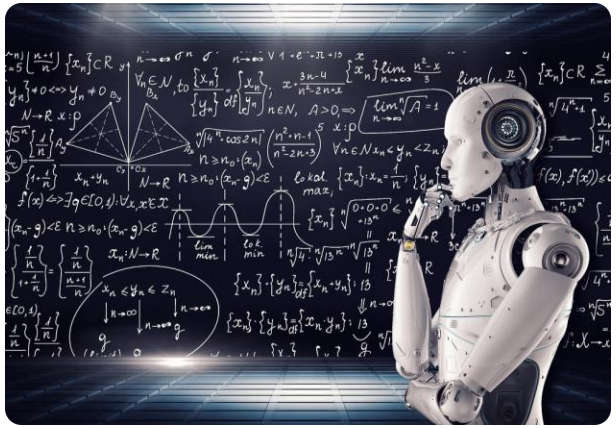
Informatik II Woche 12

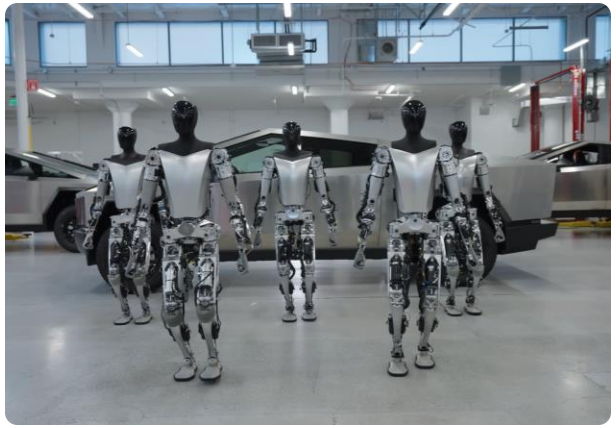
Intro ML, Decision Trees, Lineare Regression

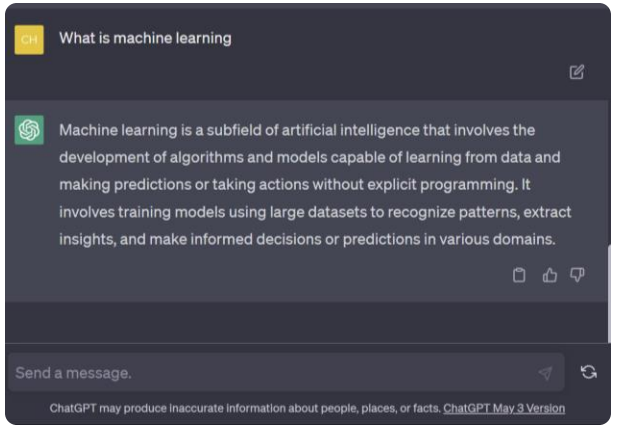
Website: n.ethz.ch/~kvaratharaja/

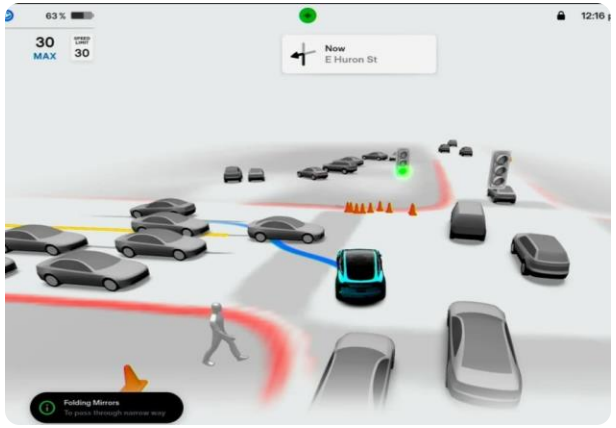
Die Slides basieren auf den offiziellen Übungsslides der Kurswebsite: https://lec.inf.ethz.ch/mavt/informatik2/2025/

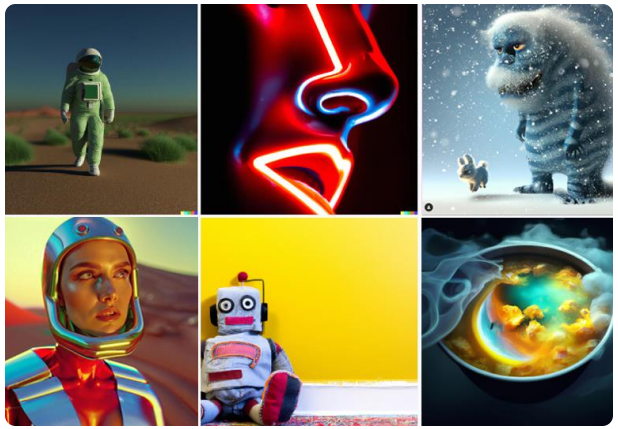
Machine Learning (ML)





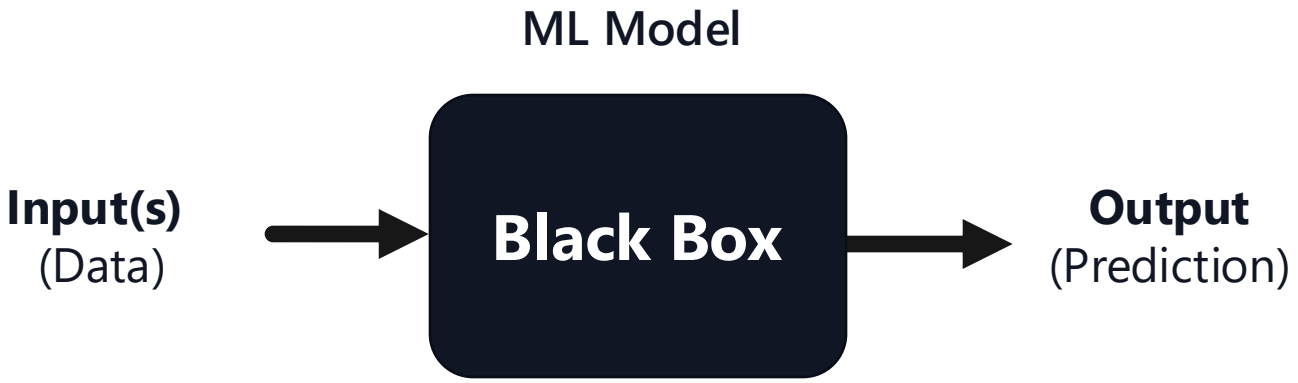






Machine Learning ist

- Entwicklung von Modellen, die ohne explizite Programmierung Muster in **Daten** lernen können
- Allgemeiner ausgedrückt: Abbildung einer Eingabe (Daten) auf eine Ausgabe (Vorhersage)



