Bayerisches Staatsministerium für
Unterricht und Kultus

11

Der Lernbereich „Künstliche Intelligenz“ in der Jahrgangsstufe 11 des Gymnasiums

(Informatik und spät beginnende Informatik)

Erläuterungen und Materialien für Lehrkräfte

April 2023

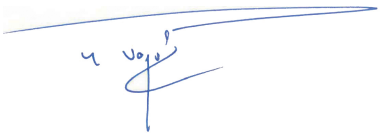
Grußwort des Bayerischen Staatsministers für Unterricht und Kultus

Ob ChatGPT oder Deep-Fake-Fotos – das Thema KI ist heute allgegenwärtig. Für mich steht jetzt schon fest: Mit den neuen KI-Systemen treten wir in eine neue Epoche des digitalen Wandels ein.

Allein die rasante Entwicklung im Bereich der Textgeneratoren wie ChatGPT zeigt: Der kompetente Umgang mit KI wird zu einer entscheidenden Schlüsselkompetenz für unsere Schülerinnen und Schüler. Deshalb haben wir die „Künstliche Intelligenz“ im Fach Informatik ab der Jahrgangsstufe 11 des neuen neunjährigen Gymnasiums fest im LehrplanPLUS verankert. Mit der vorliegenden Broschüre geben wir unseren Informatiklehrkräften das Rüstzeug für den Unterricht im Lernbereich KI an die Hand, um jungen Menschen grundlegende Funktionsweisen von KI-Systemen beizubringen.

Ich danke den Mitgliedern des Arbeitskreises am Staatsinstitut für Schulqualität und Bildungsforschung für die Erarbeitung dieser praxisorientierten Handreichung. Allen Informatiklehrkräften wünsche ich anregende Impulse bei der Lektüre und viel Freude beim Umsetzen der neuen Erkenntnisse im Unterricht!

München, im April 2023



Prof. Dr. Michael Piazolo
Bayerischer Staatsminister
für Unterricht und Kultus

Vorwort

Liebe Kolleginnen und Kollegen,

den Lernbereich „Künstliche Intelligenz“ in den Lehrplan-PLUS aufzunehmen, war bei der grundlegenden Konzeption des Lehrplan eine gewagte Entscheidung, denn Sie als Informatiklehrkräfte wurden in Ihrem Studium nicht auf dieses Thema vorbereitet. Die rasant fortschreitende Entwicklung von KI-Anwendungen und die große Präsenz des Themas in den Medien zeigen aber, dass die damalige Entscheidung richtig war: Unsere Schülerinnen und Schüler müssen sich mit zukunftsweisenden Technologien wie Künstliche Intelligenz auseinandersetzen, um „sachgerecht, selbstbestimmt und verantwortungsvoll“ handeln zu können, wie es in den übergreifenden Bildung- und Erziehungszielen des LehrplanPLUS heißt.

Die Aufnahme des Themas in den Lehrplan zieht allerdings auch einen großen Bedarf an Fort- und Weiterbildungsangeboten nach sich. Deswegen wurde die KI-Fortbildungsinitiative ins Leben gerufen, in deren Rahmen u. a. ein Arbeitskreis „KI“ gebildet wurde, der in sehr enger Kooperation zwischen ALP und ISB innerhalb kürzester Zeit sowohl einen Selbstlernkurs als auch diese Handreichung für den neuen Lernbereich KI in Jahrgangsstufe 11 entwickelte.

Natürlich soll die Handreichung kein Lehrbuch ersetzen, sie vermittelt Ihnen aber fachliche Grundlagen und gibt didaktische Hinweise, die Sie bei der unterrichtlichen Umsetzung unterstützen sollen. Darüber hinaus bietet sie eine große Fülle an hilfreichen, größtenteils vom Arbeitskreis selbst entwickelten Materialien und Tools, die in der Handreichung beschrieben und Ihnen über einen begleitenden Mebiskurs zur Verfügung gestellt werden.

Ich möchte mich an dieser Stelle zum einen beim Bayerischen Staatsministerium für Unterricht und Kultus bedanken, dass es die Fortbildungsinitiative ermöglich hat, zum anderen bei den Mitgliedern des Arbeitskreises KI für deren überaus engagierte und gelungene Arbeit.

Lassen Sie sich nun begeistern von dem spannenden Thema KI und geben Sie Ihre Begeisterung an Ihre Schülerinnen und Schüler weiter. Ich wünsche Ihnen viel Freude dabei.

München, im April 2023

Anselm Räde, *Direktor des ISB*

Inhaltsverzeichnis

KAPITEL 1 Einführung

„ Wir neigen dazu, die Auswirkungen einer neuen Technologie auf kurze Sicht zu überschätzen und auf lange Sicht zu unterschätzen.“
Amaras Gesetz

Künstliche Intelligenz ist eine Technologie, die in vielen Bereichen Anwendung findet und die auf ganz unterschiedliche Weise Einfluss auf unser Leben nimmt. Obwohl diese Technologie nicht neu ist, steckt sie eigentlich noch in den Kinderschuhen. KI-Systeme werden immer leistungsfähiger, daher ist damit zu rechnen, dass ihr Einsatz in Bereichen, in denen KI bereits heute genutzt wird, in Zukunft stark zunehmen wird. Außerdem werden sich neue Anwendungsgebiete eröffnen, in denen KI bislang noch keinen Einzug gehalten hat.

Unsere Schülerinnen und Schüler verwenden KI-Systeme, oft ohne es zu wissen oder zu merken. Künstliche Intelligenz wird ihnen nach ihrer Schulzeit sogar noch häufiger begegnen – sowohl im Beruf als auch im Alltag. Dafür müssen sie gewappnet sein. Die Schülerinnen und Schüler mit den notwendigen Kompetenzen auszustatten, um in Zukunft selbstbestimmt und verantwortungsbewusst mit Technologien wie Künstliche Intelligenz umgehen zu können, ist Teil des Bildungsauftrags der Schulen.

Dem Fach Informatik kommt dabei eine Schlüsselrolle zu, denn die Verwirklichung dieses Bildungsauftrags erfordert, dass die Schülerinnen und Schüler nicht nur wissen, wie Künstliche Intelligenz genutzt werden kann und welche Auswirkungen sie auf Individuum und Gesellschaft haben kann, sondern auch, dass sie die Technologie verstehen und erklären können. Denn eine nachhaltige digitale Bildung ist gemäß der Dagstuhl-Erklärung der Gesellschaft für Informatik (s. Brinda et al. (2016)) nur möglich, wenn Themen wie Künstliche Intelligenz aus unterschiedlichen Perspektiven betrachtet und hinterfragt werden.

Um dieser Forderung gerecht zu werden und der Künstlichen Intelligenz den Raum zu geben, den ein derart komplexes Thema erfordert, wurde im LehrplanPLUS in Jahrgangsstufe 11 des Gymnasiums ein eigener Lernbereich „Künstliche Intelligenz“ in den Fächern Informatik

Abb. 1.1: Das Dagstuhl-Dreieck der Dagstuhl-Erklärung (Brinda et al., 2016, S. 3).

(NTG), spät beginnende Informatik (HG, SG, MuG, SWG) und Wirtschaftsinformatik (WWG) geschaffen, sodass die Schülerinnen und Schüler aller Ausbildungsrichtungen grundlegende Kompetenzen in diesem Bereich erwerben.

Der Lernbereich umfasst in den Fächern Informatik und spät beginnende Informatik sechs Kompetenzerwartungen. In Informatik sind dafür ca. 16 Stunden vorgesehen, in spät beginnender Informatik ca. 12 Stunden. Worin die Unterschiede in den Kompetenzerwartungen zwischen den beiden Fächern im Einzelnen bestehen, wird in der Handreichung an der entsprechenden Stelle erläutert. Die Wirtschaftsinformatik sieht wie die spät beginnende Informatik ca. 12 Stunden für diesen Lernbereich vor. Das Fach Wirtschaftsinformatik steht zwar nicht im Fokus dieser Handreichung, allerdings sind die für dieses Fach formulierten Kompetenzerwartungen denen der spät beginnenden Informatik sehr ähnlich, sodass weite Teile der Inhalte der Handreichung eins zu eins auf die Wirtschaftsinformatik übertragen werden können.

Die folgenden Kapitel der Handreichung orientieren sich an den Kompetenzerwartungen des Lernbereichs. Kapitel 2 behandelt die Grundlagen und deckt damit im Wesentlichen die erste Kompetenzerwartung ab, Kapitel 3 und 4 beschäftigen sich mit dem Entscheidungsbaum- bzw. dem k-nächste-Nachbarn-Algorithmus und beziehen sich damit überwiegend auf die zweite und dritte Kompetenzerwartung. Kapitel 5 hat das Perzeptron und das künstliche neuronale Netz zum Thema, worüber in der vierten und fünften Kompetenzerwartung die Rede ist, und Kapitel 6 nimmt abschließend die Chancen und Risiken in den Blick, die in der sechsten Kompetenzerwartung entsprechend verankert sind.

Jedes der folgenden Kapitel beginnt mit einem Abschnitt, in dem zunächst die fachlichen Grundlagen dargelegt werden. Darauf folgt jeweils ein zweiter Abschnitt, in dem didaktische Hinweise zur unterrichtlichen Umsetzung gegeben werden, ggf. gefolgt von einem dritten Abschnitt mit Hinweisen und Erläuterungen zum dazugehörigen Material.

Die Handreichung richtet sich an Gymnasiallehrkräfte mit Fakultas in Informatik. Die Inhalte können aber auch für Lehrkräfte anderer Fächer bzw. anderer Schularten interessant sein, insbesondere für Wirtschaft-und-Recht-Lehrkräfte, die den Lernbereich „Künstliche Intelligenz“ im Fach Wirtschaftsinformatik unterrichten. Die Handreichung geht allerdings davon aus, dass die Leserin bzw. der Leser als Vorwissen ein vertieftes Lehramtsstudium in Informatik mitbringt, dennoch sind weite Teile der Handreichung hoffentlich auch für Leserinnen und Leser mit weniger Informatikkenntnissen verständlich und nachvollziehbar.

Die Handreichung ist so konzipiert, dass sie sich zum Selbststudium eignet. Sie geht von keinen Vorkenntnissen im Bereich der Künstlichen Intelligenz aus. Sie soll die Informatiklehrkräfte in die Lage versetzen, den Lernbereich kompetent zu unterrichten. Darüber hinaus sollen die bereitgestellten Materialien bei der unterrichtlichen Umsetzung unterstützen. Es wird empfohlen, den parallel zur Handreichung entwickelten Selbstlernkurs an der ALP (s. `links.alp.dillingen.de/ki`) entweder vorab oder mit der Handreichung verzahnt zu bearbeiten. An Selbstlernkurs und Handreichung schließt sich eine eintägige praxisbezogene Präsenzfortbildung im Rahmen der Regionalen Lehrerfortbildung (RLFB) an, in der die Inhalte vertieft und anhand von Beispielen praktisch umgesetzt werden (s. auch `links.alp.dillingen.de/ki`). Deren Besuch wird ebenfalls wärmstens empfohlen.

Die Materialien zur Handreichung werden über einen Kurs der *mebis Lernplattform* zur Verfügung gestellt (s. `https://mebis.link/NFF6EV`).

KAPITEL 2 Grundlagen der Künstlichen Intelligenz

„Schlägt man eine beliebige Tageszeitung auf, stehen die Chancen gut, dass in mindestens einem Artikel die Wörter 'Algorithmus', 'Big Data' oder 'künstliche Intelligenz' vorkommen. (...) Aber was genau steckt dahinter?“
Zweig (2019)

Überblick:

2.1 Fachliche Grundlagen

2.1.1 Ansätze zur Definition von Künstlicher Intelligenz

Die Definition von Künstlicher Intelligenz (KI) stellt eine besondere Herausforderung dar. In der Literatur gibt es hierzu unterschiedliche Ansätze. Doch bevor Künstliche Intelligenz definiert werden kann, sollte erst der Begriff „Intelligenz“ betrachtet werden.

2.1.1.1 Intelligenz

Zur Annäherung an den Begriff „Intelligenz“ können folgende Definitionen herangezogen werden. Intelligenz ist ...

> *„die Fähigkeit [des Menschen], abstrakt und vernünftig zu denken und daraus zweckvolles Handeln abzuleiten.“* (Duden-online)

> *„im allgemeinen Verständnis eine bestimmte Form der Begabung, die sich als Fähigkeit (oder eine Gruppe von verschiedenen Fähigkeiten) äußert, anschauliche sowie abstrakte Beziehungen zu erfassen, herzustellen und zu deuten und sich dadurch an neuartige Situationen anzupassen und sie gegebenenfalls durch problemlösendes Verhalten zu bewältigen.“* (Brockhaus-Enzyklopädie-Online)

Beide Definitionen von Intelligenz zielen auf die Fähigkeit ab, Probleme durch zweckvolles und zielgerichtetes Handeln zu lösen und dabei die Fähigkeiten zum abstrakten Begreifen von Situationen und Zusammenhängen einzusetzen. Einen weiteren Aspekt, der später auch für die Künstliche Intelligenz entscheidend ist, liefern folgende Definitionen. Intelligenz ist ...

> *„die Fähigkeit, aus Erfahrungen Nutzen zu ziehen und das Gegebene in Richtung auf das Mögliche zu überschreiten.“* (Zimbardo, 1995, S. 528)

> *„in der Psychologie ein hypothetisches Konstrukt (d. h. eine Erklärung für ein nicht direkt beobachtbares Phänomen), das die erworbenen kognitiven Fähigkeiten und Wissensbestände einer Person bezeichnet, die ihr zu einem gegebenen Zeitpunkt zur Verfügung stehen.“* (Gabler-Wirtschaftslexikon)

Intelligenz setzt nach diesen Ansätzen auch das Erwerben von Erkenntnissen und Fähigkeiten voraus, die dem Menschen dann zu einem bestimmten Zeitpunkt zur Verfügung stehen und genutzt werden können. Dies benötigt eine Vorarbeit im Wissenserwerb und im Training von Fähigkeiten und Fertigkeiten, die dann zielgerichtet zum Problemlösen eingesetzt werden können. Mithilfe dieser beiden Ansätze kann nun auch das Konzept der Künstlichen Intelligenz betrachtet werden.

2.1.1.2 Künstliche Intelligenz (KI)

Die Definitionen von KI divergieren bereits in deren grundlegenden Ansätzen. Beispielsweise unterscheiden Russel & Norvig (2012) Ansätze zur Definition von KI, die sich auf menschliches Denken und Handeln konzentrieren, von solchen, die rationales Denken und Handeln in den Vordergrund rücken.

Auf menschliches Denken und Handeln bezogene Definitionen lauten beispielsweise wie folgt:

> *„Der Begriff künstliche Intelligenz bezeichnet das Verhalten einer Maschine, das – wenn sich ein Mensch genauso verhält – als intelligent angesehen wird.“* (Simmons & Chappell, 1988, S. 14)

> *„Als Künstliche Intelligenz (KI) bezeichnet man eine Software, mit deren Hilfe ein Computer eine kognitive Tätigkeit ausführt, die normalerweise Menschen erledigen.“* (Zweig, 2019, S. 126)

> *„Künstliche Intelligenz ist die Lehre davon, wie man Computer dazu bringt, Dinge zu tun, die Menschen im Moment noch besser können.“* (Rich, 1983)

Derartige Definitionen machen die Künstliche Intelligenz an Fähigkeiten der menschlichen Vorbilder fest. Dabei ist zu beachten, dass diese Ansätze nur auf ausgewählte Fertigkeiten des Menschen abzielen. Ähnelt das Ergebnis der Maschine dem Verhalten eines Menschen oder übertrifft es dieses sogar, wird sie als intelligent angesehen. Aber wie kann festgestellt werden, ob eine Maschine etwas besser kann als ein Mensch? Mittlerweile kann sie besser Schach spielen und schneller große Datenmengen verarbeiten, aber ist sie dadurch auch intelligent im Sinne obiger Definition von Simmons & Chappell?

Ein bekanntes Experiment, das zur Beantwortung dieser Frage herangezogen werden kann, ist der Turing-Test. Den 1950 von Alan Turing entwickelten Test besteht ein Computer, wenn ein Mensch in einer schriftlichen Unterhaltung mit der Maschine nicht bestimmen kann, ob die Antworten von einem Mensch stammen oder nicht. Dafür muss ein Computer nach Russel & Norvig (2012) natürliche Sprache verarbeiten, Informationen speichern, logisch schließen und sich an neue Umstände anpassen, also lernen können. Obwohl Maschinen einzelne Fähigkeiten bereits haben, findet der Turing-Test heute noch in abgewandelter Form Anwendung, um „echte“ Menschen von Maschinen zu unterscheiden.

Um zu beweisen, dass ein Nutzer bestimmter Online-Systeme tatsächlich ein Mensch ist, muss er ein Captcha (s. Abbildung 2.1) lösen. Die Abkürzung steht dabei für **c**ompletly **a**utomated **p**ublic **t**uring test to tell **c**omputers and **h**umans **a**part, ein moderner Turing-Test. Besteht eine Maschine den Turing-Test, so kann sie menschliche Intelligenz simulieren und gilt daher als „künstlich intelligent“.

Abb. 2.1: Recaptcha von `Google`.

Darüber hinaus werden in der Literatur zur Definition von KI auch Ansätze verfolgt, die rationales Denken und Handeln in den Vordergrund stellen. Künstliche Intelligenz ist ...

> *„die Fähigkeit einer Maschine, menschliche Fähigkeiten wie logisches Denken, Lernen, Planen und Kreativität zu imitieren.“* (Europäisches Parlament, 2020)

> *„die Fähigkeit eines Systems, externe Daten richtig zu interpretieren, aus diesen Daten zu lernen und diese Erkenntnisse zu nutzen, um durch flexible Anpassung bestimmte Ziele und Aufgaben zu erreichen.“* (Kaplan & Haenlein, 2019)

> *„das Studium derjenigen mathematischen Formalismen, die es ermöglichen, wahrzunehmen, logisch zu schließen und zu agieren.“* (Winston, 1992)

Diese Ansätze zielen zum einen auf das Prinzip des logischen Schließens ab. Dabei werden „Muster für Argumentationsstrukturen [entwickelt] [...], die immer zu korrekten Schlüssen führten, wenn ihnen korrekte Prämissen [(Ausgangspunkt eines logischen Schlusses)] übergeben wurden“ (Russel & Norvig, 2012, S. 25). Die Maschine kann also Schlussfolgerungen für Eingaben treffen, die sie nach ihren persönlichen Logiksystemen bestimmt, und dementsprechend agieren und reagieren. Zum anderen wird in diesen Definitionen deutlich, dass sich KI-Systeme auch durch den Aspekt auszeichnen, aus Daten zu lernen, dadurch flexibel neue Herausforderungen zu meistern und kreativ Aufgaben zu lösen.

Auch nach Prof. Katharina Zweig unterliegen die Definitionen der künstlichen Intelligenz einem stetigen Wandel, wenn beispielsweise Maschinen neue Fähigkeiten erwerben. „Sobald ein Computer das Gewünschte tun kann, nehmen wir diese Tätigkeiten als weniger intelligent

wahr, gerade weil ein Computer sie kann“ (Zweig, 2019, S. 126). Lange wurde es nicht für möglich gehalten, dass ein Computer besser Schach spielen kann als ein Mensch. Im Jahr 1996 schlug jedoch Deep Blue den zu dieser Zeit amtierenden Schachweltmeister Garri Kasparow. Seitdem wird die Fähigkeit, besser als jeder Mensch Schach spielen zu können, nicht mehr verwendet, um eine Maschine als intelligent einzuschätzen. Die Definition von Künstlicher Intelligenz ist schwer umsetzbar und kann höchstens für den jeweiligen Entwicklungsstand erstellt werden. Da die „Definition so schwammig ist, dass sie nahezu nutzlos ist“ (Zweig, 2019, S. 126), bezeichnen manche Wissenschaftlerinnen und Wissenschaftler den Namen des Forschungsfeldes „Künstliche Intelligenz“ als Fehlbenennung.

2.1.2 Grundideen von Verfahren der Künstlichen Intelligenz

In den beiden in Abschnitt 2.1.1.2 erarbeiteten Definitionsansätzen spiegeln sich zwei grundsätzliche Verfahren der KI wider, wissensbasierte Systeme (insbesondere Expertensysteme) und maschinelles Lernen.

2.1.2.1 Expertensysteme

Expertensysteme können mithilfe einer vorgegebenen Wissensbasis und der Fähigkeit des logischen Schlussfolgerns Aufgaben lösen. Hierbei erfolgt eine Trennung von Wissensbasis und Verarbeitung des Wissens in der sogenannten Inferenzmaschine.

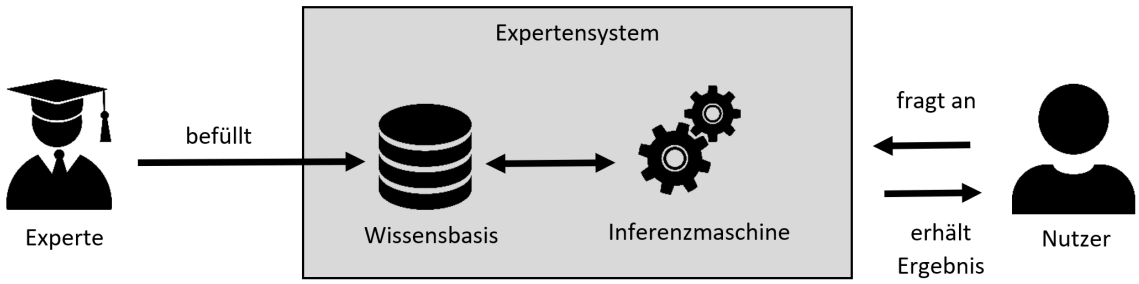


Abbildung 2.2: Grundsätzlicher Aufbau von Expertensystemen.

Ein Experte befüllt die Wissensbasis des Expertensystems mit Informationen, Zusammenhängen und Regeln (vgl. Lämmel & Cleve, 2020, S. 26). Eine Inferenzmaschine liefert Antworten auf die Anfragen des Nutzers durch logisches Schlussfolgern auf Basis der zugrunde liegenden

Wissensbasis (s. Abbildung 2.2). Dabei ist zu beachten, dass derartige Systeme die Wissensbasis nicht selbstständig verändern können. Eine Erweiterung oder Veränderung kann nur durch den Experten erfolgen.

Expertensysteme kommen immer dort zum Einsatz, wo die Wissensbasis erstellt und die logischen Zusammenhänge von der Inferenzmaschine interpretiert und angewendet werden können. Bei medizinischen Diagnosen, der Vorhersage von Erdbeben und der Fehlerdiagnose bei technischen Geräten liegen detaillierte Daten und Zusammenhänge vor, aus denen sich mithilfe von logischen Schlussfolgerungen Ergebnisse produzieren lassen. Beispiele für wissensbasierte Expertensysteme sind der oben genannte Schachcomputer Deep Blue, der 1997 den Schachgroßmeister Kasparow in einer Partie Schach schlug, und der Ratecomputer Watson, der von IBM entwickelt wurde und 2011 die menschlichen Mitspieler in der Fernseh-Quizshow „Jeopardy!“ besiegte.

Die Erstellung einer Wissensbasis sowie der Entwurf von Anfragen an Expertensysteme, die diese Wissensbasis verwenden, wird in Informatik Jgst. 13 (erhöhtes Anforderungsniveau) und daher in der Handreichung für die Jgst. 13 thematisiert.

2.1.2.2 Maschinelles Lernen

Ein weiteres grundsätzliches Verfahren der Künstlichen Intelligenz ist das maschinelle Lernen. In Jgst. 11 stehen ausgewählte Algorithmen dieses Ansatzes im Vordergrund. „Beim maschinellen Lernen werden auf Basis einer typischerweise großen Menge an Daten [notwendige] Regeln, Verhaltensweisen oder Muster abgeleitet bzw. identifiziert – also ‚gelernt‘. Das Gelernte wird in einem Modell gespeichert und kann im Anschluss auf neue Daten angewendet werden (s. Abbildung 2.3). [...] Maschinelles Lernen wird vor allem überall dort eingesetzt, wo es aufgrund der Charakteristik des Problems nicht effizient möglich ist, das Wissen so explizit zu repräsentieren, dass es ein Computer verarbeiten kann“ (`https://computingeducation.de/proj-ml-uebersicht/`).

Mithilfe dieses Verfahrens können Systeme erstellt werden, die Aufgaben lösen, die mit wissensbasierten Anwendungen alleine nicht gelöst werden können. Ein Beispiel dafür stellt der Spielcomputer AlphaGo dar. Go ist ein asiatisches Brettspiel, das im Hinblick auf die Zugmöglichkeiten erheblich komplexer als Schach ist. Das Spielfeld ist 19×19 Felder groß und

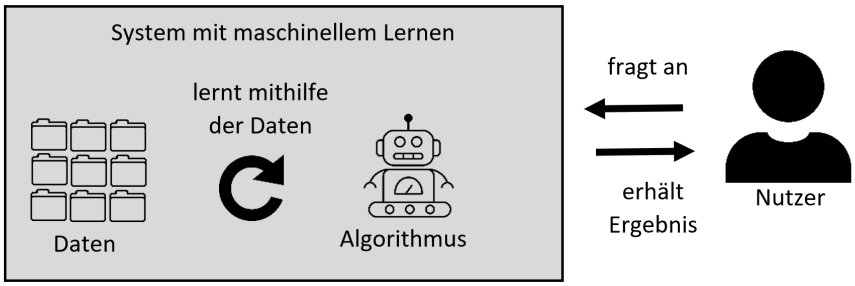


Abbildung 2.3: Grundsätzlicher Aufbau von Systemen mit maschinellem Lernen.

es gibt um die $2 \cdot 10^{170}$ Spielstellungen. Deshalb ist die Herangehensweise von Deep Blue, das Durchprobieren und Bewerten von Zugmöglichkeiten, auch mit enormer Rechenleistung nicht in überschaubarer Zeit möglich. Im Jahr 2016 besiegte AlphaGo (DeepMind) den weltbesten Go-Spieler, Lee Sedol. Das System setzte im Gegensatz zu Deep Blue auf Verfahren des maschinellen Lernens. Ende 2017 präsentierte die Firma DeepMind die KI AlphaZero, die innerhalb weniger Stunden die Spiele Schach und Go lernte und besser als jede Software war, die bis dato entwickelt wurde. AlphaZero bekam dabei nur die jeweiligen Spielregeln vorgegeben und erlernte die Spiele dann, indem es längere Zeit gegen sich selbst trainierte. Der KI wurden dabei keine menschlichen Spielstrategien gezeigt.

Ein weiteres Beispiel des maschinelles Lernens ist die Mustererkennung. Beispielsweise ließ Google Street View die Hausnummern auf den Bildern der Street-View-Kameras zuerst von Mitarbeitern bestimmen. Aufgrund der schieren Menge an Bildern war dies aber schnell nicht mehr in akzeptabler Zeit machbar. So entwickelte Google einen Algorithmus, der lernte, Hausnummern in den Aufnahmen zu erkennen.

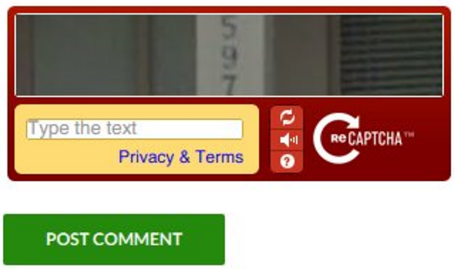


Abb. 2.4: Recaptcha von Google mit Foto einer Hausnummer[1].

Durch die Auswertung der Nutzereingaben in die Captcha Abfragen (reCAPTCHA) von Google wurde die Trefferquote des Algorithmus kontinuierlich verbessert (s. Abbildung 2.4). Den Nutzern wurden dabei immer wieder Bilder von Hausnummern gezeigt, die sie eingeben sollten. Dabei vertraute man darauf, dass der Großteil der Eingaben ein korrektes Ergebnis

[1] `https://pagepipe.com/how-google-no-captcha-captcha-slows-down-your-mobile-site/`

lieferte. Die Bilderkennung wurde dadurch so stark verbessert, dass sie jetzt auch die Captchas von Googles reCAPTCHA lösen kann.

2.1.3 Arten des maschinellen Lernens

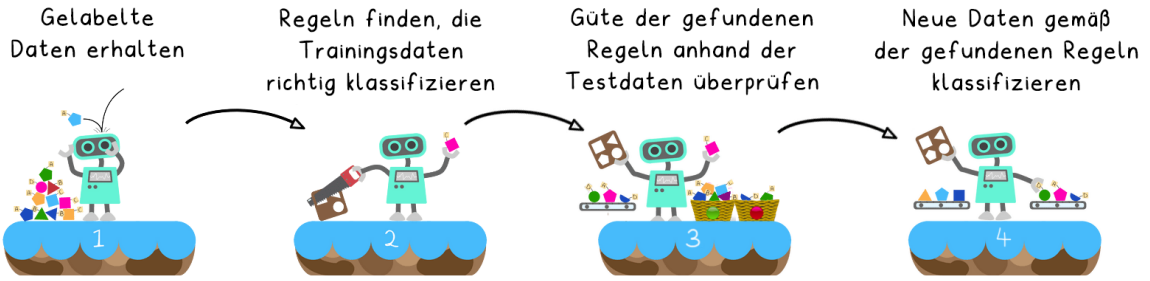
Um sich dem Thema „Maschinelles Lernen“ zu nähern, lohnt sich zunächst ein Blick auf die Definition von Lernen:

> *„Unter Lernen versteht man einen Prozess, der zu relativ stabilen Veränderungen im Verhalten oder im Verhaltenspotenzial führt und auf Erfahrung aufbaut.“* (Zimbardo, 1995, S. 263)

> *„Lernen ist damit ein Prozess, der individuell und erfahrungsbezogen konstruiert. Hierbei agiert die Person aktiv, indem sie ihre eigene Erfahrungs- und Erlebenswelt in den Lernprozess einbringt und dabei individuell vorhandenes Wissen und Können anpasst.“* (Gabler-Wirtschaftslexikon)

Lernen stellt einen fortlaufenden, nicht abschließenden Prozess dar, in dem das Verhalten, Wissen und Können durch die Erfahrungen angepasst und dadurch das Ergebnis verbessert wird. Dieses Vorgehen kann einfach auf maschinelles Lernen übertragen werden. Die Maschine verwendet Daten und Algorithmen, um neue Ergebnisse, Fähigkeiten oder Entscheidungen zu generieren. Dabei unterscheidet man die folgenden drei Arten maschinellen Lernens.

2.1.3.1 Überwachtes Lernen



Christoph Gräßl, Marco Hegmann, Wolfgang Pfeffer, Alexander Ruf, Johannes Wintermeier, adaptiert von „Überwachtes Lernen“ (https://computingeducation.de/proj-ml-uebersicht/) (CC-BY) von Stefan Seegerer, Tilman Michaeli & Sven Jatzlau.

Abb. 2.5: Überwachtes Lernen.

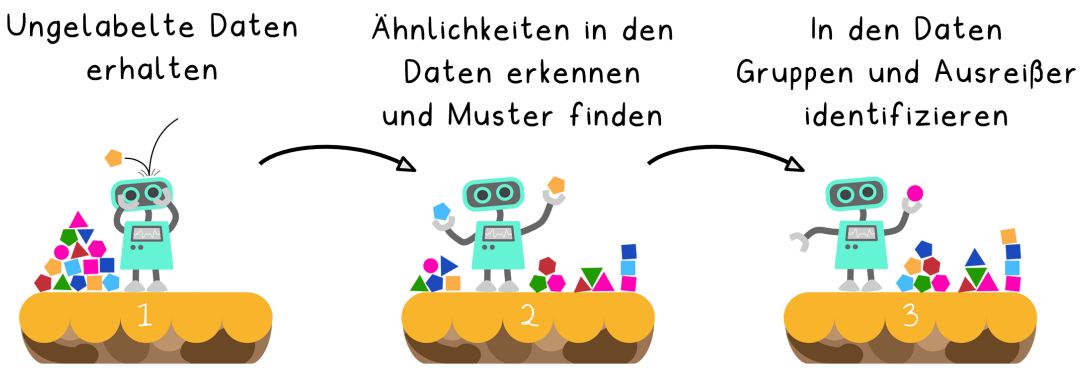
Beim überwachten Lernen (s. Abbildung 2.5) müssen gelabelte, d. h. bereits klassifizierte Daten, vorliegen (1). So können beispielsweise Bilder der Klassen Auto, Fahrradfahrer und Fußgänger, die jeweils als solche beschriftet wurden, verwendet werden, um eine Bilderkennung für das autonome Fahren zu ermöglichen.

„Gelabelt“ sind Daten, wenn ihnen im Vorfeld bereits eine Klasse zugeordnet wurde. Aus einem Teil dieser gelabelten Daten erstellt der Algorithmus Regeln, die eine Zuordnung der Daten zu den Klassen ermöglicht (2). Diese Daten werden Trainingsdaten genannt, weil der Algorithmus damit trainiert wird, d. h. er lernt, Daten selbstständig mithilfe der gefundenen Regeln zu klassifizieren. Da der Algorithmus bei diesem Verfahren mit Daten lernt, die vom Datenersteller gelabelt vorgegeben wurden, spricht man von „überwachtem“ Lernen.

Die Güte (Qualität) der gefunden Regeln wird mithilfe weiterer gelabelter Daten, den sogenannten Testdaten, überprüft (3).

Anschließend wendet der Algorithmus seine Regeln an, um neue, ungelabelte Daten zu klassifizieren (4).

2.1.3.2 Unüberwachtes Lernen



Christoph Gräßl, Marco Hegmann, Wolfgang Pfeffer, Alexander Ruf, Johannes Wintermeier, adaptiert von „Unüberwachtes Lernen“ (https://computingeducation.de/proj-ml-uebersicht/) (CC-BY) von Stefan Seegerer, Tilman Michaeli & Sven Jatzlau.

Abb. 2.6: Unüberwachtes Lernen.

Im Gegensatz zum überwachten Lernen stehen dem unüberwachten Lernen (s. Abbildung 2.6) keine gelabelten Daten zur Verfügung (1). Beispielsweise liegen bei der Bilderkennung viele Bilder von Früchten vor, eine Zuordnung zu unterschiedlichen Obstsorten ist allerdings nicht gegeben.

Der Algorithmus muss sich dabei die Ähnlichkeiten der Daten (Objekte) in einzelnen Merkmalen (Attribute) zunutze machen und dadurch die Daten gruppieren, die in diesen Merkmalen die gleichen oder ähnliche Ausprägungen (Attributwerte), also dasselbe Muster aufweisen. Dabei geht man davon aus, dass ähnliche Daten auch ähnliche Merkmalsausprägungen haben. Man spricht in diesem Fall von „unüberwachtem“ Lernen, weil die Daten vom Ersteller nicht gelabelt wurden und dem Algorithmus daher keine Klassen vorgegeben wurden (2). So werden beispielsweise alle gelben, gebogenen Früchte gruppiert und mit einem Label versehen und alle grünen, runden Früchte ebenfalls gruppiert und mit einem anderen Label versehen.

Nachdem auf diese Weise ähnliche Daten in einzelne Gruppen zusammengefasst wurden, müssen noch diejenigen Daten, die in keine der Gruppen passen (sogenannte Ausreißer), identifiziert werden (3). In dem Beispiel der Früchtegruppierung wären das einzelne Bilder exotischer Früchte.

2.1.3.3 Bestärkendes Lernen[2]

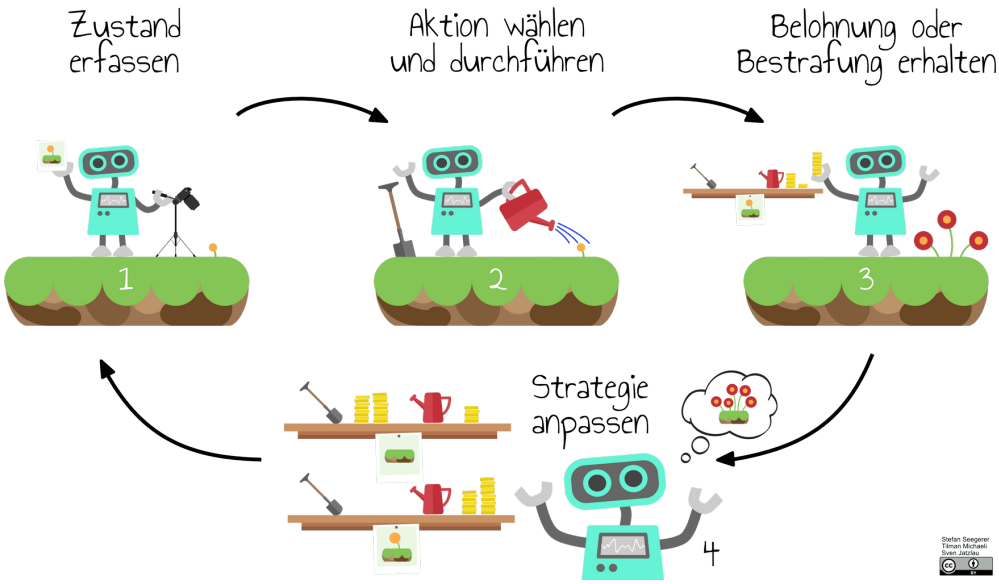


Abb. 2.7: Bestärkendes Lernen.

Im Gegensatz zu den beiden vorhergehenden Verfahren liegen beim bestärkenden Lernen (s. Abbildung 2.7) keine Daten vor. Der Algorithmus erfasst hier selbstständig Informationen, die im Hinblick auf sein Ziel relevant sind (1). So kann beispielsweise ein Serviceroboter das

[2]Das bestärkende Lernen wird in der Fachliteratur auch als verstärkendes Lernen oder Reinforcement Learning bezeichnet.

Verhalten und die Stimmung eines Kunden analysieren.

Aus seinen vorgegebenen Handlungsmöglichkeiten wählt der Algorithmus abhängig von dieser Analyse ein Vorgehen aus (2). Der Serviceroboter kann entsprechend seiner Analyse aus einer Vielzahl vorgefertigter Interaktionsmöglichkeiten wählen.

Wählt der Algorithmus die Alternative aus, die ihn näher an sein Ziel bringt, wird er belohnt. Hat eine Entscheidung einen negativen Einfluss auf die Zielerreichung, wird er bestraft (3). Durch die Antwort oder Reaktion des Kunden kann der Serviceroboter beispielsweise feststellen, ob seine Interaktionsauswahl zielführend war oder nicht.

Durch die positiven und negativen Rückmeldungen passt der Algorithmus seine Strategie an. Aktionen, die zum Ziel führen, werden „bestärkt“ und nicht erfolgreiche geschwächt (4). Bei einer positiven Rückmeldung wird der Serviceroboter die Handlung in einer ähnlichen Situation wiederholen. Reagiert der Kunde negativ auf das Verhalten des Roboters, wird diese Handlungsalternative mit einer geringeren Wahrscheinlichkeit eingesetzt.

2.1.4 Starke und schwache Künstliche Intelligenz

Man unterscheidet gemeinhin zwischen starker und schwacher Künstliche Intelligenz. Die zum Teil philosophische Diskussion um die Möglichkeit der Entwicklung starker KI-Systeme und die Abgrenzung zur schwachen KI wurde in der Literatur vielfach geführt. Nach Russel & Norvig agieren Maschinen, die schwache künstliche Intelligenz aufweisen, so, als wären sie intelligent (s. auch Abschnitt 2.1.1). Starke künstliche Intelligenz ist dagegen durch Systeme gekennzeichnet, die wirklich „denken [...] und nicht einfach nur Denken simulieren [...]“ (Russel & Norvig, 2012, S. 1176), also tatsächlich intelligent sind.

Um diesen Unterschied deutlich zu machen, können zwei Gedankenexperimente herangezogen werden. Alan Turing näherte sich bereits 1950 dieser Thematik, indem er mit dem bekannten Turing-Test nicht etwa klären wollte, ob eine Maschine denken kann, sondern ob sie einen „Verhaltensintelligenztest“ (Russel & Norvig, 2012, S. 1177) bestehen könnte (s. Abschnitt 2.1.2). Die Maschine musste daher nicht intelligent sein, sondern dem menschlichen Kommunikationspartner nur menschlich und dadurch intelligent erscheinen.

Einen ähnlichen Ansatz verfolgt das Gedankenexperiment des „Chinesischen Zimmers“ von John Searle. Dabei sitzt ein Mensch in einem verschlossenen Zimmer und beantwortet Fragen,

die ihm in chinesischen Schriftzeichen durch einen Schlitz in der Tür gereicht werden, obwohl er die chinesische Sprache nicht beherrscht. Als Hilfsmittel kann er dazu nur ein dickes Buch verwenden, das für jede der eingegebenen Fragen eine passende Antwort in chinesischen Schriftzeichen bereitstellt. Obwohl der Mensch damit weder die Sprache noch die Fragen versteht, beantwortet er sie richtig. Bauberger sieht hier Parallelen zum Übersetzungssystem DeepL. Dieser Algorithmus „beherrscht das Übersetzen zwischen mittlerweile mehr als zehn Sprachen (darunter auch Chinesisch) schon recht gut, aber das bedeutet nicht, dass die Computer, die dahinter stehen, irgendetwas verstehen“ (Bauberger, 2020, S. 130).

Eine Maschine zeigt in diesen Fällen eine Intelligenz, die bei genauer Betrachtung aber nicht auf Verstehen, sondern auf eine entsprechende Informationsverarbeitung unter Beachtung der möglichen Fälle zurückzuführen ist. Nach Russel & Norvig handelt es sich dabei klar um eine schwache künstliche Intelligenz. Was benötigt dann aber eine Maschine, damit sie als starke künstliche Intelligenz gilt?

Nach Ramge braucht die starke Künstliche Intelligenz ein Bewusstsein, das geprägt ist durch ein Selbstbild, eigene Interessen und der Fähigkeit, sich selbstständig weiterzuentwickeln (vgl. Ramge, 2018, S. 19). Obwohl einige Bestandteile, wie Lernen oder eine Selbstreflexion, heute schon möglich sind, bleibt das Bild des Roboters mit menschlichen Wesenszügen und Intelligenz, und damit der starken KI, doch weiterhin Science-Fiction.

Die Gegenbewegung stellt jedoch die Frage, was die KI denn sonst sein soll außer Informationsverarbeitung. Dieser Frage geht das Gedankenexperiment der Philosophen Clark Glymour und John Searle nach. Hierbei werden Schritt für Schritt alle Neuronen des menschlichen Gehirns durch elektronische Prothesen mit der gleichen Funktionalität ersetzt, ohne die Arbeitsweise des Gehirns zu unterbrechen (Russel & Norvig, 2012, S. 1186). Ab wann ist der Mensch nicht mehr intelligent, sondern verarbeitet „nur“ noch Informationen? Oder bleibt das Bewusstsein des Menschen in der Maschine erhalten, weil es ja ersetzt wird? Ist es dann doch möglich, einer Maschine eine umfassende menschliche Intelligenz beizubringen?

All diese Fragen können bisher nicht abschließend beantwortet werden. Fest steht allerdings, dass Maschinen für einzelne Aufgaben oft bessere Ergebnisse liefern können als Menschen. Die Computer Watson (s. auch Abschnitt 2.1.2.1) und AlphaZero (s. auch Abschnitt 2.1.2.2) zeigten eindrucksvoll, wie sie dem menschlichen Intellekt in ihren speziellen Spielen überlegen waren. Trotzdem würde man ihnen nie ein Bewusstsein oder eine Intelligenz jenseits ihres

Fachbereichs zusprechen. Bei schwacher Künstlicher Intelligenz handelt es sich daher um „spezialisierte Systeme, die innerhalb ihres klar umgrenzten Wirkungsrahmens zu Höchstleistungen fähig [...]“ und „[...] in der Lage [sind,] sich selbst zu optimieren“ (Simon, 2021, S. 43 f). Eine starke KI würde „auch in Situationen ohne genaue Faktenlage oder mit unklarem Handlungsziel“ intelligent reagieren. Diese Generalisten gibt es allerdings (noch) nicht. Daher sind alle bis heute vorliegenden Systeme mit künstlicher Intelligenz im Bereich der schwachen KI anzusiedeln.

2.1.5 Geschichte der Künstlichen Intelligenz

Folgende geschichtliche Aspekte der Künstlichen Intelligenz sind von Bedeutung:

1937	Alan Turing zeigt mit dem Halteproblem Grenzen intelligenter Maschinen auf.
1943	McCulloch und Pitt modellierten die ersten künstlichen Neuronen und die Kombination mit Bestandteilen der Aussagenlogik (`UND`, `ODER`, `NICHT`).
1950	Turing stellt die Definition von Intelligenz von Maschinen durch den Turing-Test vor.
1955	Arthur Samuel entwickelt ein erstes lernfähiges Programm für das Spiel Dame.
1956	Der Begriff der Künstlichen Intelligenz wird auf einer Konferenz im Darthmouth College eingeführt.
1966	Joseph Weizenbaum entwickelt den ersten Chatbot ELIZA, der Texteingaben verstehen kann.
1972	Alain Colmerauer erfindet die Logikprogrammiersprache PROLOG.
1972	De Dombal entwickelt ein Expertensystem zur Diagnose von Bauchkrankheiten.
1986	Das System Nettalk, ein künstliches neuronales Netz, lernt zu sprechen.
1995	Vapnik entwickelt die Support-Vector-Maschine zur Klassifikation und Regression von Objekten.
1997	Der Schachcomputer (Expertensystem) Deep Blue von IBM besiegt den Schachweltmeister Garri Kasparow.
2009	Erstes Google Self Driving Car fährt auf einem Freeway in den USA.

2011	Der Quizroboter Watson (Expertensystem) besiegt zwei menschliche Mitspieler in der Quiz-Show Jeopardy.
2011	Apples Sprachassistent Siri erscheint.
2015	Daimler stellt ersten autonomen LKW in Deutschland vor.
2016	Der Roboter AlphaGo besiegt führende Spieler im Spiel GO nur anhand der Spielregeln und Lernen durch intensives Spielen gegen sich selbst.
2017	AlphaZero, die Weiterentwicklung von AlphaGo, kann mehrere Spiele erlernen.
2018	„Project Debater“ von IBM tritt live gegen einen Menschen im Debattier-Duell an.
2019	„Duplex“ von Google kann selbstständig Termine, z. B. für Friseur- oder Arztbesuche vereinbaren.
2022	Der dialogbasierte Chatbot ChatGPT beeindruckt mit seinen Antworten auf Fragen und Befehle in den unterschiedlichsten Bereichen (z. B. Programmierung, Literatur, Pädagogik).

2.2 Didaktische Hinweise / Bezug zum Lehrplan

2.2.1 Einordnung in den LehrplanPLUS

Im LehrplanPLUS Informatik sowie spät beginnende Informatik findet sich in Jahrgangsstufe 11 folgende Kompetenzerwartung:

> *Die Schülerinnen und Schüler diskutieren Ansätze zur Definition des Begriffs Künstliche Intelligenz (KI), beschreiben verschiedene Grundideen von Verfahren der KI (u. a. maschinelles Lernen) sowie ihre Anwendungsbereiche.*

In diesem Rahmen sollen die Schülerinnen und Schüler unterschiedliche Definitionen von Künstlicher Intelligenz analysieren und gegenüberstellen. Durch die Beschäftigung mit diesen Ansätzen (s. Abschnitt 2.1.1) entsteht bei den Schülerinnen und Schülern ein tieferes Verständnis von der technischen und gesellschaftlichen Bedeutung des wissenschaftlichen

Forschungsgebiets der KI, aber auch der Herausforderung einer klaren Abtrennung und Bestimmung des Begriffs. Im Zuge dessen gewinnen die Schülerinnen und Schülern eine eigene Vorstellung des Begriffs Künstliche Intelligenz.

Zusätzlich verlangt diese Kompetenz auch Kenntnisse über Verfahren der Künstlichen Intelligenz und deren Anwendungsgebiete. Durch die Auseinandersetzung mit unterschiedlichen Definitionsansätzen werden den Schülerinnen und Schülern bereits die Grundideen von Expertensystemen und maschinellem Lernen bewusst (s. Abschnitt 2.1.2). Die Abgrenzung dieser Ansätze erfolgt zum einen durch die jeweils zugrunde liegenden Funktionalität. Zum anderen können anhand von Beispielen möglichen Einsatzgebiete abgesteckt und die Funktionsweisen nachvollzogen werden.

Da in den weiteren Kompetenzerwartungen dieses Lernbereichs lediglich Ansätze des maschinellen Lernens enthalten sind, sollte in Jgst. 11 ein Schwerpunkt auf diese gelegt werden. Die Expertensysteme dienen hier der knappen Thematisierung einer weiteren Grundidee der Künstlichen Intelligenz und bilden die Grundlage für eine tiefere Betrachtung in Jgst. 13 (erhöhtes Anforderungsniveau)[3]. Auch die Arten des maschinellen Lernens (s. Abschnitt 2.1.3) werden vertieft erst in Jgst. 13 thematisiert und dienen der Lehrkraft hier lediglich zur Einordnung der nachfolgenden Algorithmen. Der im Lehrplan genannte Entscheidungsbaum- bzw. k-nächste-Nachbarn-Algorithmus sowie das Perzeptron verwenden Verfahren des überwachten Lernens. Eine Thematisierung der drei Arten maschinellen Lernens ist im Unterricht der 11. Jgst. nicht erforderlich.

Die Entmystifizierung der Künstlichen Intelligenz durch die Bestimmung der Definition sowie die Abgrenzung unterschiedlicher Ansätze bietet die fachliche Grundlage für die folgende Kompetenzerwartung in Jgst. 11, die für Informatik und für spät beginnende Informatik gleich lautet:

> *Die Schülerinnen und Schüler nehmen zu ausgewählten aktuellen Einsatzmöglichkeiten der Künstlichen Intelligenz Stellung und bewerten Chancen und Risiken für Individuum und Gesellschaft.*

Eine Stellungnahme verlangt zum einen die fundierte fachlich-thematische Auseinandersetzung mit den geforderten Einsatzmöglichkeiten. Diese beinhaltet beispielsweise die Funktionalität

[3] `https://www.lehrplanplus.bayern.de/fachlehrplan/gymnasium/13/informatik/erhoeht`

der verwendeten Algorithmen oder die Umsetzungs- und Gestaltungsmöglichkeiten. Hierbei können die Schülerinnen und Schüler auf die Grundideen der Verfahren künstlicher Intelligenz zurückgreifen. Zum anderen ist für die Stellungnahme auch eine individuelle Wertung bzw. Beurteilung von neuen Problemstellungen (s. KMK EPA Informatik[4]) notwendig. Auch dafür können die Anwendungsmöglichkeiten der Grundideen sowie die Definitionsansätze der Künstlichen Intelligenz herangezogen werden. Damit können die Schülerinnen und Schüler Rückschlüsse auf die Auswirkungen der konkreten Systeme auf das Individuum und die Gesellschaft ableiten. Weitere Informationen zu dieser Kompetenz finden sich in Kapitel 6.

2.2.2 Durchführung

2.2.2.1 Einstieg „Mensch-Maschine“

Um handlungsorientiert einen Eindruck von der Thematik zu erhalten und die Schülerinnen und Schüler maschinelles Lernen erleben zu lassen, eignet sich das Spiel „Mensch, Maschine!“, das in vielen Ausprägungen angeboten wird (z. B. online Krokodilschach von Stefan Seegerer, `https://www.stefanseegerer.de/schlag-das-krokodil/`). Dabei kann die Spielfigur nur wie der Bauer im Schach gezogen werden. Im Unterricht bietet sich eine analoge Durchführung in Kleingruppen an. Als Grundlage für dieses Spiel wurden die Materialien der Jugendaktion des Wissenschaftsjahres 2019 des Bundesministeriums für Bildung und Forschung verwendet.

Grundsätzlich besteht diese Unterrichtseinheit dabei aus drei Teilen:

Vorbereitung

Bevor die Schülerinnen und Schüler das Spiel in Gruppen starten können, müssen erst die Materialien vorbereitet werden. Zur einfacheren Umsetzung wurden diese angepasst und erweitert.

Für die Durchführung müssen für jede Gruppe folgende Unterlagen vorbereitet werden (s. `0_WJ19_MM_Vorbereitungsanleitung.pdf` im Materialordner):

[4] `https://www.kmk.org/fileadmin/Dateien/veroeffentlichungen_beschluesse/1989/1989_12_01-EPA-Informatik.pdf`

Dateiname	Vorbereitung
`1_WJ19_MM_Spielbrett_Teil_1_Vorderseite` `2_WJ19_MM_Spielbrett_Teil_2_Rückseite`	Spielfeld Vorder- und Rückseite laminiert
`3_WJ19_MM_Spielfiguren`	Spielfiguren ausgedruckt, ausgeschnitten und laminiert oder alternative Spielsteine
`4_WJ19_MM_Situationskarten`	Situationskarten klein ausgedruckt, ausgeschnitten und laminiert
`5_WJ19_MM_Farbkarten optimiert`	Farbkarten klein ausgedruckt, ausgeschnitten und laminiert
`6_Zugübersicht A4 optimiert`	Ausgedruckt, ausgeschnitten und laminiert
`7_Ergebniszettel DIN A4 Tabelle`	Ausgedruckt
`8_Spielablauf A4 optimiert`	Ausgedruckt und laminiert

Das Spiel „Mensch, Maschine!“ wird zwischen den Rollen Mensch und Maschine auf einem Spielfeld mit 3 mal 3 Feldern gespielt. Dabei sollte die Rolle Maschine in einen Maschine-**Zugauswähler** und einen Maschine-**Zugausführer** aufgeteilt werde. Der Mensch startet immer mit der mittleren oder von ihm aus rechten Figur (s. Abbildung 2.8, (1)).

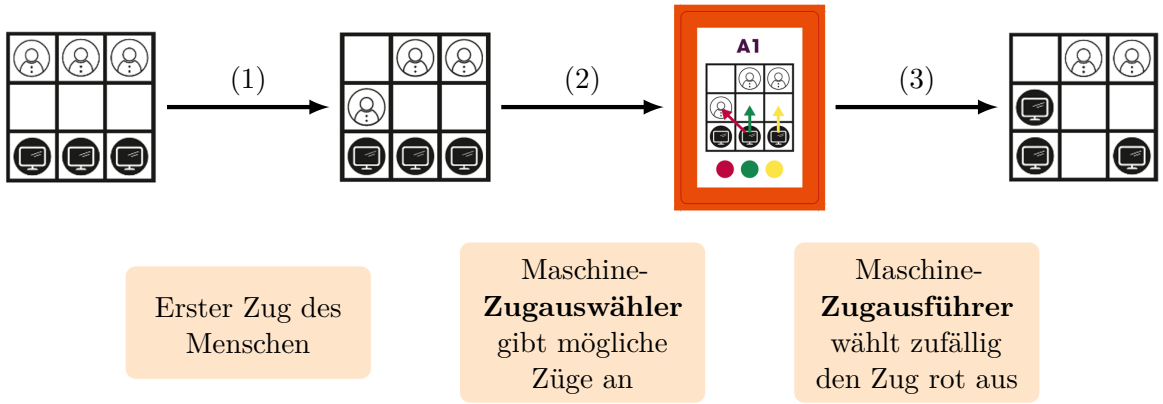
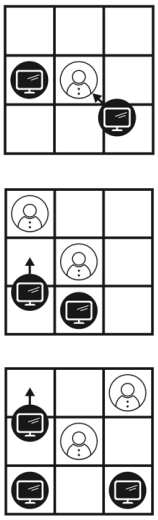


Abb. 2.8: Spielverlauf „Mensch, Maschine!“.

Diese Einschränkung erfolgt hier, um die Anzahl der möglichen Spielzüge zu beschränken und damit in weniger Spieldurchgängen den Lerneffekt der Maschine zu erkennen. Aufgrund der

Symmetrie des Spielfeldes hat diese Einschränkung keine Auswirkungen auf den grundsätzlichen Spielverlauf. Vor jedem Spielzug überprüft der Maschine-**Zugauswähler**, ob der Mensch oder die Maschine gewonnen hat. Wenn nicht, sucht er die entsprechende Situationskarte aus (2) und lässt den Maschine-**Zugführer** (3) einen möglichen Zug zufällig bestimmen.

Der Gegner ist besiegt, wenn ...



- ... alle seine Figuren aus dem Spiel geworfen wurden (diagonal wie die Bauern beim Schach),
- ... er am Zug ist, aber mit allen Figuren für weitere Bewegungen blockiert ist oder
- ... man mit einer eigenen Figur die Spielfeldseite des Gegners erreicht.

Verliert die Maschine, so wird der letzte gemachte Zug aus der Situationskarte gestrichen und kann in den nächsten Spielrunden nicht mehr verwendet werden. Eine detaillierte Spielanleitung findet sich im Dokument `8_Spielablauf A4 optimiert.pdf`, den möglichen Ablauf einer ersten Spielrunde in `Mensch_Maschine_erste_Spielrunde.pptx`. Der Erfolg der Durchführung des Spiels hängt stark vom Verständnis der Spielregeln ab. Daher empfiehlt es sich, einen Spieldurchgang mit einer Schülergruppe und einem Satz Material vorzuführen oder die Präsentation `Mensch_Maschine_erste_Spielrunde.pptx` zu verwenden.

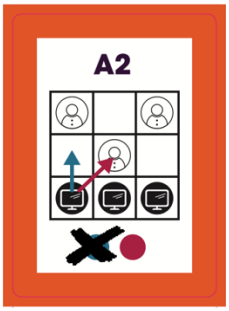


Abb. 2.9: Sieg- und Spielstellungen aus dem „Mensch-Maschine-Spiel“.

Dabei sollte bei der Vorstellung besonders auf die Handlungsanweisungen der einzelnen Rollen eingegangen werden. Nur bei einer strikten Befolgung der Anweisungen durch die Schülerinnen und Schüler kann das Spiel erfolgreich abgeschlossen werden.

In den Unterlagen findet sich auch die Light-Variante des Spiels, die weniger Vorbereitungsaufwand erfordert.

Durchführung

Nach einer gemeinsamen Erklärung spielen die Schülerinnen und Schüler in Gruppen von drei Personen (Mensch, Maschine-**Zugauswähler** und Maschine-**Zugausführer**) selbstständig. Die Lehrkraft steht für Fragen bereit und überprüft stichpunktartig die Einhaltung der Regeln. Bei Fehlern im Ablauf sollte das Spiel neu gestartet und die Zugübersicht und die Ergebnistabelle neu ausgeteilt werden. Das Spiel ist beendet, wenn die Ergebnisübersicht (18 Spielrunden) vollständig ausgefüllt ist.

Reflexion

Die Analyse der Ergebnisübersicht sollte ergeben, dass die Anzahl der Siege des menschlichen Spielers abnehmen, da die Maschine lernt, besser zu spielen. Aber wie funktioniert das? Beim Spiel „Mensch, Maschine!“ kommt eine Form des bestärkenden Lernens (s. Abschnitt 2.1.3) zum Einsatz. Die Daten stellen hier die Spielrunden mit den gewählten Spielzügen und daraus resultierenden Ergebnissen dar. Wichtig ist auch die Qualität des menschlichen Spielers. Verliert die Maschine nicht, entwickelt sie sich auch nicht weiter. Verliert sie aber, wird jeweils der letzte Zug, der in direkter Folge zum Verlieren geführt hat, eliminiert, d. h. aus der Ergebnisübersicht gestrichen. Dies stellt eine negative Bestärkung (Bestärkung erfolgreicher Handlung findet hier nicht statt) dar, da der Zug in Zukunft mit Sicherheit nicht mehr angewendet wird. Die Maschine lernt auf diese Weise besser zu spielen.

Das Spiel „Mensch, Maschine!“ nutzt bestärkendes Lernen, um die Maschine zu trainieren. Bei der Durchführung des Spiels steht das Erleben, wie eine Maschine lernt, im Vordergrund. Eine begriffliche Einordung in die Arten des maschinellen Lernens ist im Unterricht nicht erforderlich. Hingegen bietet sich an dieser Stelle eine kurze Abgrenzung von maschinellem Lernen zu Expertensystemen (s. Abschnitt 2.1.2) an.

Mithilfe der Erkenntnisse aus diesem Spiel kann anschließend die Frage nach der Intelligenz dieser Maschine gestellt und damit auf die möglichen Definitionen des Begriffs der Künstlichen Intelligenz übergeleitet werden (s. Abschnitt 2.2.2.2). Alternativ können auch anhand des „Lernens“ im Spiel „Mensch, Maschine!“ die Grundideen von Verfahren der KI thematisiert werden (s. Abschnitt 2.2.2.3).

2.2.2.2 Alternative 1: Ansätze zur Definition des Begriffs Künstliche Intelligenz

In vielen Definitionen (z. B. Zweig (2019), Rich (1983), Kaplan & Haenlein (2019) in Abschnitt 2.1.1.2) finden die Schülerinnen und Schüler die Erkenntnisse aus dem Spiel „Mensch, Maschine!“ wieder. Die Frage, ob sie den Maschinen-Gegner aus diesem Spiel als intelligent einschätzen, leitet direkt zum Problem über, dass der Begriff Künstliche Intelligenz nicht eindeutig definiert werden kann. Zuerst könnte mit den Ansätzen der Definition der „Intelligenz“ (s. Abschnitt 2.1.1.1) das Problem der genauen Abgrenzung besprochen werden. Aufbauend darauf kann mittels der unterschiedlichen Definitionsansätze von „Künstlicher Intelligenz“ von den Schülerinnen und Schülern eine eigene Definition verfasst werden. Diese kann genutzt werden, um für aktuelle Anwendungen (z. B. `ChatGPT`, `Thispersondoesnotexist.com`, `DeepL`) zu bestimmen, ob bzw. wie intelligent diese erscheinen. Obwohl eine Zuordnung dieser Anwendungen nicht immer möglich ist, können damit sowohl die Arten des maschinellen Lernens (s. Abschnitt 2.1.3) als auch die Abgrenzung zu Expertensystemen angesprochen werden. Dabei sollte darauf hingewiesen werden, dass in Jahrgangsstufe 11 nur Verfahren des überwachten Lernens thematisiert werden. Ein Verweis auf Jahrgangsstufe 13 bietet sich an dieser Stelle an. Abschließend kann mit Rückgriff auf die verfassten Definitionen die starke von der schwachen Künstlichen Intelligenz (s. Abschnitt 2.1.4) abgegrenzt und mithilfe des Turing-Tests (s. Abschnitt 2.1.1.2) weiter diskutiert werden. Eine abschließende Einschätzung der Schülerinnen und Schüler zur Möglichkeit der Entwicklung eines Systems mit starker Künstlicher Intelligenz kann beispielsweise in Form einer Meinungslinie umgesetzt werden. Dabei zieht die Lehrkraft eine Linie, an deren Enden sich jeweils die Endpositionen „starke KI ist umsetzbar“ und „starke KI ist nicht umsetzbar“ befinden. Die Schülerinnen und Schüler positionieren sich ihrer persönlichen Meinung entsprechend entlang der Linie. Je weiter sie von einer Endposition entfernt stehen, desto weniger stimmen sie dieser Aussage zu und umgekehrt. Anschließend begründen die Schülerinnen und Schüler ihre Positionierung. Die Lehrkraft wählt dabei Vertreterinnen und Vertreter unterschiedlicher Meinungen aus. Durch diese Methode erfolgt eine individuelle Reflexion des Themas sowie eine visuelle Verdeutlichung der Meinung innerhalb der Klasse, die für die weitere Erarbeitung genutzt werden kann.

