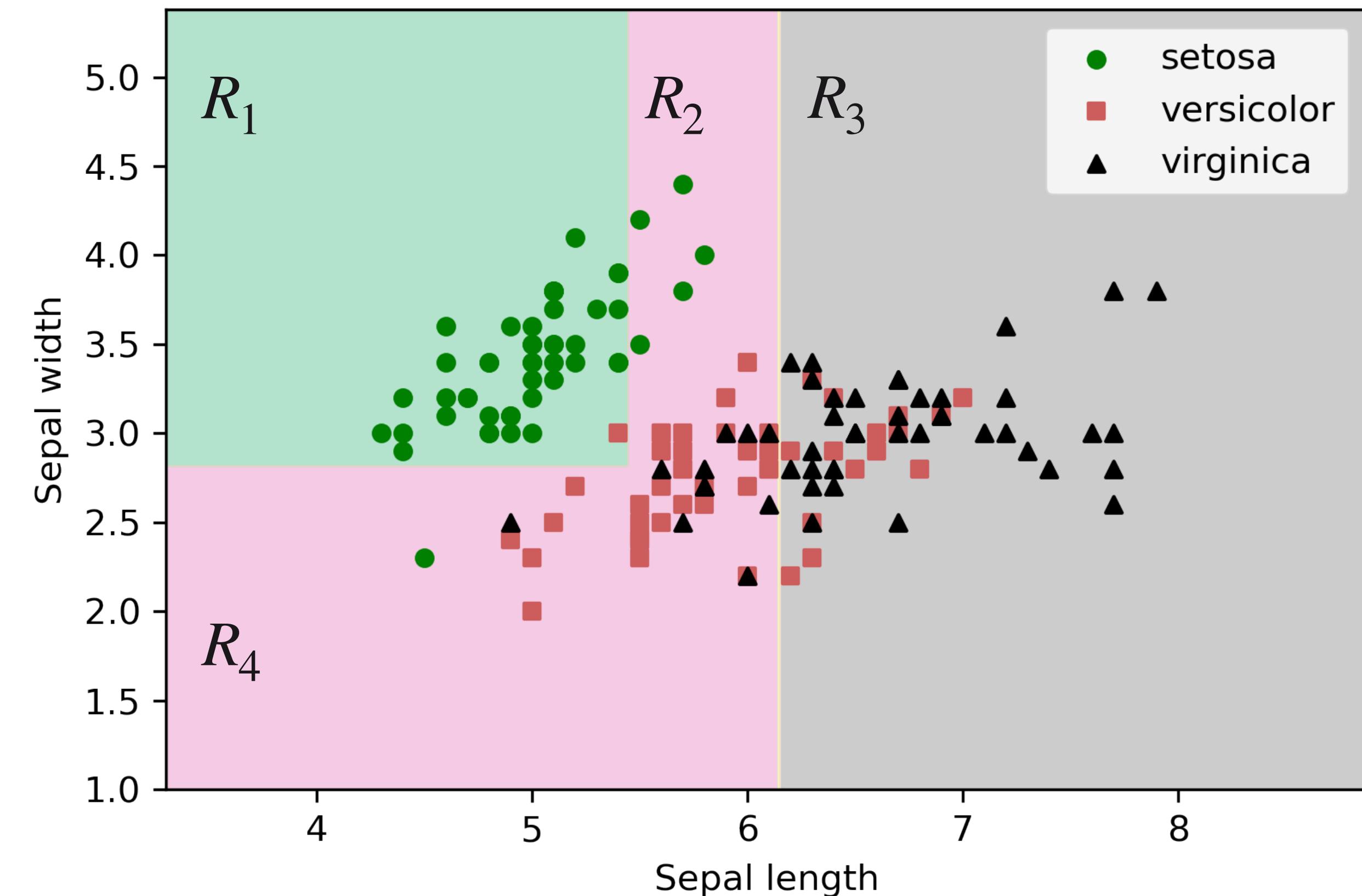
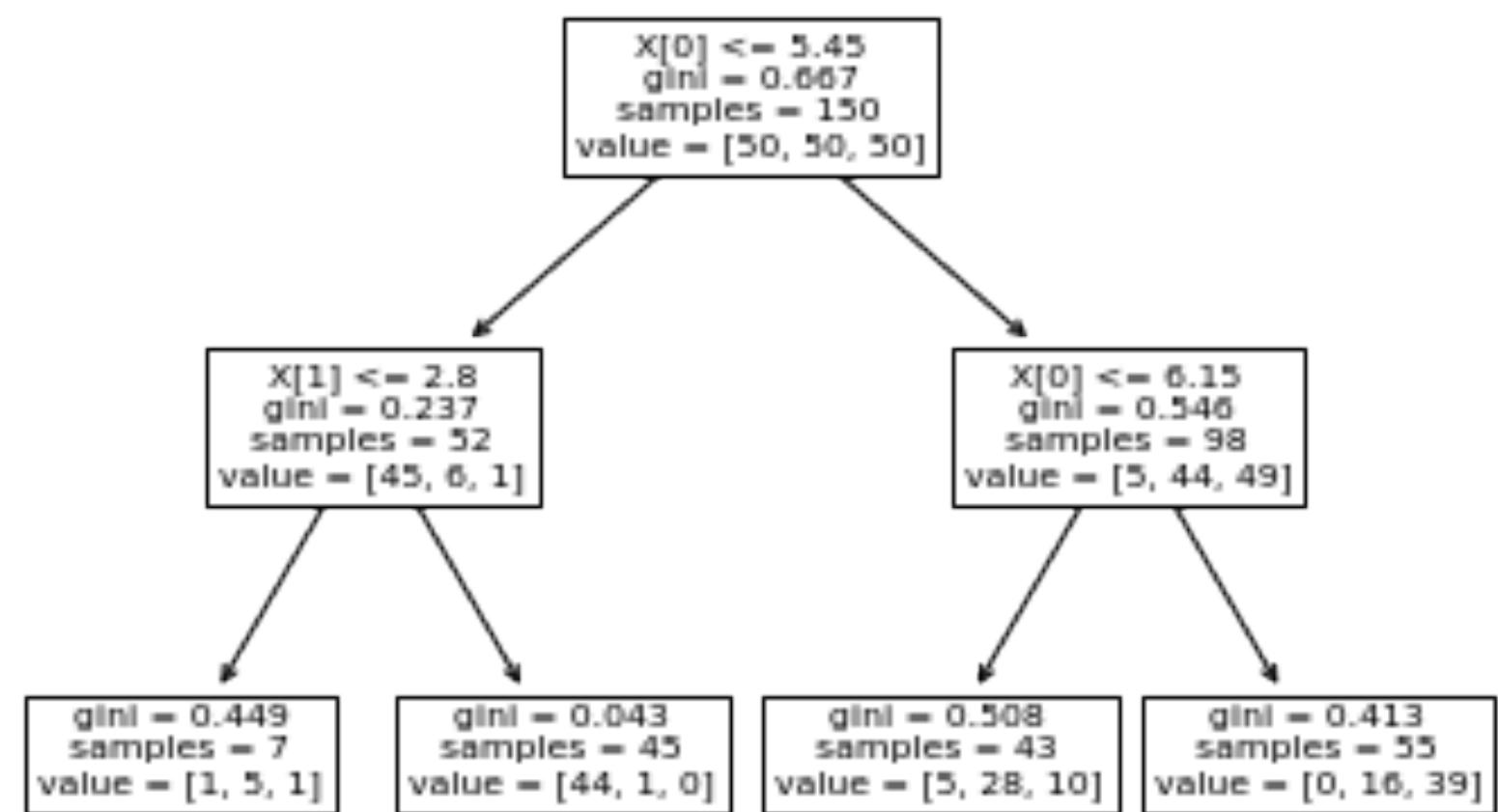
gini = 0.546
samples = 98
value = [5, 44, 49]



