

# DATA SCIENCE 1

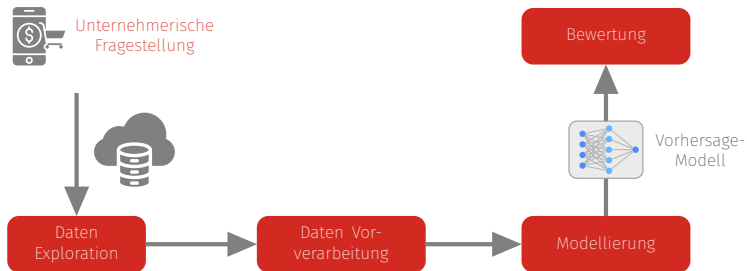
VORLESUNG 4 - MASCHINELLES LERNEN

PROF. DR. CHRISTIAN BOCKERMANN

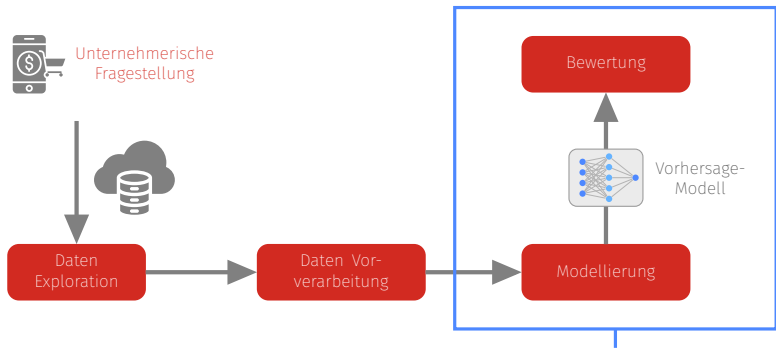
HOCHSCHULE BOCHUM

SOMMERSEMESTER 2026

## Wir erinnern uns: Vorgehen bei der Datenanalyse (CRISP-DM)



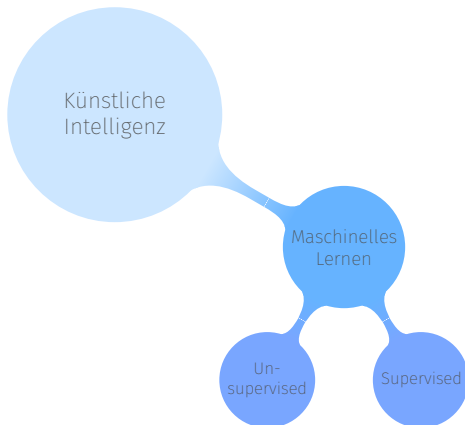
**Wir erinnern uns:** Vorgehen bei der Datenanalyse (CRISP-DM)



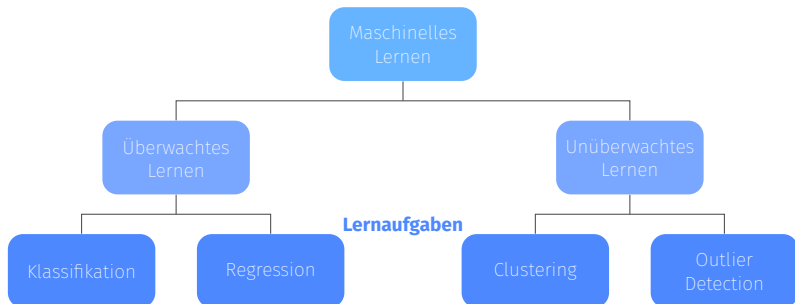
**Modellierung + Validierung**  
mit Python/SciKit-Lern (später)

- 1 Überblick - Maschinelles Lernen
  - Lernaufgaben
  
- 2 Überwachtes Lernen
  - Lernen von Modellen
  - Validierung von Modellen

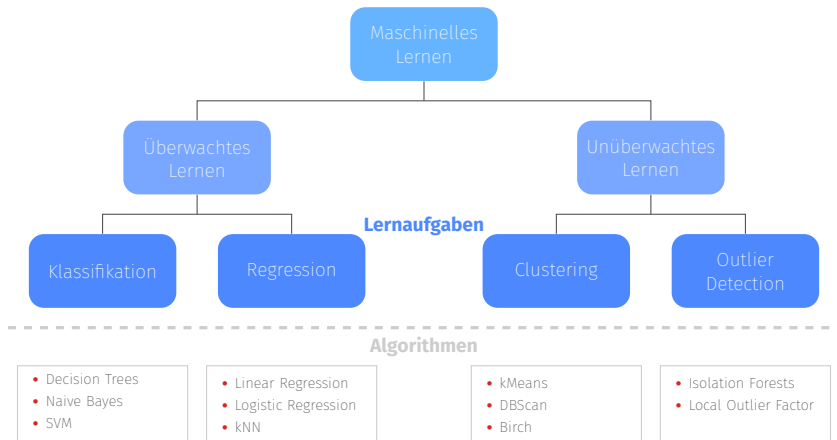
Maschinelles Lernen ist Teilgebiet der künstlichen Intelligenz