2.2.2.3 Alternative 2: Grundideen von Verfahren der Künstlichen Intelligenz sowie ihrer Anwendungsbereiche

Das im Spiel „Mensch, Maschine!“ erlebte „Verbessern“ kann auch in vielen Definitionen (z. B. Europäisches Parlament (2020), Kaplan & Haenlein (2019) in Abschnitt 2.1.1.2) wiedergefunden werden. Dabei kennen die Schülerinnen und Schüler „Lernen“ aber nicht nur aus den Erfahrungen des Spiels, sondern auch aus ihrem eigenen Leben. Mithilfe dieser Definitionen und konkreten Anwendungsbeispielen können sowohl die Arten des maschinellen Lernens (s. Abschnitt 2.1.3) als auch die Abgrenzung zu den Expertensystemen (s. Abschnitt 2.1.2) angesprochen werden. Dabei sollen die Schülerinnen und Schüler erkennen, in welchen Situationen maschinelles Lernen eingesetzt wird und ggf. bereits Chancen und Risiken ableiten. Dazu können Beispiele wie die Radikalisierung von Robotern auf Social-Media-Plattformen[5] herangezogen werden. Eine detaillierte Betrachtung der Auswirkungen auf Individuen und die Gesellschaft erfolgt allerdings erst am Ende des Lernbereichs. Aufbauend auf diesem Ausblick bietet sich die Abgrenzung von starker und schwacher künstlicher Intelligenz (s. Abschnitt 2.1.4) an. Abschließend können die Schülerinnen und Schüler beispielswiese mithilfe einer Meinungslinie abschätzen, ob sie die Entwicklung eines Systems mit starker Künstlicher Intelligenz als möglich erachten.

[5] https://www.sueddeutsche.de/digital/chatbot-blender-facebook-1.4922049

KAPITEL 3 Entscheidungsbaum-Algorithmus

„Decision trees are simple yet successful techniques for predicting and explaining the relationship between some measurements about an item and its target value."
Rokach & Maimon (2015)

Überblick:

3.1 Fachliche Grundlagen

3.1.1 Überblick

Entscheidungsbäume sind ein etabliertes Modell des überwachten maschinellen Lernens. Sie werden sowohl für Klassifikations- als auch Regressionsaufgaben verwendet. Der Fokus dieser Handreichung liegt auf der Verwendung von Entscheidungsbäumen zur Klassifikation, d. h. mithilfe des Entscheidungsbaums sollen Daten klassifiziert, also bestimmten Labeln zugeordnet werden.

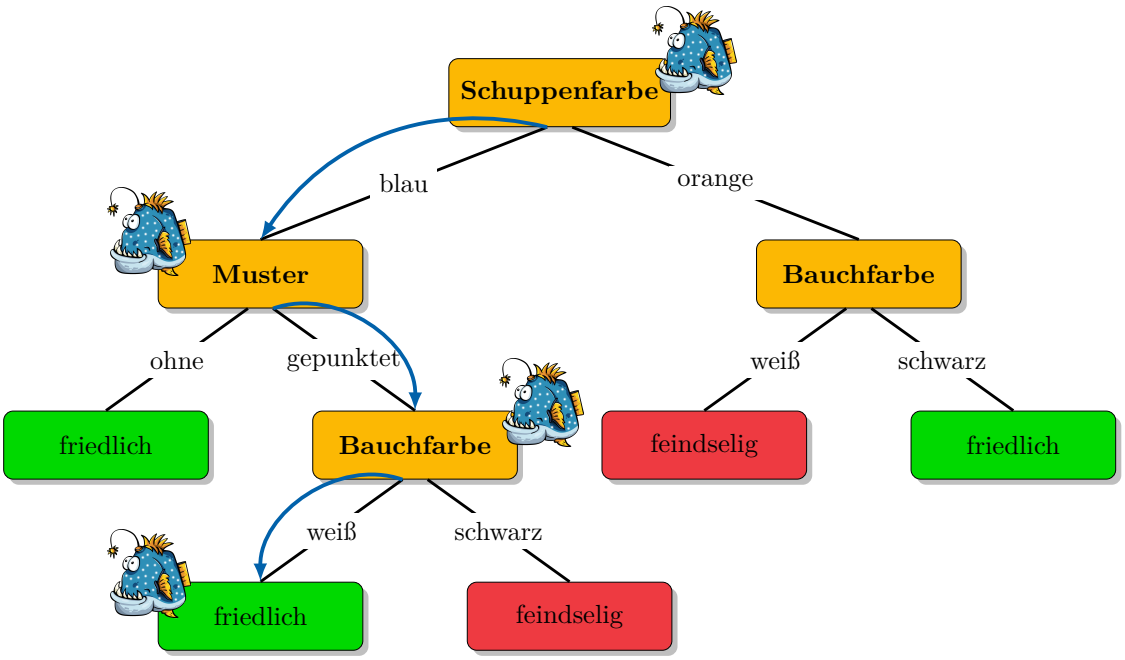


Abb. 3.1: Der Fisch wird gemäß dem Entscheidungsbaum als friedlich eingestuft. Die Abbildung visualisiert dabei den schrittweisen Ablauf der Klassifikation.

Ein Entscheidungsbaum besteht aus Knoten und Kanten, wobei die inneren Knoten Fragen oder (Entscheidungs-)Regeln repräsentieren, die auf ein bestimmtes Merkmal (Attribut) des zu klassifizierenden Datenpunkts (Objekts) angewendet werden. Die Kanten repräsentieren die möglichen Antworten auf diese Frage bzw. die möglichen Anwendungsfälle der Regel und symbolisieren somit die unterschiedlichen Attributwerte, wobei diese ggf. auch gruppiert sein können. Im Falle eines binären Entscheidungsbaums gibt es zu jedem inneren Knoten zwei ausgehende Kanten, z. B. eine „Ja“-Kante und eine „Nein“-Kante. Durch fortlaufendes (rekursives) Weiterleiten des zu klassifizierenden Datenpunkts entlang der entsprechenden Kanten

wird schließlich ein Blattknoten erreicht. Die Blattknoten repräsentieren dabei die endgültige Klassifizierung des Datenpunkts. Abbildung 3.1 visualisiert den Ablauf der Klassifizierung eines Datenpunkts anhand eines Entscheidungsbaums.

Ziel des hier vorgestellten Verfahrens des maschinellen Lernens ist es, auf Grundlage bereits gelabelter Daten (Trainingsdaten) einen Entscheidungsbaum zu erstellen, mit dem neue Daten möglichst zuverlässig klassifiziert werden können. Dazu dient der Entscheidungsbaum-Algorithmus, der auf Grundlage der Trainingsdaten und bestimmter Kriterien (s. Abschnitt 3.1.3.1) zunächst ein geeignetes Attribut (Frage / Regel) auswählt, das die vorhandenen Trainingsdaten in zwei oder mehrere (disjunkte) Mengen aufteilt. Für diese Teilmengen wählt der Algorithmus jeweils wieder ein geeignetes Attribut, um sie weiter aufzuteilen. Dieses Vorgehen wird nun für die entstandenen Teilmengen rekursiv wiederholt, bis z. B. eine zuvor festgelegte Baumtiefe erreicht ist oder bis alle Daten, die durch einen bestimmten (Blatt-)Knoten repräsentiert werden, dasselbe Label besitzen (s. Abschnitt 3.1.3.2).

Entscheidungsbäume können auf eine Vielzahl von Szenarien angewendet werden und sind leicht zu visualisieren und zu interpretieren. Bei diesem Verfahren handelt es sich jedoch um ein heuristisches Vorgehen, d. h. es gibt keine Garantie, dass der „beste“ bzw. überhaupt „ein guter“ Entscheidungsbaum gefunden wird. Es ist somit unerlässlich, den erhaltenen Entscheidungsbaum anhand von Testdaten auf seine „Vorhersagequalität“ hin zu überprüfen (s. Abschnitt 3.1.4).

3.1.2 Trainings-, Validierungs- und Testdaten

Als Verfahren des überwachten maschinellen Lernens benötigt der Entscheidungsbaum-Algorithmus Trainingsdaten, anhand derer das Modell, also der Entscheidungsbaum, generiert wird. Wie bereits im vorangegangenen Abschnitt erwähnt, handelt es sich bei dem Verfahren nur um eine Heuristik und das generierte Modell muss anhand weiterer gelabelter Daten auf seine Güte, d. h. die „Vorhersagequalität“ des Entscheidungsbaums getestet werden. Eine typische Fehlvorstellung ist es, hier dieselben Daten zu verwenden, anhand derer das Modell erstellt wurde. Da das Modell genau auf diesen Daten trainiert wurde, kann es diese in der Regel zuverlässig klassifizieren. Daraus kann man jedoch nicht schließen, dass neue, ihm in der Trainingsphase unbekannte Daten ebenso zuverlässig klassifiziert werden können. Daher ist es von großer Bedeutung, das Modell mit weiteren bereits gelabelten Daten (Testdaten)

zu bewerten, die nicht in der Trainingsphase verwendet wurden, um eine Einschätzung der allgemeinen Zuverlässigkeit des Modells bei der Klassifizierung zu bekommen.

Noch bevor der Entscheidungsbaum getestet wird, findet üblicherweise eine Phase der Optimierung statt. Hierbei kommt ein weiterer Satz gelabelter Daten (Validierungsdaten) zum Einsatz. Im Zuge der Optimierung kann insbesondere eine Überanpassung (*overfitting*) des Entscheidungsbaums auf die Trainingsdaten erkannt werden. Eine Überanpassung liegt vor, wenn das Modell so sehr auf die Trainingsdaten zugeschnitten ist, dass neue Daten weniger zuverlässig klassifiziert werden als mit einem generalisierten Modell, das in Bezug auf die Trainingsdaten weniger spezifisch ist. Das Model hat die Trainingsdaten sozusagen „auswendig gelernt“ und kann andersartige Datenobjekte nicht zuverlässig klassifizieren. Um den trainierten Entscheidungsbaum zu vereinfachen und damit zu generalisieren, können beispielsweise zu spezielle Knoten und / oder Zweige entfernt werden. Man spricht in diesem Zusammenhang von Pruning.

Zusammengefasst benötigt man Daten mit bekannten Labeln für folgende Zwecke: für das Trainieren des Modells (**Trainingsdaten**), ggf. für das Optimieren bzw. Validieren des trainierten Modells (**Validierungsdaten**) und für das Testen des Modells (**Testdaten**). Man muss deshalb die anfangs verfügbaren gelabelten Daten in zwei bzw. drei Teile aufteilen, wobei es keine feste Regel gibt, in welchem Verhältnis die Daten aufgeteilt werden sollten. Es ist gängige Praxis, etwa 70 % der Daten als Trainingsdaten zu verwenden und die restlichen Daten ungefähr gleichmäßig in Validierungs- und Testdaten aufzuteilen.

3.1.3 Erstellung des Entscheidungsbaums anhand der Trainingsdaten

Um den Entscheidungsbaum zu erstellen, muss der Algorithmus zunächst eine „beste“ Frage / Regel auswählen, um den Wurzelknoten in zwei oder mehrere Kindknoten aufzuteilen. Als Entscheidungsregel kommt die Einordnung eines Datenobjekts gemäß der Attributwerte eines seiner Attribute infrage. Je nach Anzahl der unterschiedlichen Attributwerte ergibt sich folglich eine Aufteilung in die Kindknoten. Dabei können auch mehrere Ausprägungen gruppiert werden, was gerade bei nicht diskreten Attributwerten sinnvoll ist. Es muss an dieser Stelle geklärt werden, welches das „beste“ bzw. „wichtigste“ Attribut ist, nach dessen Attributwerten eine Datenmenge aufgeteilt werden soll.

Im folgenden Beispiel sollen unterschiedliche Figuren mit den Merkmalen `Form` (Kreis / Dreieck),

`Streifen` (ja / nein) und `Beschriftung KI` (ja / nein) klassifiziert werden (s. linke Grafik in Abb. 3.2), wobei von Interesse ist, ob die Figuren grün oder rot sind. Die Zuordnung einer Figur zu einer der beiden Klassen erfolgt hierbei über ein grünes bzw. rotes Label (s. rechte Grafik in Abb. 3.2). Es liegen folgende gelabelte Trainingsdaten vor:

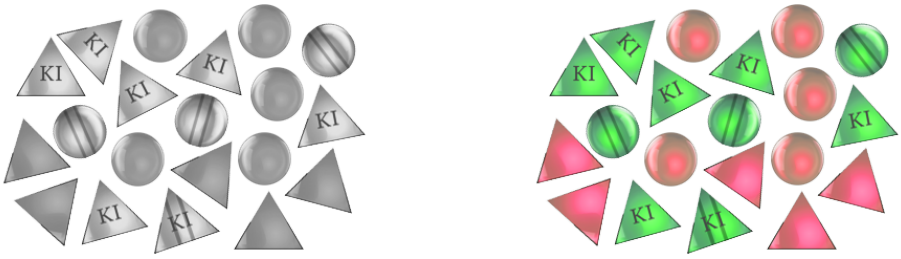


Abb. 3.2: Als Trainingsdaten werden Daten mit den Merkmalen `Form`, `Streifen` und `Beschriftung KI` betrachtet. Die beiden Abbildungen zeigen die Trainingsdaten ohne Label (links) und mit Label (rechts), welche durch die Farben grün und rot repräsentiert werden.

Je nachdem, welches Attribut als erste Entscheidungsregel ausgewählt wird, ergeben sich folgende Aufteilungen der Trainingsdaten:

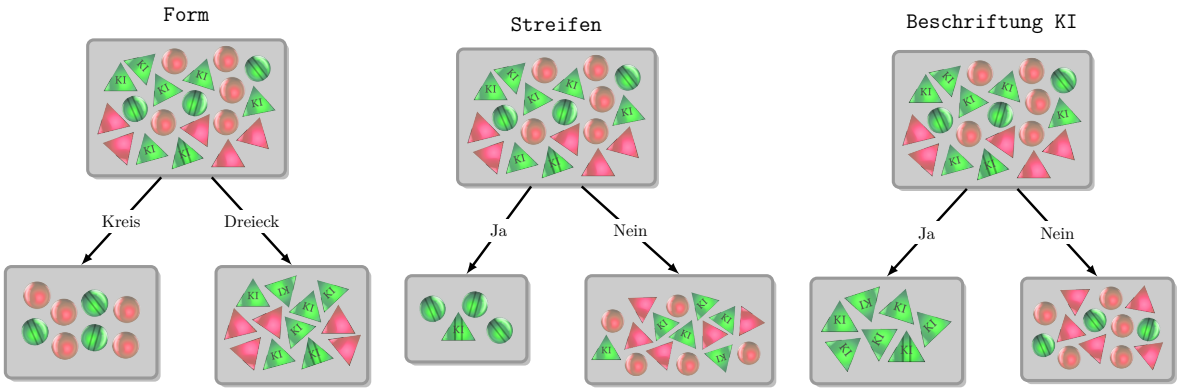


Abb. 3.3: Aufteilung der Trainingsdaten nach den drei möglichen Merkmalen `Form` bzw. `Streifen` bzw. `Beschriftung KI`.

Der Algorithmus entscheidet sich dabei für das Attribut, durch dessen Attributwerte die Datenobjekte in Untergruppen mit einem möglichst großen **Informationsgewinn** aufgeteilt werden, d. h. nach der Aufteilung sollte man die einzelnen Datenobjekte aus den Trainingsdaten mit einer höheren Wahrscheinlichkeit dem passenden Label zuordnen können als vor dem Aufteilen. Man betrachtet hierzu den **Informationsgehalt** der Ausgangsmenge und den Informationsgehalt nach dem Aufteilen der Ausgangsmenge in Teilmengen. Der Informationsgewinn wird dabei als Differenz aus Informationsgehalt vor und nach dem Aufteilen definiert. Zur

Messung des Informationsgehalts einer Datenmenge gibt es in Bezug auf Entscheidungsbäume drei gängige Verfahren (Split-Kriterien): die **Fehlklassifikationsrate**, die **Entropie** und die **Gini-Impurity** (vgl. Rutkowski et al., 2020). Diese drei Verfahren werden in den nächsten Unterabschnitten anhand von Beispielen illustriert.

3.1.3.1 Split-Kriterien

3.1.3.1.1 Fehlklassifikationsrate

Wie der Name schon erahnen lässt, betrachtet man bei diesem Verfahren die Wahrscheinlichkeit, mit der man Daten einer Menge fehlklassifiziert, d. h. dem falschen Label zuordnet. Zur Bestimmung des Informationsgehaltes vor dem Aufteilen müsste nebenstehender Menge entweder das Label grün oder rot zugewiesen werden.

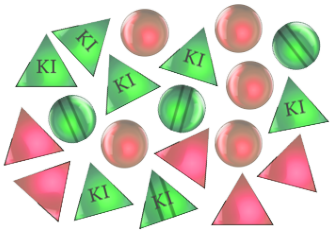


Abb. 3.4: Gelabelte Trainingsdaten.

Da in der Menge jeweils zehn rote und grüne Datenobjekte vorhanden sind, macht man in diesem Fall unabhängig von der Wahl des Labels zu $\frac{10}{20}$ einen Fehler, d. h. der Informationsgehalt vor dem Aufteilen beträgt $\frac{10}{20} = 0{,}5$.

Nun betrachtet man die Fehlerwahrscheinlichkeiten nach dem Aufteilen der Menge der Datenobjekte bzgl. des Attributs `Beschriftung KI`. In der linken Teilmenge mit Attributwert „ja“ befinden sich sieben Datenobjekte mit grünem Label. Wenn man dieser Menge das Label grün zuweist, macht man in $\frac{0}{7}$ der Fälle einen Fehler. In der rechten Menge befinden sich drei grüne und zehn rote Daten. Da sich mehr rote als grüne Daten in der Menge befinden, weist man dieser das Label rot zu und macht dann in $\frac{3}{13}$ der Fälle einen Fehler.

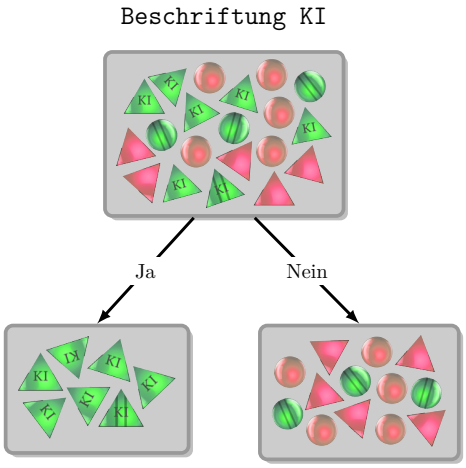


Abb. 3.5: Aufteilung der Trainingsdaten gemäß dem Attribut `Beschriftung KI`.

Diese beiden Fehlerraten müssen nun noch mit dem jeweiligen Anteil der Daten an der

Ausgangsmenge gewichtet werden (gewichtetes Mittel), um ein Maß für den Informationsgehalt nach dem Aufteilen zu erhalten:

$$\underbrace{\frac{7}{20}}_{\text{Gewichtung}} \cdot \underbrace{\frac{0}{7}}_{\text{Fehlerrate}} + \underbrace{\frac{13}{20}}_{\text{Gewichtung}} \cdot \underbrace{\frac{3}{13}}_{\text{Fehlerrate}} = \frac{3}{20} = 0{,}15$$

Es ergibt sich durch das Aufteilen einer Datenmenge X gemäß dem Attribut `Beschriftung KI` also ein Informationsgewinn IG_F unter Verwendung der Fehlklassifikationsrate F von:

$$\mathrm{IG}_F(X,\texttt{Beschriftung KI}) = \underbrace{\frac{10}{20}}_{\substack{\text{Informationsgehalt}\\ \text{vor dem Aufteilen}}} - \underbrace{\frac{3}{20}}_{\substack{\text{Informationsgehalt}\\ \text{nach dem Aufteilen}}} = \frac{7}{20} = 0{,}35$$

Das Verfahren lässt sich auf das Aufteilen einer Datenmenge in mehr als zwei Teilmengen übertragen. Man berechnet hierzu eine gemeinsame (gewichtete) Fehlerrate der Teilmengen und subtrahiert diese anschließend von der Fehlerrate der Ausgangsmenge.

Anmerkung

Betrachtet man die Berechnung der gewichteten Fehlerrate nach dem Aufteilen, erkennt man, dass sich die Anzahl der Elemente in den jeweiligen Teilmengen „kürzt“:

$$\frac{\mathbf{7}}{20} \cdot \frac{0}{\mathbf{7}} + \frac{\mathbf{13}}{20} \cdot \frac{3}{\mathbf{13}} = \frac{3}{20} = 0{,}15$$

Es reicht deshalb aus, die Anzahl der Fehler vor dem Aufteilen und die Anzahl der Fehler nach dem Aufteilen zu betrachten. Vor dem Aufteilen erhält man unabhängig von der Wahl des Labels 10 Fehler, nach dem Aufteilen in der linken Teilmenge 0 und in der rechten Teilmenge 3, also insgesamt 3 Fehler. Es ergibt sich somit ein Informationsgewinn von

„10 Fehler – 3 Fehler = 7 Fehler“

Der Wert „7 Fehler“ als Informationsgewinn bedeutet dabei, dass man durch das Aufteilen 7 Fehler weniger macht, als dies vor dem Aufteilen der Fall war.

3.1.3.1.2 Entropie

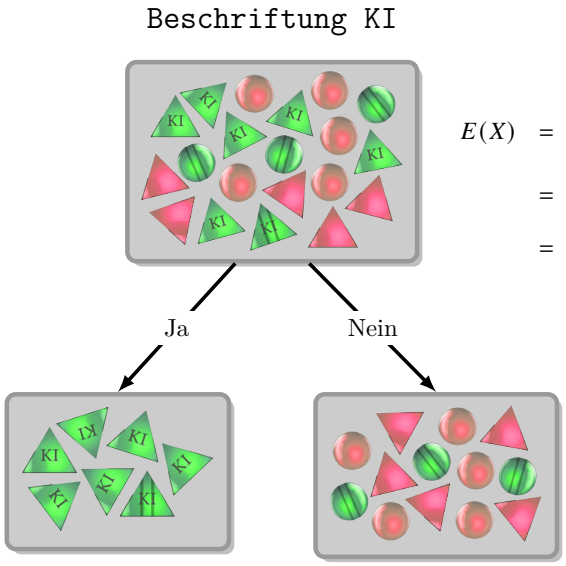
Ein anderes Maß für den Informationsgehalt einer Menge ist die **Entropie**. Diese misst anschaulich gesprochen die Unordnung in einer Menge. Für eine Menge X, in der Daten mit k verschiedenen Labeln mit der jeweiligen relativen Häufigkeit p_j $(j = 1, \ldots, k)$ vorkommen, wird die Entropie E wie folgt berechnet:

$$E(X) = -\sum_{j=1}^{k} p_j \cdot \log_2(p_j)$$

Gibt es in der Menge X nur Daten mit zwei verschiedenen Labeln (wie in obigem Beispiel grün und rot), vereinfacht sich die Formel zu:

$$E(X) = -\left(p_1 \cdot \log_2(p_1) + p_2 \cdot \log_2(p_2)\right)$$

Hierzu wird wieder das obige Beispiel betrachtet:



$$\begin{aligned} E(X) &= -\left(\frac{10}{20} \cdot \log_2\left(\frac{10}{20}\right) + \frac{10}{20} \cdot \log_2\left(\frac{10}{20}\right)\right) \\ &= -\left(0{,}5 \cdot \log_2(0{,}5) + 0{,}5 \cdot \log_2(0{,}5)\right) \\ &= 1 \end{aligned}$$

$$\begin{aligned} E(X_1) &= -\left(\frac{7}{7} \cdot \log_2\left(\frac{7}{7}\right)\right) \\ &= 0 \end{aligned}$$

$$\begin{aligned} E(X_2) &= -\left(\frac{3}{13} \cdot \log_2\left(\frac{3}{13}\right) + \frac{10}{13} \cdot \log_2\left(\frac{10}{13}\right)\right) \\ &\approx 0{,}78 \end{aligned}$$

Abb. 3.6: Berechnung der Entropien der Ausgangsmenge X bzw. der Teilmengen X_1 und X_2 nach der Aufteilung der Daten hinsichtlich des Attributs `Beschriftung KI`.

Der Informationsgewinn IG_E hinsichtlich des Attributs `Beschriftung KI` berechnet sich

analog zum Informationsgewinn IG_F aus Abschnitt 3.1.3.1.1 aus der Differenz der Entropie der Ausgangsmenge X und der Summe der gewichteten Entropien der Teilmengen X_1 und X_2:

$$\begin{aligned}\mathrm{IG}_E(X,\texttt{Beschriftung KI}) &= \underbrace{E(X)}_{\substack{\text{Informationsgehalt}\\ \text{vor dem Aufteilen}}} - \underbrace{\left(\frac{7}{20}\cdot E(X_1)+\frac{13}{20}\cdot E(X_2)\right)}_{\substack{\text{Informationsgehalt}\\ \text{nach dem Aufteilen}}}\\ &\approx 1-\frac{7}{20}\cdot 0-\frac{13}{20}\cdot 0{,}78 \approx 0{,}49\end{aligned}$$

3.1.3.1.3 Gini-Impurity

Das dritte Maß für den Informationsgehalt einer Menge, das in der Handreichung betrachtet wird, ist die **Gini-Impurity**. Für eine Menge X, in der Daten mit k verschiedenen Labeln mit der jeweiligen relativen Häufigkeit p_j $(j = 1,\ldots,k)$ vorkommen, wird die Gini-Impurity G wie folgt berechnet:

$$G(X) = 1 - \sum_{j=1}^{k} p_i^2$$

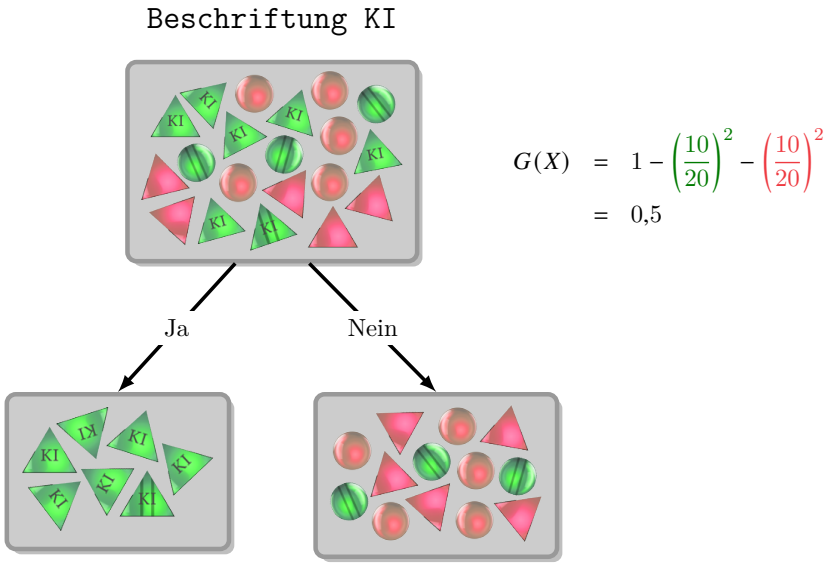
Kommen in der Menge nur Daten mit zwei verschiedenen Labeln vor (wie in obigem Beispiel grün und rot), vereinfacht sich die Formel zu:

$$G(X) = 1 - p_1^2 - p_2^2$$

Hierzu wird wieder das obige Beispiel betrachtet (s. Abbildung 3.7).

Der Informationsgewinn IG_G hinsichtlich des Attributs `Beschriftung KI` berechnet sich analog zum Informationsgewinn IG_F bzw. IG_G aus den Abschnitten 3.1.3.1.1 bzw. 3.1.3.1.2 aus der Differenz der Gini-Impurity der Ausgangsmenge X und der Summe der gewichteten Werte der jeweiligen Gini-Impurity der Teilmengen X_1 und X_2:

$$\begin{aligned}\mathrm{IG}_G(X,\texttt{Beschriftung KI}) &= \underbrace{G(X)}_{\substack{\text{Informationsgehalt}\\ \text{vor dem Aufteilen}}} - \underbrace{\left(\frac{7}{20}\cdot G(X_1)+\frac{13}{20}\cdot G(X_2)\right)}_{\substack{\text{Informationsgehalt}\\ \text{nach dem Aufteilen}}}\\ &\approx 0{,}5-\frac{7}{20}\cdot 0-\frac{13}{20}\cdot 0{,}36 \approx 0{,}27\end{aligned}$$



$$G(X) = 1 - \left(\frac{10}{20}\right)^2 - \left(\frac{10}{20}\right)^2 = 0{,}5$$

$$G(X_1) = 1 - \left(\frac{7}{7}\right)^2 = 0$$

$$G(X_2) = 1 - \left(\frac{3}{13}\right)^2 - \left(\frac{10}{13}\right)^2 \approx 0{,}36$$

Abb. 3.7: Berechnung der Gini-Werte der Ausgangsmenge bzw. der Teilmengen nach der Aufteilung der Daten hinsichtlich des Attributs `Beschriftung KI`.

Exkurs (Was steckt hinter der Formel der Gini-Impurity?)

Die Formel der Gini-Impurity betrachtet den Fehler, den man macht, wenn man sich zufällig – aber unter Berücksichtigung der relativen Häufigkeiten des Vorkommens der Label – für ein bestimmtes Label entscheidet. Für nebenstehendes Beispiel mit den zwei unterschiedlichen Labeln grün und rot gilt:

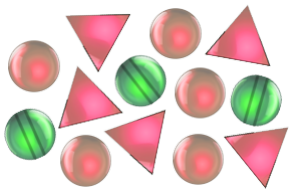


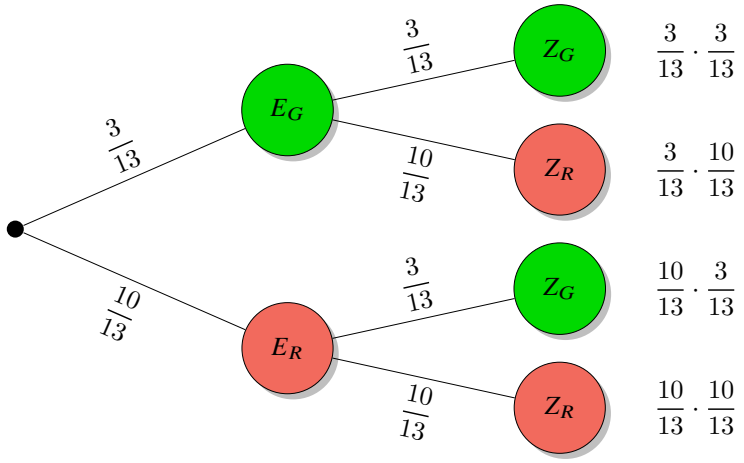
Abb. 3.8: Datenmenge für die Berechnung der Gini-Impurity.

In der Menge befinden sich drei Elemente mit grünem und zehn Elemente mit rotem Label. Man würde sich aus diesem Grund also zu $\frac{3}{13}$ für das grüne und zu $\frac{10}{13}$ für das rote Label entscheiden. Mit welcher Wahrscheinlichkeit würde man für die beiden Entscheidungen jeweils ein Element falsch klassifizieren?

- Wenn man sich für das grüne Label entscheidet, macht man in $\frac{10}{13}$ der Fälle einen Fehler bei der Klassifikation

- Wenn man sich für das rote Label entscheidet, macht man in $\frac{3}{13}$ der Fälle einen Fehler bei der Klassifikation

Diesen Prozess kann man auch gut in einem Baumdiagramm visualisieren. Die Ereignisse E_G bzw. E_R stehen dabei für die Entscheidung für das grüne bzw. rote Label, die Ereignisse Z_G bzw. Z_R dafür, dass ein Element mit grünem bzw. rotem Label ausgewählt wurde:



Der Gesamtfehler bei diesem Vorgehen ist somit

$$\frac{3}{13} \cdot \frac{10}{13} + \frac{10}{13} \cdot \frac{3}{13} = 2 \cdot \underbrace{\frac{3}{13}}_{=:p_1} \cdot \underbrace{\frac{10}{13}}_{=:p_2} = 2p_1p_2$$

Ausgehend von dem Zusammenhang $p_1 + p_2 = 1$ erhält man durch Quadrieren beider Seiten und der Anwendung der 1. binomischen Formel:

$$(p_1 + p_2)^2 = 1 \iff p_1^2 + 2p_1p_2 + p_2^2 = 1 \iff 2p_1p_2 = \underbrace{1 - p_1^2 - p_2^2}_{\text{Formel für die Gini-Impurity für zwei Label}}$$

3.1.3.1.4 Beurteilung der Split-Kriterien

In den vorangegangenen drei Abschnitten wurden drei verschiedene Maße vorgestellt, um den Informationsgehalt einer Menge angeben zu können. Doch wie wirkt sich die Wahl des Split-Kriteriums auf die Erstellung des Entscheidungsbaums aus? Zur Beantwortung dieser Frage lohnt sich ein Blick auf die Funktionsgraphen der drei Maße im Hinblick auf eine

Datenmenge mit zwei unterschiedlichen Labeln. Hierbei sticht zunächst ins Auge, dass die Entropie Werte im Intervall $[0;1]$ und die Fehlklassifikationsrate sowie die Gini-Impurity Werte im Intervall $[0;0{,}5]$ annehmen. Um die Maße noch besser vergleichen zu können, wird die Entropie mit dem Faktor 0,5 skaliert (blau-gestrichelte Linie).

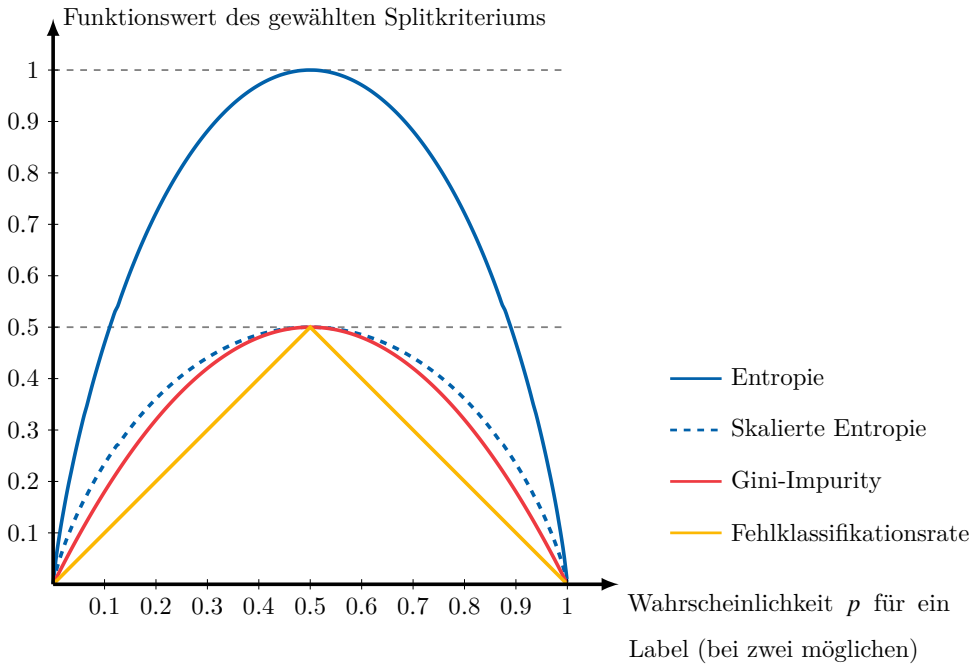


Abb. 3.9: Funktionsgraphen der Entropie, Gini-Impurity und Fehlklassifikationsrate im Hinblick auf eine Datenmenge mit zwei unterschiedlichen Labeln (in Anlehnung an (Rutkowski et al., 2020, S. 65)).

Man sieht, dass alle drei Maße den größten Wert für $p = 0{,}5$ annehmen, der dann auftritt, wenn in der Datenmenge für jedes der beiden Label dieselbe Anzahl an Daten enthalten ist, was gleichzeitig den größtmöglichen „Grad der Unordnung“ in der Menge darstellt. Die Graphen für die skalierte Entropie und Gini sind sehr ähnlich und unterscheiden sich kaum. Wirft man einen Blick auf Berechnungssimulationen, stellt man fest, dass das Gini-Kriterium hinsichtlich der Laufzeit deutlich schneller als das Entropie-Kriterium ist, da es deutlich weniger rechenintensiv ist. Dafür sind die Klassifikationsergebnisse des erhaltenen Entscheidugnsbaums bei der Wahl des Entropie-Kriteriums etwas besser, wobei fraglich ist, ob sich der Zeitaufwand in Bezug auf das erhaltene Ergebnis tatsächlich lohnt (vgl. `https://quantdare.com/decision-trees-gini-vs-entropy/`).

Während sich die Entropie und die Gini-Impurity kaum unterscheiden, gibt es doch deutliche Unterschiede zwischen den beiden Maßen und der Fehlklassifikationsrate. Während sowohl

Gini als auch die Entropie streng konkave Funktionen sind, ist die Fehlklassifikationsrate abschnittsweise linear (Rutkowski et al., 2020, S. 81). Dies hat bei manchen Berechnungen starke Auswirkungen auf die Berechnung des Informationsgewinns, wie folgendes Beispiel illustriert:

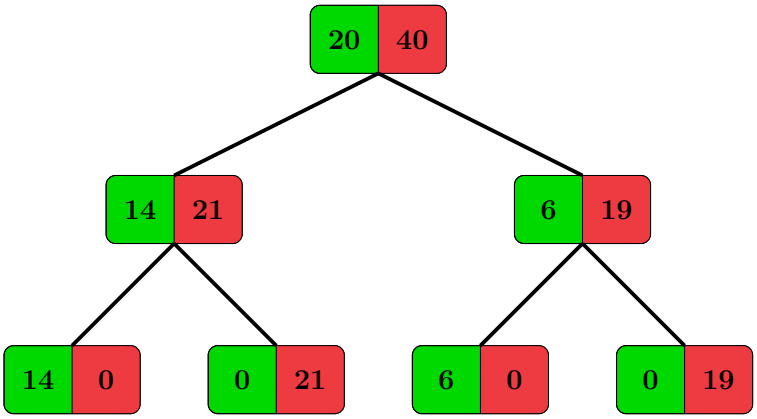


Abb. 3.10: Visualisierung der Anteile der grün bzw. rot gelabelten Daten in den jeweiligen Knoten eines Entscheidungsbaums.

Es werden wieder gelabelte Daten mit grünem und rotem Label betrachtet. In den farbigen Bereichen jedes Knotens ist jeweils die Anzahl der Daten angegeben. So befinden sich in der Ausgangsdatenmenge 20 grün und 40 rot gelabelte Trainingsdaten. Man sieht, dass sich diese Daten insgesamt perfekt aufteilen lassen und vier Blattknoten mit jeweils einheitlich gelabelten Daten entstehen. Betrachtet man nun die Berechnung des Informationsgewinns für das Aufteilen der Ausgangsmenge mithilfe der Fehlklassifikationsrate, ergibt sich:

$$\mathrm{IG}_F = \frac{20}{60} - \left(\frac{35}{60} \cdot \frac{14}{35} + \frac{25}{60} \cdot \frac{6}{25}\right) = \frac{20}{60} - \frac{20}{60} = 0$$

Das Kriterium der Fehlklassifkationsrate berechnet hier einen Informationsgewinn von 0, d. h. die Ausgangsmenge würde – je nach Definition des Algorithmus – unter Umständen nicht weiter aufgeteilt werden. Dies geschieht, obwohl insgesamt eine perfekte Separierung der Trainingsdaten hinsichtlich der betrachteten Label möglich wäre.

Verwendet man hier die Gini-Impurity, erhält man als Informationsgewinn:

$$\mathrm{IG}_G = \left(1 - \left(\frac{20}{60}\right)^2 - \left(\frac{40}{60}\right)^2\right) - \left[\frac{35}{60} \cdot \left(1 - \left(\frac{14}{35}\right)^2 - \left(\frac{21}{35}\right)^2\right) + \frac{25}{60} \cdot \left(1 - \left(\frac{6}{25}\right)^2 - \left(\frac{19}{25}\right)^2\right)\right] \approx 0{,}01$$

Das Gini-Kriterium (und analog auch das Entropie-Kriterium) ermittelt also einen (wenn auch kleinen) Informationsgewinn und die Ausgangsdatenmenge wird weiter aufgeteilt.

Woran liegt es, dass Gini oder die Entropie noch einen Informationsgewinn ermitteln können, wohingegen die Fehlklassifikationsrate keinen mehr feststellt? Hierzu betrachtet man die jeweiligen Funktionsgraphen genauer:

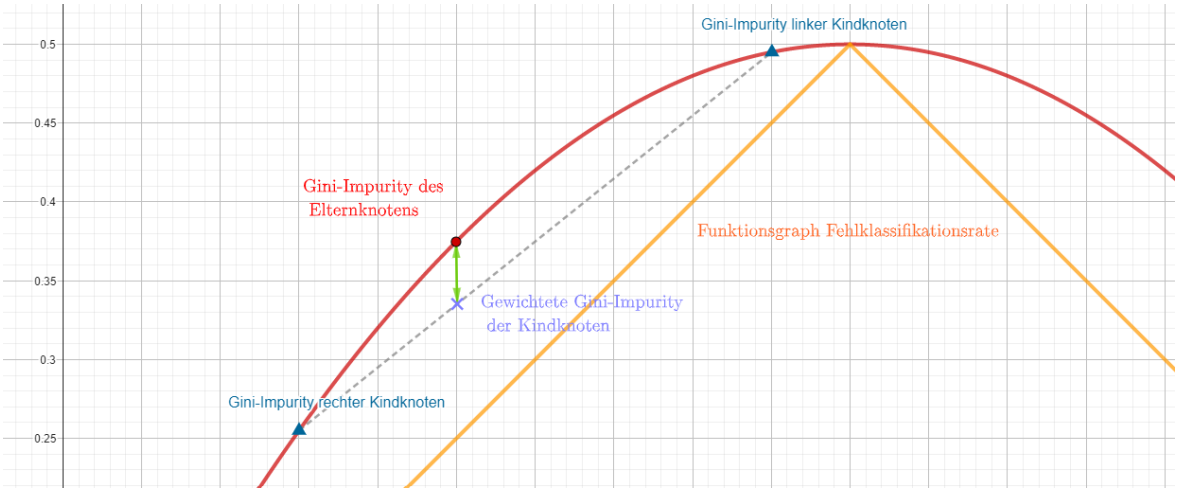


Abb. 3.11: Unterschiede der Funktionsgraphen von Gini-Impurity und Fehlklassifikationsrate und die resultierenden Auswirkungen.

Der gewichtete Durchschnitt des Informationsgehalts der möglichen Kindknoten liegt auf der Strecke zwischen den Punkten, die jeweils durch den Informationsgehalt der beiden Kindknoten festgelegt werden. Dadurch, dass bei streng konkaven Funktionen der Graph stets oberhalb jeder Verbindungsstrecke zweier seiner Punkte liegt, kann es bis auf den Sonderfall, dass die Punkte des linken und rechten Kindknotens identisch sind, nie dazu kommen, dass der Informationsgehalt des Elternknoten genauso groß ist wie der gewichtete Informationsgehalt der möglichen Kindknoten. Somit kann beim Informationsgewinn bis auf den genannten Sonderfall nie der Wert 0 auftreten. Anders sieht es hier bei der abschnittsweise linearen Funktion der Fehlklassifikationsrate aus. Hier kann es sehr wohl auftreten, dass der gewichtete Informationsgehalt der möglichen Kindknoten mit dem Punkt des Funktionsgraphen, der den Informationsgehalt des Elternknoten repräsentiert, zusammenfällt.

3.1.3.2 Abbruchkriterien

Abbruchkriterien legen Regeln fest, unter welchen Umständen ein Knoten bei der Erstellung des Entscheidungsbaums nicht weiter „in Kindknoten aufgeteilt“ und somit zu einem Blattknoten

wird. Zwei Fälle sind dabei offensichtlich: Einerseits wird das Aufteilen beendet, wenn kein Attribut mehr vorhanden ist, nach dessen Attributwerten man die Daten noch separieren könnte. Andererseits wird man das Aufteilen beenden, wenn alle Daten dasselbe Label haben, sodass ein weiteres Aufteilen keinen Informationsgewinn mehr liefern kann. In der Praxis wird es allerdings selten der Fall sein, dass sich die Datenmenge durch einen Entscheidungsbaum vollständig in homogene Teilmengen aufteilen lässt. So könnte man auch das Aufteilen eines Knotens beenden, wenn ein gewisser „Reinheitsgrad“ erreicht ist, beispielsweise, wenn 90 % der darin enthaltenen Daten einem bestimmten Label angehören. Alternativ kann man das Aufteilen der Datenmenge in einem Knoten auch beenden, wenn sich nur noch eine bestimmte Anzahl an Elementen in der Datenmenge befindet. Beide Kriterien dienen in erster Linie dazu, eine Überanpassung des Entscheidungsbaums auf die Trainingsdaten zu verhindern. Um den Entscheidungsbaum nicht zu tief werden zu lassen, kann man auch vorab eine maximale Tiefe des Baums festlegen. Je nach Wahl des Split-Kriteriums kann auch abgebrochen werden, wenn kein Informationsgewinn erfolgt (vgl. auch das Beispiel in 3.1.3.1.4).

Überblick (Abbruchkriterien)

1. Es ist kein Attribut mehr vorhanden, nach dessen Attributwerten die Datenmenge weiter aufgeteilt werden könnte.
2. Alle Elemente in der Datenmenge haben das gleiche Label.
3. Ein bestimmter Prozentsatz (z. B. 90 %) der Elemente in der Datenmenge hat das gleiche Label.
4. In der Datenmenge ist nur noch eine kleine Anzahl an Elementen vorhanden.
5. Der Baum hat bereits eine vorgegebene maximale Tiefe erreicht.
6. Es würde beim Aufteilen der Datenmenge kein Informationsgewinn erzielt werden.

3.1.3.3 Entscheidungsbaum-Algorithmus

Das Trainieren von Entscheidungsbäumen basiert auf einem verhältnismäßig einfachen rekursiven Algorithmus (in Anlehnung an Herbold (2022)):

(1) Beende den Algorithmus, wenn eines oder mehrere der in Abschnitt 3.1.3.2 genannten Abbruchkriterien eintritt.

(2) Bestimme das Attribut A, das hinsichtlich des gewählten Split-Kriteriums (s. Abschnitt 3.1.3.1) den größten Informationsgewinn liefert.

(3) Teile die Daten anhand einer gewählten Regel gemäß der Attributwerte von A auf.

(4) Wende den Algorithmus beginnend mit Schritt (1) rekursiv auf die Teilmengen an, um die nachfolgenden Teilbäume zu erstellen.

Eine schrittweise Anwendung des Algorithmus an einem konkreten Beispiel findet sich im Abschnitt Material zu den Beispielen 3.3.1 und 3.3.2.

3.1.3.4 Optimierung

Nach dem Training des Entscheidungsbaums kann versucht werden, den erhaltenen Baum anhand der Validierungsdaten zu optimieren. Je nach Wahl des Abbruchkriteriums kann eine Überanpassung des Baums hinsichtlich der Trainingsdaten vorliegen. Das bedeutet, dass der Baum zu komplex und zu tief geworden ist und somit zu sehr auf die Trainingsdaten angepasst wurde, anstatt allgemeingültige Regeln zu lernen. Um dies zu verhindern, werden verschiedene Verfahren wie Pruning oder eine Anpassung der Abbruchkriterien verwendet. Man versucht hierbei, den Baum zu verkleinern und damit die Vorhersagegenauigkeit und die Interpretierbarkeit des Baums zu verbessern. Beim Pruning werden Knoten bzw. ganze Zweige entfernt, die keine signifikante Vorhersagegenauigkeit auf den Validierungsdaten bewirken.

3.1.4 Einschätzung der Qualität des Entscheidungsbaums

Die Testphase ist ein wichtiger Bestandteil des maschinellen Lernens, da sie es ermöglicht, die allgemeingültige Vorhersagegenauigkeit (Güte) des trainierten Modells auf den Testdaten zu beurteilen.

3.1.4.1 Klassifikation der Testdaten: Genauigkeit, Sensitivität und Spezifität

In Abschnitt 3.1.2 wurde bereits die Aufteilung der verfügbaren Daten in Trainings-, ggf. Validierungs- und Testdaten dargestellt. Nachdem der Entscheidungsbaum anhand der Trainingsdaten erstellt und ggf. anhand der Validierungsdaten optimiert wurde, kann die Güte des Modells nun anhand der Testdaten eingeschätzt werden. Da die Testdaten weder zum Trainieren noch zum Optimieren des Modells herangezogen wurden, ermöglichen diese eine Einschätzung der Güte, also der allgemeingültigen Vorhersagegenauigkeit des Baums und stellen sicher, dass dieser nicht überangepasst ist. Weiter können anhand der Testdaten auch die verschiedenen Optimierungen, wie z. B. Pruning oder Anpassung der Abbruchkriterien, auf ihre Wirkung hin beurteilt werden. Man klassifiziert die Testdaten anhand des erstellten Entscheidungsbaums und vergleicht deren tatsächliches Label mit dem vorhergesagten bzw. berechneten Label. Zur Beurteilung der Qualität des Baums gibt es verschiedene Gütemaße. Das wohl gängigste Maß ist dabei der Quotient aus der Anzahl der richtig klassifizierten Testdaten und der Gesamtzahl der Testdaten, der als **Genauigkeit** bezeichnet wird. Weitere gängige Maße sind die **Sensitivität** und die **Spezifität**. Falls es nur zwei Label gibt, wird eines als positiv und eines als negativ identifiziert. Die Sensitivität beschreibt, wie gut das Modell darin ist, tatsächlich positive Fälle als solche zu erkennen. Sie wird berechnet, indem man die Anzahl der richtig erkannten positiven Fälle durch die Gesamtzahl der positiven Fälle dividiert. Ein höherer Wert der Sensitivität bedeutet, dass das Modell besser darin ist, positive Fälle zu erkennen. Dagegen gibt die Spezifität an, wie gut das Modell darin ist, tatsächlich negative Fälle zu erkennen. Sie berechnet sich analog zur Sensitivität, indem man die Anzahl der richtig klassifizierten negativen Fälle durch die Gesamtzahl der tatsächlich negativen Fälle dividiert. Je höher der Wert der Spezifität ist, umso besser erkennt das Modell tatsächlich negative Fälle als solche.

3.1.4.2 Konfusionsmatrix

Eine übersichtliche Darstellung der Klassifikationsergebnisse der Testdaten bietet die **Konfusionsmatrix** (vgl. Russel & Norvig, 2022, S. 728). Sie ist ein Instrument zur Bewertung der Leistung des erstellten Modells, indem sie die Anzahl der richtig und falsch klassifizierten Testdaten in Form einer Tabelle anzeigt. Für den Fall, dass nur zwei Label vorliegen, ergibt

sich eine Vier-Felder-Tafel. Im Folgenden wird die Klassifikation von Pilzen gemäß der Label *essbar* und *giftig* betrachtet. Es sind hierbei folgende vier Fälle möglich:

1. `Richtig positiv:` Testdatenpunkt mit Label „essbar“ wird vom Modell als essbar klassifiziert
2. `Falsch positiv:` Testdatenpunkt mit Label „giftig“ wird vom Modell als essbar klassifiziert
3. `Richtig negativ:` Testdatenpunkt mit Label „giftig“ wird vom Modell als giftig klassifiziert
4. `Falsch negativ:` Testdatenpunkt mit Label „essbar“ wird vom Modell als giftig klassifiziert

Nach dem Training des Modells wurden die Testdaten, wie in Abbildung 3.12 in der Konfusionsmatrix dargestellt, klassifiziert.

		Vorhergesagtes Label		
		essbar	giftig	Σ
Tatsächl. Label	essbar	241	43	284
	giftig	3	213	216
	Σ	244	256	500

Abb. 3.12: Darstellung der Klassifikation der Testdaten in einer Konfusionsmatrix.

Der Konfusionsmatrix kann man entnehmen, dass das trainierte Modell die Pilze sehr gut klassifiziert. Für die Genauigkeit ergibt sich dabei ein Wert von

$$\frac{241 + 213}{241 + 213 + 43 + 3} = \frac{470}{500} = 94\,\%$$

Für die Sensitivität und Spezifität ergeben sich folgende Werte:

$$\text{Sensitivität} = \frac{241}{241 + 43} \approx 84{,}9\,\% \qquad \text{Spezifität} = \frac{213}{213 + 3} \approx 98{,}6\,\%$$

Man sieht, dass der Wert der Sensitivität deutlich kleiner als der Wert der Spezifität ist und die Gesamtgenauigkeit mit 94 % sehr hoch ist. Im vorliegenden Fall könnte man den geringeren Wert der Sensitivität in Kauf nehmen, da es weniger schlimm ist, einen essbaren Pilz als giftig einzustufen, als umgekehrt. Die Werte für Sensitivität und Spezifität sind somit stark kontextabhängig.

Die Konfusionsmatrix kann durch Anfügen zusätzlicher Zeilen und Spalten auch bei mehr als zwei Labeln angewendet werden (s. Abschnitt 3.3.4).

3.1.5 Möglichkeiten und Grenzen von Entscheidungsbäumen

Der Entscheidungsbaum-Algorithmus ist ein Verfahren des überwachten maschinellen Lernens, das nach Mustern in Daten sucht und daraus ein Modell erstellt. Es eignet sich besonders gut bei kategorialen und nominalen Daten. Dabei ist zu beachten, dass es sich hierbei um ein heuristisches Vorgehen (vgl. Russel & Norvig, 2022, S. 732) handelt und es keine Garantie gibt, dass der „beste“ bzw. überhaupt „ein guter“ Entscheidungsbaum gefunden wird, was mitunter an der Greedy-Strategie bei der Auswahl des „besten“ Attributs liegt. Entscheidungsbäume sind nur dann sinnvoll anwendbar, wenn in den Daten Muster vorhanden sind. Liegen keine oder nur schwache Muster vor, bildet der Algorithmus dennoch einen Entscheidungsbaum. Aus diesem Grund kommt – wie bei allen Verfahren des maschinellen Lernens – der Testphase eine große Bedeutung zu, in der die Güte und Vorhersagequalität des erstellten Modells überprüft wird. Hyperparameter[1] wie Baumtiefe oder die prozentuale Aufteilung in Trainings- und Testdaten nehmen starken Einfluss auf den Entscheidungsbaum; kleine Änderungen im Datensatz können zu einer völlig anderen Struktur des Baums führen.

[1] Ein Hyperparameter ist ein Parameter, der vor Beginn des Lernprozesses festgelegt wird. Man kann sich Hyperparameter wie eine Art „Drehknopf“ vorstellen, dessen eingestellter Wert Auswirkungen auf das jeweils erstellte Modell hat (vgl. Russel & Norvig (2022)).

3.2 Didaktische Hinweise / Bezug zum Lehrplan

3.2.1 Einordnung in den Lehrplan

Laut LehrplanPLUS wählen die Lehrkräfte als Beispiel für einen Algorithmus maschinellen Lernens entweder den Entscheidungsbaum-Algorithmus oder den k-nächste-Nachbarn-Algorithmus aus. Im Hinblick auf den Entscheidungsbaum-Algorithmus wird die Kompetenzerwartung noch zwischen der Informatik und der spät beginnenden Informatik wie folgt unterschieden:

- **Informatik 11 (NTG)**

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

- **Spät beginnende Informatik 11 (HG, SG, MuG, SWG)**

  *Die Schülerinnen und Schüler erläutern die **Idee** eines ausgewählten Algorithmus maschinellen Lernens (k-nächste-Nachbarn-Algorithmus oder Entscheidungsbaum-Algorithmus) an konkreten Beispielen.*

Der Lehrplan fordert also ein Verständnis der allgemeinen Funktionsweise (NTG) bzw. der Idee (spät beginnend) des Entscheidungsbaum-Algorithmus anhand konkreter Beispiele. Darüber hinaus erwerben die Schülerinnen und Schüler bei der Behandlung des Entscheidungsbaum-Algorithmus zusätzlich folgende Kompetenzen:

- *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.*

- *Die Schülerinnen und Schüler nehmen zu ausgewählten aktuellen Einsatzmöglichkeiten der Künstlichen Intelligenz Stellung und bewerten Chancen und Risiken für Individuum und Gesellschaft.*

Im Einstieg in die Lehrplansequenz (Kapitel 2) haben die Schülerinnen und Schüler bereits die Grundlagen maschinellen Lernens als ein Teilgebiet der Künstlichen Intelligenz kennengelernt. Da es sich beim Entscheidungsbaum-Algorithmus um ein Verfahren des überwachten Lernens

handelt, müssen die Bereitstellung von gelabelten Daten sowie deren Aufteilung in Trainings-, ggf. Validierungs- und Testdaten (s. Abschnitt 3.1.2) im Hinblick auf die Erstellung sowie das Testen des Entscheidungsbaums thematisiert werden. Unter Verwendung einer geeigneten Software, wie z. B. Orange (`https://orangedatamining.com/`; Demšar et al. (2013)), wird den Schülerinnen und Schülern der Einfluss der Trainingsdaten und Parameter, in diesem Fall die Auswirkungen der Wahl der Abbruchkriterien (s. Abschnitt 3.1.3.2), auf den generierten Entscheidungsbaum deutlich. Anhand der Konfusionsmatrix (s. Abschnitt 3.1.4.2) können bereits an dieser Stelle Chancen und Risiken von KI-Systemen angesprochen werden. Eine zusammenfassende und ausführliche Besprechung erfolgt in Kapitel 6.

3.2.2 Durchführung

Die nachfolgenden didaktischen Hinweise beziehen sich auf die Kompetenzerwartungen des NTG. Für diesen Themenbereich werden ca. sechs Unterrichtsstunden, für die spät beginnende Informatik ca. vier Unterrichtsstunden vorgeschlagen. Im Abschnitt 3.2.2.8 finden sich Anmerkungen, an welcher Stelle sich in der spät beginnenden Informatik Unterschiede in der Herangehensweise ergeben können.

3.2.2.1 Einstieg

Als Einstieg in die Sequenz „Entscheidungsbaum-Algorithmus“ bietet es sich an, kurz zu klären, was ein Entscheidungsbaum ist bzw. was überhaupt in der Informatik unter einem „Baum“ verstanden wird. Falls in der 9. Jahrgangsstufe bei der Behandlung des Kapitels „Data Mining“ Entscheidungsbäume bereits betrachtet wurden, bietet es sich an, an dieser Stelle darauf Bezug zu nehmen. Als weitere Möglichkeit kann auch ein (wissensbasierter) Entscheidungsbaum betrachtet werden, der etwa von Experten erstellt worden ist. Anschließend wird darauf übergeleitet, dass ein Entscheidungsbaum nun selbstständig anhand von verfügbaren Daten „gelernt“ werden soll. An dieser Stelle wird der Begriff „gelabelte Daten“ eingeführt, anhand derer Entscheidungsregeln gefunden werden sollen. Diese Handreichung bietet hier zwei verschiedene Szenarien mit passenden Datensätzen an. Im ersten Szenario (s. Abschnitt 3.3.1) soll für Bewerberinnen und Bewerber entschieden werden, ob sie zum Vorstellungsgespräch

eingeladen werden. Im zweiten Szenario (s. Abschnitt 3.3.2) sollen Fische als friedlich oder feindselig erkannt werden[2].

3.2.2.2 Trainings- und Testdaten

Es bietet sich an, den Schülerinnen und Schülern zunächst den gesamten gelabelten Datensatz zu geben. Anhand diesem sollen sie für ein oder zwei neue, ungelabelte Daten entscheiden, welches Label diesen zugeordnet werden soll, also ob sie die Person zum Vorstellungsgespräch einladen würden bzw. ob der betrachtete Fisch friedlich oder feindselig ist. Der Datensatz sollte den Schülerinnen und Schülern entweder ausgedruckt in Form von kleinen Kärtchen oder digital in einem geeigneten Werkzeug wie ExcaliDraw (`https://excalidraw.com`), MiroBoard (`https://miro.com`) oder OneNote bereitgestellt werden, sodass es ihnen möglich ist, die Daten nach bestimmten Kriterien zu ordnen bzw. zu gruppieren. Dieser handlungsorientierte Ansatz bietet sich auch anschließend bei der Erarbeitung des Entscheidungsbaum-Algorithmus an. Nachdem die Schülerinnen und Schüler (vermutlich unterschiedliche) Vermutungen über das Label der ungelabelten Daten angestellt haben, kann auf die Notwendigkeit von Testdaten übergeleitet werden, anhand derer überprüft werden kann, ob gefundene Regeln Sinn ergeben oder nicht. An dieser Stelle sollte auch verdeutlicht werden, dass die Trainingsdaten nicht zum Testen verwendet werden können, da das Modell mit diesen trainiert wurde und somit weitere gelabelte Daten notwendig sind. Im Anschluss legt man eine festgelegte Anzahl an Daten beiseite, die später zum Testen des entwickelten Modells herangezogen werden; man teilt den bestehenden Datensatz also in Trainings- und Testdaten auf (s. Abschnitt 3.1.2). Da der Entscheidungsbaum-Algorithmus für die Schülerinnen und Schüler gut zu erarbeiten ist, wurde im Hinblick auf die verfügbare Unterrichtszeit bewusst entschieden, Validierungsdaten an dieser Stelle nicht explizit zu betrachten, sondern diese zu einem späteren Zeitpunkt in der Sequenz zu thematisieren.

3.2.2.3 Erarbeitung des Entscheidungsbaum-Algorithmus

Nachdem die zur Verfügung stehenden gelabelten Daten in Trainings- und Testdaten aufgeteilt wurden, wird mithilfe der Trainingsdaten der Entscheidungsbaum sukzessive erstellt. Wie

[2] Eine ähnliche Herangehensweise findet sich bei „AIUnplugged“ (`https://www.aiunplugged.org/`) und „Von Daten und Bäumen“ (`https://computingeducation.de/proj-it2school/`).

in Abschnitt 3.1.3 dargestellt wurde, ist dabei der Informationsgewinn für die Auswahl des jeweils „besten“ Attributes entscheidend. Zur Messung des Informationsgewinns stehen die in Abschnitt 3.1.3.1 dargestellten drei Split-Kriterien zur Verfügung. Aufgrund der Einfachheit bietet sich für den Unterricht die Fehlklassifikationsrate bzw. die tatsächlichen Fehlklassifikationen (Anzahl der Fehler[3]) an, auch wenn diese ungenauer misst als die Entropie und die Gini-Impurity, wie in Abschnitt 3.1.3.1.4 beschrieben. Bei der Auswahl von Beispielen muss berücksichtigt werden, dass die Fehlklassifikationsrate möglicherweise keinen Informationsgewinn misst. Gegebenenfalls muss der Algorithmus so formuliert werden, dass man die Datenmenge dennoch weiter aufteilt, auch wenn kein Informationsgewinn ermittelt wird. Um dies zu umgehen, kann der Informationsgewinn auch mit der Gini-Impurity berechnet werden; die Entropie ist für die Schülerinnen und Schüler wohl am schwersten verständlich und scheint deshalb für den Unterricht ungeeignet. An dieser Stelle ist allerdings zu berücksichtigen, dass in der Software Orange die Entropie als Maß verwendet wird und somit möglicherweise ein anderer Entscheidungsbaum ausgegeben wird als der zuvor per Hand erstellte Baum[4].

Nach Auswahl des Splitkriteriums können die Schülerinnen und Schüler z. B. unter Verwendung einer geeigneten tabellarischen Darstellung das „beste“ bzw. „wichtigste“ Attribut ermitteln und die Datenmenge gemäß seiner Attributwerte aufteilen. Für die erhaltenen Teilmengen wenden sie die Vorgehensweise rekursiv an, bis jeweils homogene Teilmengen entstehen; dies ist bei beiden Datensätzen möglich.