Beispiel: Iris

- ▶ Decision Trees
- ▶ Klassifikationsbäume

- Abbildung des Baums und Feature Space für Tiefe = 2



Beispiel: Iris

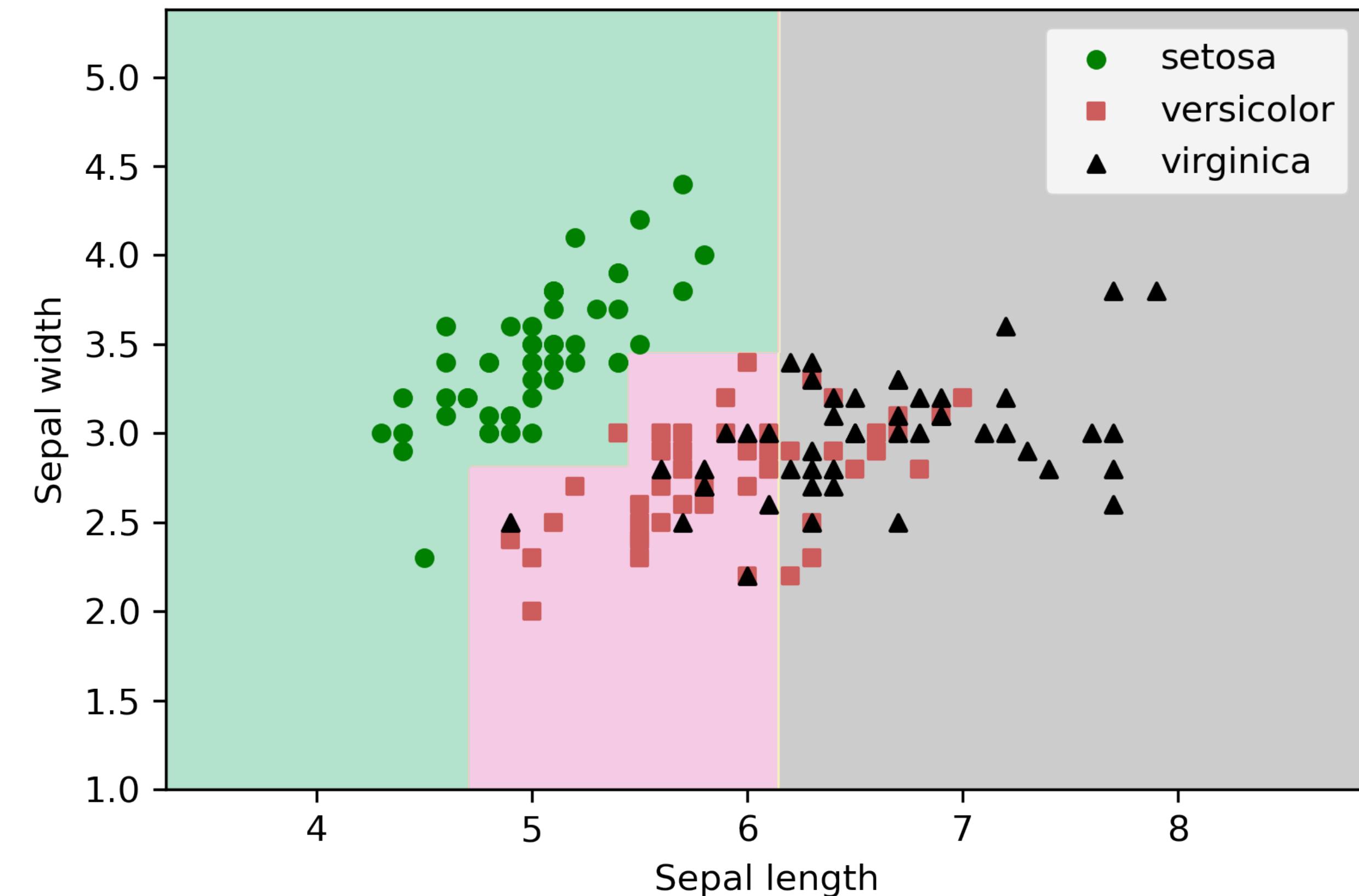
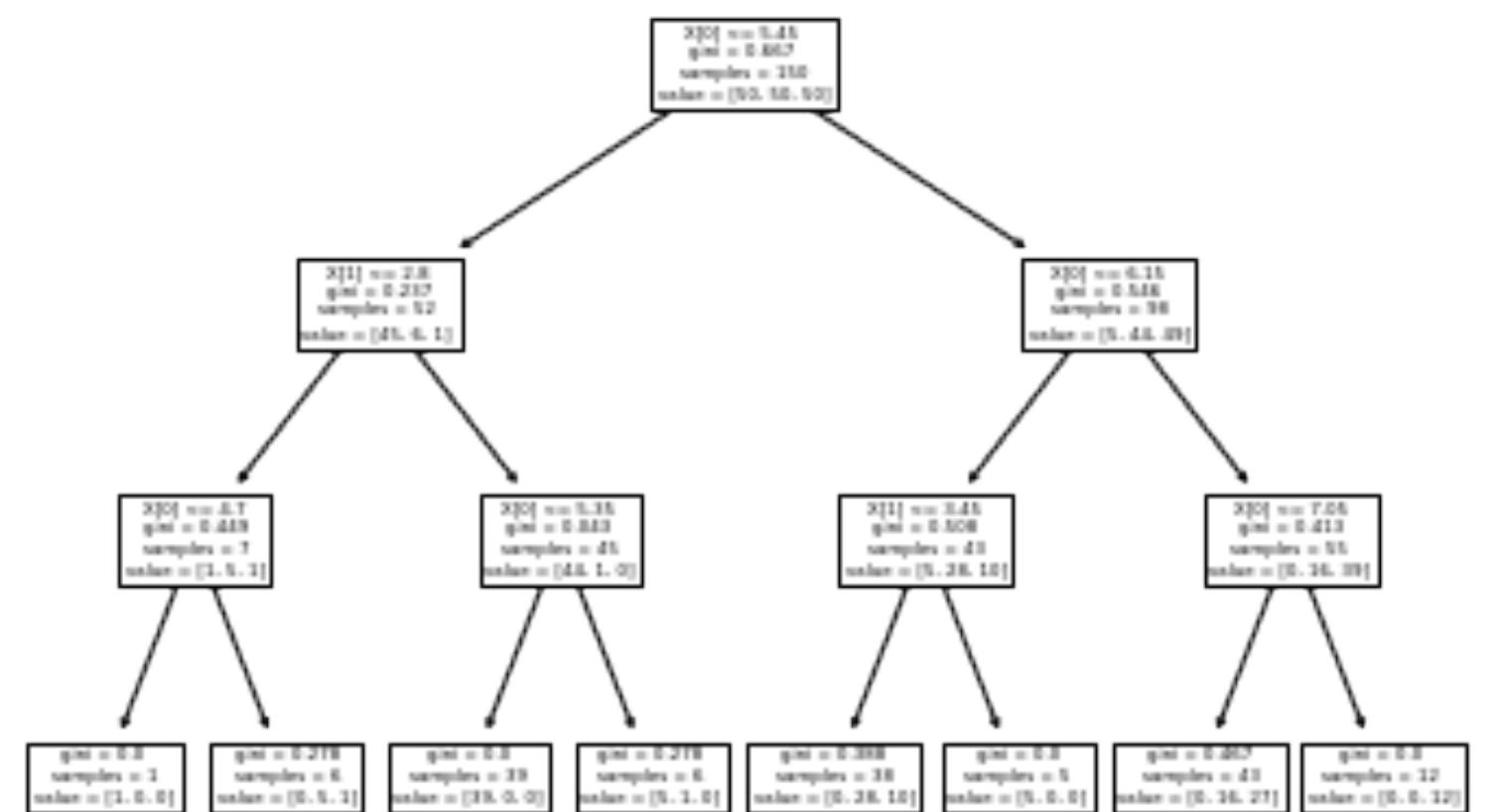
► Decision Trees

► Klassifikationsbäume

se

UDE

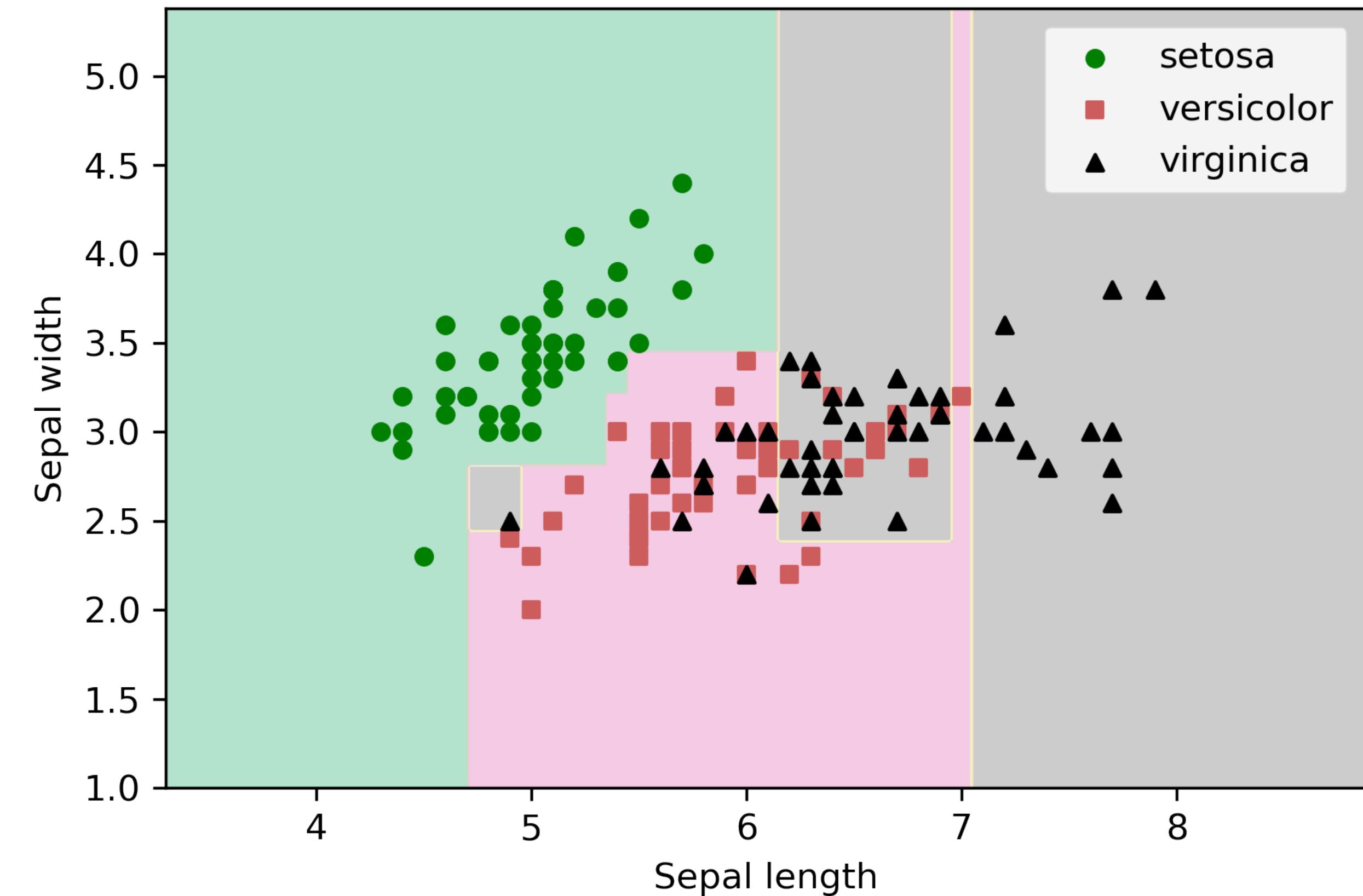
- Abbildung des Baums und Feature Space für Tiefe = 3



