

# Von Daten zu Prognosen

## Machine Learning mit Python

Maierbrugger, Raab

# EINFÜHRUNG IN ML

## **Supervised Learning**

„Lernen mit Lehrer“

Zielgröße vorhersagen,  
Regression oder Klassifikation

## **Unsupervised Learning**

„Lernen ohne Lehrer“

Strukturen in Daten finden,  
Clustering, Ausreißer erkennen

# **Arten von Machine Learning**

## **Reinforcement Learning**

„Verstärkendes Lernen“

Agent lernt eine Strategie  
Robotik, Spiele

# Supervised Learning

- Ausgangslage: Daten mit beschreibenden Variablen („Features“)  $x_i$  und bekannten Werten für die Zielgröße  $y$  („Labels“)

Id	Position	Experience	Skill	Country	City	Salary (USD)
1	Developer	0	1	USA	New York	103100
2	Data Scientist	3	1	USA	Washington	146800
3	Accountant	8	1	USA	Los Angeles	124200
4	...					

- Ziel: auch für neue Datenpunkte eine Vorhersage für  $y$  treffen können

Id	Position	Experience	Skill	Country	City	Salary (USD)
173	Accountant	1	0	USA	Washington	???

# Supervised Learning - Grundaufgaben

## Klassifikation

- Vorhersage der Zugehörigkeit zu einer Klasse
- oft: binäre Klassifikation (ja/nein, 0/1)
- Beispiel: Binäres Bildklassifikations-Problem „Muffin vs. Chihuahua“



<https://www.kaggle.com/datasets/samueltcortinhas/muffin-vs-chihuahua-image-classification>

## Regression

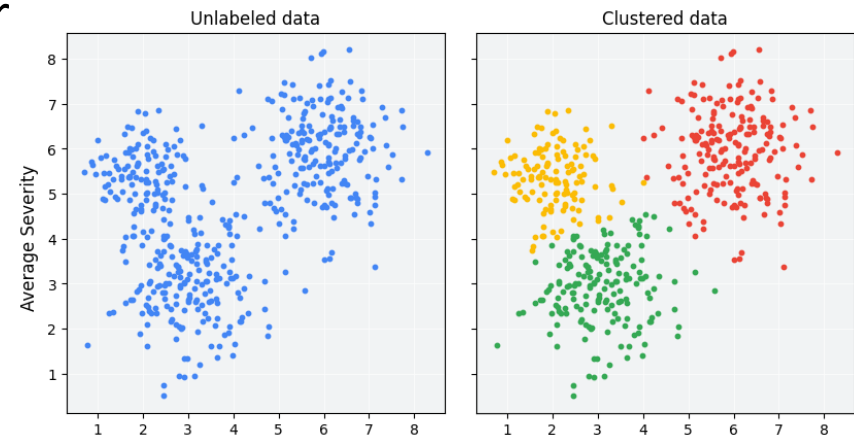
- Vorhersage einer (praktisch) kontinuierlichen Größe
- Beispiel: Hauspreise vorhersagen anhand von Features wie Quadratmeter, Anzahl der Zimmer, Baujahr, Entfernung zur nächsten Stadt



<https://www.kaggle.com/datasets/muhamedumarjamil/house-price-prediction-dataset>

# Unsupervised Learning

- Ausgangslage: Daten mit beschreibenden Variablen (Features)  $x_i$ , es liegen KEINE Werte einer Zielgröße vor
- Mögliche Ziele:
  - **Clustering:** Einteilung der Daten in Gruppen
  - **Dimensionsreduktion:** Darstellung vereinfachen, ursprüngliche Daten zu wenigen wichtigen reduzieren



# Reinforcement Learning

- Verstärkendes Lernen
  - Ein Agent kann in einer (simulierten) Umgebung („state“) Aktionen setzen
  - Agent bekommt einen „reward“ (positiv oder negativ)
  - Umgebung verändert sich
- Ziel des Agenten: rewards **langfristig** maximieren
- Aktuell: Deep Reinforcement Learning (DRL), mit Neuronalen Netzen

