Lehrplan
Certified Tester
AI Testing (CT-AI)

Version 2021 1.0D

International Software Testing Qualifications Board

Deutschsprachige Ausgabe
Herausgegeben durch German Testing Board e. V.
in Zusammenarbeit mit dem Austrian Testing Board
und dem Swiss Testing Board

Der englischsprachige Lehrplan wurde zur Verfügung gestellt von

Alliance for Qualification, Artificial Intelligence United,
Chinese Software Testing Qualifications Board und
Korean Software Testing Qualifications Board

A4Q

AiU Artificial Intelligence United

Chinese Software Testing Qualifications Board CSTQB

KSTQB Korean Software Testing Qualifications Board

Copyright-Hinweis

Übersetzung des englischsprachigen Lehrplans des International Software Testing Qualifications Board (ISTQB®), Originaltitel: Certified Tester AI Testing (CT-AI) Syllabus, Version V1.0 2021.

Änderungsübersicht Originalausgabe

Version	Datum	Bemerkungen
1.0	2021/10/01	Freigabe für GA

Änderungsübersicht deutschsprachige Ausgabe

Version	Datum	Bemerkungen
1.0D	2022/08/12	Übersetzungung durch GTB e.V. mit ATB und STB

Inhaltsverzeichnis

Danksagung

Der englischsprachige Lehrplan wurde von der Generalversammlung des ISTQB® am 1. Oktober 2021 formell freigegeben.

Er wurde von einem Team des International Software Testing Qualifications Board erstellt: Klaudia Dussa-Zieger (Vorsitz), Werner Henschelchen, Vipul Kocher, Qin Liu, Stuart Reid, Kyle Siemens und Adam Leon Smith.

Das Team dankt den Autoren der drei beigetragenen Lehrpläne;

- A4Q: Rex Black, Bruno Legeard, Jeremias Rößler, Adam Leon Smith, Stephan Goericke, Werner Henschelchen
- AiU: Hauptautoren Vipul Kocher, Saurabh Bansal, Srinivas Padmanabhuni und Sonika Bengani sowie die Co-Autoren Rik Marselis, José M. Diaz Delgado
- CSTQB/KSTQB: Qin Liu, Stuart Reid

Das Team dankt den Arbeitsgruppen "Exam", "Glossary" und "Marketing" für ihre Unterstützung bei der Entwicklung des Lehrplans, Graham Bath für seine technische Bearbeitung und den Mitglieds-Boards für ihre Anregungen und Beiträge.

Die folgenden Personen haben an den Reviews und der Kommentierung dieses Lehrplans mitgewirkt: Laura Albert, Reto Armuzzi, Árpád Beszédes, Armin Born, Géza Bujdosó, Renzo Cerquozzi, Sudeep Chatterjee, Seunghee Choi, Young-jae Choi, Piet de Roo, Myriam Christener, Jean-Baptiste Crouigneau, Guofu Ding,Erwin Engelsma, Hongfei Fan, Péter Földházi Jr., Tamás Gergely, Ferdinand Gramsamer, Attila Gyúri, Matthias Hamburg, Tobias Horn, Jarosław Hryszko, Beata Karpinska, Joan Killeen, Rik Kochuyt, Thomas Letzkus, Chunhui Li, Haiying Liu, Gary Mogyorodi, Rik Marselis, Imre Mészáros, Tetsu Nagata, Ingvar Nordström, Gábor Péterffy, Tal Pe'er, Ralph Pichler, Nishan Portoyan, Meile Posthuma, Adam Roman, Gerhard Runze, Andrew Rutz, Klaus Skafte, Mike Smith, Payal Sobti, Péter Sótér, Michael Stahl, Chris van Bael, Stephanie van Dijck, Robert Werkhoven, Paul Weymouth, Dong Xin, Ester Zabar, Claude Zhang.

Die deutschsprachige Fassung wurde erstellt durch: Dr. Klaudia Dussa-Zieger, Thorsten Geiselhart, Prof. Dr. Ralf Reißing, Dr. Stephan Weißleder und Prof. Dr. Mario Winter (Leitung).

