



Hans-Sachs-Gymnasium Nürnberg  
Studienseminar Februar 2024/2026

S C H R I F T L I C H E   H A U S A R B E I T  
zur zweiten Staatsprüfung  
für das Lehramt an Gymnasien  
im Fach Informatik

**Thema:**

ENTWICKLUNG UND ERPROBUNG EINER WEB-ANWENDUNG ZUR VISUALISIERUNG  
DES K-NÄCHSTE-NACHBARN-ALGORITHMUS MIT VARIABLEN PARAMETERN  
FÜR DIE 11. KLASSE

EINGEREICHT VON: StRefin Daniela Andres  
GEBOREN AM: 21.01.2000  
FÄCHERVERBINDUNG: Mathematik/Informatik  
VORGLEGT AM: 06.08.2025  
SEMINARLEHRER: StR Dominik Lörzel



# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>5</b>
<b>2 Fachlicher Hintergrund</b>	<b>6</b>
2.1 Entwicklung des KNN-Algorithmus . . . . .	6
2.2 Metriken . . . . .	8
2.3 Normalisierung . . . . .	9
<b>3 Schulische Anforderungen</b>	<b>10</b>
3.1 Use Case 1: Laden verschiedener Datensätze . . . . .	11
3.2 Use Case 2: Visualisierung der Trainings- und Testdaten . . . . .	12
3.3 Use Case 3: Variation der Parameter des KNN-Algorithmus . . . . .	12
3.4 Use Case 4: Visualisierung des Einflusses von Trainingsdaten und Parametern	12
<b>4 Aufbau und Funktionsweise von KNNchanted</b>	<b>12</b>
4.1 Darstellungsbereich . . . . .	13
4.2 Schaltflächenbereich . . . . .	14
4.3 Qualitätseinschätzung . . . . .	15
<b>5 Didaktische Entscheidungen bei der Entwicklung</b>	<b>16</b>
5.1 KNNchanted als Webanwendung . . . . .	16
5.2 Normalisierung . . . . .	16
5.3 Darstellung im Koordinatensystem . . . . .	17
5.4 Metriken . . . . .	18
5.5 Schülereingaben . . . . .	18
5.6 Wahl der Qualitätseinschätzung . . . . .	19
5.7 Beispieldatensätze . . . . .	19
<b>6 Erprobung des Programms</b>	<b>21</b>
6.1 Charakterisierung der Lerngruppe . . . . .	21
6.2 Planung der Unterrichtssequenz . . . . .	22
6.3 Durchführung und Reflexion der Unterrichtssequenz . . . . .	23

<b>7</b>	<b>Bewertung</b>	<b>25</b>
7.1	Verifikation der Use Cases . . . . .	25
7.2	Limitierungen . . . . .	26
<b>8</b>	<b>Implementierung</b>	<b>27</b>
8.1	Umsetzung des KNN-Algorithmus . . . . .	27
8.2	Realisierung der Benutzeroberfläche . . . . .	28
<b>9</b>	<b>Ausblick</b>	<b>29</b>
<b>10</b>	<b>Eigenständigkeitserklärung</b>	<b>32</b>
<b>11</b>	<b>Anhänge</b>	<b>A1</b>
	Anhang A: Aufgabenstellung der Arbeitsblätter zur Unterrichtssequenz des KNN-Algorithmus. . . . .	A1
	Anhang B: Musterlösung der Arbeitsblätter zur Unterrichtssequenz des KNN-Algorithmus. . . . .	A7

# 1 Einleitung

Informatiksysteme, die auf Methoden der künstlichen Intelligenz (KI) basieren, sind längst fester Bestandteil unseres Alltags geworden. Seien es die angezeigten Produktempfehlungen auf Instagram, Dialogsysteme wie ChatGPT oder Übersetzungsprogramme wie DeepL – hinter vielen alltäglichen Anwendungen steckt maschinelles Lernen. Aus dieser Entwicklung folgt für die Schulen die Notwendigkeit, sowohl technische Grundlagen für Verfahren des maschinellen Lernens zu vermitteln als auch ein Grundverständnis für Chancen, Risiken und Grenzen dieser Algorithmen zu schaffen. Der bayerische Lehrplan für die gymnasiale Oberstufe trägt diesem Anspruch Rechnung, indem er in der elften Jahrgangsstufe für fast alle Zweige die vertiefte Behandlung mindestens eines konkreten Algorithmus des maschinellen Lernens vorsieht.

Ein explizit genannter Algorithmus ist der K-Nächste-Nachbarn-Algorithmus (KNN) – ein didaktisch gut zu reduzierendes Verfahren zur Klassifikation von Datenpunkten, an dem das grundsätzliche Vorgehen von Algorithmen des maschinellen Lernens deutlich gemacht werden kann. Dabei hängt die Qualität der Klassifikation wesentlich von verschiedenen Parametern wie dem gewählten  $k$ , der verwendeten Metrik sowie der Anzahl, Qualität und Verteilung der Trainingsdaten ab. Ziel dieser Arbeit ist es, eine Webanwendung zur Visualisierung des KNN-Algorithmus zu entwickeln, die es Schülerinnen und Schüler der elften Jahrgangsstufe erlaubt, den Einfluss dieser Parameter interaktiv zu untersuchen. Der Fokus liegt dabei nicht auf der reinen Vermittlung der Funktionsweise des Algorithmus, sondern auf dem selbstständigen Erforschen seiner verschiedenen Einflussfaktoren anhand konkreter Beispieldatensätze.

Den Beginn dieser Arbeit bildet eine kurze Übersicht über den fachlichen Hintergrund des KNN-Algorithmus. Im darauffolgenden Abschnitt wird auf Grundlage der im Lehrplan formulierten Kompetenzerwartungen analysiert, welche Anforderungen ein Werkzeug zur vertieften Betrachtung des KNN-Algorithmus erfüllen muss. Anschließend wird das im Rahmen dieser Arbeit erstellte Programm „KNNchanted“ vorgestellt. Neben Erklärungen des Aufbaus und der Funktionsweise von KNNchanted wird auf während der Entwicklung getroffene didaktische Entscheidungen eingegangen. Zudem werden die Ergebnisse einer Unterrichtssequenz, in deren Rahmen die Software in einer elften Jahrgangsstufe

getestet wurden, bewertet und ein Ausblick auf mögliche Weiterentwicklungen gegeben. Abschließend wird auf einige Details der Implementierung von KNNchanted eingegangen.

## 2 Fachlicher Hintergrund

### 2.1 ENTWICKLUNG DES KNN-ALGORITHMUS

Historische Vorgänger des K-Nächste-Nachbar-Algorithmus (KNN) waren sogenannte „table lookup“-Algorithmen [5, S. 687]. Um einen neuen Datenpunkt zu klassifizieren, verglichen diese Algorithmen lediglich, ob ein gleicher Datenpunkt bereits in einer gespeicherten Tabelle vorhanden war und wiesen bei Vorhandensein eines entsprechenden Punktes dem neuen Datenpunkt die gleiche Kategorie zu. Die darauf aufbauende Nearest-Neighbour-Methode verallgemeinert dieses Prinzip, indem sie nicht nach einem gleichen, sondern einem möglichst ähnlichen Punkt in den vorhandenen Daten sucht. Ähnliche Datenpunkte sollten möglichst auch in der gleichen Kategorie klassifiziert werden. Klassischerweise definiert man die Ähnlichkeit von Datenpunkten über Abstände: Zwei Datenpunkte „sind umso ähnlicher, je geringer ihr Abstand im Merkmalsraum ist“ [2, S. 218]. Grafisch veranschaulichen kann man diese Klassifizierung mit einem Voronoi-Diagramm. Bei diesem wird jeder Datenpunkt mit einem konvexen Polygon umgeben. Das Polygon besteht aus allen Punkten, die in Bezug auf eine verwendete Metrik, klassischerweise die euklidische Metrik, näher an dem Datenpunkt in der Mitte des Polygons liegen als an jedem anderen Datenpunkt. Zeichnet man die Linien des Polygons nach, wenn die benachbarten Polygone nicht die gleiche Kategorie haben, erhält man Trennlinien, die die einzelnen Kategorien voneinander abgrenzen [2, S. 219]. Damit ist die Nearest-Neighbour-Methode in der Lage, beliebig komplexe Trennlinien zwischen Kategorien zu realisieren. Ein Nachteil der Methode ist ihre Anfälligkeit gegenüber Ausreißern in den zur Verfügung stehenden Daten. Da sie lediglich den nächsten Nachbarn betrachtet, können einzelne statistische Ausreißer die Qualität der Trennlinie stark beeinflussen. Der KNN-Algorithmus versucht dieses Problem zu umgehen, indem statt lediglich eines nächsten Nachbarn die  $k$  nächsten Nachbarn,  $k \in \mathbb{N}$ , betrachtet werden. Für den neu zu klassifizierenden Punkt entscheiden die  $k$  ähnlichsten Datenpunkte des bekannten Datensatzes in einem Mehrheitsentscheid über die zugewiesene Kategorie [2, S. 220]. Der untenstehende Pseudocode fasst die Klassifizierung

eines neuen Datenpunktes durch den KNN-Algorithmus zusammen:

- 1) Berechnung der Abstände des neuen Punktes zu allen bekannten Datenpunkten.
- 2) Sortierung der Paare aus berechnetem Abstand und bekanntem Datenpunkt aufsteigend nach dem Abstand.
- 3) Entscheidung über Kategorie des neuen Punktes anhand Mehrheitsentscheid der  $k$  ersten bekannten Datenpunkte.

Beim Mehrheitsentscheid kann es je nach Anzahl der Kategorien und dem verwendeten  $k$  zu einem Gleichstand verschiedener Kategorien kommen. Dies scheint in manchen Fällen kontraintuitiv, wie in folgendem Extrembeispiel: Es existieren zwei Trainingsdatenpunkte  $A(1, 2)$  und  $B(2, 2)$  der Kategorien rot und grün. Der Testdatenpunkt  $C(2, 1)$  befindet sich mit Abstand 1 näher an dem Trainingsdatenpunkt  $B$  als an Trainingsdatenpunkt  $A$ , wird jedoch bei der Wahl von  $k = 2$  keiner Kategorie zugewiesen, da in der Abstimmung Gleichstand herrscht. Diese Zuweisung erscheint auf den ersten Blick kontraintuitiv, kann aber durch eine passendere Wahl von  $k = 1$  ausgeglichen werden. Als Erweiterung des KNN-Algorithmus besteht hier aber zusätzlich die Option beim Mehrheitsentscheid eine Gewichtung der Stimmen durchzuführen. Bei größer werdendem  $k$  existieren typischerweise mehr Nachbarn mit großem Abstand als solche mit kleinem Abstand, weshalb die weiter entfernten Nachbarn einen vergleichsweise großen Einfluss auf das Abstimmungsergebnis haben [2, S. 223]. Um diesen Einfluss auszugleichen, erhält die Stimme des nächsten Nachbarn  $x_i$ ,  $1 \leq i \leq k$ , das Gewicht

$$w_i = \frac{1}{1 + \alpha \cdot d(x, x_i)^2},$$

wobei  $x$  der aktuell zu klassifizierende Punkt,  $d$  die verwendete Metrik und  $\alpha$  eine Konstante ist, die beeinflusst, wie schnell die Gewichte mit dem Abstand abnehmen [2, S. 223]. Oft wird  $\alpha$  deshalb auch als Abnahmgeschwindigkeit bezeichnet.

Da sowohl beim KNN-Algorithmus als auch bei den vorgestellten Nearest-Neighbour-Methoden in der Lernphase des Algorithmus außer dem Abspeichern der Trainingsdaten nichts passiert, bezeichnet man solche Algorithmen auch als Lazy-Learning-Verfahren [2, S. 225].

Der KNN-Algorithmus eignet sich somit insbesondere für Problemstellungen, bei denen eine präzise lokale Approximation (beispielsweise durch Trennlinien) erforderlich ist, gleichzeitig jedoch keine hohen Anforderungen an die Ausführungsgeschwindigkeit bestehen. Darüber hinaus ist zu beachten, dass die benötigte Menge an Trainingsdaten mit zunehmender Anzahl der Kategorien signifikant ansteigt [2, S. 221].

Abschließend sei erwähnt, dass der KNN-Algorithmus nicht nur auf Klassifikationsprobleme, sondern auch auf Regressionsprobleme angewandt werden kann. Da diese Anwendung in der Schule nicht im Vordergrund steht, sei an dieser Stelle lediglich auf [2, S. 223ff] oder [5, S. 691ff] verwiesen.