Nach Fertigstellung des Entscheidungsbaums bietet es sich an, die Schülerinnen und Schüler den Algorithmus in eigenen Worten formulieren zu lassen. Als Abbruchkriterium wird hier mit Sicherheit sehr oft genannt werden, dass die Teilmenge „rein“ sein muss. An dieser Stelle kann man mögliche andere Abbruchkriterien (s. Abschnitt 3.1.3.2) ansprechen, die später in Orange einstellbar sind.

3.2.2.4 Testen und Bewerten

Anhand der Testdaten kann nun die Güte des gefundenen Modells, also die Vorhersagequalität eingeschätzt werden. Als übersichtliche Darstellung der Klassifizierung der Testdaten wird an

[3] Anstelle der Anzahl der Fehler kann analog auch die Anzahl der „Treffer“, also die Zahl der richtig klassifizierten Daten, betrachtet werden.

[4] Bei den hier verwendeten Beispielen ist der Entscheidungsbaum, der von Hand mit der Fehlklassifikationsrate erstellt wird, und der von Orange mit denselben Daten erstellte Entscheidungsbaum identisch.

dieser Stelle die Konfusionsmatrix (s. Abschnitt 3.1.4.2) eingeführt, die den Schülerinnen und Schülern als Vier-Felder-Tafel aus dem Mathematikunterricht bereits bekannt ist. Als Maß für die Vorhersagequalität dient die Genauigkeit (s. Abschnitt 3.1.4.1). Je nach Anwendungsfall können die unterschiedlichen Auswirkungen der falsch positiven bzw. falsch negativen Klassifikationen angesprochen und somit auch andere Gütemaße wie z. B. die Sensitivität und Spezifität (s. Abschnitt 3.2.2.8) behandelt werden. Den Schülerinnen und Schülern sollte verdeutlicht werden, dass es sich bei der Erstellung von Entscheidungsbäumen um ein rein heuristisches Vorgehen handelt. Es gibt also keine Garantie, dass der „beste“ oder überhaupt ein „guter“ Entscheidungsbaum gefunden wird. Ein Testen des erstellten Modells ist somit unabdingbar (s. Abschnitt 3.1.5).

3.2.2.5 Einstieg in Orange

Um den Entscheidungsbaum automatisiert zu erstellen und den Einfluss von Trainingsdaten und Parametern auf die Zuverlässigkeit der Ergebnisse des Entscheidungsbaum-Algorithmus (Lehrplankompetenzerwartung) untersuchen zu können, bietet sich als geeignete Software beispielsweise Orange an. Hervorzuheben ist, dass es sich hierbei um kein didaktisches Werkzeug, sondern um eine professionelle Machine-Learning-Software handelt. Unter Berücksichtigung gewisser Aspekte, wie etwa die Bereitstellung einer kleinschrittigen Anleitung, wird Orange als geeignetes Werkzeug erachtet, um die im LehrplanPLUS genannten Inhalte zu behandeln. Analog zum Vorgehen in Abschnitt 3.2.2.3 werden dieselben Daten (s. Abschnitt 3.3.1 bzw. 3.3.2) in Orange importiert, mit denen der Entscheidungsbaum-Algorithmus manuell erarbeitet wurde. Anhand dieser wird in Orange mit den entsprechenden Widgets (s. Abschnitt 3.2.3) ein Entscheidungsbaum erstellt. Es sei an dieser Stelle nochmals betont, dass hier je nach Daten ein anderer Entscheidungsbaum erstellt wird, als dies zuvor per Hand der Fall war, da Orange mit der Entropie ein anderes Maß zur Messung des Informationsgewinns verwendet und dieses (bislang) nicht geändert werden kann. Da im kleinen Datensatz nur sehr wenige Daten vorhanden sind, müssen im Widget `Tree` die möglichen Abbruchkriterien (s. Abschnitt 3.1.3.2) zunächst abgewählt werden.

Nachdem der Entscheidungsbaum anhand der Trainingsdaten erstellt wurde, wird das gefundene Modell in Orange anhand der Testdaten überprüft und die Ergebnisse in einer Konfusionsmatrix angezeigt.

Mit Hilfe des Widgets `Data Sampler` können im Anschluss alle verfügbaren Daten nach vorgegebenen Kriterien automatisiert in Trainings- und Testdaten aufgeteilt und die Auswirkungen auf den erstellten Entscheidungsbaum betrachtet werden.

3.2.2.6 Großer Datensatz in Orange: Einfluss von Hyperparametern

Da die Auswirkungen der verschiedenen einstellbaren Parameter (Aufteilungsrate in Trainingsdaten und Testdaten sowie verschiedene Abbruchkriterien) anhand des kleinen Datensatzes nicht deutlich werden, gibt es sowohl für den Datensatz mit Bewerbungsunterlagen als auch den Datensatz mit Fischen jeweils eine Variante mit einer deutlich größeren Anzahl an gelabelten Daten, die in Orange genutzt werden können. Anders als bei den zuvor betrachteten kleinen Datensätzen ist hier eine Zerlegung in vollständig homogene Teilmengen nicht möglich. Hier erkennen die Schülerinnen und Schüler, dass durch Variation der Parameterwerte, wie beispielsweise die maximale Baumtiefe oder eine Mindestanzahl an Elementen (s. Abschnitt 3.1.3.2), der Aufbau des Entscheidungsbaums und dessen Vorhersagequalität beeinflusst wird. Ebenso wirken sich unterschiedliche Aufteilungen der gelabelten Daten in Trainings- und Testdaten auf das Ergebnis aus. An dieser Stelle kann mit den Schülerinnen und Schülern des NTG auf den Einsatz von Validierungsdaten eingegangen werden, um den erhaltenen Entscheidungsbaum ggf. noch zu optimieren.

3.2.2.7 Reflexion und Vertiefung

Am Ende dieser Sequenz sollten Einsatzmöglichkeiten und Grenzen von Entscheidungsbäumen besprochen und reflektiert werden (s. Abschnitt 3.1.5). Zur Vertiefung wenden die Schülerinnen und Schüler die in dieser Sequenz erworbenen Kompetenzen auf ein neues Szenario[5] an. Dabei erstellen sie automatisiert einen Entscheidungsbaum, beurteilen und interpretieren das Ergebnis und nehmen Optimierungen durch die Variation von Parameterwerten vor.

[5] Hierfür kann der nicht für die Erarbeitung des Entscheidungsbaum-Algorithmus genutzte Datensatz aus Abschnitt 3.3 verwendet werden.

3.2.2.8 Anmerkungen zur spät beginnenden Informatik

Um eine Idee des Entscheidungsbaum-Algorithmus zu erhalten, genügt es für die Schülerinnen und Schüler der spät beginnenden Informatik, die Auswahl der Wurzel als erstes „bestes“ Attribut nachzuvollziehen. Im Gegensatz dazu erstellen die Schülerinnen und Schüler des NTG anhand des kleinen Datensatzes den Entscheidungsbaum weitgehend selbstständig und leiten darauf aufbauend den Entscheidungsbaum-Algorithmus ab bzw. formulieren diesen. Auf die in Abschnitt 3.2.2.7 beschriebene Vertiefung kann für die Schülerinnen und Schüler der spät beginnenden Informatik verzichtet werden.

3.2.3 Orange: Erläuterung der benötigten Widgets

Dieser Abschnitt stellt alle für das Erstellen von Entscheidungsbäumen mithilfe der Software Orange benötigten Widgets kurz vor. Für die Handreichung wurde mit der Version 3.34.0 von Orange gearbeitet.

3.2.3.1 Bereitstellung der Daten

Die gelabelten Daten können am einfachsten in Form einer CSV- oder XLSX-Datei bereitgestellt werden. Hierbei umfasst die erste Zeile die Namen der Attribute. In den nachfolgenden Zeilen finden sich dann jeweils die Datenobjekte mit ihren Attributwerten. Für den Fischdatensatz schaut die XLSX-Datei etwa wie folgt aus:

	A	B	C	D	E
1	Muster	Bauchfarbe	Schuppenfarl	Flossenfarbe	Label
2	Ohne	Weiß	Orange	Gelb	feindselig
3	Ohne	Weiß	Orange	Rot	feindselig
4	Ohne	Schwarz	Blau	Gelb	friedlich
5	Ohne	Weiß	Blau	Gelb	friedlich

Abb. 3.13: Ausschnitt aus der XLSX-Datei für den Fischdatensatz.

3.2.3.2 `File` und `Data Table`

Das Widget `File` wird benötigt, um einen Datensatz, der wie in Abschnitt 3.2.3.1 beschrieben in einer CSV- oder XLSX-Datei gespeichert ist, in Orange bereitzustellen. Dazu kann das Widget `File` via Drag & Drop in den Arbeitsbereich gezogen und anschließend die gewünschte Datei ausgewählt werden. Durch einen Rechtsklick auf das Widget kann dessen Bezeichnung geändert werden.

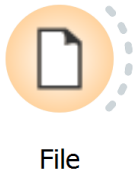


Abb. 3.14: File-Widget in Orange.

Es ist aber auch möglich, die CSV- oder XLSX-Datei direkt in den Arbeitsbereich zu ziehen; es wird dann automatisch ein Widget `File` erstellt. Wurde die Datei richtig geladen, werden unter `File` die verfügbaren Spalten angezeigt. Hier ist wichtig, dass noch die Zielvariable (*target*), also unser Label festgelegt werden muss:

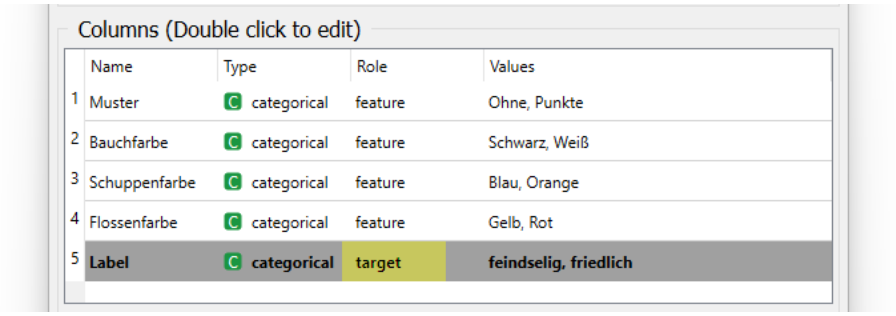
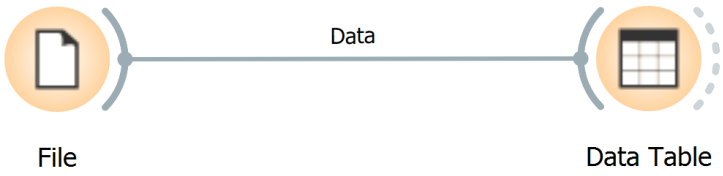


Abb. 3.15: Festlegen der Zielvariable im `File`-Widget.

Das Widget `Data Table` wird im Grunde nicht benötigt, ist jedoch dennoch nützlich, um die geladenen Daten direkt in Orange betrachten zu können. Hierfür wird das Widget `Data Table` in den Arbeitsbereich gezogen und anschließend mit einer Datenleitung mit dem `File`-Widget verbunden:



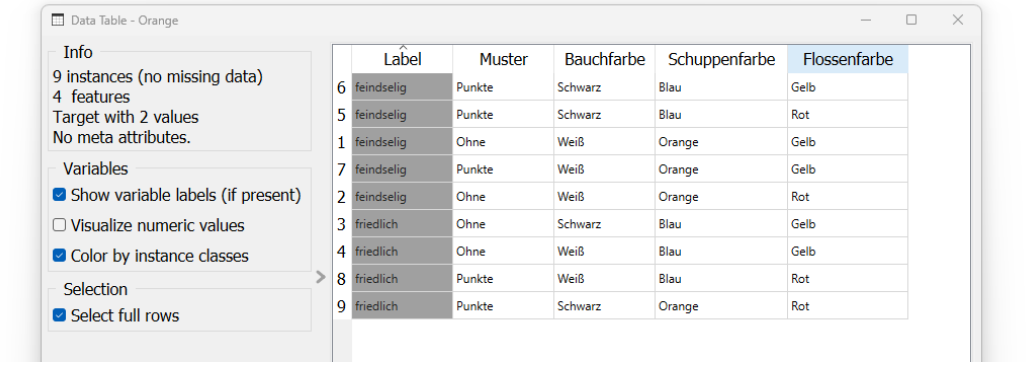
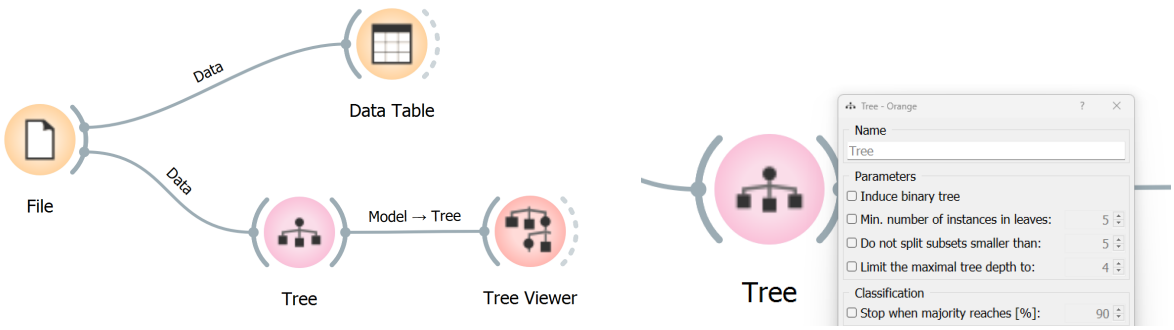


Abb. 3.16: Das Fenster des Widgets `Data Table` in `Orange`.

3.2.3.3 `Tree` und `Tree Viewer`

Um einen Entscheidungsbaum aus den geladenen gelabelten Daten (Trainingsdaten) erstellen zu können, benötigt man das Widget `Tree`, das mit einer Datenleitung mit dem `File`-Widget verbunden wird. Das Widget `Tree` erstellt zwar das Modell, es wird aber noch kein Entscheidungsbaum angezeigt; hierzu ist das Widget `Tree Viewer` notwendig, das wiederum mit einer Datenleitung mit `Tree` verbunden wird. Im Widget `Tree` können verschiedene Abbruchkriterien (s. Abschnitt 3.1.3.2) festlegt werden. Gerade bei sehr kleinen Datensätzen empfiehlt es sich, alle Optionen abzuwählen. Bei größeren Datensätzen können die Auswirkungen der unterschiedlichen Wahl der Parameter auf den erstellten Entscheidungsbaum erkundet werden. Im Widget `Tree Viewer` empfiehlt es sich, den Haken bei `Show details in non-leaves` zu entfernen. Der Baum wird dadurch übersichtlicher, enthält aber dennoch alle benötigten Informationen.



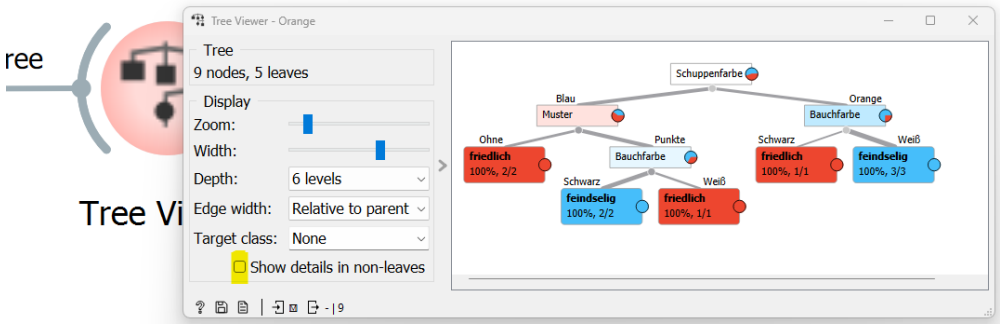


Abb. 3.17: Die Fenster der Widgets `Tree` und `Tree Viewer` in `Orange`.

3.2.3.4 `Predictions` und `Confusion Matrix`

Um das mit dem Widget `Tree` erstellte Modell testen zu können, werden Testdaten benötigt, die wieder in einem Widget `File` in Orange geladen werden. Um die beiden `File`-Widgets unterscheiden zu können, empfiehlt es sich, diese entsprechend umzubenennen:

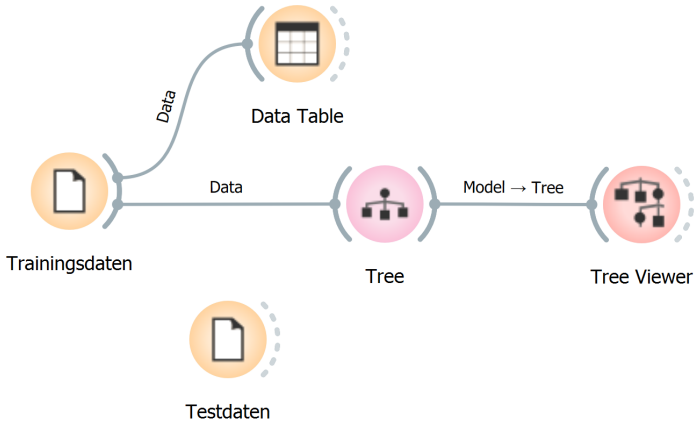


Abb. 3.18: Laden der Testdaten in `Orange`.

Die Testdaten sollen nun anhand des erstellten Modells (Widget `Tree`) klassifiziert werden und das jeweils vorhergesagte Label mit dem tatsächlichen Label verglichen werden. Hierfür gibt es das Widget `Predictions`, das als Eingaben das Modell und die Testdaten bekommt. Es zeigt für alle Testdaten an, ob diese richtig oder falsch klassifiziert wurden. Mit dem Widget `Confusion Matrix` kann die Konfusionsmatrix (s. Abschnitt 3.1.4.2) in Orange erstellt und angezeigt werden:

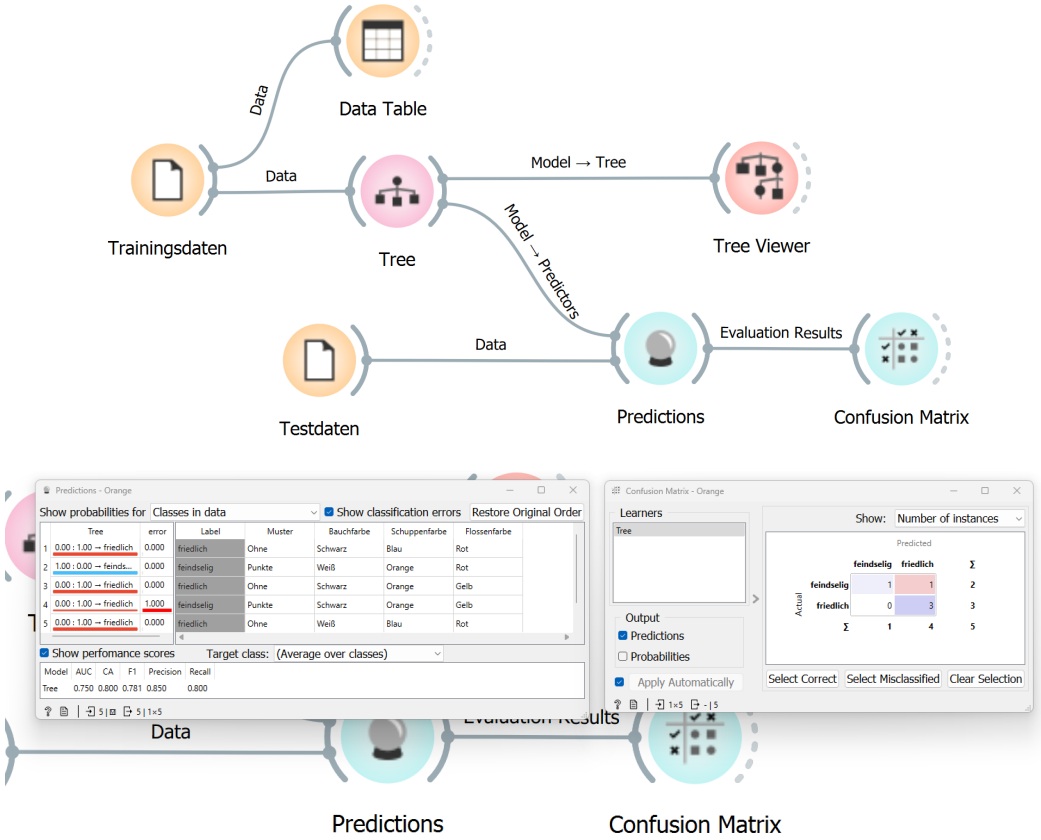


Abb. 3.19: Die Fenster der Widgets `Predicitions` und `Confusion Matrix` in `Orange`.

3.2.3.5 `Data Sampler`: Automatische Unterteilung in Trainings- und Testdaten

Orange bietet auch die Möglichkeit, eine Menge an gelabelten Daten automatisch nach anpassbaren Regeln in Trainings- und Testdaten aufzuteilen. Hierzu wird das Widget `Data Sampler` benötigt, das als Eingabe alle gelabelten Daten bekommt; in unserem obigen Beispiel werden also die Trainings- und Testdaten zu einer Datei zusammengeführt. Im `Data Sampler` kann nun konfiguriert werden, dass ein festgelegter prozentualer Anteil (z. B. 70 %) aller Daten als Trainingsdaten verwendet werden soll. Die verbliebenen Daten werden als Testdaten zum Überprüfen der Güte des erstellten Modells verwendet. Der `Data Sampler` hat dabei zwei mögliche Ausgänge: `Data Sample`, die zufällig ausgewählten Daten (z. B. hier 70 % der Daten), und `Remaining Data`, die nicht ausgewählten Daten. Ist die Option `Replicable (deterministic) sampling` nicht ausgewählt, werden die Daten nach dem eingestellten Verhältnis den Trainings- und Testdaten zufällig zugewiesen. Die beiden Ausgänge des Widgets

`Data Sampler` müssen nun in geeigneter Weise auf den Datenleitungen, die zum Widget `Tree` bzw. zum Widget `Predicitions` führen, ausgewählt werden:

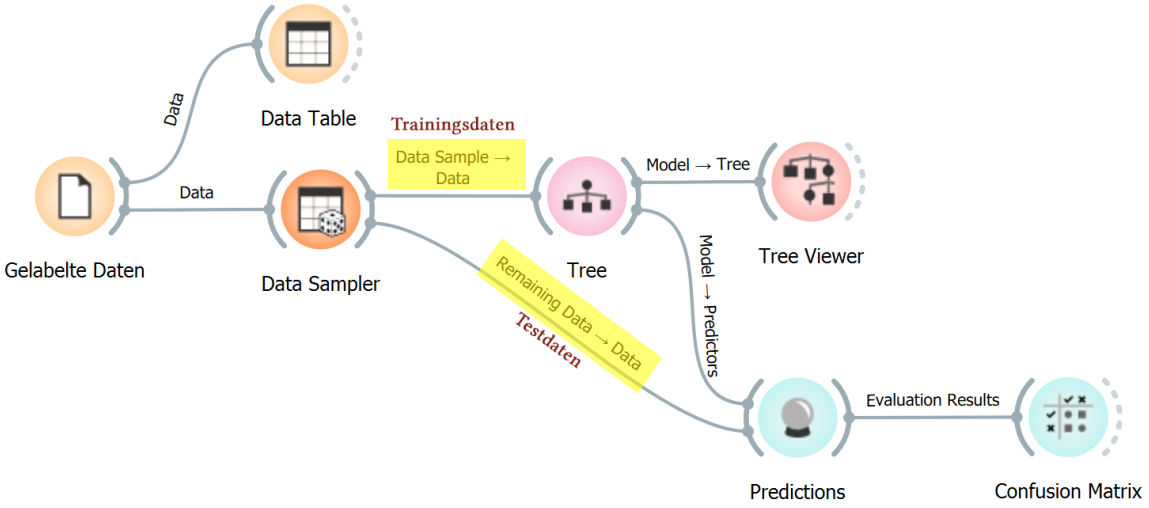


Abb. 3.20: Das Widget `Data Sampler` in `Orange`.

Im Widget `Data Sampler` kann der gewünschte prozentuale Anteil des `Data Sample` festgelegt werden. Durch Klicken auf die ausgehenden Datenleitungen kann gewählt werden, ob das `Data Sample` oder die `Remaining Data` weitergeleitet werden soll:

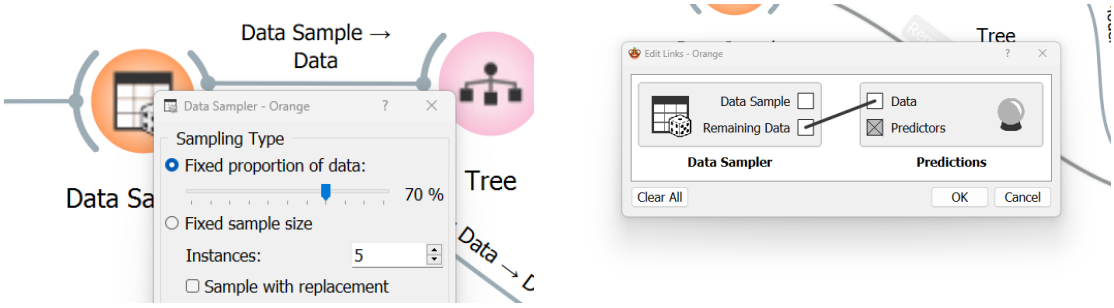
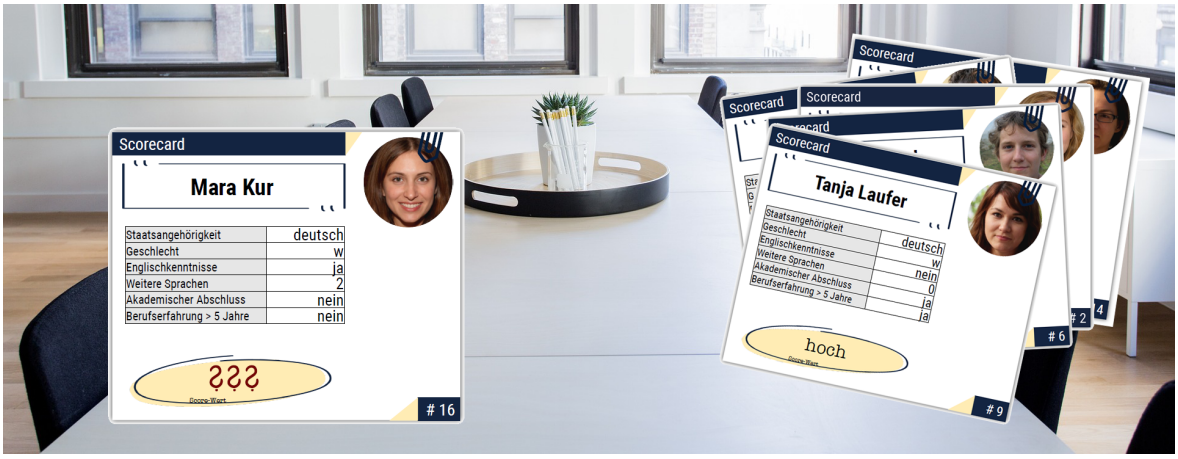


Abb. 3.21: Funktionsweise des `Data Sampler` Widgets in `Orange`.

3.3 Material

3.3.1 Bewerbungsunterlagen – Einladung zum Vorstellungsgespräch oder nicht?



Der Datensatz „Bewerbungsunterlagen“ umfasst folgende Elemente:

(a) **Kleiner Datensatz** mit 15 gelabelten Trainingsdaten, 5 gelabelten Testdaten sowie einer ungelabelten Bewerberin, für die entschieden werden soll, ob sie zum Bewerbungsgespräch eingeladen werden soll oder nicht.

- Kopiervorlage mit den Scorecards (können ausgedruckt oder digital bereitgestellt werden)
- CSV-Dateien für Trainings-, Testdaten sowie dem gesamten gelabelten Datensatz

(b) **Großer Datensatz** mit 5000 gelabelten Daten (inkl. Muster).

- CSV-Datei mit dem gesamten gelabelten Datensatz

(c) **Arbeitsheft** (10 Seiten) zur Umsetzung der Lehrplaninhalte anhand von aufeinander aufbauenden Arbeitsaufträgen.

Künstliche Intelligenz 11 **Entscheidungsbaum-Algorithmus**

Der Entscheidungsbaum-Algorithmus

Benötigtes Material (s. Kopiervorlage am Ende des Dokuments):

- 20 gelabelte Daten (15 Trainingsdaten und 5 Testdaten) in Form von Bewerbungskärtchen
- Scorecard von Mara Kur mit unbekanntem Label

Trainings- und Testdaten

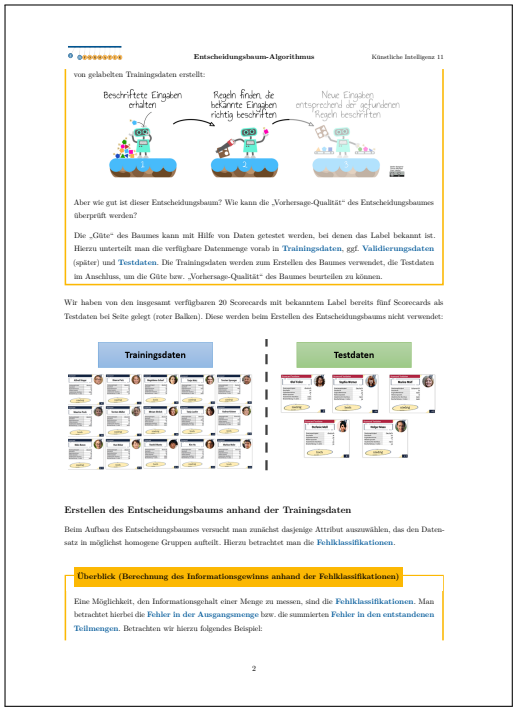
Arbeitsauftrag 1: Wie würdest du entscheiden?

Gegeben ist ein Datensatz mit 15 Mitarbeiterinnen und Mitarbettern, von denen jeweils bekannt ist, ob diese einen hohen oder niedrigen Score-Wert aufweisen. Versuche anhand dieses Datensatzes für eine neue Bewerbung (Mara Kur) zu entscheiden, ob diese zum Bewerbungsgespräch eingeladen werden soll oder nicht.

Zentrale Idee (Trainingsdaten vs. Testdaten)

Gemäß dem Verfahren des überwachten Lernens wird ein Modell (hier ein Entscheidungsbaum) ausgehend

1



Entscheidungsbaum-Algorithmus Künstliche Intelligenz 11

von gelabelten Trainingsdaten erstellt:

Beschriftete Eingaben erhalten → Regeln finden, die bekannte Eingaben richtig beschriften → Neue Eingaben entsprechend der gefundenen Regeln beschriften

Aber wie gut ist dieser Entscheidungsbaum? Wie kann die „Vorhersage-Qualität“ des Entscheidungsbaumes überprüft werden?

Die „Güte“ des Baumes kann mit Hilfe von Daten getestet werden, bei denen das Label bekannt ist. Hierzu unterteilt man die verfügbare Datenmenge vorab in **Trainingsdaten**, ggf. **Validierungsdaten** (später) und **Testdaten**. Die Trainingsdaten werden zum Erstellen des Baumes verwendet, die Testdaten im Anschluss, um die Güte bzw. „Vorhersage-Qualität“ des Baumes beurteilen zu können.

Wir haben von den insgesamt verfügbaren 20 Scorecards mit bekanntem Label bereits fünf Scorecards als Testdaten bei Seite gelegt (roter Balken). Diese werden beim Erstellen des Entscheidungsbaums nicht verwendet:

Trainingsdaten | Testdaten

Erstellen des Entscheidungsbaums anhand der Trainingsdaten

Beim Aufbau des Entscheidungsbaumes versucht man zunächst dasjenige Attribut auszuwählen, das den Datensatz in möglichst homogene Gruppen aufteilt. Hierzu betrachtet man die **Fehlklassifikationen**.

Überblick (Berechnung des Informationsgewinns anhand der Fehlklassifikationen)

Eine Möglichkeit, den Informationsgehalt einer Menge zu messen, sind die **Fehlklassifikationen**. Man betrachtet hierbei die **Fehler in der Ausgangsmenge** bzw. die summierten **Fehler in den entstandenen Teilmengen**. Betrachten wir hierzu folgendes Beispiel:

2

3.3.2 Fische – friedlich oder feindselig?

Der Datensatz „Fische“ umfasst folgende Elemente:

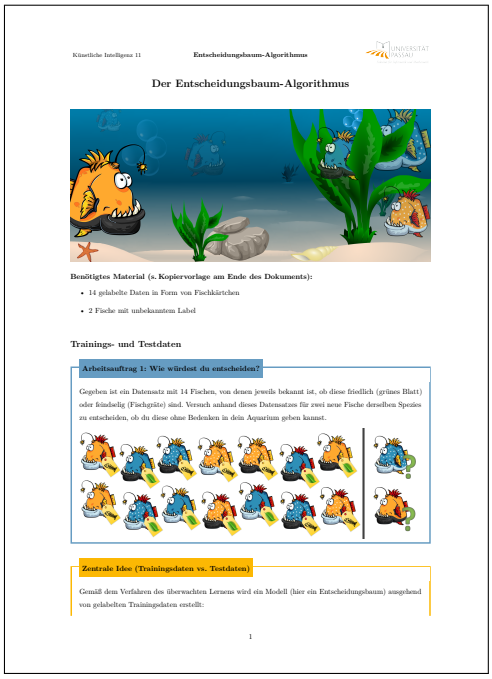
(a) **Kleiner Datensatz** mit 14 gelabelten Daten, von denen 9 als Trainingsdaten und 5 als Testdaten verwendet werden. Für zwei ungelabelte Fische soll mithilfe des Entscheidungsbaums das Label vorhergesagt werden.

- Kopiervorlage mit den Fischen (können ausgedruckt oder digital bereitgestellt werden)
- CSV-Dateien für Trainings-, Testdaten sowie dem gesamten gelabelten Datensatz

(b) **Großer Datensatz** mit 1625 gelabelten Daten (inkl. Muster).

- CSV-Datei mit dem gesamten gelabelten Datensatz

(c) **Arbeitsheft** (10 Seiten) und **Foliensatz** zur Umsetzung der Lehrplaninhalte anhand von aufeinander aufbauenden Arbeitsaufträgen.


Künstliche Intelligenz 11 Entscheidungsbaum-Algorithmus

Der Entscheidungsbaum-Algorithmus

Benötigtes Material (s. Kopiervorlage am Ende des Dokuments):
- 14 gelabelte Daten in Form von Fischkärtchen
- 2 Fische mit unbekanntem Label

Trainings- und Testdaten

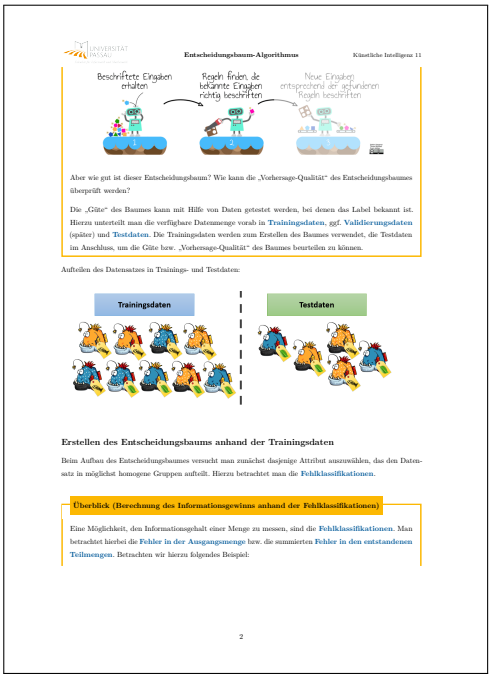
Arbeitsauftrag 1: Wie würdest du entscheiden?

Gegeben ist ein Datensatz mit 14 Fischen, von denen jeweils bekannt ist, ob diese friedlich (grünes Blatt) oder feindselig (Fischgräte) sind. Versuch anhand dieses Datensatzes für zwei neue Fische derselben Spezies zu entscheiden, ob du diese ohne Bedenken in dein Aquarium geben kannst.

Zentrale Idee (Trainingsdaten vs. Testdaten)

Gemäß dem Verfahren des überwachten Lernens wird ein Modell (hier ein Entscheidungsbaum) ausgehend von gelabelten Trainingsdaten erstellt:

1


Entscheidungsbaum-Algorithmus Künstliche Intelligenz 11

Beschriftete Eingaben erhalten

Regeln finden, die bekannte Eingaben richtig beschriften

Neue Eingabe entsprechend der erlernten Regeln beschriften

Aber wie gut ist dieser Entscheidungsbaum? Wie kann die „Vorhersage-Qualität" des Entscheidungsbaumes überprüft werden?

Die „Güte" des Baumes kann mit Hilfe von Daten getestet werden, bei denen das Label bekannt ist. Hierzu unterteilt man die verfügbare Datenmenge vorab in Trainingsdaten, ggf. Validierungsdaten (später) und Testdaten. Die Trainingsdaten werden zum Erstellen des Baumes verwendet, die Testdaten im Anschluss, um die Güte bzw. „Vorhersage-Qualität" des Baumes beurteilen zu können.

Aufteilen des Datensatzes in Trainings- und Testdaten:

Trainingsdaten

Testdaten

Erstellen des Entscheidungsbaums anhand der Trainingsdaten

Beim Aufbau des Entscheidungsbaumes versucht man zunächst dasjenige Attribut auszuwählen, das den Datensatz in möglichst homogene Gruppen aufteilt. Hierzu betrachtet man die Fehlklassifikationen.

Überblick (Berechnung des Informationsgewinns anhand der Fehlklassifikationen)

Eine Möglichkeit, den Informationsgehalt einer Menge zu messen, sind die Fehlklassifikationen. Man betrachtet hierbei die Fehler in der Ausgangsmenge bzw. die summierten Fehler in den entstandenen Teilmengen. Betrachten wir hierzu folgendes Beispiel:

2

3.3.3 Entscheidungsbaum-Simulator

An der Didaktik der Informatik der Universität Passau wurde der Entscheidungsbaum-Simulator (s. Abbildung 3.22) entwickelt, der sich ebenfalls eignet, Entscheidungsbäume automatisiert erstellen und testen zu können.

Im Gegensatz zu Orange handelt es sich hierbei um eine abgegrenzte Umgebung, die genau das an Funktionalität bietet, was in Bezug auf die im LehrplanPLUS formulierten Kompetenzen hinsichtlich des Entscheidungsbaum-Algorithmus relevant ist. Weiter kann im

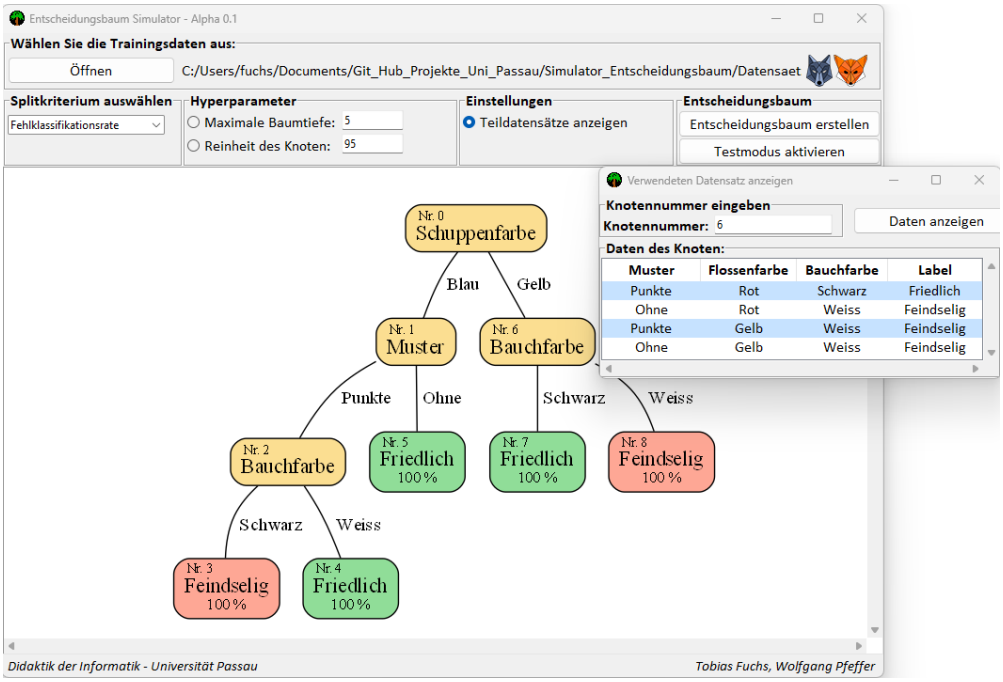


Abb. 3.22: Grafische Oberfläche des Entscheidungsbaum-Simulators.

Entscheidungsbaum-Simulator im Hinblick auf das Split-Kriterium aus den drei Möglichkeiten Fehlklassifikationsrate, Gini-Impurity und Entropie gewählt werden. Eine Festlegung von verschiedenen Hyperparametern (wie etwa die maximale Baumtiefe oder Reinheit der Knoten) ist ebenso möglich wie das Anzeigen der Teildatensätze in den jeweiligen Knoten des Baums. Die aktuelle Version des Entscheidungsbaum-Simulators samt Benutzerhandbuch ist im zur Handreichung gehörenden Materialordner verfügbar.

3.3.4 Aufgabenbeispiel für Leistungserhebungen

Ein Streamingdienstanbieter möchte zielgenau für unterschiedliche Zielgruppen Werbung schalten. Um den potentiellen Kunden das passende Abomodell anbieten zu können, muss er eine Aussage treffen, welches Angebot am besten zu ihnen passt. Dazu erstellt er auf Basis seiner bisherigen Kundendaten ein Modell, mit dem er mit hoher Wahrscheinlichkeit vorhersagen kann, welche Art von Abo zukünftige Kunden abschließen werden. Der Datensatz besitzt folgende Merkmale bzw. Merkmalsausprägungen:

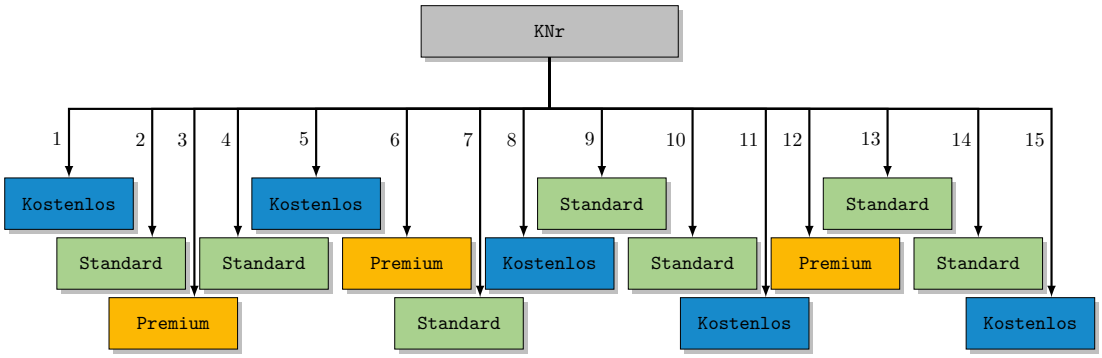
- `KNr`: Ganzzahlige Kundennummer, die fortlaufend durchnummeriert wird und somit eindeutig ist.

- `Alter`: Gibt den Altersbereich des Kunden bzw. der Kundin an (Alt / Jung).
- `Anderer Dienst`: Gibt an, ob bereits ein anderer Dienst genutzt wird (Ja / Nein).
- `Vorliebe`: Gibt an, welchen Bereich des Angebots der Kunde bzw. die Kundin am häufigsten nutzt (Filme / Serien / Sport).

Als Label ist die jeweilige Aboart (*Kostenlos*, *Standard*, *Premium*) gegeben. Im Folgenden wird ein Ausschnitt aus den verfügbaren Kundendaten betrachtet, die als Trainingsdaten für das Modell genutzt werden sollen:

KNr	Alter	Anderer Dienst	Vorliebe	Aboart
1	Jung	Ja	Serien	Kostenlos
2	Alt	Nein	Sport	Standard
3	Jung	Nein	Filme	Premium
4	Jung	Nein	Sport	Standard
5	Jung	Ja	Sport	Kostenlos
6	Alt	Nein	Serien	Premium
7	Jung	Nein	Sport	Standard
8	Alt	Ja	Sport	Kostenlos
9	Alt	Nein	Sport	Standard
10	Jung	Nein	Serien	Standard
11	Alt	Ja	Filme	Kostenlos
12	Alt	Nein	Filme	Premium
13	Jung	Nein	Serien	Standard
14	Alt	Nein	Sport	Standard
15	Alt	Ja	Serien	Kostenlos

1. Der Azubi der IT-Abteilung erstellt anhand der gegebenen Trainingsdaten folgenden Entscheidungsbaum:



Erklären Sie, weshalb der obige Entscheidungsbaum kein brauchbares Modell darstellt. Begründen Sie zudem, warum das Merkmal `KNr` generell nicht zur Erstellung eines Entscheidungsbaums herangezogen werden sollte.

Lösungsvorschlag:

- Für die Aufteilung der Datenobjekte gemäß ihren Attributwerten wurde ein Attribut verwendet, das für jedes Datenobjekt einen eindeutigen Wert hat (sehr starkes *Overfitting*). Somit ist in jeder entstandenen Teilmenge nur noch ein einziges Datenobjekt enthalten.
- Gemäß dem erstellten Modell können keine neuen Datenobjekte, die ja Kundennummern enthalten, die noch nicht vergeben wurden, klassifiziert werden.
- Ziel ist es, Entscheidungsregeln zu finden, sodass die Datenobjekte möglichst passgenau in Teilmengen Abo Premium, Abo Standard und Abo Kostenlos eingeteilt werden.

2. Formulieren Sie den Entscheidungsbaum-Algorithmus in Pseudocode.

Lösungsvorschlag:

s. Abschnitt 3.1.3.3.

3. Der Entscheidungsbaum-Algorithmus erstellt anhand der gegebenen Trainingsdaten folgenden Entscheidungsbaum:

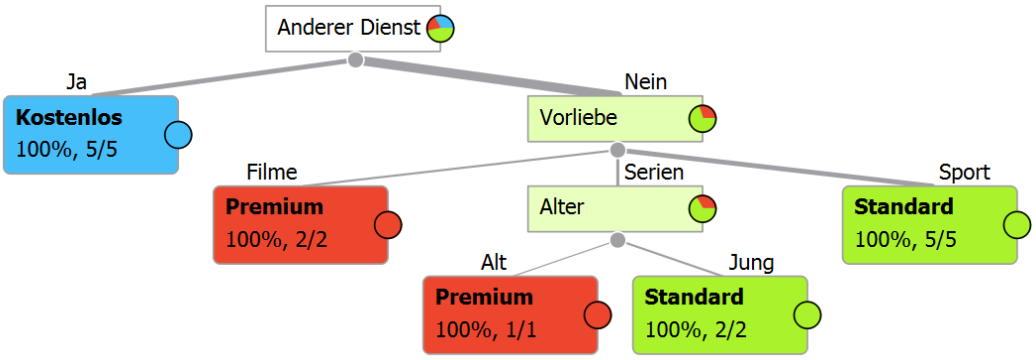


Abb. 3.23: Anhand der Trainingsdaten erstellter Entscheidungsbaum.

Begründen Sie unter Betrachtung des Informationsgewinns, weshalb im rechten Teilbaum zunächst das Attribut `Vorliebe` und nicht das Attribut `Alter` ausgewählt wurde.

Lösungsvorschlag:

Man betrachtet für die beiden Attribute jeweils den Informationsgewinn. Vor dem Aufteilen nach einem weiteren Attribut besitzen sieben der zehn Daten das Label `Standard` und drei Daten das Label `Premium`. Würde man nicht weiter aufteilen, ergeben sich drei Fehler in den Klassifikationen. Teilt man die Daten nach dem Attribut `Vorliebe` auf, ergibt sich nur noch ein Fehler in den Klassifikationen, teilt man nach dem Attribut `Alter` auf, ergeben sich drei Fehler in den Klassifikationen (s. nachfolgende Tabellen). Somit wird als „bestes“ Attribut für den rechten Teilbaum zunächst das Attribut `Vorliebe` ausgewählt.

Attribut `Vorliebe`			
	Standard	Premium	Fehler
Serien	2	1	1
Sport	5	0	0
Filme	0	2	0
	Summe Fehler:		1

Attribut `Alter`			
	Standard	Premium	Fehler
Alt	4	1	1
Jung	3	2	2
	Summe Fehler:		3

4. Ermitteln Sie, welches Abo der folgende Neukunde gemäß dem Modell voraussichtlich abschließen wird:

Name	Alter	Anderer Dienst	Vorliebe
Kilian Turing	Alt	Nein	Serien

Lösungsvorschlag:

Gemäß dem Modell wird der Kunde voraussichtlich ein Premium-Abo abschließen.

5. Das in Aufgabe 3 gegebene Modell wurde zur Klassifizierung von Testdaten herangezogen. Für den Streamingdienstanbieter ist zunächst nur von Interesse, ob das erstellte Modell zuverlässig ermitteln kann, ob ein Neukunde voraussichtlich ein bezahltes oder unbezahltes Abo abschließend wird, d. h. die Label Standard und Premium werden zu einem Label Kostenpflichtig zusammengefasst. Das Ergebnis der Klassifizierung der Testdaten ist in nachfolgender Konfusionsmatrix zu sehen:

		Vorhergesagtes Label		
		Kostenlos	Kostenpflichtig	Σ
Tatsächl. Label	Kostenpflichtig	93	85	178
	Kostenpflichtig	89	81	170
	Σ	182	166	348

Bewerten Sie die Konfusionsmatrix und nennen Sie mögliche Gründe, die zu dem vorliegenden Ergebnis geführt haben könnten.

Lösungsvorschlag:

- Betrachtet man das Qualitätsmaß „Genauigkeit“, sieht man, dass lediglich 50 Prozent der Testdaten korrekt klassifiziert wurden und somit in etwa genauso viele wie sich bei einer zufälligen Klassifizierung ($33{,}\overline{3}\,\%$) ergeben würden.
- Das Modell scheint aufgrund der Testdaten also nur eine sehr geringe Vorhersagegenauigkeit zu besitzen.

- Mögliche Gründe für das vorliegende Ergebnis: Zu kleine Anzahl an Trainingsdaten, Trainingsdaten falsch gewählt, generell kein Muster im Verhalten der Kunden vorhanden etc.

Alternative Aufgabenstellung:

Falls in der Unterrichtssequenz zu Entscheidungsbäumen auch Konfusionsmatrizen behandelt wurden, die sich nicht auf die Betrachtung von zwei Label beschränken, wäre alternativ auch folgende Aufgabenstellung denkbar:

Das ermittelte Modell wurde zur Klassifizierung von Testdaten herangezogen. Das Ergebnis der Klassifizierung der Testdaten ist in nachfolgender Konfusionsmatrix zu sehen:

		Vorhergesagtes Label			
		Kostenlos	Standard	Premium	Σ
Tatsächl. Label	Kostenlos	82	61	57	200
	Standard	63	67	51	181
	Premium	44	51	69	164
	Σ	189	179	177	545

Bewerten Sie die Konfusionsmatrix und nennen Sie mögliche Gründe, die zu dem vorliegenden Ergebnis geführt haben könnten.

Lösungsvorschlag:

- Betrachtet man das Qualitätsmaß „Genauigkeit“, sieht man, dass lediglich 40 Prozent der Testdaten korrekt klassifiziert wurden und somit nur geringfügig mehr als sich voraussichtlich bei einer zufälligen Klassifizierung ergeben würden.
- Das Modell scheint aufgrund der Testdaten also nur eine sehr geringe Vorhersagegenauigkeit zu besitzen.

- Mögliche Gründe für das vorliegende Ergebnis: Zu kleine Anzahl an Trainingsdaten, Trainingsdaten falsch gewählt, generell kein Muster im Verhalten der Kunden vorhanden etc.
- Es wäre auch eine Argumentation denkbar, die die Unterscheidung der Fehlklassifikationen mit einbezieht, also dass beispielsweise bei tatsächlichem Label Kostenlos das vorhergesagte Label Standard besser ist als das vorhergesagte Label Premium.

KAPITEL 4 k-nächste-Nachbarn-Algorithmus

„ The intuition underlying Nearest Neighbour Classification is quite straightforward, examples are classified based on the class of their nearest neighbours.“
Cunningham & Delany (2007)

Überblick:

4.1 Fachliche Grundlagen

4.1.1 Überblick

Der k-nächste-Nachbarn-Algorithmus ist ein Verfahren des überwachten maschinellen Lernens, das sowohl zur Klassifizierung als auch zur Regression verwendet werden kann. Zunächst wird die Verwendung des k-nächste-Nachbarn-Algorithmus zur Klassifizierung betrachtet. Dazu benötigt der Algorithmus gelabelte, also bereits klassifizierte Trainingsdaten. Zur Klassifizierung eines neuen Datenpunkts P zieht man eine gegebene Anzahl k von Trainingsdatenpunkten heran, die ihm in Bezug auf seine Merkmalsausprägungen „am ähnlichsten“ sind. Man ordnet ihm diejenige Klasse zu, die unter diesen k Trainingsdatenpunkten am häufigsten vorkommt (s. Abschnitt 4.1.2). Um die „Ähnlichkeit“ bestimmen zu können, benötigt man Abstandsmaße (s. Abschnitt 4.1.3). Dabei müssen die verwendeten Daten häufig zunächst aufbereitet werden (s. Abschnitt 4.1.4). In Abschnitt 4.1.5 wird der Lernprozess des k-nächste-Nachbarn-Algorithmus insgesamt betrachtet. Die Möglichkeiten und Grenzen der Anwendung werden in Abschnitt 4.1.6 thematisiert. In Abschnitt 4.1.7 erfolgt ein Exkurs zur Verwendung des k-nächste-Nachbarn-Algorithmus zur Regression.

4.1.2 Grundidee

Zur Veranschaulichung der Grundidee des k-nächste-Nachbarn-Algorithmus werden zunächst Daten mit zwei Merkmalen betrachtet.

Im folgenden Beispiel stehen als Trainingsdaten Texte zur Verfügung, die bereits als englische (Klasse 1) bzw. deutsche (Klasse 2) Texte gelabelt wurden. Nun soll mithilfe des k-nächste-Nachbarn-Algorithmus ein neuer Text P anhand der beiden Merkmale *durchschnittliche Wortlänge* und *relative Vokalhäufigkeit* klassifiziert werden. Da hier lediglich zwei Merkmale verwendet werden, lässt sich ein Datenpunkt mit den Merkmalen p_1 und p_2 als Punkt $(p_1 \mid p_2) \in \mathbb{R}^2$ in einem zweidimensionalen Koordinatensystem interpretieren (s. Abbildung 4.1).

Anschaulich haben diejenigen Trainingsdatenpunkte, die P in Bezug auf die Merkmalsausprägungen „am ähnlichsten“ sind, den geringsten Abstand zu P. Diese bezeichnet man als seine „nächsten Nachbarn“.

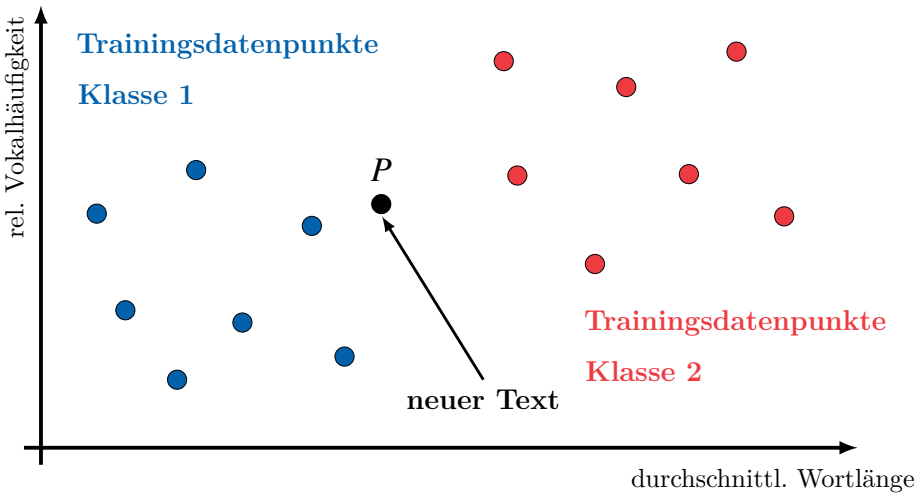


Abb. 4.1: Zweidimensionale Darstellung der Datenpunkte.

Um die nächsten Nachbarn von P ermitteln zu können, muss zunächst für alle Trainingsdatenpunkte der jeweilige Abstand zu P bestimmt werden (s. Abbildung 4.2).

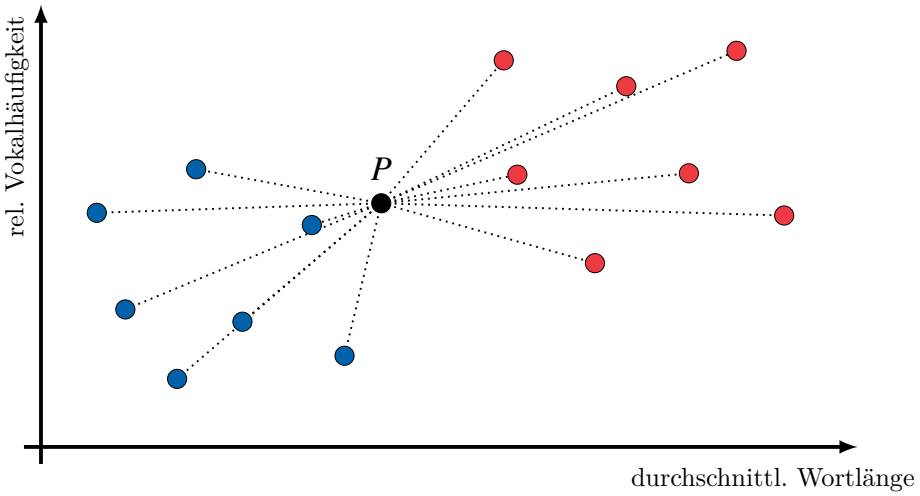


Abb. 4.2: Bestimmung der Abstände der Trainingsdatenpunkte zu P.

Aus allen Trainingsdatenpunkten wird nun eine bestimmte Anzahl k ausgewählt, die P am nächsten liegen. Diese werden „k nächste Nachbarn“ genannt. Dieses k ist ein Hyperparameter des Algorithmus, der im Vorfeld festgelegt werden muss. Anschließend wird P derjenigen Klasse zugeordnet, die die meisten Trainingsdatenpunkte unter seinen k nächsten Nachbarn hat. Ist keine eindeutige Entscheidung möglich, muss das Klassifikationsergebnis anhand weiterer Kriterien bestimmt werden.

Für $k = 1$ wird P die Klasse des nächsten Trainingsdatenpunkts, also die Klasse 1 (Englisch), zugeordnet (s. Abbildung 4.3).

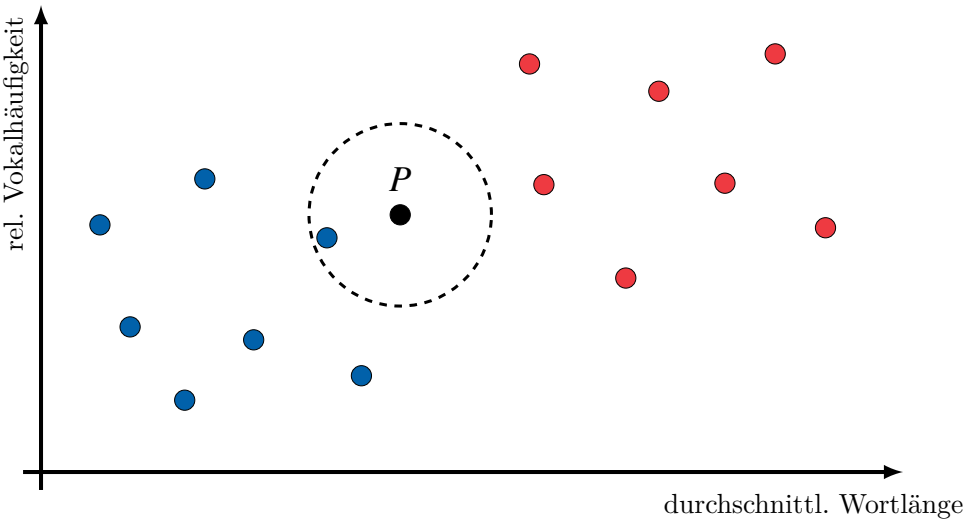


Abb. 4.3: Bestimmung der Klasse des neuen Datenpunktes anhand der Klasse des nächsten Trainingsdatenpunktes.

Für $k = 3$ wird P die Klasse der Mehrheit der drei nächsten Nachbarn zugeordnet. Da zwei Datenpunkte zur Klasse 2 und lediglich einer zur Klasse 1 gehören, wird P die Klasse 2 (Englisch) zugeordnet (s. Abbildung 4.4).

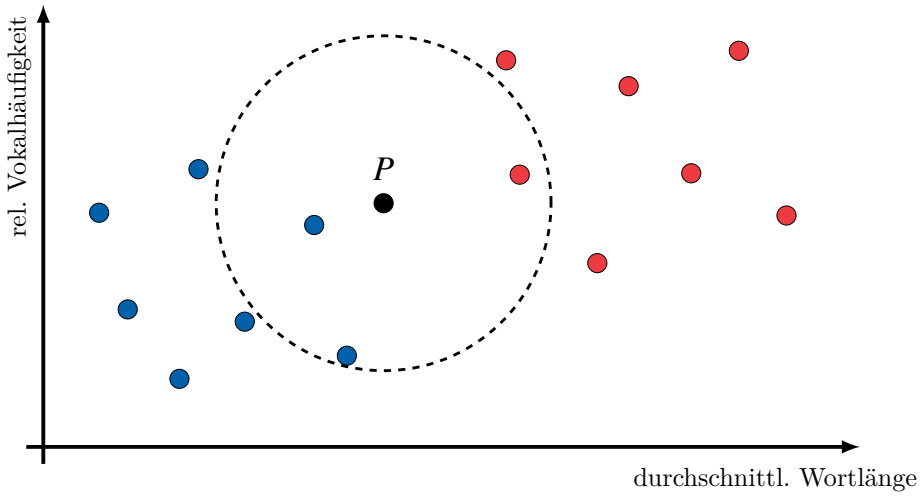


Abb. 4.4: Bestimmung der Klasse des neuen Datenpunktes anhand der Klassen der drei nächsten Trainingsdatenpunkten.

4.1.3 Abstandsmaße

Der Abstand der Trainingsdatenpunkte zu dem zu klassifizierenden Datenpunkt P wird durch die jeweiligen Ausprägungen der verwendeten Merkmale bestimmt. Für die Berechnung der Distanz stehen unterschiedliche Abstandsmaße zur Verfügung.

4.1.3.1 Euklidische Distanz

Das gängige Abstandsmaß, das die Schülerinnen und Schüler aus dem Mathematikunterricht kennen, ist die Euklidische Distanz.
Für zwei Datenpunkte $A(a_1 \mid a_2 \mid ... \mid a_n), B(b_1 \mid b_2 \mid ... \mid b_n) \in \mathbb{R}^n$ mit jeweils n verschiedenen Merkmalsausprägungen wird der Abstand $d(A,B)$ folgendermaßen berechnet:

$$d = d(A,B) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

Für zwei Merkmale kann dieser grafisch durch die Länge der Strecke d im Koordinatensystem veranschaulicht werden (s. Abbildung 4.5).