Motivation: Problem solving

Heute

1. ML-Begriffe
2. Decision Trees
3. Lineare Regression
4. Theoretische/Praktische Übungen
5. Hausaufgaben

1. ML-Begriffe

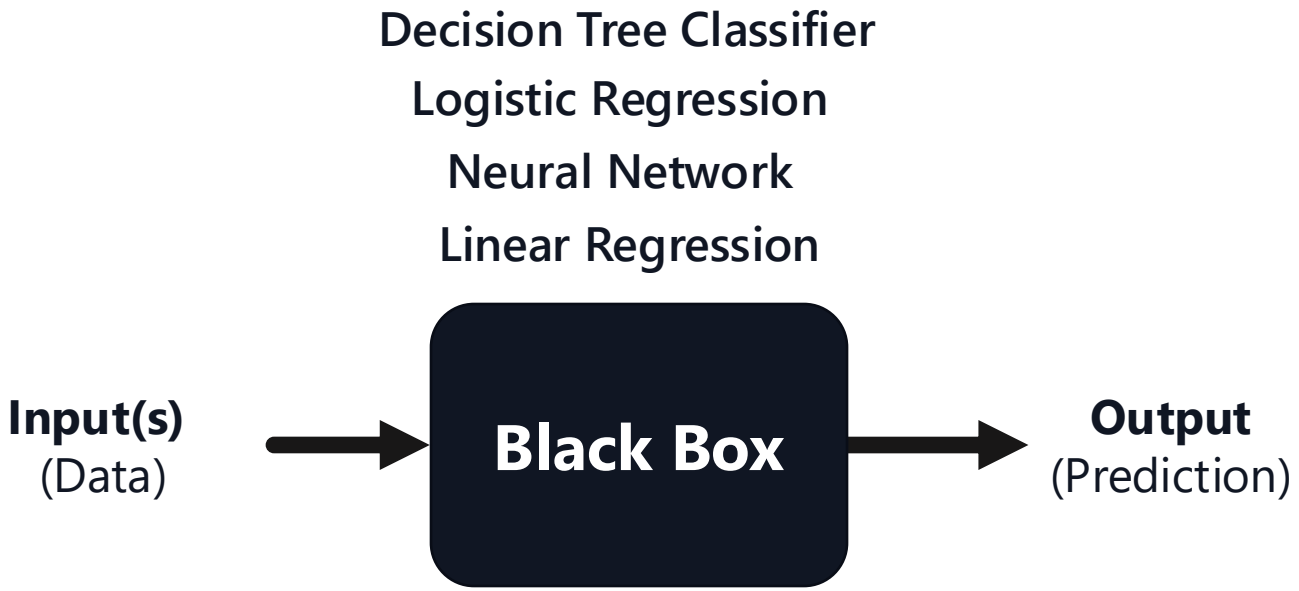
Begriffe und Vorgehen

Use-case von ML

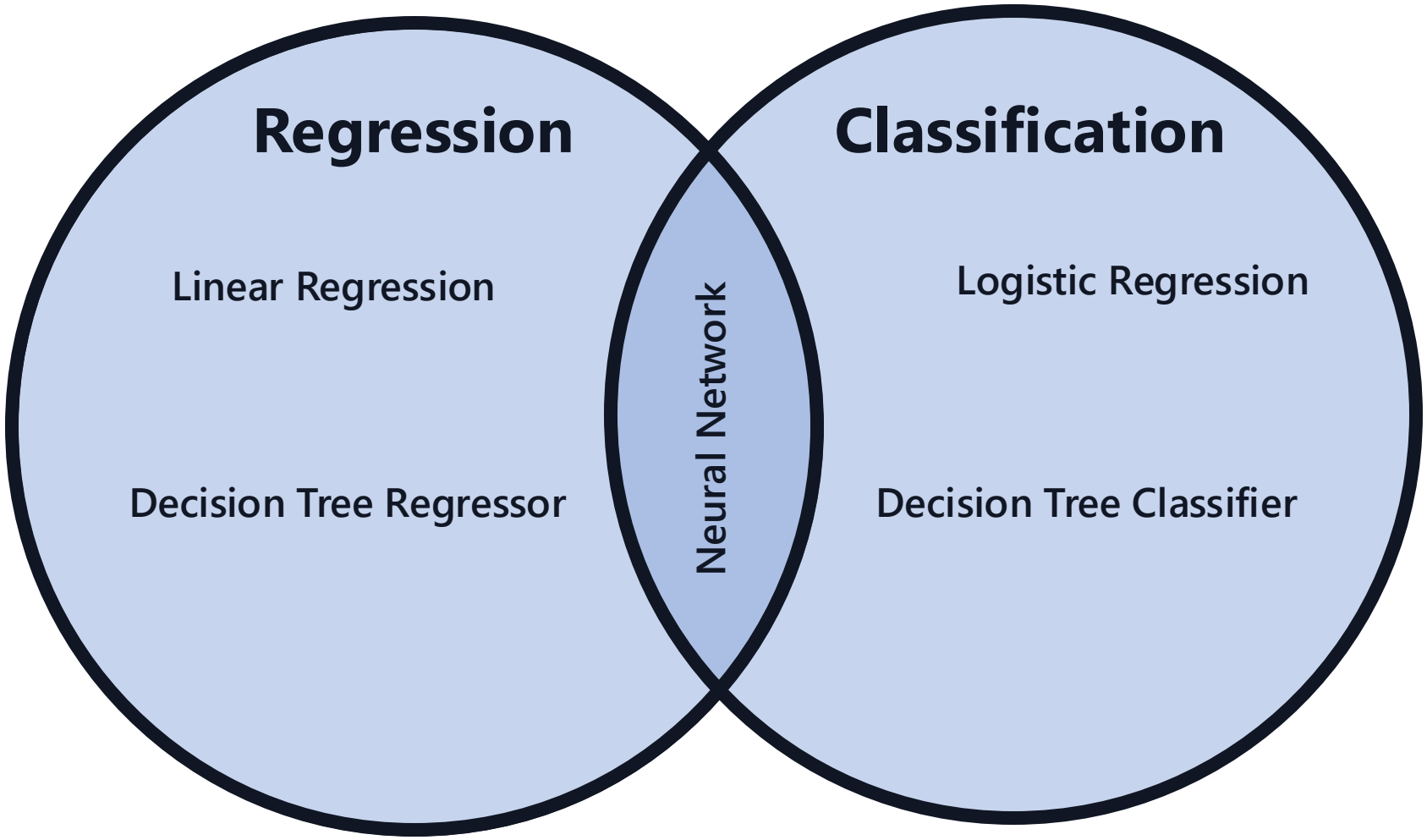
- Viele Daten
- Viel Rechenleistung
- Nicht in der Lage oder zu faul, eine mathematische Beziehung zwischen Eingaben und Ausgaben abzuleiten

Das Ziel

- Eine „Box“, die eine Eingabe annimmt und die gewünschte Ausgabe liefert
- Box durch einen ML-Algorithmus ersetzen:



Typen von (supervised) ML Algorithmen



Klassifikation vs. Regression

Klassifizierung: Welche **Kategorie** wird angesichts bestimmter Informationen vorhergesagt?

- Beispiel: Ein Haus hat 10 Zimmer, 3 Etagen und eine Fläche von 500 m^2. Wird es ein teures oder billiges Haus sein?

Regression: Wie lautet der vorhergesagte **Wert** bei bestimmten Informationen?

- Beispiel: Ein Haus hat 10 Zimmer, 3 Etagen und eine Fläche von 500 m^2. Wie viel kostet das Haus?

Quiz

Ein Student hat an 10 von 14 Informatik II-Übungseinheiten teilgenommen. Welche Note wird der Student bei der Basisprüfung erhalten?

Welche Art von ML-Algorithmus wird benötigt?

A. Classification

B. Regression

Quiz

Ein Student hat an 10 von 14 Informatik II-Übungseinheiten teilgenommen. Welche Note wird der Student bei der Basisprüfung erhalten?

Welche Art von ML-Algorithmus wird benötigt?

A. Classification

B. Regression

Vorgehen

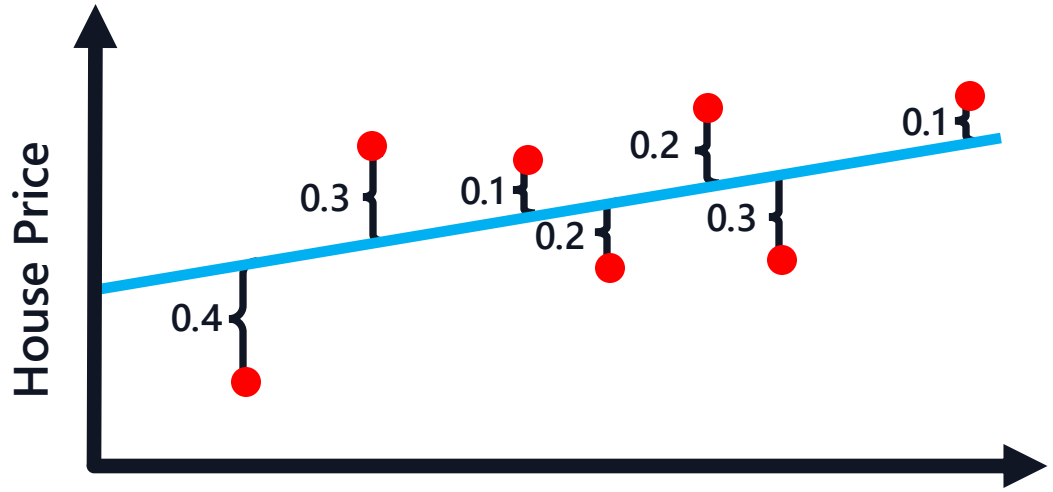
Nehmen wir an, du willst den Preis eines Hauses vorhersagen, wenn du einige Daten über das Haus hast:

- Wähle einen ML-Algorithmus.
- Trainiere ihn mit vorhandenen Daten.
- Validiere das Modell*.

* Modell = trained ML Algorithmus

ML Algorithmus trainieren

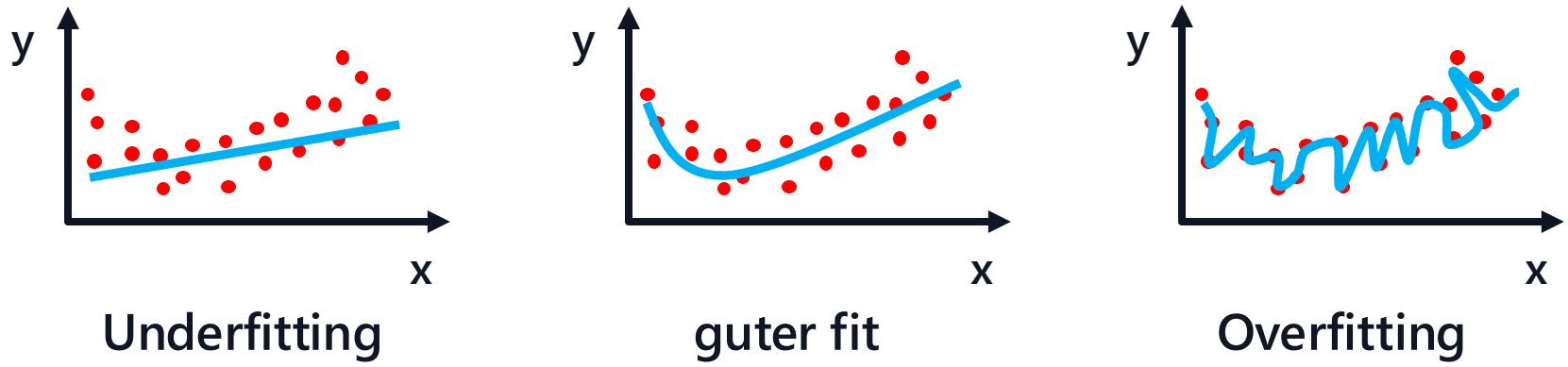
- **Im Wesentlichen:** wir füttern den Algorithmus mit Beispielen und er "lernt" das Muster
- **Hinter den Kulissen:** passt die Parameter an, bis eine **Verlustfunktion minimiert** ist.
- **Verlustfunktion:** eine Metrik, die misst, wie schlecht die Ausgabe ist
- Beispiel (Mean Squared Error Loss):



$$\text{Loss} = \frac{1}{7}(0.4^2 + 0.3^2 + 0.1^2 + 0.2^2 + 0.2^2 + 0.3^2 + 0.1^2)$$

Probleme

- **Underfitting**: Modell lernt das Muster zu wenig/schlecht.
  - Loss: **hoch** für training Daten, **hoch** für neue Daten
- **Overfitting**: Modell lernt die Trainingsbeispiel zu stark (auswendig)
  - Loss: **niedrig** für training Daten, **hoch** für neue Daten



Validation

- Wir messen, wie gut unser Modell ist, indem wir es an Beispielen testen, die es **noch nie gesehen hat**
- Die Daten müssen in Trainingsbeispiele und Testbeispiele aufgeteilt werden:

	Area / m^2	House Price / $
Training set	169	2'900'000
	250	5'000'000
	70	1'200'000
	10	200'000
Test set	73	1'500'000
	109	2'100'000

Quiz

Du hast gerade ein Modell trainiert. Du verwendest eine Verlustfunktion, um zu bewerten, wie gut dein Modell ist. Der Verlust für die Testmenge ist hoch, aber der Verlust für die Trainingsmenge ist niedrig.