Maschinelles Lernen ist Teilgebiet der künstlichen Intelligenz



Maschinelles Lernen ist Teilgebiet der künstlichen Intelligenz



## Kategorien des Lernens

### Überwachtes Lernen

*supervised learning*

- Trainingsdaten enthalten Zielvariable (z.B. Spam=Ja/Nein)
- Zielvariable oft als *Klasse*, *Label* oder *Class* bezeichnet
- Mit Trainingsdaten, unbekannte Daten vorhersagen

### Unüberwachtes Lernen

*unsupervised learning*

- Trainingsdaten enthalten keine Zielinformation
- Unbekannte Muster/Gruppen in Daten finden

**Lernaufgaben** definieren Ein- und Ausgabe, sowie das Ziel der Modellierung, z.B.

“Entscheide für einen Text **x** ob er zur Klasse *Spam* oder zur Klasse *KeinSpam* gehört.”

**Lernaufgaben** definieren Ein- und Ausgabe, sowie das Ziel der Modellierung, z.B.

“Entscheide für einen Text  $\mathbf{x}$  ob er zur Klasse *Spam* oder zur Klasse *KeinSpam* gehört.”

Eingabedaten werden typischerweise in einen **Merkmalsraum**  $\mathcal{X}$  der Dimension  $d$  abgebildet

$$\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$$

Die Ausgabemenge  $\mathcal{Y}$  kann eine Menge von Klassen oder eine reelle Zahl sein, z.B.

$$\mathcal{Y} = \{\text{Spam}, \text{KeinSpam}\}$$

Das Ziel besteht darin, eine Funktion (Modell)  $f : \mathcal{X} \rightarrow \mathcal{Y}$  zu lernen, mit

$$f(\mathbf{x}) = \begin{cases} +1, & \text{falls } \mathbf{x} \text{ Spam Nachricht} \\ -1, & \text{sonst} \end{cases}$$

Das Ziel besteht darin, eine Funktion (Modell)  $f : \mathcal{X} \rightarrow \mathcal{Y}$  zu lernen, mit

$$f(\mathbf{x}) = \begin{cases} +1, & \text{falls } \mathbf{x} \text{ Spam Nachricht} \\ -1, & \text{sonst} \end{cases}$$

Bei der **binären Klassifikation** wird häufig  $\mathcal{Y} = \{-1, +1\}$  gewählt.

Das Ziel besteht darin, eine Funktion (Modell)  $f : \mathcal{X} \rightarrow \mathcal{Y}$  zu lernen, mit

$$f(\mathbf{x}) = \begin{cases} +1, & \text{falls } \mathbf{x} \text{ Spam Nachricht} \\ -1, & \text{sonst} \end{cases}$$

Bei der **binären Klassifikation** wird häufig  $\mathcal{Y} = \{-1, +1\}$  gewählt.

Für die **Regression** gilt  $\mathcal{Y} = \mathbb{R}$ .

Lern-Algorithmen erwarten Daten häufig in Form einer Tabelle:

$d$ Merkmale					
ID	$a_1$	$a_2$	$\dots$	$a_d$	$y$
1	0	0	$\dots$	1	-1
2	0	1	$\dots$	1	+1
3	1	0	$\dots$	1	-1

$$\begin{aligned}\text{Beispiel } \mathbf{x}_2 &= (x_{a_1}, x_{a_2}, \dots, x_{a_d}, y) \\ &= (0, 1, \dots, 1, +1)\end{aligned}$$

- Beispiele werden auch *examples* oder *instances* genannt
- Merkmale (engl. *features*) werden auch *attributes* oder *Variablen* (Statistik) bezeichnet

$a_1$	$a_2$	$\dots$	$a_d$	$y$
0	0	$\dots$	1	-1
0	1	$\dots$	1	+1
1	0	$\dots$	1	-1

Trainingsdaten  $\mathbf{X}, \mathbf{y}$ Algorithmus/  
Optimierung $f: \mathcal{X} \rightarrow \mathcal{Y}$ 

Modell

$a_1$	$a_2$	...	$a_d$	$y$
0	0	...	1	-1
0	1	...	1	+1
1	0	...	1	-1

Trainingsdaten  $\mathbf{X}, \mathbf{y}$ Algorithmus/  
Optimierung $f: \mathcal{X} \rightarrow \mathcal{Y}$ 

Modell

---

$a_1$	$a_2$	...	$a_d$	$y$
1	1	...	0	?