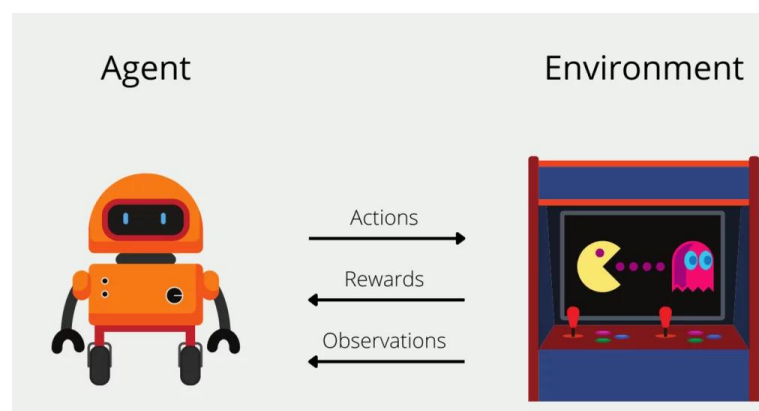
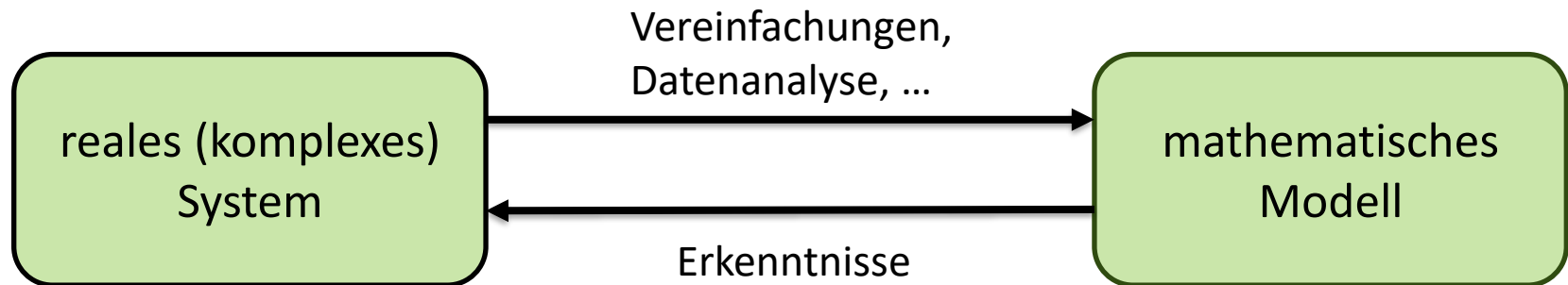


Image Source: <https://databasecamp.de/en/ml/reinforcement-learnings>

# Modellierung



- Ist meist als Optimierungsaufgabe formuliert
- Ziel im Supervised Learning: Vorhersagefehler minimieren
  - dafür benötigt man sinnvoll definierte **Metriken!**
- Modellparameter
  - Werden aus den Daten ermittelt (Allgemeine Info aus den Daten)
- Hyperparameter
  - Nehmen Einfluss auf den Trainingsprozess, nicht im Modell sichtbar

# Metriken für Regression

Situation: wir haben  $N$  labels  $y_i$  und  $N$  Modellvorhersagen  $\hat{y}_i$

- **Mean Absolute Error (MAE)**

- $MAE = \frac{1}{N} \cdot \sum_{i=1}^N |y_i - \hat{y}_i|$

- **Mean Squared Error (MSE)**

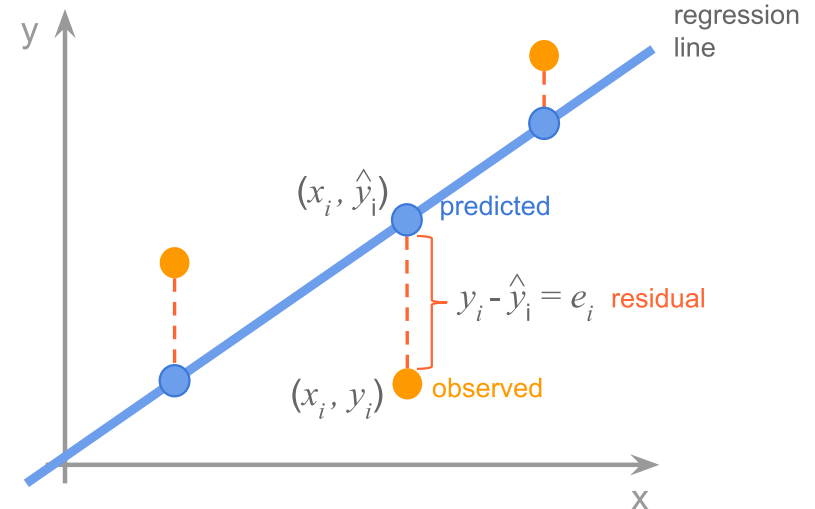
- $MSE = \frac{1}{N} \cdot \sum_{i=1}^N (y_i - \hat{y}_i)^2$

- “Bestraft” große Abweichungen stärker als MAE

- **Root Mean Squared Error (RMSE)**

- $RMSE = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^N (y_i - \hat{y}_i)^2}$

- Gleiche Einheit wie die response Variable  $y$



# Metriken für Klassifikation

- Situation: Wir haben ein binäres Klassifikationsproblem
- Beispiel: Bewerber – Variable “Hired“ (Positive / Negative)
- Das Modell Klassifiziert jede Variable als *Positive* oder *Negative*
- Das Ergebnis der Vorhersage wird in einer **Confusion Matrix** dargestellt

		Predicted class	
		Positive	Negative
Actual class	Positive	TP	FN
	Negative	FP	TN

# Metriken für Klassifikation

Die folgenden Metriken werden aus der Confusion Matrix abgeleitet.

