Nils Röttger · Gerhard Runze · Verena Dietrich

# Basiswissen

# KI-Testen

Qualität von und mit KI-basierten Systemen

Aus- und Weiterbildung zum
Certified Tester AI Testing
- Foundation Level Specialist
- nach ISTQB®-Standard

dpunkt.verlag

# Einleitung

*In diesem Kapitel erzählen wir, wie es dazu kam, dieses Buch zu schreiben. Außerdem geben wir dir eine Art Bedienungsanleitung an die Hand, um dieses Buch zu lesen und damit zu arbeiten: Wo steht was, warum machen wir dies oder jenes? Und wir beschreiben das Zusammenspiel zwischen dem ISTQB®-Lehrplan [GTB 2021] und diesem Buch. Fachliche Inhalte findest du ab Kapitel 1.*

## Wie es dazu kam, dieses Buch zu schreiben

Als die Idee entstand, dieses Buch zu schreiben, haben wir alle drei bei der imbus AG in Möhrendorf gearbeitet. imbus beschäftigt sich seit vielen Jahren u.a. mit der Qualitätssicherung (QS) von IT-Systemen. Um immer auf dem Laufenden zu bleiben, gibt es jedes Jahr neue interne Projekte, die sich mit neuen Technologien, neuen Methoden, neuen Testvorgehen etc. befassen. Vor etwa fünf Jahren wurde die Idee geboren, sich in diesem Rahmen intensiv dem Thema »künstliche Intelligenz« (KI) zu widmen. Kurz zuvor hatte Nils sich mit dem Thema Ethik im Kontext von QS für KI auseinandergesetzt. Nils wollte sein Wissen folglich in dieses Projekt einbringen. Gerhard hatte in seinem Berufsleben immer mal wieder mit KI zu tun und war ebenfalls von Beginn an dabei. Verena hatte sich bereits während ihres Studiums mit dem Thema künstliche Intelligenz beschäftigt. Ihre Expertise in diesem Bereich passte also sehr gut ins Team. Das interne Forschungsprojekt hatte und hat bis heute das Ziel, in das Thema Testen von und Testen mit KI einzutauchen. Wir wollten und wollen neue Methoden finden und etablieren, um erfolgreich KI-Systeme qualitätssichern zu können. Unsere Zusammenarbeit lief gut und wir schrieben gemeinsam Artikel oder hielten Vorträge auf Konferenzen. Verena und Gerhard übernahmen dann auch die KI-Schulung zum »A4Q AI and Software Testing« [A4Q 2019], die seit Dezember 2020 bei imbus angeboten wurde. Als dann im ISTQB® (International Software Testing Qualifications Board, *istqb.org*) der neue Lehrplan zum KI-Testen erstellt wurde und dessen Review anstand, engagierte sich Gerhard dort.

Wir flachsten immer mal wieder darüber, ein gemeinsames Buch zum Thema KI und QS zu schreiben. Aus diesen Scherzen wurde dann irgendwann Ernst. Wir hoffen, dass das Ergebnis nicht nur uns, sondern auch dir gefällt.

## Wie verhält sich das Buch zum Lehrplan des ISTQB®?

Ende des Jahres 2021 hat das ISTQB® den Lehrplan zum »Certified Tester AI Testing« (CT-AI) veröffentlicht [ISTQB 2021a]. Dieser beinhaltet im Wesentlichen zwei Themen:

- Das Testen *von* KI-basierten Systemen
- Das Testen *mit* KI-basierten Systemen, also die Testunterstützung durch KI-basierte Systeme

Das Buch deckt beide Bereiche ab. Da sich unser beruflicher Alltag aber viel häufiger um den Test *von* KI-basierten Systemen dreht, konzentrieren wir uns in diesem Buch – wie auch der Lehrplan selbst – darauf. Hier werden wir die einzelnen Passagen mit Praxisbeispielen aus unserem Projektalltag untermalen, viele Übungen beschreiben und Lösungsbeispiele aufzeigen.

Wir haben den Anspruch, dass dieses Buch dabei hilft, eine Schulung für den CT-AI mit anschließender Zertifizierungsprüfung erfolgreich zu absolvieren, und zugleich ein Nachschlagewerk für den Arbeitsalltag von Testerinnen und Testern darstellt. Häufig ist ein Lehrplan allein nicht geeignet, um den Kurs im Selbststudium zu absolvieren. In diesem Fall kann das Buch als Begleitmaterial zu einer Schulung oder zum Selbststudium verwendet werden. Das Buch wurde von uns so konzipiert, dass es die Inhalte des Lehrplans abdeckt, diese entsprechend mit weiteren Informationen anreichert sowie die im Lehrplan genannten Übungen aufzeigt und dabei hilft, sie zu meistern. An manchen Stellen erschien es uns sinnvoll, zusätzliche Übungen einzubauen.

Aufgrund unserer Erfahrungen als Trainerin und Trainer sowohl für den ISTQB® CT-AI als auch dessen Vorgänger »A4Q AI and Software Testing« [A4Q 2019] sind wir sicher, dass dieses Buch eine gute Unterstützung bei der Vorbereitung auf die Zertifizierungsprüfung sein wird. Um die eigenen Kenntnisse zu überprüfen, empfehlen wir die Bearbeitung der zusammen mit dem Lehrplan veröffentlichten ISTQB®-CT-AI-Übungsfragen [ISTQB 2023]. Die originale Veröffentlichung des Lehrplans ist in Englisch verfasst. Das German Testing Board (GTB) hat im darauffolgenden Jahr sowohl den Lehrplan als auch die Übungsfragen in deutscher Sprache veröffentlicht (vgl. [GTB 2021]; [ISTQB 2023]).

Die Kapitel in diesem Buch stimmen mit den zugehörigen Kapiteln des ISTQB®-CT-AI-Lehrplans überein. An einigen Stellen haben wir in den (Unter-)Kapiteln selbst Exkurse eingefügt. Diese ändern jedoch nichts an der Übereinstimmung der einzelnen Abschnitte mit dem Lehrplan. Die Schlüsselbegriffe für die einzelnen Kapitel führen wir in Englisch und Deutsch auf.

## Das ISTQB®- und das Certified-Tester-Ausbildungsschema

Das International Software Testing Qualifications Board (ISTQB®) ist eine global agierende Organisation, die das erfolgreichste Ausbildungs- und Zertifizierungsschema zum Testen von Software entwickelt hat. Es ist in regionalen bzw. nationalen Boards organisiert. Im deutschsprachigen Raum sind dies das Austrian Testing Board (ATB), das German Testing Board (GTB) und das Swiss Testing Board (STB). Das Ausbildungsschema des ISTQB® ist in drei Stufen organisiert: Foundation Level, Advanced Level und Expert Level. Innerhalb jeder Stufe gibt es Lehrpläne zu mehreren Themenbereichen rund um den (Software-)Test, die von ehrenamtlich tätigen Autorinnen und Autoren aus Industrie und Forschung erstellt werden. Neben diesen drei Stufen gibt es weitere Spezialmodule und Module zum Thema »Testen in agilen Projekten«.

Weitere Informationen zur Organisation, dem Ausbildungs- und Zertifizierungsschema sowie den Lehrplänen finden sich auf der Webseite des ISTQB® [URL: ISTQB] sowie auf den Seiten der nationalen Testing Boards (vgl. [URL: ATB]; [URL: GTB]; [URL: STB]).

Für den ISTQB® »Certified Tester Foundation Level« (CTFL) wurde im August 2023 ein neuer Lehrplan in der Version 4.0 – zunächst in englischer Sprache – veröffentlicht [GTB 2023]. In diesem sind beispielsweise Begriffe wie *Testorakel* entfallen. Der aktuelle CT-AI-Lehrplan bezieht sich noch auf den CTFL in der Version 3.1. Für dich als Leserin oder Leser ist es daher wichtig, zumindest für die Zertifikatsprüfung die Begrifflichkeiten der Version 3.1 des CTFL parat zu haben. Wir beziehen uns im Buch immer auf den CTFL 3.1 [GTB 2018].

## Schlüsselbegriffe und Keywords

Zu Beginn jedes Kapitels listen wir die Schlüsselbegriffe und die dazugehörigen Begriffe aus dem Lehrplan auf. Zusätzliche Begriffe, die uns wichtig erscheinen, ergänzen diese Auflistung.

Manche Begriffe verwenden wir im Projektalltag in der Regel in Englisch. Ein Beispiel ist *Bias*. Der Begriff wird auch im Deutschen oft verwendet, der deutsche Begriff *Verzerrung* hingegen eher weniger. Wir haben versucht, diesem Umstand im Buch gerecht zu werden, und verweisen dann bei solchen Begriffen auf den Projektalltag. In einer möglichen Zertifikatsprüfung solltest du auf jeden Fall den deutschen Begriff kennen.

## Notwendige Vorbereitungen für die praktischen Übungen

Dieses Buch enthält zu vielen Kapiteln auch praktische Übungen. In den meisten veranschaulichen wir die Lerninhalte an Codebeispielen in der Programmiersprache *Python*. Die Aufgaben und Lösungen befinden sich auf GitHub unter *https://github.com/KI-Testen/Uebungen*, unserem KI-Testen-Repository. Wir verwenden hauptsächlich *Jupyter Notebooks*[1], da diese es uns ermöglichen, den Code für dich anschaulich zu dokumentieren, und du ihn interaktiv ausführen kannst. *Jupyter Notebooks* finden nicht nur an (Hoch-)Schulen, sondern auch in Lehrmaterialien häufig Verwendung.

Um unsere Jupyter Notebooks nutzen zu können, benötigst du Zugriff auf eine JupyterLab-Umgebung. Wir empfehlen die Installation auf deinem eigenen PC oder Laptop. Dann kannst du darauf Python und die notwendigen Bibliotheken selbst installieren. Eine Anleitung dafür findest du in unserem GitHub-Repository.

Wenn du auf deinem PC nichts installieren kannst oder willst, gibt es verschiedene Möglichkeiten, auf bereits installierte JupyterLab-Umgebungen im Web zuzugreifen, wie z. B.:

- GitHub Account: über *https://github.com/codespaces* einloggen und mit dem Jupyter Notebook-Template einen Codespace anlegen.
- Mit Google-Konto: auf *https://colab.research.google.com/* mit deinem Google Account einloggen.
- Auf Jupyter.org: ohne Login für Versuche ein JupyterLab über *https://jupyter.org/try-jupyter* starten.

Du musst in jeder dieser Varianten die Dateien aus unserem Git-Repository manuell dort hineinkopieren. Achte dabei darauf, die Verzeichnisstruktur beizubehalten. Wir können jedoch nicht garantieren, dass diese Umgebungen immer verfügbar sind und mit unseren Übungen funktionieren.

## Weitere Hinweise

### Anrede

Wir arbeiten meist in jungen und agilen Teams, in denen wir Produkte mitentwickeln und testen. In diesen steht das Team im Mittelpunkt und der Kommunikationsstil ist so angepasst, dass sich in der Regel alle Teammitglieder duzen. Die-

1. Einen Einstieg in Jupyter Notebooks und wie man damit arbeitet, findest du auf dieser Webseite: *https://jupyter-tutorial.readthedocs.io/de/latest/index.html*.

sen Kommunikationsstil haben wir auch für dieses Buch übernommen und duzen daher die Leserin und den Leser. Selbstverständlich kann uns auch jeder duzen. Wir hoffen, dass du dich durch diesen Kommunikationsstil nicht unangemessen angesprochen oder sogar angegriffen fühlst. Sollte dies doch der Fall sein, möchten wir uns dafür entschuldigen.

#### Geschlechtergerechte Sprache

Wie bereits in den vorangehenden Abschnitten zu sehen war, versuchen wir im gesamten Buch, eine geschlechtergerechte Sprache zu verwenden. Wir nutzen sowohl die weibliche als auch die männliche Form. Zum Beispiel schreiben wir die Nutzerin und der Nutzer. Auf die Verwendung von Gendersternchen oder Ähnlichem verzichten wir an dieser Stelle. Wir tun dies ausschließlich aus Gründen der besseren Lesbarkeit. Wir beabsichtigen damit keine Diskriminierung von Menschen unterschiedlichen Geschlechts. Sollten sich manche durch diese Entscheidung diskriminiert fühlen, möchten wir uns dafür entschuldigen und versichern, dass dies nicht unsere Absicht ist.

#### Praxisbeispiele

Die Praxisbeispiele stammen überwiegend aus unserer beruflichen Praxis. Um den Schutz der Kunden und deren Geschäftsgeheimnisse zu wahren, mussten wir abstrahieren oder kleinere Anpassungen vornehmen.

Einige Praxisbeispiele haben keinen direkten Bezug zu Projekten mit unserer Beteiligung. Sie sind, wie ein Spamfilter für E-Mails, aber Beispiele, die uns und vielleicht auch dir im täglichen Leben begegnen.

Die Praxisbeispiele sind folgendermaßen gekennzeichnet:

**Praxisbeispiel 0–1: Beispielthema (ggf. Erweiterung)**

So stellen wir Praxisbeispiele dar. Bei diesem Kasten handelt es sich um ein Anschauungsbeispiel, das zeigen soll, wie wir Situationen aus unserem (oder einem vergleichbaren) Projektalltag im weiteren Verlauf des Buches beschreiben.

#### Englische Abbildungen

Wir haben echte Screenshots eingefügt, um das Geschriebene bildlich zu verdeutlichen. Es gibt ein Bild nur mit englischem Inhalt. Da dieses Bild selbsterklärend ist und den Text ergänzen soll, wurde auf eine Übersetzung verzichtet.

#### Übungen

Wir empfehlen dringend, die im Buch enthaltenen Übungen durchzuführen und nicht nur zu lesen. Diese selbst durchzuführen, fördert wesentlich das Verständnis der Lerninhalte. Theoretische Kenntnisse wirken nur unterstützend für die praktische Anwendung. Die Übungen helfen dir, Ideen zu generieren und festzulegen, wie du mit der aktuellen Aufgabenstellung im Projektkontext umgehen kannst. Vielleicht findet sich für die Übungen eine Sparringspartnerin oder -partner aus der Testing Community oder unter den Arbeitskollegen.

Alle Aufgaben und Lösungen zu den Übungen findest du im online verfügbaren GitHub-Repository (*https://github.com/KI-Testen/Uebungen*).

#### Hinweis zum Glossar

Solltest du im Buch auf einen dir unbekannten Begriff stoßen, empfehlen wir dir, direkt in unserem Glossar am Ende des Buches nachzuschlagen. Wir haben darin alle Begriffe, die unserer Meinung nach zum besseren Verständnis beitragen, aufgelistet und mit dem domänenspezifischen Glossar des CT-AI-Lehrplans [GTB 2021] und dem allgemeinen Glossar des ISTQB® [URL: Glossar] abgeglichen. Bei den Begriffen und/oder Definitionen kann es andere Formulierungen geben, die Inhalte stimmen jedoch aus unserer Sicht überein. Sollten in diesem Buch weitere unbekannte Wörter oder Begriffe auftauchen, empfehlen wir dir als erste Anlaufstelle die beiden genannten Quellen.

#### Quellen und Links

Quellen und Links wurden von uns zuletzt im August 2023 überprüft. Sollten sich nach diesem Datum Änderungen ergeben haben, sind diese nicht im Buch berücksichtigt.

## Danksagungen

Wir möchten Danke sagen: Ein großes Dankeschön geht an Anna, Brigitte und Felix für ihre Unterstützung, ihre Geduld und ihre Nachsicht während der letzten fast zweieinhalb Jahre.

Danke auch an Florian Fieber und Binia Sonnen für ihre vielen Anmerkungen und Tipps, die das Buch qualitativ deutlich aufgewertet haben. Ebenso bedanken wir uns bei Andreas Spillner, Mario Winter und Jonas Mönnich für ihre wertvollen Anmerkungen sowie bei unseren zahlreichen Kolleginnen und Kollegen, mit denen wir viele Gespräche über das Thema KI und damit auch über das Buch geführt haben.

Zu guter Letzt geht unser Dank auch an den dpunkt.verlag und Christa Preisendanz für die gute Betreuung während des Schreibens.

# 1 Einführung in KI

*»Dinge wahrzunehmen ist der Keim der Intelligenz.«*
Laotse

*In diesem Kapitel erklären wir allgemein den Begriff der künstlichen Intelligenz (KI) und die damit in Verbindung stehenden Systeme. Wir vermitteln, wie der Begriff entstand und sich bis heute entwickelt hat, aber auch was heutige künstlich intelligente Systeme charakterisiert und sie von konventioneller Software unterscheidet. Ebenso werfen wir einen Blick darauf, wie und mit welchen Frameworks KI-basierte Systeme entwickelt werden und was bei deren Betrieb zu beachten ist: als Service in der Cloud, auf dedizierter Hardware oder im regulatorischen Umfeld.*

**KI-spezifische Schlüsselbegriffe aus dem Lehrplan:**
AI as a Service (AIaaS), KI-Entwicklungs-Framework (AI development framework), KI-Effekt (AI effect), KI-basiertes System (AI-based System), künstliche Intelligenz (KI, artificial intelligence), neuronales Netzwerk (neural network), Deep Learning (DL), tiefes neuronales Netzwerk (deep neural network), starke KI (general AI), Datenschutzgrundverordnung (DSGVO, General Data Protection Regulation, GDPR), maschinelles Lernen (ML, machine learning), schwache KI (narrow AI), vortrainiertes Modell (pre-trained model), künstliche Superintelligenz (super AI), technologische Singularität (technological singularity), Transfer-Lernen (transfer learning)

**Weitere Schlüsselbegriffe in diesem Kapitel:**
keine

## 1.1 Definition von KI und der KI-Effekt

Der Begriff der *künstlichen Intelligenz (KI)* ist in unserem Leben fast allgegenwärtig geworden. Wir haben ihn schon oft in den Nachrichten gehört, in Artikeln und Büchern dazu gelesen oder Filme darüber gesehen. So verwundert es nicht, wenn viele von sich sagen, den Begriff *KI* verstanden zu haben. Wir ordnen dann gerne KI als aktuelle *Hightech-Innovation* ein.

Neu ist die Idee einer von Menschen geschaffenen Intelligenz aber keinesfalls. Schon weit vor dem heute beobachtbaren gesellschaftlichen Hype führte die wissenschaftliche Forschung den Begriff der künstlichen Intelligenz ein. Er geht auf die 1950er-Jahre zurück, als Forscherinnen und Forscher der damals bereits wachsenden technischen Möglichkeiten ein besonderes Ziel vor Augen hatten: Rechenmaschinen zu bauen, die den Menschen und seine Fähigkeiten imitieren können. Als wichtigstes Ereignis wird heute der Workshop am Dartmouth College in New Hampshire im Jahr 1956 angesehen, der erstmals eine Studie zu künstlicher Intelligenz zum Thema hatte.[1]

Seitdem beschäftigt sich die Forschung in Theorie und Praxis mit diesem Ziel. Mit den Fortschritten in der mathematischen Modellierung, den technischen Möglichkeiten und neuen potenziellen Anwendungsgebieten hat sich auch die Definition des Begriffs der künstlichen Intelligenz ständig weiterentwickelt. Im derzeitigen Sprachgebrauch versteht man Folgendes darunter:

**Künstliche Intelligenz (KI):** Die Fähigkeit eines technischen Systems, Wissen und Fertigkeiten zu erwerben, zu verarbeiten, zu entwickeln und anzuwenden.[a]

a. Aus der englischen Definition des technischen Reports ISO/IEC TR 29119-11 [ISO/IEC TR 29119-11].