Neue Daten  $\mathbf{x}'$ ,  
 $\mathbf{y}$  unbekannt

$a_1$	$a_2$	...	$a_d$	$y$
0	0	...	1	-1
0	1	...	1	+1
1	0	...	1	-1

Trainingsdaten  $\mathbf{X}, \mathbf{y}$ Algorithmus/  
Optimierung $f: \mathcal{X} \rightarrow \mathcal{Y}$ 

Modell

$a_1$	$a_2$	...	$a_d$	$y$
1	1	...	0	?

Neue Daten  $\mathbf{x}'$ ,  
 $\mathbf{y}$  unbekannt

Vorhersage

$$\hat{y} = f(\mathbf{x}')$$

## Definition von Lernaufgaben

Im Folgenden werfen wir einen genaueren Blick auf die Definition von Lernaufgaben, die wir in den nächsten Vorlesungen behandeln:

### Überwachtes Lernen

- Klassifikation
- Regression

### Unüberwachtes Lernen

- Clustering
- (Outlier-Detection)
- Frequent Itemsets / Frequent Patterns

## Klassifikation ordnet Beispielen diskrete Klassen zu

- Vorgegebene Klassen  $\mathcal{Y} = \{C_1, \dots, C_k\}$
- Gegeben Menge  $\mathbf{X} \times \mathbf{y} \subset \mathcal{X} \times \mathcal{Y}$  bei der jedem Beispiel  $\mathbf{x}_i$  die zugehörige Klasse zugeordnet ist:  $(\mathbf{x}_i, \mathbf{y}_i)$
- Qualitätsfunktion  $q : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \rightarrow \mathcal{Y}) \rightarrow \mathbb{R}$

### Ziel:

- Finde Modell

$$f : \mathcal{X} \rightarrow \mathcal{Y},$$

das die Qualitätsfunktion optimiert.

## Klassifikation ordnet Beispielen diskrete Klassen zu

- Vorgegebene Klassen  $\mathcal{Y} = \{C_1, \dots, C_k\}$
- Gegeben Menge  $\mathbf{X} \times \mathbf{y} \subset \mathcal{X} \times \mathcal{Y}$  bei der jedem Beispiel  $\mathbf{x}_i$  die zugehörige Klasse zugeordnet ist:  $(\mathbf{x}_i, y_i)$
- Qualitätsfunktion  $q : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \rightarrow \mathcal{Y}) \rightarrow \mathbb{R}$

### Ziel:

- Finde Modell

$$f : \mathcal{X} \rightarrow \mathcal{Y},$$

das die Qualitätsfunktion optimiert.

**Lernen als Optimierungsproblem!**

**Beispiel: Klassifikation von Schwertlilien**

- Klassen:  $\mathcal{Y} = \{\text{setosa}, \text{versicolor}, \text{virginica}\}$
- Menge  $\mathbf{X} \times \mathbf{y}$  mit 150 Beispiele mit Spalte "species"
- Qualitätsfunktion

$$q(\mathbf{X} \times \mathbf{y}, f) = \sum_{(x, y) \in \mathbf{X} \times \mathbf{y}} \underbrace{\text{err}(y, f(x))}_{=\hat{y}}, \quad \text{err}(y, \hat{y}) = \begin{cases} 0, & \text{falls } y = \hat{y} \\ 1, & \text{sonst.} \end{cases}$$

**Beispiel: Klassifikation von Schwertlilien**

- Klassen:  $\mathcal{Y} = \{\text{setosa}, \text{versicolor}, \text{virginica}\}$
- Menge  $\mathbf{X} \times \mathbf{y}$  mit 150 Beispiele mit Spalte "species"
- Qualitätsfunktion

$$q(\mathbf{X} \times \mathbf{y}, f) = \sum_{(x,y) \in \mathbf{X} \times \mathbf{y}} \underbrace{\text{err}(y, f(x))}_{=\hat{y}}, \quad \text{err}(y, \hat{y}) = \begin{cases} 0, & \text{falls } y = \hat{y} \\ 1, & \text{sonst.} \end{cases}$$

**Funktion  $q$  zählt die Anzahl der Vorhersagefehler des Modells  $f$  auf der Menge  $\mathbf{X}$**

**Beispiel: Klassifikation von Schwertlilien**

- Klassen:  $\mathcal{Y} = \{\text{setosa}, \text{versicolor}, \text{virginica}\}$
- Menge  $\mathbf{X} \times \mathbf{y}$  mit 150 Beispiele mit Spalte "species"
- Qualitätsfunktion

$$q(\mathbf{X} \times \mathbf{y}, f) = \sum_{(x,y) \in \mathbf{X} \times \mathbf{y}} \underbrace{\text{err}(y, f(x))}_{=\hat{y}}, \quad \text{err}(y, \hat{y}) = \begin{cases} 0, & \text{falls } y = \hat{y} \\ 1, & \text{sonst.} \end{cases}$$