Alle 4 Metriken nehmen Werte zwischen **0 (schlechtestes)** und **1 (bestes)** an.

$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$	Relative Häufigkeit der korrekten Klassifikationen
$\text{Precision} = \frac{TP}{TP+FP}$	Welcher Anteil der Fälle, die als positiv klassifiziert werden, sind tatsächlich positiv? Wenn FP problematisch sind (z.B. Spam-Filter)
$\text{Recall} = \frac{TP}{TP+FN}$	Welcher Anteil der tatsächlich positiven Fälle wurde erkannt? Wenn FN problematisch sind (z.B. Krebsdiagnose)
$\text{F1-Score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$	Harmonisches Mittel von Precision und Recall

# Modellkomplexität

wichtige Aufgabe beim Modellieren: Modellkomplexität „richtig“ wählen

- zu einfaches Modell → wichtige Eigenschaften werden nicht wiedergegeben (zu starker „**bias**“)
- zu komplexes Modell → zu starke Anpassung an die Trainingsdaten („**variance**“), Modell kann nicht mehr generalisieren
- → Bias-Variance-Tradeoff

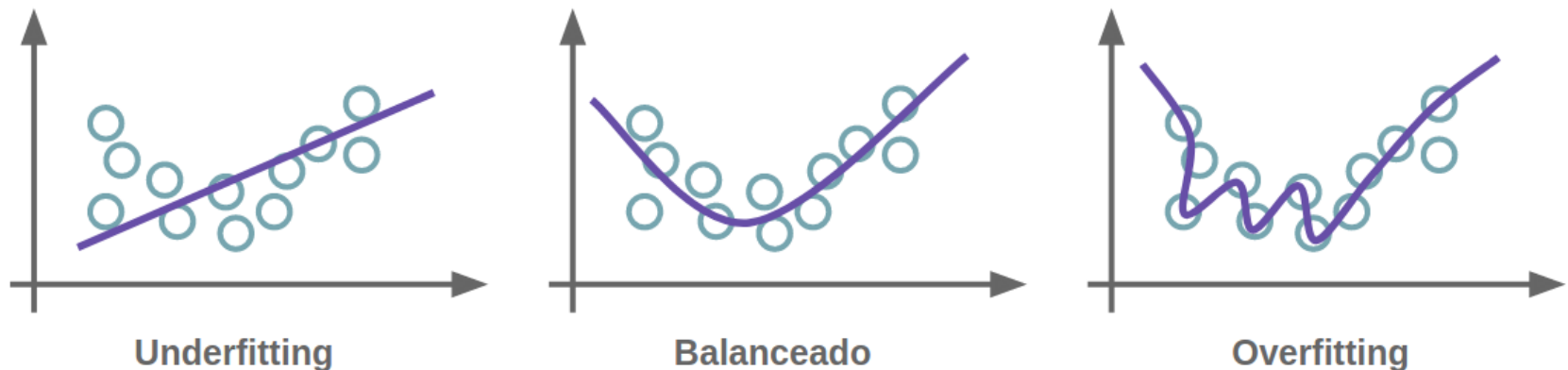
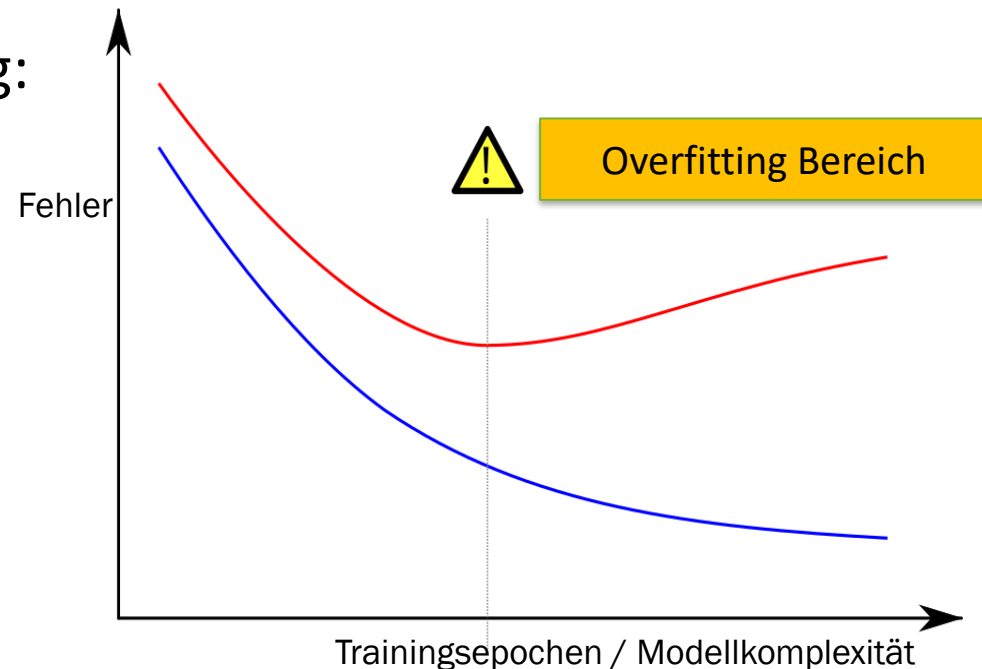