Der Wandel, dem die Definition des KI-Begriffs bis heute unterworfen ist, erklärt sich vor allem aus einer veränderten Wahrnehmung technischer Systeme und deren Leistungen. Insbesondere ist die Grenze zwischen konventioneller Software und Systemen mit künstlicher Intelligenz nicht festgeschrieben, sondern ebenfalls einem Wandel unterworfen.

Noch in den 1970er-Jahren konnten mäßig geübte Schachspielerinnen und -spieler die damaligen Schachcomputer besiegen. Ein Programm, das in der Lage gewesen wäre, einen Schachweltmeister zu schlagen, schien dagegen unmöglich. Das Schachspiel ist ein Paradebeispiel für eine ganze Reihe von Problemen, die trotz weniger Regeln und begrenztem Spielfeld eine hohe strategische Komplexität aufweisen. Als 1997 dann *Deep Blue* den damals amtierenden Schachweltmeister Garri Kasparow besiegte, wurde dies als Siegeszug der künstlichen Intelligenz gefeiert. Aus heutiger Sicht wird die damalige Software jedoch von vielen nicht mehr als KI bezeichnet. Deep Blue nutzte damals eine *Brute-Force*-Methode, also eine auf purer Rechenleistung beruhende Methode, um den nächsten Zug zu ermitteln. Ein Erwerben oder Weiterentwickeln der eigenen Fertigkeiten, wie nach der aktuellen Definition einer KI, war hier gar nicht vorgesehen.

Mit dem Begriff der künstlichen Intelligenz verbinden wir durchaus eine Art von Bewunderung für eine unerwartete Leistung. Insbesondere, wenn man diese

1. *https://en.wikipedia.org/wiki/Dartmouth_workshop*, abgerufen am 08.08.2023.

nicht ohne Weiteres technisch, einfach und nachvollziehbar erklären kann. Eine kleine Anekdote dazu: Selbst die Entwicklerinnen und Entwickler von Deep Blue waren damals über bestimmte Züge ihres Programms überrascht. Später fanden sie heraus, dass einer diese Züge auf einen Programmierfehler zurückzuführen war, der die Maschine zunächst in eine Endlosschleife versetzte. Ein Notfallmechanismus hat dann die Zugsuche abgebrochen und per Zufall einen beliebigen ausgewählt. Just dieser völlig unorthodoxe Zug hatte Kasparow damals aus dem Konzept gebracht, weil er dahinter eine besondere Genialität vermutete. Das Blatt wendete sich und Kasparow verlor [Silver 2012].[2]

Ebenso zählte die Fachwelt *Expertensysteme* der 1970er- und 1980er-Jahre lange Zeit zu den Systemen mit künstlicher Intelligenz. Diese arbeiten mit klar beschriebenen Regeln, z.B. Wenn-dann-Bedingungen, sowie mit in Datenbanken gespeichertem Expertenwissen. Meist können sie komplizierte Fragestellungen aus einem bestimmten Wissensgebiet gut beantworten. Schon die ersten Expertensysteme zeigten, trotz einfach strukturierter Regelwerke, Fähigkeiten, die man damals nur menschlichen Fachexpertinnen und -experten zutraute. Heute werden die damaligen Expertensysteme nicht mehr zur KI gezählt, moderne Expertensysteme hingegen schon.

Die Verschiebung dieser Grenze, also die Wahrnehmung, ob man ein System als künstliche Intelligenz ansieht oder nicht, bezeichnet man als *KI-Effekt* [URL: Wikipedia]. Im Laufe der Zeit hat sich diese Einschätzung in der Gesellschaft verändert und sie verändert sich vermutlich auch weiterhin. Diesem Umstand trägt der alternative Begriff *Lernende Systeme* Rechnung, der sich in den vergangenen Jahren gerade in der Forschung etabliert hat.

In diesem Buch benutzen wir den Begriff *KI* sehr oft. Damit meinen wir die sehr weit gefasste Menge an Systemen im Sinne der oben genannten Definition. In den seltensten Fällen ist jedoch eine reine *KI-Software* allein im Einsatz. Meistens ist sie in konventionelle Software[3] eingebettet. Um dies zu betonen, werden wir diese dann auch als *KI-basierte* Systeme bezeichnen. Im weiteren Verlauf des Buches müssen wir andererseits aber auch präziser werden und sprechen dann z.B. von *logikbasierten Systemen* (siehe Abschnitt 1.4) oder *Systemen des maschinellen Lernens* (ML-Systemen, ab Kap. 3).

## 1.2 Schwache KI, starke KI und die künstliche Superintelligenz

Viele kennen die künstlichen Superintelligenzen in den Science-Fiction-Filmen wie *2001 – Odyssee im Weltraum* oder *Terminator*. Sie greifen nicht nur auf übermenschliches Wissen zu, sondern sind sogar fähig, die Menschheit auszulöschen. Wenn du dir aber anschaust, wozu KI heute in der Lage ist, dann wird schnell

2. *https://en.wikipedia.org/wiki/Deep_Blue_(chess_computer)#Deep_Blue_versus_Kasparov*, abgerufen am 08.08.2023.
3. In Projekten verwenden wir synonym auch den Begriff *klassische Software*.

klar, dass diese Superintelligenzen derzeit nicht realisierbar sind. KI, so wie sie heute eingesetzt und entwickelt wird, ist lediglich in der Lage, spezifische Aufgaben in einem limitierten Kontext auszuführen. Wir sprechen hier von *begrenzter* oder *schwacher* KI, da ihr Einsatzgebiet auf einen sehr begrenzten Problemraum beschränkt ist.

**Schwache KI:** KI, die auf eine einzelne klar definierte (engl. well-defined) Aufgabe konzentriert ist, um ein spezifisches Problem zu lösen.[a]

a. Übersetzt aus [ISO/IEC TR 29119-11].

Solche begrenzten KIs zeigen uns immer wieder beeindruckende Spielkünste in strategischen Spielen, wie z.B. *AlphaZero,* veröffentlicht von der Google-Firma DeepMind im Dezember 2017 [Silver et al. 2017]. AlphaZero kann innerhalb von wenigen Stunden eigenständig Spielstrategien verschiedener Brettspiele nur anhand der Spielregeln erlernen. Damit zeigt AlphaZero Fähigkeiten, die bei Weitem die eines Menschen übersteigen. Dennoch betrachten wir dieses System als begrenzte KI, da sie mit ihren Fähigkeiten auf das sehr spezifisches Aufgabenfeld der strategischen Brettspiele beschränkt ist.

In unserem Alltag finden wir alle mittlerweile viele Situationen, in denen schwache KIs Aufgaben übernehmen: Spamfilter, die unerwünschte E-Mails aussortieren, Sprachassistenten, die unsere Telefonate entgegennehmen, Bordcomputer in unseren Autos, die uns mitteilen, welcher Kundenservice ansteht. All diese KIs zeigen ihr Können nur in ihrem definierten Aufgabenbereich. Für neue Aufgaben außerhalb ihres vorgesehenen Einsatzgebiets sind sie nicht entwickelt.

Stell dir vor, du trainierst eine KI mit sehr vielen Hundebildern, um auf Bildern von Hunden deren Rasse erkennen zu können.[4] Wenn du der KI Hundebilder präsentierst, funktioniert diese erstaunlich gut. Was passiert aber, wenn auf dem Bild kein Hund zu sehen ist? Dafür ist die KI nicht trainiert. Sie versucht dennoch, das Bild einer Hunderasse zuzuordnen, obwohl es sich bei dem Bild gar nicht um einen Hund handelt. Die offensichtlich begrenzte KI bleibt in ihrem eng gefassten Aufgabenbereich.

Die offene Forschungsplattform *OpenAI* hat mit *GPT-3* im Juni 2020 eine KI zur Verarbeitung von Texten auf einem noch nicht dagewesenen Level veröffentlicht [Brown et al. 2020]. Seit Anfang 2023 ist die neuere Version GPT-3.5 als Chatbot *ChatGPT* für alle kostenlos verfügbar. GPT-3 ist ein durch *Deep Learning* trainiertes neuronales Netz[5], das zur Textverarbeitung im weitesten Sinne eingesetzt werden kann. Das Einsatzgebiet für GPT-3 reicht von der Erstellung von journa-

4. Auf der offenen Plattform Kaggle findest du dafür geeignete Datensätze, z.B.: *https://www.kaggle.com/jessicali9530/stanford-dogs-dataset*.
5. Wir verwenden abweichend vom Syllabus im gesamten Buch den Begriff *neuronales Netz* anstelle von *neuronalem Netzwerk*, siehe dazu unseren Hinweis zu Beginn von Kapitel 6.

listischen Artikeln über Gedichte bis hin zu Programmcode. Sind damit die Grenzen der begrenzten KI hin zur generellen KI überschritten?

**Starke KI:** KI, die über das gesamte Spektrum der kognitiven Fähigkeiten ein dem Menschen vergleichbares intelligentes Verhalten zeigt. [a]

a. Übersetzt aus [ISO/IEC TR 29119-11].

In der Definition der *starken* KI, angelehnt an die ISO/IEC TR 29119-11:2020 (der Lehrplan spricht hier von der *allgemeinen* KI), wird die Summe der menschlichen Fähigkeiten ins Zentrum gestellt. Im Gegensatz zur begrenzten KI sprechen wir von einer generellen KI, wenn die Fähigkeiten nicht auf eine Aufgabenstellung beschränkt ist. Eine starke KI hat weitreichende Fähigkeiten, vergleichbar mit denen eines Menschen, wie beispielsweise Begründungen zu formulieren, die Umwelt zu verstehen, Schlussfolgerungen zu ziehen und danach zu handeln.

Auch wenn die Fähigkeiten von GPT-3 so mächtig erscheinen, ist GPT-3 auf rein textuelle Aufgaben beschränkt und nicht in der Lage, wie ein Mensch Sachverhalte zu verstehen, sondern ahmt Formulierungen aus den Trainingsdaten, wie im Training gelernt, lediglich nach. Daher sprechen wir bei GPT-3 nicht von einer generellen KI. Gleiches gilt für die bessere Version GPT-4. Im Kontext der gegebenen Definitionen und nach Stand von August 2023 wurde noch keine starke KI realisiert, kurz: Es gibt keine.

Die oberste Stufe der künstlichen Intelligenz ist die der *künstlichen Superintelligenz*. Darin werden KIs zusammengefasst, die das Verhalten von uns Menschen nicht nur nachahmen können, sondern unsere Fähigkeiten durch die Verarbeitung von gewaltigen Datenmengen sogar übertreffen. Dazu nutzen sie einen praktisch unlimitierten Datenspeicher und Zugang zu allem menschlichen Wissen, wie es z.B. im Internet zu finden ist. Der Punkt, an dem die KI von der generellen KI zur künstlichen Superintelligenz übergeht, wird auch als *technologische Singularität* beschrieben.

**Künstliche Superintelligenz:** Eine KI, deren Fähigkeiten die Fähigkeiten des Menschen überschreiten bzw. weit übertreffen.

Auch wenn diese künstlichen Superintelligenzen bis jetzt nur in Filmen existieren, hielt Stephen Hawking sie in der Zukunft für realisierbar und warnt vor deren Fähigkeiten.

> *»Ich fürchte, dass die künstliche Intelligenz den Menschen insgesamt ersetzen könnte. Wenn Menschen Computerviren entwerfen, wird jemand eine künstliche Intelligenz entwerfen, die sich selbst verbessert und vermehrt. Das wird eine neue Lebensform sein, die den Menschen überragt.«*
>
> Stephen Hawking, 2017

## 1.3 KI-basierte Systeme und klassische Systeme

*»KI ist auch nur eine Software.«* So hört man es des Öfteren. Das ist sicher richtig, aber KI-Systeme werden nicht wie *klassische Software* (auch *konventionelle Software* genannt) entwickelt. Denke an klassische Softwaresysteme, hier haben Menschen die darin enthaltenen logischen Zusammenhänge und Abläufe hergeleitet. Diese werden in Form von Anweisungen wie *Wenn-dann-sonst-Konstrukten* (engl. if-then-else) und Schleifen im Programmcode festgehalten. Die dafür verwendeten Programmiersprachen werden auch als *imperative Programmiersprachen* bezeichnet. So entscheiden die Entwicklerinnen und Entwickler, bei welchem Input sich die Software wie verhalten soll.

Für Menschen ist es im Normalfall relativ einfach, mit entsprechendem Vorwissen die logischen Zusammenhänge zu verstehen. Sie können nachvollziehen, wie die Softwareeingaben in Ausgaben[6] umgewandelt werden und warum sich das System so verhält, wie es sich verhält. Selbst, wenn diese klassischen Softwaresysteme sehr komplex sind, sind doch Menschen mit Vorwissen in der Lage, diese Software zu verstehen und den Grund für ein bestimmtes Verhalten zu ermitteln.

KI-basierte Systeme, die durch *maschinelles Lernen* (engl. machine learning, ML) trainiert werden, erlernen während der Trainingsphase das gewünschte Verhalten. Um diese Systeme zu trainieren, gibt es eine Vielzahl von Techniken, auf die wir in Abschnitt 1.4 genauer eingehen. Unabhängig von der gewählten Technik, wird in der Trainingsphase eines KI-basierten Systems anhand von erlernten Datenmustern bestimmt, wie es in Zukunft auf neue Daten reagieren soll. Die zugrunde liegenden logischen Zusammenhänge werden nun nicht mehr von Entwicklerinnen oder Entwicklern vorgegeben, sondern automatisiert aus den Datenmustern durch das ML abgeleitet (für eine detaillierte Erklärung des maschinellen Lernens siehe Kap. 3). Du kannst dir leicht vorstellen, dass die Qualität der Daten einen wesentlichen Einfluss darauf hat, wie gut die KI in der Trainingsphase lernt. Wenn die Lernphase mit Daten, die keine repräsentativen Datenmuster und Merkmale beinhalten, stattgefunden hat, lernt die KI während der Trainingsphase unzureichende oder sogar falsche Regeln (siehe die Abschnitte 2.4 und 3.5).

Während bei der klassischen Softwareentwicklung die Entwicklerinnen und Entwickler direkten Einfluss auf die Logik der Software haben, haben sie bei KI-basierten Systemen nur indirekt Einfluss auf deren Verhalten, nämlich lediglich über die Wahl des Inputs während des Trainings und die Auswahl der KI-Technik und des dazugehörigen ML-Algorithmus (siehe Abb. 1–1). Ein *ML-Algorithmus* ist eine mathematische Vorgehensweise, die zur Erstellung eines ML-Modells aus einem Trainingsdatensatz verwendet wird.

---

6. Im Folgendem schreiben wir oft Input und Output. Diese Begriffe werden meist im KI-Kontext verwendet.

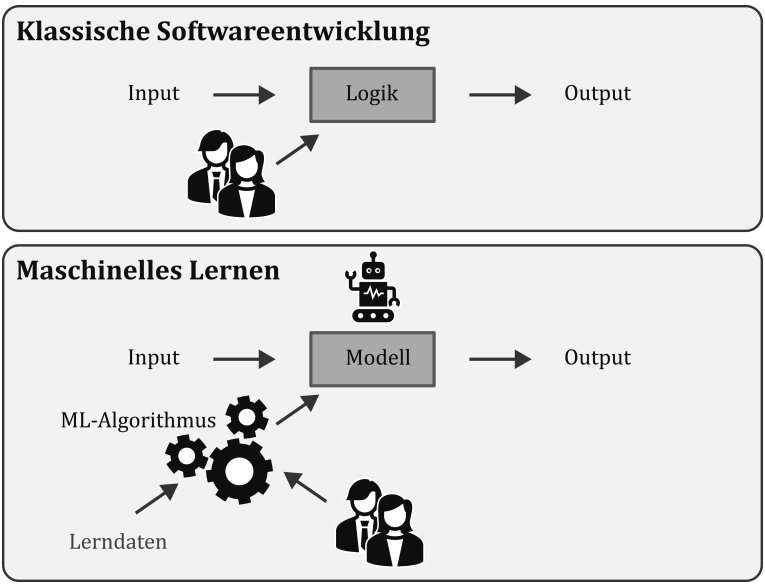


***Abb. 1–1*** *Klassische und KI-basierte Software im Vergleich*

KI-basierte Systeme haben im Vergleich zu klassischen Systemen meist den Nachteil, dass die logischen Regeln für Menschen weniger leicht bis überhaupt nicht nachvollziehbar sind – je nach Komplexität der KI-Technik (siehe Abschnitt 2.7). Daher spricht man bei KI-Software oft auch von einer *Blackbox*.

Um den Unterschied zwischen der Entwicklung von KI-basierten und klassischen Systemen noch deutlicher zu machen, schauen wir uns ein Praxisbeispiel aus dem Bereich des maschinellen Lernens an.

**Praxisbeispiel 1–1: Testframework für eine Heizungssteuerung**

Im Rahmen dieses Projekts ging es um funktionale Tests einer Heizungssteuerung (Smarthome-Steuerung). Auf einem Touchdisplay konnten die Raumtemperatur sowie viele andere Einstellungen der Heizung vorgenommen werden. Im Projekt musste für verschiedene Testszenarien bewertet werden, ob das Touchdisplay die richtigen Informationen anzeigt. Dafür wurde im Test der Smarthome-Steuerung jede Minute das Touchdisplay fotografiert und das Foto in einer Datenbank abgelegt. Mithilfe des KI-Systems sollte der Testaufwand für die manuelle Auswertung der Fotos reduziert werden. Es wurde ein Testframework entwickelt, das den Inhalt der Bilder automatisiert erkennen und auswerten sollte. Das Framework setzte sich aus klassischen und KI-Softwarekomponenten zusammen.

Ein Teil des Touchdisplays stellte eine Temperaturanzeige mit vier Sieben-Segment-Ziffern dar (siehe Abb. 1–2). Wir wählten für die sehr einfach aufgebaute Anzeige einen klassischen Softwareansatz für das Auslesen der Temperaturanzeige: Wir überlegten uns,

→

welche Merkmale auf der Anzeige ausschlaggebend sind und welche Logik im Programmcode angewandt werden musste, um festzulegen, um welche Ziffer es sich an den einzelnen Positionen handelt. Dies gestaltete sich recht aufwendig, da die Software auch bei leicht variiertem Winkel des Fotos oder sich änderndem Lichteinfall fähig sein sollte, die Anzeige abzulesen. Nach langem Tüfteln mit den Belichtungszeiten und den Kontrasteinstellungen hatten wir eine zuverlässige Testautomatisierung entworfen. Diese war in der Lage, die auf der Temperaturanzeige abgebildete Zahl mit einem Sollwert zu vergleichen.