Das Modell ist:

A. Overfitted

B. Underfitted

Quiz

Du hast gerade ein Modell trainiert. Du verwendest eine Verlustfunktion, um zu bewerten, wie gut dein Modell ist. Der Verlust für die Testmenge ist hoch, aber der Verlust für die Trainingsmenge ist niedrig.

Das Modell ist:

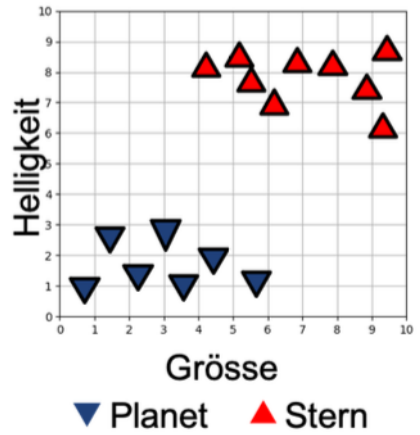
A. Overfitted

B. Underfitted

2. Decision Trees

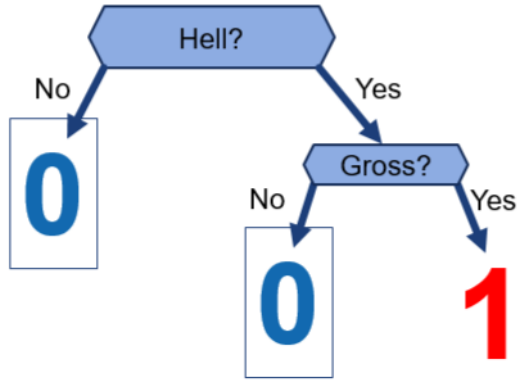
Decision Trees

Sterne und Planeten Datensatz



Sterne sind gross und hell.
Planeten sind kleiner und weniger hell.

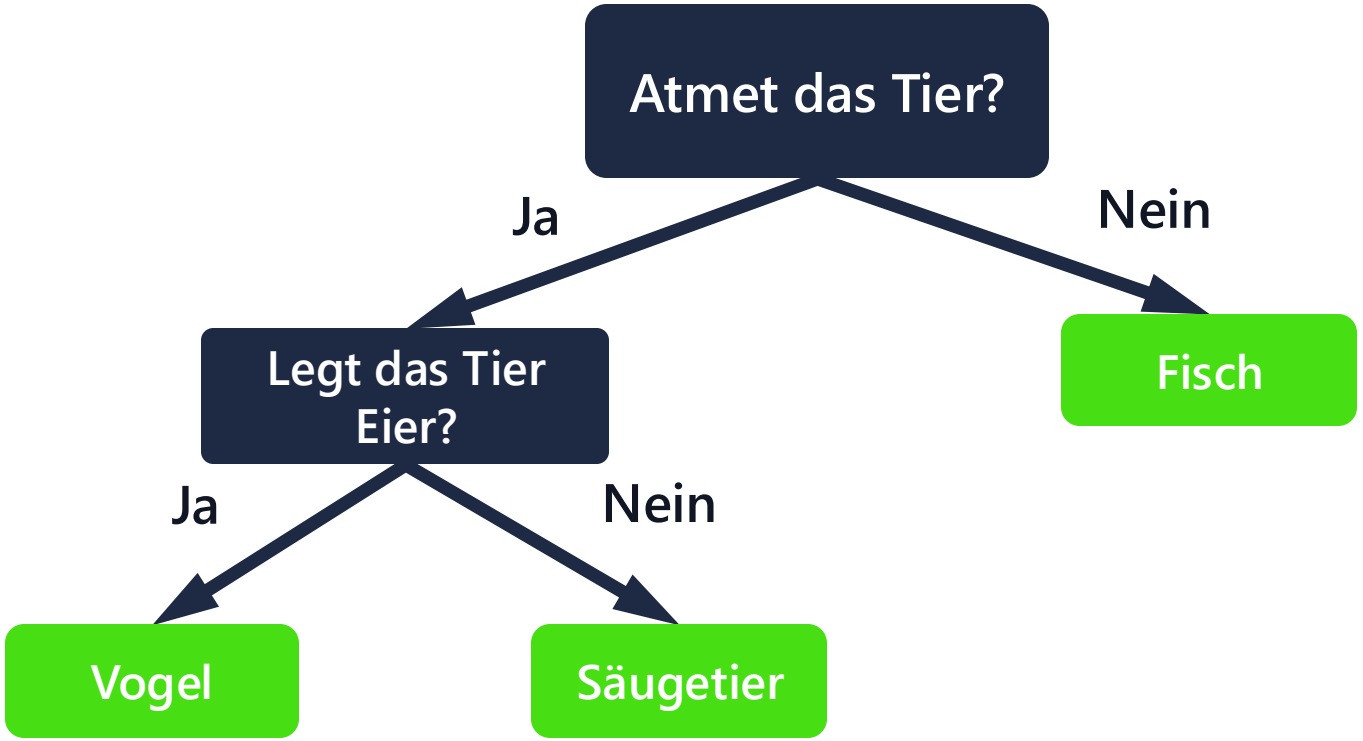
Entscheidungsbaum



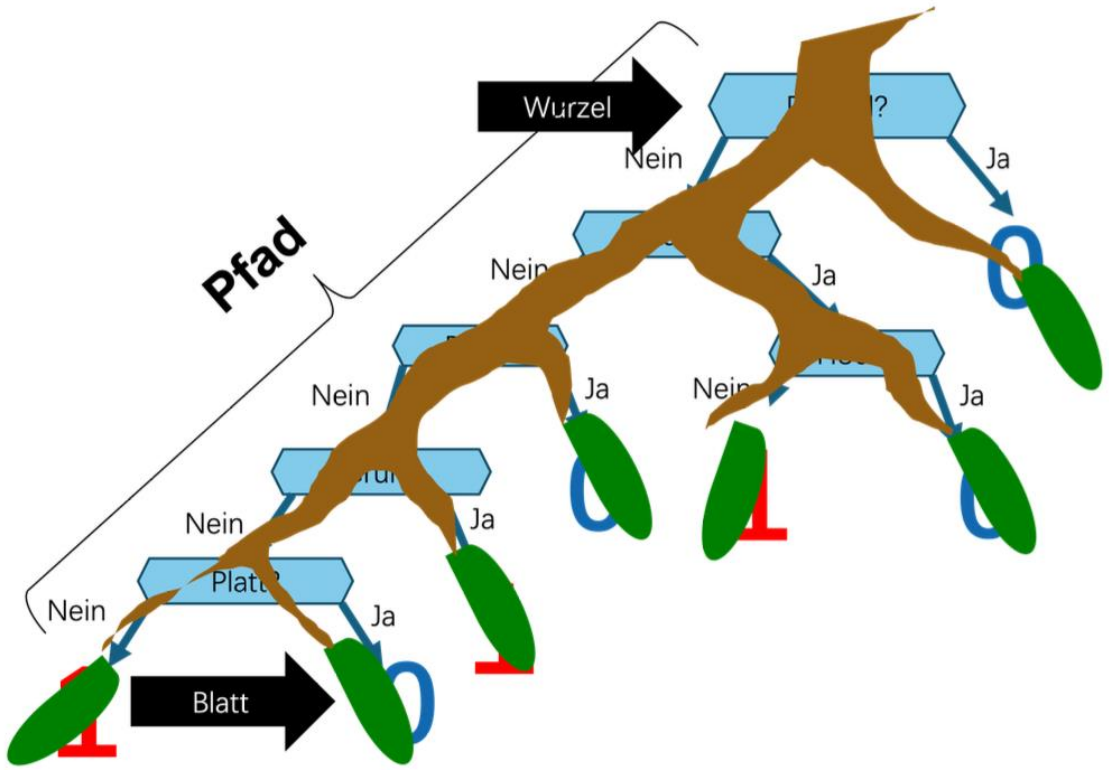
Planeten: 0, Sterne: 1
Tiefe des Baumes: Anzahl von Knoten im längsten Pfad (zwischen Wurzel und Blatt).