## 2.2 METRIKEN

Der Abstand von zwei Punkten  $a \in \mathbb{R}^n$  und  $b \in \mathbb{R}^n$  kann auf unterschiedliche Weise bestimmt werden [2, S. 261]. Die wohl bekannteste Metrik ist der euklidische Abstand:

$$d_e(a, b) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2},$$

welcher sich für zwei Punkte  $P = (x, y) \in \mathbb{R}^2$  und  $Q = (x', y') \in \mathbb{R}^2$  zu

$$d_e(P, Q) = \sqrt{(x - x')^2 + (y - y')^2}$$

vereinfachen lässt. Oft wird auch der einfacher zu berechnende Manhattan-Abstand

$$d_m(a, b) = \sum_{i=1}^n |a_i - b_i|$$

verwendet, welcher sich mit analoger Notation im zweidimensionalen ebenfalls noch weiter vereinfacht:

$$d_m(P, Q) = |x - x'| + |y - y'|.$$

In vielen Anwendungen sind bestimmte Merkmale wichtiger als andere. Daher ist es oft sinnvoll, die Merkmale durch Gewichte  $w_i \in \mathbb{R}$  unterschiedlich zu skalieren [2, S. 218]. Die

angepassten, allgemeinen Formeln lauten entsprechend:

$$d_e(a, b) = \sqrt{\sum_{i=1}^n w_i \cdot (a_i - b_i)^2}$$

und

$$d_m(a, b) = \sum_{i=1}^n w_i \cdot |a_i - b_i|.$$

### 2.3 NORMALISIERUNG

In der Praxis kann es vorkommen, dass einzelne Merkmale eines Datenpunktes sehr unterschiedliche Wertebereiche haben. Berechnet man den Abstand zwischen solchen Datenpunkten, haben die einzelnen Merkmale mit einem vergleichsweise kleinen Wertebereich einen geringeren Einfluss auf den berechneten Abstand im Vergleich zu den anderen Merkmalen. Dadurch können wichtige Informationen verloren gehen [2, S. 228]. Um dies zu vermeiden, kann eine Normalisierung vorgenommen werden. Dabei werden die Merkmale  $a_i$  eines Datenpunkts  $a \in \mathbb{R}^n$  auf das Intervall  $[0, 1]$  abgebildet. Dies kann durch die sogenannte Min-Max-Normalisierung geschehen, bei der sich das normalisierte Merkmal  $\tilde{a}_i$  durch

$$\tilde{a}_i = \frac{a_i - \min_i}{\max_i - \min_i}$$

berechnet, wobei  $\max_i$  für das Maximum des  $i$ -ten Merkmals steht und  $\min_i$  für das Minimum. Diese Art der Normalisierung kann zu Problemen führen, wenn es in den Trainingsdaten einige wenige Ausreißer mit Extremwerten gibt, die dadurch ebenfalls in das Intervall  $[0, 1]$  skaliert werden und die Berechnung aller anderen normalisierten Werte beeinflussen [2, S. 229].

Als Alternative existiert die Standardisierung unter Verwendung von Mittelwert  $\mu_i$  und Standardabweichung  $\sigma_i$  der Merkmale mittels

$$\tilde{a}_i = \frac{a_i - \mu_i}{\sigma_i}.$$

Hier liegen typischerweise nicht mehr alle normalisierten Punkte in dem festen Intervall  $[0, 1]$ , schwanken aber trotzdem um Null mit Standardabweichung Eins [2, S. 229].

Zum Abschluss sei erwähnt, dass eine Normalisierung im Allgemeinen nicht durch

eine individuelle Gewichtung der Merkmale ersetzt werden kann. Eine Gewichtung der Merkmale verändert die Einflussstärke der Merkmale bei der Berechnung der Abstände zwischen einzelnen Punkten. Sie bewirkt keine einheitliche Skalierung, sondern eine inhaltliche Umgewichtung der Merkmale. Eine Normalisierung hingegen passt lediglich die Wertebereiche der Merkmale an, ohne den Informationsgehalt zu verzerren.

### 3 Schulische Anforderungen

Um die schulischen Anforderungen für ein Werkzeug im Bereich der künstlichen Intelligenz in Jahrgangsstufe 11 zu erarbeiten, werden zunächst die verschiedenen Lehrpläne für Informatik in dieser Jahrgangsstufe herangezogen. Sowohl bei Informatik 11 (NTG) als auch bei spätbeginnender Informatik (HG, SG, MuG, SWG) formuliert der bayerische Lehrplan eine ähnliche Kompetenzerwartung:

„Die Schülerinnen und Schüler [...] erläutern die Funktionsweise eines ausgewählten Algorithmus maschinellen Lernens (k-nächste-Nachbarn-Algorithmus oder Entscheidungsbaum-Algorithmus) allgemein und an konkreten Beispielen.“ [8]

Während auf vertieftem Niveau die Erläuterung der Funktionsweise gefordert ist, wird auf dem nicht vertieften Niveau lediglich die Erläuterung der „Idee“ [10] eines Algorithmus des maschinellen Lernens gefordert. Der KNN-Algorithmus wird dabei explizit als Unterrichtsinhalt vorgeschlagen. Zusätzlich soll in beiden Informatikkursen ein Algorithmus des maschinellen Lernens vertieft behandelt werden:

„Die Schülerinnen und Schüler [...] analysieren den Einfluss von Trainingsdaten und Parametern auf die Zuverlässigkeit der Ergebnisse eines Verfahrens maschinellen Lernens, ggf. unter Verwendung eines geeigneten Werkzeugs.“ [8, 10]

Daher motiviert sich die Entwicklung eines Werkzeugs, mit dem der Einfluss von Trainingsdaten und Parametern auf das Ergebnis der Klassifizierung des KNN-Algorithmus beobachtet und analysiert werden kann. Die Anforderungen an ein solches Programm

können direkt aus den Kompetenzerwartungen des Lehrplans abgeleitet werden: Um den Einfluss von Trainingsdaten und Parametern analysieren zu können, muss eine geeignete Visualisierung der Klassifizierung der Trainings- und Testdatenpunkte zur Verfügung gestellt werden. Gleichzeitig fordert die Kompetenzerwartung, dass die Schülerinnen und Schüler in der verwendeten Software die Parameter des KNN-Algorithmus, also insbesondere das verwendete  $k$  sowie die verwendete Metrik, verändern können. Auch die Trainingsdaten beeinflussen die Klassifizierung des KNN-Algorithmus. Eine hohe Anzahl von Trainingsdaten verringert üblicherweise die Fehlerquote der Klassifizierung, erhöht aber auch die benötigte Rechenzeit des Algorithmus. Ist die Anzahl im Vergleich zu den möglichen Kategorien zu gering, kann dies zu Underfitting führen, der KNN-Algorithmus kann neue Datenpunkte nicht ausreichend genau klassifizieren. Auch die Qualität des Trainingsdatensatzes beeinflusst die Klassifikation. Hat ein Trainingsdatensatz viele Ausreißerdatenpunkte, kann es durch ein zu geringes  $k$  zu einer Überanpassung an die Ausreißerpunkte kommen, wodurch die Güte der Klassifizierung abnimmt. Dies wird auch als Overfitting bezeichnet. Eine ungleichmäßige Verteilung der Trainingsdaten, in der eine oder mehrere Kategorien überrepräsentiert sind, führt zu unzuverlässigen Ergebnissen, da insbesondere für höhere Werte von  $k$  ein neuer Datenpunkt mit einer höheren Wahrscheinlichkeit einer der dominierenden Kategorien zugeordnet wird. Diese Betrachtungen führen zu folgenden Anforderungen an die Trainingsdatensätze des Programms: Die Datensätze müssen klein genug sein, damit eine schnelle Berechnung der Klassifizierung möglich ist. Gleichzeitig sollen die Datensätze in den Parametern Anzahl, Qualität und Verteilung variieren, um die genannten Effekte demonstrieren zu können.

Insgesamt ergeben sich aus diesen Betrachtungen mindestens die folgenden Anforderungen an das Programm:

### 3.1 USE CASE 1: LADEN VERSCHIEDENER DATENSÄTZE

Den Schülerinnen und Schüler soll eine Auswahl verschiedener Datensätze zur Verfügung stehen, die sie in das Programm laden und mit diesem klassifizieren können. Jeder Datensatz besteht dabei aus einem Trainingsdatensatz und einem Testdatensatz, um Overfitting- und Underfitting-Effekte erkennen zu können. Gleichzeitig sollen die angebotenen Datensätze in den Parametern Anzahl, Qualität und Verteilung variieren.

### **3.2 USE CASE 2: VISUALISIERUNG DER TRAININGS- UND TESTDATEN**

Trainings- und Testdaten sollen in einem passenden Koordinatensystem angezeigt werden. Durch ihre grafische Darstellung sollen Trainings- und Testdaten voneinander unterscheidbar sein.

### **3.3 USE CASE 3: VARIATION DER PARAMETER DES KNN-ALGORITHMUS**

Die Schülerinnen und Schüler sollen den Parameter  $k$  der betrachteten nächsten Nachbarn individuell einstellen können. Zudem sollen verschiedene Metriken zur Bestimmung des Abstandes zur Verfügung stehen.

### **3.4 USE CASE 4: VISUALISIERUNG DES EINFLUSSES VON TRAININGSDATEN UND PARAMETERN**

Die durch den KNN-Algorithmus ermittelte Kategorie der Testdatenpunkte soll sichtbar gemacht werden. Damit können die Schülerinnen und Schüler den Einfluss der Parameter auf die Klassifizierung erkennen. Zusätzlich soll eine Konfusionsmatrix angezeigt werden, um die Qualität der Klassifizierung einschätzen zu können.

Nicht von den Anforderungen an das Programm umfasst sind beispielsweise die Bestimmung des „optimalen“  $k$  sowie die Validierung es gewählten  $k$ . Auch ist es nicht möglich den bereits geladenen Datensatz nachträglich im Programm zu verändern. Insbesondere ist es nicht Ziel des Programms das Prinzip des KNN-Algorithmus zu vermitteln. Grundlegendes Wissen über die Funktionsweise des Algorithmus wird vom Nutzer des Programms bereits erwartet.

## **4 Aufbau und Funktionsweise von KNNchanted**

Auf Grundlage dieser Anforderungsanalyse wurde das Programm KNNchanted entwickelt. KNNchanted ist eine Webanwendung, die speziell zur Erfüllung der erarbeiteten Use Cases entwickelt wurde. Im Folgenden wird das Programm vorgestellt.

KNNchanted ist in einen oberen Darstellungsbereich, einen mittleren interaktiven

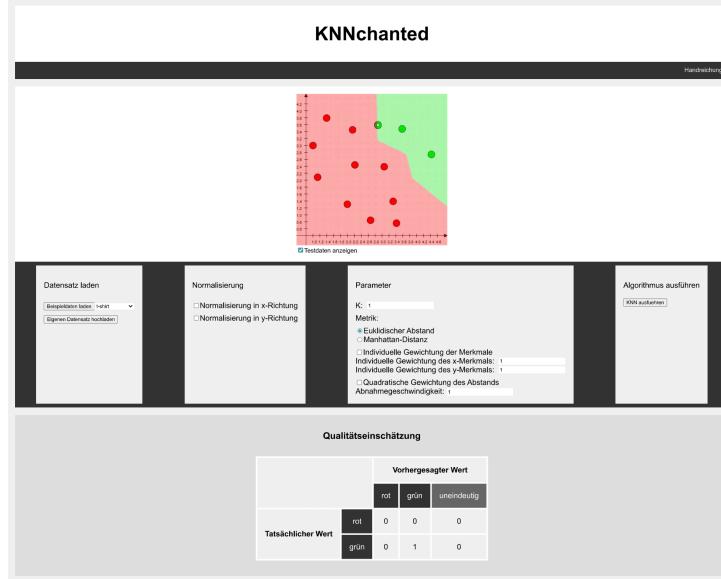


Abbildung 1. Oberfläche von KNNchanted

Schaltflächenbereich und einen unteren Bereich zur Qualitätseinschätzung aufgeteilt.

