k-Nearest Neighbors (kNN)

Dr. Ralf Höchenberger, ERGO Group AG

07.01.2025

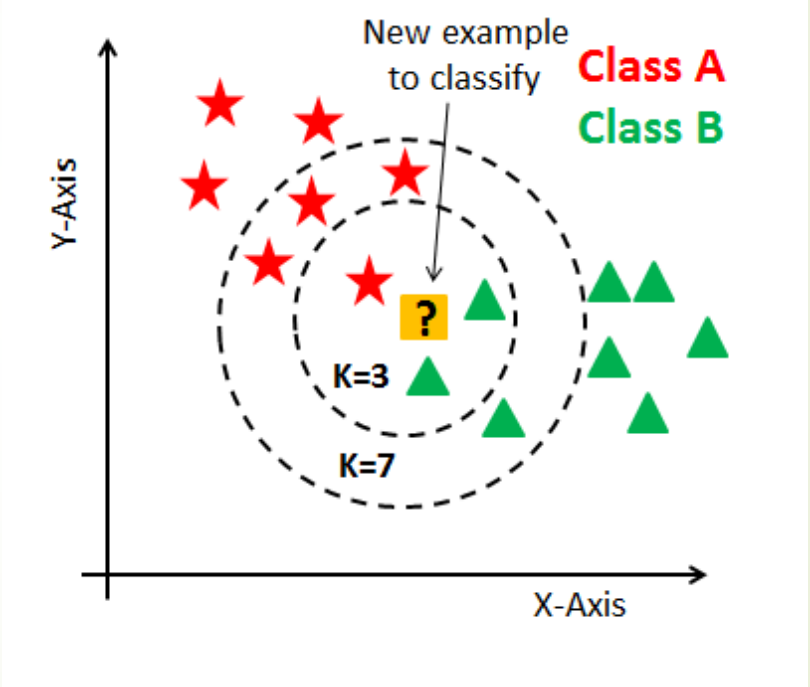
Worum geht es beim kNN-Algorithmus?

- kNN ist ein **Data Science-Algorithmus**
- kNN gehört zu den Algorithmen des **überwachten Lernens**, **d.h. die Vorhersagen basieren auf bereits bekannten Mustern**
- **Problem: Wie können neue Daten in die Muster zugeordnet werden?**
- **Anwendungsbeispiele:**
  - **Klassifikation von Kunden** (z.B. (nicht) zahlungsfähige Kreditnehmer)
  - **Vorhersage von Krankheiten in der Medizin**
  - (Schätzung fehlender Daten)

Problemstellung
Vorgehensweise
Algorithmus
Optimierung der Genauigkeit
Diskussion

Grundidee des kNN-Algorithmus

- **Neue Datenpunkte** sollen an der **Eigenschaften seiner Nachbarn klassifiziert w**erden: Er bekommt die Klasse, die die Mehrheit der Nachbarpunkte hat!
- Dabei definiert der Parameter k die **Anzahl der nächsten Nachbarn, die man betrachtet**
- Die Wahl von k ist entscheidend dafür, wie genau der Algorithmus arbeitet!
- In diesem Beispiel: bei k=3 wird der neue Datenpunkt in B (grün) klassifiziert, bei k=7 in A (rot)!