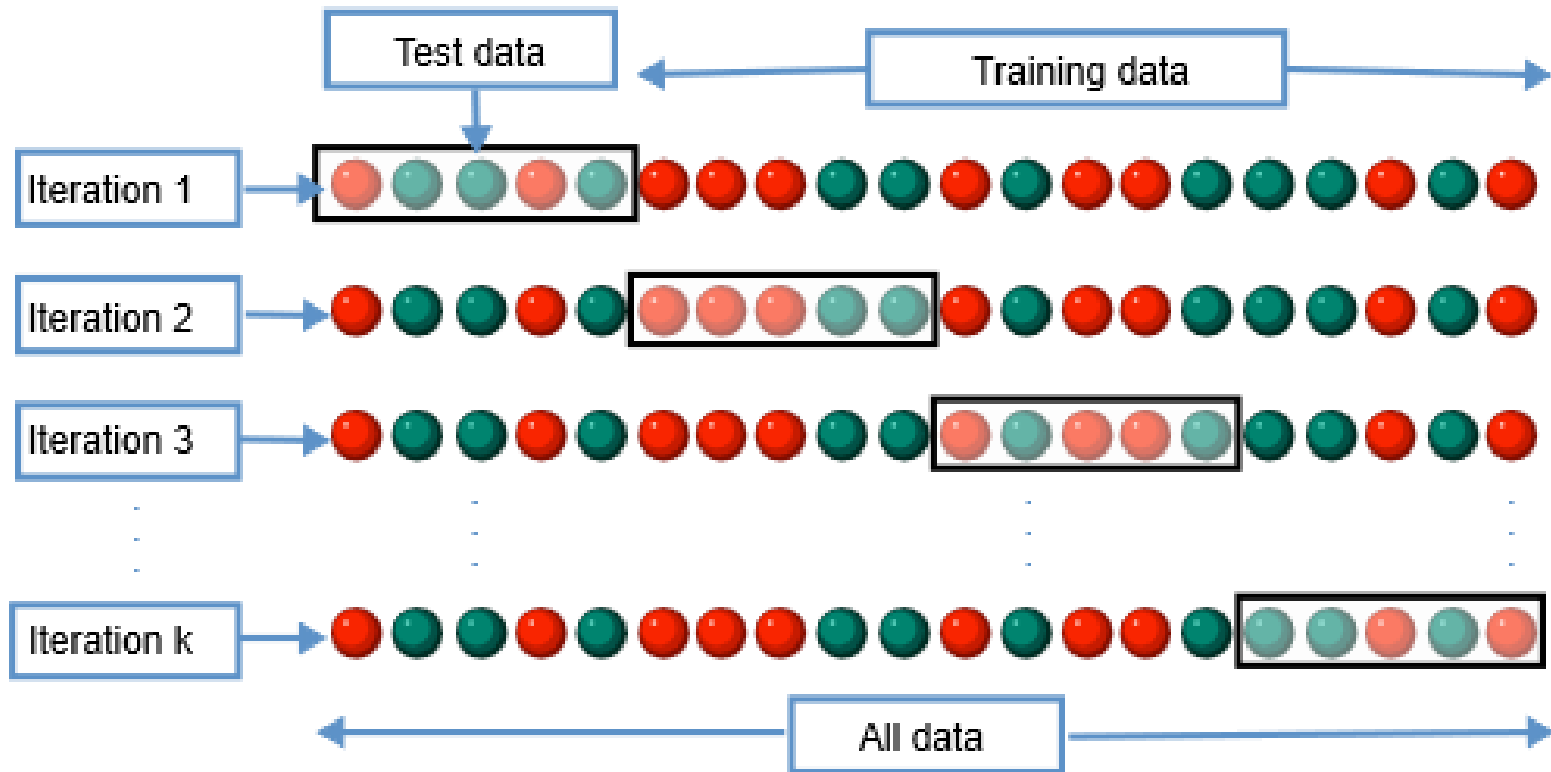
Image Source: [https://commons.wikimedia.org/wiki/File:Underfitting\\_e\\_overfitting.png](https://commons.wikimedia.org/wiki/File:Underfitting_e_overfitting.png)

# Modellkomplexität

- Modelltraining → Optimierungsaufgabe → Suche nach Minimum des Fehlers
- Fehler auf Trainingsdaten vs. Fehler auf Testdaten
- Strategien zur Vermeidung:
  - Baseline Modell
  - Train-Test-Split
  - Kreuzvalidierung