#### 4.1 DARSTELLUNGSBEREICH

Der Darstellungsbereich enthält ein Koordinatensystem, in dem die geladenen Trainings- und Testdaten angezeigt werden können. Die Farben der punktförmigen Trainingsdaten zeigen die zugehörige Kategorie an, welche dem KNN-Algorithmus auch bekannt ist. Über eine Auswahlschaltfläche kann die Anzeige der Testdaten aktiviert werden. Testdatenpunkte werden als Kreisringe dargestellt. Wieder codiert die Farbe der Testdatenpunkte die zugehörige Kategorie, die dem KNN-Algorithmus allerdings nicht bekannt ist. Die Klassifizierung wird über die Anzeige der Trennlinien der Kategorien dargestellt. Diese werden durch eine Aufteilung des Hintergrunds in  $256 \times 256$  quadratische Felder erzeugt, welche entsprechend der Kategorisierung ihres Zentrumspunkts durch den KNN-Algorithmus eingefärbt werden. Dafür legt sich eine transparente Ebene über die Datenpunkte, damit die entstehenden Trennlinien nicht von den Datenpunkten verdeckt werden. Wird dem Zentrumspunkt keine eindeutige Kategorie zugewiesen, wird das entsprechende Quadrat weiß gefärbt.

## 4.2 SCHALTFLÄCHENBEREICH

Die Interaktion mit dem Programm erfolgt über den Schaltflächenbereich. Der vorgesehene Arbeitsablauf erfolgt in vier Zonen von links nach rechts: In der ersten Zone „Datensatz laden“ kann über ein Auswahlmenü ein vorgefertigter Beispieldatensatz vom Server geladen werden. Die im Programm hinterlegten Datensätze heißen „anzahl\_gross“, „anzahl\_klein“, „ausreisser“, „drei\_punkte“, „geraete“, „t-shirt“, „verteilung“ und „zwei\_ringe“. Die Datensätze variieren in den in Abschnitt 3 genannten Parametern Anzahl, Qualität und Verteilung und dienen dazu, den Einfluss dieser Parameter zu veranschaulichen. Auch ist es möglich einen eigenen Datensatz von der Festplatte zu laden. Die Datensätze müssen in einer unformatierten Textdatei (.txt) nach folgendem Schema gespeichert werden: In der ersten Zeile stehen die Merkmale, die auf der x- und der y-Achse aufgetragen werden sollen sowie das Merkmal nach dem klassifiziert werden soll. In der zweiten Zeile werden alle vorkommenden Kategorien angegeben. Die Reihenfolge, in der die Kategorien hier angegeben werden, bestimmt die Reihenfolge, in der sie später in der Konfusionsmatrix auftreten werden. In der dritten Zeile steht lediglich der Eintrag „Trainingsdaten“, woraufhin in den weiteren Zeilen alle vorhandenen Trainingsdaten folgen. Diese werden nach dem in der ersten Zeile angegebenen Muster notiert. Schlussendlich können noch die Testdaten angegeben werden. Dazu schreibt man in Anschluss an den letzten Trainingsdatensatz das Wort „Testdaten“ und gibt darunter alle Testdaten analog zu den Trainingsdaten an. Alle Angaben werden grundsätzlich durch ein Semikolon getrennt, als Dezimaltrennzeichen wird der Punkt verwendet. Zur Veranschaulichung sei hier ein Ausschnitt aus dem Beispieldatensatz „geraete“ gegeben:

```
Breite in mm;Hoehe in mm;Geraeteart  
Mobiltelephon;Tablet;Computer  
Trainingsdaten  
75.7;160.1;Mobiltelephon  
...  
287;181;Computer  
Testdaten  
77.4;157.5;Mobiltelephon  
...
```

Die Farben, die den Kategorien zugewiesen werden, sind fest vorgegeben und können nicht individuell angepasst werden. Sie folgen der festen Reihenfolge: Rot, Grün, Blau, Gelb, Magenta, Cyan, Grau, Lila und Orange. Ab der zehnten Kategorie wiederholen sich die Farben.

Wird sich an das beschriebene Format gehalten, ist KNNchanted in der Lage auch unbekannte Datensätze einzulesen und zu verarbeiten. Damit existiert für Lehrkräfte auch die Möglichkeit, einen individuell mit der Klasse entwickelten Datensatz mit KNNchanted vertieft zu behandeln.

Die zweite Zone „Normalisierung“ stellt zwei Auswahlschaltflächen zur Normalisierung der x- bzw. y-Koordinate zur Verfügung. Diese Normalisierung ist optional und kann bei Datensätzen mit ähnlichen Wertebereichen in beiden Koordinaten übersprungen werden, ohne die Qualität der Klassifizierung signifikant negativ zu beeinflussen.

In der Zone „Parameter“ können die Anzahl der betrachteten nächsten Nachbarn  $k$  sowie die verwendete Metrik ausgewählt werden. Der Parameter  $k$  kann durch ein Eingabefeld frei eingegeben werden. Für die Metrik stehen zwei Optionen – der euklidische Abstand und die Manhattan-Distanz – zur Verfügung. Auch ist es möglich die Merkmale individuell zu gewichten. Voreingestellt sind hier die Gewichte 1, welche aber ebenfalls durch direkte Eingabe in Eingabefelder angepasst werden können. Als letzter Parameter des KNN-Algorithmus kann über eine weitere Checkbox die quadratische Gewichtung der Stimmen aktiviert werden. Der Parameter  $\alpha$ , der die Abnahmgeschwindigkeit der Gewichte bestimmt, kann ebenfalls durch ein Textfeld direkt eingegeben werden.

Die letzte Zone „Algorithmus ausführen“ stellt eine Schaltfläche zur Verfügung, mit der der KNN-Algorithmus für die geladenen Daten und die eingestellten Parameter ausgeführt wird. Die berechnete Klassifizierung wird im Koordinatensystem des Darstellungsbereichs angezeigt. Zusätzlich wird die Konfusionsmatrix befüllt.

#### 4.3 QUALITÄTSEINSCHÄTZUNG

Zur Qualitätseinschätzung steht unter dem Schaltflächenbereich eine Konfusionsmatrix zur Verfügung. Zusätzlich zu dem üblichen Aufbau einer Konfusionsmatrix, wird in KNNchanted eine zusätzliche Spalte „uneindeutig“ eingeführt. Falls ein Testdatenpunkt nicht eindeutig zu einer einzigen Kategorie zugewiesen werden kann, wird er in dieser

Restkategorie aufgeführt. Die Reihenfolge der restlichen angezeigten Kategorien entspricht der Reihenfolge wie sie in dem in Unterabschnitt 4.2 beschriebenen Format angegeben wurden.

## 5 Didaktische Entscheidungen bei der Entwicklung

Im Folgenden sollen einige didaktische Entscheidungen bei der Entwicklung von KNNchanted genauer begründet werden. Zunächst wird auf die Umsetzung von KNNchanted als Webanwendung eingegangen. Daraufhin werden die verwendete Normalisierung, die im Koordinatensystem gewählte Visualisierung der Datenpunkte, die Metriken, das Behandeln von Schülereingaben sowie die Wahl der Qualitätseinschätzung erklärt. Abschließend folgt eine Beschreibung und didaktische Begründung der beigefügten Beispieldatensätze.

### 5.1 KNNCHANTED ALS WEBANWENDUNG

Die Entscheidung für eine Webanwendung anstatt eines lokal ausführbaren Programms hatte verschiedene Gründe: Zum einen entfällt bei einer Webanwendung der Prozess des Herunterladens (und eventueller Installation) des Programms, welcher sonst von der Lehrkraft in Vorbereitung auf die Stunde organisiert oder alternativ in der Stunde von den Schülerinnen und Schülern durchgeführt werden müsste. Beide Möglichkeiten kosten Zeit und verringern so die Wahrscheinlichkeit, dass KNNchanted auch in der Schulpraxis zur Anwendung kommt. Zum anderen sind Aktualisierungen und Anpassungen von KNNchanted als Webanwendung einfacher möglich. Änderungen sind sofort für alle Nutzer verfügbar und müssen nicht erst erneut für jeden Rechner heruntergeladen werden. Da das Programm noch nicht ausreichend in der Schulpraxis getestet wurde, sind in der Zukunft Anpassungen noch wahrscheinlich.

### 5.2 NORMALISIERUNG

Als Normalisierung wurde die Min-Max-Normalisierung anstatt der Normalisierung über die Standardabweichung gewählt. Das grundlegende Prinzip der Min-Max-Normalisierung kann durch einige einfache Beispielrechnungen gut vermittelt werden und ist demnach für

die Schülerinnen und Schüler leicht zugänglich. Die für den KNN-Algorithmus fachlich passendere Normalisierung wäre die Normalisierung über die Standardabweichung. Der Begriff der Standardabweichung wird jedoch in der bayerischen gymnasialen Oberstufe im Mathematik erst in der 12. Jahrgangsstufe im Kapitel „M12 2 Zufallsgrößen und Binomialverteilung“ [9] eingeführt. Ein Behandeln der Standardabweichung in Informatik in der vorhergehenden Jahrgangsstufe ist nicht vorgesehen und würde diesen Lehrplaninhalt vorwegnehmen.

### 5.3 DARSTELLUNG IM KOORDINATENSYSTEM

Damit Trainings- und Testdatenpunkte voneinander unterscheidbar sind, werden Trainingsdatenpunkte als farbige Kreise und die Testdatenpunkte als farbige Kreisringe dargestellt. Durch die Darstellung der Testdatenpunkte kann mithilfe der Färbung in der Leerstelle die durch den KNN-Algorithmus zugewiesene Kategorie leicht abgelesen werden.

Ziel bei der Entwicklung von KNNchanted war es zudem, das Koordinatensystem so zu skalieren, dass die eingegebenen Daten grafisch gut lesbar dargestellt werden können, ohne dabei eine Normalisierung vorwegzunehmen. Die Skalierung soll die Datenpunkte angemessen abbilden, jedoch nicht den Eindruck erwecken, als seien die Werte bereits transformiert oder standardisiert worden. Zur Ermittlung geeigneter Abstände zwischen den Achsenstrichen wird folgender algorithmischer Ansatz verwendet: Zunächst wird das Maximum der Spannweite der Merkmale der x- und y-Achse berechnet. Dieser Maximalwert wird als Referenz für die weitere Skalierung beider Achsen verwendet, weshalb bei einem großen Unterschied zwischen den Wertebereichen der x- und y-Achsen eine Normalisierung immer noch notwendig ist. Um die grafische Lesbarkeit zu erhöhen, werden beide Achsen so skaliert, dass die Abstände zwischen den Achsenstrichen stets Teil der logarithmisch abgestuften Folge der Zahlen 1, 2, 5 und 10 sind. Durch die Verwendung dieser standardisierten Teilung wird gewährleistet, dass die Achsenbeschriftung sowohl bei kleinen als auch bei großen Wertebereichen gut lesbar bleibt und keine willkürlich wirkenden oder schwer interpretierbaren Skalenschritte entstehen.

## 5.4 METRIKEN

Als Metriken wurden in KNNchanted lediglich der euklidische Abstand und die Manhattan-Distanz umgesetzt. Damit wurde sich auf die grundlegendsten und bekanntesten Metriken für zweidimensionale Datensätze aus reellen Zahlen beschränkt. Dies liegt zum einen daran, dass Abstandsmaße zuvor in der Mathematik nie als allgemeines Konstrukt behandelt wurden. Ein Abstandsmaß als austauschbaren Parameter eines Algorithmus anzusehen, ist folglich eine nicht zu unterschätzende Transferleistung. Daraus folgt die Beschränkung auf die beiden genannten Metriken. Der euklidische Abstand ist durch den Satz des Pythagoras aus vorherigen Jahrgangsstufen prinzipiell bereits bekannt und die Manhattan-Distanz kann aufgrund ihrer Einfachheit leicht eingeführt werden. Zum anderen stammen die von KNNchanted verarbeitbaren und anzeigbaren Datensätze alle aus dem  $\mathbb{R}^2$ . Komplexere Metriken, wie die Hamming-Distanz, die zur Messung von Abständen von binären Bit-Vektoren optimiert ist, oder die allgemeine Minkowski-Distanz, die für höherdimensionale Daten vorgesehen ist [5, S. 687f], sind demnach für die von KNNchanted verarbeitbaren Datensätze nicht geeignet. Die Erkenntnis, dass die gewählte Metrik das Ergebnis der Klassifikation des KNN-Algorithmus beeinflusst, kann auch bei nur zwei zur Verfügung stehenden Metriken vermittelt werden.