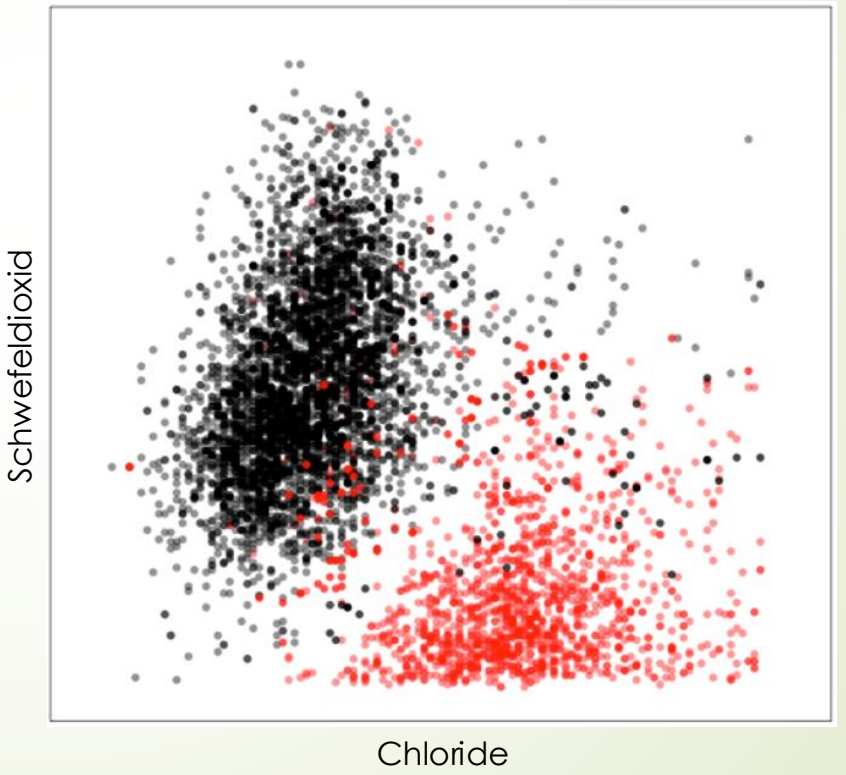


Beispiel: Weine erkennen

- Rotweine (im Bild rot) enthalten eher Chloride, Weißweine (im Bild schwarz) eher Schwefeldioxid
- **Frage:**

  Kann man die Farbe des Weines nur an Hand des Chlorid- und Schwefeldioxidgehalts erkennen (also ohne seine Farbe zu sehen)?
- **Idee des kNN:**

  Ein neuer Wein ist am ehesten von der Weinsorte wie die meisten benachbarten Weine



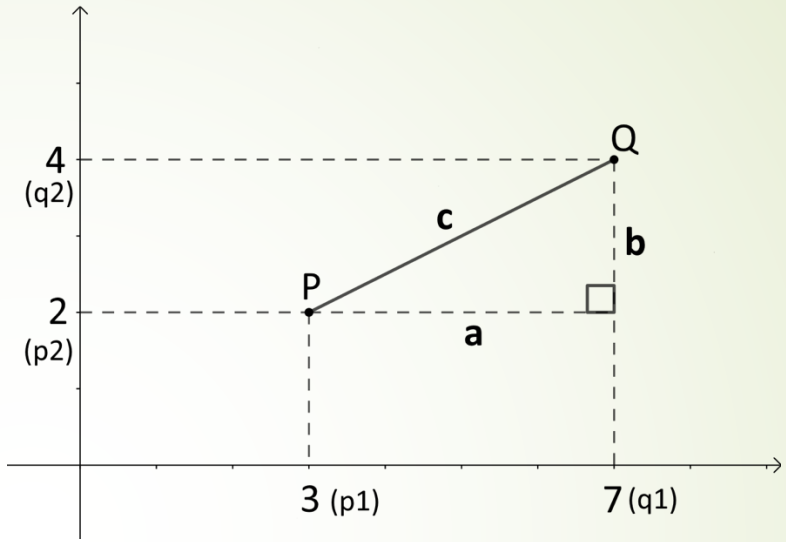
Problemstellung
Vorgehensweise
Algorithmus
Optimierung der Genauigkeit
Diskussion

Wie geht der kNN-Algorithmus vor?

1. Definiere eine Funktion, die den **Abstand zwischen zwei Punkten** berechnet.
2. Benutze diese Funktion, um den **Abstand zwischen einem Testpunkt und allen bekannten Datenpunkten** zu erhalten.
3. Sortiere die Abstände, um die Punkte zu finden, die dem **Testpunkt am nächsten** sind.
4. Benutze die **Zielvariable dieser Punkte** (z.B. Rot- oder Weißwein), **um die Zielvariable des Testpunktes vorherzusagen** (wenn mehr Rot- als Weißweine, dann Rotwein, sonst umgekehrt).
5. Schritte 1. bis 4. solange wiederholen, bis alle Datenpunkte klassifiziert sind.