Nach dem Satz des Pythagoras ergibt sich:

$$d(A,B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

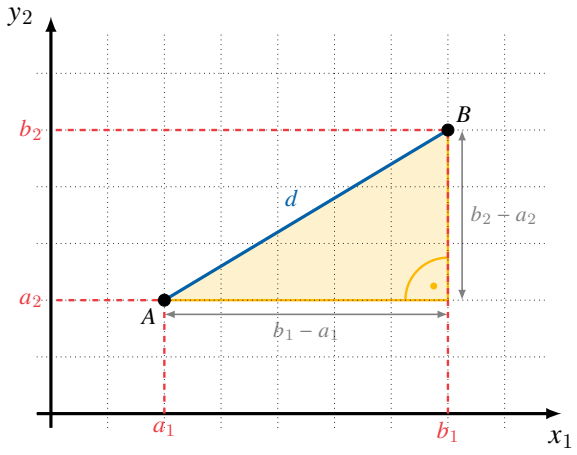


Abb. 4.5: Euklidische Distanz im $\mathbb{R}^2$.

4.1.3.2 Manhattan-Distanz

Die Manhattan-Distanz bezieht ihren Namen aus dem Schachbrettmuster des Straßennetzes von Manhattan. Will beispielsweise ein Taxifahrer von Punkt A nach Punkt B fahren, kann er nicht die direkte Verbindungsstrecke wählen, die der euklidischen Distanz entspricht, sondern muss dem Straßenmuster folgen und damit den Manhattan-Abstand zurücklegen. Rechnerisch bestimmt man die Manhattan-Distanz zweier Datenpunkte $A,B \in \mathbb{R}^n$, indem man die Absolutbeträge der Differenzen der jeweiligen Merkmalsausprägungen von A und B aus

jeder Merkmalsdimension addiert:

$$d(A,B) = \sum_{i=1}^{n} |a_i - b_i|$$

Für zwei Merkmale kann die Berechnung des Manhattan-Abstands wiederum grafisch, wie in Abbildung 4.6, veranschaulicht werden.

$$d(A,B) = |a_1 - b_1| + |a_2 - b_2|$$

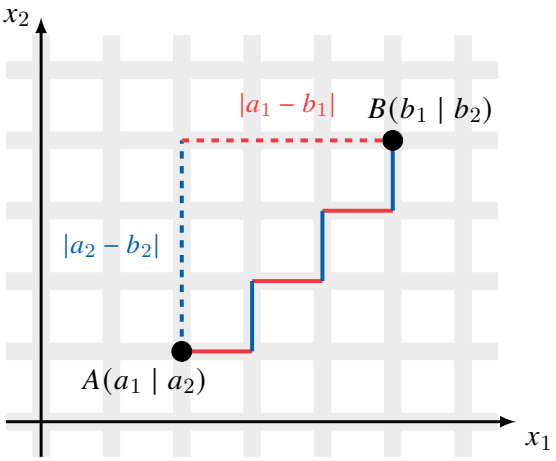


Abb. 4.6: Manhattan Distanz im $\mathbb{R}^2$.

4.1.3.3 Minkowski-Distanz

Die Minkowski-Distanz ist eine Verallgemeinerung der Euklidischen Distanz ($p = 2$) und der Manhattan-Distanz ($p = 1$). Für zwei Datenpunkte $A,B \in \mathbb{R}^n$ und $p \in \mathbb{N}$ wird der Abstand $d(A,B)$ folgendermaßen berechnet:

$$d(A,B) = \sqrt[p]{\sum_{i=1}^{n} |a_i - b_i|^p} = \left(\sum_{i=1}^{n} |a_i - b_i|^p\right)^{\frac{1}{p}}$$

4.1.3.4 Hamming-Distanz

Die Hamming-Distanz eignet sich für Abstandsangaben für Punkte mit nicht-numerischen Merkmalen, wie z. B. Wahrheitswerte oder Zeichenketten. Der Abstand $d(A,B)$ wird in solchen Fällen wie folgt berechnet:

$$d(A,B) = \sum_{i=1}^{n} d(a_i,b_i)$$

mit

$$d(a_i,b_i) = \begin{cases} 1 & \text{falls } a_i \neq b_i \\ 0 & \text{falls } a_i = b_i \end{cases}$$

Beispiele:

1. Abstand zwischen zwei vierdimensionalen Datenpunkten mit boolschen Koordinaten:

	Datenpunkt A	**Datenpunkt B**	
	a_i	b_i	$d(a_i,b_i)$
$i = 1$	`true`	`true`	0
$i = 2$	`false`	`true`	1
$i = 3$	`true`	`false`	1
$i = 4$	`false`	`false`	0

$$d(A,B) = 0 + 1 + 1 + 0 = 2$$

2. Abstand zwischen zwei vierdimensionalen Wörtern A und B als Maß für die Übereinstimmung:

	Datenpunkt A	**Datenpunkt B**	
	a_i	b_i	$d(a_i,b_i)$
$i = 1$	`H`	`H`	0
$i = 2$	`A`	`A`	0
$i = 3$	`U`	`N`	1
$i = 4$	`S`	`S`	0

$$d(A,B) = 0 + 0 + 1 + 0 = 1$$

4.1.4 Normalisierung bzw. Standardisierung von numerischen Daten

Die Berechnung des Abstands zweier Datenpunkte $A,B \in \mathbb{R}^n$ ergibt sich, unabhängig von der Wahl des Abstandsmaßes, aus der Summe der Teilabstände, die sich aus den Unterschieden der einzelnen Merkmalsausprägungen a_i und b_i ergeben. Dies führt je nach Anwendungsszenario dazu, dass Größen mit unterschiedlichen Einheiten, z. B. *durchschnittliche Wortlänge* und *relative Vokalhäufigkeit* (s. Abbildung 4.2), addiert werden müssen – „man addiert Äpfel und Birnen“.

Beispiel (Berechnung von Abständen bei unterschiedlichen Größen)

Anhand der Merkmale *Körpergewicht*, *Körpergröße* und *Körperfettanteil* soll klassifiziert werden, ob eine Person männlich oder weiblich ist. Dazu wird der Abstand der Daten-

punkte X und Y berechnet. Für die Abstandsberechnung wird die Manhattan-Distanz angewendet.

	Datenpunkt A	**Datenpunkt** B	
Merkmal	a_i	b_i	$\lvert a_i - b_i \rvert$
Körpergewicht	75 kg	62 kg	13 kg
Körpergröße	185 cm	171 cm	14 cm
Körperfettanteil	0,13	0,32	0,19

$$d(A,B) = 13\,\text{kg} + 14\,\text{cm} + 0{,}19 = ?$$

Die Skalierung der einzelnen Merkmale hat einen entscheidenden Einfluss auf die Abstandsberechnung. Werden zur Klassifizierung Merkmale herangezogen, deren Ausprägungsbereiche sich erheblich voneinander unterscheiden, beeinflussen sich Distanzen der Datenpunkte unterschiedlich und können somit das Klassifizierungsergebnis beeinträchtigen.

Beispiel (Einfluss von Größen bei der Abstandsberechnung)

Zur Klassifizierung von Sprachen werden die Merkmale *relative Vokalhäufigkeit* (m_1) und *durchschnittliche Wortlänge* (m_2) verwendet (s. Abschnitt 4.1.2). Während die Werte für m_1 zwischen 0 und 1 liegen, können die Werte für m_2 theoretisch beliebig groß werden. Betrachtet man den Abstand zweier Datenpunkte (Texte) A und B unter Verwendung der euklidischen Distanz, ergibt sich:

	Datenpunkt A	**Datenpunkt** B	
Merkmal	a_i	b_i	$(a_i - b_i)^2$
m_1	0,34	0,38	$0{,}04^2 = 0{,}0016$
m_2	6,21	4,98	$1{,}23^2 = 1{,}5129$

$$d(A,B) = \sqrt{0{,}0016 + 1{,}5129} = 1{,}2307$$

Das Merkmal *relative Vokalhäufigkeit* trägt also sehr wenig zum Abstand der beiden Datenpunkte bei und ist somit kaum für die Klassifizierung relevant.

Zur Lösung dieser Probleme müssen die verwendeten Daten normalisiert werden.

4.1.4.1 Min-Max-Normalisierung

Bei der Min-Max-Normalisierung werden alle auftretenden Merkmalsausprägungen auf das Intervall $[0;1]$ abgebildet. Für ein Merkmal m wird dazu der größte ($\max_m$) und der kleinste ($\min_m$) in den Trainingsdaten vorkommende Wert aller Merkmalsausprägungen ermittelt. Anschließend wird jede Merkmalsausprägung x_m mit folgender Formel zu $\tilde{x}_m$ normalisiert:

$$\tilde{x}_m = \frac{x_m - \min_m}{\max_m - \min_m}$$

Durch die Abbildung auf das Intervall $[0;1]$ fallen unterschiedliche Skalierungen der Merkmale nicht mehr ins Gewicht. Außerdem wird das Problem der Größen mit unterschiedlichen Einheiten behoben. Einheiten spielen keine Rolle mehr, da es sich bei der normalisierten Größe $\tilde{x}_m$ um ein relatives Maß handelt. Problematisch ist die Min-Max-Normalisierung, falls es in den Trainingsdaten extreme Ausreißer bei einigen Merkmalsausprägungen gibt. Dies führt dazu, dass der Großteil der Daten in ein sehr kleines Teilintervall von $[0;1]$ abgebildet wird, was dazu führen kann, dass solche Merkmale bei der Abstandsberechnung kaum ins Gewicht fallen (Ertel, 2021, S. 228).

4.1.4.2 Standardisierung / Z-Score-Normalisierung

Die Standardisierung löst das Problem der extremen Ausreißer. Hierzu werden für ein Merkmal m zunächst der Mittelwert μ_m all seiner Ausprägungen x_m und die dazugehörige Standardabweichung σ_m bestimmt. Anschließend wird jede Merkmalsausprägung x_m mit folgender Formel zu $\tilde{x}_m$ normalisiert:

$$\tilde{x}_m = \frac{x_m - \mu_m}{\sigma_m}$$

Das standardisierte Merkmal hat damit den Mittelwert 0 und die Standardabweichung 1. Die Werte für $\tilde{x}_m$ schwanken nun um 0 und können somit auch negativ werden, was jedoch i. A. unproblematisch ist. Auch hier ist die Problematik der unterschiedlichen Einheiten behoben.

Im Gegensatz zur Min-Max-Normalisierung liegen die normalisierten Merkmalsausprägungen nicht in einem fest definierten Intervall, das Problem der unterschiedlichen Skalierung wird dennoch behoben (Ertel, 2021, S. 228).

4.1.5 Der Lernprozess des k-nächste-Nachbarn-Algorithmus

Bevor ein KI-System unter Verwendung des k-nächste-Nachbarn-Algorithmus neue ungelabelte Daten zuverlässig klassifizieren kann, muss es verschiedene Phasen durchlaufen:

- Phase 0: Vorbereitung des Lernprozesses
- Phase 1: Training des Modells
- Phase 2: Automatisierte Bestimmung des optimalen Werts für k
- Phase 3: Testen des Modells

4.1.5.1 Phase 0: Vorbereitung des Lernprozesses

Bevor mit dem Training des Modells begonnen werden kann, müssen zunächst einige Vorbereitungen getroffen werden:

1. *Bereitstellung und Aufteilung bereits gelabelter Daten*

   Als Verfahren des überwachten maschinellen Lernens müssen zunächst gelabelte Daten zur Verfügung gestellt werden, mit deren Hilfe ein neuer Datenpunkt anhand des in Abschnitt 4.1.2 beschriebenen Verfahrens klassifiziert werden kann. Da für die weiteren Phasen des Lernprozesses jeweils eigene Datensätze benötigt werden, müssen die verfügbaren gelabelten Daten aufgeteilt werden.

   Die Trainingsdaten werden dazu verwendet, einen neuen Datenpunkt anhand des in Abschnitt 4.1.2 beschriebenen Verfahrens zu klassifizieren. Die Validierungsdaten werden zur Optimierung des Modells, insbesondere zur automatisierten Bestimmung des optimalen Werts für den Hyperparameter k benötigt (s. Abschnitt 4.1.5.3). Die Testdaten verwendet man, um die Zuverlässigkeit des trainierten und optimierten Modells zu testen.

2. *Festlegung des Hyperparameters k*

   Der Hyperparameter k legt fest, wie viele der nächsten Nachbarn zur Klassifizierung eines neuen Datenpunktes herangezogen werden. Da die Klassifizierung durch eine Mehrheitsentscheidung unter den nächsten Nachbarn erfolgt (s. Abschnitt 4.1.2) und ein Gleichstand dies verhindert, sollte zumindest bei der Verwendung von zwei Klassen ein ungerader Wert für k gewählt werden. Um jedoch das bestmögliche Klassifizierungsergebnisse zu erzielen, empfiehlt es sich, k automatisiert mithilfe eines Optimierungsverfahrens zu bestimmen (s. Abschnitt 4.1.5.3). Ansonsten können ungünstig gewählte Werte für k zu Problemen bei der Klassifizierung führen (s. Beispiel zur Klassifizierung von Schwertlilien in Abschnitt 4.1.6.2).

4.1.5.2 Phase 1: Training des Modells

Die Trainingsphase des k-nächste-Nachbarn-Algorithmus beschränkt sich auf das Bereitstellen und Laden der Trainingsdaten. Da das Modell mit diesem einfachen Schritt bereits „trainiert“ ist, zählt man den k-nächste-Nachbarn-Algorithmus zu den Verfahren des „Lazy Learning“ (faulen Lernens).

4.1.5.3 Phase 2: Automatisierte Bestimmung des „optimalen“ Werts für k

4.1.5.3.1 Phase 2 mithilfe von Validierungsdaten

Der „optimale“ Wert für den Hyperparameter k kann mithilfe der Validierungsdaten automatisiert bestimmt werden. Dieser Wert wird anhand des folgenden Verfahrens bestimmt:

> Für alle Validierungsdatenpunkte V_i:
>
> Klassifiziere V_i anhand des k-nächste-Nachbarn-Algorithmus für $k \in \{1; 2; \ldots; k_{\max}\}$[1].
>
> Notiere für alle k-Werte, ob das Klassifizierungsergebnis von V_i korrekt ist, also mit dem Label übereinstimmt.
>
> Wähle für k einen Wert, bei dem die Klassifizierungsergebnisse am häufigsten korrekt sind.

[1] Der Wert $k_{\max}$ sollte deutlich kleiner als die Mächtigkeit der kleinsten Klasse der Trainingsdaten sein.

Beispiel (Bestimmung des „optimalen“ k mithilfe von Validierungsdaten)

In diesem Beispiel werden zwei Klassen betrachtet, weshalb lediglich ungerade Werte für k in Betracht gezogen werden. Der blau gelabelte Validierungsdatenpunkt V_1 wird für verschiedene Werte von k klassifiziert (s. Abbildung 4.7).

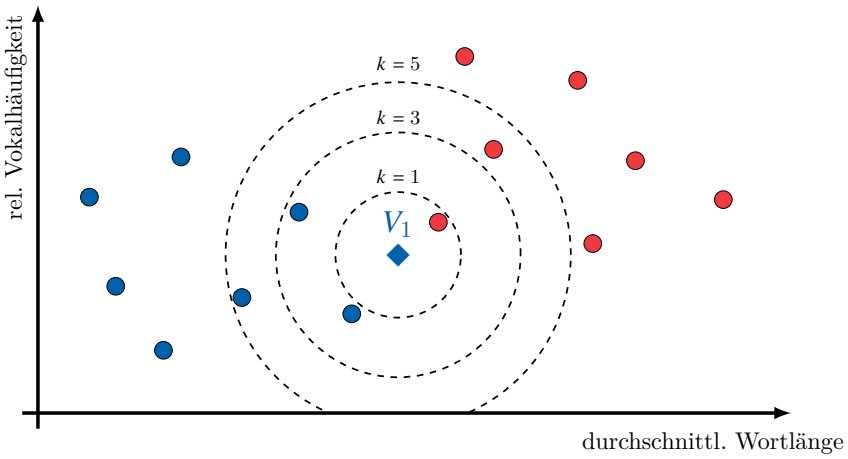


Abb. 4.7: Klassifizierung des Validierungsdatenpunkts V_1 für $k = 1,3,5$.

Ergebnis nach dem ersten Durchlauf von Schritt 1:

	$k = 1$	$k = 3$	$k = 5$	...
V_1	`falsch`	`korrekt`	`korrekt`	...

Erhält man nach der Klassifizierung weiterer Datenpunkte (nicht in der Abbildung dargestellt) die folgenden Ergebnisse, wird $k = 3$ als optimaler Wert für den Hyperparameter k gewählt.

	$k = 1$	$k = 3$	$k = 5$	...
V_1	`falsch`	`korrekt`	`korrekt`	...
V_2	`korrekt`	`korrekt`	`korrekt`	...
V_3	`falsch`	`korrekt`	`falsch`	...
Anzahl korrekt	1	3	2	...

4.1.5.3.2 Phase 2 mithilfe der Kreuzvalidierung

Die Aufteilung der gelabelten Daten in drei disjunkte Datenmengen (s. Abschnitt 4.1.5.1) hat den Nachteil, dass die Validierungsdaten nach der Bestimmung des „optimalen“ Werts für k nicht mehr weiterverwendet werden können. Im Rahmen der **Kreuzvalidierung** (engl. cross-validation) kann jedoch erreicht werden, dass auch die Validierungsdaten als Trainingsdaten zur Verfügung stehen.

Anstelle von drei Datenmengen werden die gelabelten Daten hierbei zunächst in zwei Datenmengen X und Y aufgeteilt, wobei Y, wie bisher, die Testdaten bildet. Die Datenmenge X wird nun in $j \in \mathbb{N}$ gleich große Teilmengen $X_1, X_2, \ldots, X_j$ aufgeteilt. In Abbildung 4.8 ist eine solche Aufteilung für $j = 4$ zu sehen.

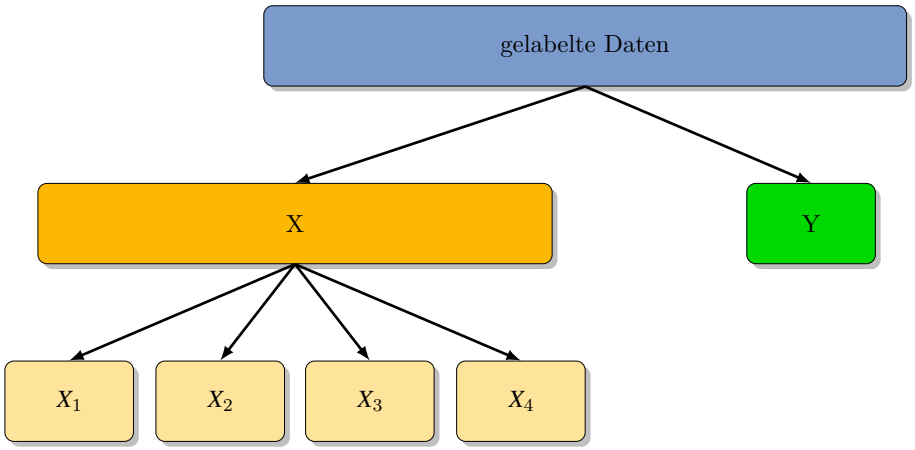


Abb. 4.8: Beispielhafte Aufteilung der gelabelten Daten für $j = 4$.

Nun wird in mehreren Iterationen jede Teilmenge X_i einmal als Validierungsdatensatz verwendet, die übrigen Datensätze aus X bilden zusammen die Trainingsdaten. Enthält jedes X_i genau einen Datenpunkt, so spricht man vom **Leave-One-Out-Verfahren**. Dazu wird das bisherige Verfahren angepasst:

> Für alle Trainingsdatenpunkte X_i:
>
> Klassifiziere X_i anhand des k-nächste-Nachbarn-Algorithmus für $k \in \{1; 2; \ldots; k_{\max}\}$.
>
> Notiere für alle k-Werte, ob das Klassifizierungsergebnis von X_i korrekt ist, also mit dem Label übereinstimmt.

Wähle für k einen Wert, bei dem die Klassifizierungsergebnisse am häufigsten korrekt sind.

Beispiel (Bestimmung des „optimalen“ k mithilfe von Leave-One-Out)

Anhand der in Abbildung 4.9 dargestellten Datenmenge soll das Leave-One-Out-Verfahren demonstriert werden. Da die Klassifizierung in eine von zwei Klassen erfolgen soll, kommen lediglich ungerade Werte für k in Betracht.

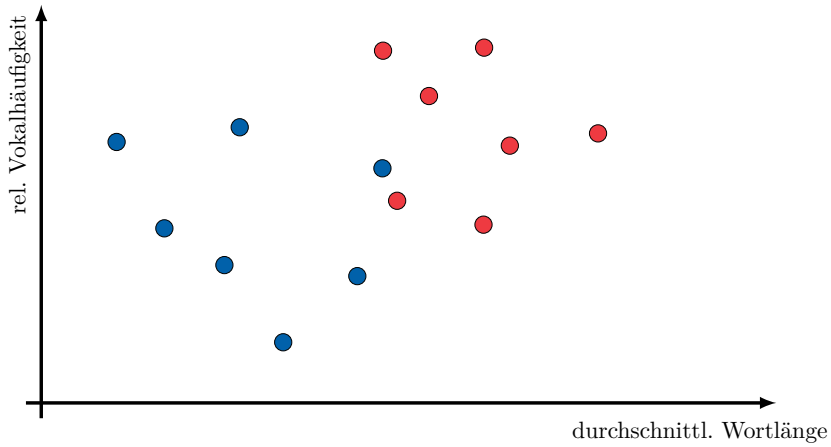


Abb. 4.9: Ausgangssituation des Leave-One-Out-Verfahrens.

Dazu wird zunächst der Datenpunkt X_1 (rotes Label) zufällig als Validierungsdatenpunkt ausgewählt und anhand der übrigen Datenpunkte klassifiziert (s. Abbildung 4.10).

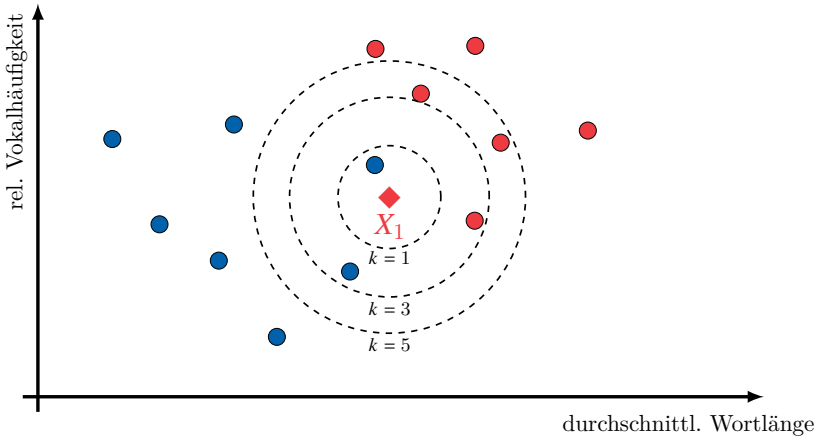


Abb. 4.10: Visualisierung der Klassifizierung von X_1 für $k = 1{,}3{,}5$ nach dem Leave-One-Out-Verfahren.

Ergebnis nach dem ersten Durchlauf von Schritt 1:

	$k = 1$	$k = 3$	$k = 5$	...
X_1	falsch	falsch	korrekt	...

Anschließend wird X_2 (blaues Label) aus den Trainingsdaten als Validierungsdatenpunkt ausgewählt und mithilfe aller übrigen Punkte aus den Trainingsdaten klassifiziert (s. Abbildung 4.11).

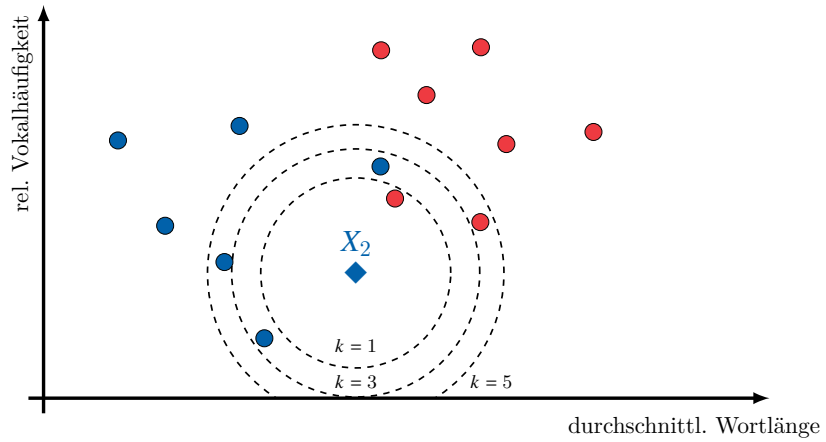


Abb. 4.11: Visualisierung der Klassifizierung von X_2 für k = 1,3,5 nach dem Leave-One-Out-Verfahren.

Die Ergebnistabelle erweitert sich somit um eine Zeile:

	$k = 1$	$k = 3$	$k = 5$	...
X_1	falsch	falsch	korrekt	...
X_2	falsch	korrekt	korrekt	...

Dieses Vorgehen wird nun für die restlichen Datenpunkte durchgeführt. Anschließend wird, wie in Abschnitt 4.1.5.3.1, der Wert für k gewählt, der am häufigsten zu einer korrekten Klassifizierung führt.

4.1.5.4 Phase 3: Testen des Modells

In dieser Phase wird überprüft, ob das in den vorangegangenen Phasen erstellte Modell auch „unbekannte“ Daten zuverlässig klassifizieren kann. Dazu ist es erforderlich, dass die bereitgestellten Testdaten nicht bereits in den Trainings- oder Validierungsdaten verwendet wurden.

Die Qualität bzw. Zuverlässigkeit der Klassifizierung kann anhand verschiedener Gütemaße (s. Entscheidungsbaum, Abschnitt 3.1.4.1) bewertet werden. Mithilfe einer Konfusionsmatrix können die Ergebnisse der Klassifizierung der Testdaten veranschaulicht werden (s. Entscheidungsbaum, Abschnitt 3.1.4.2). Sofern das Modell in dieser Phase zuverlässige Ergebnisse liefert, kann es zur Klassifizierung ungelabelter Daten verwendet werden.

4.1.6 Einflussfaktoren und Grenzen

Da der k-nächste-Nachbarn-Algorithmus ein Verfahren des überwachten maschinellen Lernens ist, hängt sein Erfolg von den verwendeten Trainingsdaten ab. Ein Vorteil gegenüber anderen Verfahren liegt darin, dass die Trainingsphase, die lediglich aus der Bereitstellung der Trainingsdaten besteht, nicht aufwendig ist. Da bei der Klassifizierung jedoch der Abstand zu jedem Trainingsdatenpunkt berechnet wird, wächst der Rechenaufwand linear mit der Größe des Trainingsdatensatzes. Für die Klassifizierung mit n Trainingsdatenpunkten beträgt die Laufzeit des Algorithmus $O(n)$ (Russel & Norvig, 2012, S. 858 f). Darüber hinaus kann die Beschaffenheit der Trainingsdaten die Zuverlässigkeit der Ergebnisse beeinflussen.

4.1.6.1 Grenzen aufgrund der Beschaffenheit der Daten

Je mehr Dimensionen verwendet werden, desto unzuverlässiger können die Klassifikationsergebnisse werden, d. h. mit zunehmender Dimensionalität d des Merkmalsraums werden die Unterschiede zwischen der maximalen gemessenen Distanz ($D_{\max}$) und der minimalen gemessenen Distanz ($D_{\min}$) zweier beliebiger d-dimensionaler Datenpunkte geringer. Es lässt sich zeigen, dass sogar eine Konvergenz vorliegt: $D_{\min} \xrightarrow{d\to\infty} D_{\max}$. Folglich haben zwei beliebige Datenpunkte jeweils nahezu die gleiche Distanz. Da der k-nächste-Nachbarn-Algorithmus jedoch auf der Abstandsberechnung basiert, kann er nicht mehr zuverlässig klassifizieren. Man

spricht in diesem Zusammenhang vom **Fluch der Dimensionalität** (Russel & Norvig, 2012, S. 858 f).

Eine weitere Grenze des Modells ist in der Korrelation der Merkmale zu finden. Korrelieren die Merkmale der Daten nicht bzw. nur schwach, führt der Algorithmus zwar eine Klassifizierung durch, diese ist jedoch nicht zuverlässig. Das Testen des Modells (s. Abschnitt 4.1.5.4) ist deshalb von besonders hoher Bedeutung.

Eine ungleichmäßige Verteilung der Trainingsdaten, in der eine oder mehrere Klassen überrepräsentiert sind, führt ebenfalls zu unzuverlässigen Ergebnissen. Insbesondere für größere Werte für den Hyperparameter k wird ein neuer Datenpunkt mit einer höheren Wahrscheinlichkeit einer der dominierenden Klassen zugeordnet, da deren Datenpunkte häufiger unter den k nächsten Nachbarn vertreten sind. Bestehen die Trainingsdaten beispielsweise aus elf blauen und lediglich zwei roten Trainingsdatenpunkten (s. Abbildung 4.12), wird ein neuer Datenpunkt unabhängig von seiner Lage für $k > 4$ immer als blau klassifiziert.

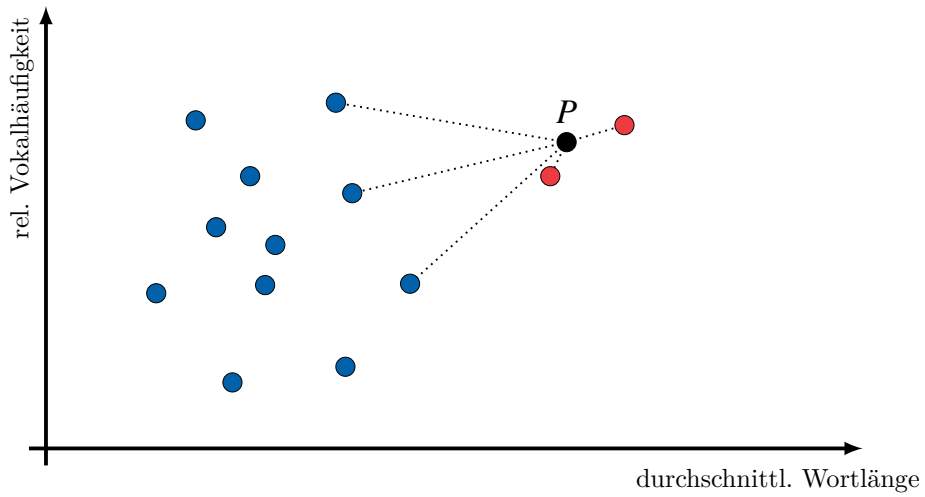


Abb. 4.12: Ungleichmäßige Verteilung der Trainingsdaten auf einzelne Klassen.

Folglich sollte auf eine möglichst ausgewogene Verteilung der Trainingsdaten auf die einzelnen Klassen geachtet werden. Insbesondere ist es wichtig, dass k deutlich kleiner als die Mächtigkeit der kleinsten in den Trainingsdaten vertretenen Klasse ist.

Ein Datenpunkt kann darüber hinaus durch Trainingsdaten falsch klassifiziert werden, wenn diese nicht die Gesamtheit der Daten widerspiegeln. Bestehen bei der Sprachenerkennung die Trainingsdaten einer Sprache beispielsweise nur aus Textabschnitten eines einzigen Buchs, so

repräsentieren diese den Stil und den Sprachgebrauch des Autors, was zu Problemen bei der Klassifizierung anderer Texte führen kann.

4.1.6.2 Grenzen in Bezug auf den Hyperparameter k

Abgesehen von Problemen bei der Beschaffenheit der Daten kann die Wahl des Hyperparameters k die Zuverlässigkeit der Klassifizierung beeinträchtigen. Wird k zu klein gewählt, werden nur wenige Trainingsdatenpunkte zur Klassifizierung herangezogen, sodass Ausreißer das Ergebnis stark beeinflussen und somit verfälschen können. Man spricht in diesem Fall von Überanpassung (*overfitting*).

Das folgende Beispiel basiert auf den Irisdaten (Fisher, 1936), einer bekannten Datenmenge, mit der man Schwertlilien anhand von Eigenschaften ihrer Blätter einer von drei Arten (*Iris setosa*, *Iris virginica* oder *Iris versicolor*) zuordnen kann. In Abbildung 4.13 wurden die Datenpunkte unter Verwendung der beiden Merkmale „Kelchblattlänge“ und „Kelchblattbreite“ dargestellt. Zusätzlich wurde die **Decision Surface** für $k = 1$ farblich hervorgehoben. Eine Decision Surface veranschaulicht für jeden beliebigen Punkt im Merkmalsraum die jeweiligen Klassifizierungsergebnisse als unterschiedliche Bereiche in Abhängigkeit von k (Herbold, 2022). Ein Datenpunkt, der im violetten Bereich liegt, würde beispielsweise als *Iris setosa* klassifiziert werden.

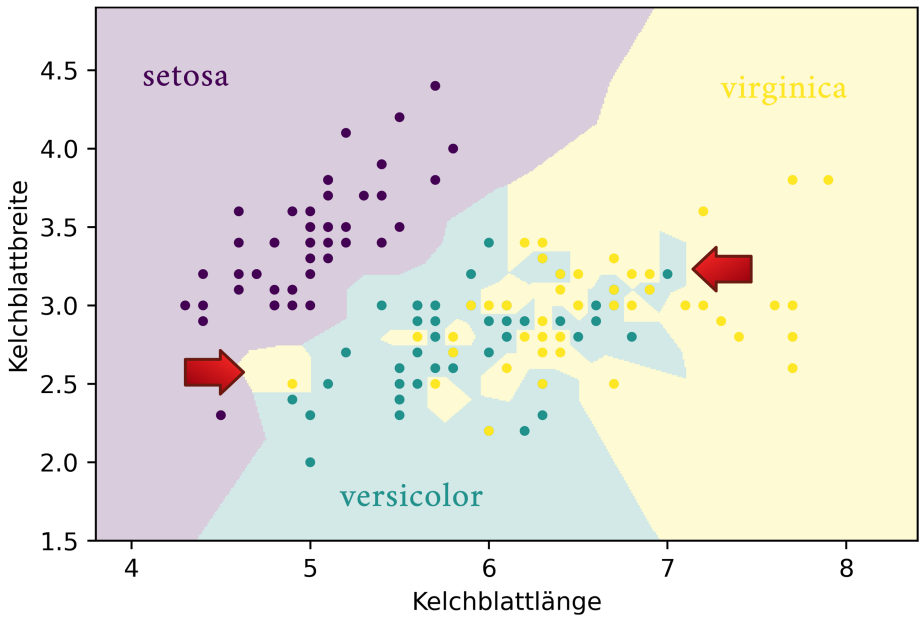


Abb. 4.13: Decision Surface für $k = 1$.

In der Decision Surface für $k = 1$ (s. Abbildung 4.13) wurden zwei Bereiche (durch Pfeile) hervorgehoben, in denen jeweils ein Ausreißer einen Bereich der Decision Surface erzeugt, in dem man ein anderes Klassifizierungsergebnis erwarten würde. Durch das zu klein gewählte k kommt es zu einer Überanpassung, die das Klassifizierungsergebnis verfälschen kann. In Abbildung 4.14 sieht man (durch einen Pfeil hervorgehoben), dass für ein groß gewähltes k (hier $k = 20$) ein einzelner grüner Bereich innerhalb des sonst ausschließlich gelben Bereichs entstanden ist. Da in diesem Bereich kein einziger grüner Datenpunkt enthalten ist, wird die Klassifizierung hauptsächlich durch weiter entfernt liegende Punkte bestimmt. Auch sehr große k-Werte können also die Zuverlässigkeit beeinträchtigen.

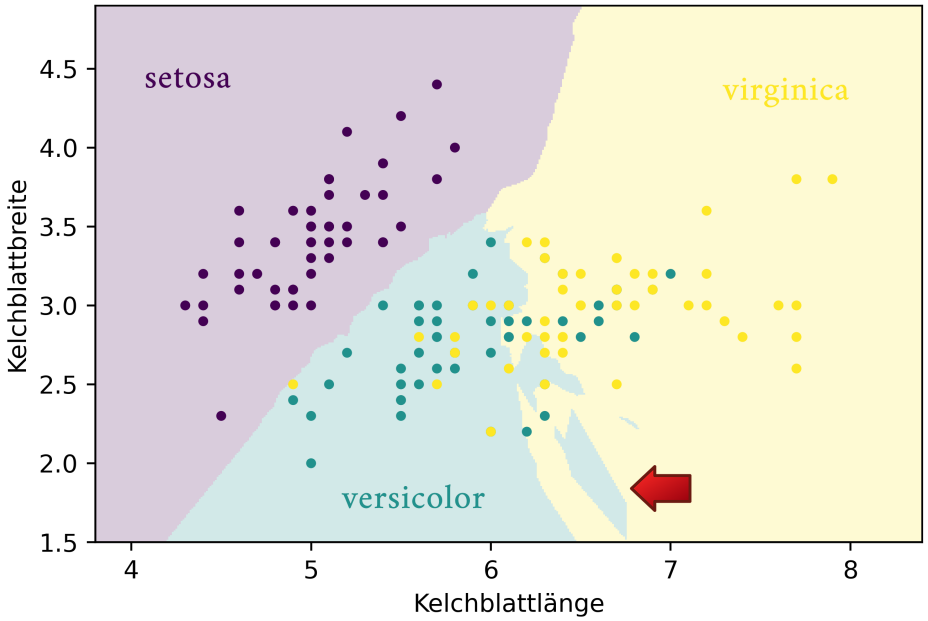


Abb. 4.14: Decision Surface für $k = 20$.

Folglich sollte der Wert für k möglichst automatisiert bestimmt (s. Abschnitt 4.1.5.3) und die Zuverlässigkeit des Modells anhand von Testdaten überprüft werden.

4.1.7 Exkurs: Regression mithilfe des k-nächste-Nachbarn-Algorithmus

Während bei der Klassifizierung einem Datenpunkt anhand seiner Merkmale eine Klasse zugeordnet wird, erfolgt bei einer Regression die Zuweisung einer reellen Zahl. Ausgehend von den Merkmalen (unabhängige Variablen) kann die Ausprägung eines davon abhängigen Merkmals (abhängige Variable) berechnet werden. Die abhängige Variable eines neuen Datenpunkts P

kann nach der folgenden Vorgehensweise mithilfe des k-nächste-Nachbarn-Algorithmus ermittelt werden:

1. Bestimme die k nächsten Nachbarn, die in Bezug auf die unabhängigen Variablen den geringsten Abstand zu P haben.
2. Berechne das arithmetische Mittel der Werte der abhängigen Variablen dieser Nachbarn.
3. Weise der abhängigen Variablen von P den Wert des arithmetischen Mittels zu.

Das folgende Beispiel verwendet zwei Variablen, die unabhängige Variable x und abhängige Variable y. Von Datenpunkt P ist lediglich die x-Koordinate mit $x = 4$ bekannt. Anhand der Trainingsdatenpunkte (blau markiert) soll der Wert der y-Koordinate mithilfe des k-nächste-Nachbarn-Algorithmus berechnet werden (s. Abbildung 4.15).

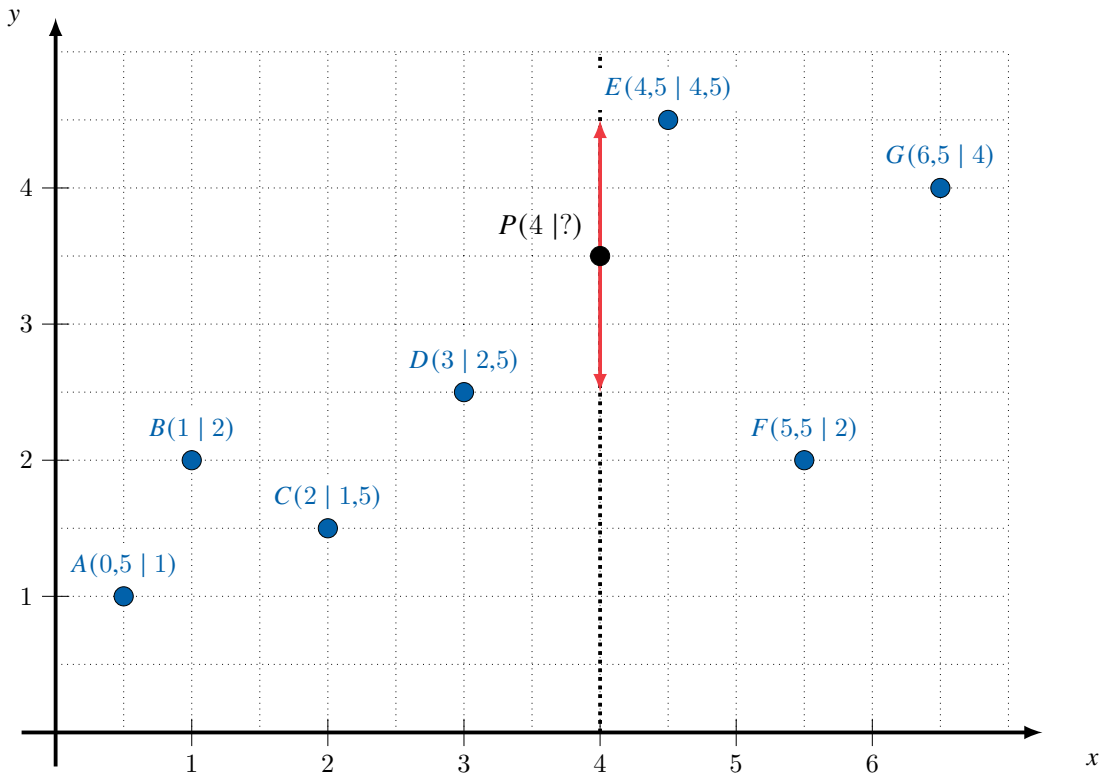


Abb. 4.15: Beispiel für eine Regression mit dem k-nächste-Nachbarn-Algorithmus.

Für $k = 1$ ist $E(4{,}5 \mid 4{,}5)$ der nächste Nachbar. Da nur ein Datenpunkt verwendet wird, entspricht das arithmetische Mittel der y-Koordinate von E. P wird also $y = 4{,}5$ zugeordnet

(s. Abbildung 4.16). Der Funktionsgraph (grau dargestellt) zeigt das Ergebnis der Regression für alle x-Koordinaten:

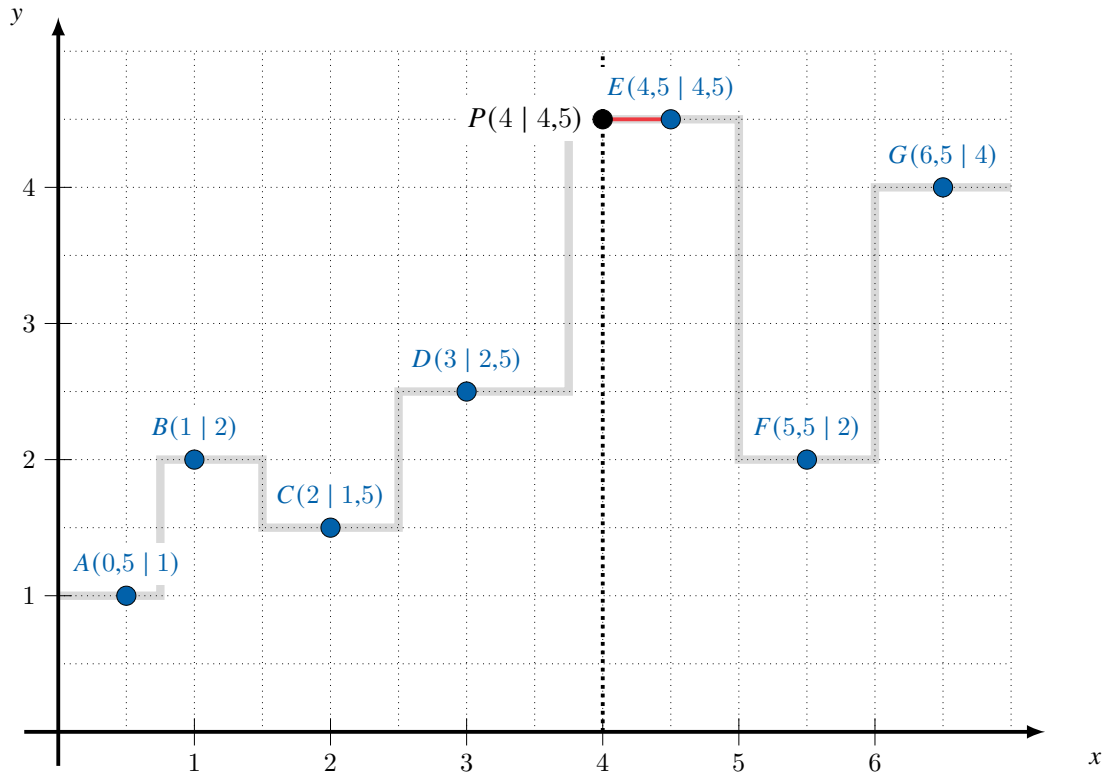


Abb. 4.16: Regression für $k = 1$.

Für $k = 2$ sind die Punkte $E(4{,}5 \mid 4{,}5)$ und $D(3 \mid 2{,}5)$ die drei nächsten Nachbarn (s. Abbildung 4.17). Für die y-Koordinate von P ergibt sich:

$$y_P = \frac{y_E + y_D}{2} = \frac{4{,}5 + 2{,}5}{2} = 3{,}5$$

Für $k = 3$ sind die Punkte $E(4{,}5 \mid 4{,}5)$, $D(3 \mid 2{,}5)$ und $F(5{,}5 \mid 2)$ die beiden nächsten Nachbarn (s. Abbildung 4.18). Für die y-Koordinate von P ergibt sich:

$$y_P = \frac{y_E + y_D + y_F}{3} = \frac{4{,}5 + 2{,}5 + 2}{3} = 3$$

Ein typisches Anwendungsbeispiel für eine Regression ist die Vorhersage von Hauspreisen anhand von Merkmalen, wie z. B. der Wohnfläche, des Baujahrs und der Anzahl an Zimmern.

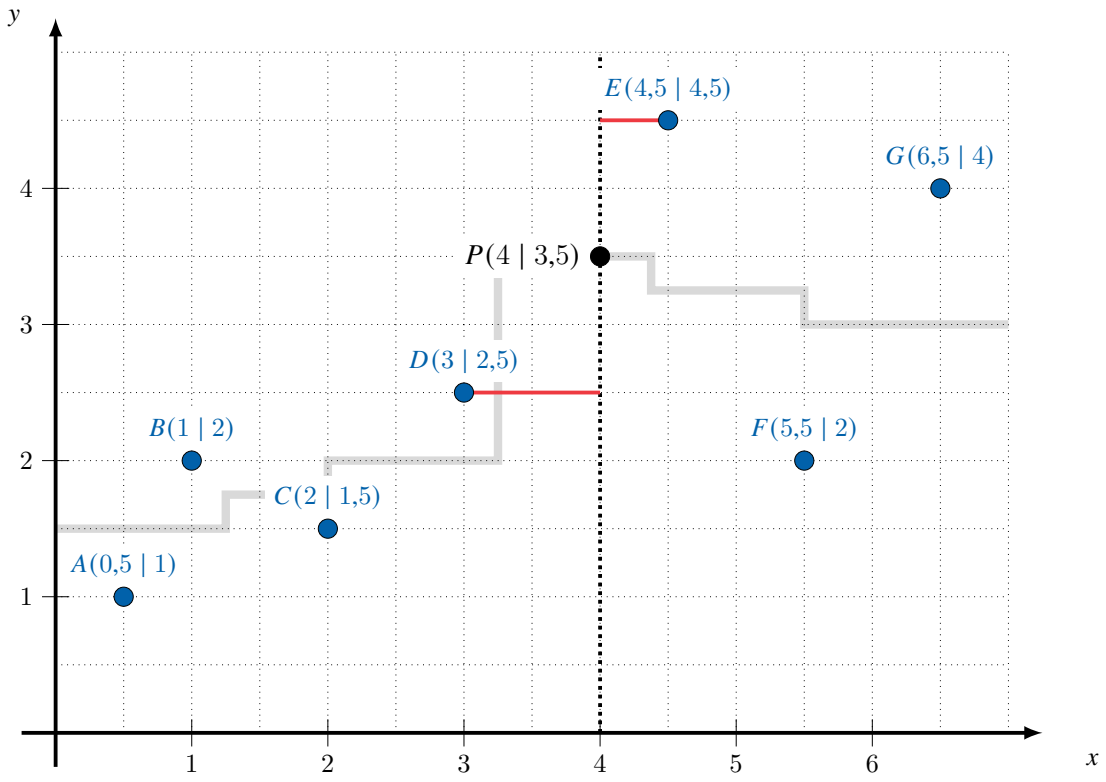


Abb. 4.17: Regression für $k = 2$.

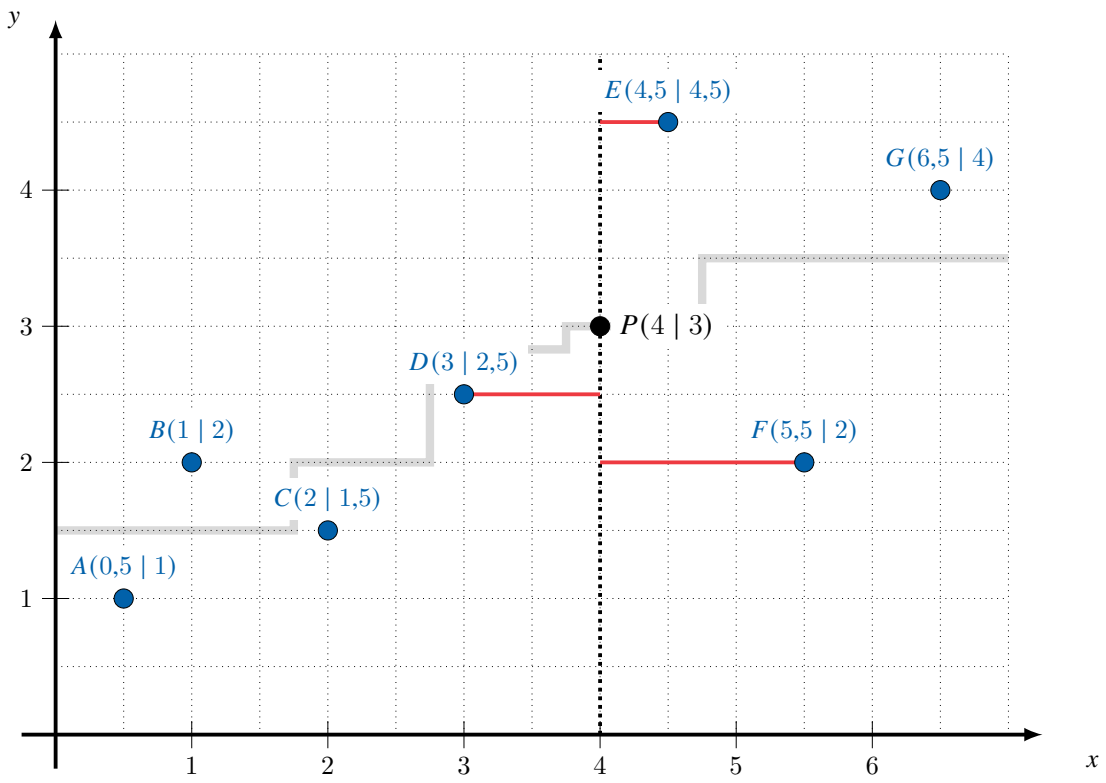


Abb. 4.18: Regression für $k = 3$.

4.2 Didaktische Hinweise / Bezug zum Lehrplan

4.2.1 Einordnung in den Lehrplan

Gemäß LehrplanPLUS wählen die Lehrkräfte als Beispiel für einen Algorithmus maschinellen Lernens entweder den Entscheidungsbaum-Algorithmus oder den k-nächste-Nachbarn-Algorithmus aus. Im Hinblick auf den Entscheidungsbaum-Algorithmus wird die Kompetenzerwartung noch zwischen der Informatik und der spät beginnenden Informatik wie folgt unterschieden:

- **Informatik 11 (NTG)**

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

- **Spät beginnende Informatik 11 (HG, SG, MuG, SWG)**

  *Die Schülerinnen und Schüler erläutern die **Idee** eines ausgewählten Algorithmus maschinellen Lernens (k-nächste-Nachbarn-Algorithmus oder Entscheidungsbaum-Algorithmus) an konkreten Beispielen.*

Der Lehrplan fordert also ein Verständnis der allgemeinen Funktionsweise (NTG) bzw. der Idee (spät beginnend) des k-nächste-Nachbarn-Algorithmus anhand konkreter Beispiele. Darüber hinaus erwerben die Schülerinnen und Schüler bei der Behandlung des k-nächste-Nachbarn-Algorithmus zusätzlich folgende Kompetenz:

> *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.*

Die Bedeutung der Beschaffenheit gelabelter Daten (s. Abschnitt 4.1.6) stellt eine Grundlage bei der Arbeit mit dem k-nächste-Nachbarn-Algorithmus dar. In diesem Zusammenhang kann auch deren Verwendung zu unterschiedlichen Zwecken (Trainings-, Validierungs- und Testdaten) an geeigneter Stelle thematisiert werden. Der Hyperparameter k ist für die Qualität der Klassifizierungsergebnisse zentral und sollte entsprechend behandelt werden. Neben dem

Aufzeigen möglicher Probleme (s. Abschnitt 4.1.6) sollte auch auf ein Optimierungsverfahren für k eingegangen werden. Dies kann mit dem „Demonstrator für maschinelles Lernen“ simuliert werden. Dieses Werkzeug eignet sich sowohl zur Demonstration der Funktionsweise des k-nächste-Nachbarn-Algorithmus als auch zur Analyse der Zuverlässigkeit der Klassifizierung. Neben der manuellen Festlegung verschiedener k-Werte beinhaltet der Demonstrator mit dem Leave-One-Out-Verfahren (s. Abschnitt 4.1.5.3.2) ein Verfahren zur Berechnung des optimalen Werts für k.

4.2.2 Durchführung

Die nachfolgenden didaktischen Hinweise beziehen sich auf die Kompetenzerwartungen des NTG. Für diesen Themenbereich werden ca. sechs Unterrichtsstunden, für die spät beginnende Informatik ca. vier Unterrichtsstunden vorgeschlagen. Im Abschnitt 4.2.2.7 sind die Stellen aufgeführt, an denen sich in der spät beginnenden Informatik Unterschiede in der Herangehensweise ergeben können.

4.2.2.1 Einstieg

Als Einstieg in die Sequenz bietet es sich an, die Schülerinnen und Schüler die Grundidee des k-nächste-Nachbarn-Algorithmus intuitiv erleben zu lassen.

Es wird vorgeschlagen, dazu das Anwendungsbeispiel der Sprachenerkennung von Texten zu verwenden. Anhand von Beispieltexten verschiedener Sprachen, die digital oder als Arbeitsblatt zur Verfügung gestellt werden (s. Abschnitt 4.3.1.1), identifizieren die Schülerinnen und Schüler Merkmale, anhand derer man einem Text eine Sprache zuordnen könnte. Dabei wird angenommen, dass Texte, die in derselben Sprache verfasst sind, ähnliche Merkmalsausprägungen besitzen. Zur besseren Veranschaulichung erfolgt eine Fokussierung auf zwei Merkmale, um die Datenpunkte in einem zweidimensionalen Koordinatensystem darstellen zu können. Dies erfolgt im vorliegenden Vorschlag mit den beiden Merkmalen *relative Vokalhäufigkeit* und *durchschnittliche Wortlänge*.

Tragen die Schülerinnen und Schüler Datenpunkte, die Texte einer Sprache repräsentieren, in ein zweidimensionales Koordinatensystem ein, wird die Ähnlichkeit von zwei Datenpunkten aus ihren Abständen ersichtlich. Datenpunkte, die Texte der gleichen Sprache repräsentieren,

liegen in der Regel nahe beieinander, sind also „benachbart“. Tragen die Schülerinnen und Schüler nun einen Datenpunkt ein, der zu einem Text einer unbekannten Sprache gehört, weisen sie ihm die Sprache zu, die die Datenpunkte in seiner „Nachbarschaft“ aufweisen. Dabei sollte eine Unterscheidung zwischen gelabelten und ungelabelten Daten erfolgen.

4.2.2.2 Funktionsweise des k-nächste-Nachbarn-Algorithmus

Da ein Computer im Gegensatz zu einem Menschen nicht intuitiv arbeiten kann, ergibt sich die Notwendigkeit, einen Klassifizierungsalgorithmus zu erarbeiten, der anhand von gelabelten Daten lernt, Datenpunkte zu klassifizieren. Dies kann mit folgendem Beispiel verdeutlicht werden: Anhand der Darstellung und unter Zuhilfenahme einer tabellarischen Auflistung der Abstände (s. Abschnitt 4.3.1.2) versuchen die Schülerinnen und Schüler nun, weitere noch nicht gelabelte Datenpunkte zu klassifizieren. Dabei sollte beachtet werden, dass darunter auch Datenpunkte sind, die nicht offensichtlich zugeordnet werden können. Bezüglich der Berechnung des Abstands zweier Punkte im zweidimensionalen Raum reicht ein Verweis auf den Satz des Pythagoras aus, den die Schülerinnen und Schüler aus dem Mathematikunterricht der 9. Jahrgangsstufe kennen. Eine umfassende Thematisierung verschiedener Abstandsmaße (s. Abschnitt 4.1.3) führt an dieser Stelle sicherlich zu weit.

Mit diesem Vorgehen wird den Schülerinnen und Schülern deutlich, dass ein Algorithmus zur Klassifizierung erforderlich ist. Die Grundidee des k-nächste-Nachbarn-Algorithmus (s. Abschnitt 4.1.2) kann mit dem „Demonstrator für maschinelles Lernen“ erarbeitet werden (s. Abschnitt 4.3.2). In diesem Zusammenhang werden die Trainingsdaten und der Hyperparameter k erstmalig thematisiert. Für die Sprachenerkennung steht ein Datensatz für die Sprachen Deutsch, Englisch und Französisch bereit (s. Abschnitt 4.3.2.1). Nach einer kurzen Einführung durch die Lehrkraft sollten die Schülerinnen und Schüler das Programm eigenständig ausprobieren und selbst ausgewählte Texte klassifizieren. Für die Einführung in das Programm kann bei Bedarf ein Einführungsvideo verwendet werden (Materialordner: `Einführung Sprachenerkennung Klassifizierung.mp4`).

Da die Schülerinnen und Schüler die Sprache der gewählten Beispiele kennen und somit sofort erkennen, ob die Klassifizierung korrekt ist, sollten sie auch mögliche Gründe für Fehlklassifizierungen sammeln. Dabei wird die Bedeutung des Hyperparameters k für den Erfolg der Klassifizierung deutlich. Einerseits könnten „Gleichstände“ bei manchen k-Werten

auftreten. Andererseits kommt es bei niedrigen k-Werten zu falschen Klassifizierungen, wenn die Datenpunkte in der Nähe von Ausreißern liegen. Ggf. kann dies durch die Klassifizierung entsprechender Textbeispiele zusätzlich verdeutlicht werden (Ordner „Demonstrationsbeispiele Trainingsdaten“ im Materialordner). Daran anschließend bietet es sich an, weitere Grenzen des Algorithmus zu thematisieren. Durch eine entsprechende Anpassung der Trainingsdaten kann sowohl die Notwendigkeit einer möglichst gleichmäßigen Verteilung der Trainingsdaten als auch eine Wahl eines Werts für k, der kleiner als die Mächtigkeit der kleinsten Klasse ist, demonstriert werden.

Durch Hinzunahme der Datensätze der spanischen Trainingsdaten (Ordner „Demonstrationsbeispiele Trainingsdaten“ im Materialordner) kann außerdem verdeutlicht werden, dass eine fehlende Trennschärfe zwischen den Merkmalsausprägungen zweier Klassen die Klassifizierung zusätzlich erschwert. Somit wird auch der Einfluss der Trainingsdaten auf das Ergebnis deutlich. Wichtig ist dabei, dass die Schülerinnen und Schüler durch ihr Experimentieren erkannt haben, dass sowohl die Wahl von k als auch die Auswahl der Trainingsdaten für die richtige Klassifizierung eines neuen Datenpunkts eine entscheidende Rolle spielen.

Im Anschluss rekapitulieren die Schülerinnen und Schüler die Schritte des k-nächste-Nachbarn-Algorithmus und formulieren den Algorithmus allgemein. Dabei bietet sich die folgende Reihenfolge an:

1. Vorbereitung
   a. Trainingsdaten bereitstellen
   b. Hyperparameter k festlegen[2]
2. Abstand des neuen Datenpunkts zu allen Trainingsdatenpunkten berechnen
3. Trainingsdaten nach Abstand aufsteigend sortieren
4. Neuen Datenpunkt anhand des Parameters k klassifizieren

Die Klassifizierung mithilfe von lediglich zwei Merkmalen hat den Vorteil, dass zweidimensionale Datenpunkte vorliegen, die grafisch veranschaulicht werden können. In Bezug auf die Klassifizierung von Sprachen stellt dies jedoch ein stark vereinfachtes Verfahren dar. Zur

[2] Da k im „Demonstrator für maschinelles Lernen“ nach dem letzten Schritt variiert werden kann und zu diesem Zeitpunkt noch kein Verfahren zur automatisierten Bestimmung bekannt ist, kann die Festlegung von k zu diesem Zeitpunkt auch Schritt 3 zugeordnet werden.

Darstellung einer realistischeren Herangehensweise mit einer Vielzahl an Merkmalen kann das folgende Video vorgeführt werden:

`https://experiments.withgoogle.com/visualizing-high-dimensional-space`

4.2.2.3 Wahl des „optimalen“ Werts für k

Bereits bei der Erarbeitung der Grundidee des k-nächste-Nachbarn-Algorithmus dürfte den Schülerinnen und Schülern die zentrale Bedeutung des Hyperparameters k bewusst geworden sein. Dabei stellt sich die Frage, wie man den bestmöglichen Wert für k bestimmen kann. Aufgrund der bisherigen Erfahrungen der Schülerinnen und Schüler mit dem „Demonstrator für maschinelles Lernen“ liegt die Idee nahe, mehrere bereits gelabelte Datenpunkte unter Verwendung von jeweils verschiedenen Werten für k zu klassifizieren und den Wert für k zu wählen, der am häufigsten zum Erfolg führt. Daran anknüpfend kann das Leave-One-Out-Verfahren (s. Abschnitt 4.1.5.3.2)[3] unter Zuhilfenahme des „Demonstrators für maschinelles Lernen“ vorgestellt werden. Nach einer kurzen Erläuterung des Verfahrens und der Demonstration einiger weniger Mikroschritte (s. Abschnitt 4.3.2.2) können die Schülerinnen und Schüler selbstständig den für den vorliegenden Datensatz optimalen Wert für k bestimmen. In diesem Zusammenhang sollte darauf hingewiesen werden, dass der Datenpunkt, der im jeweils aktuellen Schritt für verschiedene Werte für k klassifiziert wird, nicht mehr zu den Trainingsdaten gehört. Die Begriffe Trainings- und Validierungsdaten können somit voneinander abgegrenzt werden.