Die Anzeige enthielt neben der Temperaturanzeige auch deutlich kompliziertere Elemente, wie Warnsymbole, ein Zeichen für den Kalibrierungsmodus und eine Batteriezustandsanzeige. Der Aufwand mit der oben beschriebenen Herangehensweise wurde dadurch ungleich größer. Da die Logik im Programmcode zu komplex zu werden drohte, entschieden wir uns für einen ML-Ansatz. Wir wählten ein neuronales Netz, das sich für diese Klassifikationsaufgabe eignete. Um die KI zu trainieren, verwendeten wir den selbst erstellten Datensatz von Bildern dieser Heizungssteuerung. Anhand dieser Trainingsdaten lernte die KI, die Merkmale und Muster zu erkennen und sie den verschiedenen Symbolen zuzuordnen. Allerdings stellten wir beim Testen der KI fest, dass diese nicht die benötigte Genauigkeit hatte. Warum die KI so schlechte Ergebnisse lieferte, konnten wir anhand des Programmcodes der KI nicht mehr erkennen. Die Logik im neuronalen Netz war für uns Menschen nicht mehr nachvollziehbar.

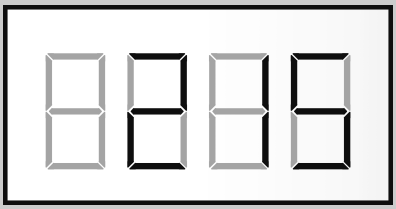

***Abb. 1–2*** *Sieben-Segment-Anzeige auf dem Display einer Heizungssteuerung*

**Praxistipp**

Auch wenn KIs beeindruckende Erfolge erzielt haben, haben sie den großen Nachteil, dass ihre Logik nur schwer bis gar nicht mehr nachvollziehbar ist. Es gibt Ansätze, die versuchen, KIs erklärbar zu machen (siehe Abschnitt 2.7). Bei sensiblen Aufgabenstellungen kann es dennoch sinnvoll sein, von der Verwendung einer KI ganz abzusehen und auf eine klassische Software zurückzugreifen. Diese ist möglicherweise aufwendiger in der Entwicklung der inneren Logik, der Entscheidungsprozess der Software ist dafür aber im Detail nachvollziehbar. Testerinnen und Tester können in einer solchen Software auch sehr einfach Reviews vornehmen und Whitebox-Tests durchführen. Reviews und Whitebox-Tests in einer KI sind im Gegensatz dazu sehr schwierig bis unmöglich.

Bei der Abgrenzung von konventionellen und KI-basierten Systemen musst du dir im Klaren darüber sein, dass sich in der Praxis die Wahrnehmung über KI-basierte Systeme in stetem Wandel befindet. Systeme, die man heute als KI-basierte Systeme bezeichnet, werden in Zukunft vielleicht den konventionellen Systemen zugeordnet (siehe Abschnitt 1.1).

## 1.4 KI-Techniken

Im vorangegangenen Abschnitt haben wir den wesentlichen Unterschied zwischen KI-basierter und konventioneller Software beschrieben. Insbesondere haben wir dort das maschinelle Lernen als eine der am weitverbreitetsten Entwicklungsmethoden erwähnt. Doch das ist nicht die einzige Methode. In Abbildung 1–3 zeigen wir eine Auswahl an *KI-Techniken*[7] (der ISTQB®-Lehrplan spricht hier gar von *Technologien*).

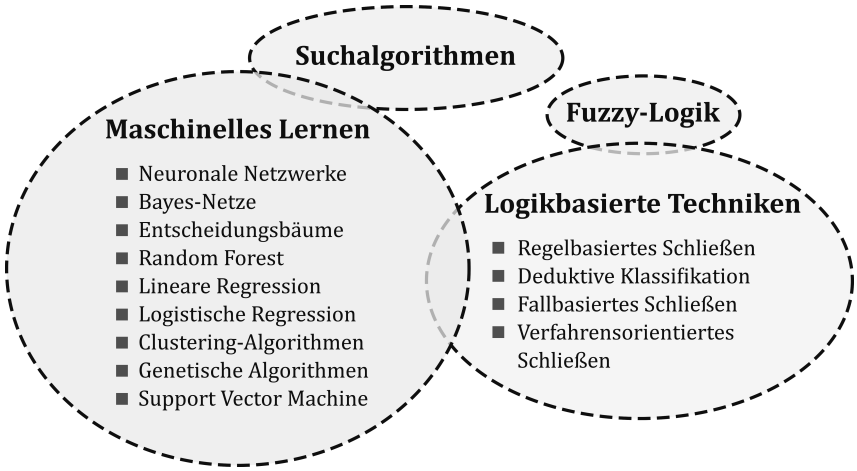


***Abb. 1–3*** *Eine Auswahl an KI-Techniken, mit denen KI-Komponenten entwickelt werden können.*

Wenn man eine KI-Komponente entwickeln will, muss man meist frühzeitig eine KI-Technik und einen der dazugehörigen ML-Algorithmen auswählen. Erst danach kann man mit dem Training beginnen. In Abschnitt 3.4 gehen wir näher darauf ein, welche Faktoren für die Auswahl eines Algorithmus zum Tragen kommen.

Jede KI-Technik hat ihre eigenen Vorzüge, aber auch Nachteile. Daher ist es nicht ungewöhnlich, dass man nach einem wenig erfolgreichen Training die Wahl des Algorithmus überdenken und mit einem vermutlich besser geeigneten Algorithmus das Training erneut beginnen muss.

Es kommt nicht selten vor, dass man in einem KI-basierten System auch mehr als eine einzige KI-Technik verwendet. Stell dir beispielsweise ein System vor, das eine Reisebuchung über einen Sprachdialog ermöglichen soll. Man wird es dort vermutlich mit neuronalen Netzen zur Spracherkennung zu tun haben. Die Auswahl des Reiseangebots könnte hingegen durch einen intelligenten Suchalgorithmus unterstützt werden.

7. Technisch Interessierte finden einen umfassenden Einblick in [Russell & Norvig 2020] oder eine Übersicht zu vielen weiteren KI-Techniken in [Wahlster et al. 2022, ab S. 42].

### 1.4.1 Exkurs: KI-Techniken im Detail

Im Folgenden gehen wir auf die in Abbildung 1–3 gezeigten KI-Techniken näher ein und beschreiben auch in groben Zügen die ein oder andere Technik, damit du diese besser einordnen kannst.

#### Suchalgorithmen

Im ersten Moment würdest du wahrscheinlich keine KI hinter einer automatisierten Suche vermuten. Gerade intelligente Suchalgorithmen haben eine enorme Bedeutung erlangt. Dabei darfst du nicht nur an die klassische (z. B. lineare) Suche durch eine Liste von Elementen denken. Vielmehr kommen oft spezialisierte Suchalgorithmen zum Einsatz, wie etwa die folgenden:

- **Heuristische Suchalgorithmen** finden beispielsweise nicht alle mathematisch korrekten Treffer, ermitteln dafür aber diejenigen, die im aktuellen Kontext am besten passen – und das in erstaunlich kurzer Zeit. Denke hier nur an die Suche bei *Google*.
- **Optimierende Suchen** finden innerhalb großer Parameterräume sehr effizient Kombinationen dieser Parameter, die eine möglichst niedrige Kostenfunktion ergeben – etwa bei Routenplanern.
- **Stark spezialisierte Suchen** nach Zeichenketten, etwa zum Auffinden ähnlicher Wörter oder Begriffe, sind ebenfalls Beispiele dieser KI-Techniken.

Suchalgorithmen zählt man verallgemeinert zu den Problemlösungsstrategien und Optimierungsverfahren.

#### Logikbasierte Techniken

Auf Logik basierende Techniken sind alle dadurch gekennzeichnet, dass sie Erfahrungswissen explizit maschinenlesbar erfassen und durch Regeln miteinander in Beziehung setzen:

- Das **regelbasierte Schließen** (Regelmaschinen) greift in der klassischen Form auf Methoden der Aussagenlogik oder Prädikatenlogik zurück. Diese sind in der Lage, die Korrektheit einer neuen Aussage anhand elementarer bekannter Aussagen und Regeln zu deren Verknüpfung entweder zu beweisen oder zu widerlegen.
- **Deduktive Klassifikatoren** leiten beim sogenannten deduktiven Schließen neue Aussagen aus den bisher bekannten ab. Sie können dadurch beispielsweise Begriffe oder Gegebenheiten in Kategorien, Klassen oder Unterklassen einordnen. Aktuell gewinnen derartige Systeme wieder an Bedeutung, denn sie erlau-

ben es, die seit einigen Jahren entstehenden und über das Internet geteilten Wissensdatenbanken von Begriffen und Bedeutungen (Ontologien) als Datengrundlage zu nutzen. Sie können damit zu mehr wissenschaftlicher Erkenntnis beitragen und sogar im Kampf gegen COVID-19 helfen [Smith 2020].

- Sowohl **fallbasiertes Schließen** (fallbezogene Argumentation) als auch **verfahrensorientiertes Schließen** (prozedurale Argumentation) erlauben es, die Lösungen für neue Problemsituationen durch den Vergleich mit bereits bekannten Problemen und den dafür bekannten Lösungsverfahren abzuleiten.

#### Fuzzy-Logik

Neben der einfachen binären Logik (wahr oder falsch) existieren auch sogenannte **mehrwertige Logiken,** die in Verfahren des logischen Schließens angewandt werden. Diese verwenden für eine Aussage neben den Alternativen *wahr* und *falsch* auch Wahrheitswerte dazwischen. Beispielsweise können damit in einer Nachbildung der Logik digitaler Schaltkreise auch Zustände zwischen den beiden sonst verwendeten Spannungspegeln *High* und *Low* modelliert werden. Gerade in der Schaltungstechnik erweist sich das als sehr hilfreich.

Die sogenannte *Fuzzy-Logik* erlaubt darüber hinaus sogar beliebig viele Wahrheitswerte. Dennoch kann man auf diesen Werten weiterhin logische Operationen wie z.B. ∧ (Und), ∨ (Oder), ¬ (Nicht) anwenden.

#### Maschinelles Lernen

Die derzeit wohl prominenteste Kategorie von KI-Techniken ist *maschinelles Lernen*. Ab Kapitel 3 widmen wir uns ausgiebig diesem Themenbereich. Im Folgenden veranschaulichen wir gleich an mehreren Stellen das Geschriebene anhand des Praxisbeispiels einer Wetterstation.

**Praxisbeispiel 1–2: Wetterstation**

Unsere Aufgabe ist es, für eine kleine ortsfeste Wetterstation eine Regenvorhersagefunktion zu programmieren. Unsere Wetterstation liefert täglich Messdaten zu Luftdruck, Temperatur und relativer Luftfeuchtigkeit an ihrem Standort. Aus diesen Messdaten soll unsere Regenvorhersage abgeleitet werden.

Zum maschinellen Lernen zählt eine Vielzahl von Techniken. In Projekten begegnen wir öfters den folgenden:

- **Neuronale Netze** haben aktuell eine große Verbreitung. Daher werden wir in Abschnitt 6.1 noch genauer auf diese eingehen.

- Die **Bayes-Netze** und **Bayes-Klassifikatoren** nutzen mathematische Herangehensweisen, die auf einer statistischen Betrachtung der vorhandenen Trainingsdaten einschließlich der jeweils erwarteten Ergebnisse (als Annotationen[8]) beruhen. Auf Basis der beobachteten Wahrscheinlichkeiten in den Trainingsdaten kann für eine neue Situation auf die (aus Trainingssicht) wahrscheinlichste Ergebnisklasse geschlossen werden. Das Praxisbeispiel 1–3 skizziert die Grundidee dieser KI-Technik für eine Wetterstation.

**Praxisbeispiel 1–3: Wetterstation (Erweiterung)**

Für die Wetterstation aus Praxisbeispiel 1–2 wollen wir vorhersagen, ob es in den kommenden drei Stunden am Ort regnen wird oder nicht. Dies wollen wir statistisch betrachten: Größen wie Temperatur, Luftdruck und Luftfeuchtigkeit der letzten Stunden an diesem Ort stehen vermutlich damit im Zusammenhang; der Statistiker sagt, sie korrelieren mit dem Ereignis Regen. Je stärker diese Korrelation in der Statistik ausgeprägt ist, desto sicherer ist die Prognose auch für neue Messwerte der Wetterstation. Dabei fließen alle Parameter, hier die Wetterverhältnisse, zugleich in die Bewertung ein.

- **Entscheidungsbäume** nutzt man meist ebenfalls für eine Klassifikation wie etwa im obigen Beispiel zur Wetterprognose. Dabei zieht man in gleicher Weise annotierte Trainingsdaten heran. Der erzeugte Klassifikator berücksichtigt jedoch die einzelnen Einflussparameter Schritt für Schritt in Form von Wenn-dann-Bedingungen. So entsteht eine baumartig verzweigte Kaskade von Entscheidungen, an deren Ende die jeweils als am wahrscheinlichsten ermittelte Klasse benannt wird (siehe Abb. 1–4). Je nachdem, welche Strategie bei der Verzweigung oder Baumtiefe angewendet werden soll, kann man aus einer Reihe von konkreten Algorithmen wählen, z. B. *CART*, *ID3* oder *C4.5* [Knuth 2021].

  Das **Random-Forest**-Verfahren geht noch einen Schritt weiter und nutzt gleich mehrere unabhängig voneinander trainierte Bäume zur Entscheidungsfindung.
- Die **Support Vector Machine** (SVM) ist eine weitere Klassifikationsmethode, die ebenfalls auf bereits annotierten Trainingsdaten beruht. Das Grundprinzip besteht darin, dass der SVM-Klassifikator für z. B. in zwei Klassen eingeteilte Trainingsdaten eine möglichst gute Trennlinie im Parameterraum der zugehörigen Eigenschaften findet. Um eine solche Grenze in Form einer *Hyperebene* (engl. hyperplane) finden zu können, muss man ggf. durch Transformationen künstlich neue Parameter (Dimensionen) erzeugen.

8. Annotationen sind Zusatzinformationen zu (Trainings-)Daten, die mit diesen gespeichert werden, z. B. das erwartete Klassifikationsergebnis.

In einem fiktiven Beispiel[9] wollen wir anhand von Länge und Gewicht eines Grizzly-Oberschenkelknochens einordnen, ob dieser von einem urzeitlichen, groß gewachsenen Exemplar oder von einer heutigen Art stammt. Betrachten wir jede Eigenschaft für sich getrennt, überlappen sich die Wertebereiche und wir können allein anhand der Länge oder des Gewichts nicht gut genug unterscheiden. Betrachten wir beide gleichzeitig wird eine Trennung durch eine Gerade offensichtlich, wie in Abbildung 1–5 zu sehen. Muss die SVM mehr als zwei Parameter (Dimensionen) einbeziehen, wird diese Grenze zur Hyperebene.

- Die **(lineare) Regression** zählt im Kontext von KI ebenso zu den Verfahren des maschinellen Lernens. Diese ermittelt im Gegensatz zur Klassifikation als Ergebnis eine Größen- oder Mengenabschätzung. Am Beispiel der Wetterprognose (siehe Praxisbeispiel 1–4) würde man etwa die erwartete Regenmenge in Liter pro Quadratmeter für die nächsten drei Stunden als Ergebnis erhalten. Die Regression beschreibt dann den Zusammenhang der Ergebnisse (abhängige Variablen) von den Trainingsdaten (unabhängigen Variablen). Die Lineare Regression optimiert hierbei lineare Gewichtungskoeffizienten von unabhängigen zu abhängigen Variablen.
- Die **logistische Regression** ermittelt ebenso Größenabschätzungen wie die lineare Regression. Allerdings ist sie spezialisiert auf die Situation, in der Eingangs- oder Ausgangsgrößen nur wenige einzelne (diskrete) Werte statt eines kontinuierlichen Wertebereichs annehmen können.
- **Clustering-Algorithmen** decken zu den bisher genannten Verfahren einen neuen Aspekt ab: Manchmal steht man einer Klassifikationsaufgabe gegenüber, hat aber keine Trainingsdaten verfügbar, die die zu unterscheidenden Klassen vorab aufzeigen. In solchen Fällen bedient man sich der Clustering-Algorithmen. Hier gibt es verschiedene Methoden, die passend zur Problemsituation ausgewählt werden. Meist gibt eine von einer Datenanalytikerin oder einem -analytiker vorgenommene *explorative Datenanalyse* (engl. exploratory data analysis, EDA) die richtigen Hinweise zur Auswahl oder aber unterschiedliche Clustering-Verfahren werden direkt bei der EDA verglichen.
- **Genetische Algorithmen**, oder allgemeiner **evolutionäre Algorithmen,** sind in der KI-Forschung schon lange bekannt. Folglich gibt es auch hier viele Verfahren, die im Wesentlichen die Wirkungsweise der Evolution aus der Natur auf die Technik übertragen: Softwarevarianten, die sich durch *Mutationen* unterscheiden oder *Rekombinationen* neu bilden können, werden einer *Selektion* ausgesetzt. So *überleben* diejenigen Lösungen, die das gegebene Ziel besser erfüllen als die bisherigen. Letztlich hat man es wieder mit Optimierungsverfahren zu tun.

9. *https://lippke.li/support-vector-machine*, abgerufen am 26.07.2023.