Euklidischer Abstand

- Die häufigste Methode, um den Abstand zwischen zwei Punkten zu berechnen, ist der **Euklidische Abstand**
- Herleitung über rechtwinkliges Dreieck, das durch die zwei Punkte entsteht
- Satz des Pythagoras $a^2 + b^2 = c^2$, wobei wir c genau die Strecke (=Verbindungslinie) zwischen den beiden Punkte bezeichnen



$$a^2 + b^2 = c^2$$

$$c = \sqrt{a^2 + b^2}$$

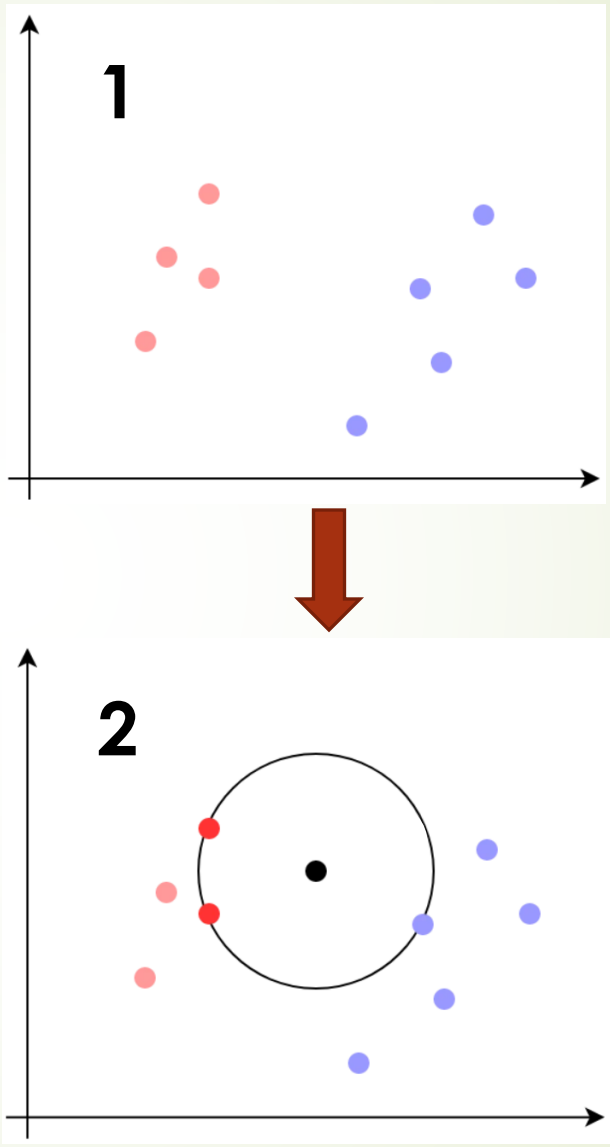
$$= \sqrt{(q1 - p1)^2 + (q2 - p2)^2}$$

$$= \sqrt{(7 - 3)^2 + (4 - 2)^2} \approx 4{,}47$$

Problemstellung
Vorgehensweise
Algorithmus
Optimierung der Genauigkeit
Diskussion