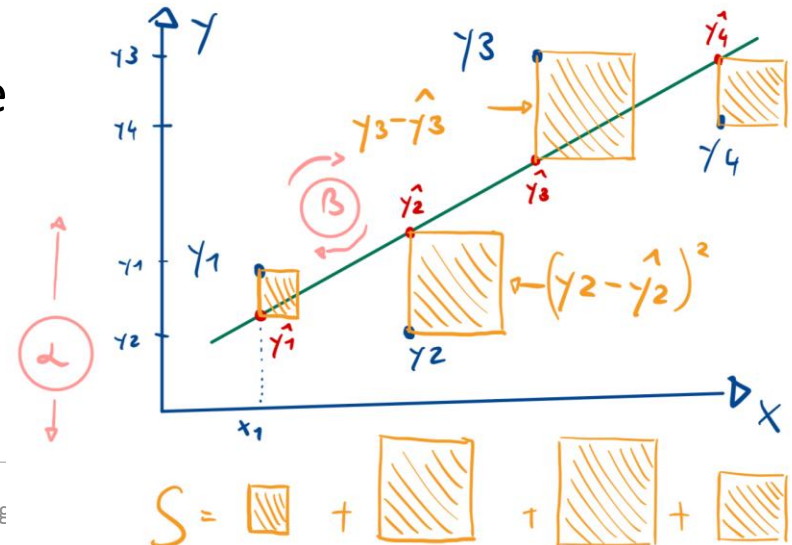
# Kreuzvalidierung



# SUPERVISED LEARNING METHODEN

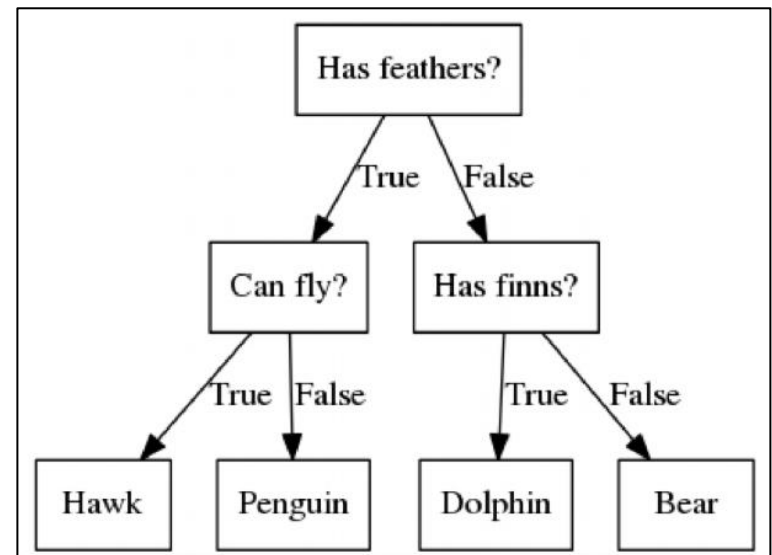
# Lineare Regression

- Einfache lineare Regression (ein Feature):
  - $\hat{y}_i = \alpha + \beta \cdot x_i$
  - $\alpha$  ... intercept (Schnitt mit der y-Achse)
  - $\beta$  ... slope (Steigung)
- Multiple lineare Regression (mehrere Features):
  - $\hat{y}_i = \beta_0 + \beta_1 \cdot x_{i1} + \beta_2 \cdot x_{i2} + \dots + \beta_p \cdot x_{ip}$
- Finden der Parameter:
  - Mithilfe der Least Squares Methode
  - $S$  minimieren
  - Analytische Methode



# Decision Trees

- Für Klassifikation und Regression
- Kreiert wenn-dann Entscheidungen (binär)
- Splitted die Daten schrittweise auf (basierend auf Variablenwerten), wählt immer den Split, der die Variabilität in den folgenden Knoten minimiert
- Sehr gut interpretierbar
- Neigt zu Overfitting

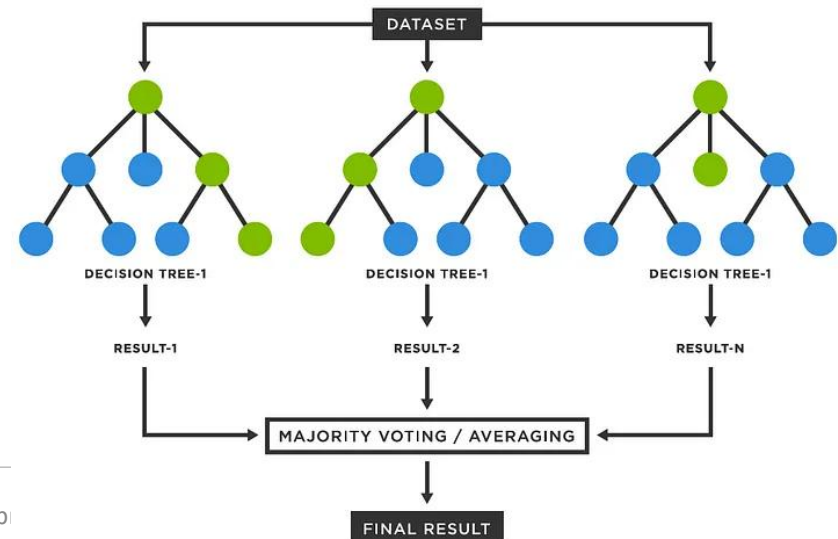


Img Source

<https://towardsai.net/p/programming/decision-trees-explained-with-a-practical-example-fe47872d3b53>