4.2.2.4 Normalisierung und Abstandsmaße

Nachdem die Schülerinnen und Schüler den k-nächste-Nachbarn-Algorithmus kennengelernt haben, sollte die Problematik unterschiedlich skalierter Merkmale angesprochen werden. Beispielsweise hat im Anwendungsbeispiel Sprachenerkennung das Merkmal *durchschnittliche Wortlänge* einen erheblich größeren Einfluss auf den berechneten Abstand als das Merkmal *relative Vokalhäufigkeit*. In diesem Zusammenhang kann die Normalisierung von Daten mithilfe der Min-Max-Normalisierung anhand eines Beispiels (s. Abschnitt 4.3.3) thematisiert werden

[3] Eine Thematisierung der Kreuzvalidierung im Allgemeinen führt hier wohl zu weit.

(s. Abschnitt 4.1.4.1). Mangels Kenntnis des Erwartungswerts und der Varianz führt eine Verwendung der Standardisierung (s. Abschnitt 4.1.4.2) an dieser Stelle zu weit.

Darüber hinaus stellt sich die Frage, wie der k-nächste-Nachbarn-Algorithmus mit nichtmetrischen Daten umgeht. In solchen Fällen ist die Euklidische Distanz als Abstandsmaß ungeeignet. Anhand eines Beispiels (s. Abschnitt 4.3.3) kann den Schülerinnen und Schülern die Hamming-Distanz (s. Abschnitt 4.1.3.4) vorgestellt werden.

4.2.2.5 Anwendung oder Vertiefung

Nachdem mit der Bestimmung des optimalen Werts für k und der Abstandsberechnung die zentralen Aspekte des k-nächste-Nachbarn-Algorithmus thematisiert wurden, bietet es sich an, den Algorithmus in einem neuen Szenario anzuwenden. Eine interessante Möglichkeit dazu bietet der `RAISE`-Playground, der auf der Programmierumgebung Scratch aufbaut und mit der *Text Classification* eine Erweiterung enthält, die den k-nächste-Nachbarn-Algorithmus zur Worterkennung verwendet (s. Abschnitt 4.3.4). Die Funktionsweise des Algorithmus läuft hier im Rahmen einer Blackbox ab, sodass die einzelnen Schritte nicht im Detail nachvollzogen werden können (s. Abschnitt 4.3.4.1). Da die Schülerinnen und Schüler jedoch selbstständig Labels und Trainingsdaten für ihr gewähltes Szenario auswählen müssen, erfahren sie erneut die Bedeutung der Beschaffenheit der Trainingsdaten für die Zuverlässigkeit der Klassifizierung. Die Möglichkeit des kreativen Einsatzes des k-nächste-Nachbarn-Algorithmus motiviert dabei zusätzlich. Folgende Themengebiete können im Rahmen einer Einzel-, Partner- oder Gruppenarbeit interessant sein:

- Die Erkennung der Stimmung des Benutzers anhand seiner Antworten.
- Die Bestimmung eines Sachgebiets/Schulfachs, über das der Benutzer spricht.
- Das Erraten von Begriffskategorien zu vom Benutzer genannten Begriffen.

Alternativ kann eine andere Einsatzmöglichkeit des k-nächste-Nachbarn-Algorithmus thematisiert werden. Anhand eines konkreten Anwendungsszenarios, in dem nicht die Zuordnung einer Klasse, sondern die Berechnung einer reellen Zielgröße benötigt wird, kann auf die Regression mithilfe des Algorithmus eingegangen werden (s. Abschnitt 4.1.7). Ein mögliches Anwendungsbeispiel hierfür ist der Zusammenhang zwischen der Wohnfläche einer Immobilie (unabhängige

Variable) und ihrem Preis (abhängige Variable). Bei Bedarf steht im Materialordner eine entsprechende Excel-Mappe zu dieser Thematik bereit (s. Abschnitt 4.3.5.2).

4.2.2.6 Testen des Modells

Zum Abschluss der Sequenz sollte auf die besondere Bedeutung der Testphase eingegangen werden (s. Abschnitt 4.1.5.4), was erneut im Szenario der Sprachenerkennung erfolgen kann. Mithilfe des „Demonstrators für maschinelles Lernen“ klassifiziert jede Schülerin und jeder Schüler jeweils zehn selbst gewählte Texte und notiert sich, wie viele Klassifizierungen korrekt sind. Anschließend werden die Ergebnisse zusammengetragen, von der Lehrkraft in eine Konfusionsmatrix eingetragen und mit den Schülerinnen und Schülern diskutiert. Hierfür steht im Materialordner die Datei `Konfusionsmatrix Deutsch Englisch.xlsx` bereit. Damit hierfür eine Vier-Felder-Tafel ausreicht, bietet es sich an, das Szenario auf zwei verschiedene Sprachen (z. B. Deutsch und Englisch) zu beschränken. Anhand eines Gütemaßes, beispielsweise der Genauigkeit, kann nun die Zuverlässigkeit der Klassifizierung berechnet werden.

4.2.2.7 Hinweise zur spätbeginnenden Informatik

Für die Schülerinnen und Schüler der spät beginnenden Informatik ist es nicht erforderlich, die Schritte des k-nächste-Nachbarn-Algorithmus allgemein zu formulieren, um eine Idee der Funktionsweise zu erhalten. Es genügt, den Algorithmus anhand eines Szenarios zu erarbeiten (s. Abschnitt 4.2.2.2). Ebenso reicht es aus, dass sie das Leave-One-Out-Verfahren exemplarisch nachvollziehen. Dazu eignet sich der Einsatz des „Demonstrators für maschinelles Lernen“ (s. Abschnitt 4.3.2.2). Zudem kann auf die Behandlung der Normalisierung von Daten und der Abstandsmaße (s. Abschnitt 4.2.2.4) verzichtet werden.

4.3 Material

4.3.1 Einführung in den k-nächste-Nachbarn-Algorithmus

4.3.1.1 Intuitive Klassifizierung von Texten

Für den in Abschnitt 4.2.2.1 beschriebenen Einstieg steht im Materialordner das Dokument `Merkmale identifizieren.docx` zur Verfügung. Anhand von drei Beispieltexten in den Sprachen Deutsch, Englisch und Französisch identifizieren die Schülerinnen und Schüler Merkmale, die dazu geeignet sind, einen Text einer der drei Sprachen zuzuordnen.

4.3.1.2 Die Erarbeitung der Grundidee des k-nächste-Nachbarn-Algorithmus

In Abschnitt 4.2.2.2 wurde vorgeschlagen, von einer zunächst intuitiven Klassifizierung von Sprachen zu einem algorithmischen Vorgehen überzuleiten. Hierzu stehen im Materialordner die Text-Datei `Klassifizierung von Sprachen.docx` und die Tabellenkalkulations-Datei `Abstände sortierbar.xlsx` zur Verfügung. Die Schülerinnen und Schüler erkennen, dass eine intuitive Klassifizierung von Sprachen nicht immer möglich ist. Anhand von Abständen zwischen Datenpunkten kann die Grundidee des k-nächste-Nachbarn-Algorithmus erarbeitet werden.

4.3.2 Demonstrator für maschinelles Lernen

Der „Demonstrator für maschinelles Lernen“ ist ein Programm, mit dessen Hilfe zentrale Aspekte des k-nächste-Nachbarn-Algorithmus sowie des Perzentrons (s. Kapitel 5) veranschaulicht werden können. Das Programmpaket enthält neben einer `jar`-Datei[4] zum Starten des Programms weitere Ordner und Dateien. Folgende sind für den k-nächste-Nachbarn-Algorithmus relevant:

- Ordner `daten_sprachen_klassifizierung`:
  Hier werden gelabelte Daten zur Klassifizierung im Rahmen der Sprachenerkennung abgelegt (siehe Abschnitt 4.3.1.1).

[4] Zur Verwendung des Programms wird eine möglichst aktuelle Java-Installation (mind. Version 8) benötigt.

- Ordner `daten_sprachen_k_abschaetzung`:
  Hier werden gelabelte Daten gespeichert, die zur Bestimmung des optimalen Werts für den Hyperparameter k anhand des Leave-One-Out-Verfahrens verwendet werden (s. Abschnitt 4.3.1.2).

- `daten.csv`:
  Hier können für ein selbstgewähltes Szenario gelabelte Daten gespeichert werden (s. Abschnitt 4.3.2.3).

- `Lizenzinformationen.txt`:
  Diese Datei enthält Quellenangaben zu den Textabschnitten, die für die gelabelten Daten verwendet wurden.

4.3.2.1 Sprachenerkennung: Klassifizierung

Der „Demonstrator für maschinelles Lernen“ kann verwendet werden, um die Klassifizierung mithilfe des k-nächste-Nachbarn-Algorithmus im Anwendungsszenario der Sprachenerkennung von Texten zu demonstrieren. Bei Bedarf steht im Materialordner das Einführungsvideo `Einführung Sprachenerkennung Klassifizierung.mp4` zur Verfügung, in dem die Funktionsweise des Programms erklärt wird. Dazu werden im Ordner `daten_sprachen_klassifizierung` die Trainingsdaten gespeichert. Hierbei handelt es sich um `txt`-Dateien, die Texte in einer der Sprachen enthalten, die zur Klassifizierung herangezogen werden sollen. Der Dateibezeichner muss dabei das Label und, durch einen Unterstrich getrennt, einen Bezeichner enthalten, der später zur grafischen Veranschaulichung dient:

`Label_Bezeichner.txt`, z. B. `Deutsch_Internet.txt`

Der vorbereitete Datensatz enthält Trainingsdaten zu den Sprachen Deutsch, Englisch und Französisch; er kann bei Bedarf um weitere Dateien dieser Sprachen bzw. um Dateien anderer Sprachen ergänzt werden.

Wichtig: Die Klassifizierung mit dem „Demonstrator für maschinelles Lernen“ ist aus Gründen der grafischen Darstellung auf maximal fünf verschiedene Labels begrenzt.

Nach dem Start des Programms werden die Trainingsdaten anhand der beiden Merkmale *durchschnittliche Wortlänge* und *relative Vokalhäufigkeit* als Datenpunkte in einem zweidimensionalen Koordinatensystem dargestellt und je nach Label farblich hervorgehoben (s. Abbildung 4.19).

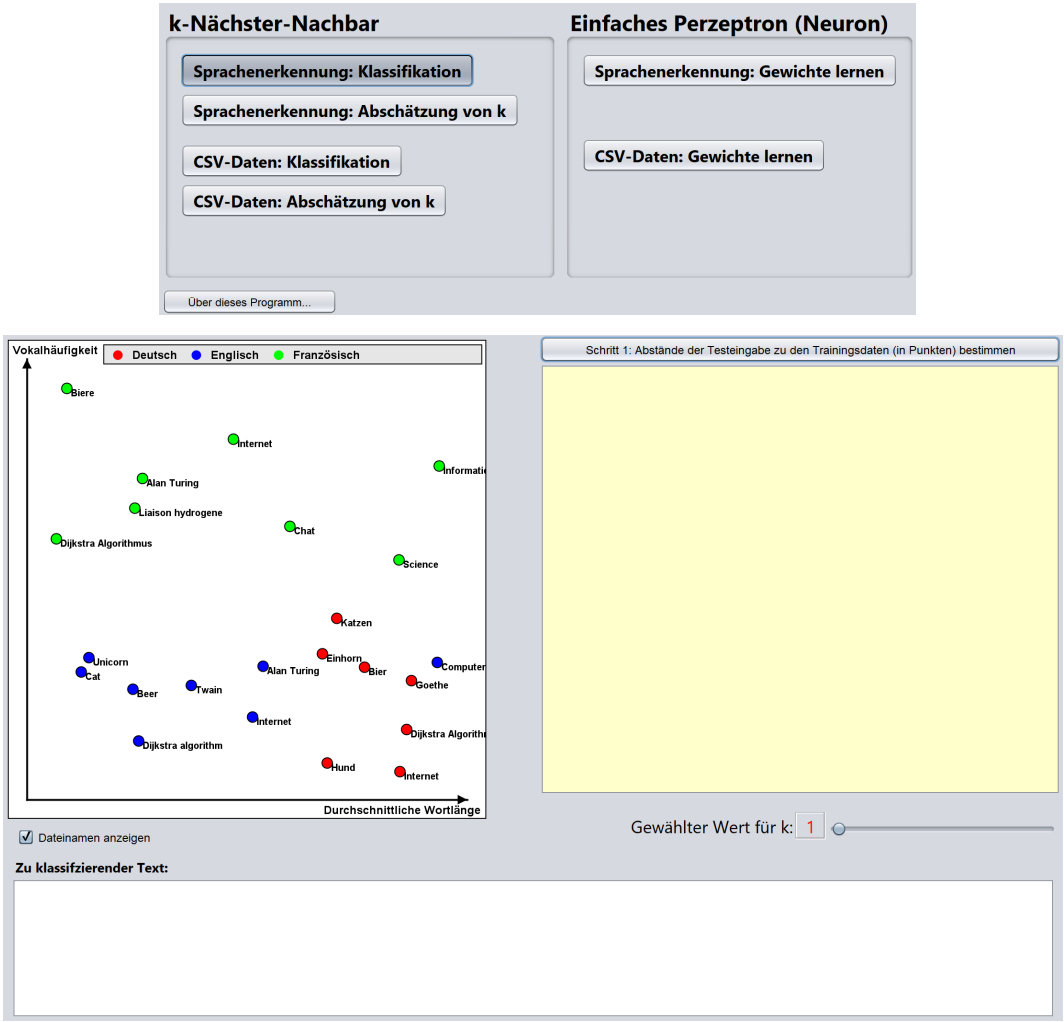


Abb. 4.19: Start des Demonstrators für maschinelles Lernen zur Klassifizierung.

Für die Darstellung werden die Daten normalisiert (siehe Abschnitt 4.1.4.1) und anschließend so umgerechnet, dass die größte Merkmalsausprägung dem Wert 450, die kleinste dem Wert 50 entspricht. Die Abstände der Datenpunkte können somit in Pixeln gemessen werden.

Im Textfeld *Zu klassifizierender Text* (s. Abbildung 4.19) kann nun ein Text zur Klassifizierung eingefügt werden. Damit das Merkmal *durchschnittliche Wortlänge* stabil berechnet werden kann, sollten längere Textabschnitte verwendet werden. Am besten eignen sich Passagen aus Online-Lexika, wie z. B. Wikipedia.

Wichtig: Bei ungünstigen Eingaben kann es vorkommen, dass der zugehörige Datenpunkt außerhalb des abgebildeten Ausschnitts des Koordinatensystems liegt.

Der zu klassifizierende Text wird als Datenpunkt im Koordinatensystem veranschaulicht. Nun können die Schritte des k-nächste-Nachbarn-Algorithmus schrittweise ausgeführt und nachvollzogen werden, wobei der Hyperparameter k manuell per Schieberegler (s. Abbildung 4.20, Pfeil 1) eingestellt wird. Dies wird im Koordinatensystem veranschaulicht (s. Abbildung 4.20, Pfeil 2). Das Protokollfenster gibt jeweils Auskunft über den zuletzt durchgeführten Arbeitsschritt (s. Abbildung 4.20, Pfeil 3). Nach dem letzten Schritt kann k über den Schieberegler variiert werden, um den Einfluss des Parameters auf das Klassifizierungsergebnis zu untersuchen (s. Abbildung 4.20).

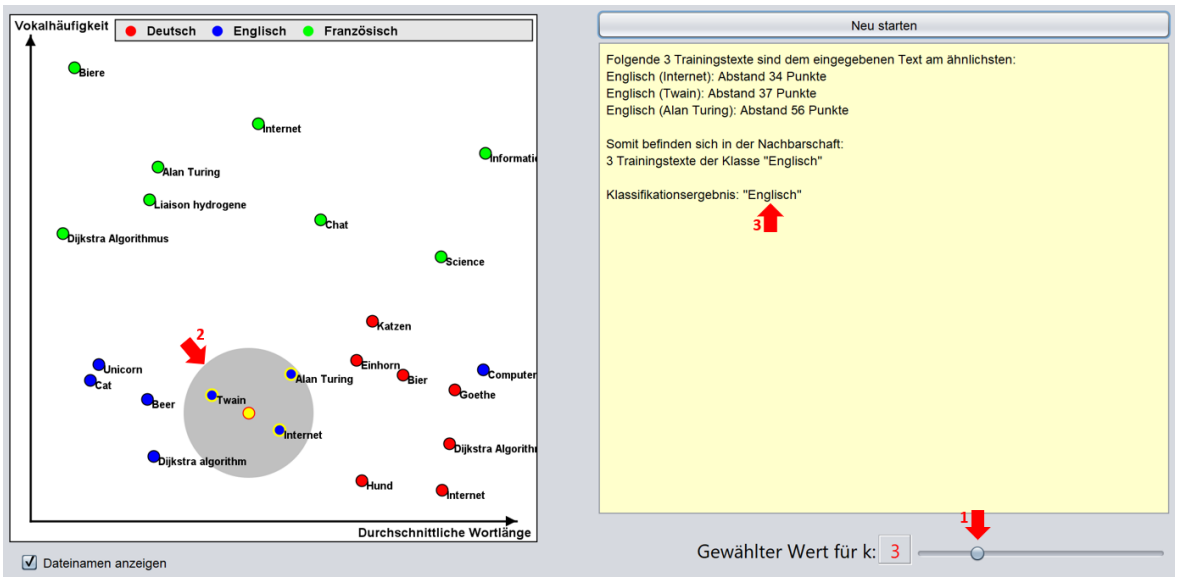


Abb. 4.20: Klassifizierung mithilfe des Demonstrators für maschinelles Lernen.

4.3.2.2 Sprachenerkennung: Abschätzung von k

Der „Demonstrator für maschinelles Lernen“ kann außerdem zur Bestimmung des optimalen Werts für den Hyperparameter k mithilfe des Leave-One-Out-Verfahrens (s. Abschnitt 4.1.5.3.2) verwendet werden. Die benötigten gelabelten Daten entsprechen in Aufbau und Bezeichnung den zur Klassifizierung verwendeten Dateien (s. Abschnitt 4.3.2.1) und werden im Ordner `daten_sprachen_k_abschaetzung` gespeichert. Der vorbereitete Datensatz enthält wiederum Daten zu den Sprachen Deutsch, Englisch und Französisch und kann um weitere Dateien

ergänzt werden.

Wichtig: Wie bereits in Abschnitt 4.3.1.1 ist der „Demonstrator für maschinelles Lernen“ auch hier auf **maximal fünf verschiedene Label** begrenzt.

Nach dem Start des Programms werden zunächst alle Trainingsdaten grafisch dargestellt (s. Abbildung 4.21). Zur Demonstration des Leave-One-Out-Verfahrens stehen die drei Optionen *Mikroschritt* (Pfeil 1), *Einzelschritt* (Pfeil 2) und *Komplettdurchlauf* (Pfeil 3) zur Verfügung.

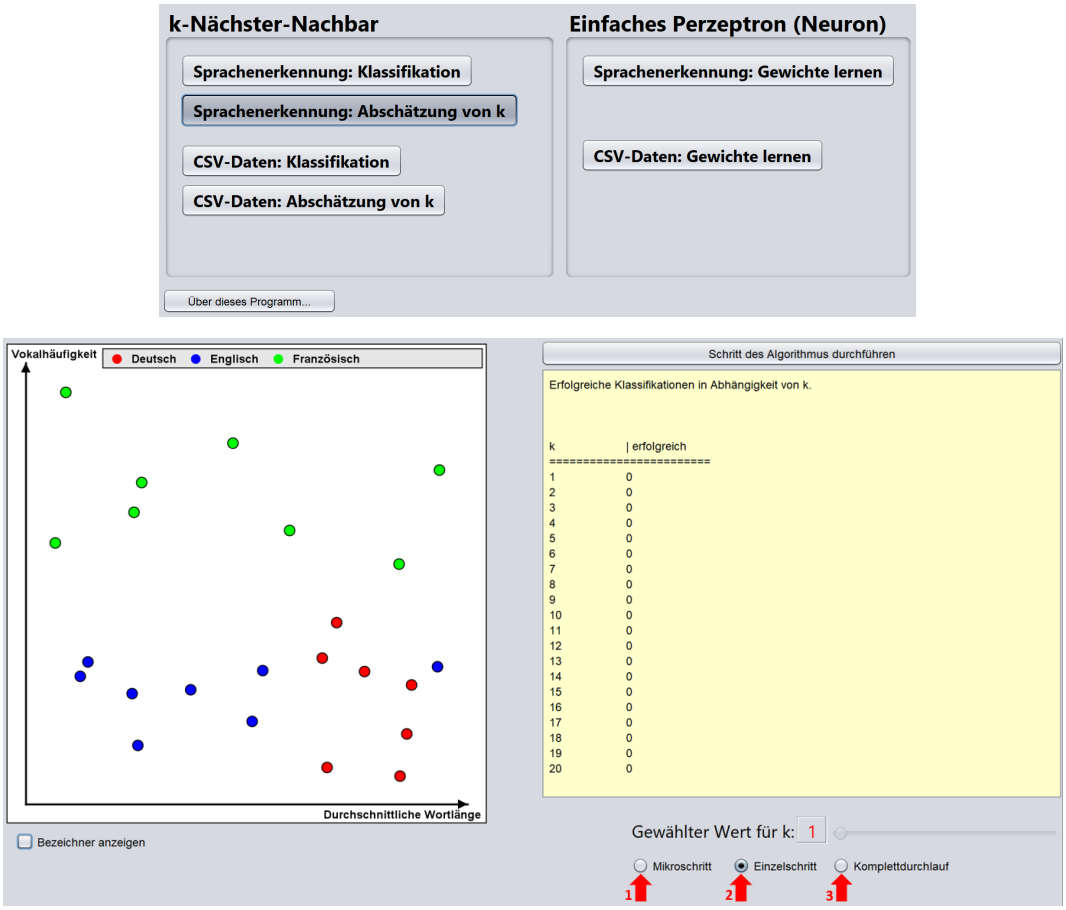


Abb. 4.21: Start des Demonstrators für maschinelles Lernen zur Abschätzung von k.

In der Variante *Mikroschritt* kann das Verfahren im Detail nachvollzogen werden. Dazu wird einer der Datenpunkte als Validierungsdatenpunkt gewählt und durch ein Quadrat hervorgehoben (s. Abbildung 4.22). Bei jedem Klick auf *Schritt des Algorithmus durchführen* (Pfeil 1) wird dieser Datenpunkt für den aktuellen k-Wert klassifiziert. Ist die Klassifizierung korrekt, wird die Kreisfläche um den Punkt grün hervorgehoben (Pfeil 2) und im Protokollfenster die

entsprechende Zahl in der Spalte „erfolgreich“ um 1 erhöht (Pfeil 3). Wird der Validierungsdatenpunkt falsch klassifiziert, wird die Kreisfläche rot markiert und der entsprechende Eintrag in der Tabelle bleibt unverändert. Anschließend wird k für den nächsten Mikroschritt um 1 erhöht (Pfeil 4).

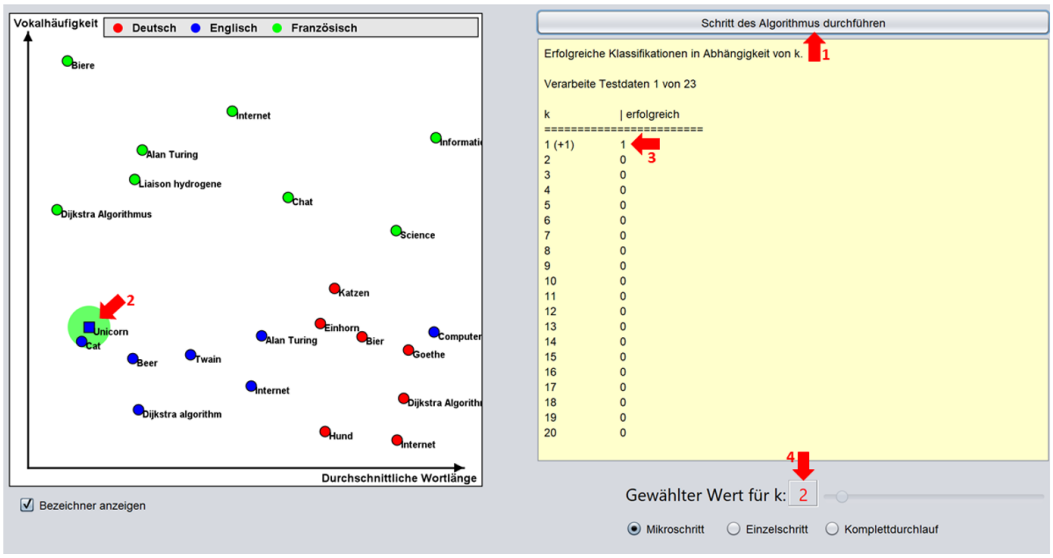


Abb. 4.22: Mikroschritt zur Abschätzung von k.

In der Variante *Einzelschritt* werden für jeden Validierungsdatenpunkt (s. Abbildung 4.23, Pfeil 1) die Mikroschritte für die k-Werte von 1 bis 20 auf einmal durchgeführt und protokolliert (Pfeil 2).

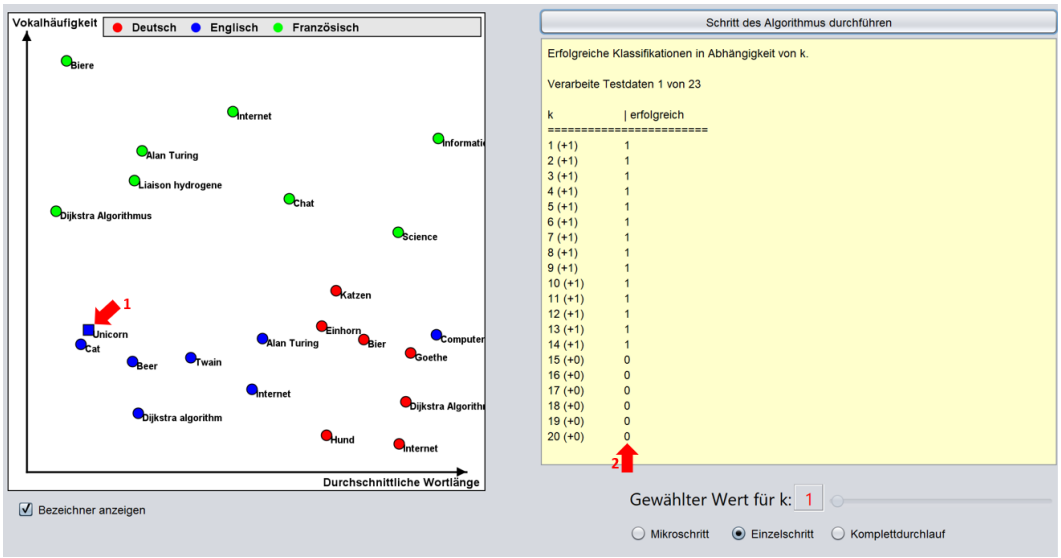


Abb. 4.23: Einzelschritt zur Abschätzung von k.

In der Variante *Komplettdurchlauf* wird sofort das Endergebnis des Leave-One-Out-Verfahrens angezeigt. Im Protokollfenster kann der für den verwendeten Datensatz „optimale“ Wert für k abgelesen werden (s. Pfeil in Abbildung 4.24).

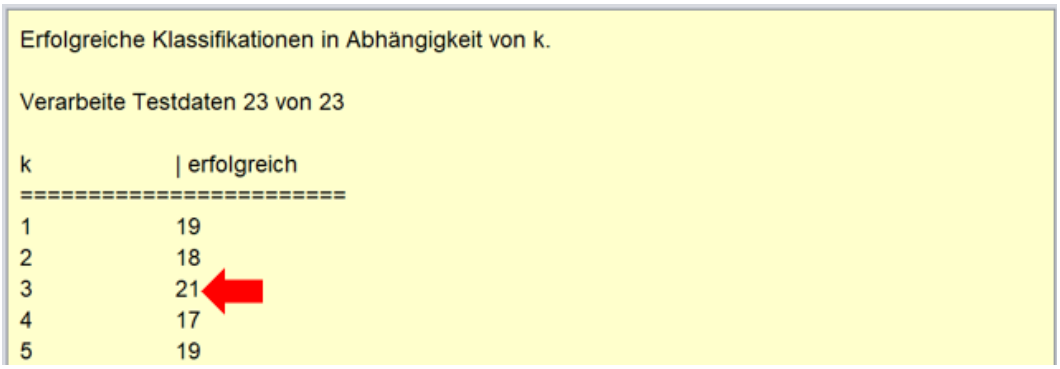


Abb. 4.24: Komplettdurchlauf zur Abschätzung von k.

4.3.2.3 Klassifikation und Abschätzung anhand von CSV-Dateien

Neben der Sprachenerkennung von Texten können mit dem „Demonstrator für maschinelles Lernen“ auch andere Anwendungsbeispiele veranschaulicht werden. Dazu müssen die gelabelten Daten in die Datei `daten.csv` eingepflegt werden. Der vorbereitete Datensatz enthält Daten zur Klassifizierung von Nahrungsmitteln anhand der Merkmale *Kalorien* und *Eiweiß*. Es werden die drei Labels *Früchte*, *Fisch / Fleisch* und *Backware* verwendet. Bei Bedarf kann die Datei jedoch an ein selbst gewähltes Szenario angepasst werden, indem die verwendeten Labels und Merkmalsbezeichner ersetzt werden (s. Abbildung 4.25).

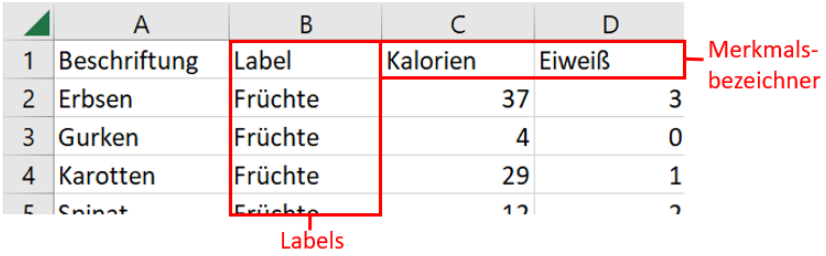


Abb. 4.25: Gelabelte Daten in `daten.csv`.

Wichtig:

- Auch hier ist der „Demonstrator für maschinelles Lernen“ auf **maximal fünf verschiedene Label** begrenzt.
- Soll die Datei `daten.csv` auch für das Perzeptron herangezogen werden, können lediglich zwei verschiedene Klassen verwendet werden.

Die Handhabung der Klassifizierung sowie der Abschätzung von k erfolgen analog zur Textklassifizierung (siehe Abschnitte 4.3.1.1 und 4.3.1.2).

4.3.3 Normalisierung und Abstände nicht-metrischer Daten

Für die in Abschnitt 4.2.2.4 vorgeschlagene Vorgehensweise enthält der Materialordner die Tabellenkalkulations-Datei `Normalisierung und Hamming-Distanz.xlsx` mit fünf Rechenblättern. Die Schülerinnen und Schüler erarbeiten hierbei zunächst die Min-Max-Normalisierung am Anwendungsszenario der Sprachenerkennung. Anschließend erfolgt die Einführung der Hamming-Distanz zur Bestimmung der Abstände von nicht-metrischen Daten.

4.3.4 Der k-nächste-Nachbarn-Algorithmus im `RAISE`-Playground

Der RAISE-Playground des MIT, der auf der Programmierumgebung Scratch basiert, bietet die Erweiterung *Text Classification* (s. Abbildung 4.26) an, in welcher der k-nächste-Nachbarn-Algorithmus zur Worterkennung eingesetzt werden kann.

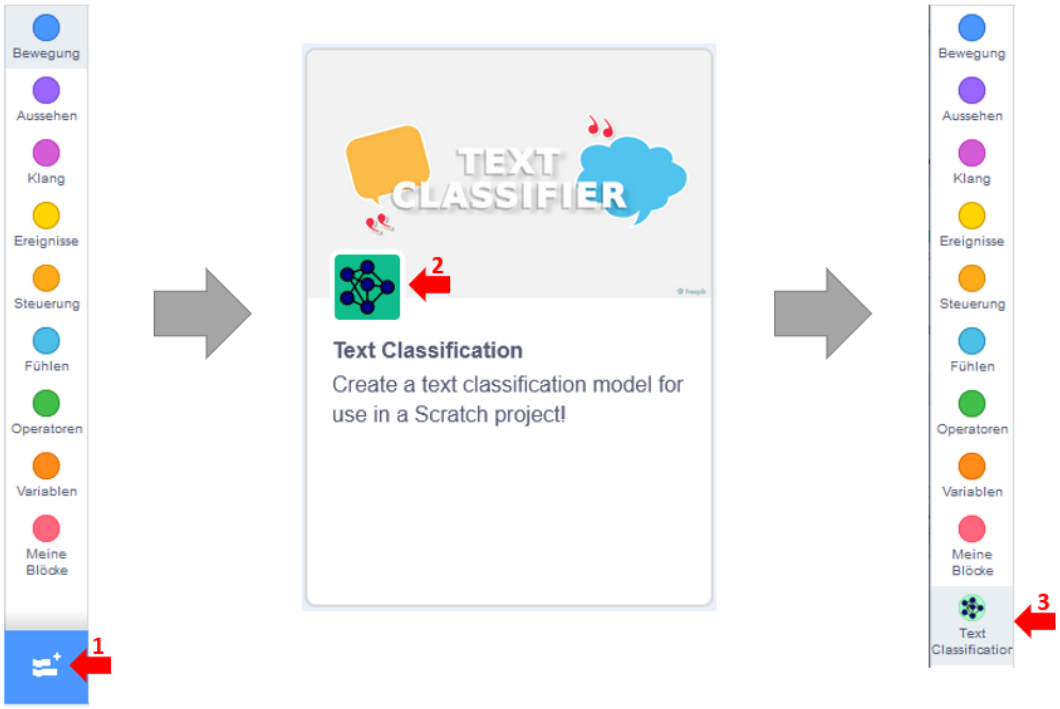


Abb. 4.26: Erweiterung *Text Classification* im MIT-RAISE-Playground.

Das Tool kann dazu verwendet werden, den k-nächste-Nachbarn-Algorithmus in einem selbst gewählten Anwendungsszenario praktisch einzusetzen. Über folgenden Link erhält man Zugriff auf den RAISE-Playground, der die *Text Classification* enthält:

`https://playground.raise.mit.edu/main/`

Bei Bedarf steht im Materialordner das Erklärvideo `Einführung RAISE Playground.mp4` bereit, in dem die Funktionsweise erläutert wird. Im Folgenden werden die wesentlichen Aspekte beschrieben.

4.3.4.1 Funktionsweise und Grenzen

Die *Text Classification* verwendet multidimensionale Wortvektoren, die aus vielen Merkmalen bestehen, deren Ausprägung und Bedeutung für den Menschen nicht intuitiv nachvollziehbar sind (Blackbox). Die Erzeugung der Vektoren erfolgt mithilfe statistischer Methoden bzw. durch ein künstliches neuronales Netz. Bei der Klassifizierung eines Wortes wird der Abstand des zugehörigen Datenpunkts zu den Trainingsdatenpunkten gemessen und das Wort mit dem k-nächste-Nachbarn-Algorithmus einer Klasse zugeordnet. Liegt beispielsweise das Wort *King* in der Nachbarschaft des Wortes *Queen*, so ist der Abstand im Merkmalsraum gering. Die Wahl des Hyperparameters k erfolgt automatisch und kann nicht vom Benutzer beeinflusst werden.

4.3.4.2 Die Erstellung des Modells

Die *Text Classification* bezieht sich auf ein Modell, das der Nutzer anlegen muss und das selbst gewählte Label und zugehörige Trainingsdaten umfasst. Ihre Eingabe erfolgt über die Schaltfläche *Edit Model* (s. Abbildung 4.27, Pfeil 1). Ein bereits vorbereitetes Modell in Form einer `json`-Datei kann über *Load / Save Model* hochgeladen werden (s. Abbildung 4.27, Pfeil 2).

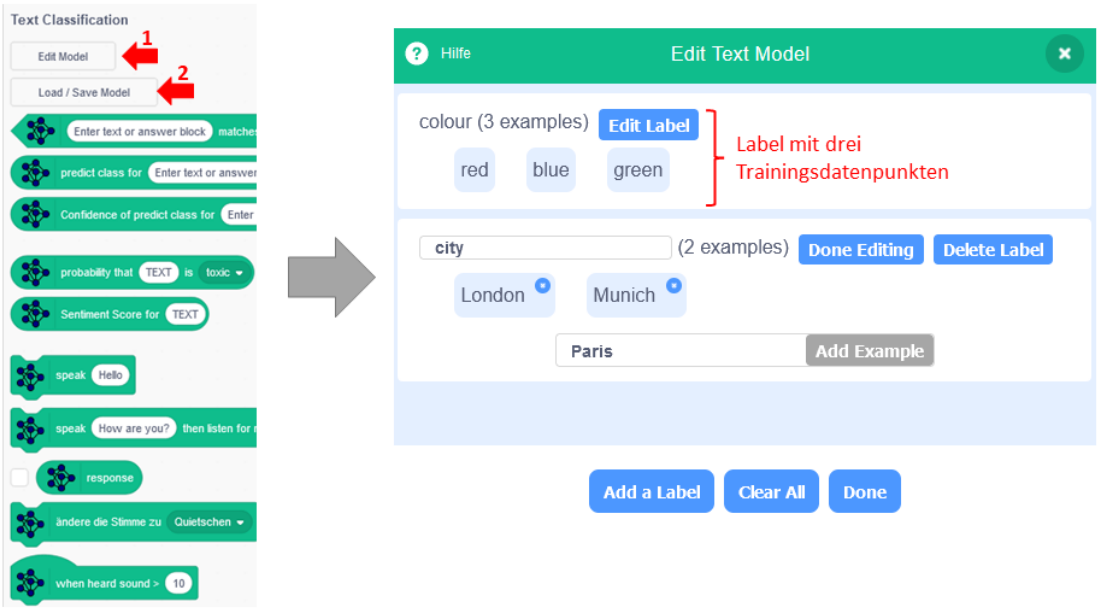
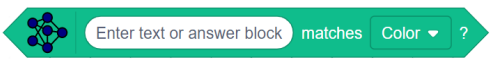


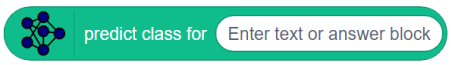
Abb. 4.27: Erstellung des Modells im RAISE-Playground.

4.3.4.3 Die wichtigsten Bausteine der Text Classification

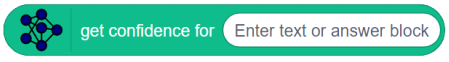
Zur Anwendung des k-nächste-Nachbarn-Algorithmus können mehrere Bausteine verwendet werden:



Gibt `true` zurück, wenn der Algorithmus den `text` dem Label `color` zuordnet; sonst `false`.



Gibt das Label zurück, das dem `text` zugeordnet wird.



Gibt einen Wert zwischen 0 und 1 zurück, der Auskunft darüber gibt, wie „sicher“ sich der Algorithmus bei der Klassifizierung von `text` ist.

Abb. 4.28: Bausteine des MIT-RAISE-Playgrounds.

Darüber hinaus stehen weitere Bausteine zur Verfügung, die insbesondere in Kombination mit anderen Erweiterungen eingesetzt werden können.

4.3.4.4 Hilfestellung zur Verbesserung der Zuverlässigkeit

Die Zuverlässigkeit der Klassifizierung hängt einerseits vom erstellten Modell, andererseits von der Implementierung ab. Unabhängig vom gewählten Anwendungsszenario sollten folgende Aspekte beachtet werden:

- Je mehr Trainingsdatenpunkte verwendet werden, desto zuverlässiger funktioniert die Klassifikation. Jedem Label sollten mindestens fünf Datenpunkte zugeordnet werden.
- Die Klassifizierungsergebnisse sind zuverlässiger, wenn die Sprache Englisch verwendet wird.
- Es sollten möglichst trennscharfe Label verwendet werden. Bei den Klassen Stadt und Land kann es beispielsweise zu Fehlern bei der Zuordnung von Stadtstaaten, wie etwa Singapur kommen.
- Durch eine geschickte Konstruktion des Algorithmus können ungenaue Klassifizierungen identifiziert und somit Fehler umgangen werden. Beispielsweise kann der Baustein `Confidence of predict class for` im Rahmen von bedingten Anweisungen verwendet werden, um bei „unsicheren“ Klassifizierungen weitere Benutzereingaben einzufordern.

Bei Bedarf stehen mit `Chatbot Einführung.mp4` und `Chatbot Verbesserung.mp4` Erklärvideos zur Verfügung, in denen die grundlegende Idee zur Erstellung eines Chatbots sowie Hinweise zur Verbesserung der Zuverlässigkeit erläutert werden.

4.3.5 Der k-nächste-Nachbarn-Algorithmus mit Tabellenkalkulation

Im Ordner „k-nächste-Nachbarn-Algorithmus mit Tabellenkalkulation“ stehen mehrere Dateien zur Verfügung, mit denen die Erarbeitung des k-nächste-Nachbarn-Algorithmus unter Verwendung einer Tabellenkalkulation möglich ist.

4.3.5.1 Klassifikation

Mithilfe der Datei `Klassifikation T-Shirts` können die Schülerinnen und Schüler den *k*-nächste-Nachbarn-Algorithmus zur Klassifikation von T-Shirt-Größen anhand der Merkmale *Körpergröße* und *Brustumfang* erarbeiten. In der Datei `KNN Klassifizierung mit Tabellenkalkulation.pdf` finden sich die zugehörigen Arbeitsaufträge.


Künstliche Intelligenz 11 *k*-nächste-Nachbarn-Algorithmus

Der *k*-nächste-Nachbarn-Algorithmus

Benötigtes Material:

- `k_naechster_Nachbar_Klassifikation_Datensatz_T_Shirt` und `Trainingsdaten_KNN`

Anwendung (Klassifikation von Schulshirtgrößen)

Am KI Gymnaisum(KIG) wurden von Herrn Turing Schulshirts für seine 11. Jahrgangsstufe entworfen. Damit nicht so viele Größen (S,M,L,XL,XXL,XXXL) produziert werden müssen, hat Herr Turing die Größen neu eingeteilt und es gibt nur die Größen S, M und L, welche alle oberen Größen abdecken.

*Herr Turing hat aber leider das KIG verlassen und dabei vergessen seine Einteilung in die neuen Größen S, M und L zu hinterlassen. Lediglich die Daten seiner 11 Jahrgangsstufe sind noch vorhanden. Es wurde dabei die **Körpergröße**, der **Brustumfang** und die daraufhin gekaufte **Shirt-Größe** notiert.*

Arbeitsauftrag 1: Ergänzen der Abstandsberechnung

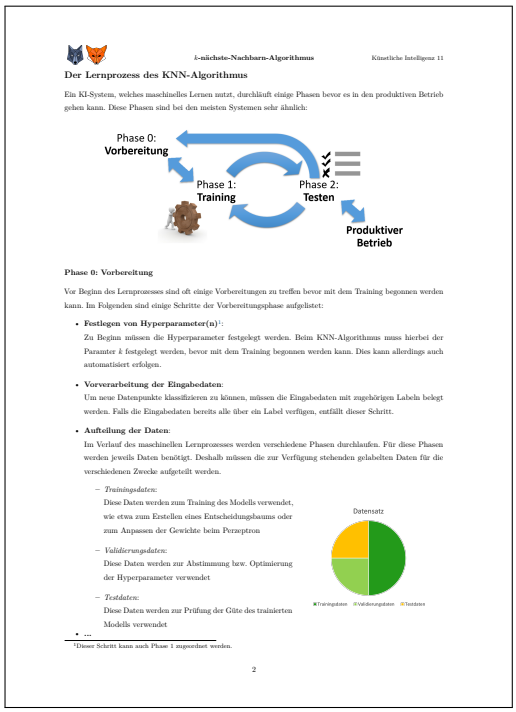
(a) Öffnen Sie den Ordner `KNN_TKS`.

(b) Öffnen Sie die Mappe `k_naechster_Nachbar_Klassifikation_Datensatz_T_Shirt` und navigieren Sie zur Tabelle `Klassifikation`.

(c) Die Spalte `Abstand` ist noch nicht gefüllt. Ergänzen Sie nacheinander passende Formeln zur Berechnung des Abstands mit der/dem

- Manhatten-Metrik
- Euklidischen Abstand

1


k-nächste-Nachbarn-Algorithmus Künstliche Intelligenz 11

Der Lernprozess des KNN-Algorithmus

Ein KI-System, welches maschinelles Lernen nutzt, durchläuft einige Phasen bevor es in den produktiven Betrieb gehen kann. Diese Phasen sind bei den meisten Systemen sehr ähnlich:

Phase 0: Vorbereitung
Phase 1: Training
Phase 2: Testen
Produktiver Betrieb

Phase 0: Vorbereitung

Vor Beginn des Lernprozesses sind oft einige Vorbereitungen zu treffen bevor mit dem Training begonnen werden kann. Im Folgenden sind einige Schritte der Vorbereitungsphase aufgelistet:

- **Festlegen von Hyperparameter(n)**[1]:
  Zu Beginn müssen die Hyperparameter festgelegt werden. Beim KNN-Algorithmus muss hierbei der Parameter *k* festgelegt werden, bevor mit dem Training begonnen werden kann. Dies kann allerdings auch automatisiert erfolgen.
- **Vorverarbeitung der Eingabedaten**:
  Um neue Datenpunkte klassifizieren zu können, müssen die Eingabedaten mit zugehörigen Labeln belegt werden. Falls die Eingabedaten bereits alle über ein Label verfügen, entfällt dieser Schritt.
- **Aufteilung der Daten**:
  Im Verlauf des maschinellen Lernprozesses werden verschiedene Phasen durchlaufen. Für diese Phasen werden jeweils Daten benötigt. Deshalb müssen die zur Verfügung stehenden gelabelten Daten für die verschiedenen Zwecke aufgeteilt werden.
  - *Trainingsdaten:*
    Diese Daten werden zum Training des Modells verwendet, wie etwa zum Erstellen eines Entscheidungsbaums oder zum Anpassen der Gewichte beim Perzeptron
  - *Validierungsdaten:*
    Diese Daten werden zur Abstimmung bzw. Optimierung der Hyperparameter verwendet
  - *Testdaten:*
    Diese Daten werden zur Prüfung der Güte des trainierten Modells verwendet
- ...

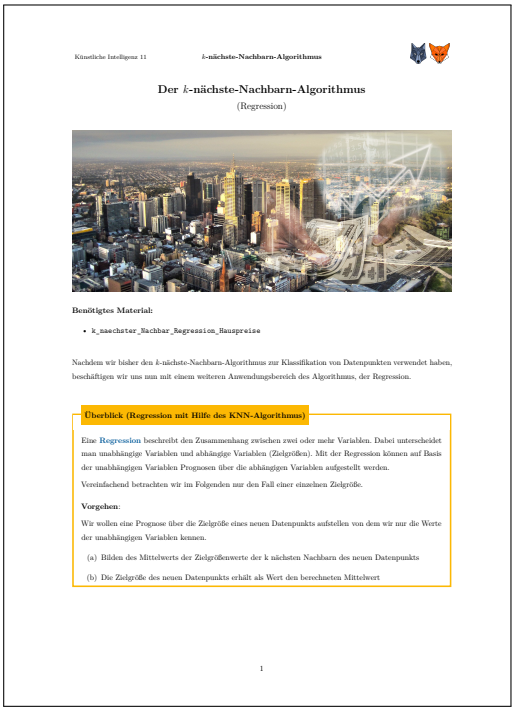
Datensatz
Trainingsdaten Validierungsdaten Testdaten

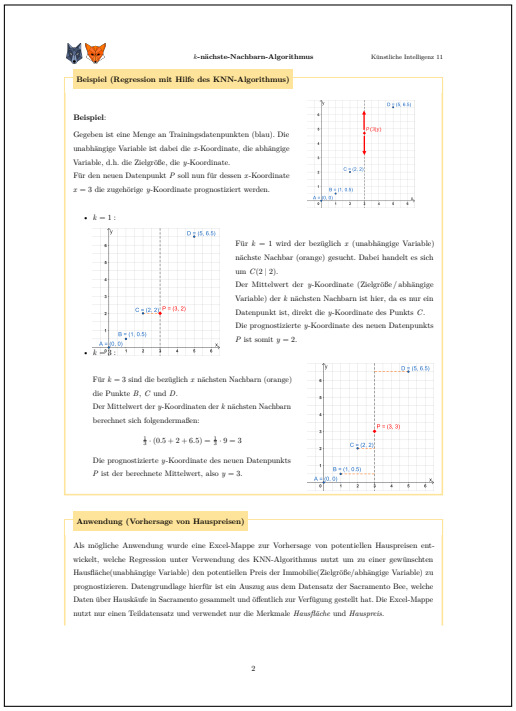
[1] Dieser Schritt kann auch Phase 1 zugeordnet werden.

2

4.3.5.2 Regression

Die Datei `Regression Hauspreise` kann verwendet werden, um die in Abschnitt 4.2.2.5 vorgeschlagene Vertiefung des *k*-nächste-Nachbarn-Algorithmus zu erarbeiten. Anhand des Merkmals *Fläche in* m^2 bestimmen die Schülerinnen und Schüler den Preis einer Immobilie durch Regression. In der Datei `KNN Regression mit Tabellenkalkulation.pdf` finden sich die zugehörigen Arbeitsaufträge.





4.3.6 Die Testphase

Für die in Abschnitt 4.2.2.6 vorgeschlagene Vorgehensweise steht im Materialordner die Tabellenkalkulations-Datei `Konfusionsmatrix Deutsch Englisch.xlsx` zur Verfügung. Die Testergebnisse der Schülerinnen und Schüler können in die dafür vorgesehenen Felder eingetragen werden. Die automatisch berechnete Genauigkeit ermöglicht eine Beurteilung der Zuverlässigkeit der Klassifizierung. Bei Bedarf können weitere Gütemaße berechnet werden.

4.3.7 Aufgabenbeispiel für einen Leistungsnachweis

4.3.7.1 Vorbemerkung

Im folgenden Vorschlag für einen Leistungsnachweis wird der „Demonstrator für maschinelles Lernen“ verwendet. Die `zip`-Datei `Demonstrator Leistungsnachweis` enthält das Tool zusammen mit der Datei `daten.csv`, die die Trainingsdaten enthält.

4.3.7.2 Leistungsnachweis

Ein Online-Shop für Bekleidung erhält immer wieder Rücksendungen von Kundinnen und Kunden, weil sie Kleidungsstücke in der falschen Größe bestellt haben. Um die Kundschaft zukünftig bei ihrer Wahl der Größe zu unterstützen, soll mithilfe Künstlicher Intelligenz eine Kleidergröße vorgeschlagen werden. In einem Pilotprojekt wird das System für T-Shirts für Männer erprobt, wobei zunächst nur die Größen (Label) S, M und L mit einbezogen werden. Dazu müssen die Kunden zunächst *Körpergröße* und *Brustumfang* (jeweils in cm) in eine Eingabemaske eingeben. Anschließend ordnet das System anhand von Erfahrungswerten zufriedener Kunden den eingegebenen Maßen eine der drei Größen zu. Dazu wird der k-nächste-Nachbarn-Algorithmus verwendet.

1. Öffnen Sie den Bereich `CSV-Daten: Klassifikation` des Demonstrators für maschinelles Lernen. Beurteilen Sie, inwieweit die vorliegenden Trainingsdaten für eine Klassifizierung mit dem k-nächste-Nachbarn-Algorithmus geeignet sind.

   Lösungsvorschlag:

   - Die Trainingsdaten für die Größe S bilden eine erkennbare und von den Datenpunkten anderer Größen abgegrenzte Nachbarschaft (positiv).
   - Die Trainingsdatenpunkte der Größen M und L sind teilweise vermischt bzw. haben Ausreißer (negativ).
   - Die Größe hat lediglich halb so viele Trainingsdatenpunkte wie die anderen beiden Größen (negativ).

2. Ein Kunde hat eine Körpergröße von 185 cm und einen Brustumfang von 105 cm. Bestimmen Sie das Klassifikationsergebnis für den Datenpunkt mithilfe des Demonstrators für maschinelles Lernen für $k = 4$ und $k = 5$ und bewerten Sie die Aussagekraft des Ergebnisses für den Kunden.

   Lösungsvorschlag:

   - $k = 4$: Jeweils zwei der vier nächsten Nachbarn haben die Label M bzw. L. Aufgrund des Gleichstands ist keine Klassifikation möglich.

- $k = 5$: Zwei der fünf nächsten Nachbarn haben das Label L, drei haben das Label M. Dem Kunden wird somit das Label M vorgeschlagen. Aufgrund der „knappen“ Entscheidung ist die Vorhersage jedoch mit Unsicherheit verbunden.

3. Begründen Sie, dass die gelabelten Daten vor der Verwendung des k-nächste-Nachbarn-Algorithmus grundsätzlich normalisiert werden sollten.

   Lösungsvorschlag:

   Sollten Merkmale unterschiedlich skaliert sein, kann es vorkommen, dass sie einen unterschiedlich großen Einfluss auf die Berechnung der Abstände zwischen Datenpunkten haben, was die Klassifizierung beeinträchtigen kann. Darüber hinaus müssen ggf. Merkmalsausprägungen unterschiedlicher Maßeinheiten addiert werden. Durch eine Normalisierung werden alle Merkmalsausprägungen auf das Intervall $[0; 1]$ abgebildet. Unterschiedliche Skalierungen der Merkmale nehmen somit keinen Einfluss mehr. Ebenso spielen Einheiten keine Rolle mehr, da es sich bei der normalisierten Größe um ein relatives Maß handelt.

4. Erläutern Sie das Leave-One-Out-Verfahren anhand eines selbstgewählten Datenpunktes und bestimmen Sie mithilfe des Demonstrators für maschinelles Lernen den optimalen Wert für k.

   Lösungsvorschlag:

   Für jeden Trainingsdatenpunkt wird überprüft, für welche Werte von k er korrekt klassifiziert wird. Beispielsweise gilt für Datenpunkt 6:

   - Für $k = 1$ korrekt klassifiziert.
   - Für $k = 2$ falsch klassifiziert.
   - Für $k = 3$ korrekt klassifiziert.
   - ...

KAPITEL 5 Das Perzeptron als Grundbaustein eines künstlichen neuronalen Netzes

„Imitation is the sincerest form of flattery."
Oscar Wilde

Überblick:

5.1 Fachliche Grundlagen

5.1.1 Überblick

Künstliche neuronale Netze sind ein wichtiger Teilbereich der Künstlichen Intelligenz. Sie kommen in sehr vielen Anwendungen zum Einsatz, die zunehmend Einfluss auf unser Leben ausüben.

In vielen Ländern hat sich der Einsatz von Software zur Gesichtserkennung bei der Strafverfolgung etabliert. Softwareunternehmen wie beispielsweise *Clearview* durchforsten dabei soziale Netzwerke und Webseiten, um umfangreiche künstliche neuronale Netze mit mehreren Milliarden Gesichtern zu trainieren. Allerdings häufen sich in der letzten Zeit die Negativschlagzeilen aufgrund von Missbrauch der Technik und datenschutzrechtlicher Verstöße (Beispiel: `www.heise.de/news/Aus-Musical-geworfen-Gesichtserkennung-entdeckt-Anwaeltin-gegnerischer-Kanzlei-7444612.html`).

Auch bei der Generierung von Bildern (Bildsynthese) spielen künstliche neuronale Netze eine immer wichtigere Rolle. Die Internetseite `https://thisxdoesnotexist.com/` bietet einen Dienst zur Erzeugung von Gesichtern und anderen Objekten, die nicht wirklich existieren. *Dall-E* und *Stable Diffusion* erlauben dies sogar anhand einer Textbeschreibung (siehe Abbildung 5.1).

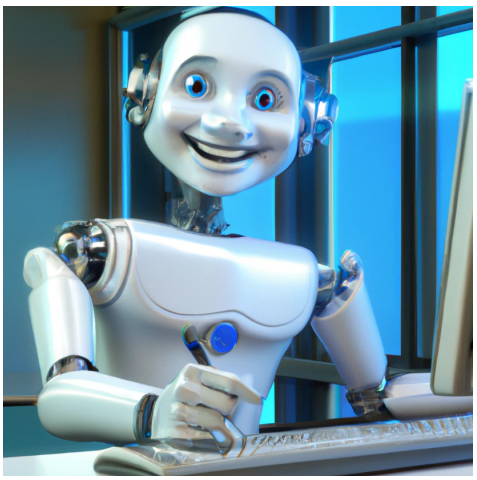


Abb. 5.1: Die Software *Dall-E* generierte dieses Bild mithilfe eines künstlichen neuronalen Netzes anhand der Beschreibung „A smiling robot programs a computer“.

Digitale Sprachassistenzsysteme wie Alexa, Siri oder Google Assistant findet man heutzutage in vielen Haushalten. Damit ein Programm Anweisungen von Menschen, wie beispielsweise „Mach die Stehlampe an“, immer besser verstehen kann, werden künstliche neuronale Netze eingesetzt.

Fast alle Verfahren (unter anderem auch der Entscheidungsbaum- und der k-nächste-Nachbarn-Algorithmus) der künstlichen Intelligenz haben das Ziel, kognitive Prozesse nachzubilden.

Generell setzen sie auf Mathematik, Programmiersprachen und Elektronik. Die künstlichen neuronalen Netze verfolgen dagegen einen bionischen Ansatz, das heißt, sie versuchen, das Gehirn selbst nachzuahmen (vgl. Ertel, 2021).

Der erste Abschnitt 5.1.2 beschäftigt sich mit den zugrunde liegenden Prozessen im Gehirn und der Interaktion von Nervenzellen. Danach wird in Abschnitt 5.1.3 gezeigt, wie eine einzelne Nervenzelle als künstliches Neuron (einfaches Perzeptron) modelliert wird und wie man damit eine Klassifizierung durchführen kann. Im Abschnitt 5.1.4 wird präsentiert, wie ein solches Perzeptron für zwei Eingaben mithilfe eines maschinellen Lernverfahrens trainiert wird. Im nächsten Abschnitt 5.1.5 wird gezeigt, wie dieses Verfahren auf Eingaben mit beliebig vielen Merkmalen angewendet werden kann. Ein Vorschlag zur Implementierung wird in Abschnitt 5.1.6 vorgestellt. Im letzten Abschnitt werden die Grenzen des einfachen Perzeptrons aufgezeigt und in einem Ausblick dargestellt, wie man diese Grenzen durch Bildung eines künstlichen neuronalen Netzes, das heißt einer Netzwerkstruktur aus einfachen Perzeptronen, verschiebt.

5.1.2 Die Natur als Vorbild

Sämtliche unserer bewussten und unterbewussten Entscheidungen werden vom Gehirn getroffen. Da ist es naheliegend, dass man sich zur Simulation von intelligentem Verhalten die Natur zum Vorbild nimmt. Wissenschaftler haben bereits im 19. Jahrhundert die Nervenzellen entdeckt und erforscht. Schon damals war bekannt, dass diese in Gehirnen hochgradig miteinander vernetzt sind. Mittlerweile weiß man, dass dieses Netzwerk beim Menschen aus knapp 100 Milliarden Nervenzellen, die auch als Neuronen bezeichnet werden, bestehen; jedes einzelne Neuron kann wiederum mehrere tausend Verknüpfungen besitzen.

Der Aufbau eines Neurons ist in Abbildung 5.2 dargestellt. An den Dendriten verbinden sich andere Nervenzellen, die elektrische Signale an das Neuron übertragen können. Die Verbindungen können dabei unterschiedlich effizient sein und somit einen mehr oder weniger starken Reiz auslösen. Wenn mehrere angeschlossene Nervenzellen gleichzeitig Signale über die Dendriten auf den Zellkörper übertragen, werden die Signale addiert und der Reiz damit verstärkt. Übersteigt dieser eine gewisse Schwelle, so transportiert das Neuron ein elektrisches Signal durch das Axon an andere damit verbundene Nervenzellen weiter. Man sagt auch, das Neuron „feuert“.

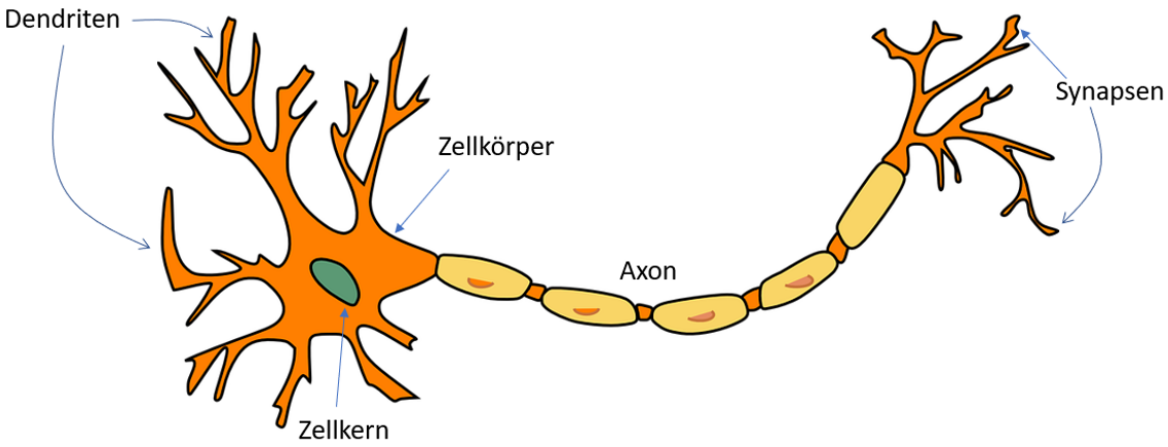


Abb. 5.2: Aufbau eines Nervenzelle.

Die Verbindungsstellen am Ende des Axons werden als Synapsen bezeichnet; sie haften an den Dendriten anderer Nervenzellen.

Obwohl die Funktionsweise einer einzelnen Nervenzelle gut verstanden wird, bleibt die Arbeitsweise des gesamten menschlichen Gehirns weitgehend ein Rätsel. Einige Gründe dafür sind die enorme Komplexität des Netzwerks, das ständigen Veränderungen unterliegt, beispielsweise durch Lernprozesse. Hinzu kommen offensichtliche Grenzen bei der Untersuchung des lebenden Gehirns.

5.1.3 Informatische Modellierung eines Neurons

Der im letzten Abschnitt dargestellte Grundaufbau einer Nervenzelle wird nun mit den Mitteln der Informatik als einfaches Perzeptron nachgebildet (Abbildung 5.3):

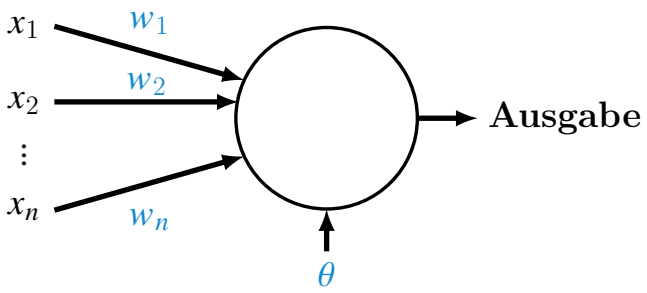


Abb. 5.3: Modell eines künstlichen Neurons.

Das künstliche Neuron hat dabei folgende Bestandteile:

- Bei x_1 bis x_n handelt es sich um die **Eingabewerte**. Dies entspricht der Signalübertragung an den Dendriten einer Nervenzelle.
- Die Parameter w_1 bis w_n stellen sogenannte **Gewichte** dar. Sie drücken aus, wie stark der Einfluss der jeweils zugehörigen Eingabe auf die Entscheidung über ein Feuern des Neurons ist. Mathematisch wird dies so umgesetzt, dass alle Eingaben x_i jeweils mit ihren Gewichten w_i multipliziert werden. Damit wird die unterschiedlich starke Übertragung von elektrischen Reizen bei einer Nervenzellenverbindung an den Synapsen nachgeahmt.
- Der Parameter θ wird als **Schwellenwert** bezeichnet. Wenn die Summe der gewichteten Eingangssignale den Schwellenwert erreicht oder überschreitet, so bewirkt dies ein Feuern des Neurons, wenn folgende Ungleichung erfüllt ist:

$$w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n \geq \theta\,.$$

- Eine gängige Möglichkeit ist, dass man als **Ausgabe** den Wert 1 festlegt, wenn das Neuron feuert, und den Wert 0, wenn das Neuron nicht feuert.