Decision Trees

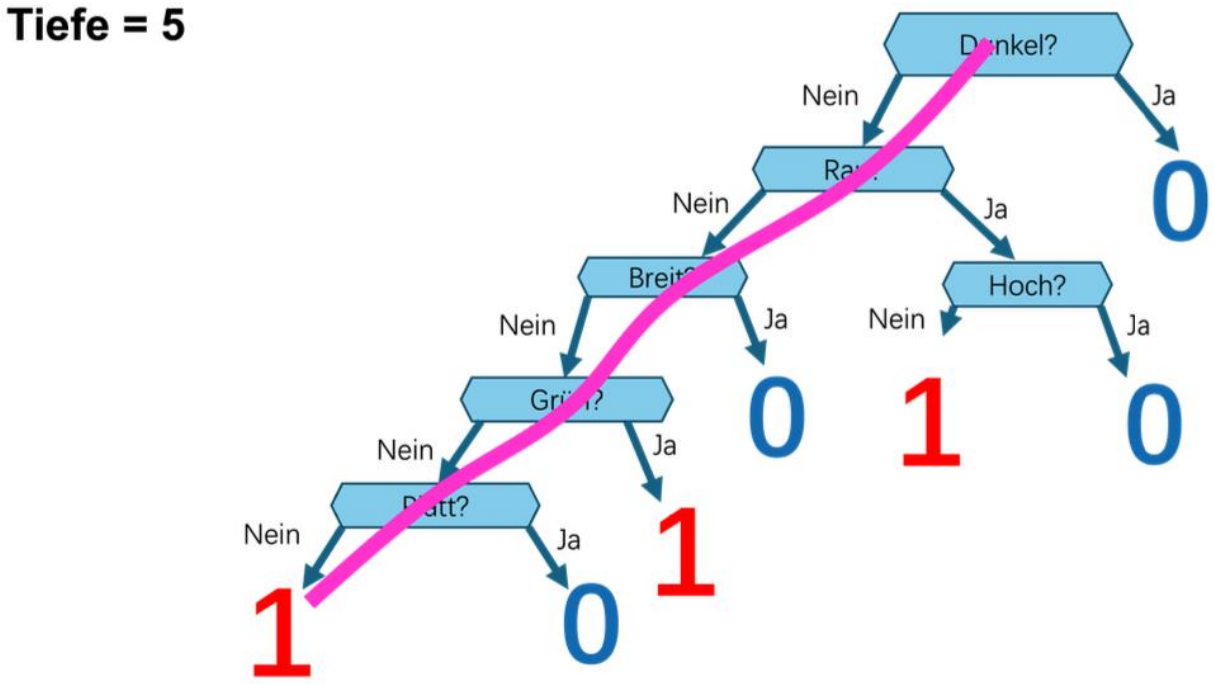
- Ein Baumtyp der verwendet, um Sachen zu kategorisieren.



Decision Trees



Decision Trees



Trainieren von Modellen in ML

- **Datensatz** Erstellen Sie einen Datensatz und teilen Sie ihn in den Trainingssatz D und den Validierungssatz D' auf.
- **Modell auswählen** Modell H: eine Menge von Entscheidungsfunktionen (z.B. Entscheidungsbäume der Tiefe ≤ 3, logistische Regression, oder spezifische NN Architektur)
- **Verlustfunktion (loss)** Funktion $L(D, f\ )$, wobei $f\ \in\ H$, die misst, wie gut f in D abschneidet.
- **Training** Finden einer Entscheidungsfunktion $f\ast \in\ H$, für die $L(D, f\ast)$ klein ist.
- **Validierung** Auswerten der Entscheidungsfunktion $f\ast$ an einem Dataset $D'\ \neq\ D$.

Initialisierung des Pseudo-Zufallszahlen-Generators

Zufälligkeit spielt beim maschinellen Lernen eine entscheidende Rolle.

- Wir haben Zufälligkeit bei der Datenerfassung, bei der Datenaufteilung (in Trainings-/Testsätze), bei den Trainingsmodellen usw.

Als Quelle der Zufälligkeit wird normalerweise ein **Pseudozufallszahlengenerator** verwendet.

- Es handelt sich um eine mathematische Funktion, die eine Folge nahezu zufälliger Zahlen generiert.
- Die Sequenz ist deterministisch und wird mit einer (oder mehreren) Anfangszahl(en) initialisiert, die seed genannt wird.

In Code Expert legen wir den Ausgangspunkt fest und machen alles deterministisch. Sie erhalten reproduzierbare Ergebnisse.

Datensatz erstellen

$D =$

Features oder Merkmale

Class label

Rot	Flackern	Gross	Hell	Stern
1	1	1	1	**1**
0	0	1	1	**1**
0	1	0	1	**0**
1	1	1	0	**0**
0	0	0	1	**0**
…	…	…	…	**…**
…	…	…	…	**…**

Aufteilen des Datensatzes

- Den Datensatz in einen Trainingsdatensatz D und einen Validierungsdatensatz D' aufteilen.

R	F	G	H	S
1	1	1	1	**1**
0	0	1	1	**1**
0	1	0	1	**0**
1	1	1	0	**0**
0	0	0	1	**0**
1	0	1	0	**0**
0	1	1	1	**1**
0	1	1	0	**0**

$D =$

R	F	G	H	S
1	1	1	1	**1**
0	0	1	1	**1**
0	1	0	1	**0**
1	1	1	0	**0**
0	0	0	1	**0**

$D' =$

R	F	G	H	S
1	0	1	0	**0**
0	1	1	1	**1**
0	1	1	0	**0**

Modell auswählen

- Modell $\mathcal{H}$: eine Menge von Entscheidungsfunktionen

Ein Beispiel eines Modells ist die Menge aller Bäume mit Tiefe 3

$\mathcal{H} =$ 

Verlustfunktion

- Funktion $L(D, f)$, wobei $f \in H$, die misst, wie gut f in D abschneidet.
- Normalerweise benutzt man die 0/1 Verlustfunktion für das Trainieren von Bäumen.

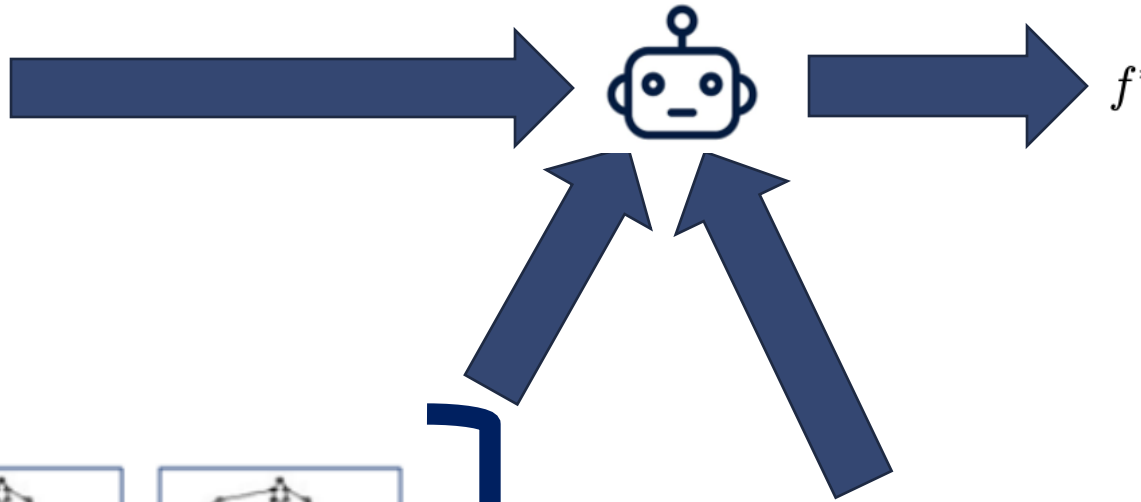
$$L(D, f) = \frac{\#\text{Beispiele in } D\text{, die von } f \text{ falsch klassifiziert wurden}}{\#\text{Beispiele in } D}$$

Trainieren

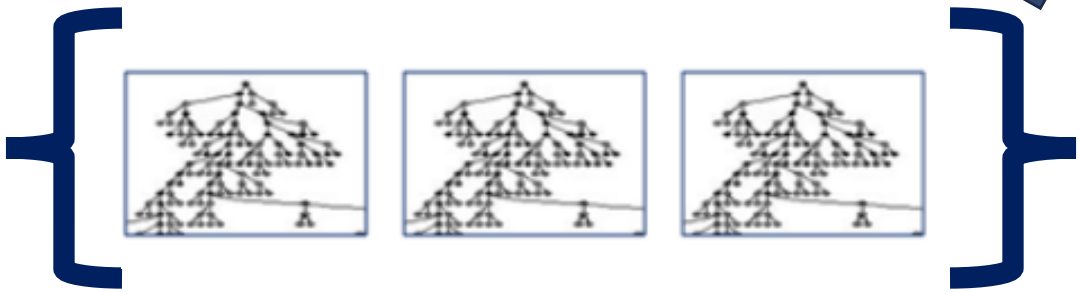
Finden Sie einen Schätzer $f* \in H$, für die der Wert von $L(D, f*)$ niedrig ist.

$D =$

R	F	G	H	S
1	1	1	1	**1**
0	0	1	1	**1**
0	1	0	1	**0**
1	1	1	0	**0**
0	0	0	1	**0**



$\mathcal{H} =$



Validieren

- Der Schätzer $f*$ auf dem Validierungsdatensatz $D' \neq D$ auswerten.

R	F	G	H	S
1	1	1	1	**1**
0	0	1	1	**1**
0	1	0	1	**0**
1	1	1	0	**0**
0	0	0	1	**0**
1	0	1	0	**0**
0	1	1	1	**1**
0	1	1	0	**0**

$D =$

R	F	G	H	S
1	1	1	1	**1**
0	0	1	1	**1**
0	1	0	1	**0**
1	1	1	0	**0**
0	0	0	1	**0**

$D' =$

R	F	G	H	S
1	0	1	0	**0**
0	1	1	1	**1**
0	1	1	0	**0**

In Python:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

#split data into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state = 42)

tree = DecisionTreeClassifier() #select model
tree.fit(X_train,y_train) #train model