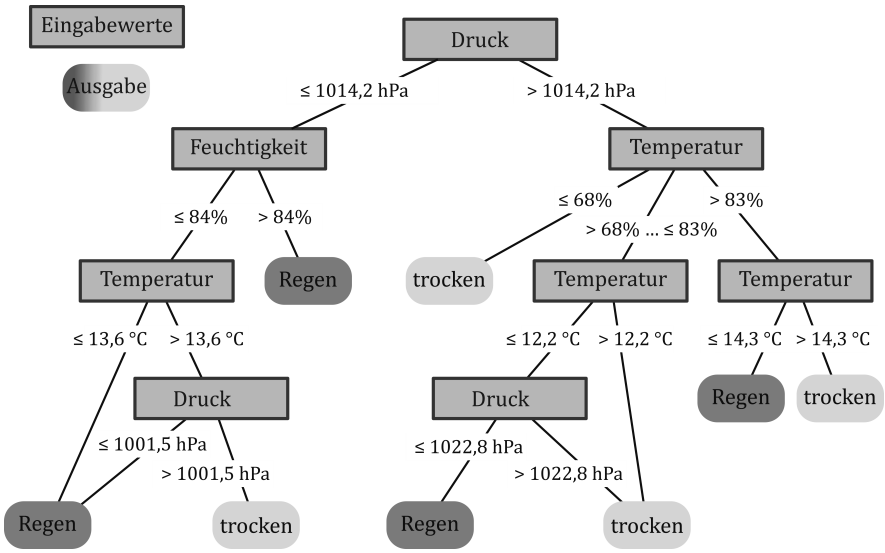


***Abb. 1–4*** *Grafische Darstellung eines Entscheidungsbaums zur Klassifikation einer Regenprognose (Ausgabe: Regen oder trocken) anhand der Eingabewerte relative Luftfeuchtigkeit, Temperatur und Luftdruck*

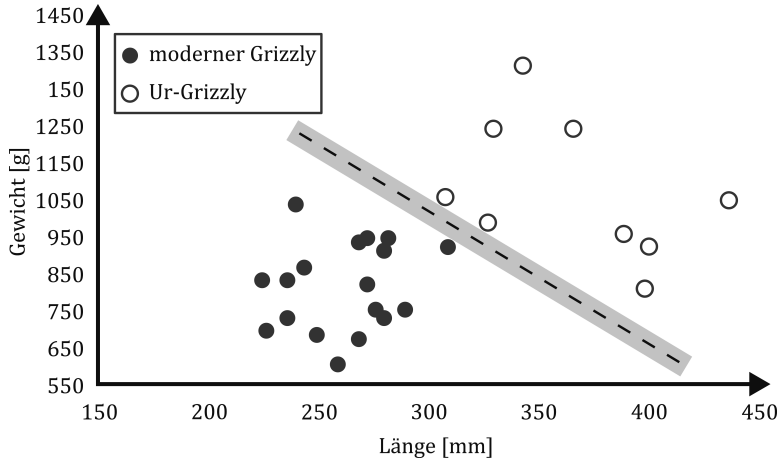


***Abb. 1–5*** *Die Support Vector Machine trennt Grizzly-Arten anhand von Knochenlänge und -gewicht.*

Du hast jetzt einige Techniken des maschinellen Lernens kennengelernt Im Folgenden zeigen wir dir, welche dieser Techniken wir für die Regenprognose in der Wetterstation ausgewählt haben.

**Praxisbeispiel 1–4: Wetterstation (Erweiterung)**

Weil wir für den vorgesehenen Standort (Portland in Australien, siehe Hinweis am Ende des Praxisbeispiels) viele Wetterdaten aus der Vergangenheit haben, entschließen wir uns, einen KI-Klassifikationsalgorithmus damit zu trainieren.

Zuerst trainieren wir einen sogenannten **Naive-Bayes-Klassifikator**. Dieser errechnet aus den historischen Daten, wie häufig beispielsweise bei Regenwetter am Tag zuvor bestimmte Temperatur-, Druck- oder Luftfeuchtigkeitsverhältnisse geherrscht haben. Aus diesen Wahrscheinlichkeiten ermitteln wir wiederum mathematisch, wie groß die Wahrscheinlichkeit für Regen am morgigen Tag ist bei Betrachtung des heutigen Wetters.

Der Klassifikator, den wir für unseren Beispielstandort Portland trainiert haben, sagt uns zu 77 % richtig voraus, ob es am nächsten Tag regnen wird.

Wir sind uns nicht sicher, ob dieser Klassifikator gut genug ist, und wollen als Alternative einen **Entscheidungsbaum** trainieren. Hier entscheiden wir uns für einen Trainingsalgorithmus, der unter dem Namen C4.5 oder J48 bekannt ist. Dieser erzeugt uns ein Schema, wie unsere Wetterdaten die Vorhersage Regen oder trocken für den morgigen Tag beeinflussen. Schrittweise werden unsere Messdaten Luftdruck, Luftfeuchtigkeit und Temperatur nacheinander herangezogen, um am Ende zu entscheiden, ob es regnen wird oder nicht (siehe Abb. 1–4).

Auch hier ermitteln wir, wie gut unsere Vorhersagen zutreffen, und erhalten einen ähnlichen Wert mit 76,5 %.

***Hinweis:*** Damit du dieses Beispiel auch selbst nachvollziehen kannst, haben wir uns öffentlich verfügbarer Wetterdaten von *kaggle.com* bedient:
*https://www.kaggle.com/datasets/sakshi20008/rainprediction-in-australia*.

## 1.5 KI-Entwicklungs-Frameworks

Im vorangehenden Abschnitt haben wir uns mit verschiedenen KI-Techniken beschäftigt. Dabei haben wir einen kleinen Einblick in die große Bandbreite von Techniken und den dazugehörigen Algorithmen gewonnen. Deren wissenschaftliche Erforschung wurde anfänglich eher an Universitäten oder in kleinen Forschungsabteilungen vorangetrieben. Seit einigen Jahren engagieren sich aber zunehmend große Firmen wie Amazon, Google, Facebook, Microsoft oder IBM – nicht zuletzt auch aus wirtschaftlichen Interessen. So liegt es nahe, dass heutige Entwicklungswerkzeuge für künstliche Intelligenz nicht nur an Hochschulen entstehen, sondern gerade von den eben genannten Global Playern im Maßstab ganzer Frameworks entwickelt und propagiert werden.

Solche KI-Entwicklungs-Frameworks haben jedes für sich Stärken und Schwächen oder sind auf bestimmte Domänen spezialisiert – so wie Certified Tester das bereits aus der Welt der Testwerkzeuge kennen. Die umfangreicheren KI-Entwick-

lungs-Frameworks[10] unterstützen uns bei vielen Arbeitsschritten im Entwicklungsprozess (siehe Abschnitt 3.2), wie z.B. bei

- der Analyse und/oder Aufbereitung von Daten,
- der Auswahl von KI-Algorithmen und
- dem Training und dem Kompilieren[11] von Modellen für die Prozessorarchitektur des Zielsystems. Zum Beispiel *Grafikprozessoren* (engl. graphics processing unit, GPU) oder *Tensor-Prozessoren*[12] (engl. tensor processing unit, TPU), die Berechnungen mit Vektoren oder Matrizen erheblich beschleunigen (siehe Abschnitt 1.6).

Daher ist es wichtig, dass du als einen der ersten Schritte ein adäquates Framework auswählst und neben der Unterstützung der Arbeitsschritte zusätzlich weitere Aspekte wie beispielsweise die Verfügbarkeit der Werkzeuge, deren Benutzerfreundlichkeit oder die unterstützten Programmiersprachen beachtest.

Die Auswahl eines KI-Entwicklungs-Frameworks hat aber noch einen weiteren Aspekt, der manchmal schwer einzuschätzen ist: Gerade bei Werkzeugen in diesem Bereich sehen wir aus unserer Erfahrung heraus eine sehr hohe Dynamik. Neue Entwicklungs-Frameworks kommen auf den Markt, werden in andere integriert oder verlieren an Bedeutung. So haben wir in Forschungsprojekten selbst erlebt, dass mehrere Entwicklungswerkzeuge noch innerhalb des ersten Projektjahrs mehrmals ausgetauscht wurden.

Dennoch gibt es KI-Entwicklungs-Frameworks, die insbesondere vom Lehrplan (April 2021) als beliebte Frameworks genannt werden und von denen einige von anhaltender Bedeutung sind:

- **Apache MxNet**
  Diese quelloffene Softwarebibliothek für Deep Learning (siehe Abschnitt 6.1) wird von der Apache Software Foundation (ASF) entwickelt und angeboten. Die ASF wird durch viele namhafte Sponsoren wie Amazon Web Services (AWS), Facebook, Google oder Microsoft unterstützt [URL: Mxnet].
- **IBM Watson Studio**[13]
  Die integrierte Sammlung von Werkzeugen und Services unterstützt die Entwicklung von KI-Lösungen – auch in der Cloud – von der Datenanalyse bis zu deren Bereitstellung [URL: IBM-a].

---

10. In diesem Buch verwenden wir – passend zum Lehrplan – den Begriff *KI-Entwicklungs-Framework*. In unseren Projekten benutzen wir auch den Begriff *KI-Entwicklungsumgebung*. Aus unserer Sicht ist das gleichbedeutend.
11. Mit Kompilieren meinen Softwareentwicklerinnen und -entwickler die Übersetzung einer Software aus einer Programmiersprache in eine andere. Oft aus einer höheren, wie z.B. Python oder C++, in Programmbefehle, die meist direkt von den Prozessoren des Zielsystems ausgeführt werden können.
12. Ein Tensor ist eine allgemein mehrdimensionale Datenstruktur, wie beispielsweise ein Vektor oder eine Matrix.
13. *https://www.ibm.com/cloud/watson-studio*.

- **Keras**
  Keras ist eine quelloffene Python-API, mit der Entwicklerinnen und Entwickler auf sehr hoher Beschreibungsebene schnell und unkompliziert Modelle trainieren und bereitstellen können. Keras nutzt selbst TensorFlow als darunterliegende ML-Plattform [URL: Keras].
- **TensorFlow**
  Diese quelloffene ML-Bibliothek von Google ist im Open-Source-Bereich eines der beliebtesten Frameworks und wird sowohl von Einsteigern oder in Schulungen als auch in großen Projekten verwendet. Mit ihr lassen sich ML-Modelle aus vorgegebenen Objekten zusammensetzen. Modell und Daten verknüpft sie zu einem Datenflussgraphen und unterstützt Entwicklerinnen und Tester von der Datenanalyse über das Training von Modellen bis hin zur Bereitstellung von KI-Komponenten für den Produktivbetrieb [URL: Tensorflow].
- **Scikit-learn**
  Die quelloffene Python-ML-Bibliothek setzt auf wissenschaftlichen Python-Bibliotheken wie *numpy* oder *scipy* auf und bietet Unterstützung von der Datenanalyse bis zur Evaluierung von trainierten Modellen [URL: Scikit-learn].
- **PyTorch**
  Diese quelloffene ML-Bibliothek [URL: Pytorch] wird von Facebook betrieben. Durch die ausgeprägten Debugging-Optionen wird PyTorch häufig auch in Wissenschaft und Forschung verwendet, meist für Anwendungen zur *Bild- oder Sprachverarbeitung* (engl. natural language processing, NLP). Die Bibliothek unterstützt sowohl Python- als auch C++-/Java-Schnittstellen.
- **CNTK**
  Das Microsoft Cognitive Toolkit (CNTK) ist ein quelloffenes Deep Learning Toolkit [URL: Microsoft], das jedoch seit dem letzten Release v2.7 nicht mehr aktiv weiterentwickelt wird.

**Praxistipp**

Die Auswahl des passenden KI-Entwicklungs-Frameworks ist selten einfach und meist mit Versuch und Irrtum verbunden. Die meisten Frameworks erlauben eine kostenfreie Installation – entweder für einen begrenzten Zeitraum oder Funktionsumfang oder sie sind sogar als Open Source frei verfügbar. Damit man sich bei der Entwicklung nicht verzettelt, empfehlen wir mit allen Projektbeteiligten die Rahmenbedingungen möglichst gut einzugrenzen:

- Welche Aufgaben soll das Framework genau übernehmen?
  Je genauer die Aufgaben bekannt sind, desto klarer kannst du die geeigneten Frameworks selektieren.

→

- Welchen Nutzen erhoffst du dir mit dem Framework?
  Das Framework wird seinen Zweck erst dann gezielt erfüllen, wenn du dir dessen bewusst bist.
- Wer soll das Framework benutzen?
  Datenanalystinnen und -analysten werden andere Ansprüche an das Framework haben als Entwicklerinnen, Tester oder Projektmanagerinnen und -manager.
- Welche Schnittstellen und Programmiersprachen muss das Framework abdecken?
  Möglicherweise sind im Unternehmen etablierte Werkzeuge, Sprachen oder Schnittstellen bereits festgelegt und die Kompatibilität mit diesen hat hohe Priorität.
- Was kennzeichnet das Produkt? Soll z.B. ein Cloud-Service, eine Datenbankanwendung oder ein Edge-Device mit GPU/TPU-Hardware entwickelt werden?
  Davon abhängig sollten unterschiedliche Arbeitsschritte im Entwicklungsprozess durch das Framework unterstützt werden.
- Welche Dateneigenschaften müssen berücksichtigt werden?
  Wichtige Eigenschaften sind z.B. die Datenmenge, Variabilität, Häufigkeit der Aktualisierung, der Datendurchsatz oder eine notwendige Archivierung, um die Datentransparenz zu stärken.

Erst danach kannst du die unterschiedlichen KI-Entwicklungs-Frameworks sinnvoll evaluieren und mit einem einfach gehaltenen Proof of Concept beginnen.

## 1.6 Hardware für KI-basierte Systeme

Wir befassen uns in diesem Abschnitt mit der Frage, wann für KI-basierte Systeme spezielle Hardware benötigt wird und welche Eigenschaften dieser Hardware dafür besonders hilfreich sein werden.

Bei Systemen mit künstlicher Intelligenz denken viele Menschen unwillkürlich an Kinoszenen mit Supercomputern, riesige Datenmengen, großen Speicherbedarf und aufwendige und rechenintensive Algorithmen. Es kann eine große Rechenleistung erfordern, einen Algorithmus mithilfe des maschinellen Lernens (siehe Abschnitt 1.4) und mit einer umfangreichen Datenbank zu trainieren. Dagegen kann ein so entstandenes und fertig trainiertes ML-Modell aber so klein sein, dass man es auf einem einfachen Universalprozessor implementieren und darauf laufen lassen kann.

Nutzt man beispielsweise Endgeräte im Bereich von Embedded Systems, haben diese üblicherweise nur begrenzte Systemressourcen wie Rechenleistung, Daten- oder Programmspeicher. In Projekten sprechen wir dann meist von *Edge-Komponenten*[14] oder *Edge-Devices*. Für diese trainiert man daher das ML-Modell zuerst abseits auf hochperformanten Systemen, die z.B. auch als Service in der Cloud angeboten werden. Ist das Training abgeschlossen, reicht es, das fertige ML-Modell

14. Der Begriff *Edge* meint in der Terminologie des *Internet of Things* (IoT) diejenigen Komponenten, die sich quasi am Rande des weitverzweigten *Netzwerks der Dinge* befinden.

auf das Endgerät zu übertragen und dort auszuführen – insbesondere, wenn keine Verbindung zum Internet besteht.

Eine Digitalkamera mit einer Erkennung lächelnder Gesichter als Selbstauslöser kann beispielsweise auf einem stromsparenden Bildprozessor laufen, benötigt aber zum Trainieren eine umfangreiche Datensammlung und performante Trainingshardware.

#### Eigenschaften von Prozessorhardware für KI-Systeme

Auch wenn sich die Rechenaufwände in der Trainings- und der Anwendungsphase von ML-Modellen unterscheiden, haben diese Gemeinsamkeiten, wie gleichförmig wiederholte arithmetische Operationen (Multiplikation, Addition usw.), die auf großen Datenmengen angewendet werden. Daher profitieren maschinelles Lernen und ML-Modelle gleichermaßen, wenn Prozessoren mit folgenden Eigenschaften verwendet werden:

- **Arithmetische Operationen mit geringer Genauigkeit**
  Für Berechnungen in ML-Modellen reicht im Regelfall eine geringe Genauigkeit von z.B. 8 Bits statt der üblichen 32 Bits aus. Die arithmetische Recheneinheit des Prozessors ist dadurch schneller, aber auch sparsamer in Energie- und Flächenverbrauch auf einem Prozessorchip. So finden mehr arithmetische Recheneinheiten auf der Chipfläche Platz (siehe parallele Verarbeitung weiter unten).
- **Operationen mit größeren Datenstrukturen (Vektoren, Matrizen)**
  Spezielle arithmetische Recheneinheiten erlauben es in wenigen oder gar einem Schritt, mehrere Zahlen parallel zu multiplizieren und deren Ergebnisse zu addieren. Damit kann ein solches Rechenwerk Vektor- oder Matrixmultiplikationen schnell und effizient durchführen (siehe Abschnitt 1.5).
- **Massiv parallele (nebenläufige) Verarbeitung**
  Oft weisen Algorithmen des maschinellen Lernens und von ML-Modellen einen hohen Grad an Parallelisierbarkeit auf. Je mehr arithmetische Recheneinheiten oder Rechenkerne parallel zu Verfügung stehen, desto schneller können ML-Modelle aus- oder deren Training durchgeführt werden.

#### Vorteile spezieller KI-Hardware

Die eben genannten Eigenschaften findet man typischerweise in Grafikprozessoren, die zur Darstellung von 2D- und 3D-Bildern bereits Ende der 1980er-Jahre entwickelt und in Grafikkarten verbaut wurden. Heutige High-End-Grafikprozessoren integrieren sogar viele Tausend dieser *arithmetischen Prozessoren* (Kerne, engl. cores). Diese auf massiv parallele Vektor- und Matrizenoperationen spezialisierten Rechenkerne sind daher ideal für Anwendungen des maschinellen Lernens geeignet.

Im Folgenden schauen wir uns dazu im Vergleich aktuelle S*tandardprozessoren* (engl. central processing units, CPUs) an: Diese glänzen zwar mit enorm hohen Taktraten bei mehreren GHz, haben dafür aber nur wenige Rechenkerne. Das ist zwar ideal für komplexe Programmstrukturen allgemeiner Problemlösungen, doch ML-Anwendungen haben dadurch kaum einen Geschwindigkeitsvorteil. Die Nutzung von GPUs ist für ML meist die bessere Wahl.

GPUs sind zwar weit verbreitet und durch viele Entwicklungswerkzeuge sehr gut unterstützt, stellen aber nicht die einzige Möglichkeit dar, ML-Anwendungen signifikant zu beschleunigen. Je genauer ein Hersteller einen Algorithmus für eine Anwendung festgelegt hat, desto effizienter kann eine genau dafür entwickelte Hardware – die sogenannten *anwendungsspezifischen integrierten Schaltungen* (engl. application specific integrated circuits, ASICs) – diese umsetzen. Derartige ASICs werden für die unterschiedlichsten Anwendungen schon seit langer Zeit entwickelt. Dies reicht von einzelnen Algorithmen bis hin zu kompletten technischen Systemen, die nicht nur einzelne Aufgaben integrieren, sondern die Funktion eines ganzen Geräts mit digitaler und analoger Schaltungstechnik und Sensorik zu einem *System-on-Chip* (SoC).

Speziell für KI-Anwendungen entwickeln Chiphersteller mit solchen ASICs unterschiedlichste Hardwarelösungen, die wir hier nur exemplarisch aufführen:

- Verschiedene Hersteller entwickeln Prozessorchips, die sich durch eine besonders hohe Anzahl von Rechenkernen auszeichnen und durch spezielle Daten- und Speicherzugriffe ideal für Hochleistungsrechner in Rechenzentren geeignet sind.
- Für neuronale Netze gibt es bereits ASICs, deren interne Daten- und Verarbeitungsstruktur mehr neuronalen Netzen gleicht und nicht der klassischen Von-Neumann-Architektur[15] gängiger Universalprozessoren folgt. Gerne bezeichnen deren Hersteller solche Chips auch als *neuromorphe Prozessoren* [Davies et al. 2021].
- Andere ASICs legen beispielsweise die Parameter von Modellen nicht im *allgemeinen Speicher* (engl. random access memory, RAM) ab, sondern in dedizierten und instantan zugreifbaren Speicherzellen des Chips, den Registern. Die Inhalte solcher Register können dann direkt und ohne Zeitverlust für Rechenoperationen benutzt werden.