Beispiel: Iris

- ▶ Decision Trees
 - ▶ Klassifikationsbäume

- Abbildung des Baums und Feature Space für Tiefe = 5



Vorhersage von Wahrscheinlichkeiten

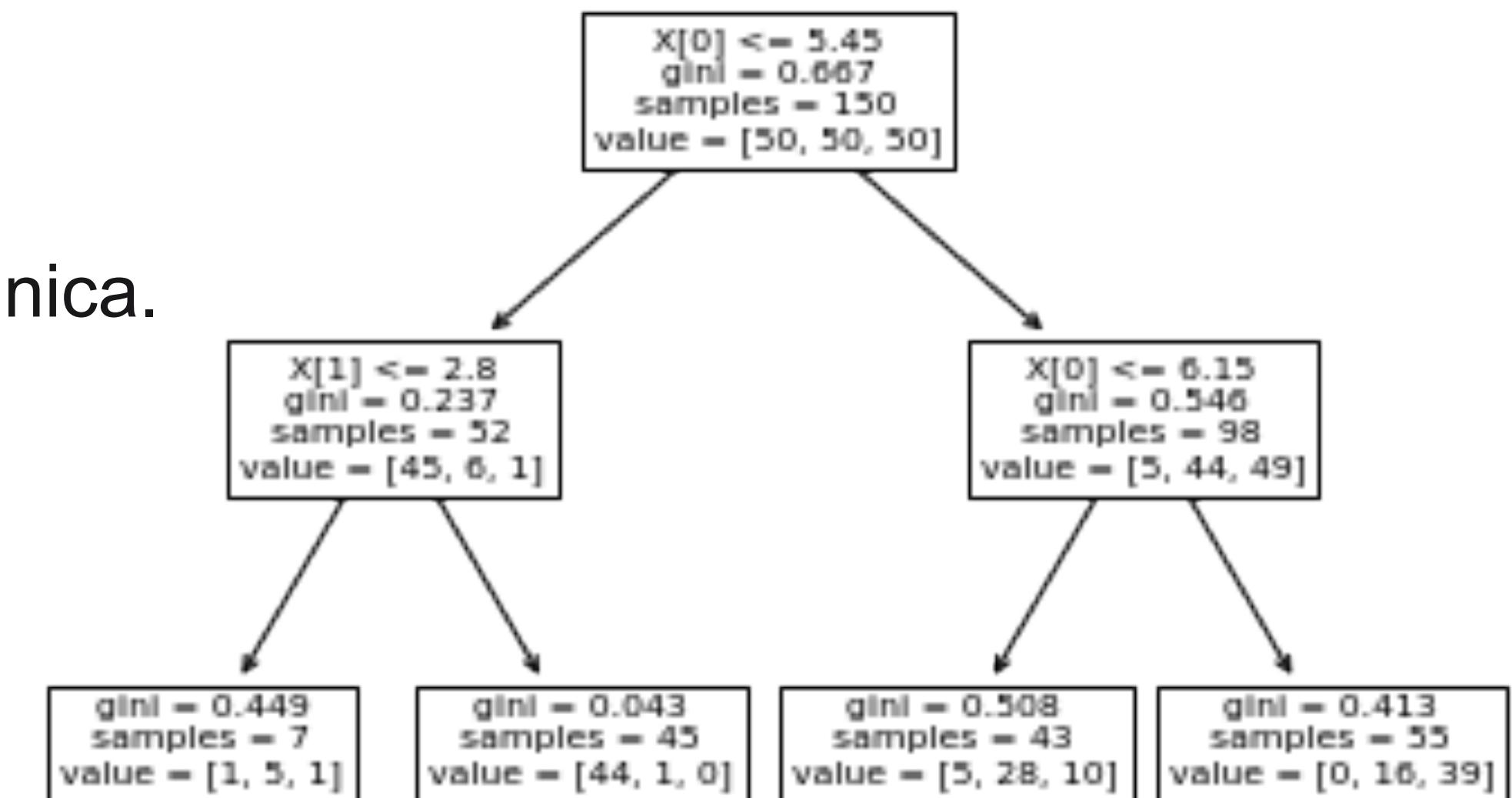
▶ Decision Trees

▶ Klassifikationsbäume

se

UDE

- Ein Klassifikationsbaum kann auch Wahrscheinlichkeiten der Zugehörigkeit eines Sample zu einer Klasse berechnen.
- Wir wollen anhand des Klassifikationsbaum mit Tiefe = 2 für Iris (siehe Abbildung des Baums unten) eine Blüte mit Kelchblattlänge = 6 und Kelchblattbreite = 7 klassifizieren.
- Dieses Sample wird dem Blatt, das ganz rechts liegt, zugeordnet, dieses enthält 55 Samples und die Wahrscheinlichkeit sind:
 - 0% für Setosa (0/55),
 - 29% für Versicolor (16/55),
 - 71% für Virginica (39/55).⇒ die Klasse, die für dieses Sample vorhergesagt wird, ist Virginica.



- Bei Klassifikationsproblemen markiert das Label y_i des Sample $x^{(i)}$, $i \in \{1, \dots, m\}$, die Zugehörigkeit zu einer Klasse $k \in \{1, \dots, K\}$.
- Fehlerkriterien bei Klassifikationsbäumen messen die *Impurity* (*Unreinheit*) der Blätter: Der Fehler ist 0, wenn das Blatt nur Datenpunkte derselben Klasse enthält, der Fehler ist maximal, wenn im Blatt keine Klasse eine Mehrheit hat.
- Klassifikationsbäume geben als Vorhersage die Mehrheitsklasse eines Blattes zurück.
→ Eindeutige Mehrheiten sind zu bevorzugen

- Die Größe $p_{lk} := \frac{1}{|\{x^{(i)} \in R_l\}|} \sum_{x^{(i)} \in R_l} I(y_i = k)$ entspricht dem Anteil der Klasse k , $k \in \{1, \dots, K\}$ in R_l mit

$$I(y_i = k) := \begin{cases} 1, & y_i = k \\ 0, & \text{sonst} \end{cases}$$

- Mögliche Ansätze zur Messung der Unreinheit eines Knotens:

- ***Misclassification error (Fehlklassifikationsrate)***: $E_l = \frac{1}{|\{x^{(i)} \in R_l\}|} \sum_{x^{(i)} \in R_l} I(y_i \neq k) = 1 - \hat{p}_{lk}$, $E_l \in [0,1]$

$$\hat{p}_{lk} := \arg \max_k p_{lk} \text{ Anteil der dominanten Klasse in } R_l, \quad I(y_i \neq k) := \begin{cases} 1, & y_i \neq k \\ 0, & \text{sonst} \end{cases}$$

- ***Gini-index / Gini impurity (Gini-Unreinheit)***: $G_l = 1 - \sum_{k=1}^K p_{lk}^2$, $G_l \in [0,1]$

- ***Entropy (Entropie)***: $H_l = - \sum_{k=1}^K p_{lk} \log(p_{lk})$, $H_l \geq 0$