y_pred = tree.predict(X_test)
validation_score = accuracy_score(y_test, y_pred) #validate model
```

Verwechslungsmatrix

- Eine Verwirrungsmatrix ist eine Tabelle mit der Verteilung der Klassifikatorleistung auf die Daten.
- Es handelt sich um eine $N \times N$ -Matrix, die zur Bewertung der Leistung eines Klassifizierungsmodells verwendet wird.

		Predicted	
		Ungiftig	Giftig
True Label	Ungiftig	True Positive (TP)	False Negative (FN)
	Giftig	False Positive (FP)	True Negative (TN)

Balanced Accuray

- Die "Balanced Accuracy" ist die durchschnittliche Genauigkeit aller Klassen.
- Sie kann als Durchschnitt der Diagonalen der normalisierten Verwirrungsmatrix berechnet werden.
- Dies ist hilfreich beim Umgang mit unausgeglichenen Daten.

		Predicted	
		Ungiftig	Giftig
True Label	Ungiftig	20	70
	Giftig	30	5000

Balanced Accuracy

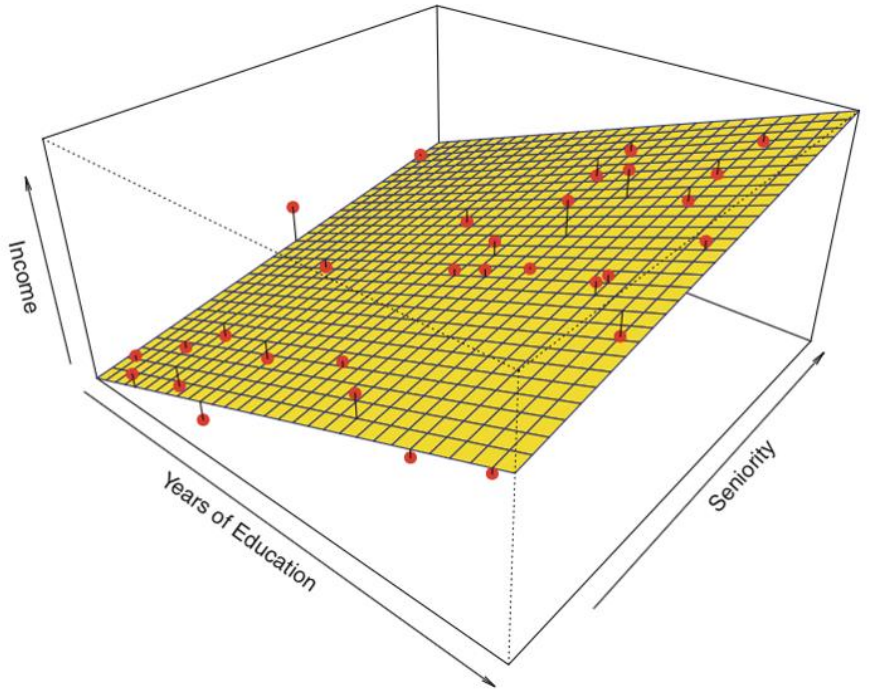
		Predicted	
		Ungiftig	Giftig
True Label	Ungiftig	20	70
	Giftig	30	5000

- Berechnen wir die Genauigkeit dieser Vorhersage: $\frac{TP+TN}{TP+FN+FP+TN} \approx 98,05\%.$
- Dieses Ergebnis ist beeindruckend, allerdings wird die Ungiftig-Spalte in der Vorhersage selbst nicht richtig behandelt.
- Verwenden Sie die "Balanced Accuracy": $\frac{1}{2}(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}) \approx 60,80\%$
- Dadurch ist die Punktzahl niedriger als von der Genauigkeit vorhergesagt, da beide Klassen das gleiche Gewicht erhalten.

3. Lineare Regression

Lineare Regression

- Eine Art von ML-Algorithmus, der Linien/Ebenen/Hyperplanes mit der besten Anpassung erstellt
- Mathematisch: $\mathrm{y} = \boldsymbol{w}^T \boldsymbol{x} + b$ oder $\mathrm{y} = w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + b$
  - y: output
  - **w:** weight vektor
  - **x**: feature vektor/input vektor
  - b: bias



Lineare Regression

Regression: Finden einer Beziehung (Funktion) zwischen einer abhängigen und unabhängigen Variable.

Lineare Regression: Finden einer Linearen Abbildung, die den Vektor **x** von unabhängigen Variablen auf eine abhängige Variable y abbildet.

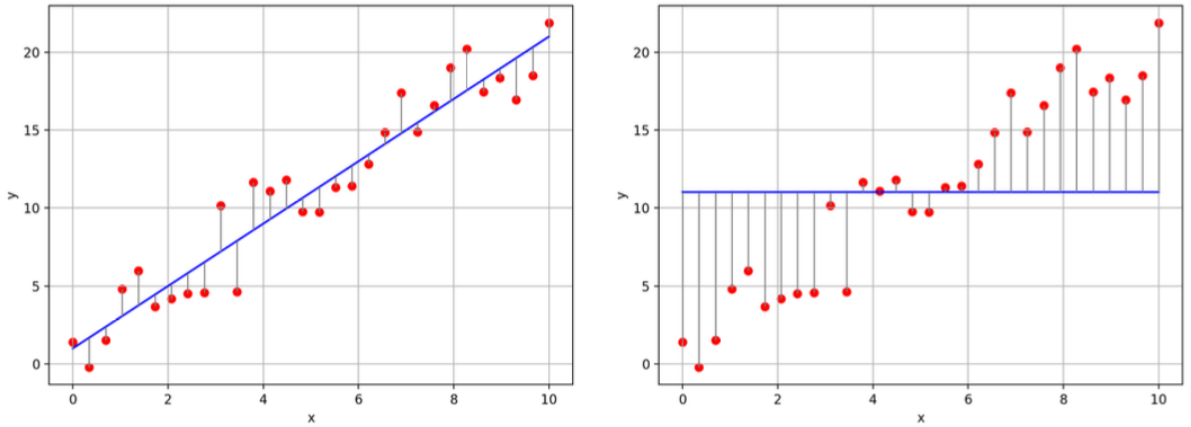
- **Datensatz**: Jeder Datenpunkt besteht aus einem Vektor **x** und einem Skalar y.
- **Modell**: Die Menge der linearen Abbildungen $\mathbb{R}^n \to \mathbb{R}$. Wir finden den Vektor **w**: $\boldsymbol{w} \cdot \boldsymbol{x} = y$.
- **Verlustfunktion**: Die Summe der Quadrate: $\sum_{\mathbf{x} \in \mathrm{D}} (\boldsymbol{w} \cdot \boldsymbol{x} - y)^2$
- **Trainieren**: Mit der Methode der kleinsten Quadrate (nicht klausurrelevant).
- **Validierung**: Messen der Summe der Quadrate auf dem Validierungsdatensatz D'.

In Python:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error #or use sth like r2_score

#split data into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state = 42)

model = LinearRegression() #select model
model.fit(X_train,y_train) #train model

y_pred = model.predict(X_test)
validation_score = mean_squared_error(y_test, y_pred) #validate model
```

R2-Score

- Wie gut ist ein Schätzer verglichen mit dem Mittelwertschätzer?



$$R2 = 1 - \frac{MSE(D, \text{Schätzer})}{MSE(D, \text{Mittelwertschätzer})} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

- $0 < R2 < 1$: Besser als Mittelwertschätzer
- $-\infty < R2 < 0$: Schlechter als Mittelwertschätzer

Gini Index

Wie unrein sind die Partitionen eines Entscheidungbaums?

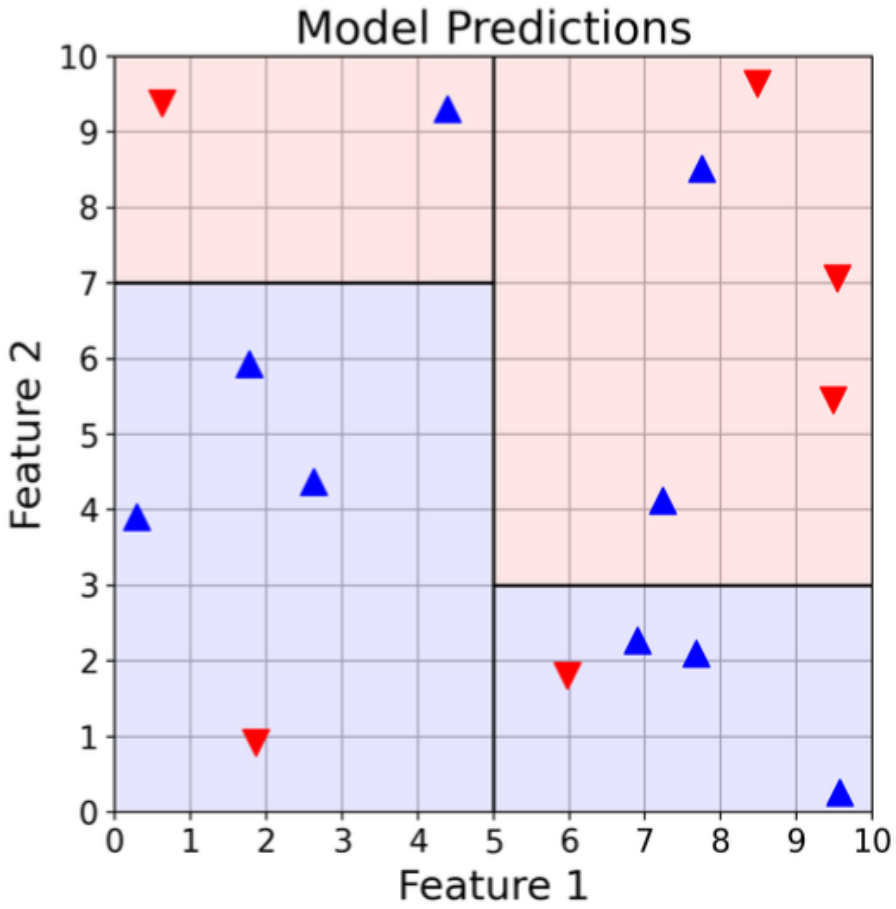
- Einzelne Partition:

$$G(D, \text{Part}_m) = \sum_k \hat{p}_{m,k} \cdot (1 - \hat{p}_{m,k})$$

$\hat{p}_{m,k}$: Anteil der Klasse k in Part_m.