**Funktion  $q$  zählt die Anzahl der Vorhersagefehler des Modells  $f$  auf der Menge  $\mathbf{X}$**

**Ziel:** Finde  $f^*$  mit minimalem  $q(\mathbf{X}, f)$

**Beispiel: Klassifikation von Schwertlilien**

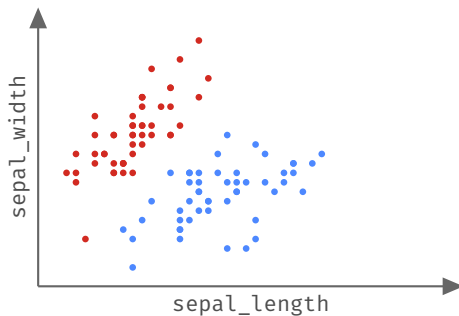
- Klassen:  $\mathcal{Y} = \{\text{setosa}, \text{versicolor}, \text{virginica}\}$
- Menge  $\mathbf{X} \times \mathbf{y}$  mit 150 Beispiele mit Spalte "species"
- Qualitätsfunktion

$$q(\mathbf{X} \times \mathbf{y}, f) = \sum_{(x,y) \in \mathbf{X} \times \mathbf{y}} \underbrace{\text{err}(y, f(x))}_{=\hat{y}}, \quad \text{err}(y, \hat{y}) = \begin{cases} 0, & \text{falls } y = \hat{y} \\ 1, & \text{sonst.} \end{cases}$$

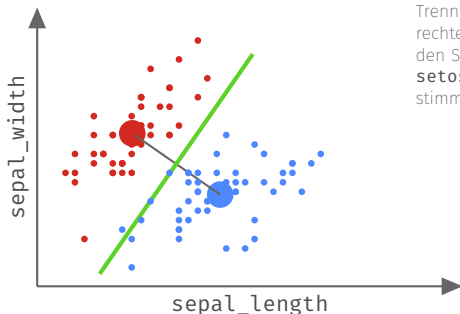
**Funktion  $q$  zählt die Anzahl der Vorhersagefehler des Modells  $f$  auf der Menge  $\mathbf{X}$**

**Ziel:** Finde  $f^*$  mit minimalem  $q(\mathbf{X}, f)$   $\rightarrow$  Optimierungsproblem

## Beispiel: **Klassifikation von Schwertlilien**



## Beispiel: Klassifikation von Schwertlilien

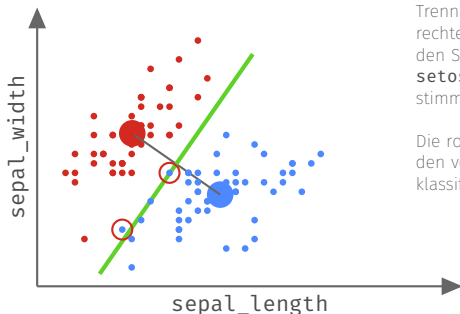


In diesem Fall wurde eine Trenn-Ebene als Mittelsenkrechte auf der Strecke zwischen den Schwerpunkten der Klasse **setosa** und **versicolor** bestimmt.

### Einfacher Algorithmus:

Trenn-Ebene über die Klassenschwerpunkte der Attribute **sepal\_length** und **sepal\_width**

## Beispiel: Klassifikation von Schwertlilien



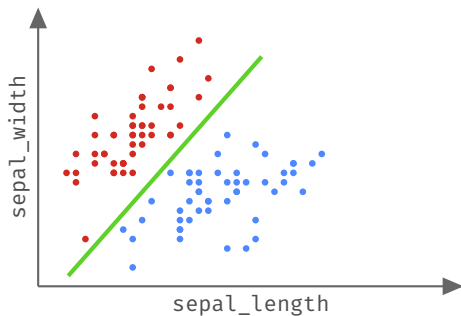
In diesem Fall wurde eine Trenn-Ebene als Mittelsenkrechte auf der Strecke zwischen den Schwerpunkten der Klasse **setosa** und **versicolor** bestimmt.

Die rot umkreisten Punkte werden von der Trenn-Ebene falsch klassifiziert.

### Einfacher Algorithmus:

Trenn-Ebene über die Klassenschwerpunkte der Attribute **sepal\_length** und **sepal\_width**

## Beispiel: Klassifikation von Schwertlilien



Die Daten sind *linear separierbar* – eine andere Ebene schafft dies ohne Fehler.  
Die Optimierung der Qualitätsfunktion sucht nach der besten Ebene.

## Regression liefert reellwertige Vorhersagen

- Für Regression gilt  $\mathcal{Y} = \mathbb{R}$
- Menge  $\mathbf{X} \times \mathbf{y}$ , d.h. jedem Beispiel  $\mathbf{x}_i$  ist ein  $\mathbf{y}_i \in \mathbb{R}$  zugeordnet
- Qualitätsfunktion  $q : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \rightarrow \mathcal{Y}) \rightarrow \mathbb{R}$

### Ziel:

- Finde Modell

$$f : \mathcal{X} \rightarrow \mathcal{Y},$$

das die Qualitätsfunktion optimiert.