Die beiden zuletzt aufgelisteten Eigenschaften erlauben es zudem, solche ASICs energie- und platzsparend zu entwerfen, sodass diese ideal für Edge-Anwendungen geeignet sind. Das vorausgehende Training solcher KI-Anwendungen benötigt meist dennoch Systeme mit hoher Rechenleistung, z.B. in der Cloud.

15. Siehe auch: *https://de.wikipedia.org/wiki/Von-Neumann-Architektur.*

### KI-Hardwareunterstützung als Geschäftsmodell

Bemerkenswert ist vor allem, dass die Entwicklung von KI-spezifischen Prozessoren nicht mehr allein durch die klassischen Unternehmen der Halbleiterindustrie getrieben wird. Hier eine kleine Auswahl von Herstellern, die auch im Lehrplan (April 2021) genannt werden:

- NVIDIA ist als Hersteller von Computergrafik-Hardware seit Langem auf dem Markt etabliert und bietet daher eine große Palette von GPU und KI-spezifischen Prozessoren und Modulen an, die beispielsweise auf den (Stand August 2022) Volta-, Ampere- oder Hopper-Architekturen basieren [URL: Nvidia-a].
- Google hat sich erst in jüngerer Zeit auf dem Halbleitermarkt für ASICs etabliert und bietet bereits konkurrenzfähige Prozessoren und Module zum Training der Modelle und zur Anwendung der trainierten KI-Modelle – der sogenannten *Inferenz* – an. Googles Cloud TPUs [URL: Google-tpu] unterstützen speziell Rechenzentren in der Cloud zum Training von Modellen, aber auch, um KI als Service (siehe Abschnitt 1.7) für Anwender verfügbar zu machen. Die Edge TPUs [URL: Google-etpu] der Coral-Plattform sind speziell für die Ausführung von KI auf individuellen Geräten entwickelt worden.
- Intel® bietet als erfahrener Hersteller von Prozessoren ebenfalls Hardwareunterstützung für KI-Entwicklung und -Anwendung: Bekannt sind die etablierten Intel®-GPUs zur Beschleunigung von Algorithmen. Spezialisierte Prozessoren wie die Movidius™-Myriad™-VPU unterstützen neuronale Netze und Bildverarbeitung.[16]
- Habana® ist eine Intel-Tochter und bietet mit dem Gaudi2 einen hochperformanten Prozessor für Rechenzentren und im Speziellen für Deep Learning (siehe Abschnitt 6.1) an und konkurriert dabei direkt mit NVIDIA[17].
- Mobileye® ist seit 2017 ebenfalls eine Intel-Tochter und stellt mit der EyeQ®-Familie SoC-Geräte für rechenintensive Bildverarbeitung her [URL: Mobileye]. Der Einsatz dieser Systeme konzentriert sich auf die energieeffiziente Nutzung im Automobilbereich.
- Apple verbaut bereits seit mehreren Jahren den Bionic-Chip, ein SoC, in iPhones und iPads [Wiltz 2019]. Dieser vereinigt mehrere Universal-CPUs, GPUs sowie spezielle Rechenkerne für KI-Anwendungen. Der aktuelle A15[18] (Stand August 2022) enthält eine sogenannte 16-Core Neural Engine.

16. *https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu.html*, abgerufen am 08.08.2023.
17. *https://habana.ai*, abgerufen am 08.08.2023.
18. *https://de.wikipedia.org/wiki/Apple_A15_Bionic*.

- HiSilicon, ein zum Huawei-Konzern gehörender ASIC-Hersteller, produzierte bereits 2017 mit dem Kirin 970[19] ein SoC für Smartphones mit spezieller KI-Unterstützung. Neben acht CPUs und 12 GPUs hat auch dieser einen für neuronale Netze zugeschnittenen Prozessor [Huawei Press Release 2017]. Seitdem sind bereits mehrere Nachfolger zur Kirin-Prozessorfamilie hinzugekommen. Huawei stellt außerdem mit der Ascend- und Atlas-Reihe ebenfalls Prozessoren und Module für Edge- und Cloud Computing her. [20]

## 1.7 KI als Service (AI as a Service, AIaaS)

Softwaredienste begegnen uns allen im IT-Umfeld schon seit vielen Jahren. Beispielsweise lassen Start-up-Unternehmen häufig ihre Firmenwebseite von einem spezialisierten IT-Anbieter betreiben oder nutzen nur dessen Rechen- oder Speicherkapazitäten als Service. Dies verringert für das Start-up die anfänglich hohen Kosten, eine eigene Hardwareinfrastruktur aufzubauen. Man kann daher leicht nachvollziehen, dass auch das Interesse an Services rund um die Entwicklung und den Betrieb von KI-Anwendungen wächst. Unternehmen, die hier einsteigen, können sich so schneller und kostengünstiger an eine für sie neue Technologie herantasten.

Solche KI-Dienste, die im Fachjargon als *AI as a Service* (*AIaaS)*[21] bezeichnet werden, haben nicht unwesentlich zur rasant zunehmenden Entwicklung und Nutzung von KI-basierten Systemen beigetragen. Die Service-Unterstützung der KI-Entwicklung in einem Unternehmen kann dabei sehr unterschiedliche Ausprägungen haben, wie es die folgenden Szenarien exemplarisch zeigen:

- Eine Organisation entwickelt ihre KI-Komponenten selbst, verwendet zu deren Betrieb aber einen KI-Inferenz-Service[22] eines IT-Dienstleisters.
- Ein Unternehmen nutzt zuerst ein von einem Drittanbieter bereits fertig trainiertes und als Dienst im Internet bereitgestelltes Modell. Nach einiger Zeit hat das Unternehmen genug Erfahrung und nutzt stattdessen ein vortrainiertes Modell (Open Source oder von Drittanbietern) und trainiert dieses durch eigene Trainingsdaten auf seine fachspezifische Anwendung.
- Die Entwicklungsabteilung einer Firma nutzt zum Training ihres KI-Modells ein als Service bereitgestelltes KI-Entwicklungs-Framework, betreibt das Modell aber selbst im eigenen Haus.

---

19. *https://www.hisilicon.com/en/products/Kirin*, *https://www.hisilicon.com/en/products/Kirin/Kirin-flagship-chips/Kirin-970*, abgerufen am 08.08.2023.
20. *https://e.huawei.com/en/products/servers/ascend*, abgerufen am 08.08.2023.
21. Die Begriffe AI as a Service (AIaaS), KI als Service, KI-Service und KI-Dienst verwenden wir in diesem Buch synonym.
22. Unter *Inferenz* versteht man die Anwendung eines trainierten KI-Modells im Betrieb.

- Ein großes Unternehmen nutzt für ein umfangreiches System einen hybriden Ansatz und verwendet Modelle zur Bilderkennung von einem Drittanbieter, entwickelt aber zugleich eigene KI-Komponenten mithilfe sensibler und firmeninterner Daten.

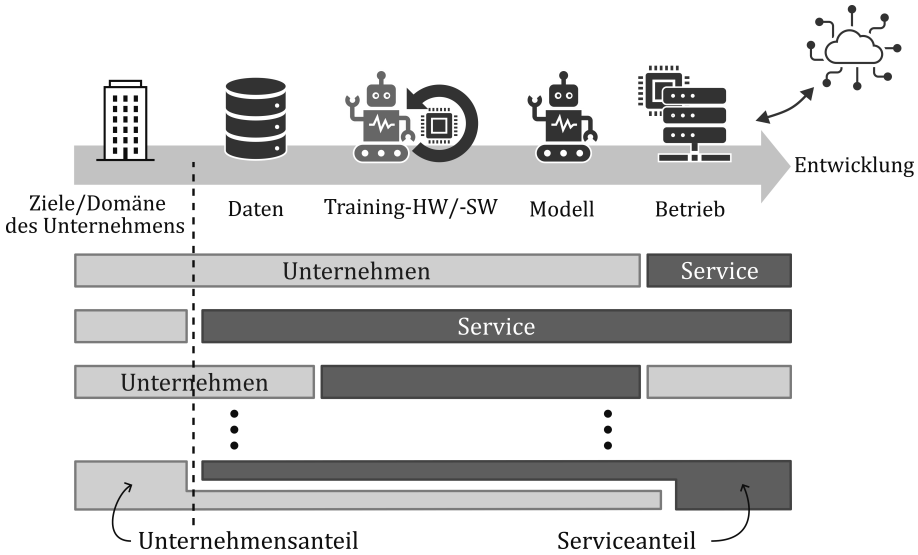


***Abb. 1–6*** *Verschiedene Variationen, mit denen ein Unternehmen KI-Dienste als komplementäre Services zur firmeneigenen Entwicklung von KI-basierten Systemen nutzen kann.*

Abbildung 1–6 zeigt schematisch die Zusammenhänge zwischen den eigenen Aufwänden eines Unternehmens, das ein KI-basiertes System erstellen möchte, und den KI-Services, die es in Anspruch nehmen kann.

Viele namhafte Anbieter wie z. B. Amazon, Microsoft oder Google stellen eine breite Palette Services wie beispielsweise Sprach-, Text- oder Gesichtserkennung bereit. Interessierte Unternehmen oder auch einzelne Personen können diese ohne besondere KI-Kenntnisse nutzen und ohne großen Mehraufwand in ihre eigenen Anwendungen oder Cloud-Dienste einbinden. Dabei müssen sie keinerlei spezielle Hardware oder Infrastruktur bereitstellen und sie benötigen auch keine speziellen Fachkenntnisse.

Kleine Unternehmen, die gerade erst in die KI-Technik einsteigen, profitieren besonders von der Erfahrung großer und etablierter Modellanbieter. Die als Service angebotenen ML-Modelle sind in vielen Fällen mit deutlich umfangreicheren und vielfältigeren Daten trainiert worden als dies einsteigenden Unternehmen oder der Open-Source-Gemeinde üblicherweise möglich ist.

#### Vertragskomponenten für KI-Services

Von außen und als IT-Service betrachtet unterscheiden sich KI-Modelle kaum von klassisch programmierten Softwarediensten. Daher ist es nicht unüblich, wenn für solche KI-Dienste ähnliche Verträge wie für klassische Software-as-a-Service(SaaS)-Angebote verwendet werden. Solche Verträge bestehen zumeist aus mehreren Anteilen, die verschiedene Aspekte des Vertrags regeln. Im Folgenden haben wir einige Beispiele aufgelistet:

- **Service Level:** Vorrangig interessiert die Nutzerinnen und Nutzer eines solchen Dienstes dessen Art und Güte. In Verträgen findet man diese unter dem Begriff der *Service-Level-Vereinbarung* (engl. Service Level Agreement, SLA). Meist sind dort Angaben zur Verfügbarkeit des Dienstes (z.B. 99,99 % Mindestverfügbarkeit innerhalb eines definierten Zeitraums) oder Reaktionszeiten für die Behebung von Fehlern im Service festgelegt.

  Selten findet man jedoch Servicezusagen, die eine Güte der Funktion selbst benennen (z.B. die Genauigkeit einer Spracherkennung für Worte oder Sätze), da diese oft nur schwer vor dem Einsatz in der Zielumgebung abschätzbar sind. Solche sogenannten funktionalen Leistungsmetriken beschreiben wir genauer in Kapitel 5.
- **Kostengestaltung:** Als weitere Komponente ist die Bezahlweise geregelt, die häufig als Abonnement angeboten wird. Dabei erhalten Kunden für einen festen Preis einen zeitlich begrenzten Servicezugang auf einem zuvor festgelegten Service Level. Diese Form kennst du schon aus anderen Servicebereichen wie etwa Mobilfunkverträgen. Hält der Anbieter seine Zusagen in puncto Verfügbarkeit oder Reaktionszeiten nicht ein, bietet dieser zumeist Gutschriften im Rahmen der Dienste an.
- **Haftungsfragen:** Im Falle eines Schadens, der z.B. durch eine fehlerhafte Funktion des KI-Service entstanden ist (die Spracherkennung hat ein falsches Wort erkannt und der Webshop hat einen falschen Artikel in die Bestellung aufgenommen), begrenzen die Verträge üblicherweise die Höhe der Haftung. Das wiederum bedeutet, dass der Markt kaum Services anbietet oder nutzt, die ein hohes Schadenspotenzial haben. Man sieht also zumeist AIaaS mit vergleichsweise geringem Risiko, bei denen der Ausfall eines Service keinen großen Schaden anrichten würde.

Wenn man als Softwareentwicklerin oder Tester einen KI-Dienst in seiner Anwendung verwenden möchte, will man zuerst feststellen, ob dieser KI-Dienst für das eigene Vorhaben tauglich ist. Das Gleiche gilt, wenn man KI für die Testdurchführung verwendet. Häufig erlaubt der Anbieter uns Nutzenden nach einer Registrierung eine initiale kostenlose Testphase. In dieser Zeit kann man dann den Dienst auf seine Eignung überprüfen. Dabei schaut man nicht nur auf Schnittstellen und die Integration in der Anwendung. Man validiert auch, ob die

Funktion und Leistung des Service, wie z.B. die Genauigkeit einer Bildklassifikation oder eines Sprachassistenten, für die Anwendung geeignet ist. Gerade solche Prüfungen sind oft unerlässlich, denn meist fehlt für diese Informationen die eigentlich notwendige Transparenz über den angebotenen Dienst (siehe die Abschnitte 2.7 und 7.5).

#### Beispiele für KI-Services

Bereits in den Abschnitten 1.5 und 1.6 wurden beispielhaft Anbieter und Hersteller für KI-Entwicklungs-Frameworks und KI-spezifische Hardware aufgelistet. Auch die hier folgende Liste hat nur exemplarischen Charakter und wird vom CT-AI-Lehrplan (Stand April 2021) genannt:

- Der IBM Watson Assistant ist ein KI-Chatbot, den Unternehmen gegen eine Servicegebühr ihren Kunden anbieten können. Die Preisgestaltung richtet sich z.B. nach der Zahl der aktiven Nutzerinnen und Nutzer pro Monat [URL: IBM-b].
- Unter Googles Cloud-Produkten findet man in der Rubrik *KI und maschinelles Lernen* eine ganze Reihe von Services. Beispielsweise eine textbasierte KI, die Informationen aus Dokumenten extrahiert und strukturiert und dafür u.a. *grafische Texterkennung* (engl. optical character recognition, OCR) und eine Analyse von Formularen verwendet. Die Preisgestaltung orientiert sich hier an der Zahl der verarbeiteten Seiten [URL: Google-ai].
- Der CodeGuru Reviewer von Amazon ist ein Werkzeug für die statische Analyse von Quelltexten in Java oder Python. Das Tool erlaubt Entwicklerinnen und Entwicklern ihren Quellcode zu verbessern, indem das Tool den Code u.a. auf duplizierte Codepassagen, potenzielle Sicherheitsangriffe oder mögliche Parallelitätsprobleme hin überprüft. Die Preisgestaltung richtet sich hier nach der Anzahl der Quelltextzeilen [URL: AWS].
- Microsofts Azure Cognitive Search ist eine KI-basierte Suche in einer Dokumentensammlung. Über die reine textuelle Suche hinaus werden relevante Treffer auch über semantische Zusammenhänge erkannt und über unterschiedliche Medien wie Bild, Text und verschiedene Dateiformate analysiert. Die Preise orientieren sich hier am Umfang der analysierten Dokumentdaten sowie der gewünschten Performanz der Suche [URL: Azure].

## 1.8 Vortrainierte Modelle

Eine KI, die zuverlässig das tut, was sie soll, fällt nicht vom Himmel. Sie muss mühsam mit Wissen gefüttert, d.h. trainiert werden. Das maschinelle Lernen und insbesondere *künstliche neuronale Netze* (kNNs, engl. artificial neural networks, aNNs; im Projekt sprechen wir häufig von *neuronalen Netzen* und meinen *künst-*

*liche neuronale Netze*) sind aktuelle Techniken, die es ermöglichen, dass KI-Modelle während eines Trainings selbstständig lernen. Sowohl das Aufbereiten von Daten zum Training als auch das Training selbst können besonders aufwendig und teuer sein (siehe Kap. 3). Gerade die Aufbereitung der Trainingsdaten kann viel manuelle, nicht automatisierbare Arbeit bedeuten und sehr viel Personal binden. Das Training selbst kann schnell sehr viel Rechenleistung benötigen. Nicht in jedem Unternehmen sind diese Ressourcen ausreichend vorhanden oder einfach bezahlbar. Bei der Verwendung von künstlichen neuronalen Netzen und Random Forest ist es gängig, ein vortrainiertes Modell als eine kostengünstige und effektive Grundlage zu wählen.

Andere haben dann bereits das aufwendige Training durchgeführt und ein neuronales Netz erstellt. Auf diesem setzt man auf.

#### Der Nutzen vortrainierter Modelle

Ein vortrainiertes Modell ist besonders gut geeignet, wenn es bereits ähnliche Anforderungen wie das benötigte Modell erfüllt. Du kannst dir beispielsweise ein Spracherkennungssystem vorstellen, das für Deutsch vortrainiert wurde und den deutschen Wortschatz gut erkennt. Solch ein Modell kannst du für eine ganze Reihe verschiedener Sprachverarbeitungsaufgaben weiterverwenden.

Verwendet man ein bereits trainiertes Modell ohne Änderungen, kann man es im Idealfall einfach in das KI-basierte System einbetten oder es als Dienst nutzen (siehe Abschnitt 1.7).

**Praxistipp**

Unabhängig davon, ob du ein vortrainiertes Modell als Dienst nutzen oder es in das System einbetten möchtest, solltest du sicherstellen, dass Schnittstellen- und Datenformate des Modells oder der Dienste mit deinen restlichen Systemkomponenten kompatibel sind.

Wenn kein vortrainiertes Modell zur Verfügung stehen sollte, muss dennoch nicht zwingend *bei null* angefangen werden. Zu vielen unterschiedlichen Anwendungsbereichen ist eine ganze Reihe gut aufbereiteter Open-Source-Datensätze zu finden, die sich für ein Vortraining eines eigenen Modells sehr gut eignen. Der wahrscheinlich bekannteste Open-Source-Datensatz ist der ImageNet-Datensatz, der über 14 Millionen Bilder mit über 1000 Kategorien enthält (Stand März 2021).

Indem vortrainierte Modelle verwendet werden, verringert man in Projekten das Risiko, erhebliche Ressourcen unnötig zu verschwenden. Man kann mit wenig Aufwand feststellen, ob die KI realisierbar ist oder nicht. So kann man nicht nur Kosten sparen, sondern erhöht gleichzeitig die Chance, ein erfolgreiches und effektives Modell zu erhalten.