- Insgesamt:

$$G(D, f) = \sum_m \frac{|\text{Part}_m|}{|\text{Part}|} \cdot G(D, \text{Part}_m)$$



Gini Index

$$G(D, \text{Part}_m) = \sum_k \hat{p}_{m,k} \cdot (1 - \hat{p}_{m,k})$$

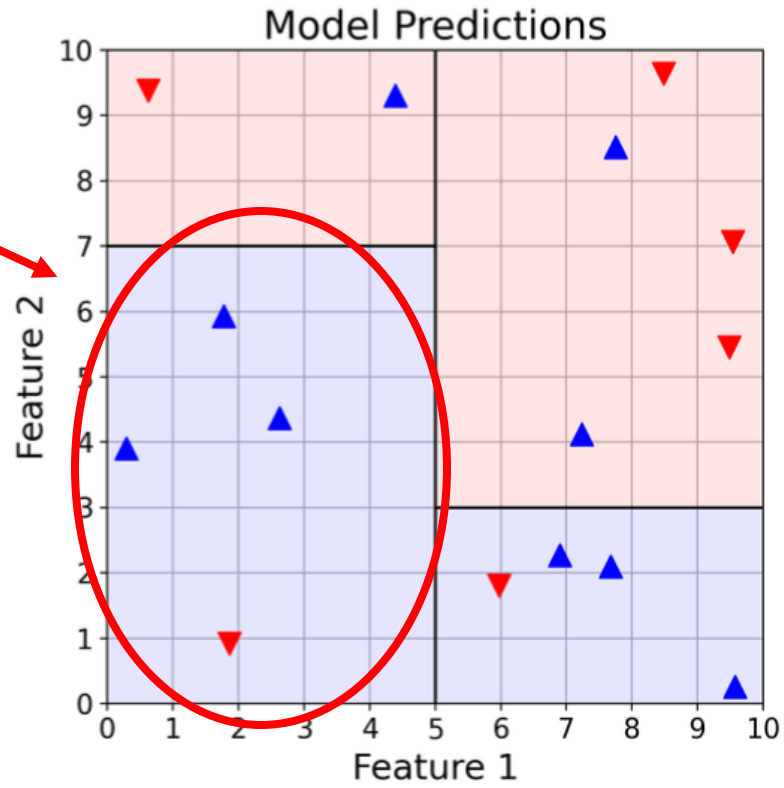
$\hat{p}_{m,k}$: Anteil der Klasse k in Part_m.

Beispiel:

Part 1 links unten:

- $\hat{p}_{1,\,blau} = \frac{3}{4}$
- $\hat{p}_{1,\,rot} = \frac{1}{4}$

$\rightarrow G(D, Part_1) = \frac{3}{4} * \frac{1}{4} + \frac{1}{4} * \frac{3}{4} = \frac{3}{8}$

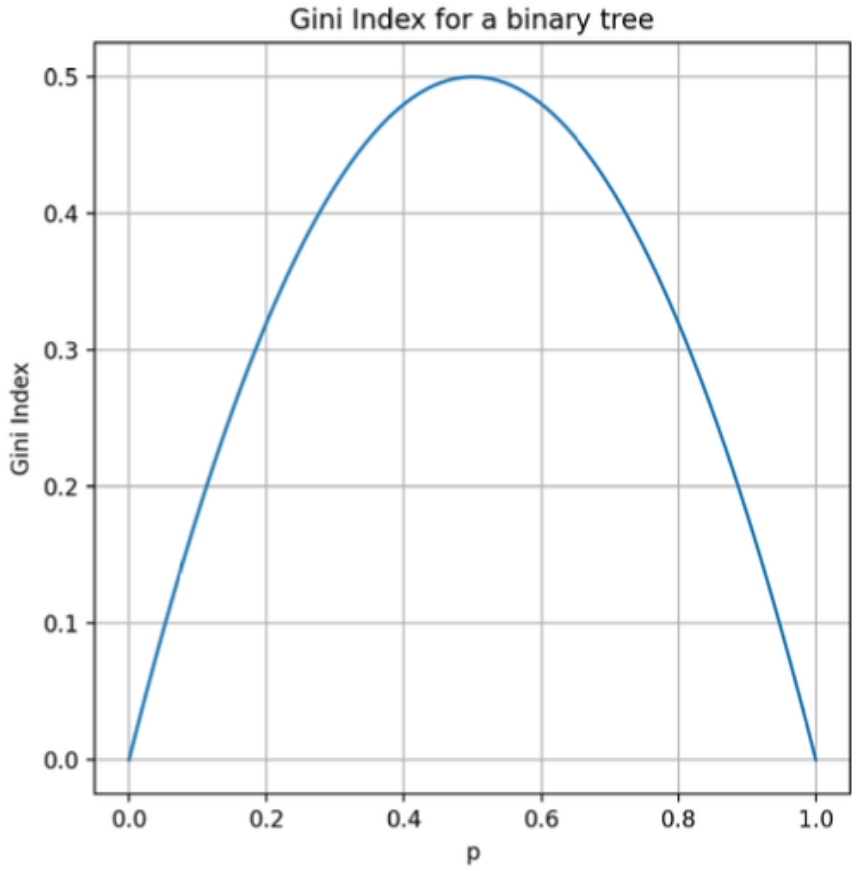


Gini Index

- $G(D, f) \in [0,1)$
- 0 gut (perfekt rein), 1: schlecht (max. unrein)
- Beispiel: In einer Partition sind drei Klassen gleich oft vertreten, d.h. $\hat{p}_1 = \hat{p}_2 = \hat{p}_3 = \frac{1}{3}$.

Der Gini Index ist $3 * \frac{1}{3} * \left(1 - \frac{1}{3}\right) = \frac{2}{3}$

- Bei zwei Klassen liegt das Maximum bei 0.5, siehe Grafik.



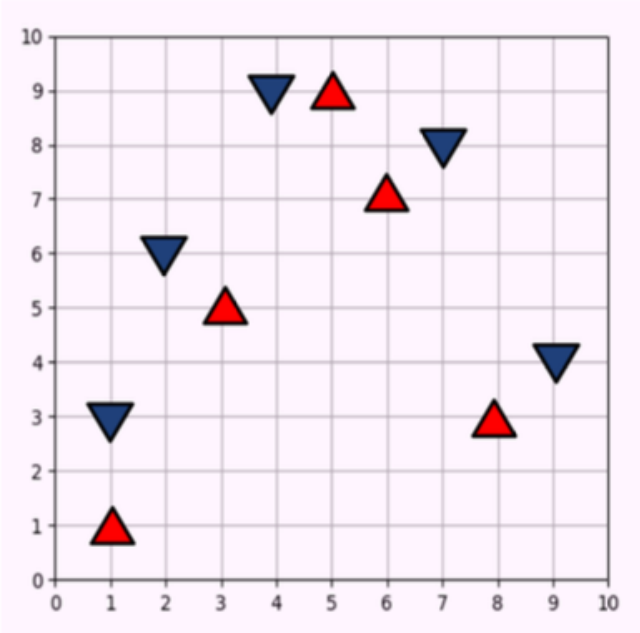
4. Theoretische/Praktische Übungen

Quiz 1:

Gegeben sei der Trainingsdatensatz D für ein dreieckiges Muster im Bild. Finde einen Baum mit der Tiefe 3, der alle Punkte im obigen Datensatz richtig klassifiziert.

$\mathcal{D} =$

x-Koord	y-Koord	Klasse
1	1	1
3	5	1
5	9	1
6	7	1
8	3	1
1	3	0
2	6	0
4	9	0
7	8	0
9	4	0