## Regression liefert reellwertige Vorhersagen

- Für Regression gilt  $\mathcal{Y} = \mathbb{R}$
- Menge  $\mathbf{X} \times \mathbf{y}$ , d.h. jedem Beispiel  $\mathbf{x}_i$  ist ein  $\mathbf{y}_i \in \mathbb{R}$  zugeordnet
- Qualitätsfunktion  $q : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \rightarrow \mathcal{Y}) \rightarrow \mathbb{R}$

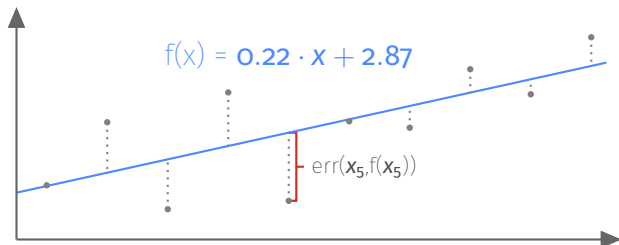
### Ziel:

- Finde Modell

$$f : \mathcal{X} \rightarrow \mathcal{Y},$$

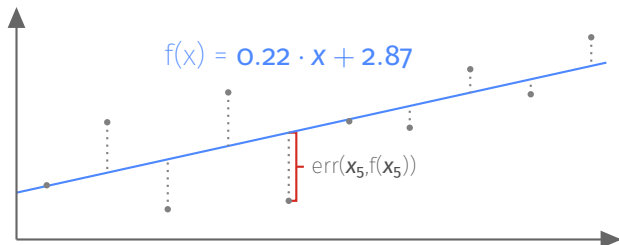
das die Qualitätsfunktion optimiert.

**Auch hier wieder: Lernen als Optimierungsproblem!**

**Beispiel: Regression****Qualitätsfunktion:**

Summe der Abstände von  $f(x)$  zu den "richtigen" Werten

$$q(X, f) = \sum_{(x, y) \in X} (y - f(x))^2 = \text{RSS}(X, f)$$

**Beispiel: Regression****Qualitätsfunktion:**

Summe der Abstände von  $f(x)$  zu den "richtigen" Werten

$$q(X, f) = \sum_{(x, y) \in X} (y - f(x))^2 = \text{RSS}(X, f)$$

**Residual Sum of Squares**

## Clustering sucht Aufteilung von Daten in ähnliche Gruppen

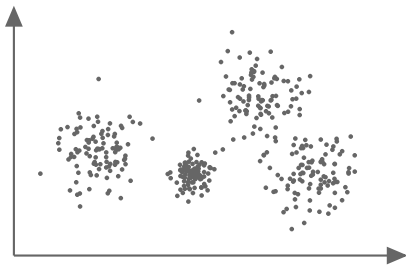
- Datenmenge  $\mathbf{X}$  von Beispielen (keine Klassen gegeben!)
- Parameter  $k$  zu findender Gruppen
- Abstandsmaß  $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$
- Qualitätsfunktion  $q$

### Ziel:

- Abstand *innerhalb* der Gruppen soll minimiert, Abstand *zwischen* den Gruppen soll maximiert werden

**Beispiel: Clustering**

Sei  $\mathbf{C} = C_1, \dots, C_k$  eine Aufteilung der Daten  $\mathbf{X}$  (ein *Clustering*)

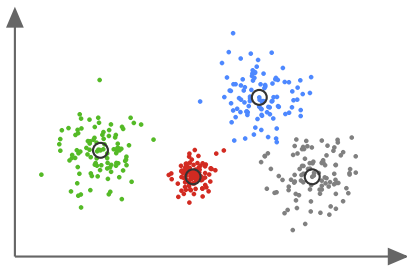


**Qualitätsfunktion:** (Innere Abstände)

$$q_{inner}(\mathbf{C}) = \sum_{i=1}^k \sum_{x \in C_i} d(x, \bar{c}_i) \quad , \text{ mit } \bar{c}_i \text{ Zentrum von } C_i$$

## Beispiel: Clustering

Sei  $\mathbf{C} = C_1, \dots, C_k$  eine Aufteilung der Daten  $\mathbf{X}$  (ein *Clustering*)



Clustering auf Datenpunkten mit  $k = 4$ . Die schwarzen Kreise markieren jeweils das Zentrum  $\bar{\mathbf{c}}_i$  des jeweiligen Cluster  $C_i$ .