### Transfer-Lernen

Wenn das vortrainierte Modell nicht genau den Anforderungen entspricht, kann man auf einem vortrainierten Modell ein verkürztes Training mit einem kleineren Datensatz aufbauen. Dann sprechen wir von Transfer-Lernen. Die Funktionen des vortrainierten Modells werden dann durch ein gezieltes Training passend zu den Anforderungen erweitert und/oder fokussiert. Um beim Beispiel des Spracherkennungssystems des vorangegangenen Abschnitts zu bleiben: Wenn die Fähigkeit des Modells für bestimmte Fachbegriffe eines gewünschten Fachgebiets (z.B. medizinische Begriffe) noch nicht ausreicht, dann kann man es auf diese Fachbegriffe erweiternd trainieren. Dafür benötigt man nur noch einen deutlich kleineren Trainingssatz.

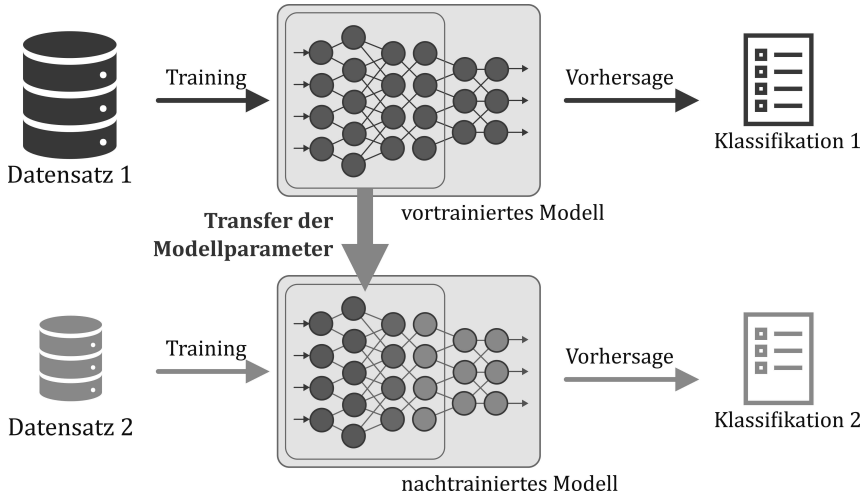


***Abb. 1–7*** *Transfer-Lernen*

Für das *Transfer-Lernen* sind künstliche neuronale Netze mit vielen Schichten, sogenannte *tiefe neuronale Netze* (engl. deep neural network), sehr gut geeignet (siehe Abb. 1–7): Stell dir einen Bildklassifikator vor, der die Architekturstile von Gebäuden erkennt. Der Klassifikator besteht aus einem tiefen neuronalen Netz, das aus mehreren aufeinanderfolgenden Neuronenschichten zusammengesetzt ist. Dabei kannst du davon ausgehen, dass die vorderen Schichten grundlegende Funktionalitäten übernehmen, wie die Unterscheidung von geraden und gekrümmten Linien. Spätere Schichten übernehmen dann spezialisiertere und abstraktere Aufgaben, wie die Unterscheidung zwischen architektonischen Gebäudetypen. Wenn du einen Bildklassifikator entwickeln willst, der neben dem Gebäudestil auch den Architekten benennen soll, kannst du den Bildklassifikator zur Bestimmung der Architekturstile wiederverwenden. Alle vorderen Schichten des Bildklassifikators übernimmst du unverändert. Die letzten Schichten werden mit einem Datensatz, der Informationen über den Architekten der Bauwerke ent-

hält, neu trainiert. Dieses Training zum Anpassen des Klassifikators benötigt deutlich weniger Daten als das Vortraining. Es gibt viele vortrainierte Open-Source-Modelle, insbesondere aus dem Bereich der akademischen Forschung. Die Modelle Inception, VGG, AlexNet und MobileNet, die alle auf der ImageNet-Datenbank[23] basieren, sind sicherlich die bekanntesten Modelle zur Bildklassifikation, die sich auch sehr gut als vortrainierte Modelle für ein Transfer-Lernen eignen. Große Bekanntheit haben das Sprachmodell BERT[24] von Google und dessen Erweiterung RoBERTa[25] von Facebook erlangt. Sie haben als vortrainierte Modelle im Bereich von NLP große Erfolge im Transfer-Lernen aufweisen können. Ob das Transfer-Lernen in einem konkreten Projekt effektiv ist, hängt weitgehend von der Ähnlichkeit der Funktionen des ursprünglichen Modells und des benötigten Modells ab. Zum Beispiel kannst du einen Bildklassifikator, der Katzenrassen identifiziert, relativ einfach modifizieren, sodass er anschließend Hunderassen identifiziert. Geht es hingegen darum, dass dieser Bildklassifikator Architekturstile von Gebäuden erkennen soll, ist das deutlich schwieriger.

#### Risiken bei der Verwendung von vortrainierten Modellen und Transfer-Lernen

Gerade weil vortrainierte Modelle oder das Transfer-Lernen in Projekten bei der Entwicklung von KI-basierten Systemen immer wieder verwendet werden, muss man sich einiger Risiken bewusst sein:

- **Reduzierte Transparenz:** Wenn das Vortraining nicht begleitet wird, hat man nur noch eingeschränkt Zugang zu den verwendeten Lernalgorithmen und den Trainingsdaten. Daher kann ein vortrainiertes Modell im Vergleich zu einem selbst entwickelten Modell intransparenter sein.
- **Unzureichende Funktionalität:** Wenn ein vortrainiertes Modell verwendet wurde, dann hängt die Qualität der fertigen KI auch wesentlich vom vortrainierten Modell ab. Wenn sich die geforderte Funktionalität stark von der Funktionalität des vortrainierten Modells unterscheidet, funktioniert die KI nicht gut. Das liegt dann aber nicht an einer schlechten Qualität des vortrainierten Modells, sondern an den unterschiedlichen Anforderungen. Da es sich bei KIs um komplexe Systeme handelt (siehe reduzierte Transparenz), kann es sein, dass die Gründe hierfür auch von Data Scientists und Analystinnen und Analysten nicht immer verstanden werden.
- **Unterschiede in der Vorverarbeitung der Inputdaten:** Häufig werden Daten aufbereitet, bevor sie der KI als Input übergeben werden (siehe Abschnitt 4.1). Wenn diese Aufbereitungsschritte des vortrainierten Modells nicht in die

23. *http://www.image-net.org/*, abgerufen am 08.08.2023.
24. *https://github.com/google-research/bert*, abgerufen am 08.08.2023.
25. *https://github.com/pytorch/fairseq/tree/main/examples/roberta*, abgerufen am 08.08.2023.

Betriebsumgebung des Systems übernommen werden, kann sich das negativ auf die Leistung der KI auswirken. Die dann im System verwendeten Daten werden vom Modell nicht so gut verarbeitet, weil es andere Daten *gewohnt* ist.

- **Übertragene Unzulänglichkeiten:** Unzulänglichkeiten wie ein Bias (siehe Abschnitt 2.4) werden vom vortrainierten Modell mit einer hohen Wahrscheinlichkeit an das endgültige Modell vererbt. Wenn diese Unzulänglichkeiten nicht offensichtlich sind oder vorher dokumentiert wurden, kannst du diese nur durch spezifische Tests finden (siehe Abschnitt 8.3). So können spezifischere Tests bei der Verwendung von vortrainierten Modellen erforderlich sein, um dieses Risiko zu mindern.
- **Übertragene Angreifbarkeit:** Die Modelle, die durch Transfer-Lernen erstellt werden, haben höchstwahrscheinlich die gleichen Schwachstellen wie das vortrainierte Modell. Wenn bekannt ist, dass ein KI-gestütztes System ein bestimmtes vortrainiertes Modell enthält oder auf einem basiert, können potenzielle Angreifer die bekannten Schwachstellen ausnutzen (siehe Abschnitt 9.1.1).

Einige der aufgeführten Risiken können unter Umständen vorher im Rahmen einer Risikobetrachtung beurteilt und sogar abgemildert werden. Für eine Risikobetrachtung ist eine ausführliche Dokumentation zu dem vortrainierten Modell sehr hilfreich (siehe Abschnitt 7.5).

## 1.9 Normen, Vorschriften und KI

Testerinnen und Tester wissen, dass viele heutige Produkte die unterschiedlichsten Standards, Normen oder Vorschriften erfüllen müssen. Insbesondere gilt dies für Softwareprodukte oder -komponenten, an die wir als Nutzerinnen und Nutzer oder Betroffene besonders hohe Anforderungen stellen und für die der Gesetzgeber deswegen entsprechende Standards, Normen oder Vorschriften fordert.

Beispielsweise muss jede Software, die in Straßenfahrzeugen eingesetzt wird, u.a. nach den Normen *ISO 26262* (functional safety) [ISO 26262-1] und *ISO/PAS 21448* (safety of the intended functionality – SOTIF) [ISO/PAS 21448] entwickelt und getestet werden. KI-basierte Systeme oder Komponenten sind von solchen Normen nicht ausgenommen.

Normen als solche sind zunächst nur von einem Expertenteam im Konsens erarbeitete Dokumente, die beispielsweise Produkteigenschaften oder Entwicklungsprozesse von Produkten beschreiben. Solche Normen können von Unternehmen jederzeit auf freiwilliger Basis befolgt werden. Eine verbindliche Vorschrift werden diese erst durch Gesetze oder Verordnungen staatlicher Stellen oder durch vertragliche Vorgaben. Würde z.B. ein Fahrzeug nicht nach den oben erwähnten ISO-Normen entwickelt werden, würde in einigen Ländern ein Verstoß gegen geltendes Recht begangen werden.

#### Ebenen der Normung mit Bezug zu KI

Derzeit (Mitte 2023) gibt es national und international eine noch kleine, aber rasch wachsende Zahl an Normen, die direkt oder indirekt KI-basierte Systeme betreffen. Die meisten davon befinden sich in der Entwicklung. Von einem Teil der Normen und Normungsgremien wollen wir dir im Folgenden einen kurzen Eindruck vermitteln.

#### Internationale Gremien

Viele Normen auf internationaler Ebene werden durch die ISO und das IEC herausgegeben. Deren gemeinsames Komitee für IT-Themen, das *Joint Technical Committee of IEC and ISO on information technology* (ISO/IEC JTC1) erarbeitet auch Normen mit Bezug zu KI. Ein Unterausschuss (ISO/IEC JTC1/SC7) hat beispielsweise in seiner Normenreihe ISO/IEC 29119 zum Softwaretest auch einen technischen Bericht mit Leitlinien zum Test von KI-basierten Systemen veröffentlicht [ISO/IEC TR 29119-11]. Ein weiter Unterausschuss (ISO/IEC JTC1/SC42 – Artificial intelligence) wurde 2017 speziell zur Erarbeitung von KI-bezogenen Normen eingerichtet (siehe Abb. 1–8). Hier ist im Juli 2022 eine grundlegende Norm ISO/IEC 22989:2022 [ISO/IEC 22989] zu KI-Konzepten und KI-Terminologie verabschiedet worden. Einige andere Normen beziehen sich nun bereits darauf. Wenn du mehr über den Weg einer Norm wissen möchtest, kannst du z. B. den Fachartikel *Wie entsteht eine Norm*[26] der Deutschen Gesellschaft für Qualität im Web lesen.

#### Die europäische Ebene

Auf europäischer Ebene gibt es ebenfalls Normungsgremien wie das *CEN* (franz. Comité Européen de Normalisation, engl. European Committee for Standardization) und *CENELEC* (franz. Comité Européen de Normalisation Électrotechnique, engl. European Committee for Electrotechnical Standardization), die in Abstimmung mit den internationalen Gremien an für KI-relevanten Standards und Dokumenten arbeiten. Beispielsweise arbeitet der Ausschuss *CEN/CENELEC JTC21 AI* an der europäischen Harmonisierung zu KI.

Die Europäische Union selbst hat bereits mehrere Verordnungen zu KI sowie zum Umgang mit Daten auf den Weg gebracht oder schon verabschiedet. Die Datenschutz-Grundverordnung[27] (DSGVO) [EU 2016] trat beispielsweise schon im Mai 2018 EU-weit in Kraft. Sie regelt und stärkt die Rechte natürlicher Personen zu ihren personenbezogenen Daten, beispielsweise zu deren Erhebung, Speicherung, Verarbeitung, der transparenten Information Betroffener oder Weiter-

26. *https://www.dgq.de/fachbeitraege/wie-entsteht-eine-iso-norm.*
27. *https://dsgvo-gesetz.de/art-12-dsgvo.*

gabe an Dritte. Aus Testperspektive ist für KI-basierte Systeme dabei besonders Folgendes von Bedeutung:

- Personenbezogene Daten müssen richtig bzw. präzise und auf dem neuesten Stand sein.[28] Dies gilt auch für Verarbeitungsergebnisse wie etwa Vorhersagen. Diese müssen nicht exakt, aber für den beabsichtigten Zweck genau genug sein.
- Entscheidungen dürfen im Grundsatz nicht ausschließlich automatisiert erfolgen und es dürfen keine potenziell diskriminierenden Merkmale wie z.B. ethnische Herkunft, politische Meinung, Religion u.Ä. dafür herangezogen werden.[29]
- Betroffene Personen haben ein Recht, transparent, präzise und leicht verständlich über die Verarbeitung ihrer Daten informiert zu werden.[30] Dies kann auch Aspekte der Nachvollziehbarkeit und Erklärbarkeit betreffen (siehe Abschnitt 2.7).

Bereits im April 2021 hat die Europäische Kommission einen Gesetzentwurf zur Regulierung von KI-Anwendungen, den *EU AI Act* [URL: Eur-lex-a] (Europäisches Gesetz über Künstliche Intelligenz), vorgelegt. Wesentlicher Bestandteil ist nach aktuellem Stand ein risikobasierter Ansatz. Je nach Risikoklasse sind darin QS-Maßnahmen und Pflichten vorgeschrieben, oder es ist ein sogenanntes *Konformitätsbewertungsverfahren* notwendig – was zu einer CE-Kennzeichnung des KI-basierten Systems führen soll. Der Einsatz besonders risikoreicher KI-Anwendungen wäre sogar untersagt. Der EU AI Act soll nach aktueller Planung 2023 in Kraft treten.

Neben dem EU AI Act gibt es eine Reihe weiterer europäischer Verordnungen, die bereits verabschiedet wurden (Cybersecurity Act, Data Governance Act, Digital Services Act) oder noch in Bearbeitung sind (Digital Markets Act, Data Act). Allerdings haben diese nur indirekten Bezug zu KI.

#### Aktivitäten auf nationaler Ebene

Auch auf nationaler Ebene findet man Normierungsaktivitäten im Kontext von KI. Für Deutschland arbeiten dafür das *Deutsche Institut für Normung* (DIN), der *Verband der Elektrotechnik Elektronik Informationstechnik* (VDE) und die *Deutsche Kommission Elektrotechnik Elektronik Informationstechnik* (DKE) eng zusammen.

Das DIN hat beispielsweise für Softwareentwicklungsprozesse von KI ein Qualitäts-Metamodell mit der Norm DIN SPEC 92001-1:2019-04 [DIN 2019] beschrieben. Im Auftrag der Bundesregierung wurde von DIN und DKE die *Deutsche Normungsroadmap KI* [Wahlster et al. 2022] erstmals im November 2020 herausgegeben. Zu dieser und der überarbeiteten Ausgabe vom Dezember 2022

28. Vgl. DSGVO Art. 5.
29. Vgl. DSGVO Art. 22 und Art. 9.
30. Vgl. DSGVO Art. 12.

durften wir neben vielen weiteren Experten aus den unterschiedlichsten Fachrichtungen selbst als Autoren beitragen.

In Deutschland beschäftigt sich zudem der *Gemeinschaftsausschuss KI* (DIN/DKE NA 043-01-42) damit, die nationalen Vorhaben mit denen auf europäischer und internationaler Ebene zu koordinieren (siehe Abb. 1–8).

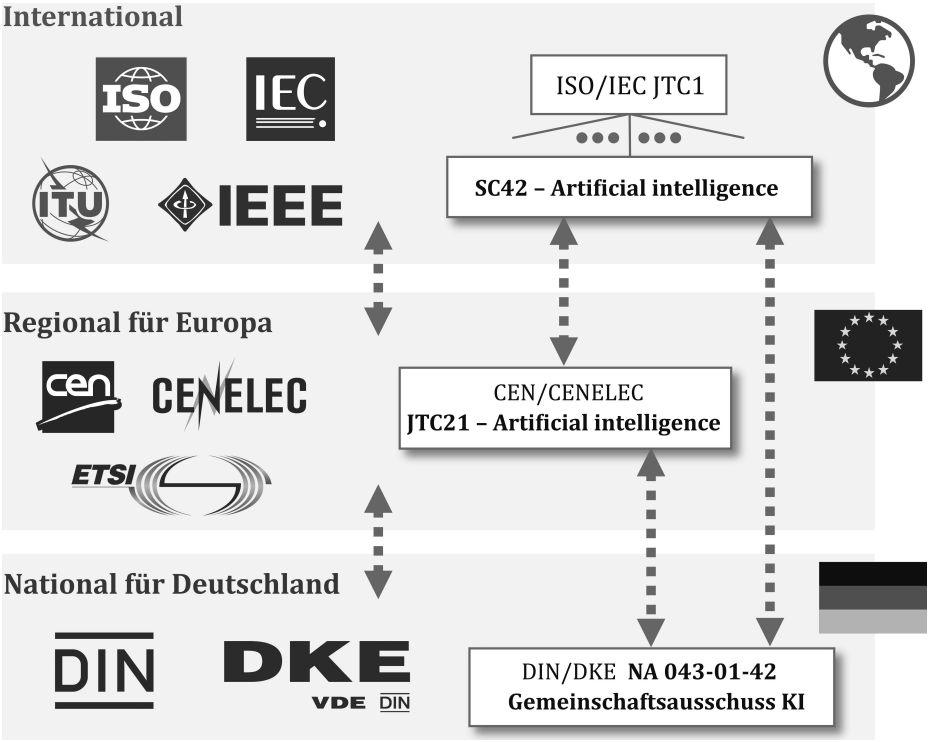


***Abb. 1–8*** *Übersicht zu den bekanntesten nationalen und internationalen Normungsgremien und deren Ausschüsse zu KI. Die gestrichelten Linien deuten deren übergreifende Zusammenarbeit an.*

#### Branchenverbände

Neben den fachübergreifenden Gremien sind auch branchenspezifische Verbände in der Normung zu KI-relevanten Themen aktiv. Beispielsweise in der Telekommunikation die *International Telecommunication Union* (ITU), das *European Telecommunication Standards Institute* (ETSI) und das *Institute of Electrical and Electronics Engineers* (IEEE).