Das Team dankt den Arbeitsgruppen "Prüfung" und "Glossar", den Reviewern sowie dem Vorstand des GTB e.V. für ihre Unterstützung bei der Lokalisierung dieses Dokuments.

0 Einführung

0.1 Zweck dieses Syllabus

Dieser Lehrplan bildet die Grundlage für den ISTQB® Certified Tester AI Testing. Das ISTQB® stellt diesen Lehrplan wie folgt zur Verfügung:

1. Nationalen Mitglieds-Boards, die den Lehrplan in ihre nationale(n) Sprache(n) übersetzen und Seminaranbieter akkreditieren dürfen. Nationale Mitglieds-Boards dürfen den Lehrplan an die Anforderungen ihrer nationale(n) Sprache(n) anpassen und Referenzen auf lokale Veröffentlichungen hinzufügen.
2. Zertifizierungsstellen zur Ableitung von Prüfungsfragen in ihre nationale(n) Sprache(n), die an die Lernziele dieses Lehrplans angepasst sind.
3. Seminaranbietern zur Erstellung von Kursmaterialien und zur Bestimmung angemessener Lehrmethoden.
4. Zertifizierungskandidaten zur Vorbereitung auf die Zertifizierungsprüfung (entweder als Teil eines Seminars oder unabhängig davon).
5. Der internationalen Software- und Systementwicklungs-Community zur Förderung der Weiterbildung und des Berufsbildes des Software- und Systemtesters und als Grundlage für Bücher und Fachartikel.

0.2 Der Certified Tester AI Testing

Der Certified Tester AI Testing richtet sich an alle in das Thema Softwaretesten involvierten Personen, die ihr Wissen über das Testen KI-basierter Systeme und/oder die Nutzung von KI zum Testen vertiefen wollen, bzw. an Personen, die sich in ihrer beruflichen Laufbahn auf diese Themen spezialisieren wollen. Dazu gehören Personen in Funktionen wie Tester, Testanalysten, Datenanalysten, Testingenieure, Testberater, Testmanager, Usability-Tester und Softwareentwickler. Diese Zertifizierung eignet sich auch für alle, die ein grundlegendes Verständnis für das Testen KI-basierter Systeme und/oder KI für das Testen erlangen möchten, wie z. B. Projektmanager, Qualitätsmanager, Softwareentwicklungsmanager, Business-Analysten, Mitglieder von Operations-Teams, IT-Direktoren und Unternehmensberater.

Das Dokument *Certified Tester AI Testing (CT-KI) Überblick über den Lehrplan* [I03] ist ein separates Dokument, das die folgenden Informationen enthält:

- Geschäftlicher Nutzen des Lehrplans.
- Verfolgbarkeitsmatrix zwischen geschäftlichem Nutzen und Lernzielen.
- Zusammenfassung dieses Lehrplans.
- Beziehungen zwischen den Lehrplänen.

0.3 Prüfbare Lernziele und kognitive Stufen des Wissens

Die Lernziele unterstützen den geschäftlichen Nutzen und werden bei der Erstellung der Certified Tester AI Testing-Prüfungen verwendet.

Die Kandidaten können aufgefordert werden, ein Schlüsselwort oder ein Konzept, das in einem der elf Kapitel erwähnt wird, zu erkennen, sich daran zu erinnern oder abzurufen. Die spezifischen Lernziele sind zu Beginn jedes Kapitels aufgeführt und wie folgt klassifiziert:

- K1: Erinnern
- K2: Verstehen
- K3: Anwenden
- K4: Analysieren

Alle Begriffe, die als Schlüsselwörter direkt unter den Kapitelüberschriften aufgeführt sind, müssen auswendig gelernt werden (K1), auch wenn sie nicht ausdrücklich in den Lernzielen erwähnt werden.

0.4 Praktische Kompetenzebenen

Der Certified Tester AI Testing umfasst praxisnahe Lernziele, die sich auf praxisnahe Fähigkeiten und Kompetenzen konzentrieren.