**Qualitätsfunktion:** (Innere Abstände)

$$q_{inner}(\mathbf{C}) = \sum_{i=1}^k \sum_{x \in C_i} d(x, \bar{\mathbf{c}}_i) \quad , \text{ mit } \bar{\mathbf{c}}_i \text{ Zentrum von } C_i$$

## Beispiel: Clustering

- Clustering unter mehreren Qualitätsaspekten:

$$q_{inner}(\mathbf{C}) = \sum_{i=1}^k \sum_{x \in C_i} d(x, \bar{\mathbf{c}}_i) \quad \rightarrow \text{Minimieren}$$

$$q_{outer}(\mathbf{C}) = \sum_{i=1}^k \sum_{x \in C_j, j \neq i} d(x, \bar{\mathbf{c}}_i) \quad \rightarrow \text{Maximieren}$$

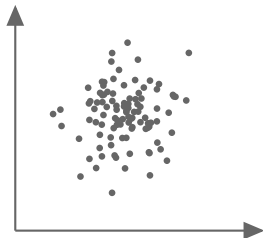
## Outlier-Detection sucht nach *isolierten* Punkten

- Gegen ist Datensatz  $\mathbf{X}$  (keine Label)
- i.d.R. noch Abstandsmaß  $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$

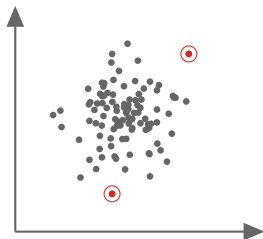
### Ziel:

- Finde Punkte, die *weit weg* von allen anderen Punkten liegen

## Beispiel: **Outlier-Detection**

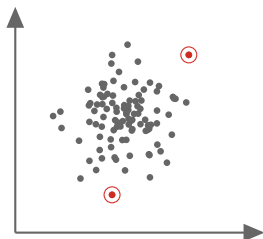


## Beispiel: Outlier-Detection



Die rot markierten Punkte sind Ausreißer die im Abstand von  $d = 0.35$  keine Nachbarpunkte haben.

## Beispiel: Outlier-Detection



Die rot markierten Punkte sind Ausreißer die im Abstand von  $d = 0.35$  keine Nachbarpunkte haben.

### Unterschiedliche Ansätze für Ausreißer-Erkennung:

- Vorgabe des minimalen Abstandes zu Nachbarn (siehe oben)
- Dichte-basiert - Verteilung der Abstände
- Über Mini-Clustering (ganz viele kleine Cluster finden)

## Frequent Itemset Mining sucht häufige Muster

- Gegeben ist Menge **S** von Symbolen (z.B. Artikel)
- Eingabe ist Menge **X** von Transaktionen (Einkäufe) über **S**

$$\mathbf{X} = \{ \mathbf{x} \mid \mathbf{x} \subseteq \mathbf{S} \}$$

### Ziel:

- Fragestellung: Welche Symbole tauchen häufig zusammen auf?
- Finde die Muster  $\mathbf{p} \in \mathcal{P}(\mathbf{S})$  die in **X** am häufigsten vorkommen

## Beispiel: Frequent Itemsets auf Einkäufen

ID	Artikel
1	{ A, B, F }
2	{ B, D, E, F }
3	{ C, E }
4	{ B, E, F }
5	{ A, B, E }

**Beispiel: Frequent Itemsets auf Einkäufen**

ID	Artikel
1	{ A, B, F }
2	{ B, D, E, F }
3	{ C, E }
4	{ B, E, F }
5	{ A, B, E }

- Artikel **B** = Muster { **B** } taucht in 4/5 der Einkäufe auf

**Beispiel: Frequent Itemsets auf Einkäufen**

ID	Artikel
1	{ A, B, F }
2	{ B, D, E, F }
3	{ C, E }
4	{ B, E, F }
5	{ A, B, E }

- Artikel **B** = Muster { **B** } taucht in 4/5 der Einkäufe auf
- Muster { **B, F** } taucht in 3/5 aller Einkäufe auf

**Beispiel: Frequent Itemsets auf Einkäufen**

ID	Artikel
1	{ A, B, F }
2	{ B, D, E, F }
3	{ C, E }
4	{ B, E, F }
5	{ A, B, E }

- Artikel **B** = Muster { **B** } taucht in 4/5 der Einkäufe auf
- Muster { **B, F** } taucht in 3/5 aller Einkäufe auf

**Welche Artikel werden häufig zusammen gekauft?**

# Überwachtes Lernen

## Charakterisierung des Überwachten Lernens

- Lernen auf Daten  $\mathbf{X}$  mit zugeordnetem Label  $\mathbf{y}$  (=“Wahrheit”)
- Label oft manuell vergeben oder Messwerte (Regression)
- Validierung von Modell  $f$  durch Vergleich mit  $\mathbf{y}$  möglich

## Charakterisierung des Überwachten Lernens

- Lernen auf Daten  $\mathbf{X}$  mit zugeordnetem Label  $\mathbf{y}$  (=“Wahrheit”)
- Label oft manuell vergeben oder Messwerte (Regression)
- Validierung von Modell  $f$  durch Vergleich mit  $\mathbf{y}$  möglich

### Beispiel: MNIST-Datensatz - Ziffernerkennung



Für Trainingsdaten: Manuelle Zuordnung der Ziffernbilder zum richtigen Label (2, 9, 6,...)