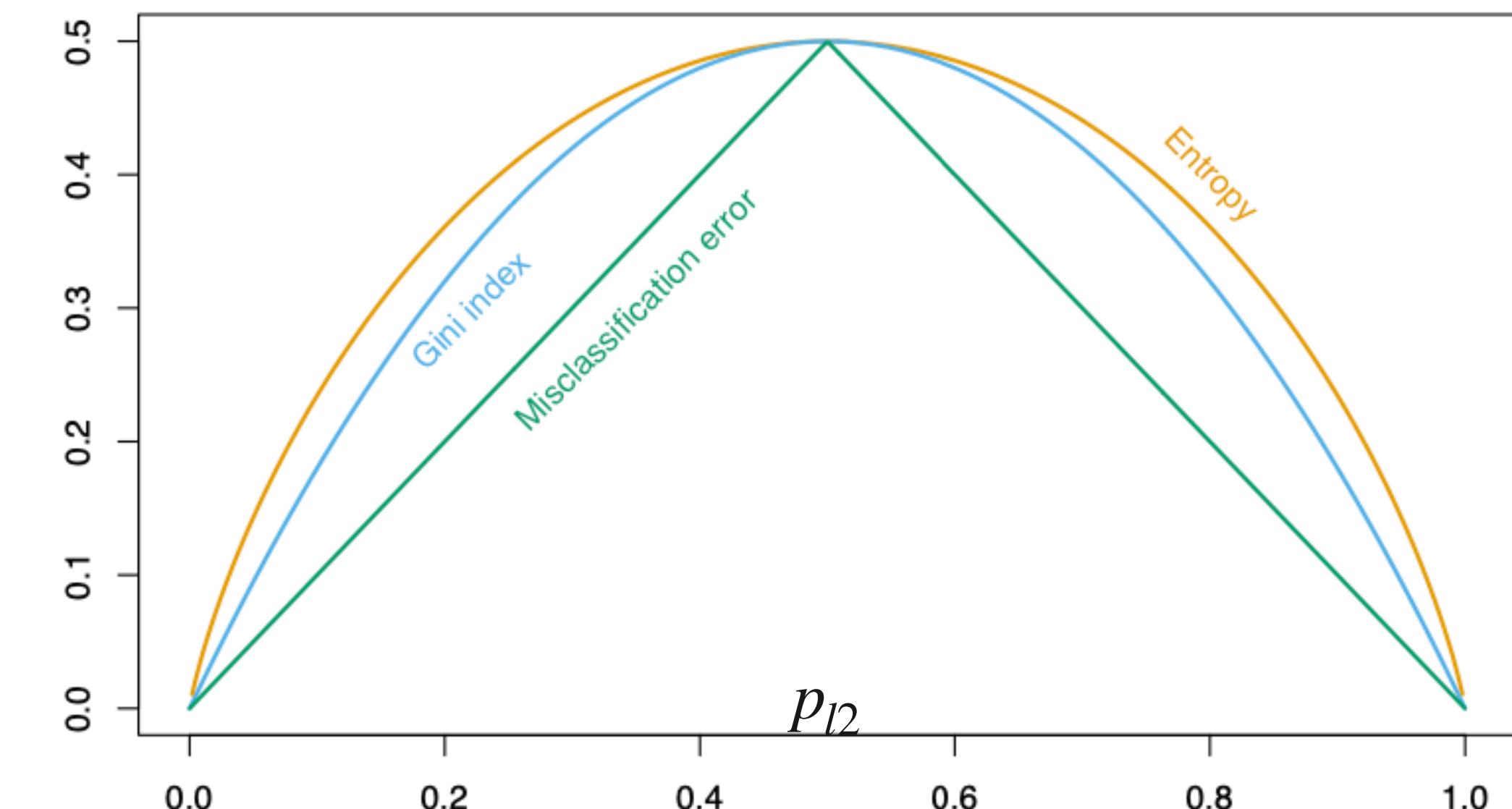
Fehlermessung bei einem Klassifikationsbaum

- ▶ Decision Trees
- ▶ Klassifikationsbäume

- Sowohl die Gini-Unreinheit G_l als auch die Entropie H_l nehmen Werte nahe 0 an, wenn alle Anteile p_{l1} bis p_{lK} nahe 0 oder nahe 1 sind, sodass die Gini-Unreinheit und die Entropie zur Messung der Unreinheit eines Knotens geeignet sind.

- Links sind die Größen für die Klassifikation von zwei Klassen dargestellt, als Funktion des Anteils p_{l2} in der

$$\text{Klasse } 2, p_{l2} = \frac{1}{|\{x^{(i)} \in R_2\}|} \sum_{x^{(i)} \in R_2} I(y_i = 2).$$



- Die jeweilige Größe ist 0, wenn alle Samples eines Knotens derselben Klasse angehören.
→ Das Splitting hört an dieser Stelle auf und der Knoten entspricht somit einem Blatt.
- CART verwendet als Kostenfunktion

$$C(j, s) = \frac{m_1}{m} G_{links} + \frac{m_2}{m} G_{rechts} \text{ oder } C(j, s) = \frac{m_1}{m} H_{links} + \frac{m_2}{m} H_{rechts}$$

$G_{links/rechts}$ ist die Gini Unreinheit des nachfolgenden Knotens links/rechts, $H_{links/rechts}$ die Entropie.

- Meist macht es keinen großen Unterschied, ob Gini-Unreinheit oder Entropie verwendet wird, da die resultierenden Bäume ähnlich sind.
 - Die Gini Unreinheit lässt sich etwas schneller berechnen.
 - Wenn aber die beiden Größen voneinander abweichen, so tendiert die Gini Unreinheit dazu, die am häufigsten vorkommende Klasse in einem eigenen Ast des Baums zu isolieren, während die Entropie etwas balanciertere Bäume erzeugt.

Decision Trees

Hyperparameter, Overfitting, Underfitting

- Es gibt Hyperparameter, die als Kriterien zum Abbruch des Algorithmus verwendet werden können: **diese sind somit keine Größen, die gelernt werden, sondern vor dem Training gesetzt werden müssen.**
- Die Klasse `DecisionTreeClassifier` von scikit-learn, die CART zum Training verwendet, bietet folgende Hyperparameter:
 - `max_depth`: Maximale Baumtiefe,
 - `min_samples_leaf`: Mindestanzahl an Samples in jedem Blatt,
 - `min_weight_fraction_leaf`: Mindestanzahl an Samples in jedem Blatt als Anteil der gesamten gewichteten Samples,
 - `min_samples_split`: Mindestanzahl an Samples in jedem Innenknoten vor dem Splitting,
 - `max_leaf_nodes`: Maximale Anzahl an Blättern,
 - `max_features`: Maximale Anzahl an Features, die beim Splitting betrachtet werden

- Es kann ebenso wie bei anderen Modellen beim Training zu Over- und Underfitting kommen.
- Ist die Baumtiefe zu klein, kommt es zu Underfitting: Bei Tiefe = 1 unterscheidet beim Beispiel Iris das Modell nur zwischen Setosa und Versicolor, obwohl drei Klassen vorliegen
→ Bei Klassifikationsbäumen muss beim Setzen der Baumtiefe die Anzahl der Klassen beachtet werden! Ist $K = 3$, so muss die Baumtiefe mindestens 2 betragen.
- Ist die Baumtiefe zu groß, wird die Decision Boundary zu kompliziert, das Modell passt sich (zu) stark an die Besonderheiten des Trainingsdatensatzes an.
- Was passiert, wenn wir die Baumtiefe nicht beschränken?