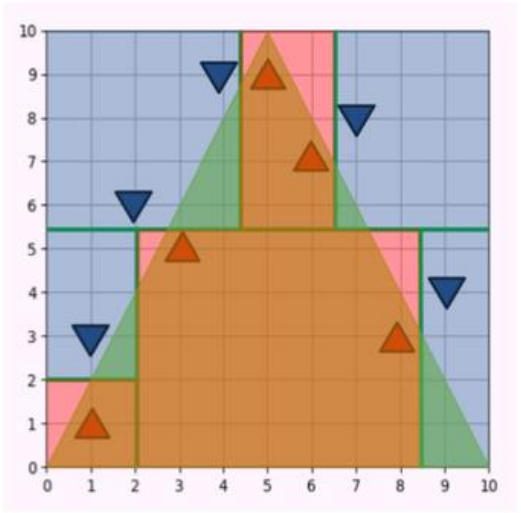


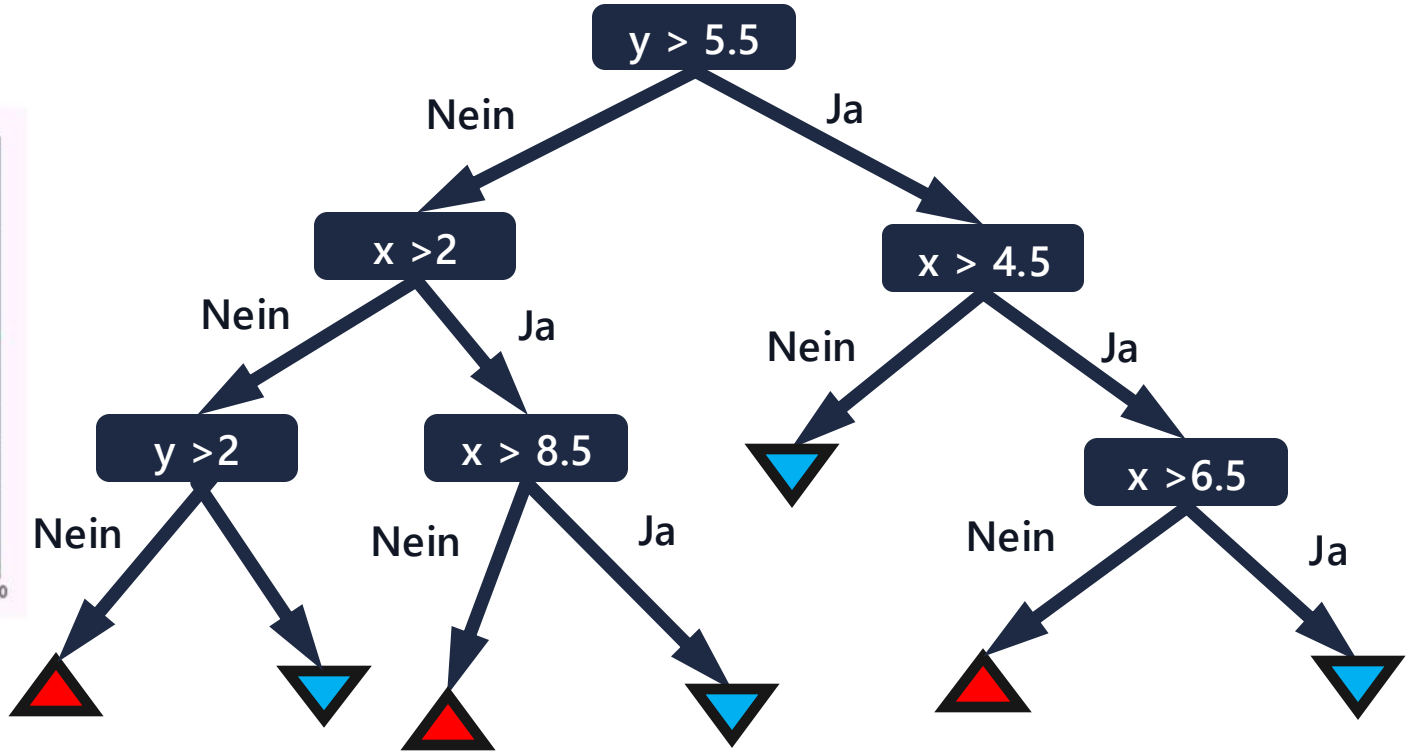
Quiz 1: Lösung

Gegeben sei der Trainingsdatensatz D für ein dreieckiges Muster im Bild. Finde einen Baum mit der Tiefe 3, der alle Punkte im obigen Datensatz richtig klassifiziert.

$\mathcal{D} =$

x-Koord	y-Koord	Klasse
1	1	1
3	5	1
5	9	1
6	7	1
8	3	1
1	3	0
2	6	0
4	9	0
7	8	0
9	4	0





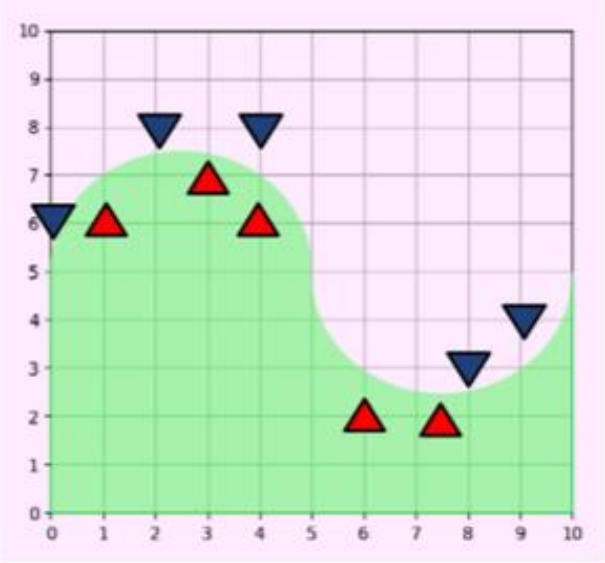
Quiz 2

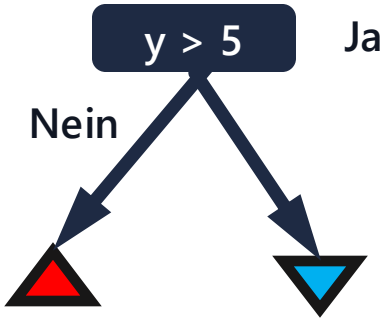
Gegeben seien 3 Decision Trees T1, T2, T3 mit Tiefe 1,2 und 6 und ein Validierungssatz D', der dem Muster im Bild entspricht.

- Berechne für jeden wie viele Punkte in D' durch den Baum **inkorrekt** initialisiert werden. → Entscheide welcher Baum der Beste ist.

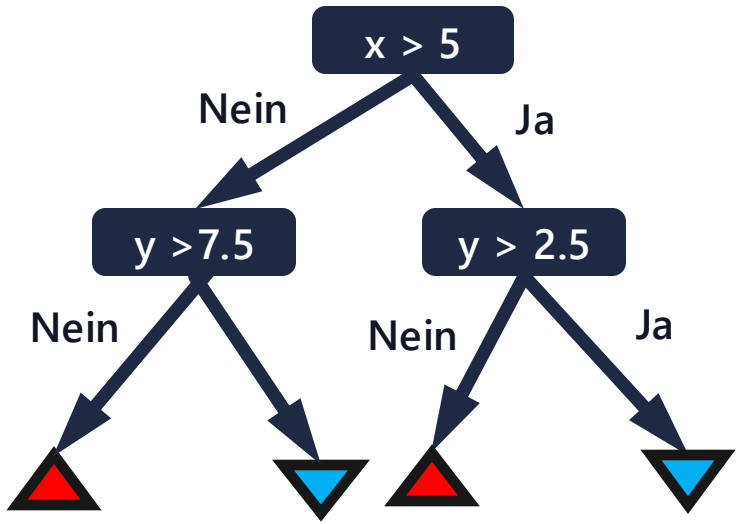
$D' =$

x-Koord	y-Koord	Klasse
1	6	1
3	7	1
4	6	1
6	2	1
7.5	2	1
0	6	0
2	8	0
4	8	0
8	3	0
9	4	0

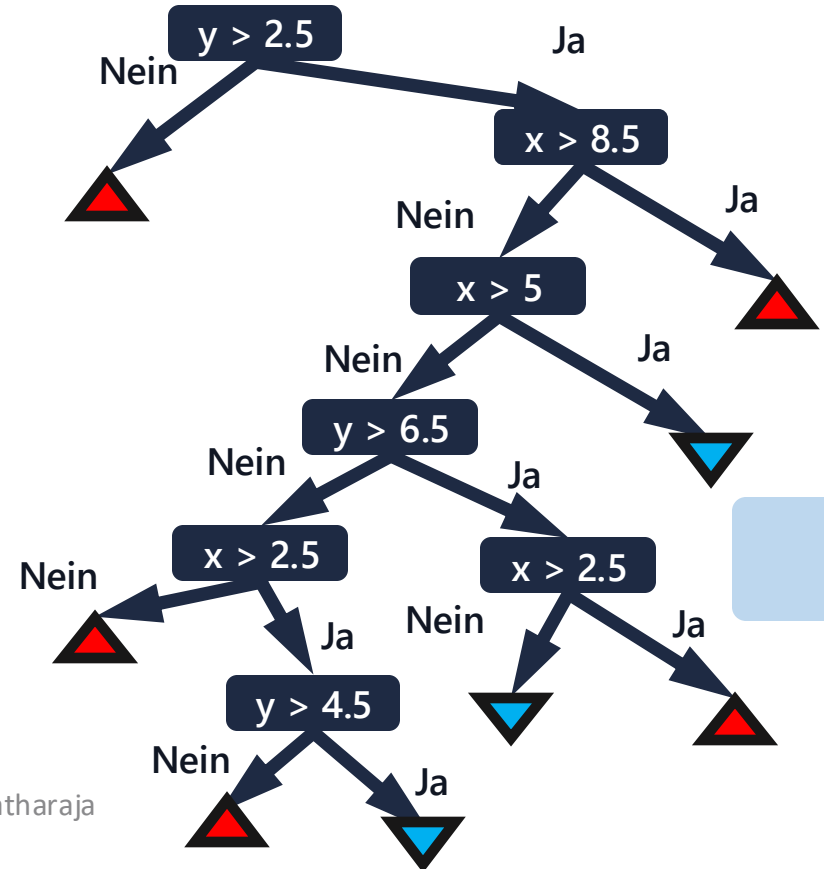




A



B



C

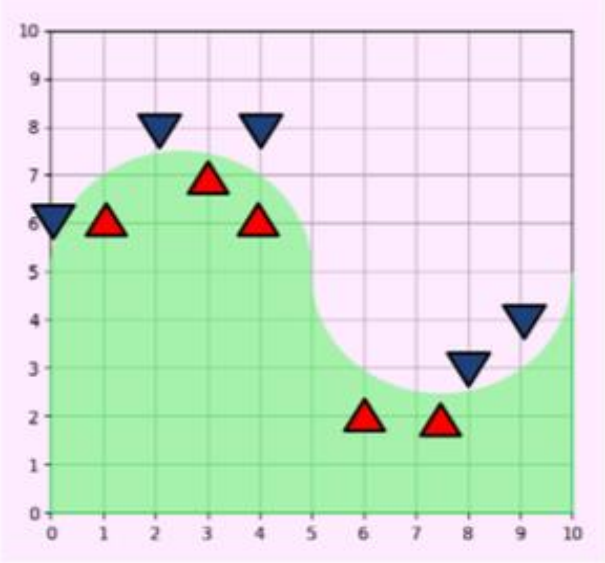
Quiz 2: Lösung

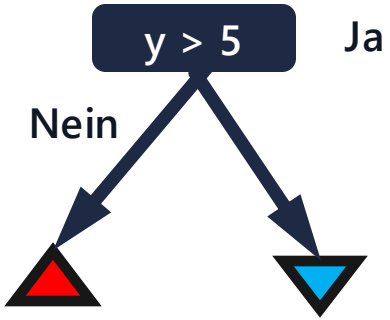
Gegeben seien 3 Decision Trees T1, T2, T3 mit Tiefe 1,2 und 6 und ein Validierungssatz D', der dem Muster im Bild entspricht.

- Berechne für jeden wie viele Punkte in D' durch den Baum **inkorrekt** initialisiert werden. → Entscheide welcher Baum der Beste ist.