## 5.5 SCHÜLEREINGABEN

Über vier Textfelder können die Schülerinnen und Schüler direkte Eingaben für Parameter des KNN-Algorithmus machen. Dabei werden die Eingaben in KNNchanted direkt durch Eingabehandler im Hintergrund kontrolliert und korrigiert. So ist es beispielsweise nicht möglich für  $k$  etwas anderes als eine natürliche Zahl einzugeben. Bei den Eingabefeldern für die Gewichtung der Merkmale und der Abnahmgeschwindigkeit sind lediglich positive Dezimalzahlen erlaubt. Dabei kann sowohl ein Komma wie auch ein Punkt als Dezimaltrennzeichen verwendet werden. Durch die Unterstützung beider Schreibweisen können potenzielle Eingabefehler aufgrund unterschiedlicher Dezimaltrennzeichen bereits im Vorfeld vermieden werden.

## 5.6 WAHL DER QUALITÄTSEINSCHÄTZUNG

Ein hilfreiches Visualisierungswerkzeug für die Güte einer Klassifizierung ist eine Konfusionsmatrix – eine zweidimensionale Tabelle, die angibt, wie oft jede Kategorie als richtig oder falsch klassifiziert wird [5, S. 710]. Insbesondere wurde die Konfusionsmatrix in der ISB-Handreichung für die elfte Jahrgangsstufe als Möglichkeit zur Qualitätseinschätzung für Klassifikationsprobleme vorgestellt [7, S. 55ff]. Aufgrund dessen wurde sich bei KNNchanted ebenfalls für eine Konfusionsmatrix zur besseren Einschätzung des Klassifikationsergebnisses entschieden.

## 5.7 BEISPIELDATENSÄTZE

Die Auswahl der Beispieldatensätze hatte zum Ziel, möglichst einfache und gut visualisierbare Datensätze zu erstellen, die die in Abschnitt 3 skizzierten Gedankengänge ermöglichen.

Der Datensatz „anzahl\_klein“ besteht aus fünf Trainingsdatenpunkten und drei Testdatenpunkten, die aus drei Kategorien stammen. Aufgrund der geringen Anzahl an Trainingsdaten — insbesondere im Vergleich zu den vorhandenen Kategorien — ist eine korrekte Klassifizierung der Testdaten nur schwer möglich. Der Datensatz „anzahl\_gross“ enthält den Datensatz „anzahl\_klein“ sowie weitere Trainingsdatenpunkte. Er kann im Anschluss an den Datensatz „anzahl\_klein“ behandelt werden, um die bessere Arbeitsweise des KNN-Algorithmus bei einer angemessenen Menge an Trainingsdatenpunkten aufzuzeigen. Beide Datensätze sind in Abbildung 2 dargestellt.

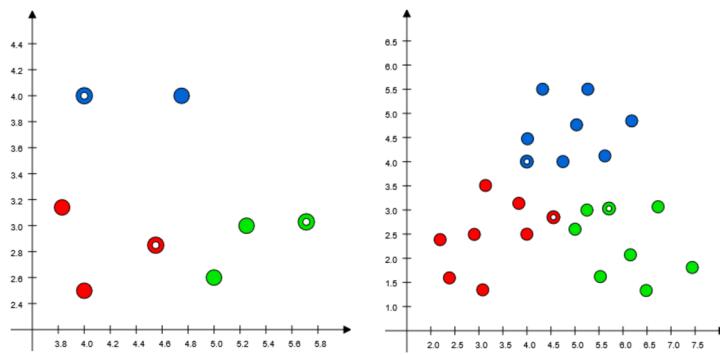


Abbildung 2. Die Datensätze „anzahl\_klein“ und „anzahl\_gross“.

Der Datensatz „ausreisser“ dient dazu, Underfitting- und Overfitting-Effekte zu demon-

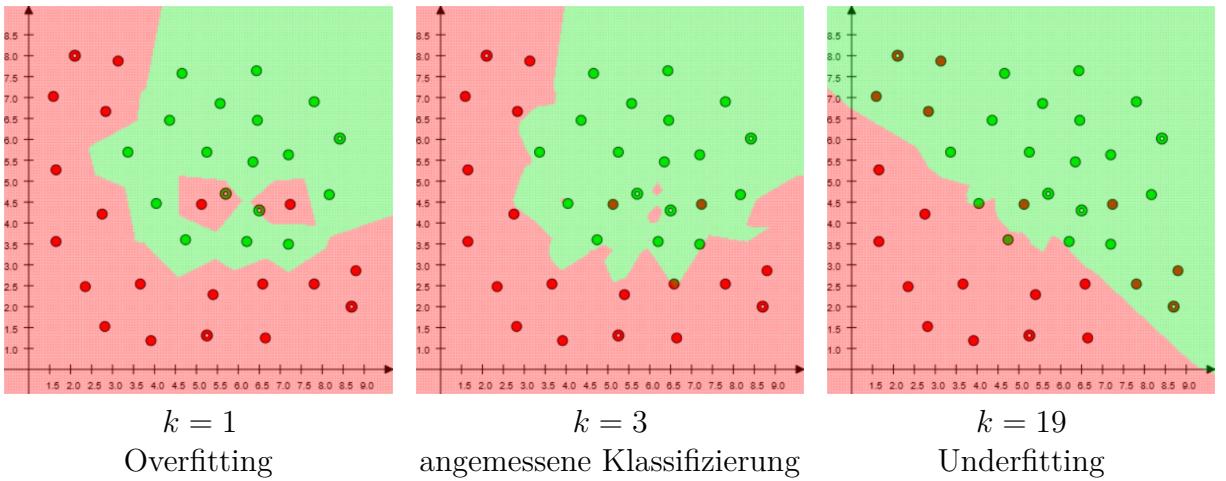


Abbildung 3. Over- und Underfitting-Effekte am Beispiel des „ausreisser“-Datensatzes

trieren. Der Datensatz besteht aus 32 Trainingsdatenpunkten und sechs Testdatenpunkten. Bei den Trainingsdatenpunkten sind deutlich zwei Ausreißer in den Daten erkennbar. Wählt man beispielsweise  $k = 1$  für die Klassifizierung, erzeugt man eine Überanpassung (Overfitting) an die Ausreißerdatenpunkte, wodurch auch die Testdatenpunkte nahe an den Ausreißerpunkten nicht korrekt klassifiziert werden. Mit einem sehr hohen Wert für  $k$ , wie  $k = 19$ , zeigt sich ein deutliches Underfitting. Für  $k = 3$  zeigt sich eine akzeptable Klassifizierung der Testdaten. Diese genannten Klassifizierungen sind in Abbildung 3 mit der Anzeige von KNNchanted visualisiert.

Auch eine ungleichmäßige Verteilung der Trainingsdatenpunkte kann die Güte des Klassifikationsergebnisses des KNN-Algorithmus negativ beeinflussen. Der Datensatz „verteilung“ demonstriert dies mit zehn Trainingsdatenpunkten der Kategorie rot und zwei Trainingsdatenpunkten der Kategorie grün. Der grüne Testdatenpunkt wird lediglich für  $k = 1$  richtig klassifiziert, da ab einem  $k > 2$  immer die roten Datenpunkte allein durch ihre Anzahl überwiegen.

Der Datensatz „drei\_punkte“ kann – wie in Unterabschnitt 2.1 beschrieben – dazu verwendet werden, die quadratische Abnahme der Stimmengewichte zu motivieren.

Mithilfe des Datensatzes „zwei\_ringe“, dargestellt in Abbildung 4, soll die Verwendung einer Normalisierung motiviert werden. Der Datensatz bildet im normalisierten Zustand einen Ausschnitt zweier deutlich zu erkennender Ringe, die die einzelnen Kategorien bilden. Aufgrund der unterschiedlichen Wertebereiche der x- und y-Koordinate, welche sich etwa in den Intervallen [100, 1000] und [0, 1] befinden, ist in der direkten Darstellung des Datensatz-

zes in KNNchanted weder eine klare Aufteilung in Kategorien erkennbar noch klassifiziert der KNN-Algorithmus diesen Datensatz erfolgreich. Erst durch die Normalisierung wird eine Klassifizierung optisch sowie unter Verwendung des KNN-Algorithmus deutlich.

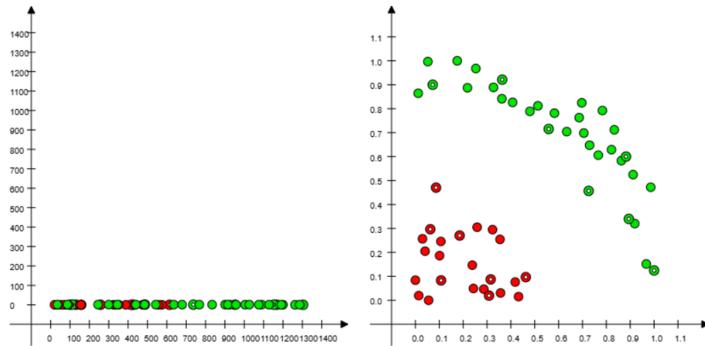


Abbildung 4. Der Datensatz „zwei\_ringe“ vor und nach der Normalisierung.

Als praktische Datensätze mit Realitätsbezug stehen die Datensätze „t-shirt“, „geraete“ und „iris“ zur Verfügung. In Anlehnung an die von Dr. Wolfgang Pfeffer und Tobias Fuchs entwickelte Umsetzung des KNN-Algorithmus mittels Tabellenkalkulation [4] stehen in „t-shirt“ einzelne T-Shirts mit den Merkmalen Körpergröße und Brustumfang zur Verfügung, die in die T-Shirt-Größen S, M und L eingeteilt werden sollen. In „geraete“ sollen Geräte, die auf eine Website zugreifen, anhand ihrer Größe in die Kategorien Mobiltelephon, Tablet und Computer eingeteilt werden. Der Datensatz wurde [1, S. 60] entnommen und auf etwa 80 Trainings- und Testdatenpunkte verkleinert, um von KNNchanted in angemessener Zeit bearbeitbar zu sein. „iris“ ist eine für KNNchanted lesbare Version des Schwertlilien-Datensatzes [3]. Da in KNNchanted nur zwei Dimensionen zur Verfügung stehen, werden lediglich die Kelchblattbreite (sepal width) auf der x-Achse und die Blütenblatlänge (petal length) auf der y-Achse angetragen.

## 6 Erprobung des Programms

### 6.1 CHARAKTERISIERUNG DER LERNGRUPPE

KNNchanted wurde mit einer elften Jahrgangsstufe an meiner Einsatzschule getestet. Die Klasse 11c des Werner-von-Siemens-Gymnasiums unterrichte ich seit dem Beginn des Schuljahres 2024/25 zwei Stunden pro Woche in spätbeginnender Informatik. Die Klasse setzt sich aus 18 Schülerinnen und 5 Schülern zusammen. Das Verhältnis zwischen den

Schülern und mir empfinde ich als sehr angenehm. Die Klasse ist im Allgemeinen äußerst diszipliniert, in ihrer Mitarbeit im Unterrichtsgespräch allerdings häufiger zurückhaltend. Die Lernatmosphäre ist insgesamt als gut zu bezeichnen. Im Verlauf des Schuljahres zeigte sich – insbesondere bei dem Lehrplaninhalt „Codierung und Verschlüsselung“ – eine deutliche Unsicherheit der Schülerinnen und Schüler sobald Inhalte aus der Mathematik in der Informatik auftraten. Daraus folgt, dass die mathematischen Teilespekte des KNN-Algorithmus, wie die verwendete Metrik, mit einem besonderen didaktischen Fokus in dieser Klasse vermittelt werden müssen.