Das IEEE hat etwa über seine internationale *Global Initiative on Ethics of Autonomous and Intelligent Systems* 2019 ein sehr umfangreiches Diskussionspapier über einen ethisch geführten Systementwurf veröffentlicht (Ethically Aligned Design) [IEEE 2019].

### 1.9.1 Exkurs: Liste einiger Normen und Standards mit KI-Bezug

Die folgende Liste (Stand Dezember 2023) nennt exemplarisch nur einige unserer Ansicht nach wichtige Normen im Kontext von KI. Sie ist keinesfalls vollständig und soll dir nur einen Einstieg in die Sicht der Normungsgremien auf KI ermöglichen.

| ISO/IEC | Titel |
|---|---|
| DIS 5259-2 | Artificial intelligence – Data quality for analytics and machine learning (ML) – Part 2: Data quality measures |
| 22989:2022 | Information technology – Artificial intelligence – Artificial intelligence concepts and terminology |
| 23053:2022 | Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML) |
| TR 24027:2021 | Information technology – Artificial intelligence (AI) – Bias in AI systems and AI aided decision making |
| TR 24028:2020 | Information technology – Artificial intelligence – Overview of trustworthiness in artificial intelligence |
| TR 24029-1:2021 | Artificial Intelligence (AI) – Assessment of the robustness of neural networks – Part 1: Overview |
| TR 24030:2021 | Information technology – Artificial intelligence (AI) – Use cases |
| 25059:2023 | Software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality model for AI systems |
| TR 29119-11:2020 | Software and systems engineering – Software testing – Part 11: Guidelines on the testing of AI-based systems |
| 42001:2023 | Information technology – Artificial intelligence – Management system |
| **ETSI** | |
| GR ENI 004 v2.2.1 | Experimental Networked Intelligence (ENI);<br>Terminology for Main Concepts in ENI |
| **DIN SPEC** | |
| 13266:2020-04 | Leitfaden für die Entwicklung von Deep-Learning-Bilderkennungssystemen |
| 92001-1:2019-04 | Artificial Intelligence – Life Cycle Processes and Quality Requirements – Part 1: Quality Meta Model |
| 92001-2:2020-12 | Artificial Intelligence – Life Cycle Processes and Quality Requirements – Part 2: Robustness |
| 92001-3:2023-08 | Artificial Intelligence – Life Cycle Processes and Quality Requirements – Part 3: Explainability |

# 7 Testen KI-basierter Systeme im Überblick

*»Vorbereitung ist vor allem der Schlüssel zum Erfolg.«*
Alexander Graham Bell, britischer Sprachtherapeut,
Erfinder und Großunternehmer

*In diesem Kapitel beschäftigen wir uns damit, wie wir KI-basierte Systeme testen und welchen Herausforderungen wir dabei als Testerinnen und Tester gegenüberstehen. Wir gehen dabei auf die Besonderheiten bei der Spezifikation oder Dokumentation solcher Systeme, die Verwendung von Testdaten und die Erweiterung der klassischen Teststufen ein. Genauer betrachtet werden die Effekte des Automatisierungsbias und des Konzeptdrifts. Abschließend widmen wir uns der Frage, mit welchen Risiken speziell ML-Systeme verbunden sind und wie wir ihnen begegnen können.*

**KI-spezifische Schlüsselbegriffe aus dem Lehrplan:**
KI-Komponente (AI component), Automatisierungsverzerrung (automation bias), Big Data (big data), Konzeptdrift (concept drift), Datenpipeline (data pipeline), funktionale Leistungsmetriken von ML (ML functional performance metrics), Trainingsdaten (training data)

**Weitere Schlüsselbegriffe in diesem Kapitel:**
Eingabedatentest (input data testing), Testen von ML-Modellen (ML model testing)

## 7.1 Spezifikation KI-basierter Systeme

Wenn Testerinnen und Tester in einem Projekt Aktivitäten planen, ist häufig eine der ersten Tätigkeiten, alle Informationsquellen, aus denen sie später Tests ableiten, also die *Testbasis*, zusammenzustellen. Schon in klassischen Softwareprojekten ist dies manchmal ziemlich herausfordernd: Nicht immer liegen eine saubere Systemspezifikation, Architekturdokumente oder Anforderungsbeschreibungen vor. Ist das der Fall, macht es die Testarbeit deutlich schwieriger – aber nicht unmöglich. Meist finden sich weitere Quellen, aus denen Wissen oder Erwartungen an das korrekte Verhalten des Systems gewonnen werden können: Altsysteme,

Konkurrenzprodukte, Skizzen oder Gespräche mit Mitgliedern des Entwicklungsteams oder Projektverantwortlichen sind nur einige Beispiele, die bei der Testanalyse herangezogen werden können.

Für das Testen sind eindeutige Systemspezifikationen auch für KI-basierte Systeme eine wertvolle Grundlage. Aus unserer Erfahrung liegen diese aber gerade für solche Systeme oder Komponenten noch seltener als bei konventionellen Systemen in einer eindeutigen Form vor.

Doch woran liegt es, dass die Spezifikation KI-basierter Systeme uns vor so große Herausforderungen stellt? Welche Ursachen erschweren es, eine klare Spezifikation oder ein Testorakel zu formulieren? Im Folgenden sollen diese Fragen aus verschiedenen KI-spezifischen Perspektiven beantwortet werden.

#### Neuartige Qualitätsmerkmale

Die Auswahl der wichtigsten Qualitätsmerkmale und der daraus abgeleiteten Anforderungen ist bereits in der klassischen Systemspezifikation eine der zentralen Aufgaben.

Es ist offensichtlich, dass daher auch die neuen Qualitätsmerkmale, wie etwa Anpassbarkeit, Flexibilität, Evolution oder Autonomie (siehe Kap. 2), in KI-Systemspezifikationen beschrieben und genau – zumindest im Kontext der Projektanforderungen – definiert werden müssen. Gerade wegen ihrer Neuartigkeit gibt es noch wenig Erfahrung oder gar etablierte Schemata, wie diese eindeutig, nachvollziehbar und letztlich messbar beschrieben werden können.

Wenn beispielsweise KI-Komponenten zur Verarbeitung von Eingaben durch Menschen verwendet werden, ist in vielen Fällen eine große Flexibilität gefordert. Diese Eingaben könnten über gesprochene Sprache oder Gestik wie etwa Gebärdensprache, durch Bildverarbeitung oder andere physische Sensoren erfolgen. In allen Varianten erwartet man, dass Sprache, Mimik oder Gestik nicht starr definiert sind, sondern sich von Mensch zu Mensch oder von Kontext zu Kontext unterscheiden, wie etwa Dialekte oder Gestik. Die Herausforderung liegt darin, diese Flexibilität bezüglich der verschiedenen Variationen von Umgebungen und Interaktionen zu identifizieren und möglichst umfangreich zu dokumentieren.

Abbildung 7–1 skizziert, warum es so schwierig ist, KI-Systeme oder auch deren Einsatzumgebung ausreichend zu spezifizieren (Bildmitte). Im unteren Bereich der Abbildung sind die Einflussfaktoren dargestellt, die sich aus den neuen Qualitätsmerkmalen ergeben. Die Abbildung enthält auch die noch im Folgenden aufgeführten Gründe für unzureichende Spezifikationen von KI-basierten Systemen (hellgraue Kästen: durch Pfeile verknüpft).

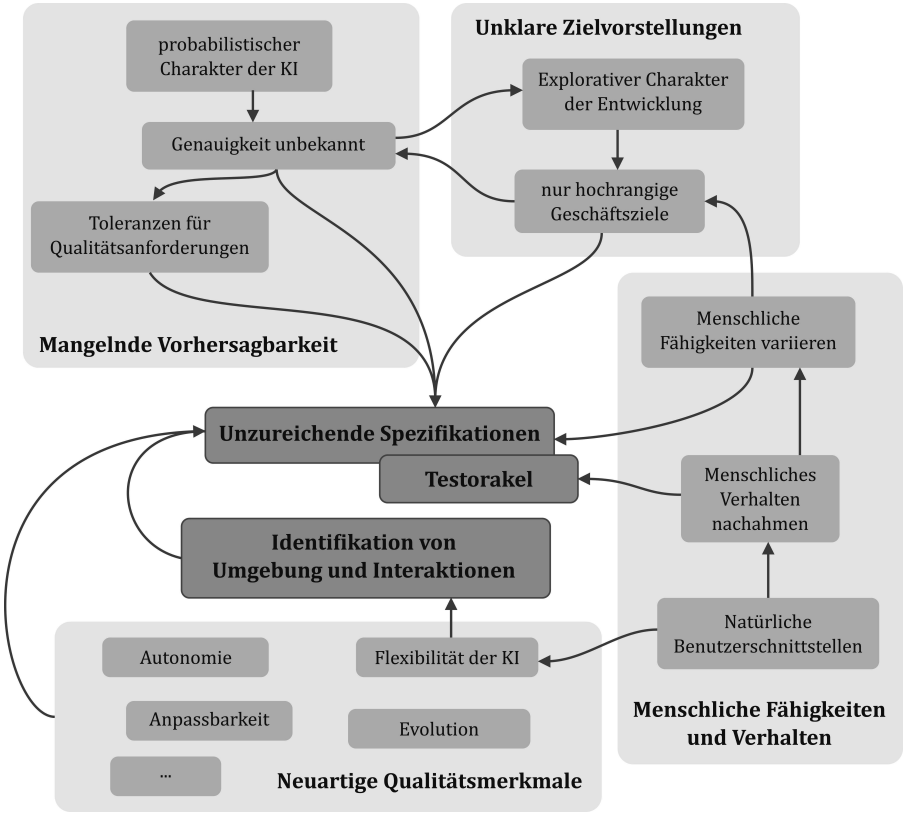


***Abb. 7–1*** *Die Gründe für unzureichende Spezifikationen von KI-basierten Systemen und die Identifikation ihrer Umgebung (Bildmitte) sind vielfältig und hier durch die umgebenden Kästen beispielhaft angedeutet.*

### Menschliche Fähigkeiten und Verhalten

Ein besonderer, wenn nicht gar der wichtigste Anwendungsbereich KI-basierter Systeme ist die Übernahme von Aufgaben, die bislang nur von Menschen ausgeführt werden konnten: beispielsweise das Lesen, Verstehen und Zusammenfassen von komplexen Texten, das Erkennen von potenziellem Krebsgewebe auf Röntgenbildern, das Zeichnen von Bildern oder das Komponieren von Musik. Doch was zeichnet gut komponierte Musik aus? Ist der Inhalt eines Textes treffend wiedergegeben? Wann ist ein generiertes Bild[1] gut genug? Oft begegnen uns dann Formulierungen wie: *»Die KI soll genauso gut wie oder sogar besser als Menschen sein.«* Erschwerend kommt hinzu, dass gerade die menschlichen Fähigkeiten, mit

1. Ein Artikel mit (positiven wie negativen) Beispielen generierender KIs aus der Designerszene findet sich unter: *https://katzlberger.ai/2022/05/04/virtuelle-models-und-ki-musik-fuer-austria-rwanda-fashion-connect*.

denen eine KI verglichen wird, stark variieren. So könntest du dir z.B. schon bei der vermeintlich klaren Aufgabenstellung »Das System soll Krebsgewebe auf einem Röntgenbild mindestens so gut wie ein Arzt erkennen« die Frage stellen, welcher Arzt gemeint ist: ein Facharzt, ein Allgemeinmediziner oder ein Onkologe mit langjähriger Berufserfahrung? Dies führt uns zum Testorakelproblem, auf das wir in Abschnitt 8.7 noch näher eingehen werden.

Die Antworten auf diese Fragen, mit denen man im Projektteam das System präziser spezifizieren will, lassen sich meist nur durch eine schrittweise Annäherung finden. Dies fordert auch den steten Vergleich mit den Fähigkeiten menschlicher Expertinnen und Experten im jeweiligen Kontext.

#### Mangelnde Vorhersagbarkeit des erwarteten Verhaltens

Viele KI-basierte Systeme verhalten sich probabilistisch und werden daher mit statistischen Maßen, wie z.B. funktionale Leistungsmetriken (siehe Kap. 5), bewertet. Mehr über die Herausforderungen beim Testen solcher Systeme kannst du in Abschnitt 8.4 lesen.

Auch wenn wir Menschen uns für die Realität perfekte KIs mit einer Genauigkeit von 100 % wünschen, sind diese meist nicht realisierbar. Die Herausforderung besteht nun darin, ausreichend hohe Akzeptanzkriterien, etwa eine Genauigkeit von 99,8 %, zu spezifizieren. Gleichzeitig sollen diese aber die Realisierbarkeit eines gut angepassten ML-Modells durch zu hohe Anforderungen nicht gefährden (siehe Abschnitt 3.5.1). Meist ist das nur möglich, wenn in einem explorativen Entwicklungsansatz – noch ohne oder nur mit vagen Spezifikationen – die Möglichkeiten des KI-basierten Systems iterativ erforscht werden. Erst nach und nach wird sich herausstellen, was erreichbar erscheint, und es können passende Akzeptanzkriterien festgelegt werden.

Aufgrund des probabilistischen Charakters der meisten KI-basierten Systeme können selbst statistische Maße, wie die oben genannte Genauigkeit, nie präzise ermittelt werden. Je nach Art und Umfang der zur Bestimmung verwendeten Testdaten lassen sich unterschiedliche Werte messen. Daher liegt es nahe, dass man auch bei solchen statistischen Werten Toleranzen zulassen oder Ober- und Untergrenzen festlegen muss. Wie eng oder wie nahe diese sein sollten, ist eine weitere Herausforderung bei der Spezifikation.

#### Unklare Zielvorstellungen

Die bislang geschilderten Gründe zeigen bereits, warum die Formulierung klarer Spezifikationen und Anforderungen für KI-basierte Systeme eine große Herausforderung ist. Aus dieser Problematik heraus bleiben in vielen Projekten auch die initial vorgegebenen Geschäftsziele eher auf abstraktem und wenig detailliertem Niveau. Ein Grund für solche wenig konkreten Geschäftsziele mag dem explorativen Charakter einer KI-Entwicklung geschuldet sein, der meist initial die technischen Möglichkeiten durch Versuch und Irrtum auslotet.

KI-Projekte beginnen daher häufig mit vorhandenen Datensätzen und dem allerersten Ziel, deren Potenzial für Vorhersagen zu verstehen. Klassische Softwareprojekte starten hingegen eher mit klaren Vorgaben und der geforderten Logik.

## 7.2 Teststufen für KI-basierte Systeme

Erfahrene Testerinnen und Tester kennen Teststufen bereits aus dem klassischen Softwaretest [ISTQB 2018]. In gut organisierten Testprojekten orientieren sich diese Stufen an den Entwicklungsschritten der Software. Typischerweise finden sich die Teststufen Komponententest, Integrationstest, Systemtest und Abnahmetest.[2] Je nach Art der Softwareentwicklung können weitere Teststufen hinzukommen – oder auch wegfallen.

KI-basierte Systeme enthalten meist neben KI-Komponenten auch viele klassische Softwarekomponenten. Es ist naheliegend, diese auch mit den klassischen Methoden des Komponententests zu prüfen [ISTQB 2018]. Abbildung 7–2 zeigt schematisch in zwei Testpyramiden sowohl die klassischen Teststufen (links) als auch eine neue Betrachtung von Teststufen für KI-basierte Systeme (rechts). Du kannst darin zwei neue Teststufen erkennen, den *Eingabedatentest* und den *ML-Modelltest* [Riccio et al. 2020]. Für diese beiden, aber auch für die danach folgenden Stufen spielen die enthaltenen KI-Komponenten eine wichtige Rolle: Sowohl Dateningenieure und Data Scientists als auch Menschen mit Fachexpertise in der Anwendungsdomäne sollten daher in allen Stufen, die KI-Komponenten enthalten, involviert sein.

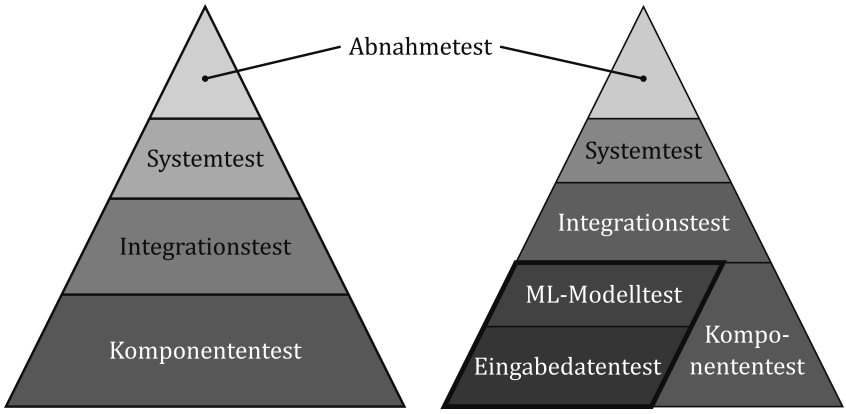


***Abb. 7–2*** *Typische Teststufen eines klassischen Softwaresystems (links) und eines KI-basierten Systems (rechts) als Testpyramide dargestellt.*

2. Hinweis: In der neuen Version 4.0 des CTFL-Lehrplans gibt es fünf Stufen (nicht relevant für die CT-AI-Prüfung): Komponententests, Komponentenintegrationstests, Systemtests, Systemintegrationstests, Abnahmetests.

*Übrigens:* In diesem Buch beziehen wir uns häufig auf ML-Systeme, also KI-basierte Systeme, die durch Methoden des maschinellen Lernens (siehe Abschnitt 1.4) entstehen. Die meisten der im Folgenden genauer beschriebenen Teststufen können aber durchaus auf alle KI-basierten Systeme angewendet werden.

### 7.2.1 Eingabedatentest

Für ML-Systeme sind qualitativ hochwertige Daten von besonderer Bedeutung. In Abschnitt 4.3 haben wir uns eingehend damit beschäftigt, welche Probleme bei Datensätzen vorliegen können, und im folgenden Abschnitt 4.4, welche Auswirkungen diese auf ML-Modelle haben können. Daher ist es das Ziel des Eingabedatentests, sicherzustellen, dass die Datensätze für das Training, die Evaluierung, den Test und den Betrieb des Systems eine möglichst hohe Qualität haben.

Die Daten durchlaufen während der Datenvorbereitung eine ganze Reihe von Verarbeitungsschritten (siehe Abschnitt 4.1), die Data Scientists meist mit einer explorativen Datenanalyse begleiten. Fast immer kommen dabei Werkzeuge zum Einsatz, die diese Schritte automatisieren. Daher liegt es nahe, zu prüfen, ob die Verarbeitungsschritte geeignet und korrekt sind und in der richtigen Reihenfolge verwendet werden. Man führt also im klassischen Sinne Komponenten- und Integrationstests der Datenpipeline durch.