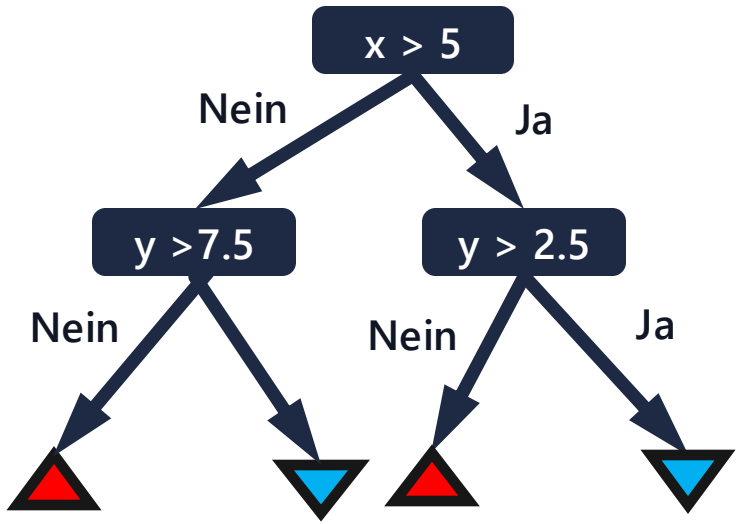
$D' =$

x-Koord	y-Koord	Klasse
1	6	1
3	7	1
4	6	1
6	2	1
7.5	2	1
0	6	0
2	8	0
4	8	0
8	3	0
9	4	0

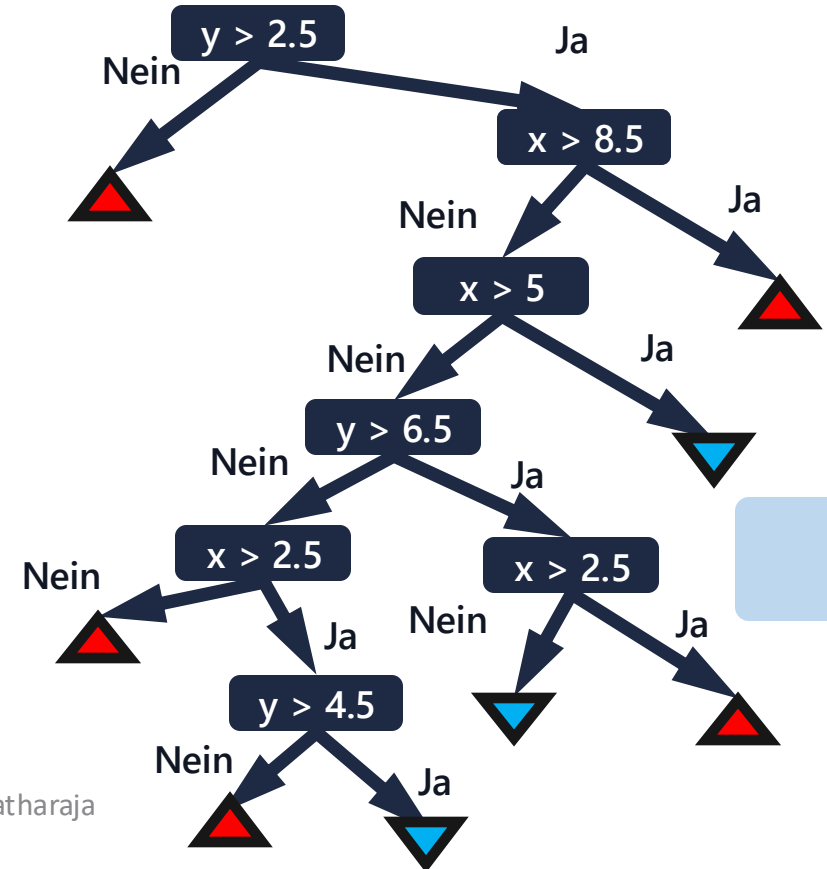




A



B

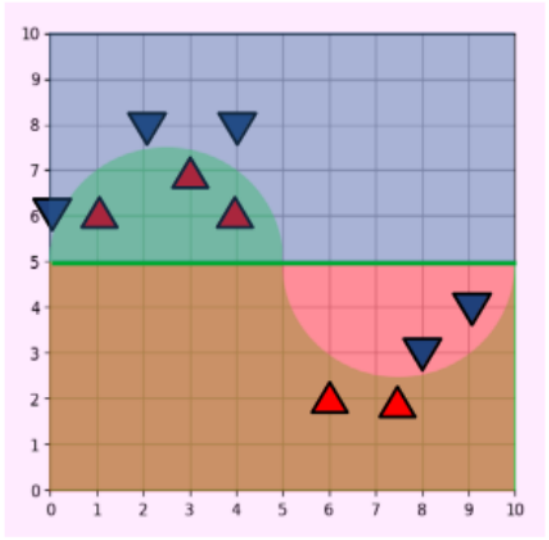


C

Quiz 2: Erklärung

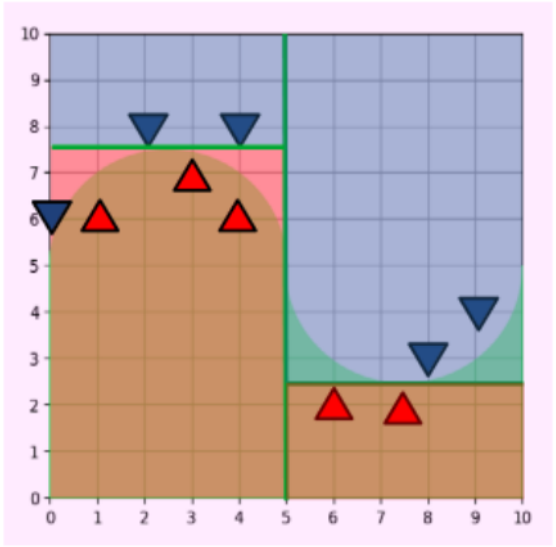
Die folgenden Bilder zeigen die Schätzungen der drei Bäume:

T1:



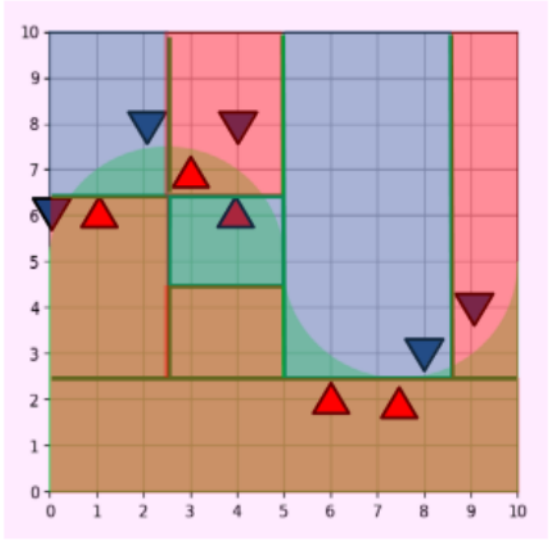
$L(D', T1) = \frac{5}{10} = 0.5$

T2:



$L(D', T2) = \frac{1}{10} = 0.1$

T3:



$L(D', T3) = \frac{5}{10} = 0.5$

Der zweite Baum ist der Beste.

Übung 1: House prices (Regression)

In dieser Übung trainieren wir eine Funktion mithilfe eines fiktiven Datensatzes, die den Preis eines Hauses anhand von seiner Fläche schätzen kann.
Machen Sie die folgenden Übungen in Code Expert:

1. Lesen Sie den Datensatz data.csv mit pandas ein.
2. Teilen Sie den Datensatz in einen Trainingsdatensatz und einen Testdatensatz auf.
3. Trainieren Sie ein lineares Regressionsmodel mit dem Trainingsdatensatz.
4. Verwenden Sie das Modell, um die Hauspreise vorherzusagen. Bewerten Sie die Qualität der Vorhersage mit dem R2-Wert.
5. Lesen Sie X_final.csv mit pandas ein, verwenden Sie das Modell, um die Preise vorherzusagen und geben Sie die Vorhersagen zurück.
6. Die Vorhersagen werden von Code Expert automatisch bewertet.

Detaillierte Anweisungen befinden sich auf Code Expert.

Übung 2: Iris (Klassifikation)

Iris Datensatz: klassischer Datensatz aus dem Jahr 1936. Er enthält Messungen von Blütenblattern (petal) und Kelchblättern (sepal) von 150 Blüten, die zu einer von 3 Arten von Iris (*Iris setosa*, *Iris versicolor*, *Iris virginica*) gehören.

Machen Sie die folgenden Übungen in Code Expert:

1. Lesen Sie den Datensatz data.csv mit pandas ein.
2. Teilen Sie den Datensatz in einen Trainingsdatensatz und einen Testdatensatz auf.
3. Trainieren Sie einen Entscheidungsbaum für die Klassifikation der Arten von Iris aus den Messungen.
4. Verwenden Sie den Baum, um die Blumen im Testdatensatz zu klassifizieren.
5. Lesen Sie X_final.csv mit pandas ein, verwenden Sie den Baum um die Blumen zu klassifizieren, und geben Sie die Vorhersagen zurück.
6. Die Vorhersagen werden von Code Expert automatisch bewertet.

Detaillierte Anweisungen befinden sich auf Code Expert.

5. Hausaufgaben

Exercise 10: Intro ML I

Auf https://expert.ethz.ch/enrolled/SS25/mavt2/exercises

- Cancer Detection
- Diabetes Prediction
- Gini Index

Abgabedatum: Montag 19.05.2025, 20:00 MEZ

NO HARDCODING

Fragen?

Feedback?

Zu schnell? Zu langsam? Weniger Theorie, mehr Aufgaben?
Dankbar für Feedback am besten mir direkt sagen oder Mail schreiben

Credits

Die Slide(-templates) stammen ursprünglich von Julian Lotzer und Daniel Steinhauser!

→Checkt ihre Websites ab für zusätzliches Material in Informatik I, Informatik II und Stochastik & Machine Learning.

- https://n.ethz.ch/~jlotzer/
- https://n.ethz.ch/~dsteinhauser/