Overfitting und Underfitting

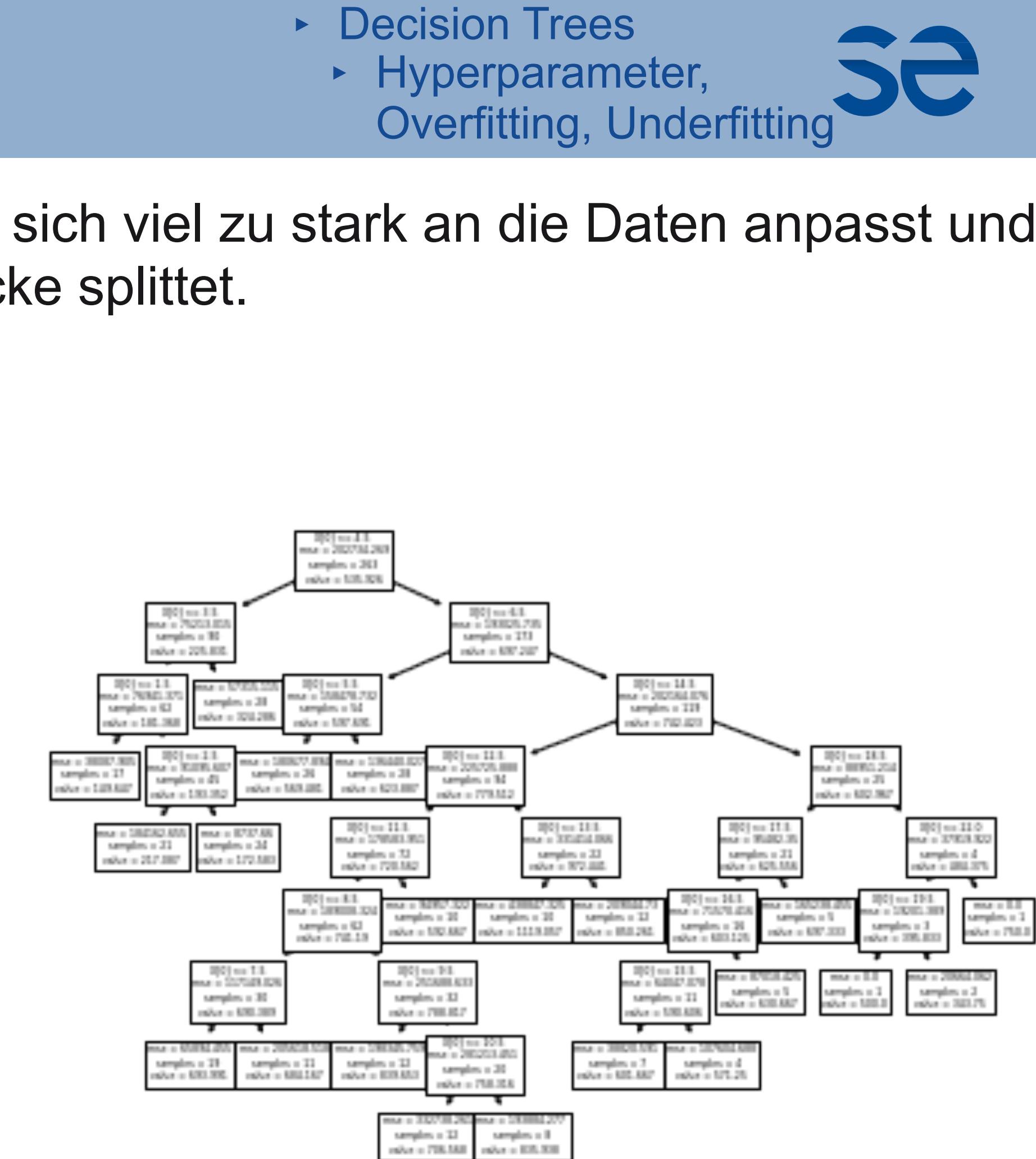
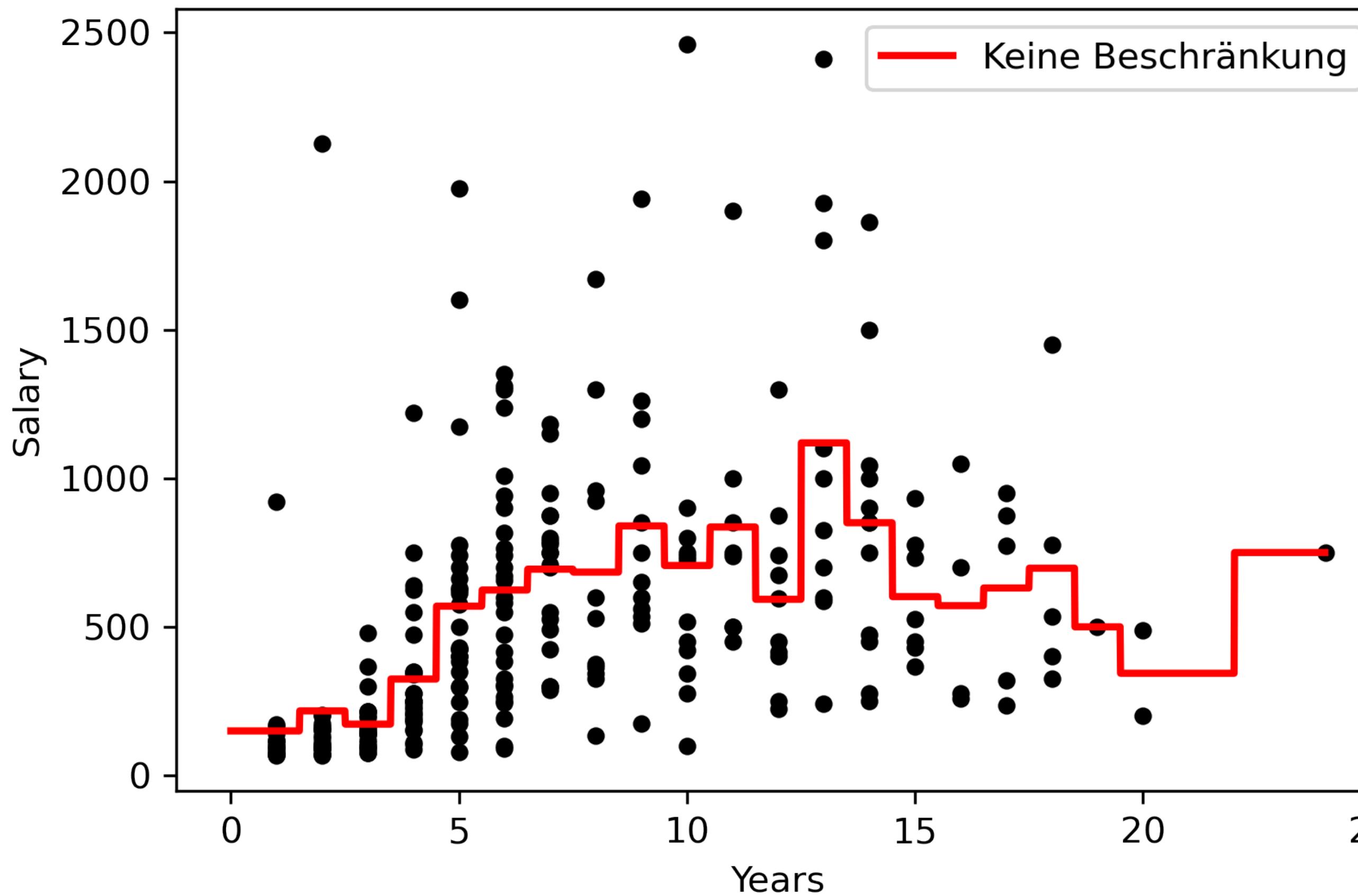
- Decision Trees
- Hyperparameter,
Overfitting, Underfitting

- Was können wir gegen Overfitting unternehmen? → *Regularisierung* (wird später in der Vorlesung detailliert behandelt): Wahl eines Modells mit geringerer Komplexität.
- Eine Möglichkeit der Regularisierung ist die Begrenzung der Hyperparameter des Baums.
Hinweis: Welche Hyperparameter es gibt, hängt vom Algorithmus ab, hier betrachten wir die vorgestellten Hyperparameter.
- Erhöhen der `min_*` Parameter und Senken der `max_*` Parameter führt zur Regularisierung, denn sonst wird der Baum ganz ohne Einschränkungen trainiert und das Training kann sich „austoben“.
- Eine andere mögliche Art der Regularisierung ist *Pruning*: Baum wird ausgewachsen und nachträglich gestutzt durch Entfernen von überflüssigen Knoten.
 - Regression: Eine mögliche Methode ist *Cost Complexity Pruning*, und zwar das Entfernen von internen Knoten in der Reihenfolge, in der sie zum geringsten Anstieg des Vorhersagefehlers führen.
 - Klassifikation: Es werden solche Innenknoten entfernt, deren beide Kinder Blätter sind und der durch sie erbrachte Zugewinn an Reinheit nicht statistisch signifikant ist, hierfür werden statistische Tests eingesetzt.