Die folgenden Stufen gelten für praxisnahe Ziele (wie in der Liste unten gezeigt):

- H0: Live-Demo einer Übung oder aufgezeichnetes Video.
- H1: Angeleitete Übung, die Lernenden folgen einer Abfolge von Schritten, die der Trainer vorgibt.
- H2: Übung mit Hinweisen, die Lernenden erhalten eine Aufgabe mit entsprechenden Hinweisen, so dass die Aufgabe innerhalb des vorgegebenen Zeitrahmens gelöst werden kann, oder sie nehmen an einer Diskussion teil.

Die Kompetenzen werden durch praktische Übungen erworben, wie sie in der folgenden Liste aufgeführt sind:

- (HO-3.5.1 H0) Demonstrieren von Überanpassung und Unteranpassung.
- (HO-4.1.1 H2) Durchführen der Datenvorbereitung zur Unterstützung der Erstellung eines ML-Modells.
- (HO-4.2.1 H2) Identifizieren von Trainings- und Testdatensätzen und Erstellen eines ML-Modells.
- (HO-5.4.1 H2) Evaluieren des erstellten ML-Modells anhand ausgewählter funktionaler Leistungsmetriken von ML.
- (HO-6.1.1 H1) Beobachten der Implementierung eines Perzeptrons.
- (HO-8.6.1 H2) Verwenden eines Werkzeugs um zu zeigen, wie Erklärbarkeit von Testern genutzt werden kann.
- (HO-9.2.1 H2) Anwenden von paarweisem Testen für Entwurf und Ausführung von Testfällen für ein KI-basiertes System.
- (HO-9.5.1 H2) Anwenden von metamorphem Testen, um Testfälle für ein bestimmtes Szenario zu entwerfen und auszuführen.
- (HO-9.6.1 H2) Anwenden von explorativem Testen auf ein KI-basiertes System.
- (HO-11.1.1 H2) Erläutern von Tätigkeiten im Testbereich, bei denen der Einsatz von KI weniger wahrscheinlich ist, an Hand von Beispielen.
- (HO-11.5.1 H2) Implementieren eines einfachen KI-basierten Fehlerprognosesystems.

0.5 Die Prüfung zum Certified Tester AI Testing

Die Prüfung zum Certified Tester AI Testing stützt sich auf diesen Lehrplan. Für die Beantwortung der Prüfungsfragen kann es erforderlich sein, Material zu verwenden, das auf mehr als einem Abschnitt dieses Lehrplans basiert. Alle Abschnitte des Lehrplans sind prüfungsrelevant, mit Ausnahme dieser Einführung und der Anhänge. Normen und Bücher sind als Referenz aufgeführt, aber ihr Inhalt ist nicht

prüfungsrelevant, abgesehen von dem, was im Lehrplan selbst aus diesen Normen und Büchern zusammengefasst ist.

Hinweis zur Zulassungsvoraussetzung: Das ISTQB® Certified Tester Foundation Level Zertifikat muss vor der Prüfung zum Certified Tester AI Testing erworben werden.

0.6 Akkreditierung

Ein nationales ISTQB-Mitglieds-Board kann Seminaranbieter akkreditieren, deren Kursmaterial diesem Lehrplan entspricht. Seminaranbieter sollten die Akkreditierungsrichtlinien von dem Mitglieds-Board (in Deutschland: German Testing Board e.V.; in der Schweiz: Swiss Testing Board; in Österreich: Austrian Testing Board) oder bei der Organisation beziehen, welche die Akkreditierung durchführt. Ein akkreditierter Kurs wird als konform zu diesem Lehrplan anerkannt und darf eine ISTQB®-Prüfung als Teil des Kurses haben.

Die Akkreditierungsrichtlinien für diesen Lehrplan folgen den allgemeinen Akkreditierungsrichtlinien, die von der Arbeitsgruppe Prozessmanagement und Compliance veröffentlicht wurden.

0.7 Detaillierungsgrad

Der Detaillierungsgrad dieses Lehrplans ermöglicht international einheitliche Kurse und Prüfungen. Um dieses Ziel zu erreichen, umfasst der Lehrplan:

- Allgemeine Lehrziele, welche die Intention des Certified Tester AI Testing beschreiben.
- Eine Liste von Begriffen, die sich die Lernenden merken können müssen.
- Lern- und Praxisziele für jeden Wissensbereich, welche die zu erreichenden Lernergebnisse beschreiben.
- Eine Beschreibung der Schlüsselkonzepte, einschließlich Verweisen auf Quellen wie anerkannte Literatur oder Normen.