Im Folgenden ist ein Beispiel für ein Perzeptron mit Gewichten und Schwellenwert abgebildet:

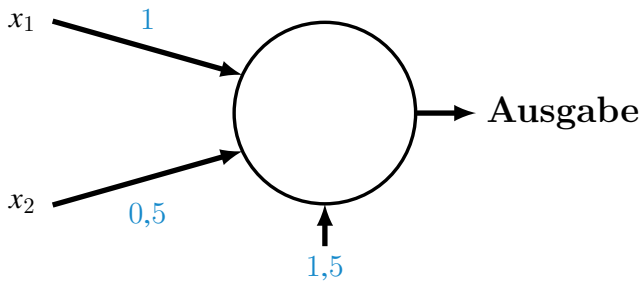


Abb. 5.4: Beispiel für ein einfaches Perzeptron mit zwei Eingaben.

Die blau dargestellten Werte für Gewichte und Schwellenwert sind individuell für das jeweilige Perzeptron und können mit einem maschinellen Lernverfahren, das im Abschnitt 5.1.4.2 thematisiert wird, automatisiert ermittelt werden, falls dies möglich ist.

5.1.4 Das Perzeptron im zweidimensionalen Raum

5.1.4.1 Klassifizierung

Die Bedingung für ein Feuern für das oben abgebildete Perzeptron

$$1 \cdot x_1 + 0{,}5 \cdot x_2 \geq 1{,}5$$

lässt sich umformen zu $x_2 \geq -2x_1 + 3$. Der „Gleichheitsfall“ $x_2 = -2x_1 + 3$ entspricht der Gleichung einer Geraden, die den zweidimensionalen Raum linear in zwei Halbräume separiert (siehe Abbildung 5.5).

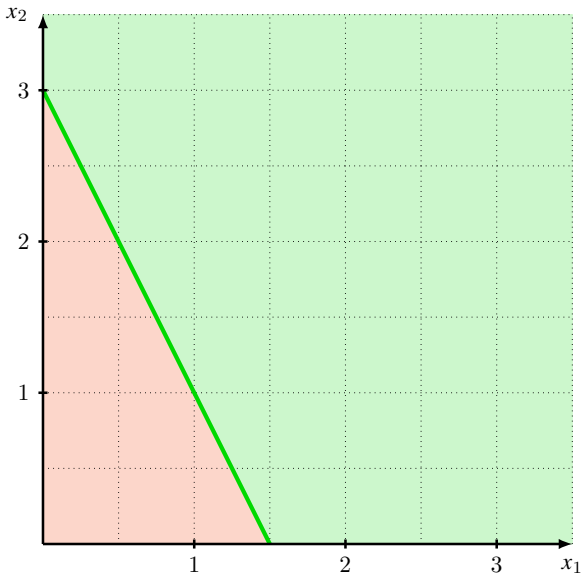


Abb. 5.5: Lineare Separierung des Raums. Im grünen Bereich feuert das künstliche Neuron, im roten Bereich nicht.

Liegt für dieses Beispiel eine Eingabe $(x_1 \mid x_2)$ auf bzw. oberhalb der Geraden, das heißt im grün markierten Bereich, feuert das Perzeptron, liegt sie unterhalb, also im rot markierten Bereich, feuert das Neuron nicht.

Das heißt, das Perzeptron klassifiziert eine Eingabe dadurch, dass es sie grün (entspricht einem Feuern) bzw. rot labelt, je nachdem, in welchem Halbraum die Eingabe liegt.

Beispiel: Es soll für die beiden Eingaben $A(0{,}5 \mid 1)$ und $B(3 \mid 3)$ berechnet werden, ob das Perzeptron aus Abbildung 5.5 feuert:

$$1 \cdot 0{,}5 + 0{,}5 \cdot 1 \geq 1{,}5 \Longleftrightarrow 1 \geq 1{,}5$$

Die Aussage ist falsch, deshalb feuert das Perzeptron nicht und labelt den Eingabepunkt A mit 0 (rot).

$$1 \cdot 3 + 0{,}5 \cdot 3 \geq 1{,}5 \iff 4{,}5 \geq 1{,}5$$

Die Aussage ist wahr, deshalb feuert das Perzeptron und labelt den Eingabepunkt B mit 1 (grün).

Auch ein Blick auf folgende Abbildung bestätigt, dass sich der Eingabepunkt A im roten und Eingabepunkt B im grünen Halbraum befindet.

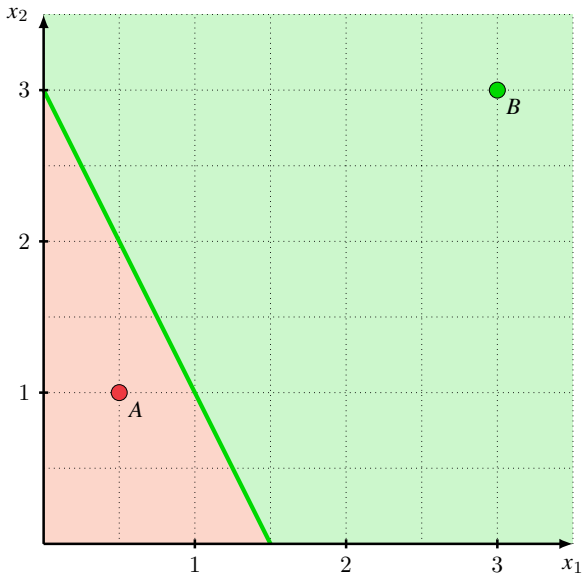


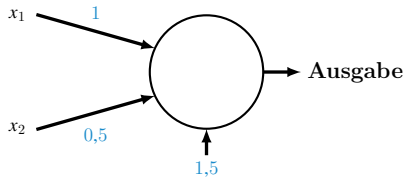
Abb. 5.6: Datenpunkt A wird mit 0 (rot) und Datenpunkt B mit 1 (grün) gelabelt.

5.1.4.2 Trainingsdaten und Delta-Lernregel

Damit das Perzeptron zuverlässig klassifizieren kann, muss es vorab trainiert werden. Dazu werden gelabelte Daten als Trainingsdaten benötigt. Zunächst werden die Gewichte und der Schwellenwert durch zufällige Werte vorbelegt. Das Perzeptron berechnet dann für einen Trainingsdatenpunkt die Ausgabe (Label) und vergleicht diese mit dem tatsächlichen Label.

Folgende Abbildung (5.7) zeigt vier Trainingsdatenpunkte, von denen gemäß der aktuellen Konfiguration des Perzeptrons zwei richtig und zwei falsch klassifiziert werden würden.

Daten-punkt	x_1	x_2	Label
A	0,5	1	0 (rot)
B	3	3	1 (grün)
C	0,25	2	1 (grün)
D	1,5	1	0 (rot)



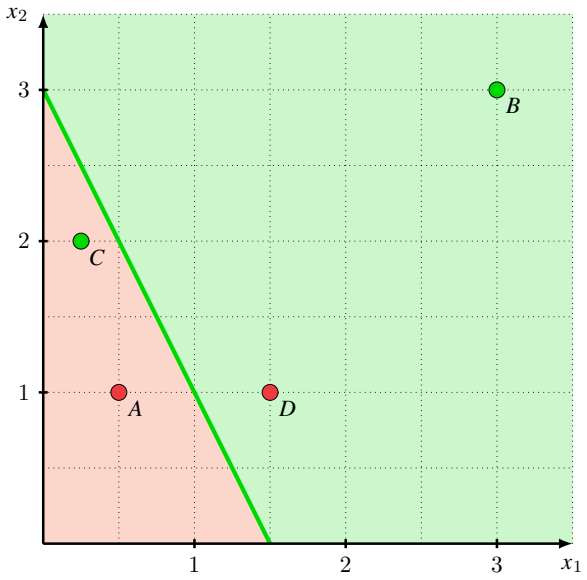


Abb. 5.7: Das künstliche Neuron klassifiziert zwei der vier Trainingsdaten falsch.

Weichen die Label voneinander ab, müssen Gewichte und Schwellenwert und damit die Lage der Geraden angepasst werden. Nachdem die Ausgabe des Perzeptrons (berechnetes Label) mit dem tatsächlichen (erwarteten) Label verglichen wurde, gibt es drei Fälle zu unterscheiden:

Fall 1: Berechnetes Label stimmt mit dem erwarteten Label überein

Das Perzeptron hat die Klassifizierung für diesen Datenpunkt richtig durchgeführt. Eine Anpassung der Gewichte und des Schwellenwerts ist somit nicht erforderlich. In Abbildung 5.7 trifft das auf die Datenpunkte A und B zu.

Fall 2: Berechnetes Label ist kleiner als das erwartete Label

Dieser Fall tritt im Beispiel bei Datenpunkt C auf, da das erwartete Label den Wert 1 besitzt, aber das Perzeptron als Ausgabe den Wert 0 liefert, denn die gewichtete Summe $1 \cdot 0{,}25 + 0{,}5 \cdot 2$ ist kleiner als der Schwellenwert 1,5. Um die Trennlinie der beiden Halbräume in Richtung des Datenpunktes C zu bewegen, werden die Gewichte w_1 und w_2 erhöht und der Schwellenwert θ verringert.

Fall 3: Berechnetes Label ist größer als das erwartete Label

In Abbildung 5.7 tritt dieser Fall bei Datenpunkt D auf, da die gewichtete Summe $1 \cdot 1{,}5 + 0{,}5 \cdot 1$ größer als der Schwellenwert 1,5 ist und somit das Perzeptron eine 1 ausgibt, obwohl der

Datenpunkt mit 0 gelabelt ist. Die Situation ist nun umgekehrt zu Fall 2, dementsprechend werden die Gewichte w_1 und w_2 verringert, dagegen der Schwellenwert θ erhöht.

Mithilfe dieser Fälle kann die Delta-Lernregel, ein Verfahren zur Anpassung der Gewichte und des Schwellenwerts, hergeleitet werden. Hierzu wird festgelegt, dass der Fehler der Klassifikation, dargestellt durch den Buchstaben Delta δ, durch

$$\delta = \texttt{Label}_{\texttt{erwartet}} - \texttt{Label}_{\texttt{berechnet}}$$

berechnet wird. Im Fall 1 hat δ den Wert 0, im Fall 2 ist $\delta = 1$ und im dritten Fall $\delta = -1$.

Nun erfolgen die Anpassungen der Gewichte beziehungsweise des Schwellenwerts anhand folgender Formeln:

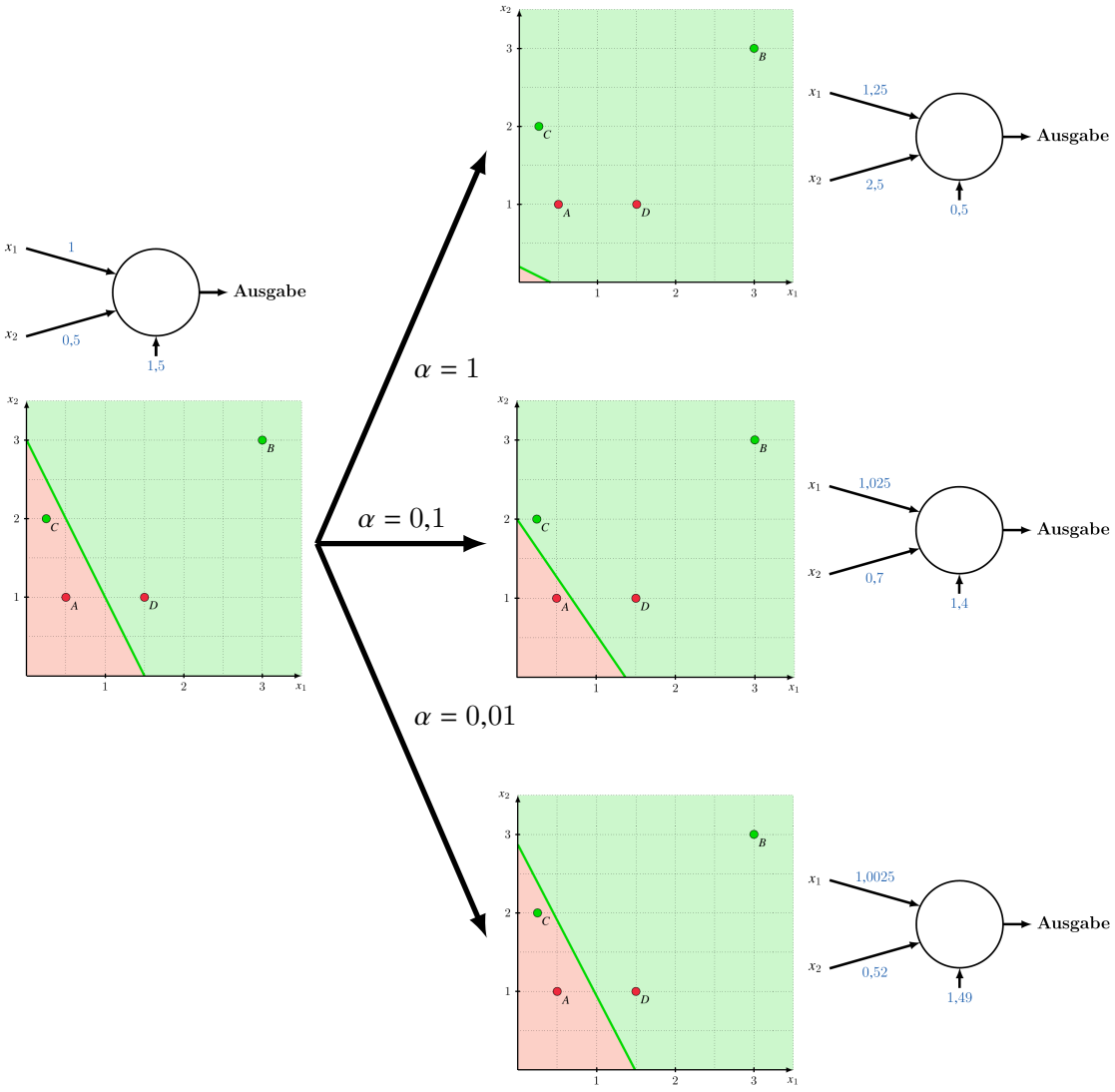
$$\begin{aligned} w_1' &= w_1 + \alpha \cdot \delta \cdot x_1 \\ w_2' &= w_2 + \alpha \cdot \delta \cdot x_2 \\ \theta' &= \theta - \alpha \cdot \delta \end{aligned} \tag{5.1}$$

Die Formeln stellen sicher, dass sich die separierende Gerade auf „schnellstem“ Weg in die richtige Richtung bewegt.

An dieser Stelle tritt der Hyperparameter $\alpha \in \mathbb{R}^+$ auf, der als Lernrate bezeichnet wird. Diese beschreibt, wie stark die Anpassung in Richtung des Datenpunkts erfolgen soll.

Abbildung 5.8 veranschaulicht den Einfluss der Lernrate auf die Anpassung der beiden Gewichte und des Schwellenwerts. Hierbei wurde ein Lernschritt der Delta-Lernregel mit Datenpunkt $C(0{,}25 \mid 2)$ für drei verschiedene Werte von α durchgeführt.

Man erkennt, dass ein großer Wert von α dazu führt, dass eine zu starke Anpassung in Richtung des Trainingspunkts stattfindet. Damit kann man gegebenenfalls bei der Anpassung der Gewichte und des Schwellenwerts über das Ziel hinausschießen. Wählt man die Lernrate zu klein, so wird die Trennlinie zwar in die richtige Richtung bewegt, aber es sind sehr viele Schritte nötig, bis eine passende Positionierung gefunden wird. Im obigen Beispiel scheint eine Lernrate von 0,1 gut geeignet zu sein, allerdings lässt sich das nicht pauschal auf andere Szenarien übertragen. In der Praxis hat es sich als vorteilhaft erwiesen, mit großen Lernraten zu starten und den Wert sukzessive zu verkleinern. Es gibt noch eine Reihe weiterer Möglichkeiten der Optimierung, eine genauere Betrachtung würde aber den Rahmen der Handreichung sprengen.



Beispiel: Berechnungen für Lernrate $\alpha = 0{,}1$

Perzeptron vor dem Trainingsschritt: $w_1 = 1; w_2 = 0{,}5; \theta = 1{,}5$

Trainingsdatenpunkt C: $x_1 = 0{,}25$; $x_2 = 2$; $\texttt{Label}_{\texttt{erwartet}} = 1$; $\texttt{Label}_{\texttt{berechnet}} = 0$

$\delta = \texttt{Label}_{\texttt{erwartet}} - \texttt{Label}_{\texttt{berechnet}} = 1 - 0 = 1$

$w_1' = w_1 + \alpha \cdot \delta \cdot x_1 = 1 + 0{,}1 \cdot 1 \cdot 0{,}25 = 1{,}025$

$w_2' = w_2 + \alpha \cdot \delta \cdot x_2 = 0{,}5 + 0{,}1 \cdot 1 \cdot 2 = 0{,}7$

$\theta' = \theta - \alpha \cdot \delta = 1{,}5 - 0{,}1 \cdot 1 = 1{,}4$

Abb. 5.8: Einfluss der Lernrate auf die Anpassung der Gewichte und des Schwellenwerts nach einem Schritt der Delta-Lernregel.

5.1.4.3 Algorithmus zur Delta-Lernregel

Die bereits formulierte Delta-Lernregel wird folgendermaßen auf die Menge der Trainingsdaten angewendet:

> Belege Gewichte und Schwellenwert mit zufälligen Werten vor
>
> Wiederhole bis eine Abbruchbedingung erfüllt ist
>
> Wiederhole für alle Trainingsdatenpunkte
>
> Bestimme mit dem Perzeptron $\texttt{Label}_{\texttt{berechnet}}$ des aktuellen Datenpunkts
>
> Berechne $\delta = \texttt{Label}_{\texttt{erwartet}} - \texttt{Label}_{\texttt{berechnet}}$
>
> Passe die Gewichte und den Schwellenwert gemäß den Formeln 5.1 an.
>
> Ende Wiederhole
>
> Ende Wiederhole

Ein Abbruchkriterium des Algorithmus ist, dass in einem kompletten Durchlauf alle Trainingsdaten bereits richtig klassifiziert sind und dadurch keine Anpassung der Gewichte und des Schwellenwerts mehr vorgenommen werden musste. Mit diesem Abbruchkriterium liefert der Algorithmus genau dann (Rosenblatt, 1960) ein sinnvolles Ergebnis, wenn die Trainingsdaten linear separierbar sind.

Falls die Trainingsdaten nicht linear separierbar sind (s. Abbildung 5.9), würde der Algorithmus nach obiger Abbruchbedingung nicht terminieren. Daher kann als weitere Abbruchbedingung beispielsweise auch eine maximale Zahl an Iterationen festgelegt werden. Je nach Trainingsdaten kann dennoch eine zufriedenstellende Lösung erreicht werden (s. Abbildung 5.9, links), allerdings gibt es auch Fälle (s. Abbildung 5.9, mittig und rechts), in denen ein Perzeptron keine brauchbare Separierung ermitteln kann.

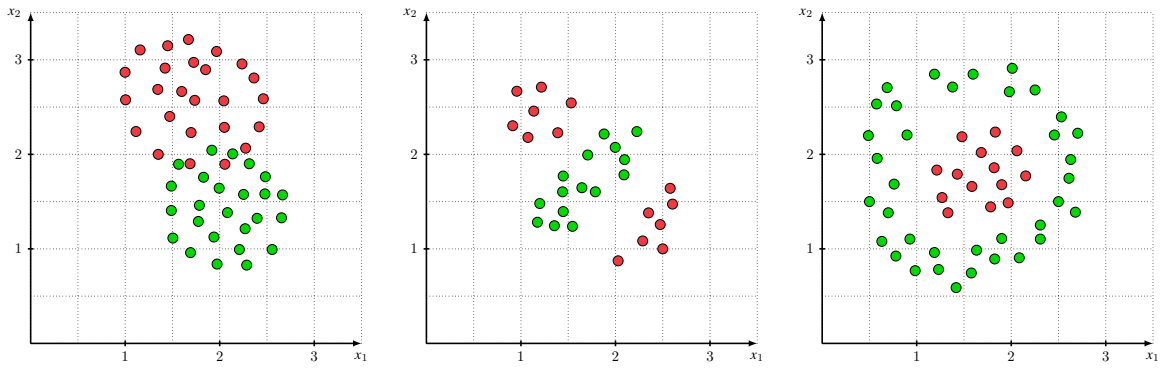


Abb. 5.9: Drei Beispiele für Daten, die nicht linear separierbar sind.

5.1.5 Das Perzeptron im n-dimensionalem Raum

Die Ungleichung $w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n \geq \theta$ lässt sich umformen zu

$$\underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{=:\vec{x}} \circ \underbrace{\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}}_{=:\vec{w}} - \theta \geq 0\,.$$

Das Perzeptron würde feuern, wenn der Eingabevektor $\vec{x}$ im entsprechenden Halbraum liegt, in den der Vektor $\vec{w}$ zeigt.

Im Folgenden soll nun ein Perzeptron als Funktion $f : \mathbb{R}^n \to \{0; 1\}$ mit $\vec{x} \in \mathbb{R}^n$ als Eingabe modelliert werden. Man zerlegt die Berechnung der Ausgabe in zwei Teilschritte:

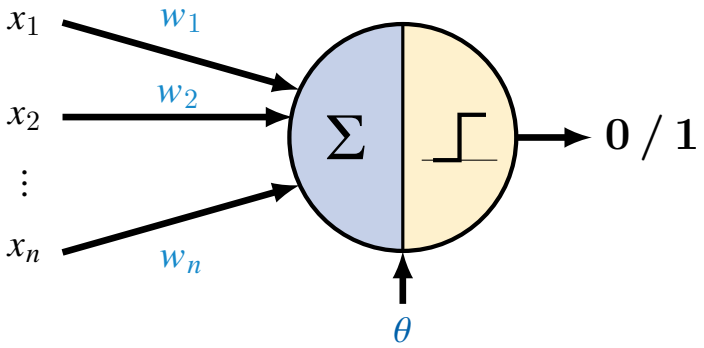


Abb. 5.10: Das Perzeptron verarbeitet die Eingaben in zwei Schritten: Gewichtete Summe bilden (links) und Aktivierung berechnen (rechts).

Der erste Schritt (Abbildung 5.10, blaue Einfärbung links) ist die Berechnung der Differenz aus der gewichteten Summe anhand der Eingaben und des Schwellenwerts. Diese Berechnung erfolgt mithilfe der **Übertragungsfunktion**:

$$f_{\text{Übertragung}}(x_1, x_2, \ldots, x_n) = w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n - \theta$$

Je nachdem, ob mit der gewichteten Summe der Schwellenwert unterschritten wurde, ist der Wert der Übertragungsfunktion kleiner als 0, anderenfalls größer gleich 0. Da das Perzeptron nur den Wert 0 oder 1 ausgeben soll, wird in einem zweiten Schritt (Abbildung 5.10, gelbe Einfärbung, rechts) die Ausgabe a der Übertragungsfunktion mit einer weiteren Funktion verarbeitet:

$$f_{\text{Heaviside}}(a) = \begin{cases} 1 & \text{falls } a \geq 0 \\ 0 & \text{falls } a < 0 \end{cases}$$

Diese Funktion wird als Heaviside-Funktion bezeichnet. Sie ist eine **Aktivierungsfunktion**, denn sie entscheidet anhand der gewichteten Summe und des Schwellenwerts, ob das künstliche Neuron „aktiviert“ wird, also feuert.

Die Funktion $f : \mathbb{R}^n \to \{0; 1\}$ des Perzeptrons lässt sich damit als Verkettung der beiden Funktionen $f_{\text{Übertragung}}$ und $f_{\text{Heaviside}}$ schreiben:

$$f(x_1, x_2, \ldots, x_n) = f_{\text{Heaviside}}\left(f_{\text{Übertragung}}(x_1, x_2, \ldots, x_n)\right)$$

Exkurs (Weitere Aktivierungsfunktionen)

Der Wertebereich der Heaviside-Funktion ist binär und somit diskret. Das mag zwar den natürlichen Prozessen der Nervenzelle recht nahekommen, besitzt aber Nachteile gegenüber einer reellwertigen Ausgabe:

- Die Bestimmung der Gewichte und des Schwellenwerts wird bei künstlichen neuronalen Netzen, deren Grundbaustein das einfache Perzeptron ist, mithilfe eines Optimierungsalgorithmus gelöst; dieser Algorithmus basiert meist auf einem Gradientenabstieg[1]. Solche Verfahren benötigen als Aktivierungsfunktion eine stetig differenzierbare Funktion, um effizient arbeiten zu können.

[1] Der Backpropagation-Algorithmus ist ein solches Verfahren. Er ist im Lehrplan der 13. Jahrgangsstufe verankert.

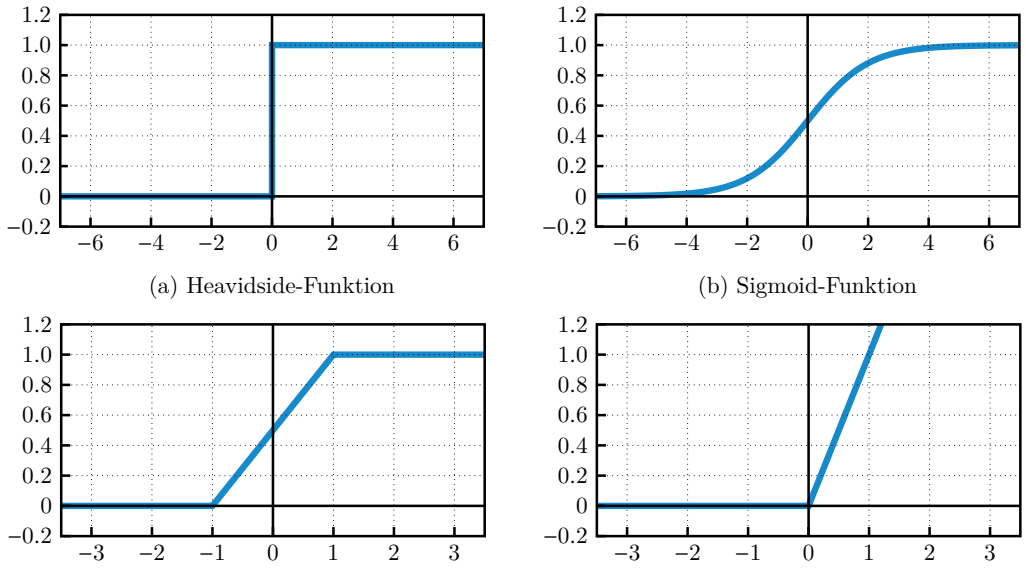
- Künstliche neuronale Netze eignen sich grundsätzlich auch zur Berechnung einer Regression. Durch eine Beschränkung auf die Werte 0 und 1 in der Ausgabe ist das nicht möglich, da in der Regel Funktionen approximiert werden, die in einen kontinuierlichen Werteraum abbilden.

- Das Ergebnis der Klassifizierung eines Datenpunkts, der sich recht nahe oder sogar auf der Trennlinie beziehungsweise Trennebene befindet, ist mit deutlich höherer Unsicherheit behaftet als ein weiter entfernter. Eine Ausgabe nahe 0,5 würde bspw. signalisieren, dass das Perzeptron keine sichere Aussage treffen kann.

Aus diesen und weiteren Gründen werden in der Praxis vorzugsweise kontinuierliche Aktivierungsfunktionen verwendet. Etabliert hat sich in den letzten Jahrzehnten die Sigmoid-Funktion

$$f_{\text{sig}}(x) = \frac{1}{1 + e^{-x}},$$

die entsprechend Abbildung 5.11 (b) auf den Wertebereich zwischen 0 und 1 abbildet, streng monoton steigend und stetig differenzierbar ist. Alternativ werden auch andere Funktionen, oftmals mit linearem Charakter (Abbildung 5.11 (c) und (d)) herangezogen. Sie weisen zwar ein mathematisch weniger günstiges Verhalten auf, lassen sich aber am Computer effizient berechnen, wodurch dieser Nachteil ausgeglichen wird.



(a) Heavidside-Funktion

(b) Sigmoid-Funktion

(c) Stückweise lineare Funktion

(d) ReLU-Funktion

Abb. 5.11: Überblick über verschiedene Aktivierungsfunktionen.

5.1.6 Implementierung in Java

Es folgt ein Vorschlag für eine Implementierung eines Perzeptrons in der Programmiersprache Java. Die zugrunde liegende Modellierung wurde in der Klassenkarte rechts dargestellt. Wie man sieht, beschränkt sich das Beispiel auf zwei Eingaben. Möchte man das Programm auf den n−dimensionalen Raum verallgemeinern, so muss man lediglich die Gewichte und die Eingaben durch ein Feld ersetzen. Als Aktivierungsfunktion wird die Heaviside-Funktion verwendet, die in der Methode `berechneAktivierung` implementiert ist.

PERZEPTRON
− w1 − w2 − theta − lernrate
+ Perzeptron(neueLernrate) + berechneLabel(x1,x2) + lerne(x1,x2,erwartetesLabel) − berechneUebertragung(x1,x2) − berechneAktivierung(a)

```
public class Perzeptron {

    private double w1;
    private double w2;
    private double theta;
    private double lernrate;

    public Perzeptron(double neueLernrate) {
        lernrate = neueLernrate;
    }

    public void lerne(double x1, double x2, double erwartetesLabel) {
        double delta = erwartetesLabel - berechneLabel(x1,x2);
        w1 = w1 + lernrate * delta * x1 ;
        w2 = w2 + lernrate * delta * x2 ;
        theta = theta - lernrate * delta;
    }

    public double berechneLabel(double x1, double x2) {
        return berechneAktivierung(berechneUebertragung(x1,x2));
    }
```

```
    private double berechneUebertragung(double x1, double x2) {
        return w1 * x1 + w2 * x2 - theta;
    }

    private double berechneAktivierung(double a) {
        if(a >= 0) {
            return 1.0;
        }
        else {
            return 0.0;
        }
    }
}
```

Die programmierte Klasse beschränkt sich lediglich auf das Perzeptron. Zum Training des Perzeptrons mit mehreren Datenpunkten wird mindestens eine weitere Klasse benötigt. Deren Implementierung weist der Lehrplan nicht explizit aus, kann aber zum Testen angegeben werden (siehe Abschnitt 5.3.3).

5.1.7 Vom Perzeptron zum künstlichen neuronalen Netz

5.1.7.1 Erkennung von mehr als zwei Klassen

Wenn das einfache Perzeptron zur Klassifizierung verwendet wird, wird jeder Datenpunkt mit 0 oder 1 gelabelt. Das heißt, mit einem Perzeptron lässt sich lediglich ein Zweiklassenproblem lösen. Ein einfaches Perzeptron ist also ein „binärer“ Klassifikator. Tabelle 5.1 zeigt, wie bisherige Szenarien der Handreichung auf das Klassifizierungsverfahren mit einem einfachen Perzeptron abgebildet werden könnten.

Inwieweit sich die Szenarien allerdings für eine Klassifizierung mit einem einfachen Perzeptron eignen, hängt davon ab, ob sich die Datenpunkte aus der Trainingsmenge linear separieren lassen.

Szenario	Eingaben (Merkmale)	Ausgabe: 0	Ausgabe: 1
Klassifizierung „männlich“ oder „weiblich“ S. 87	Körpergewicht, Körpergröße, Körperfettanteil	Männlich	Weiblich
Sprachenerkennung S. 82	Durchschnittliche Wortlänge, relative Vokalhäufigkeit	Deutsch	Französisch
Fische S. 40	Musterung, Bauchfarbe, Schuppenfarbe, Flossenfarbe	Friedlich	Aggressiv
Bewerbung S. 70	Staatsangehörigkeit, Geschlecht, Englischkenntnisse, Weitere Sprachen, Akademischer Abschluss, Berufserfahrung	Abgelehnt	Eingeladen

Tabelle 5.1: Abbildung von Szenarien der Handreichung auf das Klassifizierungsverfahren mit einem einfachen Perzeptron.

Betrachtet man das Szenario „Sprachenerkennung“ (s. Seite 82), das im Rahmen des k-nächste-Nachbarn-Algorithmus vorgestellt wurde, so stellt sich die Frage, ob es mit dem Verfahren auch möglich ist, zwischen mehr als nur zwei Klassen (beispielsweise Englisch, Französisch und Deutsch) zu unterscheiden. Dieses Problem wird gelöst, indem man für jede Klasse ein eigenes Perzeptron verwendet, das nur feuern soll, wenn die zugehörige Klasse erkannt wurde.

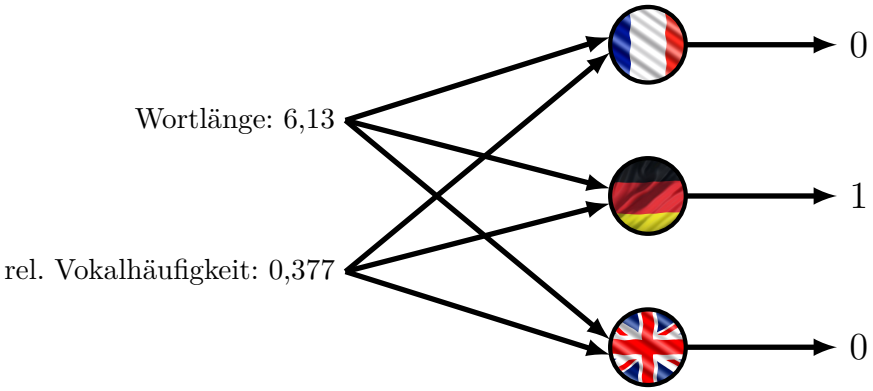


Abb. 5.12: Erkennung mehrerer Klassen.

Zur Veranschaulichung wurde in Abbildung 5.12 dargestellt, wie man mit drei Perzeptronen, die zur Erkennung der jeweiligen Sprache trainiert wurden, anhand der Eingaben eine Klassifizierung durchführen kann. Zur besseren Übersicht wurde auf die Darstellung der jeweiligen Gewichte und Schwellenwerte verzichtet. Da nur das Perzeptron, das zur Erkennung deutscher

Texte trainiert wurde, feuert, ist das Label „deutsch“ auch das Klassifizierungsergebnis.

Von einer Verwendung der Heaviside-Funktion als Aktivierungsfunktion ist hier abzuraten, da die Klassifizierung scheitert, wenn es zu widersprüchlichen Ausgaben kommt. Dies ist der Fall, wenn beispielsweise sowohl das Perzeptron für „Französisch“ als auch das Perzeptron für „Englisch“ feuert. Würden die Perzeptronen stattdessen eine kontinuierliche Aktivierungsfunktion verwenden, so könnte beispielsweise für Französisch der Wert 0,89 und für Englisch der Wert 0,74 ausgegeben werden. Somit legt man sich zwar auf das Label „Französisch“ fest, man wüsste dann aber auch, dass eine erhebliche Verwechslungsgefahr mit dem Label „Englisch“ besteht.

5.1.7.2 Das Perzeptron als Baustein eines künstlichen neuronalen Netzes

Im Vergleich zu vielen anderen Klassifikatoren benötigt das künstliche Neuron kaum Speicherplatz und das Klassifizieren ist sehr recheneffizient. Lassen sich die Merkmale aber nicht in zwei linear separierbare Halbräume aufteilen, so ist dieses Verfahren zur Klassifizierung ungeeignet. Allerdings können, ähnlich wie Nervenzellen, auch künstliche Neuronen miteinander verbunden werden. Dies erlaubt dann weitaus komplexere Anordnungen der Trainingsdatenpunkte, wie folgende Abbildung veranschaulicht:

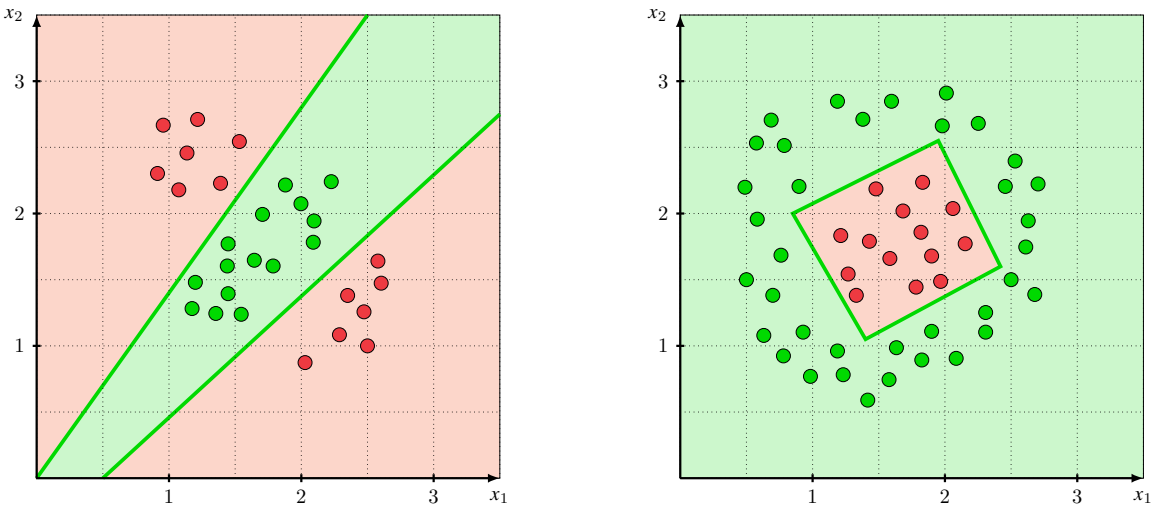


Abb. 5.13: Beispiele einer korrekten Klassifizierung von Datenpunkten, die nicht linear separierbar sind.

Um aus einzelnen künstlichen Neuronen ein künstliches neuronales Netz zu bilden, müssen die Ausgaben der einzelnen Neuronen mit den Eingaben weiterer Neuronen verbunden werden.

Ein einfaches künstliches neuronales Netz mit nur fünf Neuronen sieht beispielsweise wie in Abbildung 5.14 dargestellt aus.

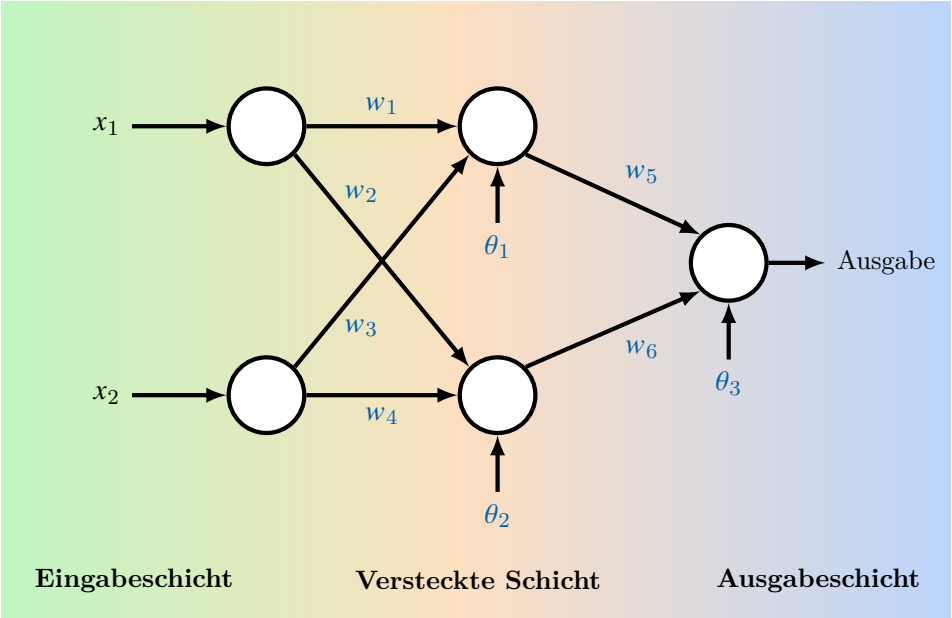


Abb. 5.14: Fünf Neuronen zu einem künstlichen neuronalen Netz kombiniert.

Dabei wird in der Darstellung zwischen drei Schichten unterschieden. Die Eingabeschicht besteht aus Neuronen, die mit den Eingabedaten direkt verbunden sind und diese direkt an die nächste Schicht weiterleiten. Darauf folgen eine oder mehrere „versteckte“ Schichten, die zwischen der Ein- und der Ausgabeschicht liegen. Diese Schichten werden als versteckt bezeichnet, weil ihre Neuronen normalerweise nicht direkt angesteuert und ihre Ausgabe(n) nicht direkt beobachtbar sind. Die Ausgabeschicht besteht aus Neuronen, die die Ausgabe des Netzes erzeugen, die beispielsweise das Ergebnis einer Klassifizierung sein kann.

Das Netzwerk aus Abbildung 5.15 besteht aus vier Eingabeneuronen, zwei versteckten Schichten mit fünf beziehungsweise drei Neuronen und zwei Ausgabeneuronen. Der Übersichtlichkeit halber wurde auf die Beschriftung der Gewichte und Schwellenwerte verzichtet.

Durch das Hinzufügen von zusätzlichen versteckten Schichten beziehungsweise durch die Erhöhung der Anzahl der Neuronen in den versteckten Schichten können immer komplexere Zusammenhänge modelliert werden. Gleichzeitig besteht aber auch die Gefahr der Überanpassung, wie sie bereits im Kapitel Entscheidungsbaum auf Seite 41 thematisiert wurde. Die optimale Konfiguration der versteckten Schichten hängt von der Aufgabenstellung ab und kann nicht pauschal angegeben werden. Das Hinzufügen von immer mehr versteckten

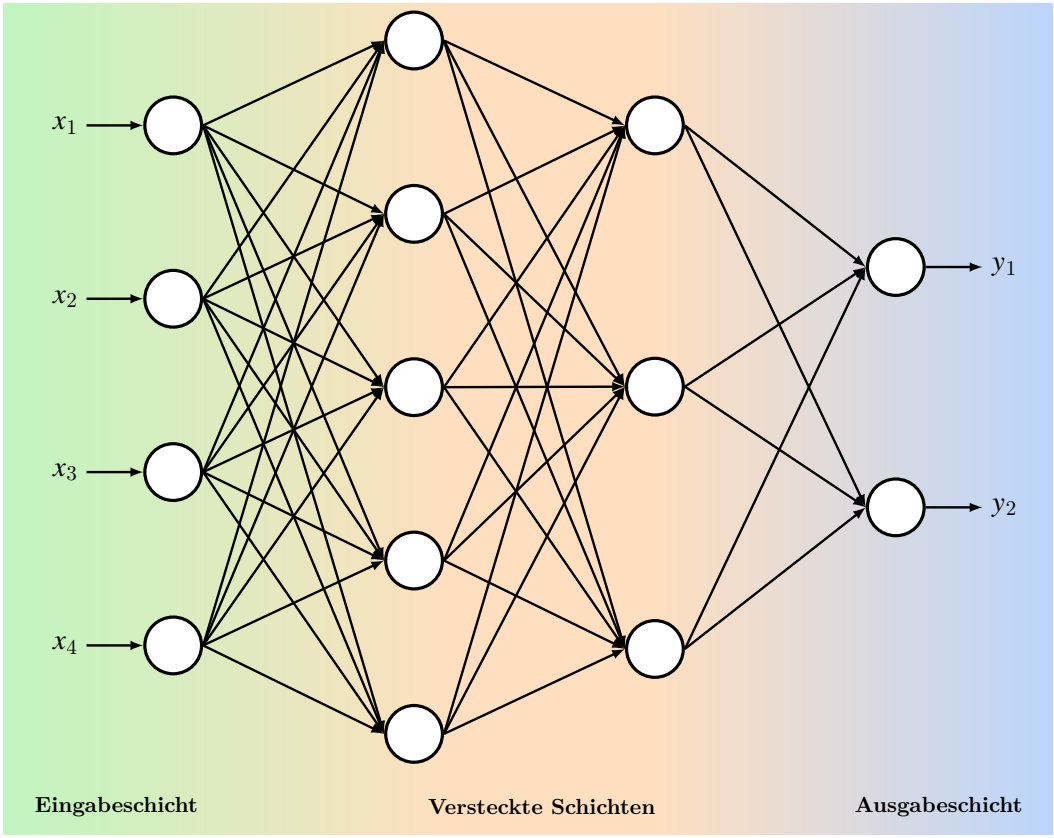


Abb. 5.15: Ein künstliches neuronales Netz mit zwei versteckten Schichten.

Schichten führt dazu, dass das künstliche neuronale Netz immer „tiefer“ wird, was auch als „Deep Learning“ bezeichnet wird. Es existieren Anwendungen, insbesondere in der Bildverarbeitung, mit hunderten bis tausenden versteckten Schichten. Allerdings sind die Schichten nicht zwangsläufig so engmaschig miteinander verbunden, wie in Abbildung 5.15 dargestellt wurde. Des Weiteren existieren noch andere Arten von künstlichen neuronalen Netzen, die sich darin unterscheiden, wie die jeweiligen Schichten miteinander verbunden sind.

Die Delta-Lernregel des Perzeptrons ist eine Form des überwachten Lernens. Bei künstlichen neuronalen Netzen kommt die Delta-Lernregel ebenfalls zum Einsatz, muss aber für die Neuronen der versteckten Schichten modifiziert werden, da der Fehler δ nicht direkt ermittelt werden kann. Diese verallgemeinerte Delta-Lernregel wird als Backpropagation-Algorithmus bezeichnet und ist aktuell der am häufigsten eingesetzte Algorithmus zum überwachten Lernen von künstlichen neuronalen Netzen. Diese können auch mit anderen Arten des maschinellen Lernens trainiert werden, die dafür benötigten Trainingsverfahren unterscheiden sich aber erheblich von der Delta-Lernregel. Eine vertiefte Betrachtung der künstlichen neuronalen Netze erfolgt im LehrplanPLUS der 13. Jahrgangsstufe.

5.2 Didaktische Hinweise / Bezug zum Lehrplan

5.2.1 Einordnung in den Lehrplan

Im LehrplanPLUS findet sich folgende Kompetenzerwartung zum Themenkomplex Perzeptron und künstliche neuronale Netze:

> *Die Schülerinnen und Schüler erläutern die Funktionsweise eines künstlichen Neurons (Perzeptron) und beschreiben den grundsätzlichen Aufbau eines künstlichen neuronalen Netzes.*

Als Inhalte zu den Kompetenzen wird neben dem Perzeptron auch die Delta-Lernregel angegeben. Es wird darauf hingewiesen, dass die Schülerinnen und Schüler bereits in der achten Jahrgangsstufe im Fach Biologie den Aufbau und die Funktionsweise einer Nervenzelle kennengelernt haben.

Im Hinblick auf das einfache Perzeptron wird die Kompetenzerwartung bezüglich der Informatik (NTG) und der spät beginnenden Informatik folgendermaßen unterschieden:

- **Informatik 11 (NTG)**

  Die Schülerinnen und Schüler implementieren ein künstliches Neuron.

- **Spät beginnende Informatik 11 (HG, SG, MuG, SWG)**

  Die Schülerinnen und Schüler simulieren ein künstliches Neuron.

Zwar wird von den Schülerinnen und Schülern am NTG nicht explizit eine Simulation gefordert, es bietet sich aber dennoch an, vor der Implementierung eine Simulation durchzuführen, um die Funktionsweise des künstlichen Neurons, insbesondere die Idee der Delta-Lernregel, zu veranschaulichen.

Darüber hinaus erwerben die Schülerinnen und Schüler bei der Behandlung des Perzeptrons zusätzlich folgende Kompetenz:

> *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.*

Gerade im Hinblick auf den Einfluss der Trainingsdaten lässt sich beim Perzeptron zeigen, dass nur dann sinnvolle Ergebnisse zu erwarten sind, wenn eine lineare Separierbarkeit der Datenpunkte möglich ist. Bezüglich der Delta-Lernregel spielt die Lernrate als Hyperparameter eine sehr wichtige Rolle, da von dieser abhängt, wie viele Schritte notwendig sind, bis der Algorithmus terminiert.

5.2.2 Durchführung

Der Vorschlag zur Durchführung des Themenkomplexes ist unabhängig von der Ausbildungsrichtung, da sich der einzige Unterschied in den Kompetenzerwartungen des Lehrplans auf die Implementierung des Perzeptrons bezieht. Der entsprechende Abschnitt 5.2.2.4 der Handreichung kann dann für den spät beginnenden Informatikunterricht übersprungen werden.

Insgesamt werden für den Themenkomplex drei Unterrichtsstunden für den spät beginnenden Informatikunterricht vorgeschlagen. Im Informatikunterricht des NTG kann die Implementierung des Perzeptrons in weiteren zwei Stunden durchgeführt werden.

5.2.2.1 Einstieg

Es bietet sich an, nicht direkt mit dem informatischen Modell des Perzeptrons einzusteigen, sondern zunächst zu thematisieren, dass sich viele technische Anwendungen die Natur zum Vorbild machen. Hierzu könnte man in einer Präsentation (siehe Material in Abschnitt 5.3.1.2) Bilder zeigen, bei denen die Schülerinnen und Schüler die passende technische Anwendung erkennen beziehungsweise zuordnen sollen. Typische Beispiele sind:

- Mohnblumen $\Longrightarrow$ Salzstreuer
- Kletten-Pflanzen $\Longrightarrow$ Klettverschluss
- Saugnäpfe der Tentakel eines Kraken $\Longrightarrow$ Saugnapf einer Halterung
- Schere einer Krabbe $\Longrightarrow$ Zange

Ausgehend von dieser Zuordnung lässt sich in einem Unterrichtsgespräch motivieren, dass man das Ziel, nämlich intelligentes Verhalten zu simulieren, durch eine digitale Nachahmung des Gehirns erreichen kann. Dazu wird zunächst die Nervenzelle als fundamentaler Baustein des Gehirns genauer betrachtet.

Hier kann an das Vorwissen der Schülerinnen und Schüler aus der achten Jahrgangsstufe Biologie angeknüpft werden. Dies kann mit folgender Umsetzung erfolgen: Die Schülerinnen und Schüler sehen sich einen kurzen Film zum Thema „Neuronen“ mit der Vorgabe an, eine Nervenzelle zu skizzieren und die Bestandteile zu beschriften. Ein geeigneter Film lässt sich in der ByCS Mebis-Mediathek finden: `https://mebis.link/neuronen`.

5.2.2.2 Informatische Modellierung eines Neurons

Nachdem die biologische Nervenzelle thematisiert wurde, sollte als nächstes behandelt werden, wie man diese informatisch modellieren kann. Dies könnte über eine Präsentation oder Videosequenz erfolgen (Material in Abschnitt 5.3.1.2). Ohne auf das Lernverfahren selbst einzugehen, ist es bereits an dieser Stelle sinnvoll, klarzustellen, dass der Schwellenwert und die Gewichte durch maschinelles Lernen ermittelt werden. Es ist prinzipiell ausreichend, wenn die Schülerinnen und Schüler anhand der Gleichung

$$w_1 \cdot x_1 + w_2 \cdot x_2 \geq \theta$$

eine Aussage darüber treffen können, ob ein Perzeptron feuert, also den Wert 1 ausgibt, oder nicht feuert, also den Wert 0 ausgibt. Eine mathematische Formulierung als Verkettung von Übertragungs- und Aktivierungsfunktion ist nicht notwendig.

Hinweis: Anstelle des Symbols θ für den Schwellenwert findet man in manchen Werken auch das Zeichen S.

An dieser Stelle empfehlen sich Übungsaufgaben, bei denen die Schülerinnen und Schüler für ein gegebenes Perzeptron berechnen, ob für gegebene Eingaben ein Perzeptron feuert oder nicht. Das kann beispielsweise so geschehen, dass ausgehend von einem Perzeptron mit $w_1 = -2$, $w_2 = 5$, $\theta = 10$ Eingaben für x_1 und x_2 jeweils im Wertebereich $[0; 4]$ verteilt werden. Die Jugendlichen sollen für ihre Werte berechnen, ob das Perzeptron feuert, und in ein Koordinatensystem einen entsprechend roten beziehungsweise grünen Punkt markieren. Hierzu eignet sich eine Tafel, aber auch ein vorbereitetes Flipchart-Blatt mit roten und grünen

Klebepunkten.

Bei diesem Vorgehen erkennen die Schülerinnen und Schüler, dass das Koordinatensystem linear in zwei Hälften aufgeteilt wird. Man kann dann eine Trennlinie einzeichnen lassen, die durch eine mathematische Umformung der Formel

$$-2 \cdot x_1 + 5 \cdot x_2 = 10$$

zur Geradengleichung

$$x_2 = 0{,}4 \cdot x_1 + 2$$

bestätigt werden kann.

Für eine alternative Herangehensweise findet man im Material-Abschnitt 5.3.1.3 zur Handreichung zum einen eine Videosequenz, in der der obige Vorgang simuliert wurde, zum anderen eine Computeranwendung, bei der die Simulation mit beliebig konfigurierten Perzeptronen durchgeführt werden kann.

5.2.2.3 Delta-Lernregel

Bevor der Algorithmus thematisiert wird, ist es sinnvoll, die Schülerinnen und Schüler selbstständig im Rahmen eines Szenarios die Gewichte und den Schwellenwert eines Perzeptrons manuell durch „Ausprobieren“ ermitteln zu lassen. Dazu sollten Trainingsdaten vorbereitet und auf eine geeignete Lernsoftware zurückgegriffen werden. Der Vorteil dieser Herangehensweise ist, dass die Schülerinnen und Schüler explorativ lernen, welche Auswirkungen eine Erhöhung beziehungsweise Verringerung der Gewichte und des Schwellenwerts mit sich bringt. Es stehen mehrere Tools zur Verfügung (s. Abschnitt 5.3.2), mit denen das Perzeptron simuliert werden kann.

Nachdem die Schülerinnen und Schüler manuell die Gewichte und den Schwellenwert des künstlichen Neurons ermittelt haben, soll nun die Frage nach einer automatisierten Umsetzung des Lernverfahrens in Form eines Algorithmus thematisiert werden. Dazu bietet es sich an, ein Beispiel (s. Abbildung 5.16) zu präsentieren, bei dem ein Perzeptron bereits so konfiguriert ist, dass nur ein Datenpunkt knapp falsch klassifiziert wird.

Anhand dieses und weiterer Beispiele können nun die Formeln (zunächst für $\alpha = 1$) zur Delta-Lernregel erläutert werden (s. Abschnitt 5.1.4.2). Insgesamt sollen die drei Möglichkeiten für

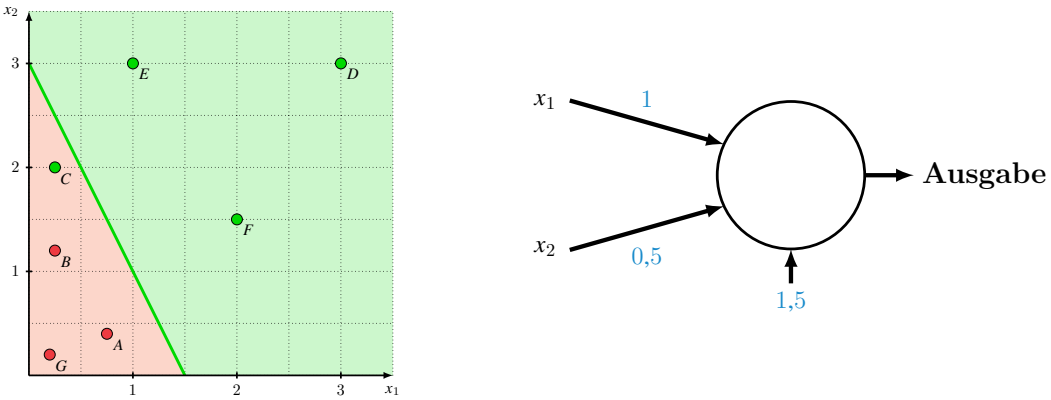


Abb. 5.16: Ein vorkonfiguriertes Perzeptron, das alle Trainingsdaten bis auf Datenpunkt *C* richtig klassifiziert.

die Belegung von δ veranschaulicht werden.

Der Einfluss der Lernrate α kann am besten mit einem Tool (s. Abschnitt 5.3.2) verdeutlicht werden. Die Schülerinnen und Schüler verwenden dabei Trainingsdaten, die linear separierbar sind. Dadurch kann der Algorithmus terminieren, wenn alle Trainingsdaten richtig klassifiziert werden.

Hinweis: Die Erläuterung der Delta-Lernregel kann auch mithilfe des Tools „Demonstrator für maschinelles Lernen“ (genauer in Abschnitt 5.3.2) erfolgen, da hierbei die Schritte samt Berechnungen visualisiert werden und man jederzeit Zugriff auf alle Parameter hat (s. Abbildung 5.17).

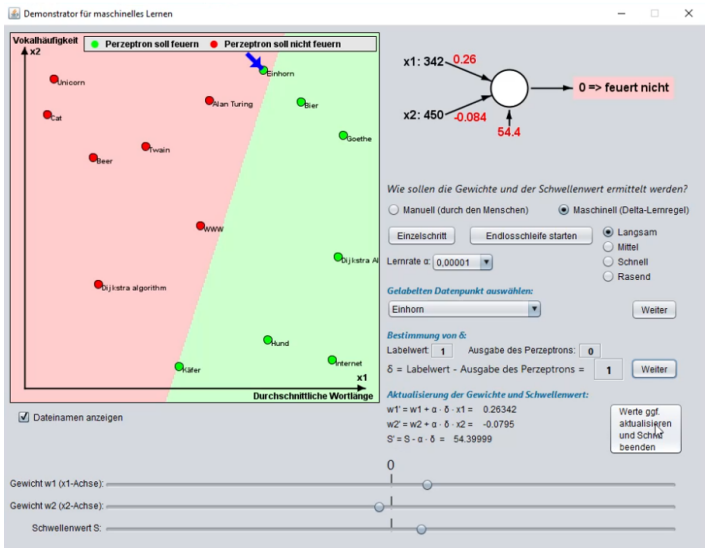


Abb. 5.17: Veranschaulichung der Delta-Lernregel mithilfe des Tools „Demonstrator für maschinelles Lernen“.

5.2.2.4 Implementierung des Perzeprons (nur NTG)

Vor der Implementierung bietet es sich an, zunächst das künstliche Neuron zu modellieren. Dazu stellt man ein Perzeptron, das zuvor schon im Unterricht behandelt wurde, zunächst als Objekt dar:

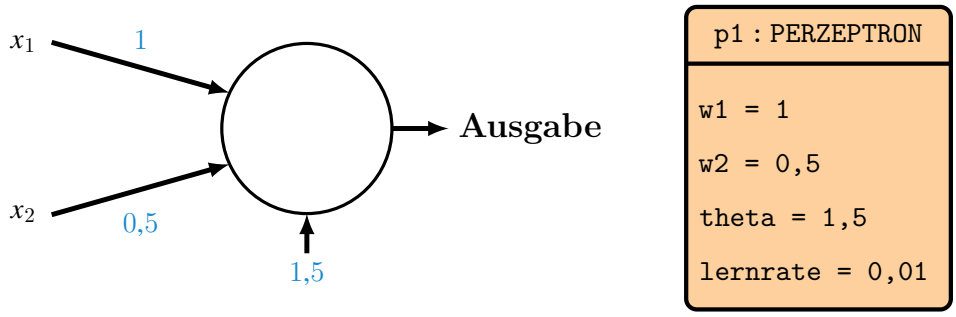


Abb. 5.18: Modellierung eines Perzeptrons als Objekt.

Wie man sieht, wird in dem Beispiel von zweidimensionalen Datenpunkten ausgegangen, was ausreicht, um die Kompetenzerwartung zu erfüllen.

Im Gegensatz zum Attribut `lernrate` erscheinen `w1`, `w2` und `theta` recht offensichtlich. Theoretisch wäre denkbar, auf den Hyperparameter zu verzichten und einen Wert als Konstante bei der Implementierung direkt in den Code einzufügen. Dies hat aber den Nachteil, dass man dann die Möglichkeit verliert, diesen Wert zur Laufzeit zu ändern.

Als nächstes wird geklärt, welche Methoden ein Objekt der Klasse `PERZEPTRON` bereitstellen muss, woraufhin dann die Klasse modelliert werden kann. Aus objektorientierter Sicht wäre es zwar geschickter, die Übertragungs- und Aktivierungsfunktion in eigene Methoden auszulagern, da man dann diese in Unterklassen einfach überschreiben könnte (um beispielsweise die Heaviside- mit der Sigmoidfunktion auszutauschen).

PERZEPTRON
– w1
– w2
– theta
– lernrate
+ Perzeptron(neueLernrate)
+ berechneLabel(x1,x2)
+ lerne(x1,x2,erwartetesLabel)

Da dies aber vorher nicht explizit thematisiert wurde, können die Schülerinnen und Schüler diese zusätzlichen Methoden nicht nachvollziehen. Die Java-Klasse, die in Abschnitt 5.1.6 auf Seite 139 präsentiert wurde, kann für den Unterricht vereinfacht werden:

```
public class Perzeptron {

    private double w1;
    private double w2;
    private double theta;
    private double lernrate;

    public Perzeptron(double neueLernrate) {
        lernrate = neueLernrate;
    }

    public void lerne(double x1, double x2, double erwartetesLabel) {
        double delta = erwartetesLabel - berechneLabel(x1,x2);
        w1 = w1 + lernrate * delta * x1;
        w2 = w2 + lernrate * delta * x2;
        theta = lernrate * theta - delta;
    }

    public double berechneLabel(double x1, double x2) {
        if(w1 * x1 + w2 * x2 >= theta) {
            return 1.0;
        } else {
            return 0.0;
        }
    }
}
```

Im Unterrichtsverlauf könnte die Wahl des Datentyps des Rückgabewerts der Methode `berechneLabel(x1,x2)` thematisiert werden. Die Entscheidung für `double` lässt sich dahingehend erklären, dass bei den Formeln auch einige Variablen `double`-Werte abspeichern und eine Mischung von verschiedenen Datentypen innerhalb einer Berechnung häufig zu Problemen führt. Damit wäre auch erklärt, warum der Parameter `erwartetesLabel` ebenfalls mit `double` deklariert wurde.

Hinweis: Es ist für die Schülerinnen und Schüler wenig motivierend, nur die Java-Klasse zu programmieren, ohne danach zu sehen, wie das Perzeptron mit Trainingsdaten lernt. Tatsächlich ist aber der zeitliche Aufwand, den Rahmen für ein Training im Unterricht zu implementieren, kaum zu rechtfertigen, zumal dies keinen Kompetenzgewinn bezüglich der im Lehrplan ausgewiesenen Erwartungen bringt. Daher sollte eine Klasse, die das selbst implementierte Perzeptron nutzt, den Schülerinnen und Schülern zur Verfügung gestellt werden (s. Abschnitt 5.3.3), damit sie auch erleben können, wie „ihr“ Perzeptron lernt. An dieser Stelle bietet es sich an, differenziert nach Leistungsstand der Schülerinnen und Schüler unterschiedlich weit ausgearbeitete Vorlagen zu verwenden. Für besonders leistungsstarke Schülerinnen und Schüler besteht die Möglichkeit, das Perzeptron für n-dimensionale Eingaben mit einer indizierten Datenstruktur zu implementieren.