# Random Forests

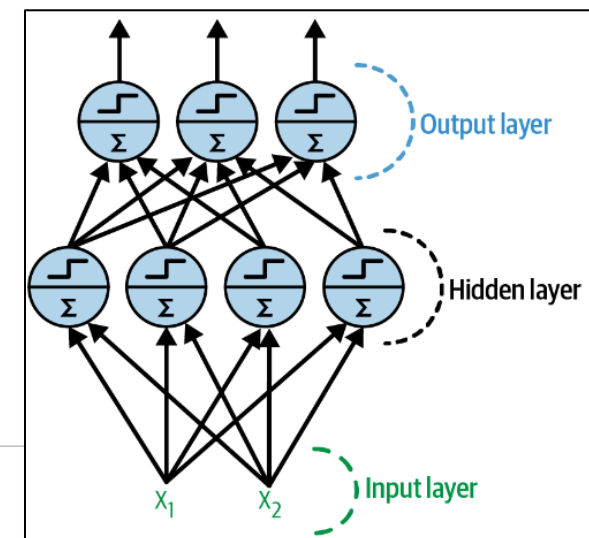
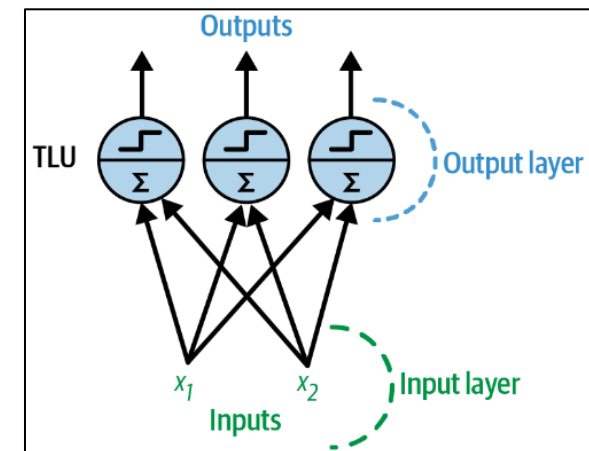
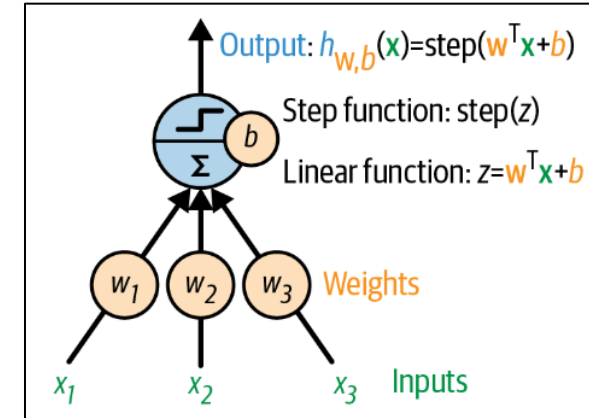
- Steigert die Vorhersagequalität und Robustheit gegenüber einzelner Decision Trees
- Man bildet zufällige Untermengen des Datensatzes, baut darauf einen DT
- Vorhersagen werden dann aggregiert (Mittelwert oder Mehrheitsvotum)
- Komplexere Beziehungen
- Weniger Overfitting
- Nicht leicht interpretierbar



Img Source: <https://www.displayr.com/what-is-a-decision-tree/> (11/12/2025)