Der Inhalt des Lehrplans ist keine Beschreibung des gesamten Wissensbereichs für das Testen KI-basierter Systeme; er spiegelt den Detaillierungsgrad wider, der in den Schulungen zum Certified Tester Specialist AI Testing behandelt wird. Er konzentriert sich auf die Einführung in die grundlegenden Konzepte der künstlichen Intelligenz (KI) und des maschinellen Lernens im Besonderen und darauf, wie Systeme, die auf diesen Techniken basieren, getestet werden können.

0.8 Aufbau des Lehrplans

Es gibt elf Kapitel mit prüfbarem Inhalt. Die oberste Überschrift jedes Kapitels gibt die Zeit für das jeweilige Kapitel an; unterhalb der Kapitelebene wird keine Zeitangabe gemacht. Für akkreditierte Ausbildungskurse erfordert der Lehrplan mindestens 25,1 Unterrichtsstunden, die sich wie folgt auf die elf Kapitel verteilen:

Kapitel 1	Einführung in KI	105 Minuten
Kapitel 2	Qualitätsmerkmale für KI-basierte Systeme	105 Minuten
Kapitel 3	Maschinelles Lernen (ML) - Überblick	145 Minuten
Kapitel 4	ML - Daten	230 Minuten
Kapitel 5	Funktionale Leistungsmetriken von ML	120 Minuten
Kapitel 6	ML - Neuronale Netzwerke und Tests	65 Minuten
Kapitel 7	Testen KI-basierter Systeme im Überblick	115 Minuten
Kapitel 8	Testen KI-spezifischer Qualitätsmerkmale	150 Minuten
Kapitel 9	Methoden und Verfahren für das Testen KI-basierter Systeme	245 Minuten
Kapitel 10	Testumgebungen für KI-basierte Systeme	30 Minuten
Kapitel 11	Einsatz von KI für Tests	195 Minuten
		1505 Minuten

1 Einführung in KI - 105 Minuten

Schlüsselbegriffe

Keine

KI-spezifische Schlüsselwörter

KI-als-Dienst (*AI as a Service*, AIaaS), KI-Entwicklungs-Framework, KI-Effekt, KI-basiertes System, künstliche Intelligenz (KI), neuronales Netzwerk, Deep Learning (DL), tiefes neuronales Netzwerk, allgemeine KI, Datenschutz-Grundverordnung (DSGVO), maschinelles Lernen (ML), schwache KI, vortrainiertes Modell, Super-KI, technologische Singularität, Transferlernen

Lernziele für Kapitel 1:

1.1 Definition von KI und KI-Effekt

AI[1]-1.1.1 K2 Beschreiben des KI-Effekts und wie er die Definition von KI beeinflusst

1.2 Schwache, Allgemeine und Super-KI

AI-1.2.1 K2 Unterscheiden zwischen schwacher KI, allgemeiner KI und Super-KI

1.3 KI-basierte und konventionelle Systeme

AI-1.3.1 K2 Unterscheiden zwischen KI-basierten Systemen und herkömmlichen Systemen

1.4 KI-Techniken

AI-1.4.1 K1 Erkennen verschiedener Techniken zur Implementierung von KI

1.5 KI-Entwicklungs-Frameworks

AI-1.5.1 K1 Identifizieren gängiger KI-Entwicklungs-Frameworks

1.6 Hardware für KI-basierte Systeme

AI-1.6.1 K2 Vergleichen der verfügbaren Hardwareoptionen für die Implementierung KI-basierter Systeme

1.7 KI-als-Dienst (AIaaS)

AI-1.7.1 K2 Erklären des Konzepts von KI-als-Dienst (AIaaS)

1.8 Vortrainierte Modelle

AI-1.8.1 K2 Erläutern der Verwendung vortrainierter KI-Modelle und damit verbundener Risiken

1.9 Normen, Vorschriften und AI

AI-1.9.1 K2 Beschreiben, inwiefern Normen für KI-basierte Systeme gelten

[1] Zur besseren Referenzierbarkeit des englischsprachigen Lehrplans werden die Lernziele auch im deutschsprachigen Lehrplan mit AI-x.y.z nummeriert.