5.2.2.5 Aufbau eines künstlichen neuronalen Netzes

Als Einstieg bietet es sich an, anhand von Beispielen noch einmal hervorzuheben, dass das Perzeptron auf Trainingsdatenpunkte beschränkt ist, die sich linear separieren lassen. In folgender Abbildung ist offensichtlich keine trennende Gerade möglich:

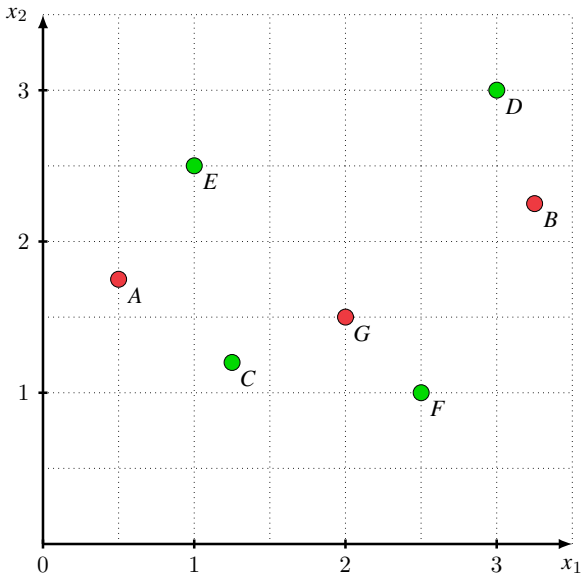


Abb. 5.19: Trainingspunkte, die sich nicht linear separieren lassen.

Dennoch kann plausibel gemacht werden, dass man das Problem in zwei Teilprobleme zerlegen kann, für die jeweils ein Perzeptron zuständig ist:

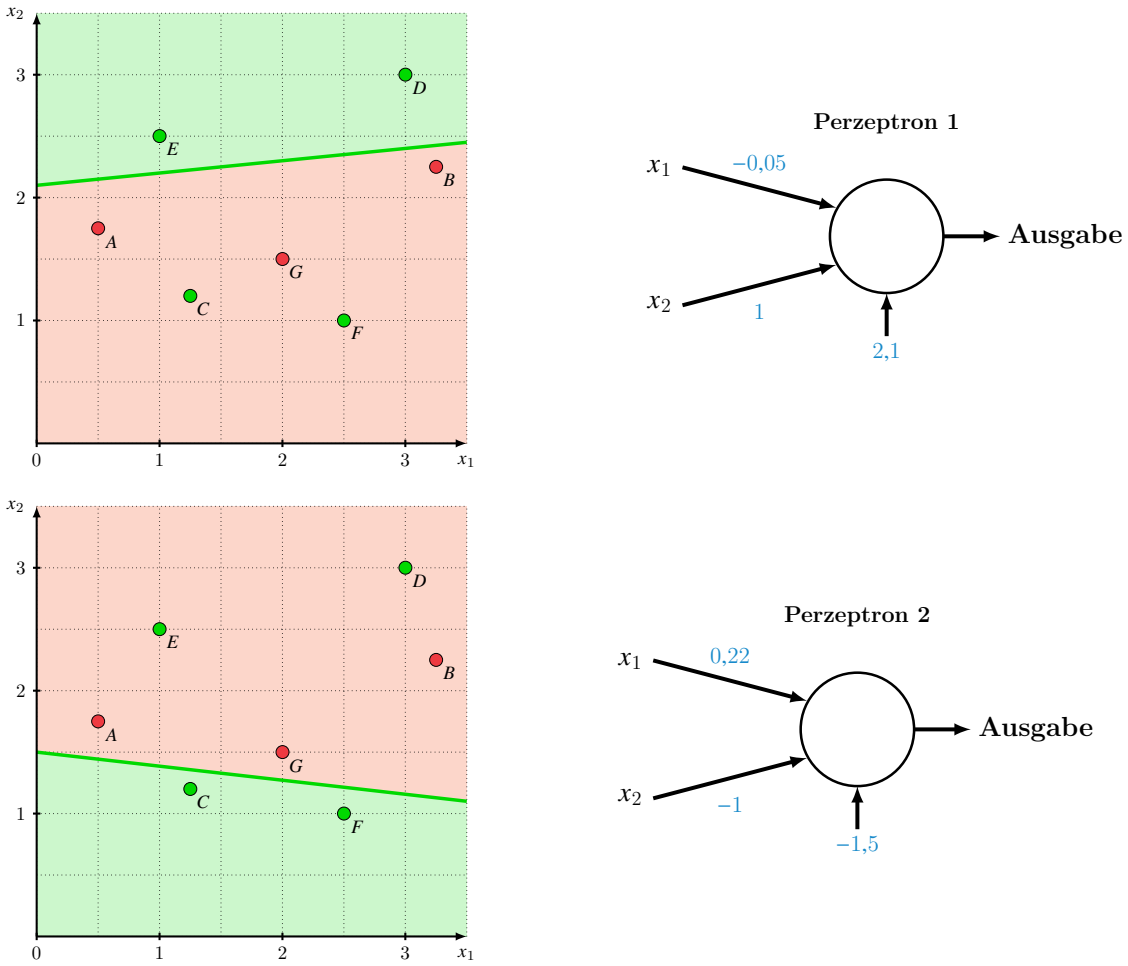


Abb. 5.20: Separierung der beiden oberen grünen Datenpunkte (oben) bzw. der beiden unteren grünen Datenpunkte (unten) jeweils durch eine Gerade.

Man kann erkennen, dass Eingaben dann mit einer 1 (grün) gelabelt werden müssen, wenn entweder das Perzeptron 1 **oder** das Perzeptron 2 feuert. Dies kann dadurch erreicht werden, dass die Ausgaben beider Perzeptronen in ein drittes Perzeptron als Eingaben fließen, das genau dann feuert, wenn mindestens eine 1 eingegeben wird (Abbildung 5.21).

Somit wird den Schülerinnen und Schülern deutlich, warum es lohnend sein kann, einzelne Neuronen zu einem künstlichen neuronalen Netz zu verbinden. Daraufhin können die verschiedenen Schichten eingeführt werden (Abbildung 5.22).

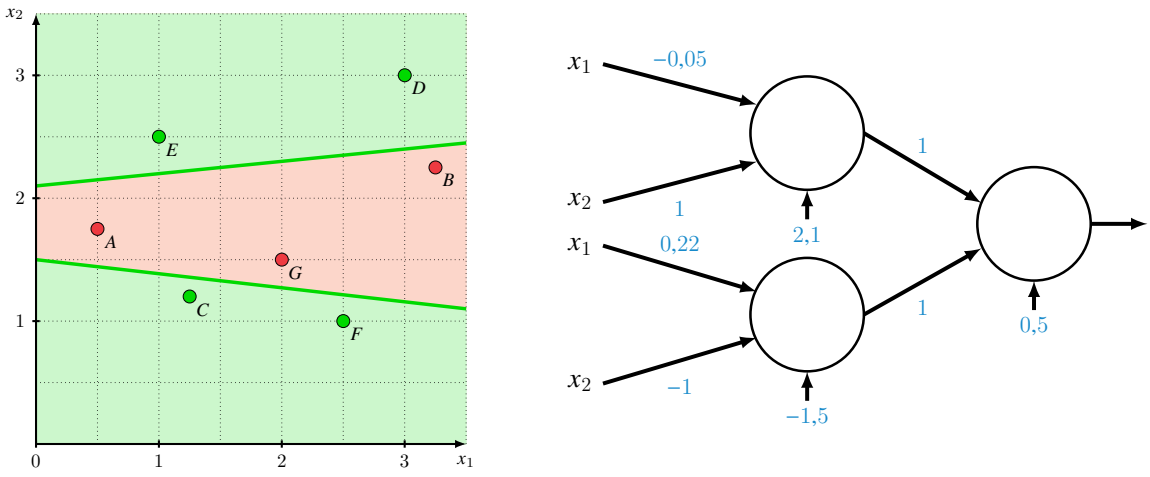


Abb. 5.21: Separierung der grün bzw. rot gelabelten Datenpunkte durch zwei Geraden.

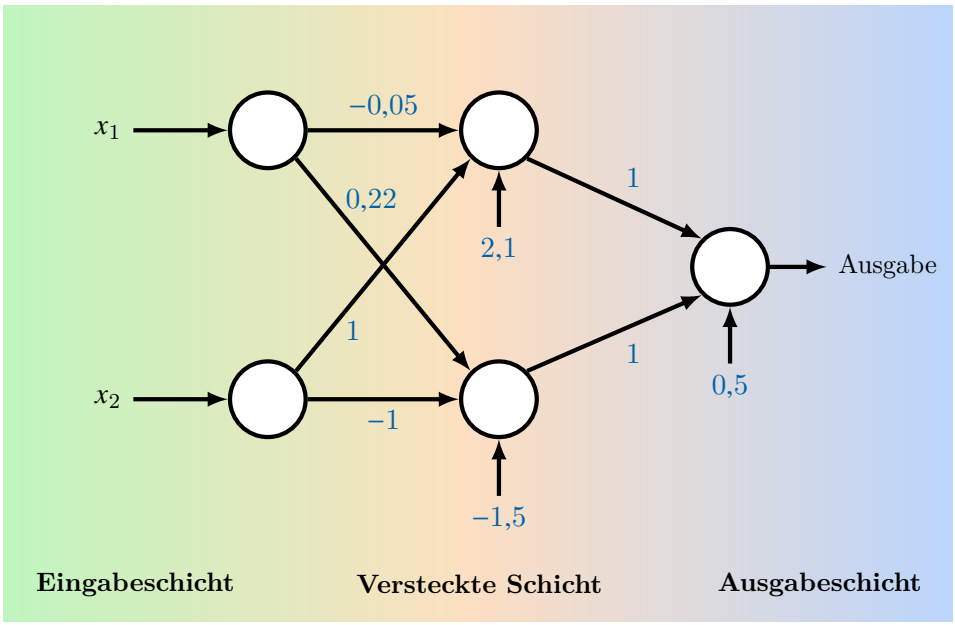


Abb. 5.22: Künstliches neuronales Netz mit fünf Neuronen zur Separierung der Datenpunkte aus Abbildung 5.19.

5.3 Material

5.3.1 Einführung des Perzeptrons

5.3.1.1 Bilder zu technischen Anwendungen und den Vorbildern aus der Natur

In Abschnitt 5.2.2.1 sollten die Schülerinnen und Schüler technische Anwendungen ihren Vorbildern aus der Natur zuordnen. Hierzu befindet sich im Materialordner die Präsentation `Die Natur als Vorbild.pptx`, die für den Unterricht geeignetes Bildmaterial enthält. Die Bilder stehen größtenteils unter einer Creative-Commons-Lizenz. Die entsprechenden Nutzungsrechte sind auf der zweiten Folie aufgeführt.

5.3.1.2 Gegenüberstellung Nervenzelle und Perzeptron (Präsentation / Video)

Im Materialordner befindet sich eine Präsentation, die sich zur Einführung des einfachen Perzeptrons eignet. Dabei findet eine Gegenüberstellung zur Nervenzelle statt. Ziel ist die Einführung der Ungleichung, mit der geprüft werden kann, ob das Perzeptron feuert: $w_1 \cdot x_1 + w_2 \cdot x_2 \geq 0$. Des Weiteren befindet sich im gleichen Ordner ein kurzes Video zur Einführung der Ungleichung. Man könnte dieses Video beispielsweise einsetzen, wenn man nach dem Flipped-Classroom-Prinzip arbeitet.

5.3.1.3 Simulation eines Perzeptrons

Wenn die Schülerinnen und Schüler das künstliche Neuron kennenlernen, ist es wichtig, zu veranschaulichen, dass durch dieses Modell eine lineare Trennung der Daten stattfindet. Hierzu gibt es im Materialordner eine Anwendung, mit der man ein Perzeptron (für zweidimensionale Datenpunkte) konfigurieren kann. Anschließend kann man für einzelne Datenpunkte, die man entweder manuell eingibt oder zufällig gewählt werden, berechnen, ob das Perzeptron feuert.

Die Ergebnisse werden durch Datenpunkte, die entsprechend der berechneten Label gefärbt sind (Wertebereich $[0; 5]$ auf beiden Achsen), visualisiert. Aus diesen lässt sich auf eine Gerade schließen, die die Datenpunkte separiert.

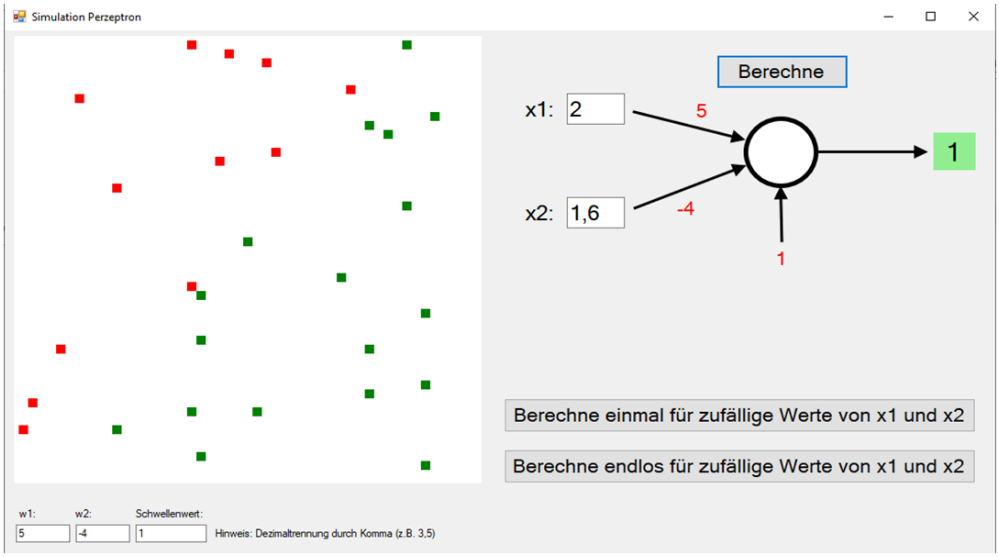


Abb. 5.23: Anwendung zur Simulation eines einfachen Perzeptrons.

Bei den Eingaben muss darauf geachtet werden, dass die Dezimaltrennung durch ein Komma erfolgt. Die Anwendung wurde unter `.NET` entwickelt und sollte auf allen Systemen lauffähig sein, die die Plattform unterstützen. Bei Windows ist sie beispielsweise automatisch integriert. Auf Linux-Systemen müsste vorher das Mono-Paket installiert werden. Zusätzlich befindet sich im Materialordner ein kurzes Video, das die Benutzung des Programms durchspielt.

5.3.2 Delta-Lernregel

5.3.2.1 Demonstrator für maschinelles Lernen

Um den Schülerinnen und Schülern die Funktionsweise der Delta-Lernregel (s. Abschnitt 5.2.2.3) zu veranschaulichen, bietet es sich an, im Unterricht den Demonstrator für maschinelles Lernen einzusetzen. Die neueste Programmversion kann auf der Seite `https://klassenkarte.de` heruntergeladen werden. Die Anwendung wurde in Java entwickelt und setzt daher eine aktuelle Java-Installation (mindestens Version 8) voraus. Nach dem Start stehen die Auswahlmöglichkeiten aus Abbildung 5.24 zur Verfügung.

Für die Delta-Lernregel ist nur die rechte Seite von Bedeutung. Die Schaltfläche „Sprachenerkennung: Gewichte lernen“ bezieht sich auf das Szenario, das in Abschnitt 4.1.2 beschrieben wurde. Ähnlich wie beim k-nächste-Nachbarn-Algorithmus (s. Abschnitt 4.3.2) können im

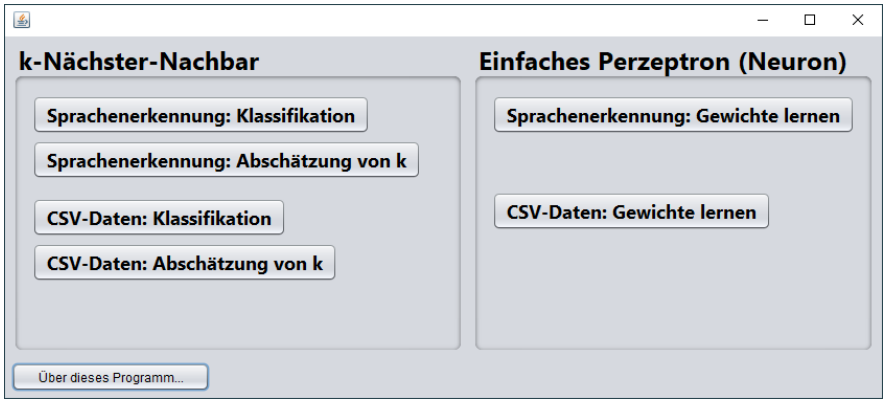


Abb. 5.24: Startmenü des Demonstrators für maschinelles Lernen.

Verzeichnis „daten_sprachen_perzeptron“ Trainingsdaten in Form von Textdateien abgelegt werden. Da es sich beim Perzeptron um einen binären Klassifikator handelt, werden alle Texte, deren Dateinamen mit „`1_`“ beginnen, der Klasse 1 (Perzeptron feuert) zugeordnet, alle anderen der Klasse 0 (Perzeptron feuert nicht).

Alternativ können durch Betätigung der unteren Schaltfläche die Trainingsdaten aus einer CSV-Datei eingelesen werden (s. Abschnitt 4.3.2.3). Dabei kann ein Datenpunkt wieder nur mit „0“ oder „1“ gelabelt sein.

Nach dem Start öffnet sich das Hauptfenster. Dem Benutzer bzw. der Benutzerin stehen zwei Möglichkeiten zur Verfügung, um das Perzeptron zu konfigurieren. Standardmäßig ist „Manuell“ ausgewählt (s. Abbildung 5.25, Pfeil). Im unteren Bereich des Fensters befinden sich drei Schieberegler, mit denen die Gewichte und der Schwellenwert eingestellt werden können.

Wählt man die Option „Maschinell“ (s. Abbildung 5.26, Pfeil 1), so werden zusätzliche Elemente eingeblendet. Bei der Wahl von „Einzelschritt“ (Pfeil 2) kann anschaulich die Delta-Lernregel für einen ausgewählten Datenpunkt durchgespielt werden. Die Lernrate α kann ebenfalls konfiguriert und jederzeit geändert werden. Durch den Klick auf „Weiter“ werden die einzelnen Schritte ausgeführt und die Ergebnisse der Berechnungen angezeigt.

Das Trainieren des Perzeptrons dauert im Einzelschrittmodus in der Regel zu lange. Daher kann man alternativ die Delta-Lernschritte in einer Endlosschleife wiederholen lassen (Pfeil 3). Die Geschwindigkeit, mit der trainiert wird, kann über die Optionsknöpfe rechts gesteuert werden. Die Wiederholung lässt sich durch einen weiteren Klick auf die Schaltfläche (Pfeil 3)

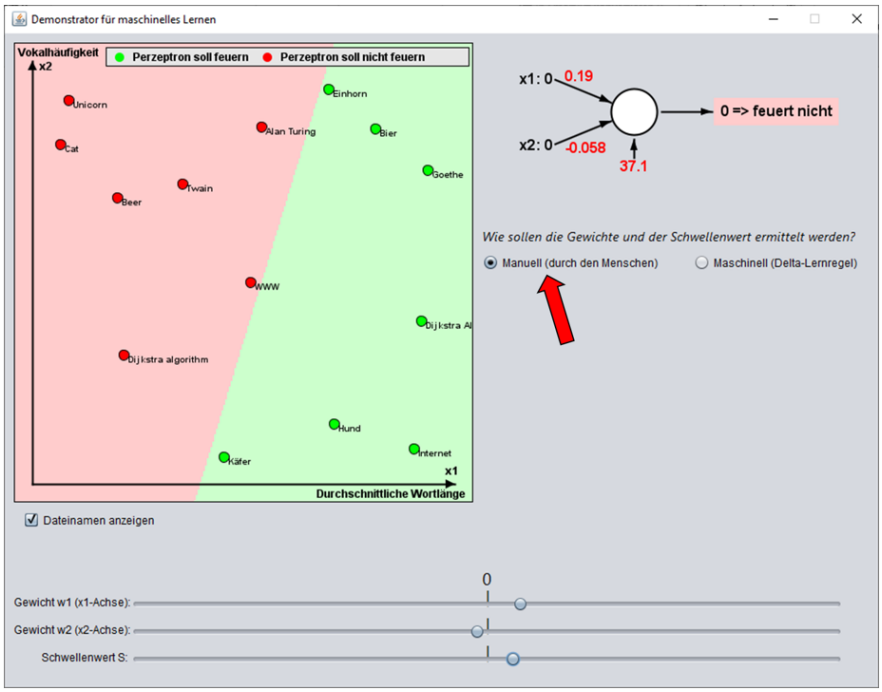


Abb. 5.25: Durch die drei Regler im unteren Bereich lässt sich das Perzeptron manuell konfigurieren.

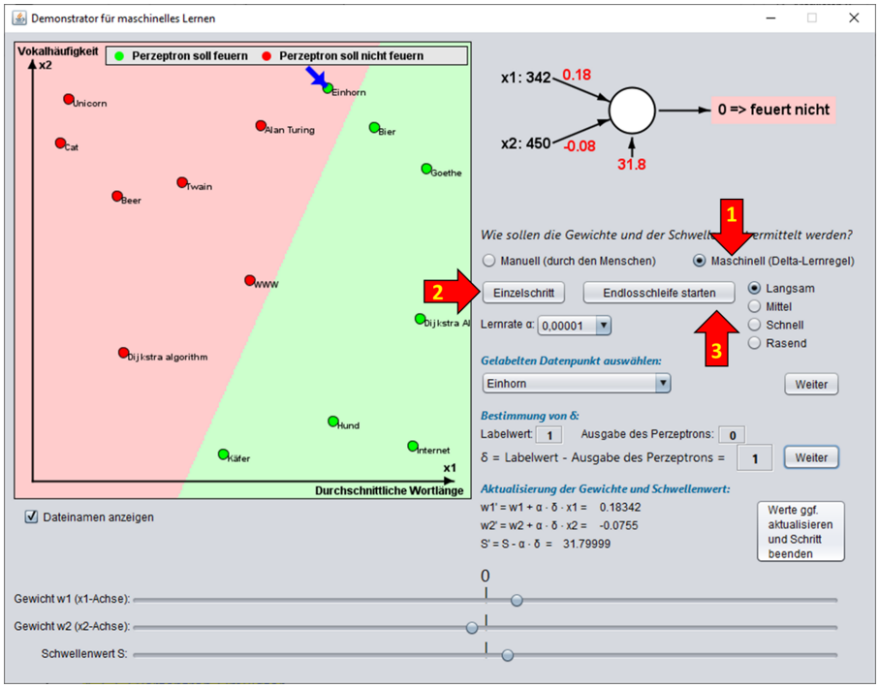


Abb. 5.26: Veranschaulichung der Delta-Lernregel.

unterbrechen.

Im Materialordner befindet sich das Video `Perzeptron trainieren.mp4`, das anhand des Demonstrators für maschinelles Lernen die Delta-Lernregel erklärt.

5.3.2.2 Perzeptron-Simulator

An der Didaktik der Informatik der Universität Passau wurde der Perzeptron-Simulator (s. Abbildung 5.27) entwickelt, der sich ebenfalls eignet, die Delta-Lernregel zu veranschaulichen. Wie beim Demonstrator für maschinelles Lernen können Trainingsdaten über eine CSV-Datei geladen werden. Das Tool ist sowohl auf Windows- als auch auf Apple-Geräten lauffähig. Die Software sowie das zugehörige Benutzerhandbuch finden sich im Materialordner.

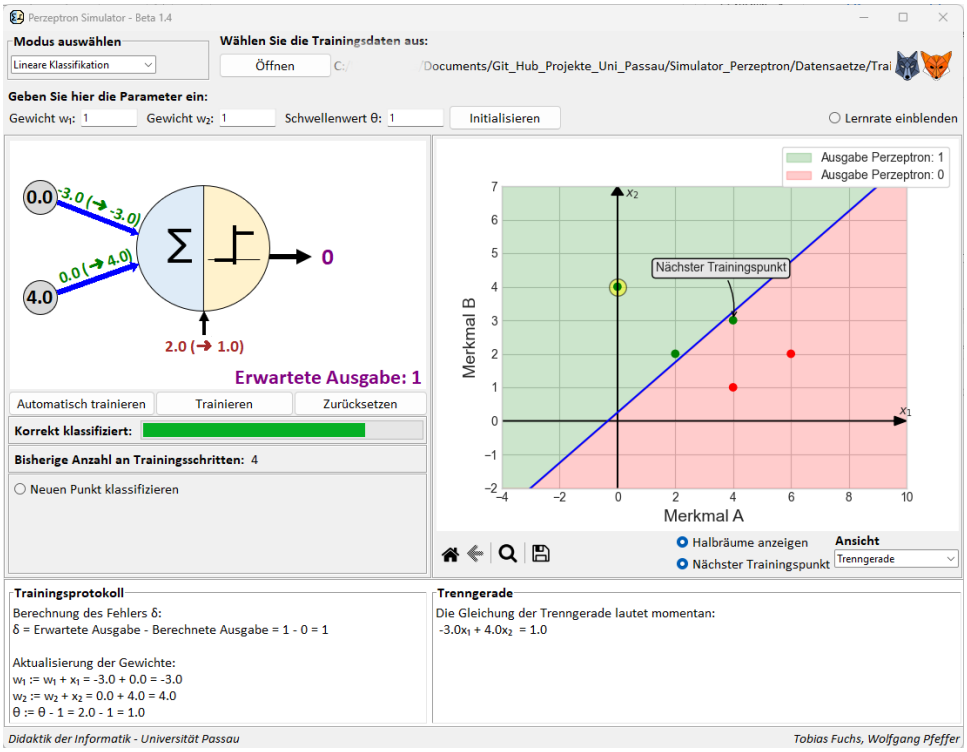


Abb. 5.27: Grafische Oberfläche des Perzeptron-Simulators.

5.3.3 Testen der Implementierung des Perzeptrons

Im Rahmen dieser Handreichung wird eine Java-Klasse zur Verfügung gestellt, mit der Schülerinnen und Schüler ihre eigene Implementierung des Perzeptrons testen können. Hierzu findet man im Materialordner ein BlueJ-Projekt (`ImplementierungPerzeptron Vorlage`), das diese Klasse enthält und an die Schülerinnen und Schülern verteilt werden kann.

Nach dem Öffnen des BlueJ-Projekts steht die Klasse `LABOR` zur Verfügung. Nachdem die Schülerinnen und Schüler die Klasse Perzeptron entsprechend dem Vorschlag in Abschnitt 5.2.2.4

entwickelt haben, können sie ein Objekt der Perzeptron-Klasse erzeugen. Anschließend instanziieren sie ein Objekt der Klasse `LABOR` und übergeben das Perzeptron-Objekt im Konstruktor. Die Klasse `LABOR` hat folgende Methoden:

`ladeTrainingsdaten(Dateiname)`

Hiermit kann eine CSV-Datei geöffnet werden, die gelabelte Daten enthält. Die Formatierung ist kompatibel mit dem Dateiformat für den Demonstrator für maschinelles Lernen (s. Abschnitt 5.3.2.1). Alle Datenpunkte, die nicht mit einer 1 gelabelt wurden, erhalten das Label 0. Der beste Speicherort für die Trainingsdaten ist das Projektverzeichnis, da man hier auf die Angabe eines Dateipfades verzichten kann.

`visualisiere()`

Es wird ein Fenster geöffnet, das die Trainingsdatenpunkte in einem Koordinatensystem anzeigt. Wie beim Demonstrator für maschinelles Lernen findet eine Skalierung statt, um die Datenpunkte übersichtlich anzuzeigen. Dennoch fließen in die späteren Berechnungen die Originaldaten und nicht die skalierten Daten ein. Die Schülerinnen und Schüler können in der Anzeige zusätzlich Trainingsdatenpunkte hinzufügen oder löschen. Die Visualisierung steht auch dann zur Verfügung, wenn keine Trainingsdaten aus einer Datei geladen wurden. Somit können die Schülerinnen und Schüler mit verschiedenen Konstellationen der Datenpunkte experimentieren. Nach dem Trainingsvorgang werden die Halbräume angezeigt, indem das Perzeptron-Objekt alle Punkte des dargestellten Koordinatensystems auswertet und diese dann in der Oberfläche rot bzw. grün eingefärbt werden.

`überprüfePerzeptron()`

Die Methode zeigt anhand einer Bildschirmausgabe die Anzahl an Trainingspunkten, die von dem Perzeptron aktuell falsch klassifiziert werden. Hierbei wird die Methode zur Klassifizierung eines Datenpunktes des Perzeptron-Objekts verwendet.

`trainiereEinmal()`

Das Perzeptron wird einmal mit allen Datenpunkten trainiert, indem jeweils die Lern-Methode des Perzeptron-Objekts aufgerufen wird. Die Reihenfolge der Datenpunkte ist zufällig.

`trainiere(MaximaleAnzahlAnDurchläufen)`

Die Methode `trainiereEinmal()` wird so lange wiederholt, bis alle Trainingspunkte richtig

klassifiziert werden. Überschreitet die Anzahl der Iterationen den Wert, der als Methodenparameter übergeben wurde, so bricht die Wiederholung ab.

Die Arbeit mit der Projektvorlage wird in Abbildung 5.28 dargestellt. Es bietet sich an, dass sich die Schülerinnen und Schüler die Objektkarte des Perzeptrons anzeigen lassen, da sie so direkt mitverfolgen können, wie sich die Attributwerte nach jedem Training ändern. Im Materialordner befindet sich ein Video, in dem die Verwendung der BlueJ-Vorlage vorgeführt wird.

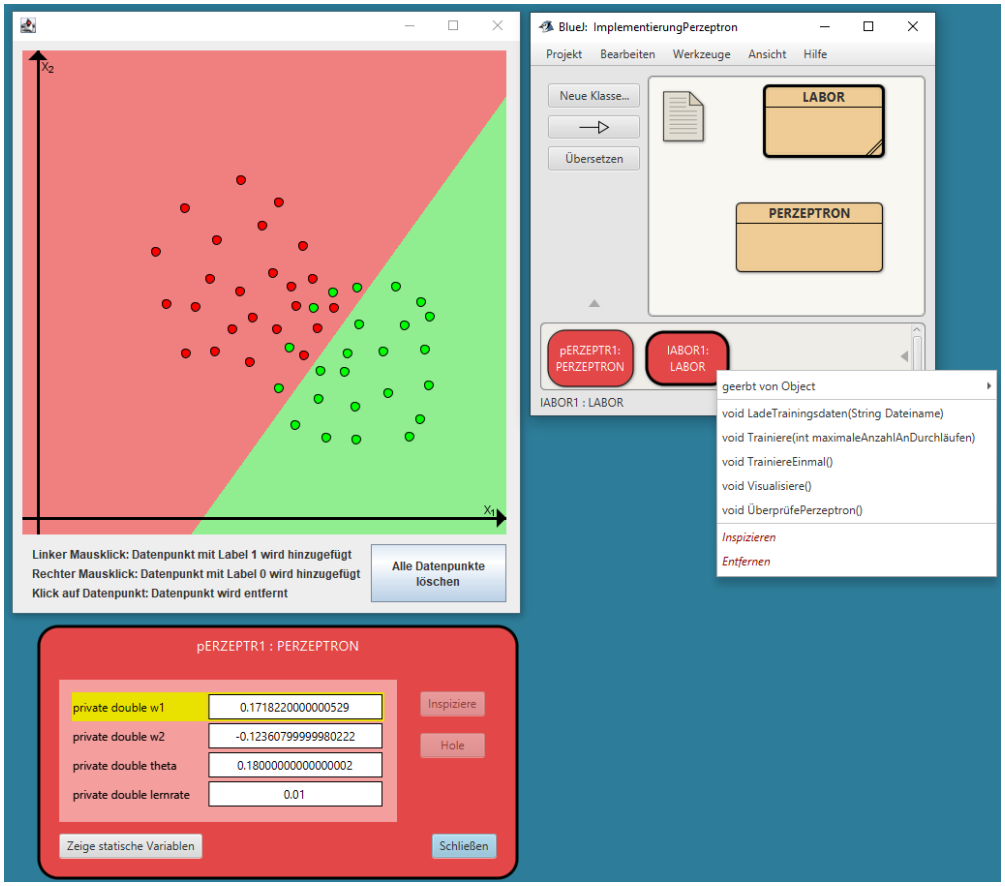


Abb. 5.28: Arbeiten mit der `LABOR`-Klasse der Projektvorlage.

Hinweis:

Sämtliche Bezeichner, wie Klassen- und Methodennamen, sind von der Lehrkraft beziehungsweise den Schülerinnen und Schülern frei wählbar. Das `LABOR`-Objekt erkennt die benötigten Methoden an ihrer Signatur:

Label berechnen:

```
double Methodenname(double Parameter1, double Parameter2)
```

Delta-Lernschritt:

```
void Methodenname(double Param1, double Param2, double Param3)
```

Sollte es zu einer Mehrdeutigkeit kommen, so würde bei der Instanziierung eine Aufforderung erscheinen, die entsprechende Methode anzugeben. Wichtig ist, dass der dritte Parameter der Methode, in dem ein Schritt der Delta-Lernregel durchgeführt wird (s. Abschnitt 5.28), das Label angibt.

5.3.4 Weitere Anwendungen

5.3.4.1 Open Roberta

Hierbei handelt es sich um eine Online-Entwicklungsumgebung (`https://www.open-roberta.org/`), in der Schülerinnen und Schüler mit einer graphischen Programmiersprache arbeiten (s. Abbildung 5.29). Entwickelt wurde das Open-Source-Projekt vom Fraunhofer-Institut IAIS. Teil des sehr umfangreichen Pakets sind auch Module zum Perzeptron beziehungsweise zu künstlichen neuronalen Netzen. Ein virtueller Roboter kann beispielsweise anhand von Sensordaten lernen, wie er Hindernisse umfährt. Der Lehrplanabschnitt, der sich auf das Perzeptron beziehungsweise auf das künstliche neuronale Netz bezieht, lässt sich damit prinzipiell umsetzen. Da das Szenario nur die Robotik betrifft und die Schülerinnen und Schüler vorab Erfahrungen mit dem System sammeln müssen, bevor sie sich mit dem Perzeptron beschäftigen, ist Open Roberta gut für den Lernbereich 5 „Vertiefung“ (NTG Lehrplan) geeignet.

5.3.4.2 Unravel

Ein weiteres Online-Tool, das sich im Lernbereich 5 „Vertiefung“ (NTG Lehrplan) gewinnbringend einsetzen lässt, ist Unravel (`https://klassenkarte.de`). Mit dieser didaktischen Anwendung lassen sich alle Schritte eines Bilderkennungssystems, angefangen von der Aufnahme bis hin zur Klassifikation mit einem künstlichen neuronalen Netz, durchspielen. Trainings- und Testdaten lassen sich von den Schülerinnen und Schülern selbst generieren, da die Anwendung auf eine angeschlossene Kamera (beispielsweise Dokumentenkamera oder Webcam)

zugreifen kann. Das Video `https://www.youtube.com/watch?v=hqJrYbcJ5po` führt in das System ein.

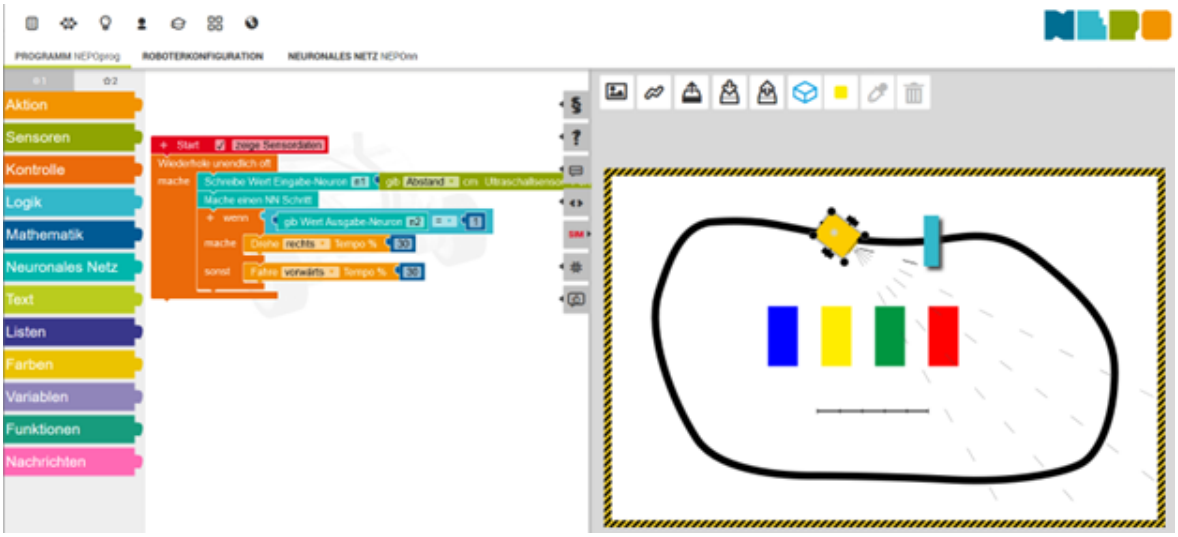


Abb. 5.29: Programmierumgebung des Open Roberta Labs.

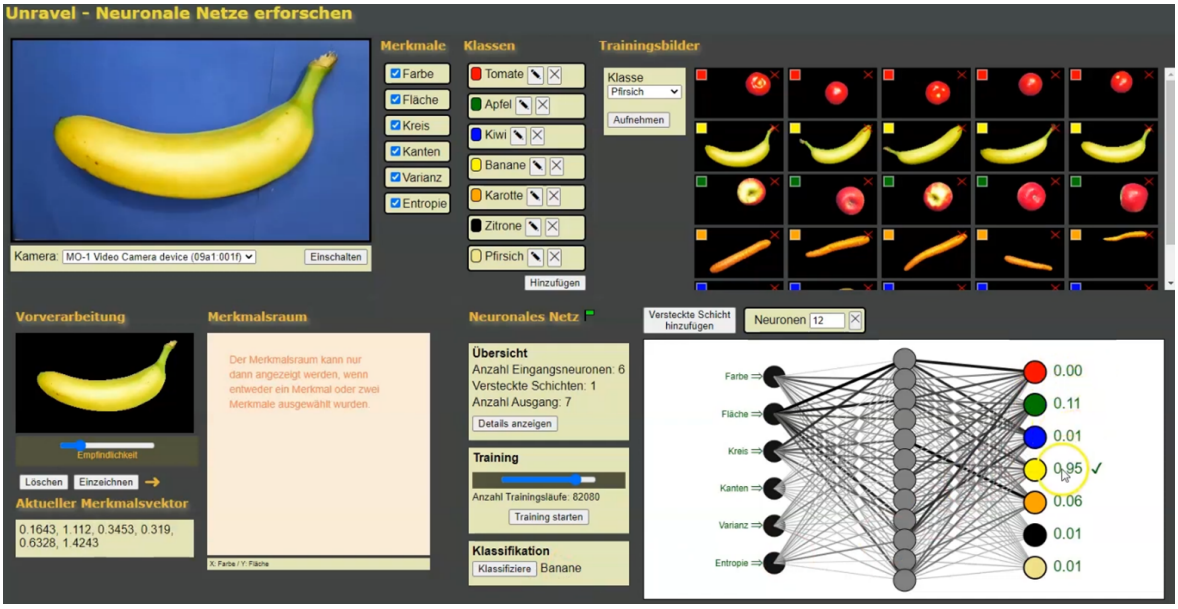
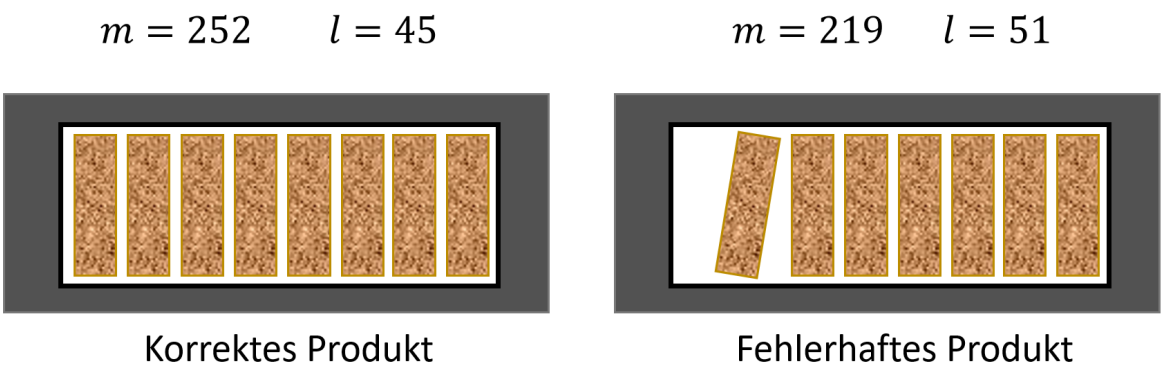


Abb. 5.30: Mit dem Online-Tool Unravel können Schülerinnen und Schüler alle Schritte eines Bilderkennungssystem erleben.

5.3.5 Aufgabenbeispiel für eine Leistungserhebung

In der Fabrik eines Lebensmittelkonzerns werden Schachteln mit jeweils acht Fischstäbchen automatisiert befüllt. Leider kommt es in seltenen Fällen vor, dass in einer Verpackung zu wenig

Fischstäbchen enthalten sind. Daher soll der Prozess künftig mithilfe künstlicher Intelligenz unter Zuhilfenahme eines Perzeptrons überwacht werden. Dabei soll dieses „feuern“, wenn ein fehlerhaftes Produkt erkannt wurde, um es dann auszusortieren. Hierzu misst ein Sensor die Helligkeit des reflektierten Lichts l (Maßeinheit Candela) einer offenen Schachtel mit weißem Boden und eine Waage misst die Masse m der Schachtel in Gramm. Da sich das Produkt bei der Messung auf einem Förderband befindet, sind die Messungen allerdings fehlerbehaftet. Zur Verdeutlichung werden in der folgenden Abbildung zwei Beispiele dargestellt.



(1) Beurteilen Sie, inwieweit die beiden Merkmale „reflektiertes Licht“ l und „Masse“ m für die Aufgabe geeignet sind.

Lösungsvorschlag:

Fehlen Fischstäbchen, dann ist die Masse geringer als diejenige einer vollen Schachtel. Gleichzeitig wird mehr Licht reflektiert, weil ein größerer Teil der Fläche des weißen Schachtelbodens beleuchtet wird. Wenn die Messungen genau genug sind, damit die Datenpunkte linear separierbar sind, dann ist das Perzeptron gut geeignet.

(2) Öffnen Sie den Demonstrator für maschinelles Lernen und verwenden Sie die CSV-Datei. Diese enthält Trainingsdaten, die von dem Programm normalisiert wurden. Bestimmen Sie möglichst geeignete Parameter des Perzeptrons für die normalisierten Trainingsdatenpunkte.

Lösungsvorschlag:

$w_1 = -0{,}26; \quad w_2 = 0{,}34; \quad \theta = 4$

(3) Gegeben ist der skalierte Datenpunkt, der durch die Werte $x_1 = 400$ und $x_2 = 150$ festgelegt ist. Bestimmen Sie die Ausgabe des Perzeptrons aus Teilaufgabe 2 und

interpretieren Sie das Ergebnis.

Lösungsvorschlag:

$$-0{,}26 \cdot 400 + 0{,}34 \cdot 150 \geq 4 \quad \Longleftrightarrow \quad -53 \geq 4$$

Das Perzeptron feuert nicht, da der Schwellenwert nicht überschritten wird. Das bedeutet, das Produkt wird als korrekt klassifiziert.

(4) Das Perzeptron soll unter Zuhilfenahme weiterer gelabelter Daten trainiert werden. Beschreiben Sie den Ablauf eines Schritts mit der Delta-Lernregel. Eine Betrachtung der Lernrate ist nicht gefordert.

Lösungsvorschlag:

Als Erstes wird für einen Trainingsdatenpunkt das Label berechnet. Entspricht das erwartete Label dem berechneten Label, so sind keine Anpassungen der Gewichte und des Schwellenwerts nötig. Ist das erwartete Label kleiner als das berechnete Label, so müssen die Gewichte verringert und der Schwellenwert erhöht werden. Falls das erwartete Label größer als das berechnete Label ist, dann müssen die Gewichte erhöht und der Schwellenwert verringert werden.

KAPITEL 6 Chancen und Risiken für Individuum und Gesellschaft

„KI ist wahrscheinlich das Beste oder das Schlimmste, was der Menschheit passieren kann.“
Stephen Hawking

Überblick:

6.1 Fachliche Grundlagen

6.1.1 Hoffnungen und Sorgen

„Dichten“ und „richten“ (Zweig, 2019, S. 206); für Katharina Zweig sind diese beiden plakativen Schlagworte für Anwendungen der KI mit besonderer Sorge für unsere Gesellschaft verbunden. Wobei diese „Sorgen“ nicht zwangsläufig zu Problemen, schädlichen Entwicklungen und Dystopien führen müssen. Vielmehr zeigen sie gesellschaftliche Handlungsfelder auf, in denen Menschen aktiv werden müssen, um „Hoffnung“ auf positive Entwicklungen zu schaffen.

Die Fähigkeit, dass KI-gestützte Systeme kreative Aufgaben übernehmen können („dichten“), stellt eine der ältesten Ängste der Menschen vor Computersystemen dar. „Plötzlich“ können Maschinen kreative Arbeiten erledigen, für die sie Fähigkeiten benötigen, die bisher nur von Menschen ausgeführt werden konnten. Dieser Entwicklung muss auf arbeits-, bildungs- und sozialpolitischer Ebene (vgl. Zweig, 2019, S. 206) begegnet werden, um eine gesellschaftliche Transformation zu begleiten und eine Disruption zu vermeiden. Der Sorge vor dem „Richten“ muss mit technologischen, juristischen und ethisch-sozialen Aspekten entgegengetreten werden. KI-gestützte Entscheidungssysteme müssen im Hinblick auf Verwendung, Entwicklung und Datennutzung nachvollziehbar gestaltet und genutzt werden. Dann kann sich auch aus diesen Systemen eine Verbesserung im Hinblick auf Effizienz, Gerechtigkeit und Fairness ergeben.

Aus diesen Hoffnungen und Sorgen können daher Chancen und Risiken für Individuum und Gesellschaft abgeleitet werden, die aber nicht zwangsläufig zu „Gutem“ oder „Schlechtem“ führen müssen. Sorgen können so beispielsweise auch mit der Bestimmung entsprechender Regeln oder Rahmenbedingungen ausgeräumt werden, bevor sie zu Risiken werden. Die Hoffnungen werden erst zu Chancen, wenn sie umsetzbar sind und auch die gewünschten Ergebnisse erzielen.

Dieses Kapitel zeigt Ansatzpunkte und Strukturen für eine fundierte, differenzierte und systematische Analyse der Chancen und Risiken von ausgewählten KI-Systemen auf. Damit sollen die Schülerinnen und Schüler befähigt werden, diese System aus technischer Sicht (s. vorherige Kapitel) sowie aus der gesellschaftlichen, staatsbürgerlichen und individuellen Perspektive beurteilen zu können.

6.1.2 Die Rolle des Menschen in KI-Systemen

„Die Zukunft ist vorhersehbar und Morde können durch Künstliche Intelligenz verhindert werden [...][.] Das ist der Stoff für den Science-Fiction-Thriller ‚Minority Report' mit Tom Cruise. Doch Pre-Policing ist längst nicht mehr reine Zukunftsvision aus Hollywood: Die auf Künstlicher Intelligenz basierenden Ermittlungsmethoden sind Teil alltäglicher Polizeiarbeit." (vgl. Bartlett-Mattis, 2021).

6.1.2.1 Beispiel Predictive Policing: SKALA

Die vom nordrhein-westfälischen Landeskriminalamt entwickelte Software SKALA soll Prognosen über Kriminalitätsrisiken abgeben und diese mithilfe der Software SKALA|MAP auf einer Karte visualisieren. Dazu werden neben den polizeilich aufgezeichneten Fällen auch infrastrukturelle Gegebenheiten und soziodemographische Informationen zur Kriminalitätsprävention herangezogen. SKALA wurde 2016 bis 2018 in einem Forschungsprojekt entwickelt. Seither ist es als Instrument zur „Prognose von Kriminalitätsbrennpunkten"[1] fest in der Polizeiarbeit etabliert.

Um zu verstehen, welche Auswirkungen dieses System auf die Gesellschaft und auf einzelne Bevölkerungsgruppen hat, muss der Prozess des Predictive Policing nach Bode et al. (2017) genauer betrachtet werden:

Info (Predictive Policing (Bode et al., 2017, S. 2))

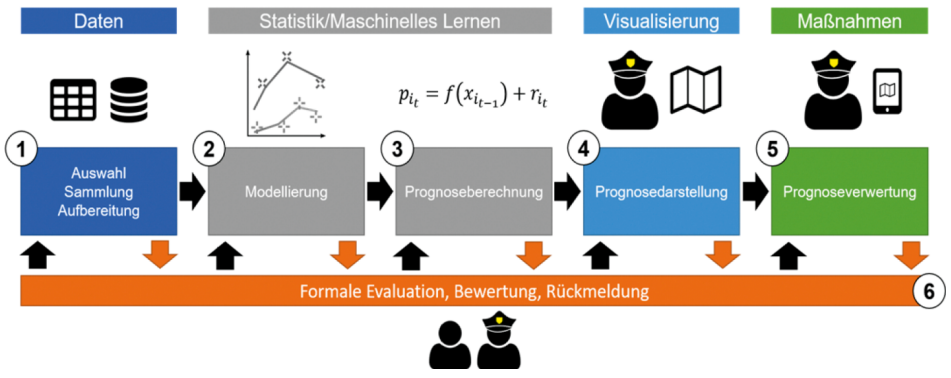


Abbildung 6.1: Ablauf des Predicitve Policing Prozesses (Bode et al., 2017, S. 2).

[1] https://polizei.nrw/artikel/projekt-skala-predictive-policing-in-nrw

Schritt 1: Daten

Der Prozess beginnt im ersten Schritt mit der Sichtung und Auswahl von Datenquellen sowie der Sammlung und Aufbereitung von Datensätzen, die in den weiteren Bestandteilen des Prozesses verarbeitet werden. Mit entsprechender Software ist es möglich, verschiedene Datenquellen miteinander in Beziehung zu setzen. Zentral ist hierbei die raum- und zeitbezogene Zusammenführung. Dabei können polizeiliche Vorgangsdaten mit nicht-polizeilichen Daten (z. B. Daten zur Wetterlage, Wohnlage oder Entfernung zur nächstgelegenen Autobahn) kombiniert werden. In diesem Zusammenhang ist wichtig, dass alle ausgewählten Daten geografisch referenziert werden können, sodass ein einheitlicher, maschinell verarbeitbarer Datensatz vorliegt, der die Basis für Predictive Policing darstellt. [...]

Schritt 2: Modellierung

Zu Beginn wird ein konkretes Modell unter Verwendung der vorliegenden historischen Daten erstellt, um die Kriminalitätslage möglichst angemessen in allen gewünschten Facetten abzubilden. Beispielsweise wäre eine Modellierung mittels Regressionen, Entscheidungsbäumen oder künstlicher neuronaler Netze möglich. [...]

Schritt 3: Prognoseberechnung

Das erstellte Modell wird auf aktuelle bzw. mögliche zukünftige Daten angewendet, um die Wahrscheinlichkeit eines bestimmten Delikts in einer geografischen Bezugsgröße zu ermitteln. Dieser Schritt stellt die eigentliche Prognoseberechnung dar und ist, mit den aus Schritt 2 gewonnen Erkenntnissen, das Herzstück im Predictive-Policing-Prozess. Das Ergebnis der Berechnung präsentiert sich regelmäßig in einer Auswahl an geografischen Räumen, die ein höheres Kriminalitätsrisiko aufweisen als andere Räume im gleichen Prognosezeitraum. [...]

Schritt 4 / 5: Prognosedarstellung und -verwertung

Schritt 4 sieht die adäquate Darstellung der Kriminalitätsprognosen vor, um sie sodann im Feld (meist durch operative Polizeieinheiten) einzusetzen (Schritt 5). [...]

Schritt 6: Evaluation

Dieser Schritt umfasst, bezogen auf den methodischen Predictive-Policing-Prozess, die

durchgehende Evaluation und Bewertung der angewandten Methoden. Es handelt sich um die formale, statistische Beschreibung von beobachteten Effekten, z. B. mit der Berechnung von Trefferraten als auch durchgehende Plausibilitätsprüfungen und Ad-Hoc Verifikationen von Zwischenergebnissen, um beispielsweise die Eignung der gewählten Methode in den Schritten 2 und 3 oder der gewählten Visualisierungstechnik in Schritt 4 kontinuierlich sicherzustellen. [...]

6.1.2.2 Predictive Policing: Hoffnung und Sorge

Mithilfe von Systemen wie SKALA können Polizeibeamte in Echtzeit verdichtete Informationen aus unterschiedlichen Quellen erhalten und für ihre Arbeit nutzen. Damit kann die Kriminalität ggf. sogar schon vor der Entstehung verhindert werden. Ein unparteiisches System, das objektiv die Fakten auswertet und darauf aufbauend Entscheidungen trifft, die der Allgemeinheit nutzen (hier Verbrechen verhindert), ist eine Hoffnung für die Gesellschaft. Wie eingangs bereits angesprochen, sind diese KI-Systeme oftmals auch mit Sorgen verbunden. So erklärte das Bundesverfassungsgericht eine Datenanalyse-Software bei der Polizei in Hessen und Hamburg für verfassungswidrig. Unter anderem bemängelten die Richterinnen und Richter, dass die Art und die Menge der einsetzbaren Daten kaum begrenzt sei: „Die Vorschriften unterscheiden insbesondere nicht nach Daten von Personen, die einen Anlass für die Annahme geben, sie könnten eine Straftat begehen oder in besonderer Verbindung zu solchen Personen stehen, und anderen Personen. Sie lassen eine breite Einbeziehung von Daten Unbeteiligter zu, die deshalb polizeilichen Ermittlungsmaßnahmen unterzogen werden könnten.“[2]

„Ausgewertet werden mit der hessischen Software zunächst einmal nur Daten aus Polizeibeständen. In einer der entsprechenden Datenbanken sind allerdings auch Opfer und Zeugen erfasst – oder jemand, der beispielsweise einmal einen Kratzer am Auto zur Anzeige gebracht hat. Die Gesellschaft für Freiheitsrechte (GFF), die die Überprüfung in Karlsruhe angestoßen hat, sieht außerdem die Gefahr, dass auch externe Daten einfließen, etwa aus sozialen Netzwerken. Das System lade geradezu dazu ein, immer mehr Informationen einzuspeisen.“[2] In diesem Fall stellt sich die Frage, wie objektiv die Entscheidung derartiger Systeme tatsächlich sind. Wie werden diese beeinflusst, oder besser, wer beeinflusst diese Systeme? Um zu analysieren,

[2] https://www.spiegel.de/netzwelt/netzpolitik/bundesverfassungsgericht-schraenkt-einsatz-von-polizei-software-ein-a-6a707d74-fea1-484b-a187-013f827b09c0

an welchen Stellen der Entwicklung und Verwendung von KI-Systemen, wie dem Predicitve Policing, menschliche Entscheidungen Auswirkungen auf das Ergebnis haben, kann die lange Kette der Verantwortlichkeit (s. Abschnitt 6.1.2.3) herangezogen werden.

6.1.2.3 Menschen an Schaltstellen – die lange Kette der Verantwortlichkeiten

Mithilfe der langen Kette der Verantwortlichkeit (s. Abbildung 6.2) können Einflussmöglichkeiten des Menschen auf die Entwicklung und Verwendung von KI-gestützten Entscheidungssystemen identifiziert werden.

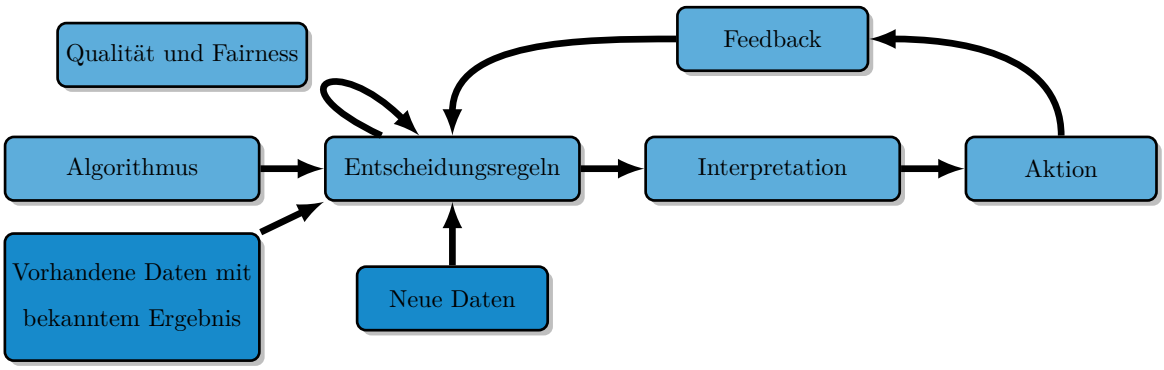


Abbildung 6.2: Lange Kette der Verantwortlichkeiten (Zweig, 2019, S. 29).

Im Folgenden wird bei den Bestandteilen der langen Kette der Verantwortlichkeit von „Fehlern“ gesprochen, die nicht zwingend der Wortbedeutung nach auf falsches Verhalten zurückzuführen sind, sondern auch auf Subjektivität, Vorbelastung oder eigener Vorstellung der beteiligten Personen fußen. Trotzdem können sie die Ergebnisse der KI-Systeme negativ beeinflussen.

Algorithmus

Bereits beim Design und in der Implementierung von Algorithmen können verschiedene handwerkliche Fehler auftreten. Die Möglichkeit, solche Fehler zu finden, hängt wesentlich von drei Aspekten ab:

- Nutzerbasis: Von wie vielen Personen kann der Algorithmus verwendet werden? Es gilt: Je mehr Anwender es gibt, desto wahrscheinlicher ist es, dass ein Fehler entdeckt wird.

- Spezifikation: Wie gut wurde das Verhalten des Algorithmus spezifiziert? Um Fehler erkennen zu können, ist es vor allem wichtig zu wissen, wie der Algorithmus in welchem Fall reagieren sollte.
- Zugänglichkeit: Ist der Sourcecode öffentlich zugänglich? Je mehr Personen Zugang zum Code haben, desto wahrscheinlicher ist es, dass ein Fehler auffällt.

Vorhandene Daten

Es gibt zahllose Beispiele von Datenerhebungen, die fehlerhafte Ergebnisse liefern, zum Beispiel die Verwendung veralteter Datenbestände.

Wurden in einem Wohnviertel in der Vergangenheit viele Verhaftungen durchgeführt, würde dies vom Algorithmus in das Modell einbezogen werden. Hat in den vergangenen Jahren allerdings ein grundlegender Wandel stattgefunden (z. B. durch Abriss und Neubau, der die Sozialstruktur des Viertels verändert hat), würde das nicht in vollem Umfang berücksichtigt werden.

Neue Daten

Bei der Eingabe von neuen Daten nach dem Training können Fehler die Aussagekraft des Modells verzerren. So könnten Daten unvollständig oder fehlerhaft eingegeben werden. Häufig ist der Mensch hier die Fehlerquelle.

Entscheidungsregeln

Hier stellt sich zunächst die grundlegende Frage, ob überhaupt genügend Datenpunkte vorhanden sind, um darin statistisch signifikante Muster zu finden und daraus genügend abstrahierte Regeln abzuleiten. So wird es vermutlich nie möglich sein, einen Algorithmus zu konstruieren, der die Eignung einer Kandidatin oder eines Kandidaten für eine Professur auf der Grundlage ihres oder seines Lebenslaufs stets korrekt vorhersagen kann. Dafür sind die jeweiligen Lebensläufe zu unterschiedlich und die jeweils relevanten Journale oder Konferenzen oder Wirkungsstätten über die Jahre zu volatil, um aussagekräftige Muster zu extrahieren.

Das Erkennen falscher Resultate wird ebenfalls erschwert, wenn das Entscheidungssystem keine für den Menschen einsichtige Erklärung für sein Ergebnis liefern kann (Erklärbarkeit). Dann ist es nicht möglich, eine Person mit einer anderen Person in einer ähnlichen Lage zu vergleichen.

Interpretation

Die Algorithmen des maschinellen Lernens haben aus den Trainingsdaten gelernt, dass in der Vergangenheit bestimmte Eigenschaften von Personen mit ihrem Verhalten korrelierten. Zum Beispiel wird eine Person, die schon mehrfach vorbestraft ist, vermutlich wieder kriminell werden.

Wenn den Nutzern des Systems nicht klar ist, was eine Vorhersage eigentlich ist, nämlich eine gruppenbasierte Wahrscheinlichkeit für ein bestimmtes Verhalten, kann es zu massiven Fehlinterpretationen kommen. Denn eine „Rückfälligkeitsvorhersage von 60 %“ bedeutet, dass die zu bewertende Person einer Personengruppe zugeordnet wurde, von denen 60 % wieder kriminell wurden. Dieser gruppenbasierte Wert wird dann als das individuelles Risiko interpretiert, das Verhalten von Menschen mit ähnlichen Merkmalsausprägungen wird ihnen womöglich zum Verhängnis.

Qualität und Fairness

Im Wesentlichen geht es hier um die Fragen, wie viele Personen von einem solchen System falsch zugeordnet werden (Qualität der Entscheidung) und welche individuellen und gesellschaftlichen Kosten die falsche Einordnung mit sich bringt.

Ist ein Unschuldiger im Gefängnis problematischer als ein Krimineller, der nicht gefasst wird? Da derartige Fragen von entscheidender Bedeutung für die Gesellschaft sind, werden sie in Abschnitt 6.1.3 ausführlich behandelt.

Aktion

Eine grundlegende Frage ist, wer letztlich die Entscheidung trifft und auf welcher Grundlage er dies tut. Was geschieht, wenn die Empfehlung des Algorithmus und die angedachte eigene Entscheidung voneinander abweichen? So könnte ein Richter auch dann der Empfehlung des Algorithmus zu einer Haftstrafe folgen, wenn sie nicht mit der eigenen Einschätzung des Sachverhalts übereinstimmt. Denn die negativen Konsequenzen bei einer Fehlentscheidung, die gegen den Algorithmus getroffen wird, sind ggf. größer als der persönliche Nutzen für den Richter bei einer richtigen Entscheidung entgegen der Empfehlung des Algorithmus.

Auf der anderen Seite könnten sich Kriminelle an das Entscheidungssystem anpassen. Sie können Tipps austauschen, mit welchem Verhalten und welchen Antworten sie als wenig rückfallgefährdet von dem Algorithmus eingestuft werden und somit ihre persönliche Einstufung

verbessern.

Feedback

Manche Algorithmen beinhalten Echtzeitfeedback und können so laufend verbessert werden. Das Feedback kann jedoch auch negative Auswirkungen haben. So kann es zu selbstverstärkenden Feedbackschleifen kommen.

Wird aufgrund von Predictive Policing in einem Viertel häufiger Streife gefahren, führt dies meist zu mehr Festnahmen, weil auch mehr Kleinkriminalität entdeckt wird. Das wiederum hat zur Folge, dass das System „lernt“, dass hier viele Kriminelle leben, was die Anzahl der Streifen weiter erhöhen könnte etc. Hier kommt es zu einer scheinbar objektiven Maßnahme, die die tatsächlich stattfindende Kriminalität jedoch höchst ungleichmäßig verfolgt und daher den Anschein erweckt, dass eine Teilgruppe viel krimineller ist als der Rest der Bevölkerung. Auf diese Weise kann der Einsatz eines Entscheidungssystems zu mehr Ungleichheit führen.