Beispiel: Regression Hitters in 2D

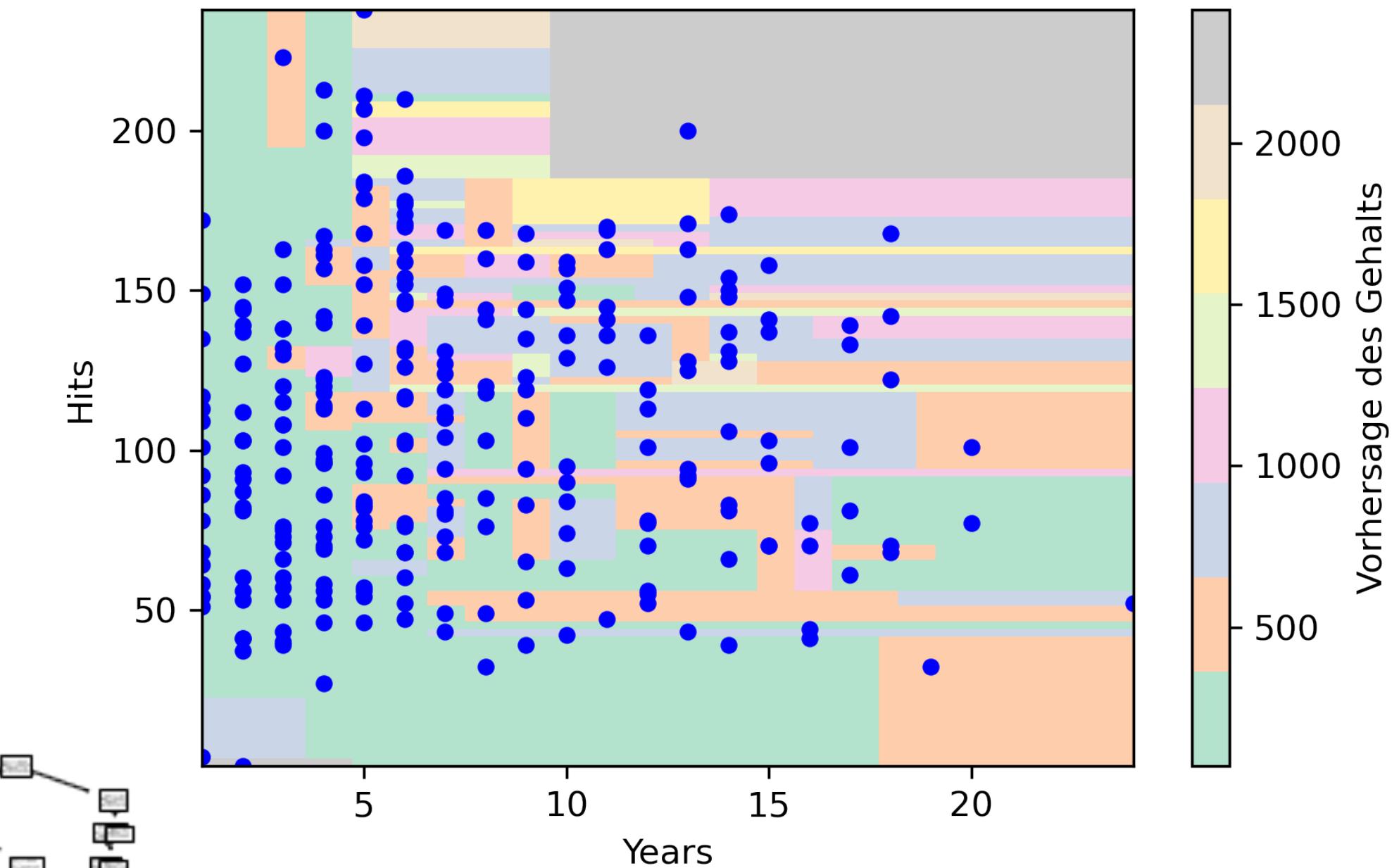
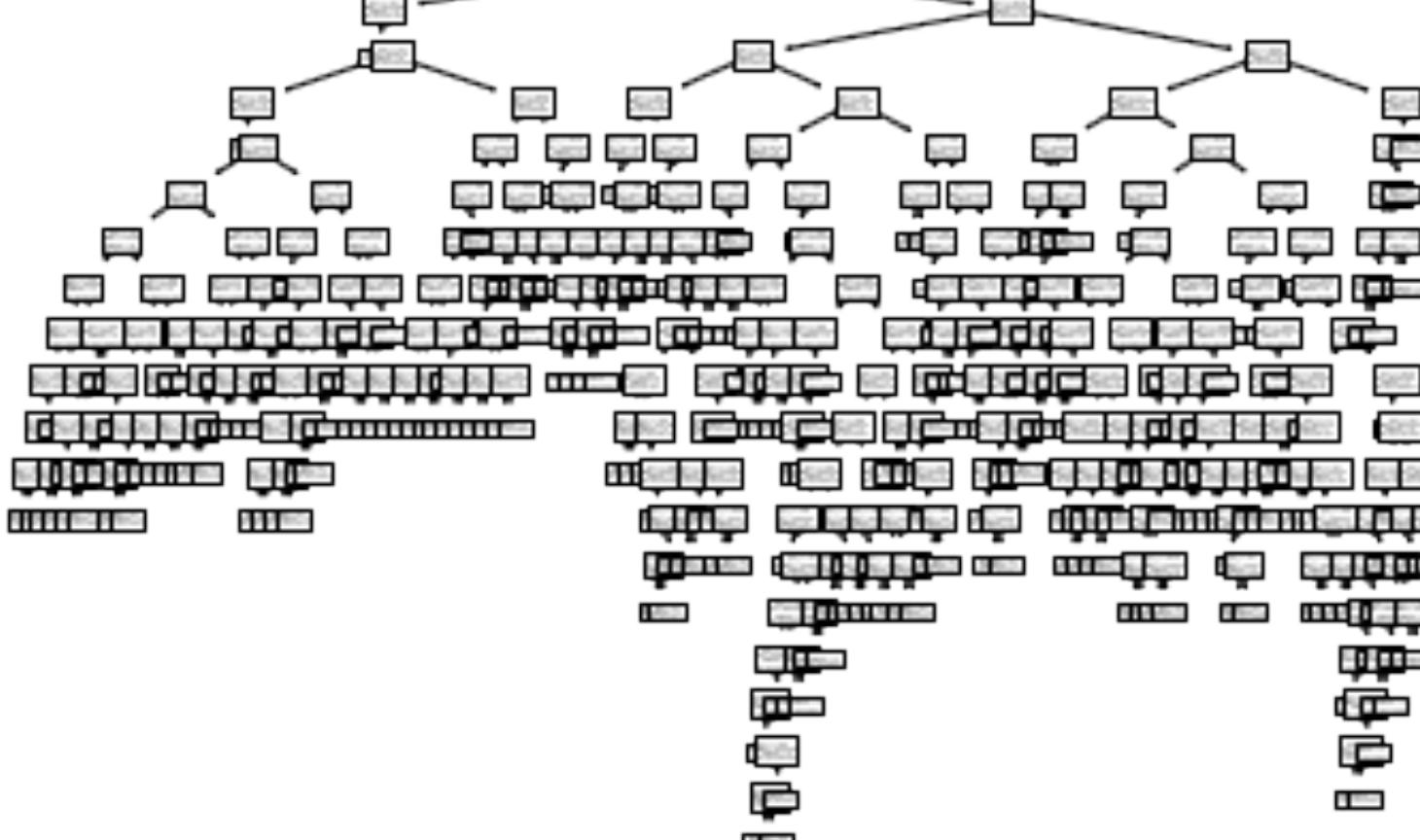
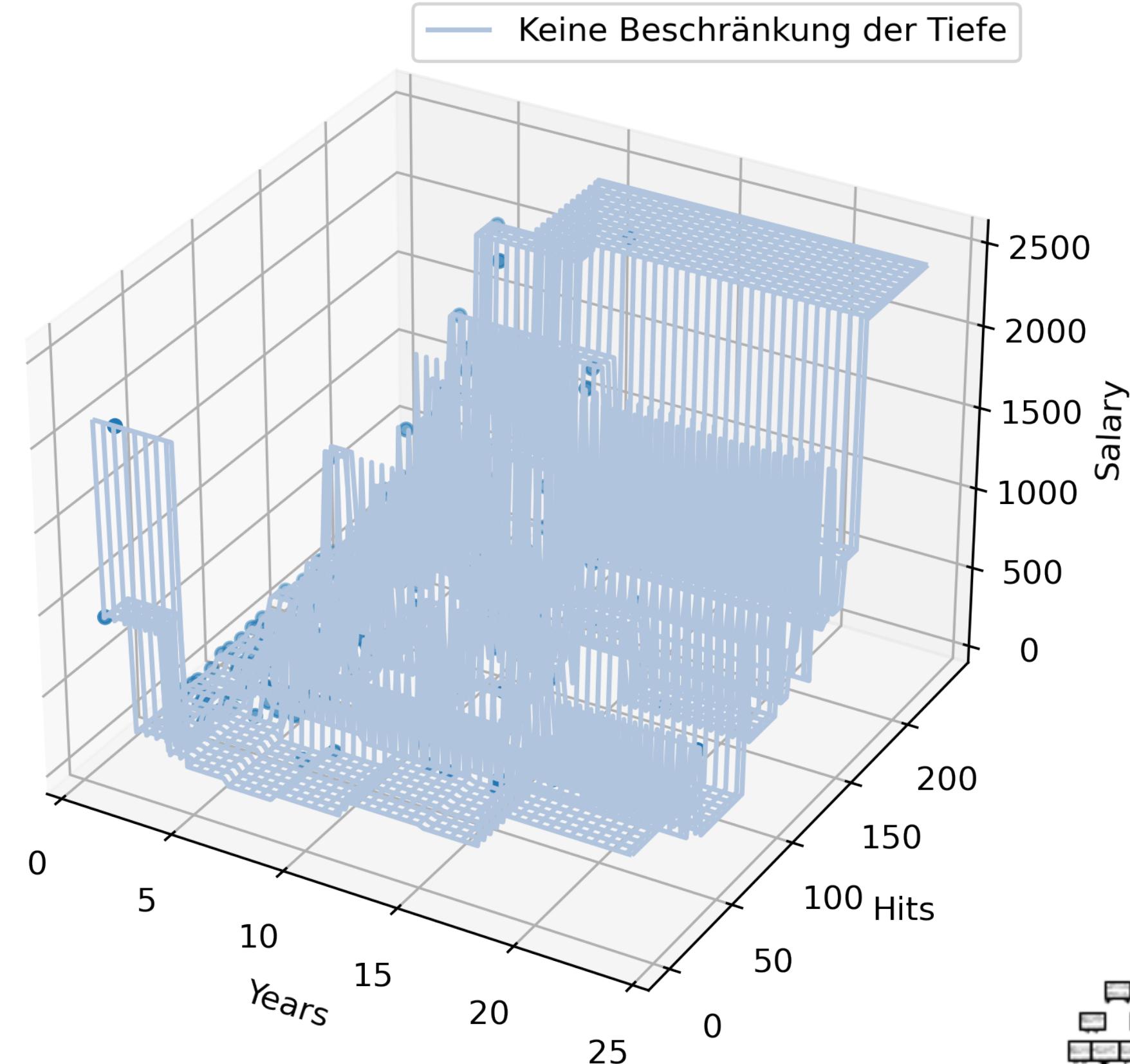
- Decision Trees
- Hyperparameter, Overfitting, Underfitting

- Wird die Baumtiefe nicht beschränkt, so entsteht ein Baum, der sich viel zu stark an die Daten anpasst und der beim Beispiel Hitters in 2D den Feature Space in 21 (!) Rechtecke splittet.