1.1 Definition von KI und KI-Effekt

Der Begriff künstliche Intelligenz (KI) geht auf die 1950er Jahre zurück und bezieht sich auf das Ziel, "intelligente" Maschinen zu bauen und zu programmieren, die in der Lage sind, den Menschen zu imitieren. Die heutige Definition hat sich erheblich weiterentwickelt, und die folgende Definition fasst das Konzept zusammen [S01]:

> *Die Fähigkeit eines technischen Systems, Wissen und Fertigkeiten zu erwerben, zu verarbeiten und anzuwenden.*

Die Art und Weise, wie die Menschen die Bedeutung von KI verstehen, hängt von ihrer aktuellen Wahrnehmung ab. In den 1970er Jahren lag die Vorstellung eines Computersystems, das einen Menschen im Schach schlagen konnte, noch in der Zukunft, und die meisten hielten dies für KI. Heute, mehr als zwanzig Jahre nachdem das computergestützte System Deep Blue den Schachweltmeister Garri Kasparow besiegt hat, wird der in diesem System implementierte "Brute-Force"-Ansatz von vielen nicht als echte künstliche Intelligenz angesehen (d. h. das System lernte nicht aus Daten und war nicht in der Lage, selbst zu lernen). Ähnlich verhielt es sich mit den Expertensystemen der 1970er und 1980er Jahre, die menschliches Fachwissen in Form von Regeln enthielten, die wiederholt ausgeführt werden konnten, ohne dass der Experte anwesend war. Diese wurden damals als KI betrachtet, gelten aber heute nicht mehr als solche.

Die sich ändernde Wahrnehmung dessen, was KI ausmacht, ist als "KI-Effekt" bekannt [R01]. In dem Maße, wie sich die Wahrnehmung von KI in der Gesellschaft ändert, ändert sich auch ihre Definition. Infolgedessen wird sich jede Definition von heute in der Zukunft wahrscheinlich ändern und nicht mehr mit der aus der Vergangenheit übereinstimmen.

1.2 Schwache KI, Allgemeine KI und Super-KI

Auf einer abstrakten Ebene kann man KI in drei Kategorien unterteilen:

- Systeme der schwachen KI können eine bestimmte Aufgabe mit begrenztem Kontext ausführen. Diese Form der KI ist derzeit weit verbreitet. Zum Beispiel Systeme, die Spiele spielen, Spam-Filter, Testfallgeneratoren und Sprachassistenten.
- Systeme der allgemeinen KI (auch bekannt als starke KI) haben allgemeine (weitreichende) kognitive Fähigkeiten, die denen des Menschen ähneln. Diese KI-basierten Systeme können wie Menschen denken und ihre Umwelt verstehen sowie entsprechend handeln. Bis zum Jahr 2021 gibt es noch keine allgemeinen KI-Systeme.
- Systeme der Super-KI sind in der Lage, die menschliche Wahrnehmung nachzubilden (allgemeine KI) und verfügen über eine enorme Verarbeitungsleistung, einen praktisch unbegrenzten Speicher und Zugang zu allem menschlichen Wissen (z. B. durch Zugang zum Internet). Man geht davon aus, dass Super-KI-Systeme nach kurzer Zeit klüger sein werden als Menschen. Der Punkt, an dem KI-basierte Systeme von allgemeiner KI zu Super-KI übergehen, wird häufig als technologische Singularität bezeichnet [B01].