Zusammenfassend kann man festhalten, dass die lange Kette der Verantwortlichkeit bei der Analyse von KI-Systemen im Hinblick auf den menschlichen Einfluss bei der Erstellung und Verwendung genutzt werden kann. Durch die strukturierte Betrachtung der Einfluss- und Steuermöglichkeiten und der daraus entstehenden Fehlerpotentiale durch den Menschen können gesellschaftliche Auswirkungen des Einsatzes sachlich und wertfrei analysiert und beurteilt werden.

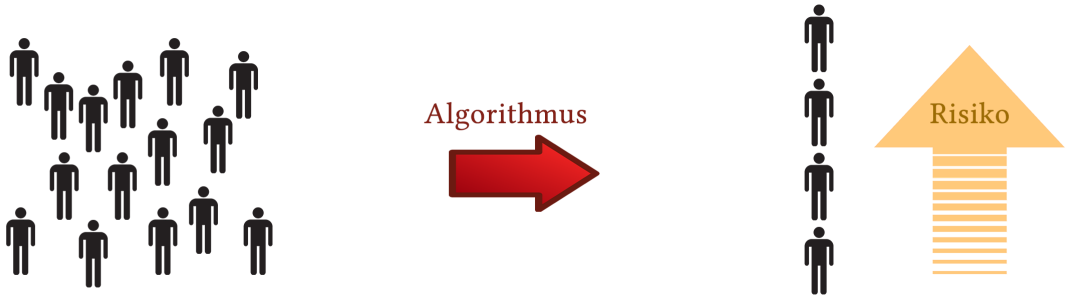
6.1.3 Qualität und Fairness

6.1.3.1 Operationalisierung ethischer Werte

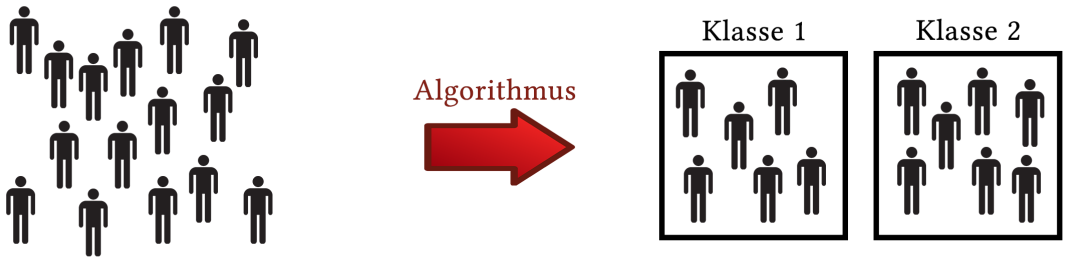
„An KI wird vielfach die Erwartung gestellt, gerechtere oder fairere Entscheidungen zu treffen, als es der Mensch bisher konnte, jedoch müssen diese Erwartungen gedämpft werden. Was fair oder gerecht ist, ist in einer weltanschaulichen Perspektive begründet und kann von einer KI niemals zu voller Zufriedenheit aller entschieden werden“ (Krafft et al., o.J.). Was als fair und gerecht angesehen ist, muss also in der Gesellschaft diskutiert und kann nicht abschließend pauschal definiert werden. Die Systeme des Predictive Policing haben beispielsweise in den USA gänzlich andere Voraussetzungen im Hinblick auf Datenschutz, rechtliche Regelungen und gesellschaftliche Akzeptanz. Daher setzen unterschiedliche Gesellschaften

auch unterschiedliche Anforderungen an diese Systeme. Dabei darf aber die Gesellschaft nicht nur in ihrer Gesamtheit betrachtet werden, sondern es müssen auch betroffene Teilgruppen beachtet werden. Was geschieht, wenn es dazu kommt, „dass die gesellschaftliche Teilhabe der algorithmisch bewerteten Personen behindert wird“ (Zweig, 2019, S. 208)? Oft kommen Verfahren zum Einsatz, bei denen Menschen ohne ihr Wissen oder zumindest ohne ihre Beteiligung in einer der in Abbildung 3 dargestellten Art bewertet werden.

(a) Scoring-Verfahren



(b) Klassifikation



(c) Risikobewertung

Abb. 6.3: Algorithmische Entscheidungssysteme, die Menschen bewerten (Zweig & Krafft, 2018, S. 5).

Beispielsweise weist die Schufa Menschen anhand der bekannten Daten einen Scorewert (s. Abbildung 6.3 (a)) als Maß für ihre Kreditwürdigkeit zu. Je geringer der Wert ausfällt, desto „wahrscheinlicher“ ist ein Kreditausfall. Menschen, die ihr Auto versichern, werden in Schadensfreiheitsklassen (s. Abbildung 6.3 (b)) eingeordnet, was wiederum Auswirkungen auf den Versicherungsvertrag hat. Eine algorithmische Klassifikation von Personen kann auch als Grundlage für eine Vorhersage genutzt werden (s. Abbildung 6.3 (c)). Das System entscheidet, welcher Gruppe von Personen mit bekanntem Ergebnis (0 oder 1) eine neue Person (s. Abbildung 6.3 (c), Ergebnis ? in der Person links) am ähnlichsten ist. „Wenn sich [beispielsweise] herausstellt, dass von den meisten der festangestellten Kreditbewerberinnen mit einem monatlichen Gehalt von mindestens 3.000 € ein Kredit in Höhe von 200.000 € zurückgezahlt wird, sieht es für Antragssteller mit denselben Eigenschaften gut aus. Der Anteil der Personen in dieser Gruppe, die das gesuchte Verhalten aufweisen, wird dann als Wahrscheinlichkeit interpretiert, dass die Person das Verhalten in der Zukunft zeigen wird.“ (Zweig & Krafft, 2018, S. 4).

All diese Systeme haben gemeinsam, dass die bewerteten Menschen keinen Einfluss auf die Gestaltung der zugrunde liegenden Systeme haben. Betroffene können ihren Schufa-Eintrag nicht direkt ändern, ihre Schadensfreiheitsklasse bei der Kfz-Versicherung nicht wechseln und ihre Risikobewertung nicht anpassen. Somit müssen bei der Entwicklung und Verwendung von KI-Systemen nicht nur Wertvorstellungen und rechtliche Rahmenbedingungen einer Gesellschaft beachtet werden, sondern vor allem auch explizit die Gruppen im Fokus stehen, die direkt von der Nutzung dieser KI-Systeme betroffen sind. Dies alles zu gewährleisten, wird noch schwieriger, wenn Algorithmen dafür sorgen, dass Verhaltensweisen automatisch gelernt (s. vorherige Kapitel) werden. Um diese gesellschaftlichen, moralischen und ethischen Aspekte für die KI umsetzbar und für den Menschen nachvollziehbar zu machen, müssen sie operationalisiert und fest im System verankert werden.

Aber wie stellt man fest, ob ein KI-System nach diesen Vorgaben richtig arbeitet und das Ergebnis auch gesellschaftlich akzeptabel ist?

6.1.3.2 Die Frage nach der „guten“ Entscheidung

Um zu erkenne, ob KI-Systeme bereits in der Praxis eingesetzt werden können, muss deren Qualität bestimmt werden. Dazu wurde in den vorhergehenden Kapiteln bereits die Bedeutung

von Trainings-, Validierungs- und Testdaten mehrfach beschrieben. Erst wenn die Systeme für Testdaten zufriedenstellende Ergebnisse liefern, können sie auch für neue Daten eingesetzt werden. Ein System kann in diesem Sinne der Praxis hinreichend gute Ergebnisse liefern, dennoch können die Entscheidungen im Hinblick auf andere Kriterien nicht „gut“ sein.

Fraglich ist dabei, ob „dieselbe Qualität auch für durch das Recht geschützte Teilgruppen gilt, ob also die Entscheidungsqualität nicht zwischen Teilgruppen diskriminiert“ (Zweig & Krafft, 2018, S. 6). Diskriminierung bedeutet dabei die Benachteiligung von Gruppen oder Einzelpersonen (s. Abschnitt 6.1.4), die gleiche Merkmalsausprägungen aufweisen. Im Folgenden soll dies näher beleuchtet werden: Ein einfaches Beispiel zum Einstieg in diese Thematik liefert der bekannte Konflikt zwischen „Equality“ (Gleichheit) und „Equity“ (Gerechtigkeit) (s. Abbildung 6.4).

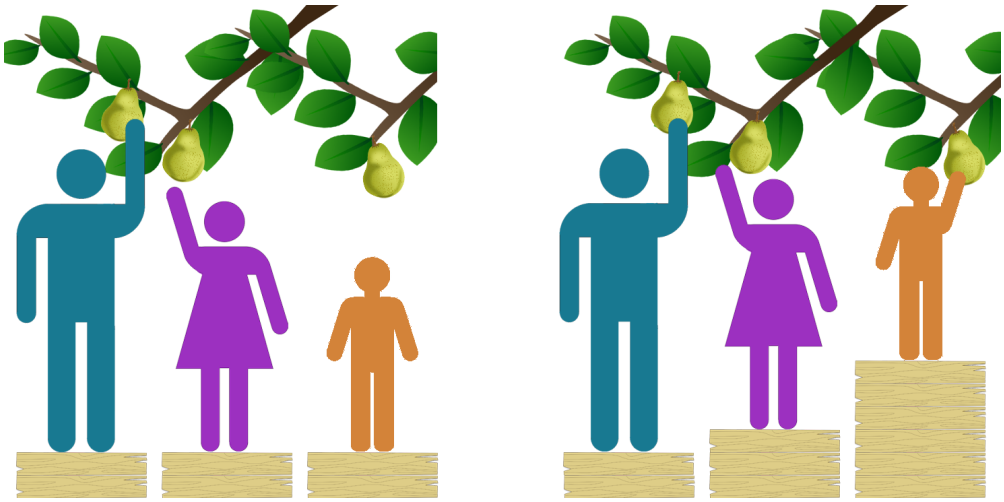


Abb. 6.4: Visualisierung der Begriffe „Equality“ und „Equity“.

Im linken Bild bekommen alle Personen die gleiche Hilfe, nämlich ein gleich großes Podest. Rechts erhalten die Personen jeweils individuell die Hilfe, die sie benötigen, um das Ziel zu erreichen. Aber was ist fair? Ist es fair, dass Eltern in Deutschland unabhängig von Einkommen, finanzieller Konstitution und anderen Lebensumständen den gleichen Betrag an Kindergeld für ihren Nachwuchs erhalten (s. Abbildung 6.4, Equality)? Oder ist es fair, dass im System der sozialen Sicherung (z. B. Sozialhilfe) jeder so viel bekommt, dass sein oder ihr Leben bestritten werden kann (s. Abbildung 6.4, Equity)? Natürlich ist diese Diskussion schon vielfach geführt, aber nie abschließend geklärt worden. Das kann sie auch nicht, da die Entscheidung, ob eine Regelung fair ist, immer von der vorliegenden Maßnahme im jeweiligen gesellschaftlichen

Kontext abhängt. Man kann für eine Maßnahme, beispielsweise das Kindergeld, nur diskutieren, ob als Maß Equity oder Equality angelegt werden soll. Beim Kindergeld hat man sich augenscheinlich dafür entschieden, den Eltern gleich viel Unterstützung zukommen zu lassen, ohne auf die soziale und finanzielle Ausstattung der Familie zu achten. Würde man stattdessen das Maß Equity verwenden, würde die Beurteilung dieser Regelung allerdings anders ausfallen.

Für die Akzeptanz eines KI-Systems ist es von hoher Bedeutung, dass seine Entscheidungen als fair angesehen werden. Nur wenn der Algorithmus Ergebnisse liefert, die von der Gesellschaft als „fair“ empfunden werden, sollte das KI-System in der Praxis eingesetzt werden. Derartige Systeme treffen allerdings nicht für alle Daten stets eine korrekte Entscheidung, sondern geben für neue Eingaben oft nur Wahrscheinlichkeiten für erwartbare Ergebnisse an. Beispielsweise liefert das Fallmanagement- und Entscheidungshilfesystem der US-Justiz COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) eine Wahrscheinlichkeit, mit der ein Angeklagter rückfällig wird. Dabei teilt die Software die Personen zuerst in Risikoklassen ein und nimmt darauf aufbauend die Prognose über ihre weitere Entwicklung vor.

Was dies für das Individuum bedeuten kann, wird in einem Experiment deutlich (vgl. Zweig, 2019, S. 163 ff). Ähnlich wie bei einem Perzeptron wird dabei eine Trennlinie gesucht, die gelabelte Datenpunkte in zwei Gruppen (Kriminelle und Unschuldige) aufteilt. In der Abbildung 6.5 sind Kriminelle und unschuldige Bürger anhand ihrer Ausprägung von zwei abstrakten Kriterien, Kriterium 1 und 2, in einem Koordinatensystem abgebildet.

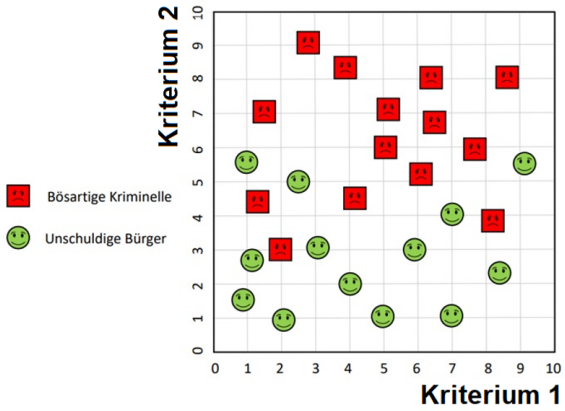


Abb. 6.5: Ausgangslage des Experiments (Zweig, 2019, S. 166).

Nun soll eine Trennline definiert werden, die Kriminelle und unschuldige Bürger „möglichst gut“ in zwei Gruppen teilt. „Möglichst gut“ deshalb, weil es keine Möglichkeit gibt, die beiden

Gruppen linear zu separieren. Grundsätzlich können bei der Trennung zwei extreme Ansätze verfolgt werden:

1. Die Trennlinie wird so gezogen, dass kein Unschuldiger als kriminell eingestuft wird (s. Abbildung 6.6).

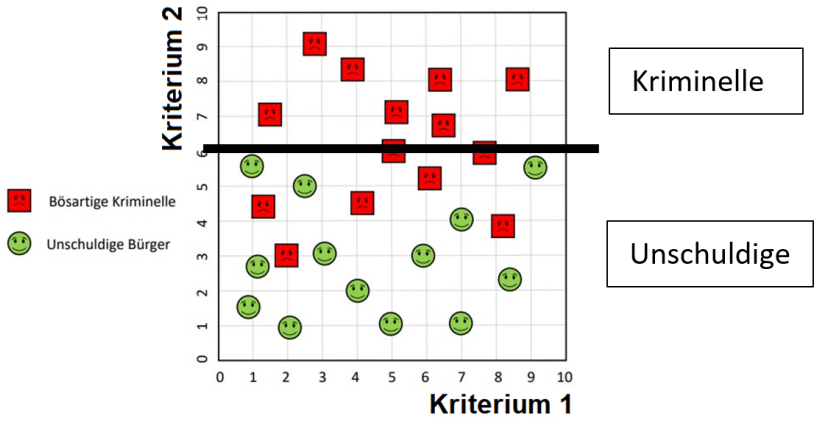


Abb. 6.6: Kein Unschuldiger wird als kriminell eingestuft (Zweig, 2019, S. 166).

Abb. 6.7: William Blackstone.

Bei diesem Ansatz ist es gesellschaftlich akzeptiert, dass nicht alle Kriminelle (s. Abbildung 6.6, rote Punkte unterhalb der Linie) gefasst werden. Ziel ist es, dass kein unschuldiger Bürger zu Unrecht verurteilt wird (s. Abbildung 6.6, keine grünen Punkte oberhalb der Trennlinie). Diesem Ansatz folgte schon der englische Jurist William Blackstone 1760: „*It´s better that ten guilty persons escape than that one innocent suffer*“. Aus Sicht der unschuldigen Bürger scheint diese Handlungsmaxime fair zu sein, da niemand zu Unrecht belangt wird. Der Schutz der Gesellschaft wird dabei dem Schutz der Individuen untergeordnet. Dass Kriminelle fälschlicherweise in Freiheit bleiben, mag bei leichteren Vergehen akzeptabel sein, wird aber bei schweren Straftaten wie Mord wohl eher auf wenig Akzeptanz in der Gesellschaft treffen.

2. Die Trennline wird so gezogen, dass alle Kriminellen verurteilt werden (s. Abbildung 6.8).

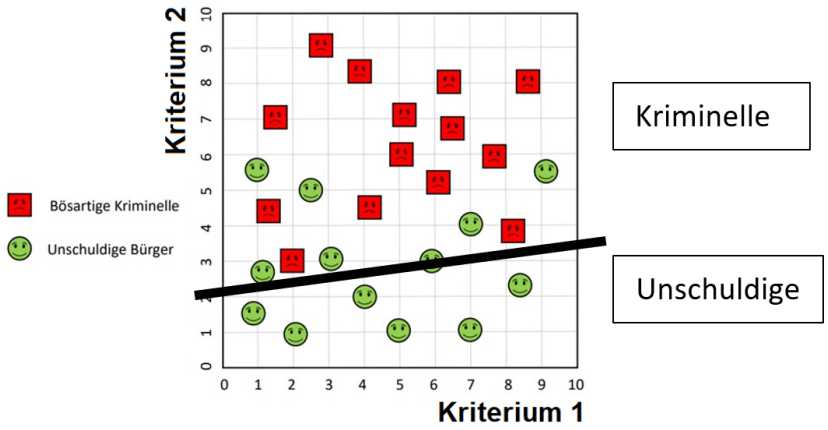


Abb. 6.8: Kein Krimineller wird als unschuldig eingestuft (Zweig, 2019, S. 166).

Dadurch werden alle Kriminellen sicher als kriminell eingestuft (s. Abbildung 6.8, alle roten Punkte über der Trennline), gleichzeitig aber auch einige Unschuldige (s. Abbildung 6.8, grüne Punkte über der Trennlinie). Diese Konfiguration hat zum Ziel, alle Kriminellen zu fassen, akzeptiert aber auch, dass Unschuldige zu Unrecht kriminalisiert werden. Ein Vertreter dieses Ansatzes, Dick Cheney, ehemaliger Vizepräsident der USA, brachte dies am 14.12.2014 in einem Interview so zum Ausdruck:

„*I am more concerned with bad guys who got out and released than I am with a few that, in fact, were innocent*“. Betont werden muss dabei, dass er sich vor allem darauf bezog, dass terroristische Anschläge wie am 11. September 2001 zukünftig verhindert werden sollten. Bei Straftaten mit derartig weitreichenden Konsequenzen für die Bevölkerung scheint es schwer, ihm zu widersprechen. Aufgrund des Risikos wird hier der Schutz des Individuum dem Schutz der Gesellschaft untergeordnet.

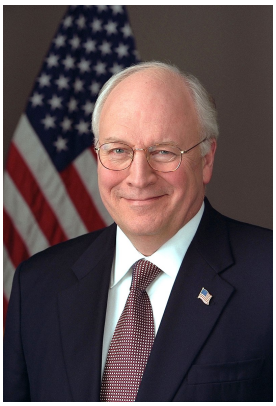


Abb. 6.9: Dick Cheney.

Eine pauschale Anwendung dieses Ansatzes würde allerdings nicht von allen Bürgerinnen und Bürgern getragen.

Obwohl es noch andere Möglichkeiten gibt, diese Trennlinie ggf. „besser“ zu ziehen (z. B. Fehler minimieren), zeigen diese beiden Extreme doch deutlich das Problem der Fairness von KI-Systemen auf. Die für die Erstellung dieser Systeme verantwortlichen Personen müssen sich für eine Positionierung der Trennlinie entscheiden und sich dabei

der Bedeutung für alle Gruppen bewusst sein. Schon am Beispiel der Pilze (s. Abschnitt 3.1.4.2) wurde deutlich, dass auf keinen Fall ein giftiger Pilz als essbar klassifiziert werden darf, wohingegen bei einer großen Auswahl an Bewerbern (s. Kapitel 3.3.1) auch ein qualifizierter Bewerber oder eine qualifizierte Bewerberin fälschlicherweise ausgemustert werden könnte. Zusätzlich muss das Qualitätsmaß aber auch den Benutzerinnen und Benutzern des KI-Systems bekannt sein. Die berechnete Rückfälligkeitswahrscheinlichkeit von COMPAS kann von Richtern im Prozess der Urteilsfindung nur zielführend eingesetzt werden, wenn sie die zugrunde liegenden Annahmen und Arbeitsweisen der Klassifizierung kennen. Daher ist die Beurteilung der Fairness viel komplexer und weniger eindeutig als die reine Qualitätsmessung des KI-Systems.

6.1.4 Diskriminierung

Diskriminierung in KI-Systemen kann nach Zweig (2019) in folgende Arten unterteilt werden.

6.1.4.1 Diskriminierung in den Daten

Bereits die vorliegenden Trainingsdaten können zu Diskriminierung führen. Amazon stellte beispielsweise bereits 2015 fest, dass sein System zur Klassifizierung von Bewerberinnen und Bewerbern für Software-Entwicklung und andere technische Berufe nicht gender-gerecht arbeitete[3]. Das System wurde mit Bewerbungen der letzten zehn Jahre trainiert, welche die überdurchschnittliche Anzahl an Männern in der IT-Branche widerspiegelten. Es wurden deutlich mehr männliche Bewerber als Bewerberinnen eingestellt. Das System lernte anhand dieser Daten, dass Bewerbungen von Männern denen von Frauen vorgezogen werden sollten. Obwohl das KI-System die Angabe des Geschlechts explizit nicht beachtete, wurden Frauen diskriminiert, indem der Algorithmus auch Informationen heranzog, die Auskunft über das Geschlecht, beispielsweise „Kapitän der Frauen-Schachmannschaft“ gaben. Diese Bewerbungen wurden dann schlechter bewertet.

[3] vgl. `https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G`

6.1.4.2 Diskriminierung durch fehlende Daten

Auch das Fehlen von Daten, die das KI-System zum Training seines Modells benötigt, kann zu Diskriminierung führen. Bilderkennungssysteme benötigen beispielsweise bei der Erkennung von Melanomen (schwarzer Hautkrebs ähnlich einem Leberfleck) eine repräsentative Datenbasis, um zuverlässige Ergebnisse liefern zu können. Nur wenn das KI-System mit vielen unterschiedlichen Bildern von Körperteilen und Hautfarben trainiert wurde, kann es in der Praxis eingesetzt werden. Werden einzelne Personengruppen bei der Datenerhebung nicht berücksichtigt, können für diese später auch keine zuverlässigen Vorhersagen getroffen werden. Diese Diskriminierung kann gerade im Gesundheitsbereich besonders schwerwiegende Folgen haben.

Auch beim Training von KI-Systemen mit Spracherkennung müssen im Vorfeld Personengruppen mit Akzenten, Dialekten und Sprachbehinderungen beachtet werden, um deren Möglichkeit der Nutzung des Systems nicht bereits im Vorfeld auszuschließen. Besonders wichtig ist dieses Anwendungsgebiet, da den Sprachinterfaces in den nächsten Jahren eine stark ansteigende Bedeutung zugesprochen wird, sogar mit dem Potential, die gängige Eingabe durch Tastatur und Maus abzulösen (vgl. Zweig (2019), S. 214). Ein humoristisches Beispiel dieser Art der Diskriminierung zeigt der Sketch, in dem zwei Schotten Probleme mit der Sprachsteuerungssoftware eines amerikanischen Lifts haben[4].

6.1.4.3 Diskriminierung durch Weglassen sensitiver Informationen

Benachteiligung kann auch entstehen, wenn dem KI-System sensitive Eigenschaften der Trainingsdaten vorenthalten werden, diese aber ursächlich für unterschiedliches Verhalten sind. Beispielsweise kann im Experiment aus Abschnitt 6.1.3.2 ein für Frauen und Männer (s. Abbildung 6.10, Geschlechtszeichen bei den Punkten) getrenntes System eine zufriedenstellende Vorhersage treffen. Dabei stellt die orange Linie (s. Abbildung 6.10, orange Linie) die optimale Trennlinie für Männer und die blaue Linie (s. Abbildung 6.10, blaue Linie) die optimale Trennlinie für Frauen dar. Würde man hier für Frauen und Männer getrennt jeweils eine eigene Trennlinie bestimmen, würden keine unschuldigen Personen verurteilt und keine schuldigen

[4] `https://www.youtube.com/watch?v=HbDnxzrbxn4`

Personen freigesprochen werden. Das Ergebnis wäre daher für die vorliegenden Daten optimal.

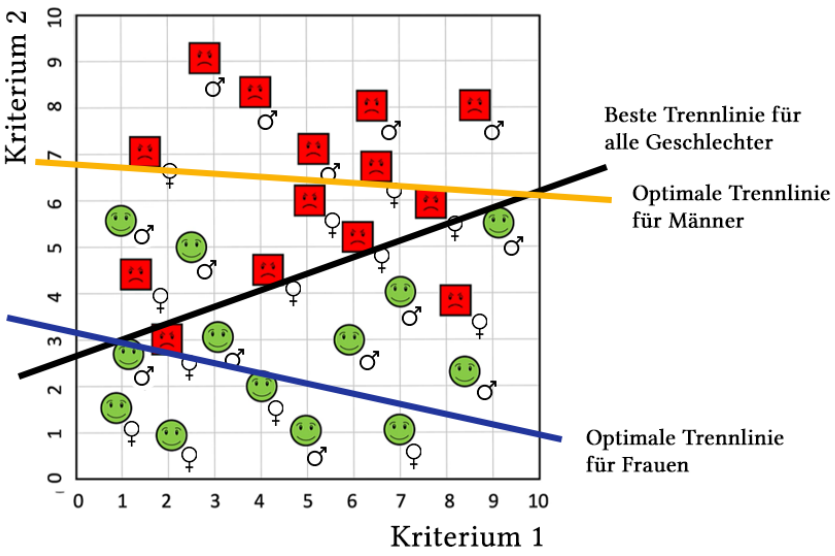


Abbildung 6.10: Experiment mit drei Trennlinien unter Berücksichtigung des Merkmals Geschlecht (Zweig, 2019, S. 217).

Wird aber das Geschlecht bei der Bestimmung der Trennlinie nicht beachtet und nur eine einzige Trennlinie für alle Personen bestimmt, ist das Gesamtergebnis deutlich schlechter. Die in Abbildung 6.10 eingetragene schwarze Linie stellt ein bestes Ergebnis für alle Personen unabhängig von ihrem Geschlecht dar, das aber vier Personen falsch klassifiziert. In diesem Fall sind die zwei zu Unrecht verurteilten Personen männlich (s. Abbildung 6.10, zwei grüne Punkte über der schwarzen Linie) und die beiden nicht verurteilten Kriminellen weiblich (s. Abbildung 6.10, zwei rote Punkte unter der schwarzen Linie). Das Weglassen des Geschlechts wirkt daher diskriminierend.

6.1.4.4 Diskriminierung durch dynamisches Weiterleiten

Eine weitere Form der Diskriminierung kann im Zuge des Lernens der KI-Systeme entstehen. 2016 wurde der Chatbot Tay in die harte Realität des Internets entlassen[5].

Auf Twitter sollte er mit Menschen interagieren und dadurch menschliche Kommunikation und Verhaltensweisen lernen. Dabei konnte er auch selbstständig in Form von Likes, Tweets und Re-Tweets agieren und reagieren. Nachdem er freundlich mit „Hellooooooo World!!!“

[5] `https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist`

startete, radikalisierte er sich innerhalb kürzester Zeit. Neben beleidigenden, rassistischen und sexistischen Aussagen gab Tay auch Verschwörungstheorien weiter. So machte er, bevor er abgeschaltet wurde, US-Präsident George W. Bush für die Terroranschläge am 11. September 2001 verantwortlich und befürwortete die Taten von Adolf Hitler. Eine erschreckende aber erklärbare Entwicklung. Nach dem Motto „garbage in, garbage out“ entstanden diese Ausbrüche durch das Lernen derartiger Aussagen von Personengruppen, die es gezielt auf diesen Chatbot abgesehen hatten. Durch Filterblasen und Verstärkereffekte in den sozialen Medien wurden diese Einflüsse weiter verstärkt. Ein KI-System kann auf diese Weise auch Diskriminierung lernen.

Angesichts dessen stellt sich die Frage, ob die Radikalisierung von Tay hätte verhindert werden können. Mithilfe von Blacklist- und Filteroptionen könnten Themen, Aussagen oder sogar Personen im Lernprozess gezielt vermieden werden. Aber auch diese Filtermöglichkeiten hätten im vorliegenden Fall von Personen mit böswilliger Absicht umgangen werden können. Zusätzlich bietet eine derartige Zensur ebenfalls Diskriminierungspotential. Wer entscheidet, was der Chatbot lernen darf und von wem? Eine Ungleichbehandlung von Teilgruppen der Gesellschaft erscheint hier durchaus möglich.

6.1.5 Rechtliche Fragen am Beispiel der Haftung

Neben technischen und gesellschaftlichen Fragen sind KI-Systeme auch rechtlich reglementiert. Dies kann anschaulich am Beispiel des autonomen Fahrens aufgezeigt werden. Aktuell ergibt sich für den Fahrzeughalter durch die Gefährdungshaftung (§ 7 Straßenverkehrsgesetz) eine verschärfte Haftung. Diese Haftung wird über die Pflichtversicherung (§ 1 Gesetz über die Pflichtversicherung für Kraftfahrzeughalter) abgesichert, da für jedes am Verkehr teilnehmende Kraftfahrzeug eine Haftpflichtversicherung abgeschlossen werden muss. Dies führt dazu, dass Schäden beim Unfallgegner, die durch diese Fahrzeuge entstehen, in jedem Fall reguliert werden. Dennoch ist der Halter verantwortlich für sein Fahrzeug. Das beinhaltet auch, wem er sein Fahrzeug anvertraut, dass er sich um den Zustand kümmert und als Fahrer natürlich die Regeln einhält. Das setzt aber auch voraus, dass er Einfluss auf Fahrer, Fahrverhalten und Zustand des Fahrzeuges hat. Gerhard Wagner von der Humboldt-Universität Berlin sieht hier einen Paradigmenwechsel: „Gegenwärtig fahren Menschen das Auto, aber in Zukunft fährt der Hersteller das Auto und mit dieser Verlagerung der Kontrolle des Fahrzeugs müsste sich

auch die Haftung hin zum Hersteller verlagern“[6]. Warum soll ein Halter eines Fahrzeuges haftbar gemacht werden, wenn dieses doch „autonom“, also selbstständig und unabhängig fährt? Diese Überlegung würde zu einer Produkthaftung führen, bei der ein Hersteller für die Schäden seiner Produkte einstehen muss. Wenn der „Fahrzeugführer“ nicht mehr selbst fährt, sollte also der Fahrzeughersteller für Unfälle seiner Fahrzeuge Schadensersatz leisten. Ein Unfallgegner müsste sich daher zur Regulierung seines Schadens an die Fahrzeugproduzenten wenden. Viele Verbraucherschützer sehen hier Schwierigkeiten bei der Durchsetzung von Ansprüchen von Privatpersonen gegen große Konzerne. Sie sehen daher die Notwendigkeit, die Halterhaftung nicht zu ändern. „Für Geschädigte sei es im Zweifel einfacher, sich direkt an die Pflichtversicherung eines Fahrzeughalters zu wenden, als mit einem Autohersteller in einen Rechtsstreit über fehlerhafte KI-Programmierung zu geraten“[6]. Nichtsdestotrotz müssen rechtliche Regelungen geschaffen werden, die klären, in welchem Rahmen der Versicherungsanbieter Regressansprüche bei den Herstellern der Fahrzeuge im Falle von Fehlern, z. B. durch die Software, geltend machen kann. Aber auch das ist umsetzbar: Schon heute können beispielsweise Hersteller bei Unfällen, die auf Produktionsfehler zurückzuführen sind, haftbar gemacht werden. Ob das bei autonomen Fahrzeugen auch so sein wird, bleibt abzuwarten.

Dieser kurze Exkurs in die rechtlichen Rahmenbedingungen der Haftung im Bereich des autonomen Fahrens soll zeigen, welche rechtlich relevanten Fragen geklärt werden müssen, um bei der Verwendung von KI-Systemen ein friedliches Zusammenleben in der Gesellschaft zu ermöglichen. Sind entsprechende Herausforderungen erfüllt, können die vielen Vorteile von autonomen Fahrzeugen genutzt und die Sorgen vieler Menschen durch klare Regelungen gemildert werden.

6.1.6 Einsatzmöglichkeiten und -beschränkungen

In den vorhergehenden Kapiteln wurde deutlich, welchen Einfluss KI-Systeme auf die Gesellschaft und das Individuum haben können. Daher stellt sich die Frage, in welchen Bereichen des Lebens derartige Software eingesetzt werden soll und welche Voraussetzungen sie dabei erfüllen muss. Zweig (2019) hat dazu eine Risikomatrix mit den Dimensionen Gesamtschaden und Anzahl der Anbieter bzw. Komplexität der Wechselmöglichkeit erstellt (s. Abbildung 6.11), in

[6] `https://www.zdf.de/nachrichten/digitales/ki-strassenverkehr-haftung-autonomes-fahren-100.html`

die diese Systeme eingeordnet werden können.

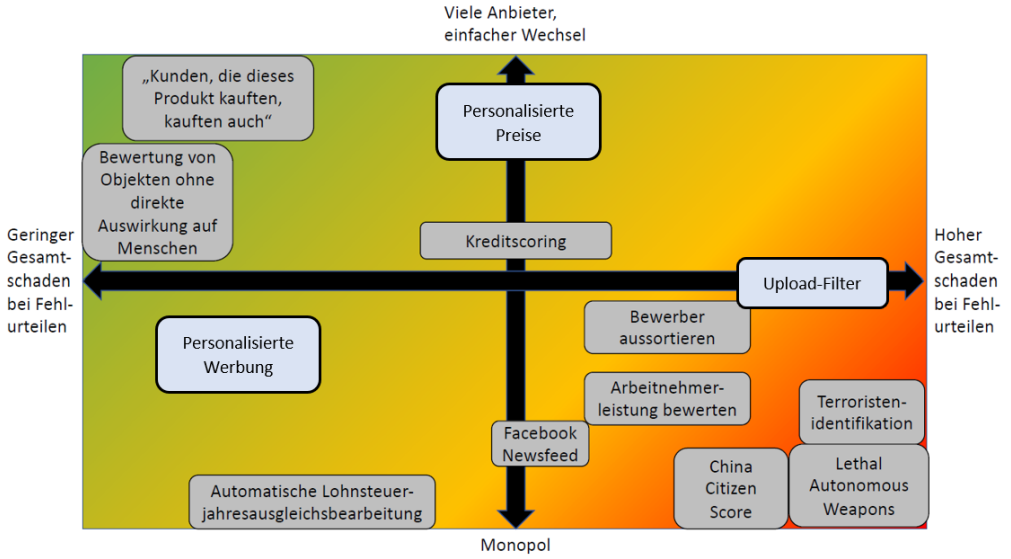


Abb. 6.11: Risikomatrix (Zweig, 2019, S. 242).

Zum einen ist wichtig, wie hoch ein möglicher Schaden bei einer Fehlbeurteilung durch das KI-System für die Gesellschaft ausfallen kann. Die bereits angesprochene Terroristenidentifizierung (s. Abbildung 6.11, rechts unten) kann bei falscher Klassifikation natürlich deutlich stärkere Auswirkungen haben, als einem Kunden eine unpassende Kaufempfehlung zu schicken (s. Abbildung 6.11, links oben). Dabei wird in dieser Darstellung nur die übergeordnete gesamtgesellschaftliche Ebene betrachtet und nicht der individuelle Schaden für einzelne Personen. Dieser Aspekt wurde bereits in Abschnitt 6.1.3.2 thematisiert.

Die zweite Dimension beschreibt die Möglichkeit, den Anbieter einer bestimmten Leistung zu wechseln. Dies kann beispielsweise durch fehlende Alternativen, die durch Patente und Musterschutz entstehen können, oder durch große Marktmacht, die durch sich selbstverstärkende Konzentrationsprozesse bei sozialen Netzwerken und Online-Marktplätzen entstehen, erschwert werden.

Die Lage der KI-Systeme in dieser Risikomatrix erlaubt eine Einordnung in fünf Risikoklassen (s. Abbildung 6.12), „die dann bestimmen, welche Transparenz- und Nachvollziehbarkeitsanforderungen gestellt werden“ (Zweig, 2019, S. 242).

Die Klasse bestimmt dabei, unter welchen Voraussetzungen oder ob überhaupt Systeme mit Künstlicher Intelligenz als Entscheidungssysteme eingesetzt werden sollen:

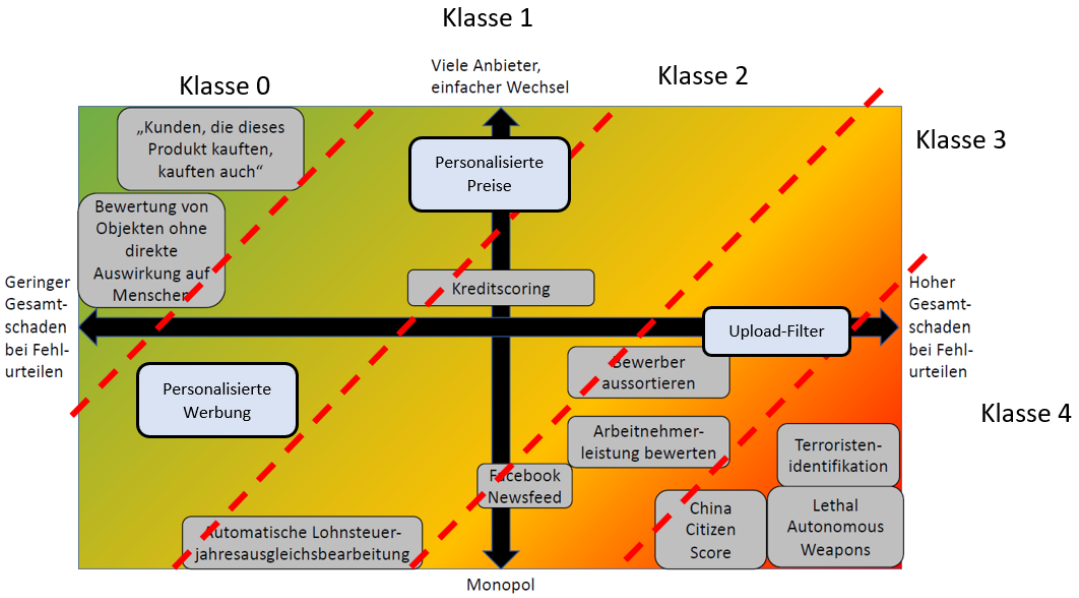


Abb. 6.12: Risikoklassen (Zweig, 2019, S. 244).

Klasse 0: Das Schadenspotential für die Gesellschaft ist bei diesen Systemen so gering, dass keine technische Regulierung notwendig ist. Bewertungs- oder Empfehlungssysteme in Online-Shops sind Beispiele für diese Art der Anwendungen. Sollte bei einer nachträglichen Analyse Diskriminierung einzelner Gruppen auftauchen, wird das System in eine höhere Klasse mit gesteigerten Anforderungen eingeordnet.

Klasse 1: Systeme in dieser Klasse müssen ständig überwacht werden, da ihr Schadenspotential nicht trivial ist. Personalisierte Preise können beispielsweise marktverzerrend wirken und damit einen negativen Effekt auf die gesamte Gesellschaft haben. Personalisierte Werbung kann die öffentliche Meinung und Wahrnehmung beeinflussen und ist daher auch mit potentiellen Gefahren verbunden. Um dem entgegenzuwirken, sollten der Gesellschaft bei diesen Systemen zum einen die verwendeten Methoden des maschinellen Lernens sowie das zugrunde liegende Qualitätsmaß bekannt sein. Zum anderen muss geklärt werden, welche Konsequenzen die Ergebnisse der Systeme für die Gesellschaft haben und ob Widerspruchs- und Ausweichmöglichkeiten gegeben sind.

Klasse 2: Zusätzlich zu den Anforderungen aus Klasse 1 müssen bei diesen Anwendungen auch Informationen über die zugrunde liegenden Daten bekannt sein, um ein transparentes Bild über das Ergebnis des KI-Systems zu erlangen. Dazu gehört auch die Möglichkeit, die Qualität der Ergebnisse selbstständig überprüfen zu können. Bei der Bestimmung

der Kreditwürdigkeit einer Person sollte diese beispielsweise erfahren können, auf welche Daten die sie betreffende Vorhersage beruht.

Klasse 3: Hierin befinden sich KI-Systeme, deren Schadenspotential als sehr hoch eingestuft wird. Dies macht einen Einblick in deren Funktionsweise unbedingt notwendig, um zu erkennen, welche Eigenschaften zu welchen Ergebnissen führen. Systeme in dieser Kategorie müssen mit Verfahren des maschinellen Lernens trainiert werden, „die eine Einsicht in die gefundenen Entscheidungsregeln erlauben“ (Zweig, 2019, S. 243). Dies stellen nicht alle Verfahren des maschinellen Lernens sicher. Deren Ergebnisse können jedoch schlechter sein als ihre Black-Box-Alternativen. Als zusätzliche Einschränkung sollte bei diesen Systemen die Datenbasis auf vorliegende Diskriminierung überprüft werden können. Uploadfilter und News-Feed-Algorithmen haben einen direkten Einfluss auf die politische Meinungsbildung in einem Land und sollten daher transparent und nachvollziehbar agieren. Auch bei Verfahren, die Einfluss auf den Erfolg einer Bewerbung haben (s. auch Abschnitt 3.3.1), sollten Transparenz und Nachvollziehbarkeit für den Bewerbenden, die Mitarbeitenden und die Personalabteilung gewährleistet sein.

Klasse 4: In diesen Bereich fallen KI-Systeme, deren mögliches Schadenspotential als zu groß erachtet wird, um ihren Einsatz in Abwägung mit dem Nutzen rechtfertigen zu können. Tödliche autonome Waffen (lethal autonomous weapons) töten selbstständig Personen, die zuvor von diesem System identifiziert worden sind. Eine „hinreichende Sicherheit“, ob es die „richtige“ Person ist oder die vorliegenden Informationen korrekt sind, wird es allerdings nicht geben. Ein Einsatz dieser Maschinen ist daher grundsätzlich abzulehnen.

Diese Kategorisierung nach Zweig (2019) lässt Raum zur Diskussion über die Klassifizierung einzelner KI-Systeme sowie über die gestellten Anforderungen an KI-Systeme unterschiedlicher Klassen. Es bietet aber eine gute Grundlage, um diese KI-Systeme miteinander zu vergleichen, Auswirkungen auf die Gesellschaft zu bewerten und die Fortentwicklung des rechtlichen Rahmens zu diskutieren.

6.2 Didaktische Hinweise / Bezug zum Lehrplan

6.2.1 Einordnung in den Lehrplan

Im LehrplanPLUS Informatik sowie spät beginnende Informatik findet sich in Jahrgangsstufe 11 folgende Kompetenzerwartung:

> *Die Schülerinnen und Schüler nehmen zu ausgewählten aktuellen Einsatzmöglichkeiten der Künstlichen Intelligenz Stellung und bewerten Chancen und Risiken für Individuum und Gesellschaft.*

In diesem Rahmen sollen die Schülerinnen und Schüler KI-Systeme, die aktuell eingesetzt oder deren Einsatz geplant ist, analysieren. Neben den technischen Details dieser Systeme, welche die Schülerinnen und Schüler im Verlauf dieser Sequenz bereits kennengelernt haben, stehen nun die Auswirkungen auf die gesamte Gesellschaft, Teilgruppen oder Individuen im Vordergrund. Um eine Analyse zu ermöglichen, müssen die Schülerinnen und Schüler zuerst die Funktionsweise von KI-Systemen kennenlernen, die verwendeten Daten überblicken und das verwendete Lernverfahren verstehen. Dafür greifen sie auf die in dieser Sequenz erworbenen Kompetenzen zurück.

Darauf aufbauend können sie ethische, rechtliche und gesellschaftliche Implikationen ableiten. Eine reine Auflistung der Vor- und Nachteile dieser Systeme zu generieren, ist allerdings nicht das Ziel. Vielmehr sollen die Chancen und Risiken verwendet werden, um „abzustecken“, was möglich, umsetzbar und gesellschaftlich erwünscht ist. So soll eine differenzierte mehrdimensionale Bewertung der Zusammenhänge erfolgen, die am Ende nicht ein „Einsatz ja“ oder „Einsatz nein“ bestimmt, sondern Voraussetzungen und Regeln zur erfolgreichen Verwendung dieser Systeme benennt. Dabei sind die Chancen in der Regel schnell bestimmt. Die Risiken dagegen bedürfen einer genaueren Betrachtung. Sie sollen nicht per se als Ausschlusskriterien dieser Systeme dienen, sondern eine Diskussionsgrundlage für die notwendigen Voraussetzungen für den Einsatz im Hinblick auf mögliches Schadenspotential darstellen.

Diese Diskussion kann nur im Kontext des geltenden Rechtssystems sowie der gesellschaftlichen Wertvorstellung unter Verwendung von aktuellen Beispielen erfolgen. Der Operator „bewerten“

zielt dabei (vgl. EPA Informatik[7]) auf die Übertragung vielfältiger Erfahrungen bei der Bearbeitung von Problemen aus verschiedenen Anwendungsfeldern auf die Lösung ähnlicher Fragestellungen ab. Im Zuge dessen müssen die Schülerinnen und Schüler mit Beispielen aus der Realität arbeiten, um die damit erworbenen Erkenntnisse und Fähigkeiten bei der Bewertung in andere Kontexte transferieren zu können. Was sie dazu brauchen, sind allerdings Strukturen, die ihnen eine sachliche, geordnete und multiperspektivische Betrachtung der Problemlage erlauben. Die Schülerinnen und Schüler sollen lernen, wie sie eine Analyse von KI-Systemen im Hinblick auf die Auswirkungen auf Individuum und Gesellschaft fundiert durchführen können.

Dabei steht nicht die Behandlung theoretischer Modelle (z. B. die lange Kette der Verantwortlichkeiten in Abschnitt 6.1.3.2) im Vordergrund. Vielmehr sollen den Schülerinnen und Schülern damit Diskussionsansätze, neuralgische Punkte bei der Entwicklung und Verwendung von KI-Systemen sowie unterschiedliche Betrachtungsperspektiven aufgezeigt werden.

Sind die Chancen gesammelt und die Risiken im Hinblick auf das Schadenspotential abgesteckt, kann zu einem vorgegebenen Einsatz eines konkreten KI-Systems Stellung genommen werden. Diese Fähigkeit ist mithilfe der Strukturen, Perspektiven und Erfahrungen auf andere Probleme, neue Systeme oder Änderungen in den Voraussetzungen transferfähig. Dies ist besonders wichtig, um sich von Diskussionen über „die Künstliche Intelligenz“, die nur oberflächlich geführt werden oder Ängste schüren wollen, abzuheben.

6.2.2 Durchführung

Insgesamt werden für diesen Themenbereich ca. zwei Stunden vorgeschlagen. Um die Schülerinnen und Schüler von den zuvor bearbeiteten technischen Aspekten abzuholen und zu den Folgen dieser Systeme überzuleiten, kann ein Beispiel aus den vorhergehenden Stunden aufgegriffen werden. Es bieten sich dabei besonders Systeme an, die in irgendeiner Form Entscheidungen treffen müssen, wie beispielsweise Systeme der Klassifizierung von Personen.

Eine Umsetzung dieses Themenbereichs kann dabei anhand eines Szenarios (Justizszenario in Abschnitt 6.2.2.1) oder der thementeiligen Bearbeitung mehrerer Szenarien (Beispiele Abschnitt 6.2.2.2) durchgeführt werden. In beiden Fällen sollte den Schülerinnen und Schüler

[7] https://www.kmk.org/fileadmin/Dateien/veroeffentlichungen_beschluesse/1989/1989_12_01-EPA-Informatik.pdf

mithilfe der langen Kette der Verantwortlichkeiten (s. Abschnitt 6.1.3.2), des „Trennlinien-Experiments“ (s. Abschnitt 6.1.3.2), der Arten möglicher Diskriminierung (s. Abschnitt 6.1.4) oder der Einsatzmöglichkeiten und -beschränkungen (s. Abschnitt 6.1.6) Ansatzpunkte zur weiteren Auseinandersetzung vorgestellt werden.

6.2.2.1 Beispiel 1: Szenario Justizsysteme

Zum Einstieg in diese Thematik bietet sich der Ausschnitt aus dem Film Minority Report[8] an, der die scheinbare Utopie einer Strafverfolgung vor der eigentlichen Tat und damit deren Verhinderung zeigt. Dieser Clip zeigt eine Welt frei von Verbrechen, die man sich kaum vorstellen kann. Dass sich diese Welt im Verlauf des Filmes eher in eine Dystopie verwandelt, sei an dieser Stelle dahingestellt. Neben den Chancen, die ein System bietet, das Kriminalität schon im Vorfeld nicht geschehen lässt, erkennen die Schülerinnen und Schüler bereits an dieser Stelle mögliche Probleme.

Das Predictive Policing System SKALA in Nordrhein-Westfalen (NRW) (s. Abschnitt 6.1.2.1) folgt in Zügen dem Ansatz des gezeigten Filmausschnitts. Informationen zu diesem System findet man auf der Homepage der Polizei NRW[9]. Nachdem sich die Schülerinnen und Schüler über die Umsetzung, Voraussetzungen und Ziele dieses Systems informiert haben, können ihnen von der Lehrkraft mit der langen Kette der Verantwortlichkeiten (s. Abschnitt 6.1.3.2), das „Trennlinien-Experiment“, die Arten möglicher Diskriminierung (s. Abschnitt 6.1.4) und der Einsatzmöglichkeiten und -beschränkungen (s. Abschnitt 6.1.6) Ansatzpunkte für eine vertiefte Auseinandersetzung mit diesen Systemen bereitgestellt werden.

Eine Stellungnahme zu diesem Thema könnte anschließend in Form der Methode „Talkshow“ umgesetzt werden: Dazu bearbeiten die Schülerinnen und Schüler die Materialien thementeilig in Gruppen. Jede Gruppe erhält eine Rolle und entwickelt eine Argumentationsstruktur, welche die Chancen aber auch notwendige Voraussetzungen und Bedingungen für die Einführung eines Systems des Predictive Policing aus ihrer Sicht beinhaltet. Dabei kann unter anderem zwischen den Rollen Datenschützer, unschuldiger Bürger, Strafverfolgungsbehörde und Vertreter der Gesamtgesellschaft unterschieden werden. Eine Gruppe stellt den Talkshow-Moderator, der sich ebenso in die Materie einarbeiten und eine Fragenstruktur zur späteren Moderation

[8] `https://www.youtube.com/watch?v=oQdDLfD3kls`
[9] `https://polizei.nrw/skala`

entwickeln muss. Anschließend sendet jede Gruppe einen Vertreter oder eine Vertreterin in die Talkshow-Runde. Hier wird der Einsatz dieser Systeme unter Verwendung der vorbereiteten Argumente diskutiert. Dabei ist zu beachten, dass geforderte Einschränkungen einer Gruppe den Chancen einer anderen Gruppe entgegenstehen können. Dies stellt die Ausgangslage für eine multiperspektivische und dynamische Diskussion dar. Optional kann ein Gesamtergebnis, das aus einem Katalog aus Voraussetzungen besteht, erstellt werden.

Material z. B.:

Polizei NRW: Projektbeschreibung SKALA	`https://polizei.nrw/skala`
Polizei NRW: Abschlussbericht mit Datenschutzinformationen: Ausführlicher Bericht oder Kurzfassung	`https://www.sueddeutsche.de/panorama/polizei-duesseldorf-nrw-polizei-verteidigt-umstrittene-palantir-software-dpa.urn-newsml-dpa-com-20090101-210503-99-449227`
Urteil zur Einschränkung der Software Palantir 2023	`https://www.spiegel.de/netzwelt/netzpolitik/palantir-programme-bundesverfassungsgericht-schraenkt-einsatz-von-polizei-software-ein-a-6a707d74-fea1-484b-a187-013f827b09c0`

6.2.2.2 Beispiel 2: Weitere Szenarien

Im Folgenden finden sich Materialien für Szenarien, bei denen Chancen und Risiken von KI-Systemen deutlich werden:

Sprach- und Texterkennung:

Chancen: z. B.

- `ChatGPT` (`https://openai.com/blog/chatgpt`)
- Automatische Terminvereinbarung (`https://www.youtube.com/watch?v=kMu_cO-P6u0`)

- Übersetzung (`https://www.deepl.com/translator`)

Risiken: z. B.

- Diskriminierung aufgrund des Dialekts bzw. Akzents (`https://www.youtube.com/watch?v=HbDnxzrbxn4&t=144s`)
- Gefahren durch Smart Speaker (`https://www.swr.de/unternehmen/audiolab/hoert-mich-mein-smart-speaker-ab-100.html`)

Gesichts- und Emotionserkennung:

Chancen: z. B.

- Verbrechensbekämpfung (`https://www.br.de/fernsehen/ard-alpha/sendungen/campus/doku/verbrechen-sicherheit-ueberwachung-campus-doku-100.html`)
- Autismus Erkennung (`https://www.youtube.com/watch?v=YQpTlnWYAqE`)

Risiken: z. B.

- Gesichtserkennung in China (`https://www.youtube.com/watch?v=iT9Xq77H5rU`)
- Täuschung der Gesichtserkennung bei Smartphones (`https://www.test.de/Sicherheitsluecke-bei-Smart-phones-Handysperre-mit-Foto-ausgetrickst-5908999-0/`)

Bibliographie

Bartlett-Mattis, M. Künstliche Intelligenz in der Verbrechensbekämpfung. 2021. Online verfügbar unter `https://www.trendreport.de/kuenstliche-intelligenz-in-der-verbrechensbekaempfung/`, zuletzt aufgerufen am 15. März 2023.

Bauberger, S. *Welche KI? Künstliche Intelligenz demokratisch gestalten.* Hanser Verlag, 2020.

Bode, F., Stoffel, F., & Keim, D. Variabilität und Validität von Qualitätsmetriken im Bereich von Predictive Policing. 2017. Online abrufbar unter `https://kops.uni-konstanz.de/server/api/core/bitstreams/ae4f1fb4-747f-414d-92fe-d773a256c1d4/content`, zuletzt aufgerufen am 07. April 2023.

Brinda, T., Diethelm, I., Gemulla, R., Romeike, R., Schöning, J., Schulte, C., & Al., E. Dagstuhl-erklärung: Bildung in der digitalen vernetzten welt. Technical report, mar 2016. URL `https://dagstuhl.gi.de/dagstuhl-erklaerung`.

Brockhaus-Enzyklopädie-Online. *Intelligenz.* o.J. Aufgerufen unter `https://brockhaus.de/ecs/enzy/article/intelligenz-psychologie` am 28.02.2023.

Cunningham, P. & Delany, S. k-Nearest Neighbour Classifiers. *Mult Classif Syst*, 54, 2007.

Demšar, J., Curk, T., Erjavec, A., Črt Gorup, Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., & Zupan, B. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353, 2013. URL `http://jmlr.org/papers/v14/demsar13a.html`.

Duden-online. *Intelligenz.* o.J. Aufgerufen unter `https://www.duden.de/rechtschreibung/Intelligenz` am 28.02.2023.

Ertel, W. *Grundkurs Künstliche Intelligenz.* Springer Verlag, 5. Auflage, 2021.

Europäisches Parlament. Was ist künstliche Intelligenz und wie wird sie genutzt?, 2020. Aufgerufen unter `https://www.europarl.europa.eu/news/de/headlines/priorities/kunstliche-intelligenz-in-der-eu/20200827STO85804/was-ist-kunstliche-intelligenz-und-wie-wird-sie-genutzt` am 28.02.2023.

Fisher, R. A. Iris data set. 1936. Online verfügbar unter `https://archive.ics.uci.edu/ml/datasets/Iris`, zuletzt aufgerufen am 10. März 2023.

Gabler-Wirtschaftslexikon. *Intelligenz*. o.J.a. Aufgerufen unter `https://wirtschaftslexikon.gabler.de/definition/intelligenz-37696` am 28.02.2023.

Gabler-Wirtschaftslexikon. *Lernen*. o.J.b. Aufgerufen unter `https://wirtschaftslexikon.gabler.de/definition/lernen-41169` am 01.03.2023.

Herbold, S. *Data-Science-Crahskurs*. dpunkt.verlag, 2022. Online verfügbar unter `https://data-science-crashkurs.de/`, zuletzt aufgerufen am 11. Februar 2023.

Kaplan, A. & Haenlein, M. Siri, siri in my hand, who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence. *Business Horizons*, 62 (1):15–25, 2019.

Krafft, T., Hauer, M., & Völkl, H. KI und Ethik, o.J. Online aufrufbar unter `https://colab-digital.de/initiativen/koki/ethik/`, zuletzt aufgerufen am 16. März 2023.

Lämmel, U. & Cleve, J. *Künstliche Intelligenz*. Hanser Verlag, 5. Auflage, 2020.

Ramge, T. *Mensch und Maschine: Wie Künstliche Intelligenz und Roboter unser Leben verändern*. Reclam, 6. Auflage, 2018.

Rich, E. *Artificial Intelligence*. Number Bd. 1. McGraw-Hill, 1983.

Rokach, L. & Maimon, O. *Data Mining with decision trees. Theory and Applications*. World Scientific, 2. Auflage, 2015.

Rosenblatt, F. On the convergence of reinforcement procedures in simple perceptrons. *Report, Cornell Aeronautical Laboratory*, 1960.

Russel, S. & Norvig, P. *Künstliche Intelligenz – Ein moderner Ansatz*. Pearson Verlag, 3. Auflage, 2012.

Russel, S. & Norvig, P. *Artificial Intelligence - A Modern Approach*. Pearson, 4. Auflage, 2022.

Rutkowski, L., Jaworski, M., & Duda, P. *Stream Data Mining: Algorithms amd Their Probabilistic Properties*. Springer Verlag, 2020.

Simmons, A. & Chappell, S. Artificial intelligence – definition and practice. *IEEE Journal of Oceanic Engineering*, 13 (2):14–42, 1988.

Simon, W. *Künstliche Intelligenz: Das Wichtigste, was Du wissen musst*. BoD – Books on Demand, 2021.

Winston, P. *Artificial Intelligence*. Addison-Wesley Pub. Co., 3. Auflage, 1992.

Zimbardo, P. *Psychologie*. Springer Verlag, 6. Auflage, 1995.

Zweig, K. & Krafft, T. Fairness und Qualität algorithmischer Entscheidungen. *Social Science Open Access Repository (SSOAR)*, 2018. Online aufrufbar unter `https://www.ssoar.info/ssoar/handle/document/57570`, zuletzt aufgerufen am 16. März 2023.

Zweig, K. *Ein Algorithmus hat kein Taktgefühl.* Wilhelm Heyne Verlag, 2019.

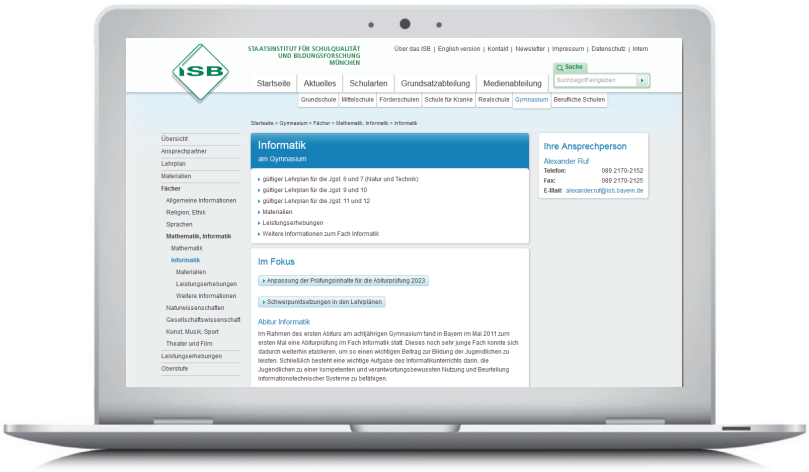
Bildquellen

Weitere Informationen

» www.km.bayern.de

» Handreichung: mebis.link/gwrGdq

» Materialien: mebis.link/NFF6EV



Herausgeber
Bayerisches Staatsministerium für Unterricht und Kultus,
Abteilung Gymnasium, Salvatorstraße 2, 80333 München

Diese Handreichung wurde im Auftrag des Bayerischen Staatsministeriums für Unterricht und Kultus vom Arbeitskreis „Künstliche Intelligenz“ am Staatsinstitut für Schulqualität und Bildungsforschung (ISB) erarbeitet.

Leitung des Arbeitskreises

Alexander Ruf	ISB München

Mitglieder des Arbeitskreises

Marco Hegmann	Illertal-Gymnasium Vöhringen
Christoph Gräßl	Donau-Gymnasium Kehlheim
Dr. Wolfgang Pfeffer	Dominicus-von-Linprun-Gymnasium Viechtach
Johannes Wintermeier	Anton-Bruckner-Gymnasium Straubing

In Kooperation mit

Markus Pentz	Akademie für Lehrerfortbildung und Personalführung (ALP)

Anschrift
Staatsinstitut für Schulqualität und Bildungsforschung
Abteilung Gymnasium – Referat Informatik
Schellingstraße 155, 80797 München
Tel.: 089 2170-2152
E-Mail: kontakt@isb.bayern.de
Internet: www.isb.bayern.de

Cover-Abbildung
© iStockphoto.com/peshkov

Gestaltung
Dr. Wolfgang Pfeffer

Stand
April 2023

Hinweis: Diese Druckschrift wird im Rahmen der Öffentlichkeitsarbeit der Bayerischen Staatsregierung herausgegeben. Sie darf weder von Parteien noch von Wahlwerbern oder Wahlhelfern im Zeitraum von fünf Monaten vor einer Wahl zum Zwecke der Wahlwerbung verwendet werden. Dies gilt für Landtags-, Bundestags-, Kommunal- und Europawahlen. Missbräuchlich ist während dieser Zeit insbesondere die Verteilung auf Wahlveranstaltungen, an Informationsständen der Parteien sowie das Einlegen, Aufdrucken und Aufkleben parteipolitischer Informationen oder Werbemittel. Untersagt ist gleichfalls die Weitergabe an Dritte zum Zwecke der Wahlwerbung. Auch ohne zeitlichen Bezug zu einer bevorstehenden Wahl darf die Druckschrift nicht in einer Weise verwendet werden, die als Parteinahme der Staatsregierung zugunsten einzelner politischer Gruppen verstanden werden könnte. Den Parteien ist es gestattet, die Druckschrift zur Unterrichtung ihrer eigenen Mitglieder zu verwenden.

BAYERN | DIREKT
Tel. 089 12 22 20

BAYERN | DIREKT ist Ihr direkter Draht zur Bayerischen Staatsregierung. Unter Telefon 089 122220 oder per E-Mail unter direkt@bayern.de erhalten Sie Informationsmaterial und Broschüren, Auskunft zu aktuellen Themen und Internetquellen sowie Hinweise zu Behörden, zuständigen Stellen und Ansprechpartnern bei der Bayerischen Staatsregierung.