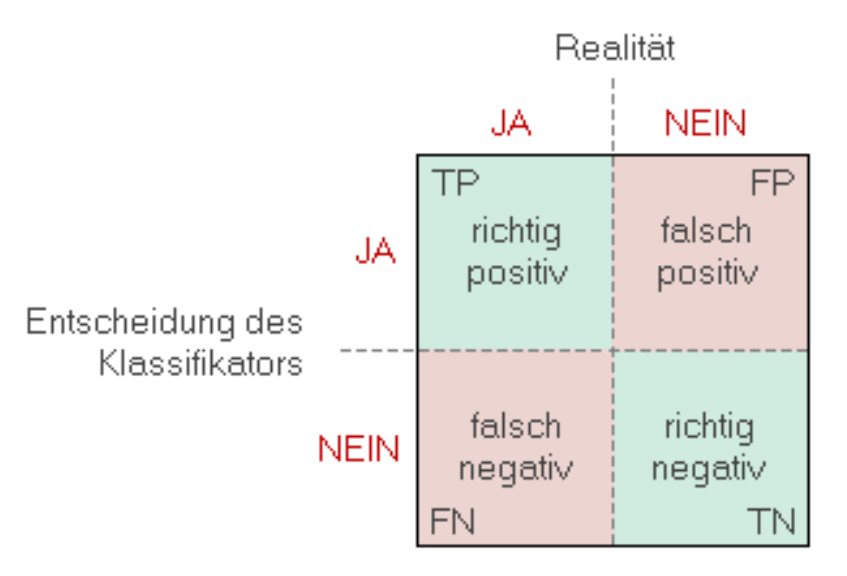
Beispiel für $k = 3$

- Punkte in Bild 1 sind **Trainingsdaten mit zwei Klassen**
- schwarzer Punkt in Bild 2 ist ein **neuer Datenpunkt** oder ein **Testpunkt aus einem Testdatensatz**
- da **zwei Nachbarn** des schwarzen Punktes **„rot“** sind und **einer „blau“**, bekommt der Punkt die Klasse **„rot“**



Wie genau arbeitet der Algorithmus?

- Die Genauigkeit kann daran gemessen werden, **wie viele der Testdaten vom Algorithmus bzw. Klassifikator richtig erkannt** werden
  - in unserem Beispiel: Werden Rotweine tatsächlich als Rotweine und Weißweine tatsächlich als Weißweine erkannt?
- **Wahrheitsmatrix**
  - **TP:** Weißweine als Weißweine erkannt
  - **TN:** Rotweine als Rotweine erkannt
  - **FP/FN** sollten möglichst klein sein!
- **Genauigkeits-Score:** $\frac{TP+TN}{TP+TN+FP+FN}$



Optimierung der Genauigkeit: k-fache Kreuzvalidierung

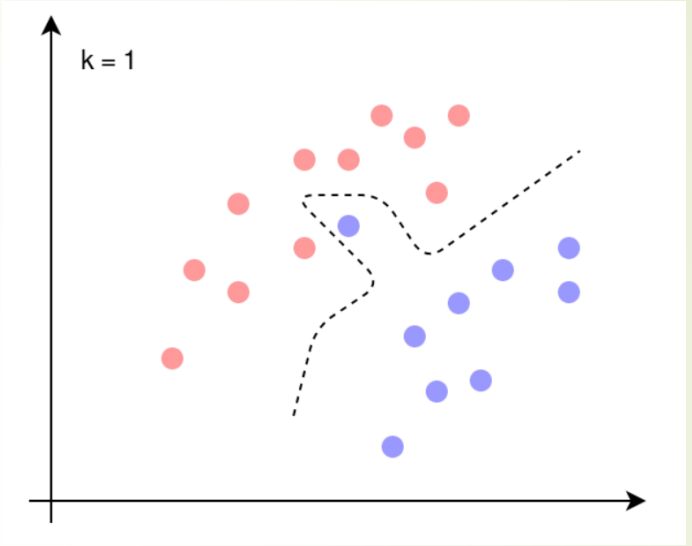
- Die initiale Einteilung in Trainings- und Testdatensatz könnte **unglücklich** gewesen sein (nicht repräsentative Ausreißer)
- Idee: **Teile den Datensatz in m Gruppen**, lasse den kNN-Algorithmus m-mal durchlaufen, wobei **jede der m Gruppen über alle m Durchläufe genau einmal Trainings- und genau einmal Testdatensatz** war
- Berechne den **Durchschnitt der Genauigkeits-Scores** aller m Durchläufe

Durchgänge	Teildatensätze 1	2	3	4	5
1.	Training	Training	Training	Training	Test
2.	Training	Training	Training	Test	Training
3.	Training	Training	Test	Training	Training
4.	Training	Test	Training	Training	Training
5.	Test	Training	Training	Training	Training