## Lernen auf Daten

a1	a2	a3	y
4	1	2	1
5	1	3	-1
3	8	7	1

**X**                      **y**

$\hat{y}$
-1
-1
1

**f(X)**

- Lernalgorithmus sucht bestes Modell  $f^*$  für Daten **X, y**
- Ziel des Trainings: Fehler auf **X, y** minimieren:

$$f^* = \arg \min_f \sum_{y \in \mathbf{y}} \text{err}(y, f(y)) \quad (\text{Trainingsfehler})$$

## Wie lernt ein Algorithmus?

- Algorithmus hat Klasse von Lösungen (z.B. Trenn-Ebenen)
- Lösungsraum wird parametrisiert und die beste Lösung gesucht ([Optimierungsproblem](#))

### Beispiel: Lineare Modelle

- Alle Ebenen in  $\mathcal{X}$  darstellbar als

$$\vec{x} \cdot \vec{n} = d \quad (\text{Hessesche Normalform})$$

- Suche  $\vec{n} = (n_1, \dots, n_k)$  und  $d$ , das möglichst viele Beispiele aus  $\mathbf{X}$  richtig klassifiziert (den Trainingsfehler minimiert)

**Zentrale Frage: Wie gut ist das gelernte Modell  $f^*$  ?**

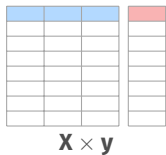
- Trainingsfehler gibt nur Auskunft über  $f^*$  auf *bekanntem Daten*  
 $\mathbf{X} \times \mathbf{y}$

**Zentrale Frage: Wie gut ist das gelernte Modell  $f^*$  ?**

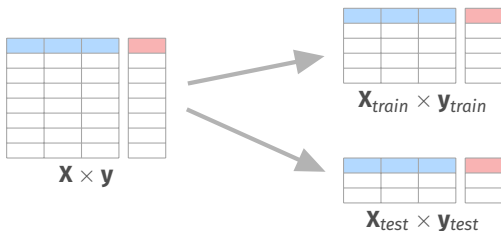
- Trainingsfehler gibt nur Auskunft über  $f^*$  auf *bekanntem Daten*  
 $\mathbf{X} \times \mathbf{y}$

**Wie gut ist  $f^*$  auf unbekanntem Daten?**

## Ansatz: Aufteilung in Trainings- und Test-Daten

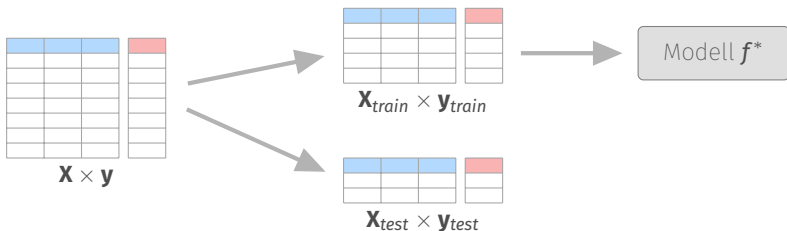


## Ansatz: Aufteilung in Trainings- und Test-Daten



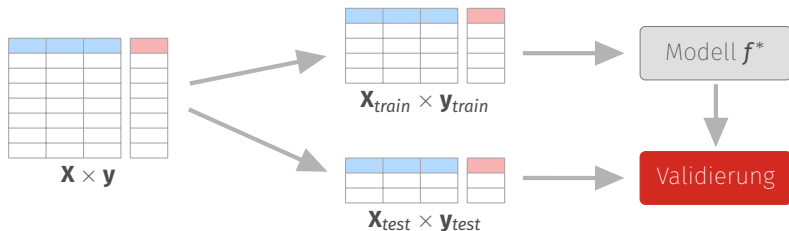
- Nutze *unabhängige Test-Daten* um  $f^*$  zu validieren!
- Oft 80% Trainingsdaten, 20% zum Testen (auch 70/30)

## Ansatz: Aufteilung in Trainings- und Test-Daten



- Nutze *unabhängige Test-Daten* um  $f^*$  zu validieren!
- Oft 80% Trainingsdaten, 20% zum Testen (auch 70/30)

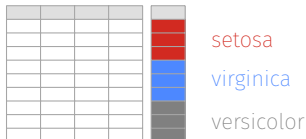
## Ansatz: Aufteilung in Trainings- und Test-Daten



- Nutze *unabhängige Test-Daten* um  $f^*$  zu validieren!
- Oft 80% Trainingsdaten, 20% zum Testen (auch 70/30)

## Aufteilung in Train/Test Daten

- Ok, nehmen wir 80:20 - was müssen wir beachten?
- Denken Sie an den Iris Datensatz (Übungsblatt 2, Aufgabe 2)!