## 6.2 PLANUNG DER UNTERRICHTSSEQUENZ

Die Unterrichtssequenz ist in drei Abschnitte unterteilt. In einer ersten Doppelstunde wird der KNN-Algorithmus als ein Algorithmus des maschinellen Lernens mithilfe des Datensatzes „t-shirt“ in GeoGebra eingeführt. Dazu klassifizieren die Schülerinnen und Schüler in Gruppen drei unbekannte Testdatenpunkte für  $k = 1$ ,  $k = 2$  und  $k = 3$ . Sie stellen fest, dass sie für unterschiedliche  $k$  zu unterschiedlichen zugewiesenen Labeln kommen und abstrahieren ihr jeweiliges Vorgehen zu einem allgemeinen Algorithmus in Abhängigkeit von  $k$ . Danach erfolgt die Sicherung über die händische Klassifizierung eines anderen kleinen Datensatzes in GeoGebra. Aufbauend auf der Idee des Algorithmus wird in der folgenden Doppelstunde der Parameter der verwendeten Metrik genauer untersucht. Der euklidische Abstand sowie die Manhattan-Distanz werden eingeführt und Beispielabstände berechnet. Zur Vertiefung wird KNNchanted als entdeckende Lernumgebung eingesetzt. Die Verwendung von KNNchanted beginnt mit einer erneuten Klassifizierung des Datensatzes „t-shirt“ für  $k = 1$ ,  $k = 2$  und  $k = 3$ . Durch die zuvor händisch durchgeführte Klassifikation der Testdatenpunkte kann die Klassifikation von KNNchanted überprüft werden und – da beide Klassifizierungen übereinstimmen – Vertrauen zum Programm geschaffen werden. In einem anschließenden Arbeitsauftrag sollen die Schülerinnen und Schüler die zur Verfügung gestellten Beispieldatensätze für verschiedene  $k$  klassifizieren. Dabei erkunden sie zum einen Phänomene wie Overfitting, Underfitting und Normalisierung, zum anderen entdecken sie einige Eigenschaften, die ein guter Datensatz des maschinellen Lernens aufweisen sollte, wie eine ausreichende Menge an Trainingsdaten oder eine gleichmäßige Verteilung der Kategorien der Datenpunkte. Insgesamt ist das Ziel dieses Arbeitsauftrags die in 5.7

beschriebenen Effekte der Beispieldatensätze selbstständig zu erkunden. Die Arbeitsblätter dieser Unterrichtssequenz befinden sich im Anhang. Die verwendeten PowerPoint-Folien sowie GeoGebra-Dateien können unter [https://stadtnbg-my.sharepoint.com/:f/g/personal/daniela\\_andres\\_schulen\\_nuernberg\\_de/EjQ6G9UjftdGm\\_x6Qsea00sBec5hJill9HKTUNUT029SUw?e=FoL18](https://stadtnbg-my.sharepoint.com/:f/g/personal/daniela_andres_schulen_nuernberg_de/EjQ6G9UjftdGm_x6Qsea00sBec5hJill9HKTUNUT029SUw?e=FoL18) abgerufen werden.

### 6.3 DURCHFÜHRUNG UND REFLEXION DER UNTERRICHTSSEQUENZ

Die Unterrichtssequenz wurde wie geplant in zwei Doppelstunden, also insgesamt vier Unterrichtsstunden durchgeführt.

Die Grundidee des Algorithmus haben die Schülerinnen und Schüler in der Gruppenarbeit sicher nachvollzogen, leichte Schwierigkeiten traten jedoch bei der Beschreibung des allgemeinen Algorithmus in Form eines Lückentextes auf. Die anschließende Klassifizierung eines Beispieldatensatzes in GeoGebra wurde wieder sicher durchgeführt. Der Umgang mit GeoGebra war durch den Mathematikunterricht bereits bekannt und die Schülerinnen und Schüler gingen souverän mit der Software um. Obwohl bereits in der Vorbereitung der Unterrichtssequenz ein Fokus auf die mathematische Einführung der Metriken gelegt wurde, erwies sich die Verwendung der euklidischen Metrik ohne Einsatz von GeoGebra als herausfordernd. Der Satz des Pythagoras war in seiner Funktion zum Messen von Abständen nicht als gesichertes Vorwissen vorhanden. Die Manhattan-Distanz hingegen wurde mit einer kurzen Erklärung schnell verstanden und konnte direkt an Beispielen angewandt werden. Auch der Ursprung der Benennung dieser Metrik konnte von den Schülerinnen und Schülern selbstständig hergeleitet werden. Am Ende des Arbeitsblattes formulierten die Schülerinnen und Schüler korrekt als Fazit, dass die verwendete Metrik das Ergebnis des KNN-Algorithmus beeinflusst.

Der Einstieg in die Verwendung von KNNchanted gestaltete sich unerwartet schwierig. Sowohl das Laden eines Datensatzes über das Auswahlmenü und die entsprechende Schaltfläche als auch das Ablesen der zugewiesenen Kategorien der Testdatenpunkte stellte für viele Schülerinnen und Schüler eine erhebliche Herausforderung dar. Trotz mehrfachen Ablesens der Kategorisierungen der Testdatenpunkte für die verschiedenen Werte von  $k$  beim „t-shirt“-Datensatz konnte bis zur Phase des Arbeitsauftrags keine durchgängig sichere Bedienung von KNNchanted erreicht werden. Hier wäre in Zukunft

mehr Zeit einzuplanen. Zudem wäre das Hinzufügen eines kurzen Tutorials denkbar, dass die Schülerinnen und Schüler individuell durch die Bedienung der Software führt.

In der anschließenden eigenständigen Arbeitsphase zeigte sich dementsprechend ein deutlicher Unterschied im Arbeitstempo: Während einzelne Schülerinnen und Schüler zügig arbeiteten, war bei anderen intensive Einzelhilfe – besonders beim Laden neuer Datensätze – notwendig. Insgesamt wurde für den Arbeitsauftrag somit mehr Zeit benötigt als ursprünglich eingeplant.

Die Untersuchung von Overfitting- und Underfitting-Effekten erfolgte mithilfe des Datensatzes „ausreisser“, bei dem nacheinander mit den Parametern  $k = 1$ ,  $k = 3$  und  $k = 19$  klassifiziert wurde. Dabei zeigte sich, dass insbesondere die Klassifizierung für  $k = 19$  auf den Schülerrechnern vergleichsweise viel Zeit in Anspruch nahm. Zudem führte die erneute Klassifikation mit einem anderen  $k$  dazu, dass die vorherigen Ergebnisse nicht mehr sichtbar waren. Für die Zukunft wäre hier eine Arbeit in Dreiergruppen empfehlenswert, sodass jedes Gruppenmitglied jeweils eine Klassifikation durchführt und ein direkter Vergleich zwischen den verschiedenen Klassifizierungen ermöglicht wird. Alternativ kann bei besonders versierten Schülerinnen und Schülern die Arbeit in mehreren Registerkarten angeregt werden. Trotz dieser organisatorischen Herausforderungen gelang es der Mehrheit der Schülerinnen und Schüler, eine sinnvolle Abstufung der Klassifizierungsergebnisse vorzunehmen. Dass die Wahl von  $k = 3$  im gegebenen Kontext zu der geeignetsten Klassifikation führt, wurde von nahezu allen Schülerinnen und Schülern korrekt beobachtet.

Das Prinzip der Normalisierung wurde in der grafischen Darstellung von den Schülerinnen und Schülern intuitiv erfasst. Sobald jedoch die zugrunde liegenden Formeln auf dem Arbeitsblatt thematisiert wurden, kam es erneut zu Verständnisproblemen. Hier wäre zukünftig zu überlegen, ob auf eine formelbasierte Darstellung zugunsten einer stärkeren Visualisierung verzichtet werden könnte.

Die Bedeutung einer ausgewogenen Verteilung der Kategorien in den Trainingsdaten wurde anhand des Datensatzes „verteilung“ thematisiert. Dabei erkannten die Schülerinnen und Schüler rasch das Grundprinzip, dass eine starke Unausgewogenheit der Kategorien — insbesondere eine sehr geringe Anzahl an Datenpunkten einer Kategorie — problematisch für die Klassifikation der Testdatenpunkte ist. Die Arbeit mit diesem Datensatz verlief besonders gut und wurde von den meisten Schülerinnen und Schülern erfolgreich bearbeitet.

Aufgrund der durch die verlängerte Einführung bereits weit fortgeschrittenen Unterrichtszeit wurde der Datensatz „drei\_punkte“ nicht mehr im Plenum behandelt. Während der Arbeitsphase erreichten lediglich einzelne leistungsstärkere Schülerinnen und Schüler diesen Teil des Arbeitsauftrags. Mit etwas Unterstützung gelang es ihnen jedoch größtenteils, den dahinterliegenden Gedanken der quadratischen Abnahme des Stimmgewichts nachzuvollziehen.

Die Motivation zur eigenständigen Variation aller bekannter Parameter und damit Klassifizierung der Datensätze „anzahl\_klein“ und „anzahl\_gross“ fiel schlussendlich – vielleicht auch aufgrund des Zeitdrucks am Ende – eher gering aus. Möglicherweise könnte mehr Zeit sowie ein motivierenderes, lebensnäheres Beispiel in einer zukünftigen Umsetzung das Interesse und die Eigeninitiative der Schülerinnen und Schüler an dieser Stelle steigern.

Zum Abschluss der Unterrichtssequenz formulierten die Schülerinnen und Schüler gemeinsam ein Fazit darüber, welche Eigenschaften ein guter Trainingsdatensatz für den KNN-Algorithmus aufweisen sollte. Dabei wurden die zentralen Aspekte – wie eine möglichst gleichmäßige Verteilung der Kategorien, eine ausreichende Anzahl an Trainingsdaten sowie das Vermeiden von Ausreißern – von den Schülerinnen und Schülern treffend benannt. Damit konnte die Unterrichtssequenz inhaltlich erfolgreich abgeschlossen werden.

## 7 Bewertung

### 7.1 VERIFIKATION DER USE CASES

Ziel dieser Arbeit war es, ein Programm zu entwickeln, dass die in Abschnitt 3 beschriebenen Use Cases erfüllt. Im Folgenden soll begründet werden, wie KNNchanted diese Anforderungen umsetzt:

Über die Schaltflächen „Beispieldaten laden“ und „Eigenen Datensatz hochladen“ können verschiedene Datensätze – bestehend aus Trainings- und Testdatenpunkten – in das Programm geladen werden. Die zur Verfügung stehenden Beispieldatensätze variieren in den in Abschnitt 3 aufgeführten Parametern. Insbesondere existiert mit „ausreisser“ die Möglichkeit Overfitting- und Underfitting-Effekte sichtbar zu machen. Somit erfüllt KNNchanted Use Case 1.

Mit dem implementierten Koordinatensystem erfüllt KNNchanted zudem Use Case

2. Die Trainings- und Testdatenpunkte werden als Kreise und Kreisringe dargestellt und sind dadurch optisch gut voneinander unterscheidbar. Insbesondere wurde bei der Implementierung des Koordinatensystems darauf geachtet, die Achsen angemessen zu skalieren, aber gleichzeitig nicht eine Normalisierung vorwegzunehmen.

Unter dem Abschnitt „Parameter“ können in KNNchanted das verwendete  $k$  sowie die verwendete Metrik eingestellt werden. Zudem ist es möglich einzelne Merkmale individuell zu gewichten sowie die Gewichtung der Stimmen der einzelnen Datenpunkte bei der Abstimmung des KNN-Algorithmus anzupassen. Mit diesen zusätzlichen Funktionen erfüllt KNNchanted insbesondere Use Case 3.

Nach Ausführung des KNN-Algorithmus von KNNchanted wird die Klassifizierung durch eine Färbung des Hintergrunds des Koordinatensystems sichtbar gemacht. Durch die gewählte Darstellung der Testdatenpunkte kann die Klassifizierung gut mit bloßem Auge abgelesen werden. Zusätzlich steht eine Konfusionsmatrix zur Verfügung. Mit dieser können anschließend weitere Werte zur Qualitätseinschätzung der Klassifizierung – wie die Fehlerrate oder Sensitivität – berechnet werden. Folglich sind die Anforderungen von Use Case 4 erfüllt.

## 7.2 LIMITIERUNGEN

Trotz der erfolgreichen Umsetzung der in Abschnitt 3 beschriebenen Use Cases unterliegt KNNchanted einigen Beschränkungen. So ist es zwar grundsätzlich möglich, Datensätze mit mehr als 50 Datenpunkten zu klassifizieren, jedoch steigt mit wachsender Anzahl an Datenpunkten der Rechenaufwand deutlich an. Insbesondere auf leistungsschwächeren Schulrechnern kann dies zu spürbaren Verzögerungen bei der Ausführung des KNN-Algorithmus führen. Aus diesem Grund empfiehlt sich im Unterrichtseinsatz die Beschränkung auf kleinere Datensätze mit einer Datenpunktanzahl im unteren zweistelligen Bereich. Zudem können lediglich neun Kategorien farblich unterschiedlich dargestellt werden, ab zehn Kategorien kommt es zu Dopplungen der Farben. Darüber hinaus ist KNNchanted aktuell ausschließlich auf Datensätze ausgelegt, deren Merkmale reelle Zahlen aus dem  $\mathbb{R}^2$  sind. Zwar ließe sich der KNN-Algorithmus mit angepassten Metriken auch auf nicht-numerische Merkmale anwenden, doch würde diese Möglichkeit über den schulischen Anwendungsbereich hinausgehen. Aus dem gleichen Grund wurden Regressionsprobleme

- die mit dem KNN-Algorithmus ebenfalls umgesetzt werden können – in KNNchanted nicht implementiert.