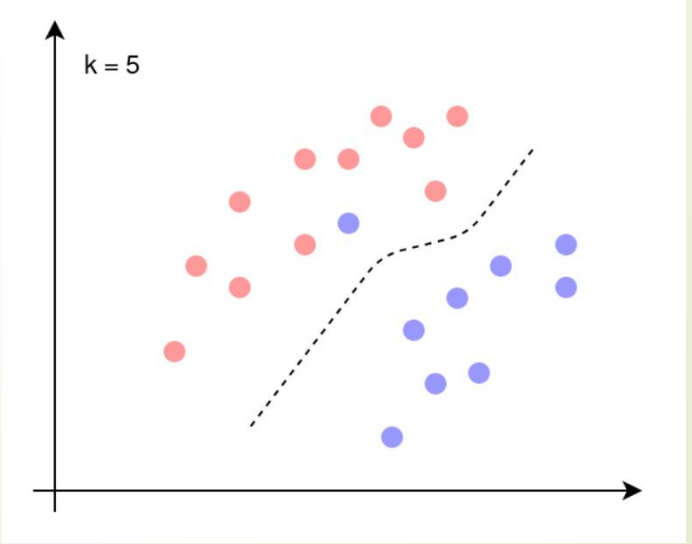
Beispiel für $m = 5$

Was ist ein optimales k?

- Der Punkt in der "Schneiße" wird als blau klassifiziert, obwohl er insgesamt den roten Punkten näher ist; niedriges k hat die Klassifikationsgrenze "zackig" warden lassen
  - **Niedrige Werte von k machen die Klassifikationsgrenze anfällig für Varianz!**



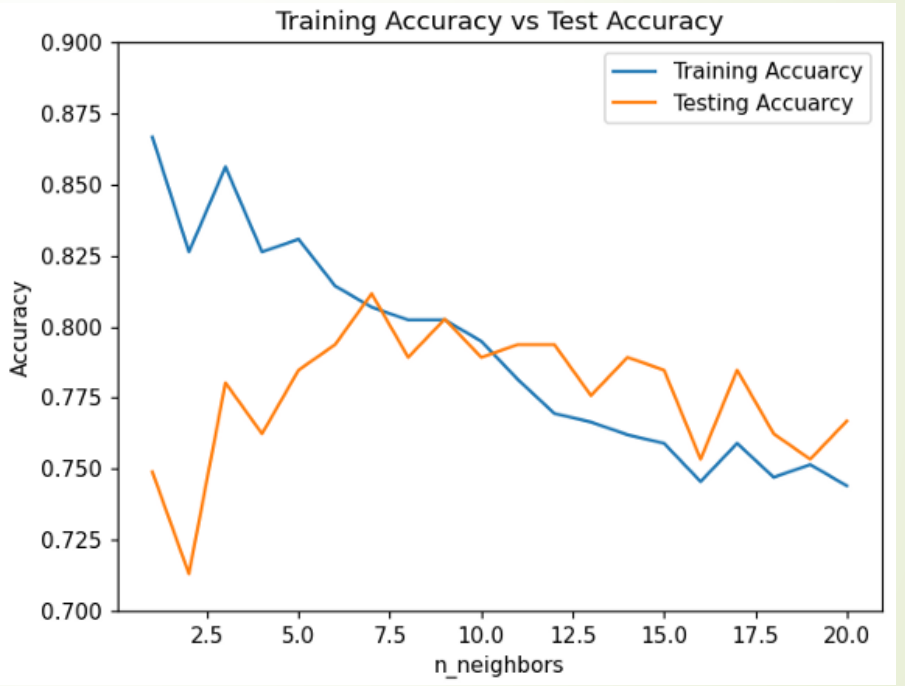
- Höhere Werte von k betrachten die Daten in einem größeren Kontext; höheres k hat die Klassifikationsgrenze "weicher" warden lassen
  - **Zwar verliert man etwas Genauigkeit, macht die Klassifikationsgrenze aber robuster für Ausreißer!**



Overfitting vs. Underfitting vs. Idealer Fit

Verwechslungsgefahr Genauigkeit Trainingsdaten vs. Genauigkeit Testdaten:

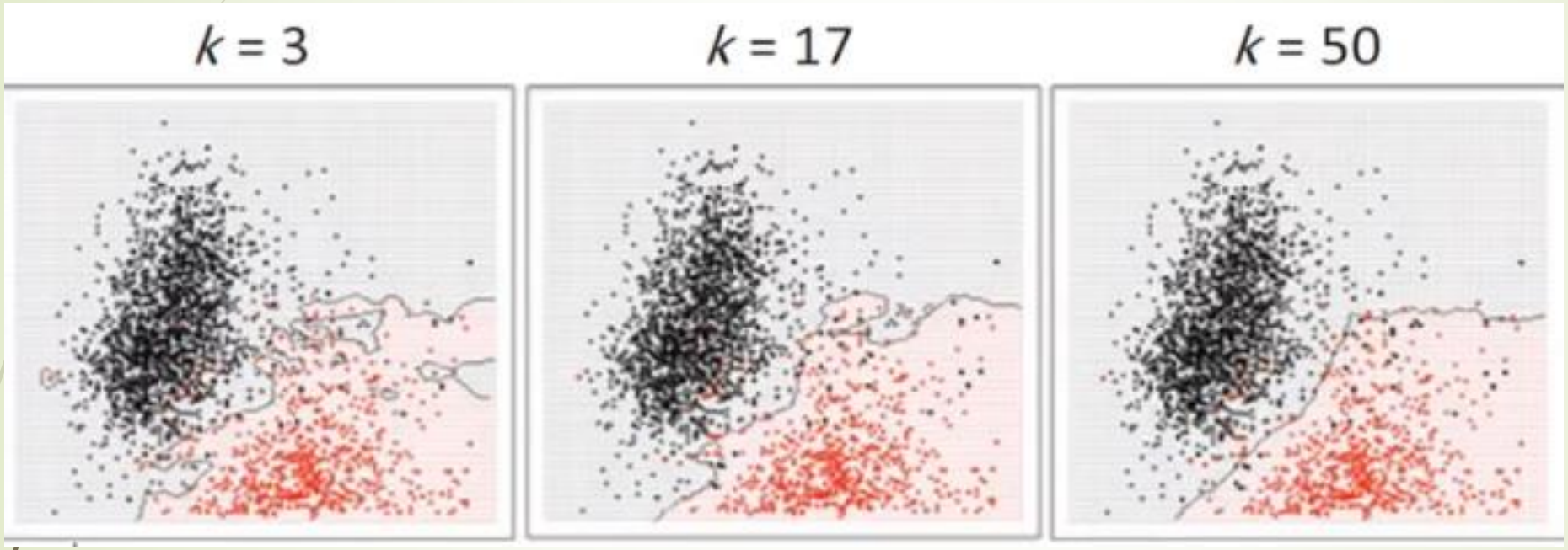
- natürlich ist die **Genauigkeit in den Trainingsdaten bei niedrigen k immer höher** (bei k=1 im Extremfall 100%), da man die Trainingsdaten perfekt angepasst hat
- ein solches Modell ist aber **unnütz für neue Daten, da jeder Zufallseinfluss systematisiert wurde**
- ideal ist also ein **mittleres k**, bei dem Überanpassung (Zufall/Ausreißer sind System) und Unteranpassung (wichtige Trends bleiben unerkannt) balanciert sind



Problemstellung
Vorgehensweise
Algorithmus
Optimierung der Genauigkeit
Diskussion

Overfitting vs. Underfitting vs. Idealer Fit

Wein-Beispiel:



Overfitting | Idealer Fit | Underfitting

- Optimales k kann mit **Hyperparameter-Tuning** gefunden werden, d.h. für jede Kreuzvalidierung werden k durchlaufen, und dasjenige mit höchster Genauigkeit zurückgegeben

Diskussion und Fazit

- kNN ist ein einfacher Algorithmus („lazy learning“), der **Datenpunkt an Hand der Werte benachbarter Datenpunkte klassifiziert**
- k ist die **Anzahl der** betrachteten, **benachbarten Datenpunkte**, **sinnvolle Werte** für diesen Parameter erhält man über **k-fache Kreuzvalidierung**
- kNN funktioniert am Besten mit wenig Prädiktoren und Klassen vergleichbarer Größe. Falsche Klassifizierungen können Hinweise auf Ausreißerwerte geben.
- kNN eignet sich also nicht nur zur **Vorhersage von Datenwerten** oder zur **Klassifikation von Datenpunkten**, sondern man kann mit kNN auch **Ausreißer erkennen**, wodurch sich bspw. Betrugsversuche aufdecken lassen

Problemstellung
Vorgehensweise
Algorithmus
Optimierung der Genauigkeit
Diskussion