## Was passiert bei folgender Aufteilung von 60:40?

```
n = iris.shape[0]           # n Beispiele
splitAt = int(0.6 * n)     # 60% zum Training

X_train = iris[0:splitAt]
X_test  = iris[splitAt:]
```

## Wie ähnlich sollten sich $\mathbf{X}_{train} \times \mathbf{y}_{train}$ und $\mathbf{X}_{test}$ sein?

Klassenverhältnis im Iris Datensatz:

- Gleichverteilt: **setosa** / **virginica** / versicolor jeweils 1/3
- Bei *linearem Splitting* im Verhältnis 60:40 ergibt sich:

50 × **setosa**  
40 × **virginica**

$\mathbf{X}_{train} \times \mathbf{y}_{train}$

10 × **virginica**  
50 × versicolor

$\mathbf{X}_{test} \times \mathbf{y}_{test}$

## Wie ähnlich sollten sich $\mathbf{X}_{train} \times \mathbf{y}_{train}$ und $\mathbf{X}_{test}$ sein?

Klassenverhältnis im Iris Datensatz:

- Gleichverteilt: **setosa** / **virginica** / versicolor jeweils 1/3
- Bei *linearem Splitting* im Verhältnis 60:40 ergibt sich:

50 × **setosa**  
40 × **virginica**

$\mathbf{X}_{train} \times \mathbf{y}_{train}$

10 × **virginica**  
50 × versicolor

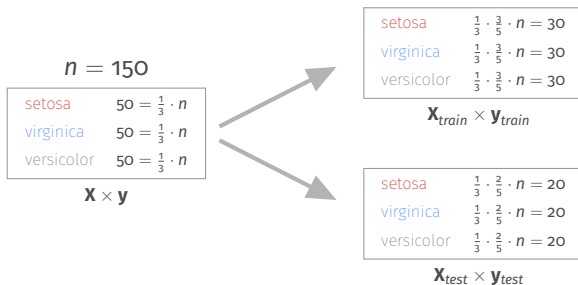
$\mathbf{X}_{test} \times \mathbf{y}_{test}$

**Klasse versicolor in Trainingsdaten nicht enthalten!**

**Klasse **setosa** in Testdaten nicht enthalten!**

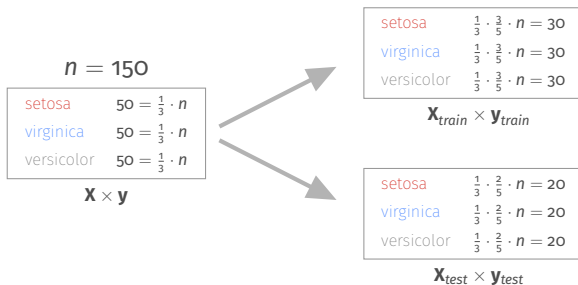
## Split gemäß der Klassenverteilung: **Stratified Sampling**

- **Stratified Sampling** erhält die Klassenverhältnisse
- Beispiel für 60:40 Split:



## Split gemäß der Klassenverteilung: **Stratified Sampling**

- **Stratified Sampling** erhält die Klassenverhältnisse
- Beispiel für 60:40 Split:



**Aber:** Was ist mit den Verteilungen der anderen Attribute?  
Zum Beispiel `sepal_length`?

## Weiteres Problem:

- Daten mit Label oft nur begrenzt verfügbar
- Wir wollen möglichst viele Daten für ein gutes Modell nutzen

## Idee: **Leave-One-Out**

1. Wähle ein Beispiel  $x_i \in \mathbf{X}$
2. Trainiere das Modell  $f$  auf  $(n - 1)$  Beispielen  $\mathbf{X} \setminus \{x_i\}$
3. Wir validieren  $f$  auf dem einen ausgewählten Beispiel  $x_i$
4. Wiederhole das für alle  $n$  Beispiel und berechne den Durchschnittsfehler

## Weitere Validierungsmethode ist **Cross Validation**

**Vorgehen:** Datenmenge sei  $\mathbf{T}$

- Teile Daten  $\mathbf{T}$  zufällig in  $k$  Teilmengen  $\mathbf{T}_1, \dots, \mathbf{T}_k$  auf
- Nutze  $\mathbf{T}_i$  als Testmenge und  $\mathbf{T} \setminus \mathbf{T}_i$  als Trainingsdaten
- Wiederhole dies für  $i = \{1, \dots, k\}$  und berechne Durchschnittsfehler über alle  $\mathbf{T}_i$



**Varianten:**

- **Stratified Cross-Validation** erhält Klassenverteilung in den  $\mathbf{T}_i$
- Spezialfall  $k = n = |\mathbf{T}|$  ergibt **Leave-one-out** Validierung