1.3 KI-basierte und konventionelle Systeme

In einem typischen konventionellen Computersystem wird die Funktion der Software von Menschen in z. B. einer imperativen Programmiersprache programmiert, die Konstrukte wie if-then-else und Schleifen enthält. Für Menschen ist es so relativ einfach nachzuvollziehen, wie das System Eingaben in Ausgaben umwandelt. In einem KI-basierten System, das maschinelles Lernen (ML) einsetzt, verwendet das System Muster in Daten, um zu bestimmen, wie es in Zukunft auf neue Daten reagieren soll (siehe Kapitel 3 für eine ausführliche Erläuterung von ML). So wird beispielsweise ein KI-basierter Bildprozessor, der Bilder von Katzen erkennen soll, mit einer Reihe von Bildern trainiert, von denen

bekannt ist, dass sie Katzen enthalten. Die KI bestimmt selbständig, welche Muster oder Merkmale in den Daten zur Identifizierung von Katzen verwendet werden können. Diese Muster und Regeln werden dann auf neue Bilder angewandt, um festzustellen, ob sie Katzen enthalten. Bei vielen KI-basierten Systemen führt dies dazu, dass das Vorhersage-Verfahren für den Menschen weniger leicht nachvollziehbar ist (siehe Abschnitt 2.8).

In der Praxis können KI-basierte Systeme durch eine Vielzahl von Techniken implementiert werden (siehe Abschnitt 1.4), und der "KI-Effekt" (siehe Abschnitt 1.1) kann beeinflussen, was aktuell als KI-basiertes System und was als konventionelles System gilt.

1.4 KI-Techniken

KI kann mit einer breiten Palette von Techniken implementiert werden (siehe [B02] für weitere Einzelheiten), wie z. B.:

- Fuzzy-Logik
- Suchalgorithmen
- Argumentationsverfahren
  - Regelmaschinen
  - Deduktive Klassifikatoren
  - Fallbezogene Argumentation (auch fallbasiertes Schließen, *case-based reasoning*)
  - Prozedurale Argumentation
- Verfahren des maschinellen Lernens
  - Neuronale Netze
  - Bayessche Modelle
  - Entscheidungsbäume
  - Random Forest
  - Lineare Regression
  - Logistische Regression
  - Clusterbildungsalgorithmen
  - Genetische Algorithmen
  - Stützvektormaschinen (*support vector machine*, SVM)

KI-basierte Systeme setzen in der Regel eines oder mehrere dieser Verfahren ein.

1.5 KI-Entwicklungs-Frameworks

Es gibt viele KI-Entwicklungs-Frameworks, von denen einige auf bestimmte Bereiche spezialisiert sind. Diese Frameworks unterstützen eine Reihe von Aktivitäten wie die Datenvorbereitung, die Auswahl von Algorithmen und die Kompilierung von Modellen für die Ausführung auf verschiedenen Prozessoren wie zentrale Recheneinheiten (CPUs), Grafikprozessoren (*graphic processing unit*, GPUs) oder Tensorprozessoren (*tensor processing unit*, TPUs). Die Auswahl eines bestimmten Frameworks kann auch von bestimmten Aspekten wie der für die Implementierung verwendeten Programmiersprache und ihrer Benutzerfreundlichkeit abhängen. Die folgenden Frameworks gehören zu den beliebtesten (Stand: 2021):

- Apache MxNet: Ein Open-Source-Framework für Deep Learning, das von Amazon für Amazon Web Services (AWS) verwendet wird [R02].

- CNTK: Das Microsoft Cognitive Toolkit (CNTK) ist ein Open-Source-Toolkit für Deep Learning [R03].
- IBM Watson Studio: Eine Tool-Suite, welche die Entwicklung von KI-Lösungen unterstützt [R04].
- Keras: Eine Open-Source-API auf hoher (Abstraktions-)Ebene, die in Python geschrieben ist und auf TensorFlow und CNTK aufsetzen kann [R06].
- PyTorch: Eine Open-Source-ML-Bibliothek, die von Facebook betrieben wird und für Anwendungen zur Bildverarbeitung und Verarbeitung natürlicher Sprache (NLP) verwendet wird. Es werden sowohl Python- als auch C++-Schnittstellen unterstützt [R07].
- Scikit-learn: Eine Open-Source-Bibliothek für maschinelles Lernen für die Programmiersprache Python [R08].
- TensorFlow: Ein Open-Source-ML-Framework auf der Grundlage von Datenflussgraphen für skalierbares maschinelles Lernen, bereitgestellt von Google [R05].