Beispiel: Regression Hitters in 3D

- ▶ Decision Trees
- ▶ Hyperparameter, Overfitting, Underfitting

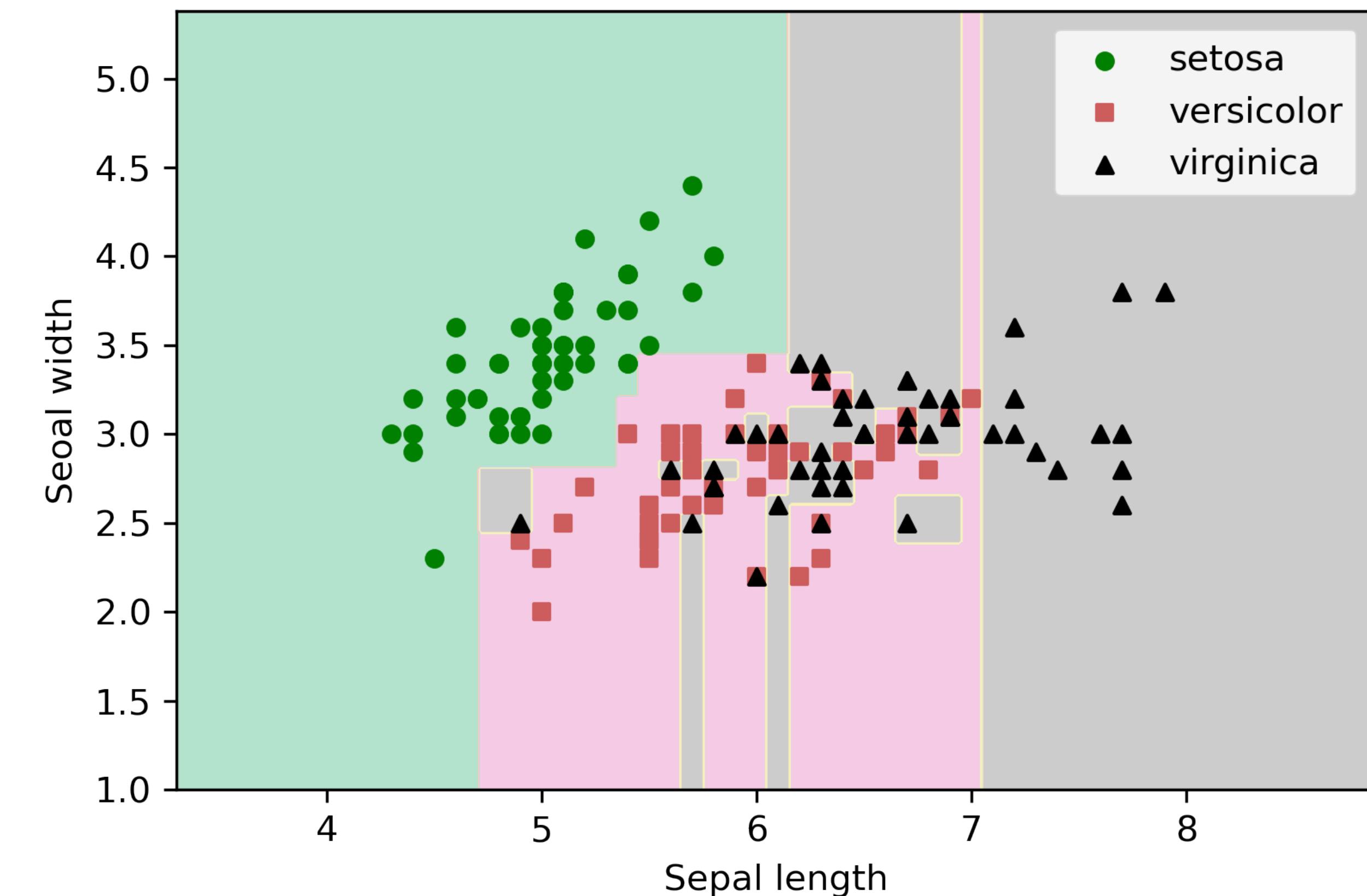
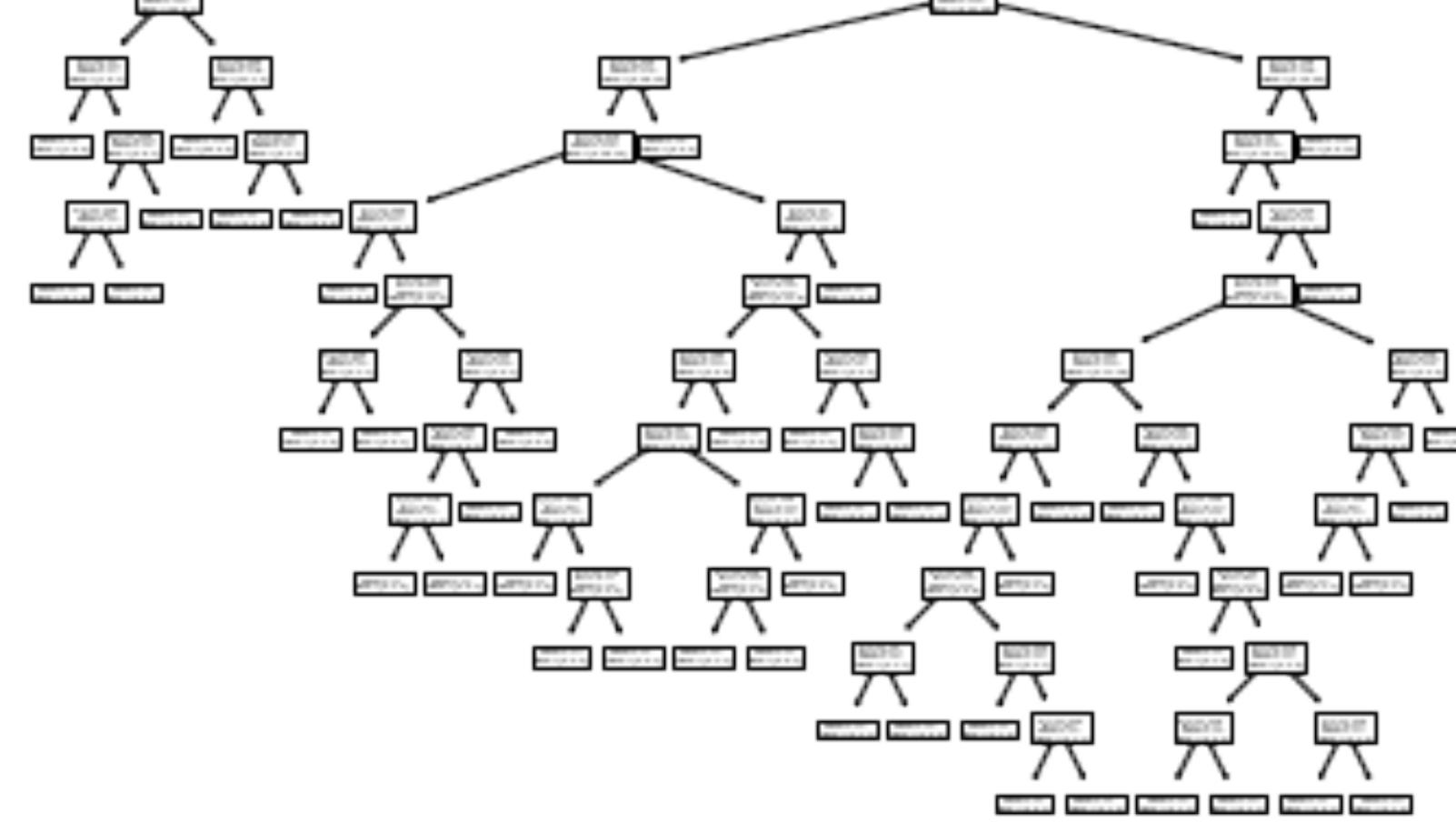