Für den Betrieb des ML-Systems ist häufig eine vergleichbare Pipeline von Verarbeitungsschritten notwendig, möglicherweise sogar dieselbe (siehe Abb. 7–3). Es kann sein, dass sich diese *Betriebsdatenpipeline* auch stark von der Datenvorverarbeitung unterscheidet. In allen Fällen ist es wichtig, dass die Gleichwertigkeit der Betriebsdatenpipeline mit der Datenvorbereitung überprüft wird, selbst wenn die Tests für beide Ketten sehr unterschiedlich aussehen. Beispielsweise könnte die Datenvorbereitung in der Programmiersprache Python geschrieben sein und direkt mit einer Datenbank interagieren, während die Betriebsdatenpipeline aus Gründen der Performanz in der Programmiersprache C++ geschrieben ist und mit Echtzeitdaten eines hardwarenahen Betriebssystems arbeitet. Für einen Vergleich beider Pipelines könnte man die Datenvorbereitung als Pseudo-Orakel heranziehen und mit der Betriebsdatenpipeline testen, wobei leichte Abweichungen toleriert werden können.

Typische Testarten können hierbei sein:

- **Reviews**, um etwa die Vollständigkeit der Verarbeitungsschritte zu überprüfen oder die Auswahl der Merkmale, die auf einen Stichprobenbias getestet wurden.
- **Statistische Techniken**, um die Eingabedaten selbst zu untersuchen, z. B. auf einen Stichprobenbias oder Ausreißer.
- Die **EDA** selbst, um die signifikanten Trends oder Probleme in den Daten zu finden und die wesentlichen Merkmale zu extrahieren und zu visualisieren.

- **Statische und dynamische Tests** der Betriebsdatenpipeline, aber auch der Datenvorbereitung, um deren Eignung und Gleichwertigkeit sicherzustellen.

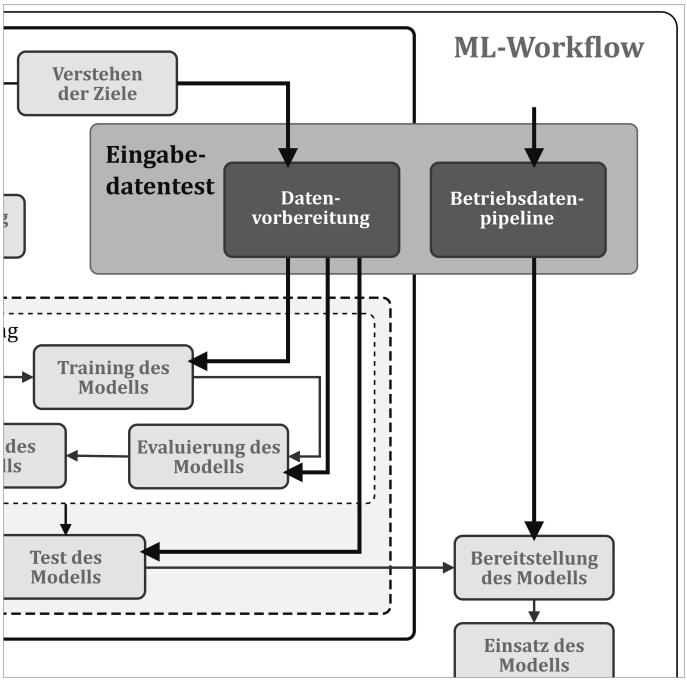


***Abb. 7–3*** *Die Teststufe des Eingabedatentests als Ausschnitt des ML-Workflows zielt sowohl auf die (Eingabe-)Daten und deren Vorbereitung als auch auf die Verarbeitung der Eingabedaten durch die Betriebsdatenpipeline ab (siehe Abb. 3–9).*

### 7.2.2 ML-Modelltest

Einen Teil des ML-Modelltests haben wir bereits bei der Beschreibung des ML-Workflows in Abschnitt 3.2 als Aktivität *Test des Modells* kennengelernt. Ziel dieser Teststufe ist aber auch die Absicherung der Aktivitäten, die das Training vorbereiten: Auswahl des ML-Frameworks und Auswahl/Umsetzung des Algorithmus (siehe Abb. 7–4).

Wenn man diese Vorbedingungen für das Training betrachtet, läuft dies normalerweise auf Reviews hinaus. Dabei soll sichergestellt werden, dass die zu Beginn des ML-Workflows festgelegten Ziele, Prioritäten und Akzeptanzkriterien (siehe Abschnitt 3.2) beispielsweise bei den getroffenen Framework-, Design- und Implementierungsentscheidungen berücksichtigt wurden. Beispielsweise werden die folgenden Design- und Implementierungsentscheidungen überprüft:

- **ML-Framework:** Ist es geeignet, die erforderlichen Datenmengen zu verarbeiten, und erlaubt es eine Skalierung der Rechenleistung bei Engpässen?

- **Algorithmus:** Ist der gewählte Algorithmus, beispielsweise ein neuronales Netz, für die zu implementierende Aufgabe geeignet?
- **Modell:** Wenn ein Modell mit einer bestimmten Architektur oder ein vortrainiertes Modell verwendet wird, ist es ein etabliertes Modell, das in vergleichbaren Anwendungen erfolgreich eingesetzt wurde? Liegt seine aktuelle Version[3] vor?
- **Modelleinstellungen:** Ist die Größe des neuronalen Netzes groß genug, um alle erforderlichen Klassifikationen treffen zu können?
- **Hyperparameter:** Wurden das Training und die Anzahl der Epochen so parametrisiert, dass ein optimales Trainingsergebnis erreicht wird (kein Underfit), und wurde darauf geachtet, dass ein Overfitting erkannt wird (kein Overfit)?

Hast du das trainierte Modell vorliegen, kannst du es durch dynamische Tests auf die festgelegten funktionalen Leistungskriterien prüfen. Dies können Werte wie die Genauigkeit, Präzision oder der F1-Wert sein. Auch andere Leistungsmetriken wie AUC oder MSE können hier gefordert sein (siehe die Abschnitte 5.1 und 5.2).

Zudem gibt es eine Reihe weiterer, meist nicht funktionaler Akzeptanzkriterien, die du ebenfalls betrachten kannst. Diese umfassen manchmal sogar Eigenschaften, die den Trainingsprozess selbst betreffen, z. B.:

- **Trainingsgeschwindigkeit:** Wie schnell konnte das Modell mit den aktuellen Trainingsdaten trainiert werden? Ist das Training damit schnell genug, in beispielsweise wöchentlichen Zyklen mit neuen Trainingsdaten aktualisiert zu werden?
- **Vorhersagegeschwindigkeit:** Wie schnell kann das Sprachmodell auf einer definierten Einsatzumgebung eine Antwort geben?
- **Genutzte Rechnerressourcen:** Wie viel GPU-Speicher, CPU-Speicher, Rechenleistung in TOPS[4] oder Datentransferraten über eine Netzwerkverbindung werden benötigt?
- **Anpassbarkeit:** Wie einfach kann das ML-Modell auf eine neue Aufgabe angepasst werden?
- **Transparenz:** Wie viel und welche Informationen sind über das trainierte ML-Modell verfügbar, und für welche Nutzerinnen und Nutzer sind diese zugänglich?

Ebenso können **Überdeckungskriterien** für neuronale Netze (siehe Abschnitt 6.2) helfen, das Vertrauen in das ML-Modell zu erhöhen: Welche Neuronenüberdeckung wurde bei einem für die funktionale Sicherheit relevanten neuronalen Netz mit den verwendeten Testdaten erreicht?

3. Bereitgestellte Modelle, wie beispielsweise das YOLO-Netzwerk zur Bildverarbeitung (*https://github.com/ultralytics/ultralytics*), werden von deren Anbietern häufig aktualisiert.
4. TOPS = Billionen Rechenoperationen pro Sekunde (engl. tera operations per second).

**Praxistipp**

Aus unserer Erfahrung wissen wir, dass gerade die nicht funktionalen Akzeptanzkriterien in einigen Projekten wenig Beachtung finden. Meist werden die oben angedeuteten Fragen erst spät im Projekt gestellt. Dann kann es sein, dass große Aufwände entstehen, um z.B. die Vorhersagegeschwindigkeit genau zu messen oder die geeigneten Werkzeuge zur Bestimmung des Überdeckungsgrads neuronaler Netze zu integrieren. In Abschnitt 7.1 haben wir zwar beschrieben, dass diese nicht funktionalen Akzeptanzkriterien möglicherweise erst im Laufe des Projekts genauer festgelegt werden können, trotzdem ist es entscheidend, sie so früh wie möglich zu berücksichtigen.

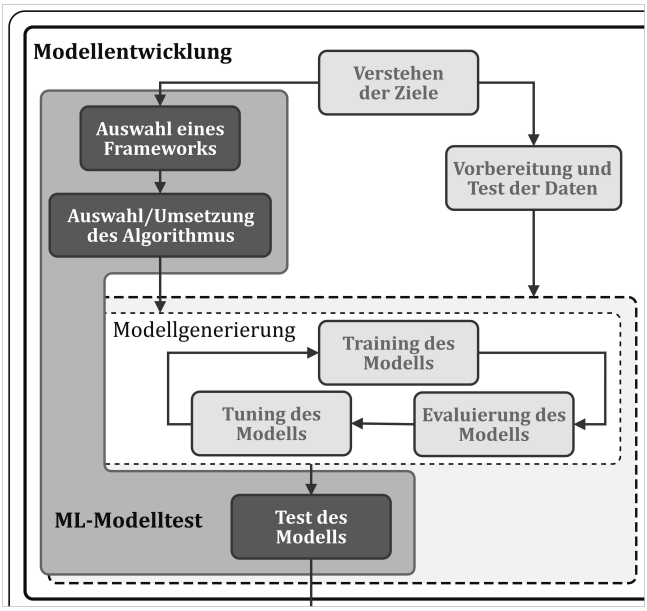


***Abb. 7–4*** *Die Teststufe ML-Modelltest bewertet das generierte ML-Modell. Sie entspricht dem Komponententest der klassischen Softwareentwicklung, bezogen auf die erzeugte KI-Komponente.*

Wir fassen die Testziele in der Teststufe ML-Modelltest im Folgenden kurz zusammen:

- Das ML-Modell erfüllt die zur Abnahme geforderten funktionalen Leistungskriterien von ML.
- Das ML-Modell erfüllt die nicht funktionalen Abnahmekriterien.
- Das Design und die Implementierung des ML-Modells bzw. die getroffene Wahl von ML-Framework, Algorithmus und Hyperparametern ist möglichst optimal, d. h., die verbleibenden Risiken wurden minimiert.

Dazu nutzt man die folgenden Testarten:

- Statische Tests z.B. Reviews von ML-Framework, Algorithmus, Modell, Modellparameter, Hyperparameter
- Dynamische Tests des ML-Modells, vorzugsweise in einer isolierten Testumgebung

### 7.2.3 Komponententest

Der *Komponententest* wird ausführlich im CTFL-Lehrplan [ISTQB 2018] behandelt. Er konzentriert sich auf alle konventionellen Softwarekomponenten, also diejenigen ohne KI. Diese Komponenten zeichnet aus, dass sie für sich allein testbar sind – ggf. in einer für sie zugeschnittenen Testumgebung.

Typische Beispiele könnten hier Softwarekomponenten einer Datenverarbeitungskette sein, Komponenten zur Kommunikation mit anderen Services oder Benutzungsschnittstellen wie z.B. grafische Benutzeroberflächen (engl. graphical user interface, GUI). Abbildung 7–5 zeigt schematisch ein System, das aus einer Verarbeitungskette mehrerer Komponenten besteht. Für die konventionellen Softwarekomponenten (Nicht-KI) sind Komponententests vorgesehen. Für die KI-Komponente (das ML-Modell) ist der ML-Modelltest anzuwenden.

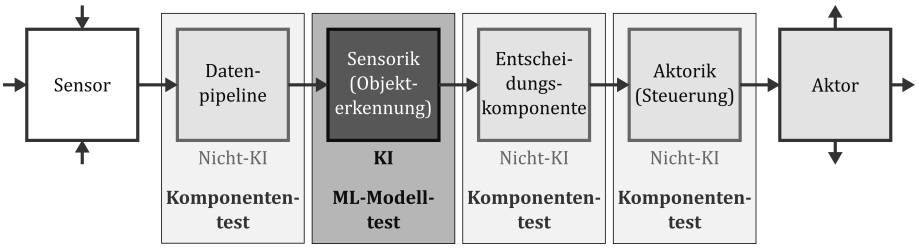


***Abb. 7–5*** *Beispiel eines Systems mit konventionellen (Nicht-KI) und KI-Komponenten und den für sie vorgesehenen Teststufen. Das gezeigte Schema lehnt sich an die Darstellung autonomer Systeme aus der ISO/IEC TR 29119-11 [ISO/IEC TR 29119-11] an.*

### 7.2.4 Komponentenintegrationstest

Ebenso wie der Komponententest ist der *Komponentenintegrationstest* eine konventionelle Teststufe, die im CTFL-Lehrplan beschrieben wird. Man konzentriert sich dabei auf die Schnittstellen zwischen den integrierten Komponenten und darauf, dass diese wie erwartet zusammenwirken.

Bei Betrachtung von Abbildung 7–6 kannst du dir vorstellen, dass einerseits die beiden Nicht-KI-Komponenten (Entscheidungskomponente und Aktorik) integriert werden und man sich auf deren gemeinsame Schnittstelle konzentriert, ganz wie im konventionellen Test.

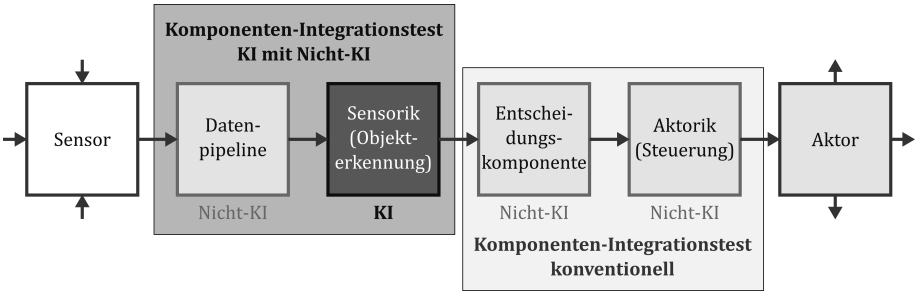


***Abb. 7–6*** *Beispiele für den Zuschnitt von Komponentenintegrationstests, aufbauend auf den Komponententests aus Abbildung 7–5.*

Betrachtet man andererseits die Integration einer KI- mit einer Nicht-KI-Komponente (Datenpipeline und Sensorik oder Sensorik und Entscheidungskomponente), so ist die Situation sehr ähnlich. Hier prüfst du zum einen, ob die Ausgaben der Datenpipeline korrekt als Eingabedaten in der Sensorik empfangen und verarbeitet werden. Zum anderen prüfst du, ob die Ausgaben der Sensorik (Objekterkennung) korrekt von der Entscheidungskomponente empfangen und richtig interpretiert werden. Ein Beispiel wäre, ob die Koordinaten der erkannten Objekte im korrekten Koordinatensystem angegeben sind (in Pixeln, $x$- und $y$-Achsen, Orientierung der Achsen). Wenn eine KI-Komponente als Dienst implementiert ist (AI as a Service (AIaaS), siehe Abschnitt 1.7), werden die Tests der API (engl. application programming interface) ebenfalls im Rahmen der Komponentenintegrationstests durchgeführt. Diese Teststufe unterscheidet sich bei KI-basierten Systemen vom konventionellen Integrationstest darin, dass hier die Testumgebung und die Testdaten teilweise schwieriger zu erstellen sind und die Testergebnisse wegen des probabilistischen Charakters der KI möglicherweise anders zu bewerten sind.

### 7.2.5 Systemtest

Sind in einem Projekt schrittweise alle Komponenten eines Systems integriert, kannst du dieses im *Systemtest* in seiner Gesamtheit prüfen (siehe Abb. 7–7). In dieser konventionellen Teststufe geht es darum, funktionale wie nicht funktionale Eigenschaften so nahe wie möglich an der eigentlichen Einsatzumgebung zu testen. Du verifizierst dabei nicht nur Akzeptanzkriterien, sondern validierst beispielsweise auch, ob das System aus Sicht des Menschen für den angedachten Einsatz geeignet ist.

Neben Feldversuchen eignen sich auch realistische Laboraufbauten, um möglichst nahe an der Zielumgebung zu sein. Für manche Systeme, etwa eine Fahrfunktion eines autonomen Autos, könnte der Aufbau einer realen Situation viel zu gefährlich sein. Solche Szenarien, aber auch Situationen, die schwer nachzustellen sind, werden dann über eine simulierte Testumgebung realisiert (siehe auch Abschnitt 10.2).

# Inhaltsübersicht

# Inhaltsverzeichnis

9 783864 909474

Nils Röttger · Gerhard Runze · Verena Dietrich

# Basiswissen KI-Testen

Umfragen in der Industrie zeigen deutlich: KI-Projekte scheitern häufiger als angenommen. Eine kontinuierliche Qualitätssicherung für KI-basierte Systeme ist daher unabdingbar.

Das Autorenteam bietet einen fundierten Überblick und einen praxisnahen Einstieg in die Konzepte, Best Practices, Problemstellungen und Lösungsansätze rund um die Qualitätssicherung von und mit KI-basierten Systemen. Im Einzelnen werden behandelt:

- Einführung in KI
- Qualitätsmerkmale KI-basierter Systeme
- Maschinelles Lernen (ML)
- ML-Daten
- Funktionale Leistungsmetriken
- Neuronale Netze und Testen
- Testen KI-basierter Systeme
- Testen KI-spezifischer Qualitätsmerkmale
- Methoden und Verfahren für das Testen KI-basierter Systeme
- Testumgebungen für KI-basierte Systeme
- Einsatz von KI beim Testen

Das Buch enthält mehrere Exkurse, z.B. »ChatGPT als Teammitglied?«, Praxisbeispiele und zu vielen Kapiteln auch praktische Übungen, wobei die Lerninhalte durch Codebeispiele und Programmierübungen in Python veranschaulicht werden. Die Aufgaben und Lösungen sind als Jupyter Notebooks auf GitHub verfügbar.

Das Buch orientiert sich am ISTQB®-Syllabus »Certified Tester AI Testing« (CT-AI) und eignet sich daher nicht nur bestens zur Prüfungsvorbereitung, sondern dient gleichzeitig als kompaktes Grundlagenwerk zu diesen Themen in der Praxis und an Hochschulen.

GitHub-Repository zum Buch:
*https://github.com/KI-Testen/Uebungen*

Thema
- **Softwaretest**
- **Qualitätssicherung**
- **Künstliche Intelligenz**
- **Softwareentwicklung**
- **Programmierung**

Richtet sich an
- **Tester*innen**
- **Testmanager*innen**
- **Softwareentwickler*innen**
- **Qualitätsverantwortliche in KI-Projekten**
- **Lehrende und Studierende**

€ 34,90 (D)
ISBN 978-3-86490-947-4