Zu beachten ist, dass diese KI-Entwicklungs-Frameworks ständig weiterentwickelt, miteinander kombiniert und durch neue Frameworks ersetzt werden.

1.6 Hardware für KI-basierte Systeme

Für das Training von ML-Modellen (siehe Kapitel 3) und die Modellimplementierung wird eine Vielzahl von Hardware verwendet. Ein Modell zur Spracherkennung kann beispielsweise auf einem einfachen Smartphone ausgeführt werden, obwohl zum Trainieren des Modells der Zugriff auf die Leistung von Cloud-Computing erforderlich ist. Ein gängiger Ansatz für den Fall, dass das Zielsystem nicht mit dem Internet verbunden ist, besteht darin, das Modell in der Cloud zu trainieren und es dann auf dem Zielsystem auszuführen.

ML profitiert in der Regel von Hardware, welche die folgenden Eigenschaften unterstützt:

- Arithmetik mit geringer Genauigkeit: Hier werden weniger Bits für die Berechnung verwendet (z. B. 8 statt 32 Bits, was in der Regel für ML ausreicht).
- Die Fähigkeit, mit großen Datenstrukturen zu arbeiten (z. B. zur Unterstützung von Matrixmultiplikationen).
- Massiv parallele (nebenläufige) Verarbeitung.

Universelle CPUs bieten Unterstützung für komplexe Operationen, die für ML-Anwendungen in der Regel nicht erforderlich sind, und verfügen nur über einige wenige Kerne. Daher ist ihre Architektur für das Training und die Ausführung von ML-Modellen weniger effizient als die von GPUs, die Tausende von Kernen haben und für die massiv parallele, aber relativ einfache Verarbeitung von Bildern ausgelegt sind. Infolgedessen sind GPUs bei ML-Anwendungen in der Regel leistungsfähiger als CPUs, obwohl CPUs in der Regel höher getaktet sind. Für ML in kleinem Maßstab sind GPUs im Allgemeinen die beste Option.

Es gibt Hardware, die speziell für KI entwickelt wurde, z. B. anwendungsspezifische integrierte Schaltungen (ASICs) und System on a Chip (SoC). Diese KI-spezifischen Lösungen verfügen über Merkmale wie mehrere Kerne, eine spezielle Datenverwaltung und die Fähigkeit zur speicherinternen Verarbeitung. Sie eignen sich am besten für Edge Computing zur prozessnahen Datenerfassung und -verarbeitung [R11], während das Training des ML-Modells in der Cloud erfolgt.

Hardware mit speziellen KI-Architekturen befindet sich derzeit (Stand 2021) in der Entwicklung. Dazu gehören neuromorphe Prozessoren [B03], die nicht die traditionelle von-Neumann-Architektur verwenden, sondern eine Architektur, welche die Neuronen des Gehirns näherungsweise imitiert.

Beispiele für Anbieter von KI-Hardware und deren Prozessoren sind (Stand: 2021):

- NVIDIA: Das Unternehmen bietet eine Reihe von Grafikprozessoren und KI-spezifischen Prozessoren an, wie z. B. Volta [R09].
- Google: Das Unternehmen hat anwendungsspezifische integrierte Schaltkreise sowohl für das Training als auch für das logische Schlussfolgern (Inferenz) entwickelt. Google TPUs [R10] können von Nutzern in der Google Cloud genutzt werden, während die Edge TPU [R11] ein speziell entwickelter ASIC ist, der für die Ausführung von KI auf einzelnen Geräten entwickelt wurde.
- Intel: Das Unternehmen bietet Nervana-Prozessoren für neuronale Netze [R12] für Deep Learning (sowohl Training als auch Inferenz) und Movidius Myriad-Bildverarbeitungseinheiten für Inferenzen in Anwendungen für Computer Vision und neuronale Netze.
- Mobileye: Das Unternehmen stellt die EyeQ-Familie als SoC [R13] her, die komplexe und rechenintensive Bildverarbeitung unterstützen. Diese haben einen niedrigen Stromverbrauch und sind für den Einsatz in Fahrzeugen gedacht.
- Apple: Das Unternehmen stellt den Bionic-Chip für die lokale KI in iPhones her [B04].
- Huawei: Ihr Kirin 970-Chip für Smartphones verfügt über die integrierte Verarbeitung neuronaler Netze für KI-Anwendungen [B05].