## 8 Implementierung

KNNchanted ist als Webanwendung auf der Basis von Scala.js [6] erstellt worden. Scala.js ist ein Scala Compiler-Plugin, welches Scala-Code in JavaScript transpilieren kann. Somit war es für die Entwicklung von KNNchanted möglich, objektorientierte Prinzipien zu nutzen. Alle Beispieldatensätze wurden mithilfe von GeoGebra und eines externen Java-Projektes zur Anpassung des Formats erstellt. Als Webanwendung genügt zur Benutzung von KNNchanted ein üblicher Browser (wie Mozilla Firefox, Google Chrome oder Microsoft Edge). Die GUI benötigt eine Mindestauflösung von 1024x768 Pixeln. Weitere Bedingungen müssen nicht erfüllt sein. Die Implementierung selbst umfasst aktuell etwa 1300 Zeilen Scala-Code und kann online unter <https://github.com/Daniela-Andres/KNNchanted> eingesehen und heruntergeladen werden. Dort beigelegt befindet sich zudem eine mit Scaladoc generierte Dokumentation des Projekts. KNNchanted selbst ist unter <https://hsgym.de/knn/> verfügbar.

### 8.1 UMSETZUNG DES KNN-ALGORITHMUS

Der KNN-Algorithmus wird in KNNchanted in dem Objekt `DataInfo`, das den aktuellen Datensatz speichert, umgesetzt. Daran beteiligt sind die Methoden `knn`, welche den zu klassifizierenden Punkt, das gewählte  $k$  und die verwendete Metrik erhält und das vom KNN-Algorithmus zugewiesene Label als ein `Option[String]` zurück gibt, sowie dessen Unterroutine `subsets` und `abstimmung`. `Option[T]` ist Scalars Version des `Optional[T]` aus Java, also ein Behälter für ein Objekt, das eventuell einen nicht `null`-Wert enthält. Dies ist nötig, da die Klassifizierung des KNN-Algorithmus nicht immer eindeutig ist. Somit kann im uneindeutigen Fall ein leeres Option-Objekt zurückgegeben werden. Die Unterroutine `subsets` berechnet alle möglichen Teilmengen für den Fall, dass es mehrere Datenpunkte gibt, die die gleiche Entfernung zum zu klassifizierenden Punkt haben wie der  $k$ -nächste Nachbar. Die Unterroutine `abstimmung` führt die Abstimmung der Datenpunkte mit den in der Zone „Parameter“ eingestellten Eigenschaften durch.

Die angesprochene verwendete Metrik wird über sogenannte Traits [11] umgesetzt. Traits sind vergleichbar mit Interfaces aus Java 8. Dies hat unter anderem den Vorteil, dass die von KNNchanted zur Verfügung gestellten Metriken leicht erweitert werden können.

## 8.2 REALISIERUNG DER BENUTZEROBERFLÄCHE

Die Implementierung der Oberfläche der Anwendung ist in fünf Klassen unterteilt.

Die Klasse `CoordinateSystem` dient zur Verwaltung der obigen Koordinatensystemanzeige. Sie verfügt über Methoden zur Zeichnung des Koordinatensystems, der Datenpunkte sowie zur Färbung des Hintergrunds nach Ausführen des KNN-Algorithmus.

Zum Erstellen der einzelnen Elemente des Schaltflächenbereichs existieren die vier Klassen `DatenAbschnitt`, `NormalisierungAbschnitt`, `AlgoAbschnitt` und `AusfuehrenAbschnitt`. Die Implementierung der Funktionalität der einzelnen Schaltflächen befindet sich ebenfalls in diesen Klassen.

Die Hauptaufgabe der Klasse `DatenAbschnitt` ist es, das Einlesen der Datensätze zu verwalten. Die Klasse `ServerResourceReader` ist dabei eine Hilfsklasse zum Laden der Datensätze vom Server. Das Einlesen eines lokalen Datensatzes ist in der Klasse `Parser` ausgelagert. Diese stellt eine Methode `parse` zur Verfügung, welche eine Textdatei in dem in Unterabschnitt 4.2 beschriebenen Format übergeben bekommt und einen von KNNchanted verwaltbaren Datensatz zurückgibt.

Die Klasse `NormalisierungAbschnitt` implementiert die beschriebene Min-Max-Normalisierung. Mittels Auswahlschaltflächen kann die Normalisierung der einzelnen Merkmale optional aus- und abgewählt werden. Das Objekt `Reverse` existiert, um diese Normalisierung ohne viel Rechenaufwand rückgängig machen zu können.

`AlgoAbschnitt` stellt die Textfelder zur Eingabe von  $k$ , den Gewichten und der Abnahmgeschwindigkeit zur Verfügung. Die in Unterabschnitt 5.5 erwähnten Eingabehandler sind direkt an die entsprechenden Textfelder als EventListener gebunden.

Die Klasse `AusfuehrenAbschnitt` enthält eine einzige Schaltfläche, welche die Ausführung des KNN-Algorithmus veranlasst sowie die Konfusionsmatrix befüllt.

## 9 Ausblick

Mit KNNchanted liegt ein Programm vor, das bei einem Algorithmus des maschinellen Lernens die Lehr- und Lernprozesse unterstützt, welche vom Lehrplan der neuen gymnasialen Oberstufe skizziert werden. Anders als bereits bestehende Ansätze zum Unterrichten des KNN-Algorithmus, wie etwa die Umsetzung in Tabellenkalkulation, setzt KNNchanted bewusst auf eine interaktive und entdeckende Lernumgebung, die eine vertiefte Auseinandersetzung mit den Parametern des KNN-Algorithmus sowie das Betrachten verschiedenen Datensätzen ermöglicht.

KNNchanted ist dabei vorrangig ein didaktisches Werkzeug: Die Entscheidung für eine webbasierte Lösung erleichtert den Einsatz im Schulalltag. Auch in der Gestaltung der Oberfläche, der Wahl der unterstützten Metriken oder der verwendeten Normalisierung wurden bewusst didaktische Vereinfachungen gewählt, die sich am Vorwissen von Schülerinnen und Schülern der elften Jahrgangsstufe orientieren. Ebenso wurden die mitgelieferten Beispieldatensätze gezielt so konzipiert, dass typische Herausforderungen maschinellen Lernens wie Over- und Underfitting sowie ungleiche Verteilung der Trainingsdaten unmittelbar erfahrbar werden.

Die unterrichtliche Erprobung von KNNchanted zeigte sowohl Potenziale als auch Grenzen der Software auf. Während viele Schülerinnen und Schüler ein gutes Verständnis für die Auswirkungen einzelner Parameter entwickeln konnten, erwies sich der Einstieg in die Bedienung der Anwendung als unerwartet anspruchsvoll. Hier besteht Optimierungsbedarf.

Insgesamt hat sich gezeigt, dass KNNchanted einen sinnvollen Beitrag leisten kann, um algorithmische Konzepte nicht nur theoretisch zu vermitteln, sondern auch durch eigenes Ausprobieren und Beobachten erlebbar zu machen. Eine breitere Erprobung in unterschiedlichen Klassen sowie mögliche Erweiterungen — etwa ein einfacher Editor für eigene Datensätze oder zusätzliche Hilfefunktionen — könnten die Zugänglichkeit und den Nutzen der Anwendung weiter erhöhen. Somit steht mit KNNchanted ein weiteres Werkzeug im schulischen Umgang mit maschinellem Lernen zur Verfügung.

## Literatur

- [1] Daniela Andres, Silvia Joachim und Martin Hennecke. „ClusterLabor: Ein Werkzeug zur interaktiven Visualisierung und Analyse von Clusteralgorithmen.“ In: *Informatische Bildung in Schulen* 2(2) (2024). URL: <https://doi.org/10.18420/ibis-02-02-09> (besucht am 02.08.2025).
- [2] Wolfgang Ertel. *Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung*. 5. Aufl. Wiesbaden: Springer, 2021.
- [3] R. A. Fisher. *Iris*. UCI Machine Learning Repository. 1936. URL: <https://doi.org/10.24432/C56C76> (besucht am 02.08.2025).
- [4] Wolfgang Pfeffer und Tobias Fuchs. *Der k-nächste-Nachbarn-Algorithmus mit Tabellenkalkulation*. URL: [https://kex.fim.uni-passau.de/k\\_naechste\\_Nachbarn/html/knn.html](https://kex.fim.uni-passau.de/k_naechste_Nachbarn/html/knn.html) (besucht am 02.08.2025).
- [5] Stuart J. Russel und Peter Norvig. *Artificial Intelligence: A Modern Approach*. 4. Aufl. Pearson, 2021.
- [6] *Scala.js - A safer way to build robust front-end web applications!* URL: <https://www.scala-js.org/> (besucht am 12.03.2025).
- [7] Staatsinstitut für Schulqualität und Bildungsforschung (ISB). *Der Lernbereich „Künstliche Intelligenz“ in der Jahrgangsstufe 11 des Gymnasiums (Informatik und spät beginnende Informatik): Erläuterungen und Materialien für Lehrkräfte*. URL: <https://www.isb.bayern.de/schularten/gymnasium/faecher/informatik/handreichung-kuenstliche-intelligenz/> (besucht am 09.07.2025).
- [8] Staatsinstitut für Schulqualität und Bildungsforschung (ISB). *LehrplanPLUS für das Gymnasium in Bayern: Informatik 11 (NTG)*. München. URL: <https://www.lehrplanplus.bayern.de/fachlehrplan/gymnasium/11/informatik/ntg> (besucht am 02.08.2025).
- [9] Staatsinstitut für Schulqualität und Bildungsforschung (ISB). *LehrplanPLUS für das Gymnasium in Bayern: Mathematik 12 (erhöhtes Anforderungsniveau)*. URL: <https://www.lehrplanplus.bayern.de/fachlehrplan/gymnasium/12/mathematik/regulaer> (besucht am 07.06.2025).

- [10] Staatsinstitut für Schulqualität und Bildungsforschung (ISB). *LehrplanPLUS für das Gymnasium in Bayern: spät beginnende Informatik 11 (HG, SG, MuG, SWG)*. München. URL: [https://www.lehrplanplus.bayern.de/fachlehrplan/gymnasium/11/informatik/mug\\_swg\\_sg](https://www.lehrplanplus.bayern.de/fachlehrplan/gymnasium/11/informatik/mug_swg_sg) (besucht am 02.08.2025).
- [11] *Tour of Scala: Traits*. URL: <https://docs.scala-lang.org/tour/traits.html> (besucht am 07.06.2025).

## **10 Eigenständigkeitserklärung**

Hiermit erkläre ich, dass die vorliegende Hausarbeit von mir selbstständig verfasst wurde und dass keine anderen als die angegebenen Hilfsmittel benutzt wurden. Die Stellen der Arbeit, die anderen Werken dem Wortlaut oder Sinn nach entnommen sind, sind in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht.

Diese Erklärung erstreckt sich auch auf etwa in der Arbeit enthaltene Zeichnungen, Kartenskizzen und bildliche Darstellungen.