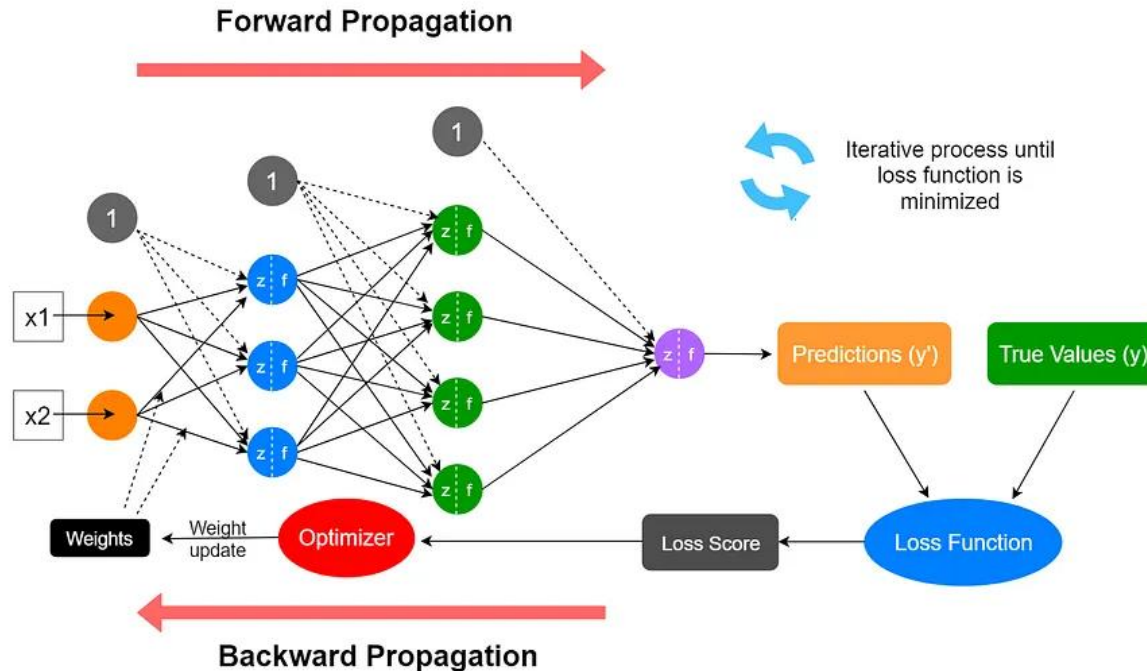
# Neuronale Netze

- Threshold Logic Unit (TLU) ist das einfachste Puzzle teil eines Artificial Neural Networks (ANN)
- Ein Perceptron ist ein Layer von TLUs
  - Fully connected
- Ein Multilayer Perceptron besteht aus
  - Input layer
  - Mehreren hidden layers
  - Output layer
- Viele hidden layers -> Deep Neural Network



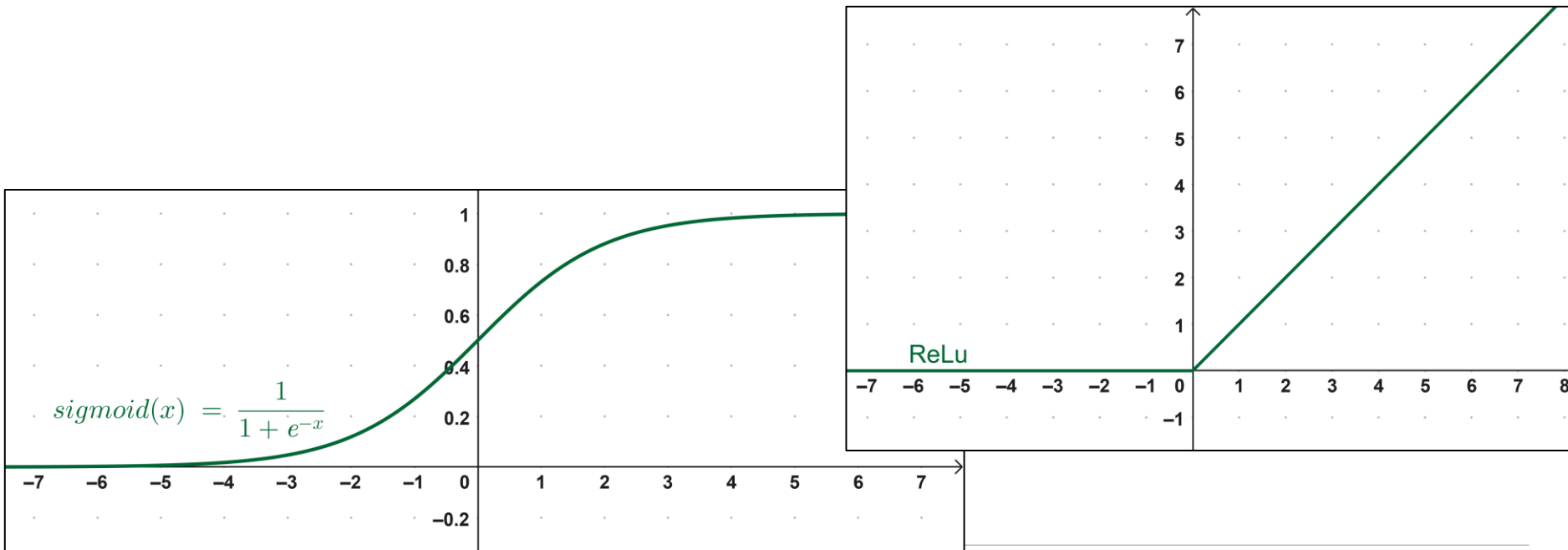
# Neuronale Netze - Training

- 3 Trainingsschritte, die vielfach wiederholt werden
  1. Forward Pass
  2. Berechnung des Loss
  3. Backpropagation