Beispiel: Klassifikation Iris

► Decision Trees

► Klassifikationsbäume

- Abbildung des Baums und Feature Space bei einer unbeschränkten Tiefe



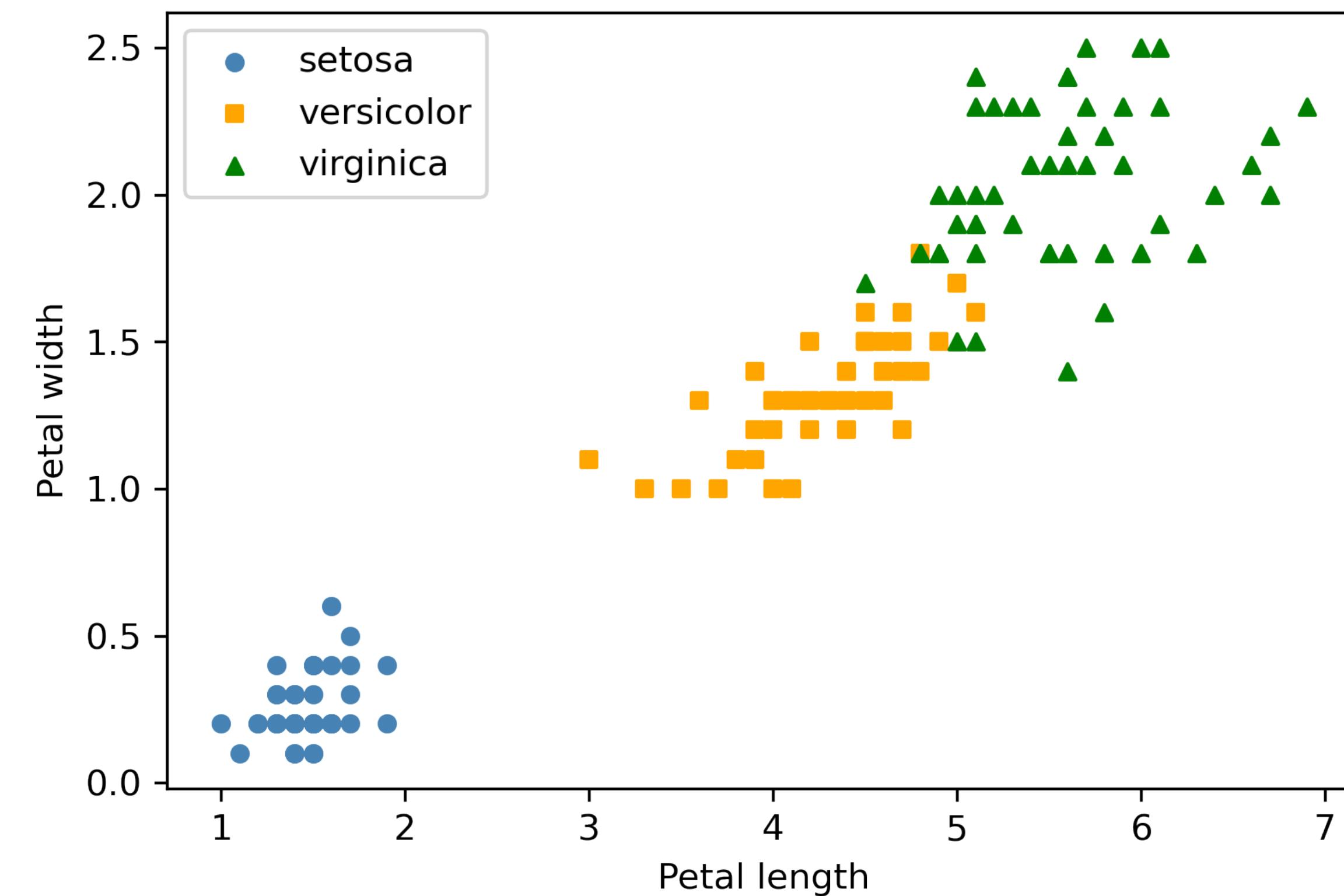
Decision Trees

Instabilität von Entscheidungsbäumen

Änderungen des Datensatzes

- ▶ Decision Trees
- ▶ Klassifikationsbäume

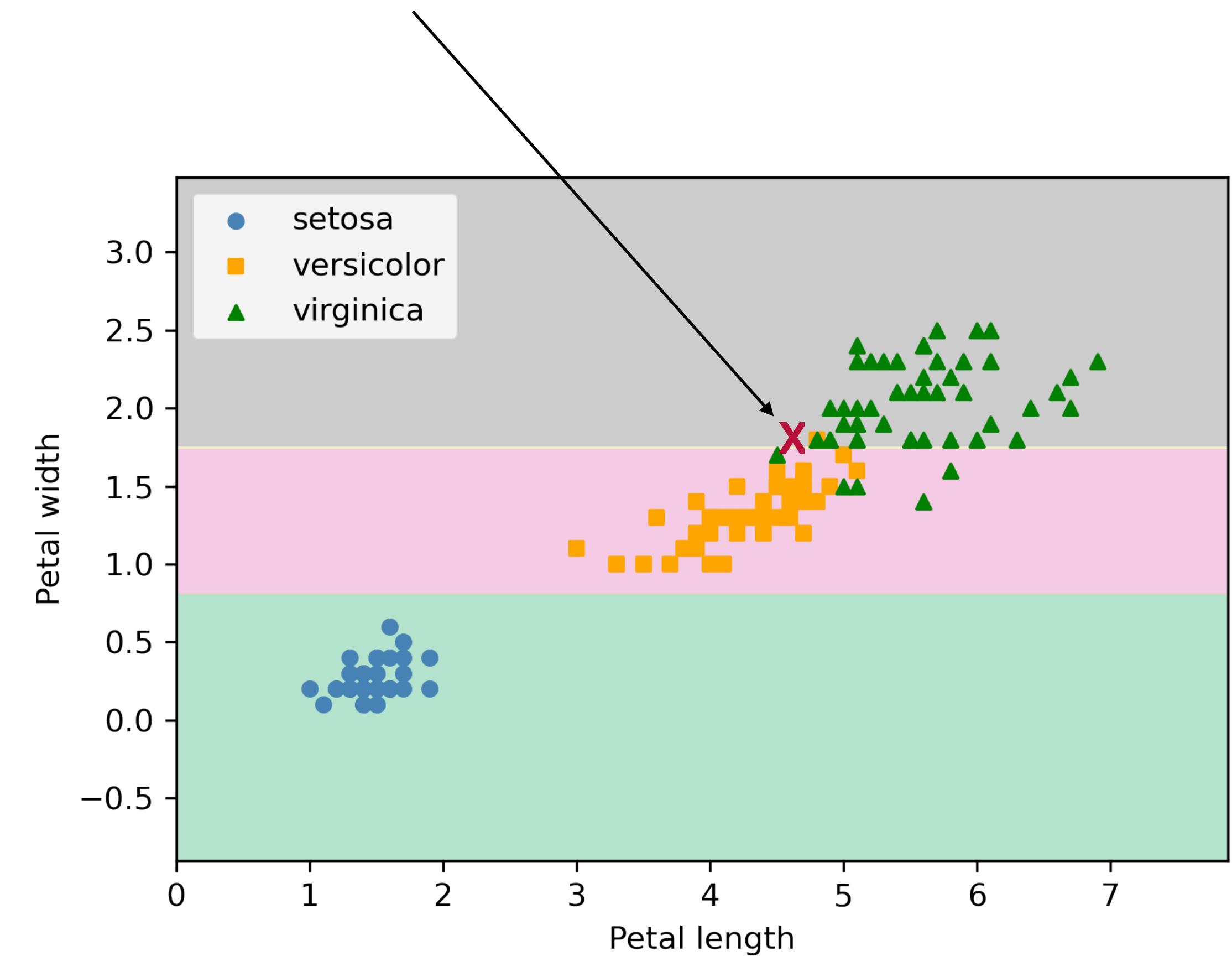
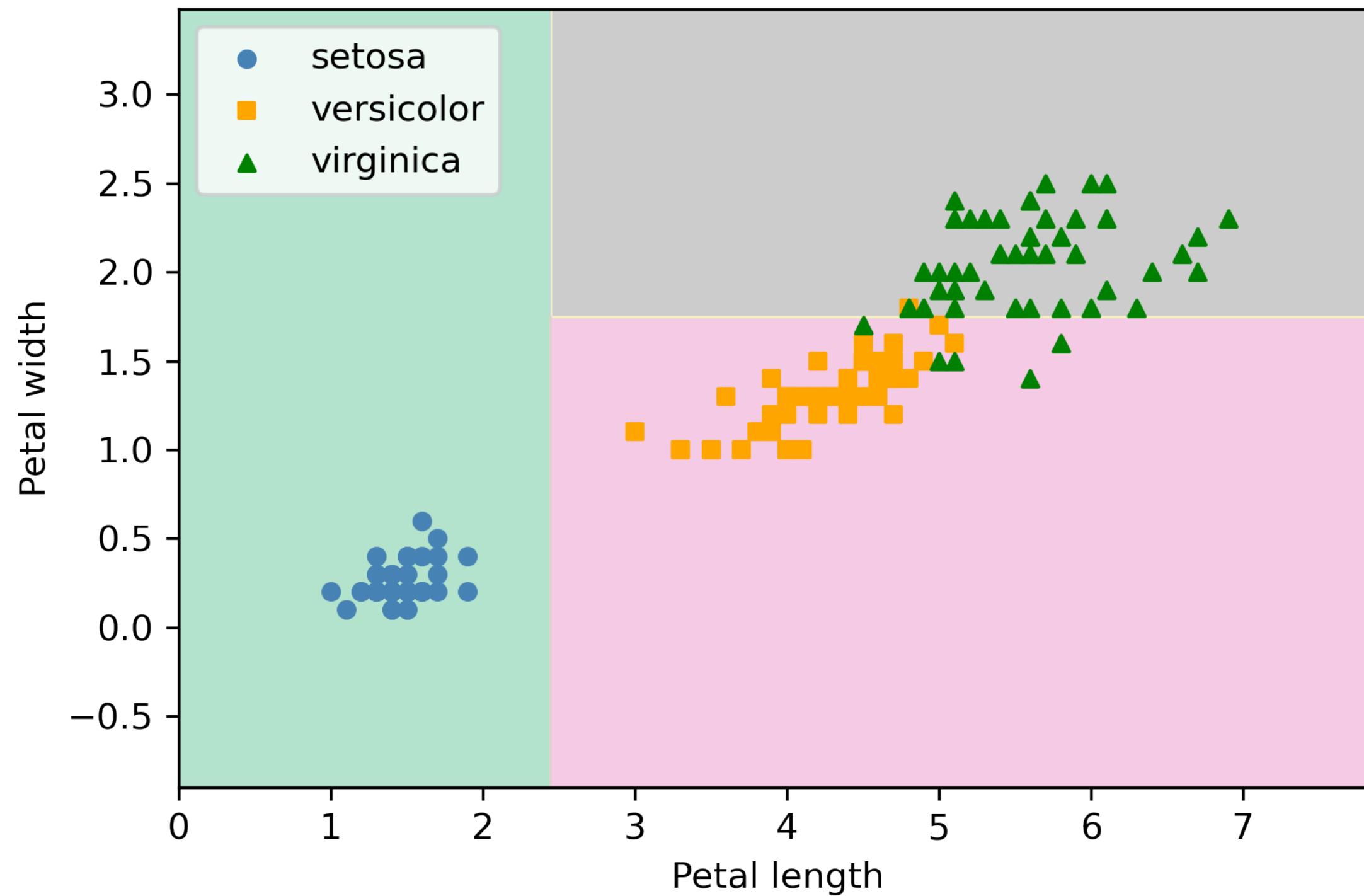
- Wir wollen Iris nun anhand der Features Blütenblattlänge und -Breite mithilfe eines Entscheidungsbäums klassifizieren.



Änderungen des Datensatzes

- ▶ Decision Trees
- ▶ Instabilität von Entscheidungsbäumen

- Entscheidungsbäume sind empfindlich gegenüber kleinen Änderungen des Datensatzes.
- Entfernen des Sample der Klasse Versicolor mit der höchsten Blütenblattlänge (Sample mit 4.8 cm Blütenblattlänge und 1.8 cm -Breite) führt bei Tiefe = 2 zu einer ganz anderen Partition, wie rechts dargestellt.



Rotationen

- ▶ Decision Trees
- ▶ Instabilität von Entscheidungsbäumen

- Da jede Entscheidungsgrenze eines Baums zu einer Achse orthogonal ist, reagieren Bäume empfindlich auf Rotationen.
- Beispiel Iris: Nach einer Rotation um 20° (rechte Spalte in der Abbildung) entsteht (Tiefe = 2) ein anderer Baum, der die Klassen jeweils linear voneinander trennt.
- Umgekehrt, sind Daten linear trennbar, so wird ein Baum die Daten nach einer Rotation (abhängig von der Stärke der Rotation) meist nicht mehr linear trennen, die Decision Boundary wird unnötig kompliziert