---

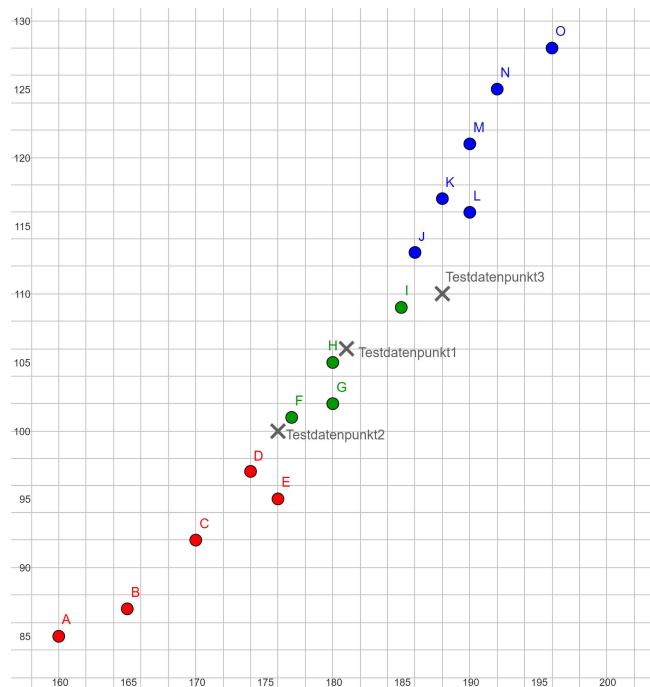
Ort, Datum

---

Unterschrift

## **11 Anhänge**

### 3.1 Klassifikation bei MY-SHIRT



Gruppe 1	Testdatenpunkt 1	Testdatenpunkt 2	Testdatenpunkt 3
nächster Datenpunkt			
zugewiesenes Label			

Gruppe 2	Testdatenpunkt 1	Testdatenpunkt 2	Testdatenpunkt 3
nächster Datenpunkt			
zweitnächster Datenpunkt			
zugewiesenes Label			

Gruppe 3	Testdatenpunkt 1	Testdatenpunkt 2	Testdatenpunkt 3
nächster Datenpunkt			
zweitnächster Datenpunkt			
dritt'nächster Datenpunkt			
zugewiesenes Label			

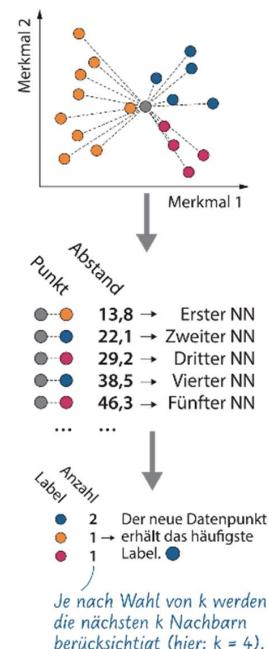
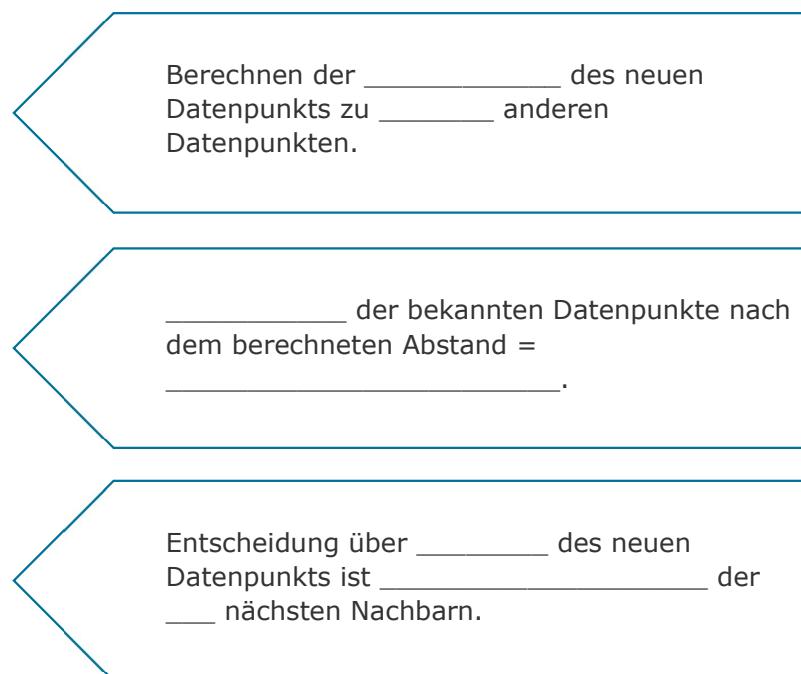
### 3.2 Der k-Nächste-Nachbarn-Algorithmus

MERKE

Der **k-Nächste-Nachbarn-Algorithmus** ist ein überwachtes Lernverfahren, das aus numerischen Daten mit bekannten Labeln lernt, neuen Daten jeweils eines der bekannten Labels zuzuordnen (**Klassifikation**). Dazu werden die Label einer bestimmten Anzahl von nächsten Nachbarn – Parameter  $k$  – betrachtet und eine Mehrheitsentscheidung gefällt.

Der KNN-Algorithmus ist ein **speicherbasiertes** Verfahren und gehört damit zur Familie der "Lazy Learning"-Modelle.

#### Ablauf des Algorithmus



Wir unterscheiden zwischen **Trainings-** und **Testdaten**.

- \_\_\_\_\_: Datenpunkt und \_\_\_\_\_ sind dem KI-System bekannt.  
Trainingsdaten dienen zum Trainieren des KI-Systems
- \_\_\_\_\_: Der Datenpunkt ist dem KI-System bekannt, das entsprechende \_\_\_\_\_ ist zwar festgelegt, aber dem KI-System **nicht** bekannt.  
Testdaten dienen zum Testen und zur Qualitätseinschätzung des KI-Systems.

#### Arbeitsauftrag

Pilze wachsen im Wald oft in Gruppen zusammen. Zwei Cluster von Pilzen sind bereits bekannt, eine Gruppe giftiger Pilze (Label rot) und eine Gruppe essbarer Pilze (Label grün).

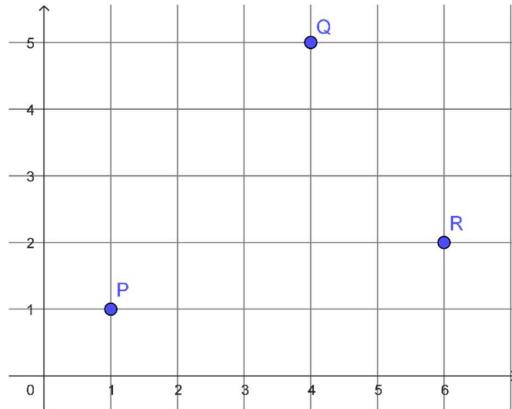
- Öffne die GeoGebra-Datei unter dem Link [kurzelinks.de/c419](http://kurzelinks.de/c419) und klassifiziere die unbekannten Pilze nach dem KNN-Algorithmus für  $k = 3$ .
- Diskutiere mit deinem Banknachbarn, ob du die von dem KNN-Algorithmus als essbar klassifizierten Pilze essen würdest.

### 3.3 Der k-Nächste-Nachbarn-Algorithmus – Parameter

Entscheidend für die Vorhersagequalität ist die richtige Wahl von  $k$ , die in der Praxis stets vom zu lösenden Problem und den gegebenen Daten abhängt. Weitere Einflussfaktoren sind das verwendete Abstandsmaß (= **Metrik**).

#### Metriken und Abstände

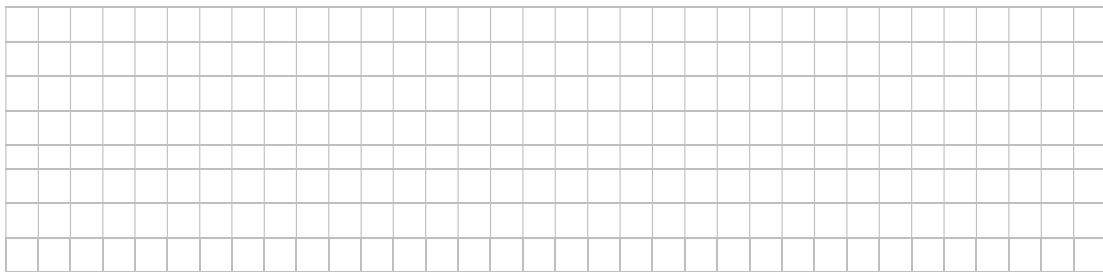
Wie nah sich zwei Punkte sind, kann auf unterschiedliche Weise bestimmt werden.



- a) Bestimme den Abstand von P zu Q und R, indem du ein Lineal verwendest. Welcher Punkt ist der nächste Nachbar zu P?

$$|PQ| = \underline{\hspace{2cm}}, |PR| = \underline{\hspace{2cm}}$$

- b) Berechne den Abstand von P zu Q und R mithilfe des Satz des Pythagoras (= euklidische Metrik). Welcher Punkt ist der nächste Nachbar zu P?



$$|PQ| = \underline{\hspace{2cm}}, |PR| = \underline{\hspace{2cm}}$$

- c) Eine weitere Möglichkeit, den Abstand zwischen zwei Punkten zu messen, ist es, die Kästchenkanten zu zählen, die zwischen den beiden Punkten liegen (= Manhattan-Distanz). Bestimme wieder den Abstand von P zu Q und R. Welcher Punkt ist jetzt der nächste Nachbar zu P?

$$|PQ| = \underline{\hspace{2cm}}, |PR| = \underline{\hspace{2cm}}$$

### 3.4 Der KNN-Algorithmus – Vertiefung

**Klassifizierte** die in der Tabelle angegebenen Datensätze mit verschiedenen Werten von k. **Aktiviere** dabei die **Testdatenpunkte** und bearbeite die gegebene Aufgabenstellung.

Datensatz	k	Aufgabenstellung	Bearbeitung
ausreisser	k = 1, k = 3, k = 19	Beschreibe, welche Problematik(en) bei den einzelnen Werten von k auftreten.	
zwei_ringe	k = 3	Stelle eine Vermutung über die Wertebereiche der x- und y-Koordinaten der Datenpunkte auf.  Aktiviere die Checkboxen im Bereich Normalisierung und klassifizierte den Datensatz erneut. Recherchiere, was eine Normalisierung ist und warum sie bei diesem Datensatz nützlich ist.	
verteilung	k = 1, k = 3	Beschreibe die Auswirkung der Verteilung der Datenpunkte für alle k > 2.	
drei_punkte	k = 1, k = 2	Erkläre, warum der KNN-Algorithmus den Testdatenpunkt für k = 2 anders klassifiziert, als für k = 1.  Aktiviere die Checkbox "Quadratische Gewichtung des Abstands" unter dem Reiter Parameter. Stelle eine Vermutung auf, was diese Änderung bewirkt und überprüfe deine Vermutung indem du den Datensatz erneut mit k = 2 klassifizierst.	
anzahl_klein und anzahl_gross	k = 1, k = 3	Teste beide angegebenen k für beide Datensätze.  Vergleiche die Klassifikationsergebnisse für beide Datensätze und bewerte, mit welchem Datensatz eine bessere Klassifikation erreicht werden kann.  Verstelle dazu auch alle zur Verfügung stehenden Hyperparameter, um eine möglichst gute Klassifizierung zu erreichen!	

**Underfitting und Overfitting**

Werden für das Training eines Systems nicht ausreichend viele repräsentative Datensätze, nur wenig relevante Merkmale oder ungünstige **Hyperparameterwerte** verwendet, so kann das System unter Umständen nicht ausreichend genau trainiert werden. Dies wird auch als **Underfitting** bezeichnet.

Beispiel: \_\_\_\_\_ .

Umgekehrt kann es problematisch sein, wenn das Modell anhand der Trainingsdaten sehr präzise lernt, wie genau diese Trainingsdaten richtig zu klassifizieren sind. Eine solche Überanpassung an die Trainingsdaten wird **Overfitting** genannt.

Beispiel: \_\_\_\_\_ .

**Normalisierung**

In der Praxis werden die einzelnen Eingabe-Variablen für einen Lernalgorithmus oft sehr unterschiedliche Wertebereiche haben.

Zum Beispiel hat der Leukozytenwert in den LexMed-Daten einen Wertebereich zwischen 0 und 50.000 während z.B. die Fieberwerte nur zwischen 35 und 42 Grad variieren.

Dieses Problem kann durch eine Normalisierung der Daten umgangen werden. Dabei werden die Werte der \_\_\_\_\_ durch untenstehende Formel auf das Intervall \_\_\_\_\_ abgebildet.

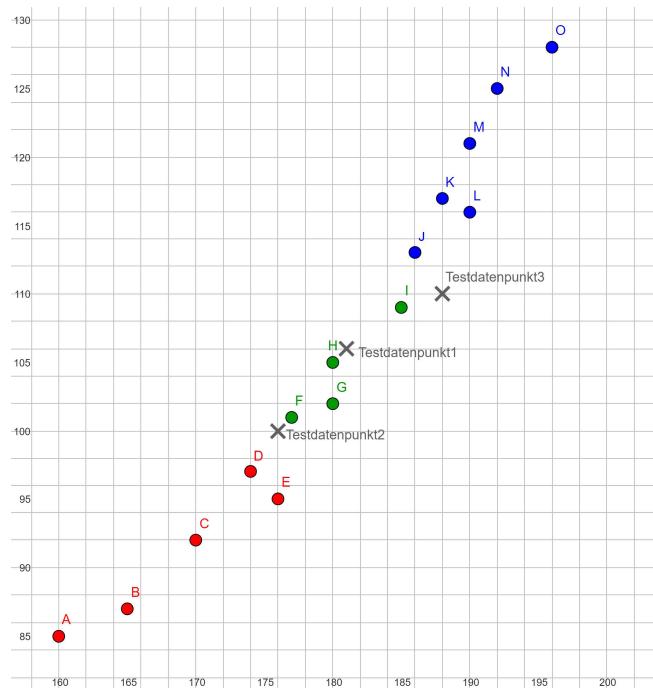
$$x_{i\_normalisiert} = \frac{x_i - min_x}{max_x - min_x} \text{ bzw. } y_{i\_normalisiert} = \frac{y_i - min_y}{max_y - min_y}$$

**Gewünschte Eigenschaften von Trainingsdatensätzen beim maschinellen Lernen**

Ein guter Testdatensatz ist/hat...

- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_
- \_\_\_\_\_

## 3.1 Klassifikation bei MY-SHIRT



Gruppe 1	Testdatenpunkt 1	Testdatenpunkt 2	Testdatenpunkt 3
nächster Datenpunkt	H	F	I
zugewiesenes Label	grün/M	grün/M	grün/M

Gruppe 2	Testdatenpunkt 1	Testdatenpunkt 2	Testdatenpunkt 3
nächster Datenpunkt	H	F	I
zweitnächster Datenpunkt	G	D	J
zugewiesenes Label	grün/M	?	?

Gruppe 3	Testdatenpunkt 1	Testdatenpunkt 2	Testdatenpunkt 3
nächster Datenpunkt	H	F	I
zweitnächster Datenpunkt	G	D	J
dritt næchster Datenpunkt	I	G	L
zugewiesenes Label	grün/M	grün/M	blau/L

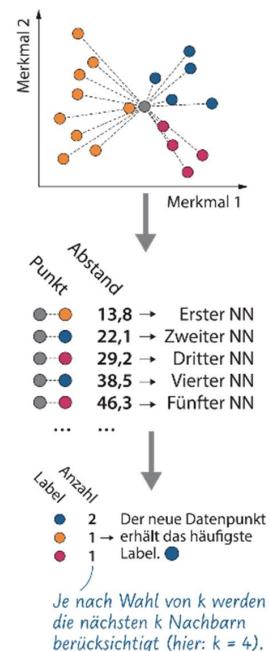
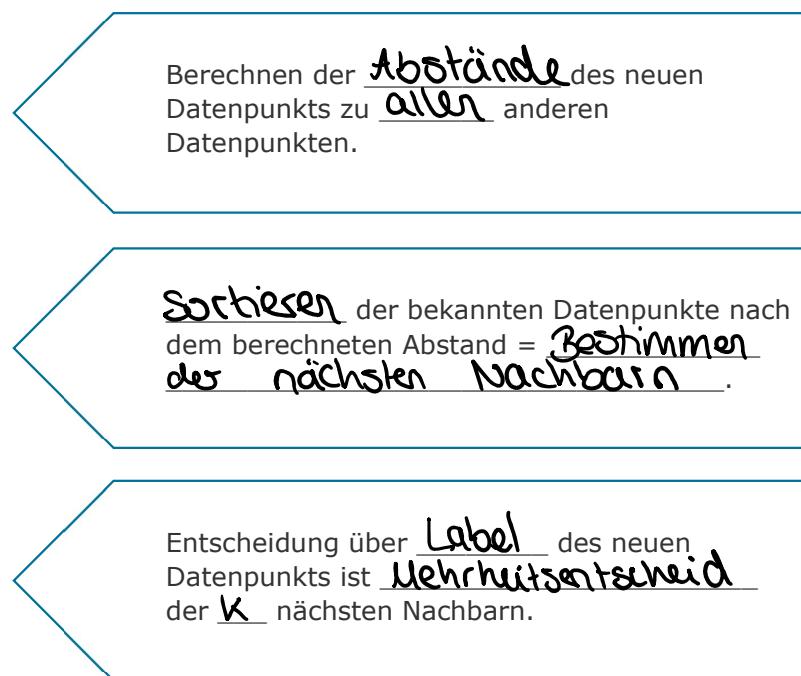
### 3.2 Der k-Nächste-Nachbarn-Algorithmus

MERKE

Der **k-Nächste-Nachbarn-Algorithmus** ist ein überwachtes Lernverfahren, das aus numerischen Daten mit bekannten Labeln lernt, neuen Daten jeweils eines der bekannten Labels zuzuordnen (**Klassifikation**). Dazu werden die Label einer bestimmten Anzahl von nächsten Nachbarn – Parameter  $k$  – betrachtet und eine Mehrheitsentscheidung gefällt.

Der KNN-Algorithmus ist ein **speicherbasiertes** Verfahren und gehört damit zur Familie der "Lazy Learning"-Modelle.

#### Ablauf des Algorithmus



Wir unterscheiden zwischen **Trainings-** und **Testdaten**.

- **Trainingsdaten**: Datenpunkt und Label sind dem KI-System bekannt.  
Trainingsdaten dienen zum Trainieren des KI-Systems
- **Testdaten**: Der Datenpunkt ist dem KI-System bekannt, das entsprechende Label ist zwar festgelegt, aber dem KI-System **nicht** bekannt.  
Testdaten dienen zum Testen und zur Qualitätseinschätzung des KI-Systems.

#### Arbeitsauftrag

Pilze wachsen im Wald oft in Gruppen zusammen. Zwei Cluster von Pilzen sind bereits bekannt, eine Gruppe giftiger Pilze (Label rot) und eine Gruppe essbarer Pilze (Label grün).

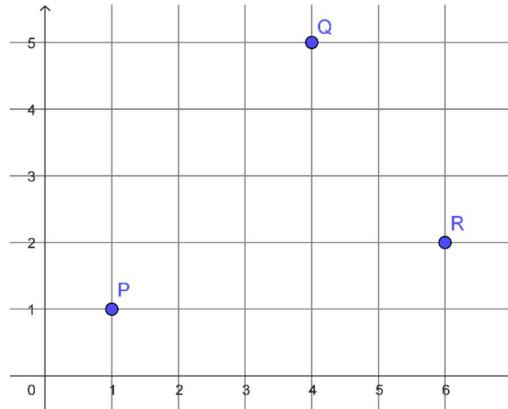
- Öffne die GeoGebra-Datei unter dem Link [kurzelinks.de/c419](http://kurzelinks.de/c419) und klassifiziere die unbekannten Pilze nach dem KNN-Algorithmus für  $k = 3$ .
- Diskutiere mit deinem Banknachbarn, ob du die von dem KNN-Algorithmus als essbar klassifizierten Pilze essen würdest.

### 3.3 Der k-Nächste-Nachbarn-Algorithmus – Parameter

Entscheidend für die Vorhersagequalität ist die richtige Wahl von  $k$ , die in der Praxis stets vom zu lösenden Problem und den gegebenen Daten abhängt. Weitere Einflussfaktoren sind das verwendete Abstandsmaß (= **Metrik**).

#### Metriken und Abstände

Wie nah sich zwei Punkte sind, kann auf unterschiedliche Weise bestimmt werden.



- a) Bestimme den Abstand von P zu Q und R, indem du ein Lineal verwendest. Welcher Punkt ist der nächste Nachbar zu P?

$$|PQ| \approx 5,1, |PR| \approx 5,1 \Rightarrow \text{nächster Nachbar unklar}$$

- b) Berechne den Abstand von P zu Q und R mithilfe des Satz des Pythagoras (= euklidische Metrik). Welcher Punkt ist der nächste Nachbar zu P?

$$|PQ| = \sqrt{3^2 + 4^2} = \sqrt{9+16} = \sqrt{25} = 5$$

$$|QR| = \sqrt{5^2 + 1^2} = \sqrt{26} \approx 5,1$$

$$|PQ| = 5, |PR| = 5,1 \Rightarrow Q \text{ ist nächster Nachbar}$$

- c) Eine weitere Möglichkeit, den Abstand zwischen zwei Punkten zu messen, ist es, die Kästchenkanten zu zählen, die zwischen den beiden Punkten liegen (= Manhattan-Distanz).

Bestimme wieder den Abstand von P zu Q und R. Welcher Punkt ist jetzt der nächste Nachbar zu P?

$$|PQ| = 7, |PR| = 6 \Rightarrow R \text{ ist nächster Nachbar}$$

$\Rightarrow$  Die Art, wie der Abstand zwischen zwei Punkten bestimmt wird (= Abstandsmaß/Metrik), beeinflusst das Ergebnis des KNN-Algorithmus!

### 3.4 Der KNN-Algorithmus – Vertiefung

**Klassifizierte** die in der Tabelle angegebenen Datensätze mit verschiedenen Werten von k. **Aktiviere** dabei die **Testdatenpunkte** und bearbeite die gegebene Aufgabenstellung.

Datensatz	k	Aufgabenstellung	Bearbeitung
ausreisser	k = 1, k = 3, k = 19	Beschreibe, welche Problematik(en) bei den einzelnen Werten von k auftreten.	$k=1 \rightarrow \text{Overfitting}$ $k=3 \rightarrow \text{appropriate fitting}$ $k=19 \rightarrow \text{Underfitting}$
zwei_ringe	k = 3	Stelle eine Vermutung über die Wertebereiche der x- und y-Koordinaten der Datenpunkte auf.  Aktiviere die Checkboxen im Bereich Normalisierung und klassifizierte den Datensatz erneut. Recherchiere, was eine Normalisierung ist und warum sie bei diesem Datensatz nützlich ist.	Unterschiedliche Wertebereiche $\hookrightarrow$ Normalisierung ermöglicht bessere Klassifizierung
verteilung	k = 1, k = 3	Beschreibe die Auswirkung der Verteilung der Datenpunkte für alle $k > 2$ .	Ab $k > 2$ mehr rote als grüne Datenpunkte! $\rightarrow$ immer Label rot
drei_punkte	k = 1, k = 2	Erkläre, warum der KNN-Algorithmus den Testdatenpunkt für $k = 2$ anders klassifiziert, als für $k = 1$ .  Aktiviere die Checkbox "Quadratische Gewichtung des Abstands" unter dem Reiter Parameter. Stelle eine Vermutung auf, was diese Änderung bewirkt und überprüfe deine Vermutung indem du den Datensatz erneut mit $k = 2$ klassifizierst.	Bei $k=2$ Unterschieden im Voting. Änderung: Näherer Punkt hat größeres Stimmengewicht.
anzahl_klein und anzahl_gross	k = 1, k = 3	Teste beide angegebenen k für beide Datensätze.  Vergleiche die Klassifikationsergebnisse für beide Datensätze und bewerte, mit welchem Datensatz eine bessere Klassifikation erreicht werden kann.  Verstelle dazu auch alle zur Verfügung stehenden Hyperparameter, um eine möglichst gute Klassifizierung zu erreichen!	anzahl - gross besser, da ausreichend Trainingsdaten vorhanden sind.

### Underfitting und Overfitting

Werden für das Training eines Systems nicht ausreichend viele repräsentative Datensätze, nur wenig relevante Merkmale oder ungünstige **Hyperparameterwerte** verwendet, so kann das System unter Umständen nicht ausreichend genau trainiert werden. Dies wird auch als **Underfitting** bezeichnet.

Beispiel: Zu hohes k gewählt, Zu wenig Trainingsdaten.

Umgekehrt kann es problematisch sein, wenn das Modell anhand der Trainingsdaten sehr präzise lernt, wie genau diese Trainingsdaten richtig zu klassifizieren sind. Eine solche Überanpassung an die Trainingsdaten wird **Overfitting** genannt.

Beispiel: Ausreißer Datenpunkt und k = 1.

### Normalisierung

In der Praxis werden die einzelnen Eingabe-Variablen für einen Lernalgorithmus oft sehr unterschiedliche Wertebereiche haben.

Zum Beispiel hat der Leukozytenwert in den LexMed-Daten einen Wertebereich zwischen 0 und 50.000 während z.B. die Fieberwerte nur zwischen 35 und 42 Grad variieren.

Dieses Problem kann durch eine Normalisierung der Daten umgangen werden. Dabei werden die Werte der x- und y-Koordinaten durch untenstehende Formel auf das Intervall [0,1] abgebildet.

$$x_{i\_normalisiert} = \frac{x_i - min_x}{max_x - min_x} \text{ bzw. } y_{i\_normalisiert} = \frac{y_i - min_y}{max_y - min_y}$$

### Gewünschte Eigenschaften von Trainingsdatensätzen beim maschinellen Lernen

Ein guter Testdatensatz ist/hat...

- Eine ausreichende Menge an Trainingsdaten
- für alle Kategorien etwa gleiche Anzahl an Trainingsdaten
- Möglichst wenige Ausreißer in den Trainingsdaten
- ...