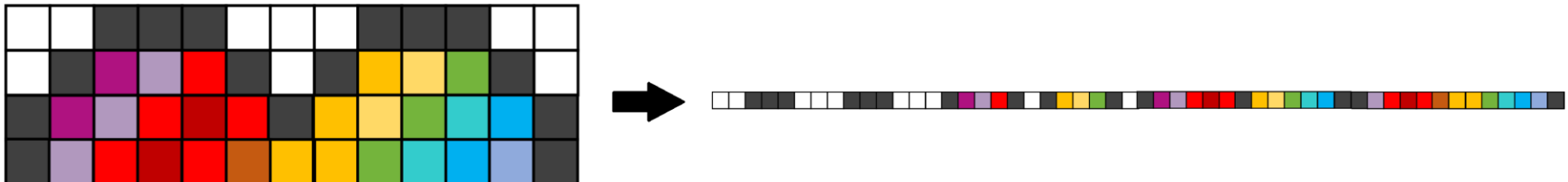
# Neuronale Netze - Aktivierungsfunktionen

- **Activation functions:**
  - Bringen “non-linearity” in das Modell – sonst nur lineare Zusammenhänge abbildbar
  - Dadurch können NNs (fast) alles lernen

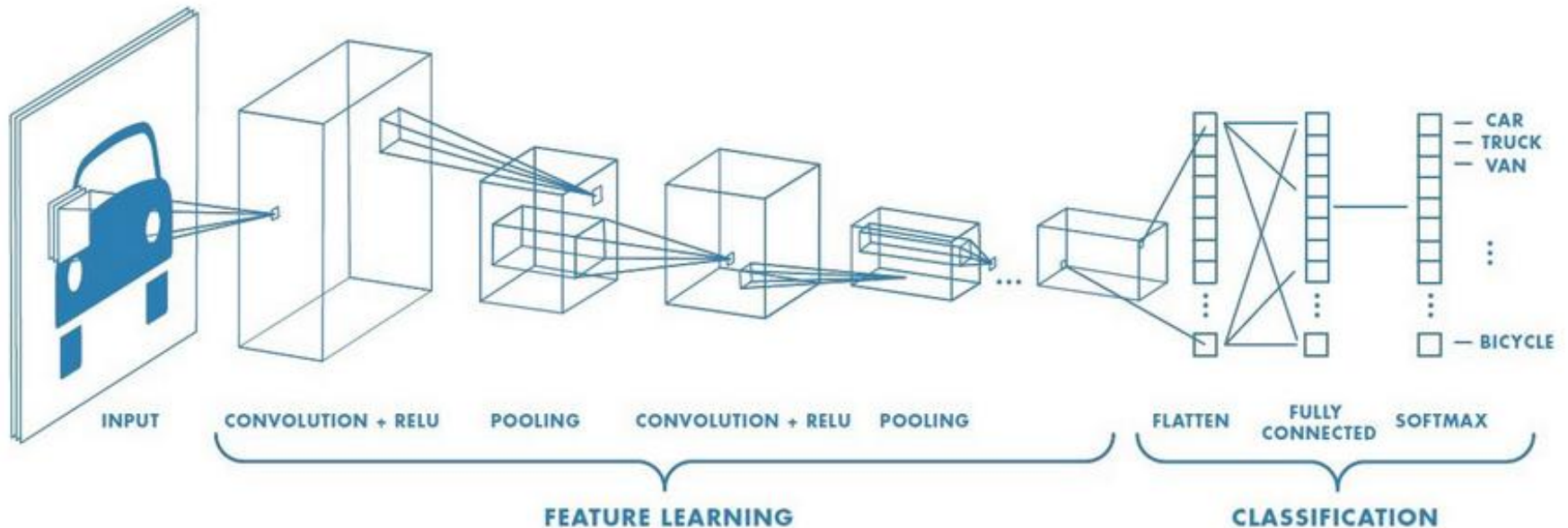


# Convolutional Neural Networks (CNN)

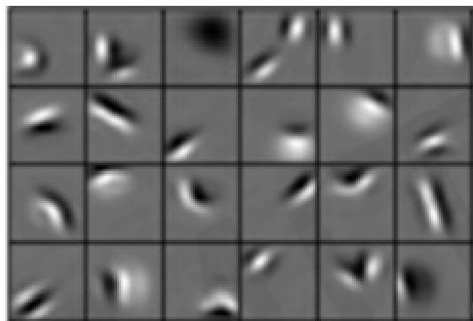
- Klassische Aufgabe: Bildklassifizierung
- Wenn ein “normales” Neuronales Netz verwendet wird, muss das Bild dekonstruiert werden (flatten)
  - Wichtiger Kontext im Bild geht verloren
  - Bei großen Bildern sehr hohe Anzahl an input neuronen
- CNNs
  - Können lokale Strukturen oder Muster lernen
  - Verbindung zwischen benachbarten Elementen bleibt bestehen
  - Verwendet „partially connected layers“ um eine zu hohe Anzahl an Gewichten zu vermeiden



# Convolutional Neural Networks (CNN)



Low level features



Edges, dark spots

Mid level features



Eyes, ears, nose

High level features



Facial structure