1.7 KI-als-Dienst (AIaaS)

KI-Komponenten, z B. ML-Modelle, können innerhalb einer Organisation erstellt, von einem Drittanbieter heruntergeladen oder als Dienst im Internet (*AI-as-a-Service*, AIaaS) genutzt werden. Auch ein hybrider Ansatz ist möglich, bei dem ein Teil der KI-Funktionalität innerhalb des Systems und ein Teil als Dienst bereitgestellt wird.

Bei der Nutzung von ML als Dienst wird der Zugang zu einem ML-Modell über das Internet bereitgestellt. Außerdem kann auch Unterstützung bei der Datenvorbereitung und -speicherung, dem Modelltraining, der Evaluierung, dem Tuning, dem Testen und der Bereitstellung geleistet werden.

Drittanbieter (z. B. AWS, Microsoft) bieten spezielle KI-Dienste wie Gesichts- und Spracherkennung an. Dadurch können Einzelpersonen und Unternehmen KI mithilfe von Cloud-basierten Diensten implementieren, selbst wenn sie nicht über ausreichende Ressourcen und Fachkenntnisse verfügen, um eigene KI-Dienste zu entwickeln. Darüber hinaus wurden ML-Modelle, die als Teil eines Drittanbieter-Dienste bereitgestellt werden, wahrscheinlich auf einem größeren und vielfältigeren Trainingsdatensatz trainiert, als er vielen Akteuren zur Verfügung steht, z. B. solchen, die erst seit kurzem auf dem KI-Markt tätig sind.

1.7.1 Verträge für KI-als-Dienst

KI-Dienste werden in der Regel mit ähnlichen Verträgen wie für nicht KI-basierte Software als Dienst (*Software-as-a-Service*, SaaS) bereitgestellt. Ein Vertrag für AIaaS umfasst in der Regel eine Dienstleistungs-Güte-Vereinbarung (*Service Level Agreement,* SLA), in der Verfügbarkeits- und IT-Sicherheitsverpflichtungen festgelegt sind. Solche SLAs umfassen in der Regel eine Verfügbarkeit für den Dienst (z. B. 99,99 % Verfügbarkeit) und eine Reaktionszeit für die Behebung von Fehlerzuständen, definieren aber selten in ähnlicher Weise funktionale Leistungsmetriken von ML (wie z. B. Genauigkeit) (siehe Kapitel 5). AIaaS wird häufig auf Basis eines Abonnements bezahlt, und wenn die vertraglich vereinbarte Verfügbarkeit und/oder Reaktionszeit nicht eingehalten wird, bietet der Dienst-Anbieter in der Regel Gutschriften für künftige Dienst-Nutzungen an. Abgesehen von diesen Gutschriften sehen die meisten AIaaS-Verträge eine begrenzte Haftung vor (abgesehen von den gezahlten Gebühren), was bedeutet, dass KI-basierte Systeme, die von AIaaS abhängen, in der Regel auf Anwendungen mit relativ geringem Risiko beschränkt sind, bei denen ein Dienst-Ausfall keinen allzu großen Schaden anrichten würde.

Die Dienste werden häufig mit einer anfänglichen kostenlosen Testphase anstelle einer Abnahmefrist angeboten. Während dieses Zeitraums soll der Nutzer des AIaaS testen, ob der angebotene Dienst seinen Anforderungen in Bezug auf die erforderliche Funktionalität und Leistung (z. B. Genauigkeit) entspricht. Dies ist in der Regel erforderlich, um die fehlende Transparenz des angebotenen Dienstes auszugleichen (siehe Abschnitte 2.8 und 7.5).

1.7.2 Beispiele für KI-als-Dienst

Im Folgenden sind Beispiele für AIaaS aufgeführt (Stand: 2021):

- IBM Watson Assistant: Dies ist ein KI-Chatbot, dessen Preis sich nach der Anzahl der monatlich aktiven Nutzer richtet [R28].
- Google Cloud KI und ML Produkte: Diese bieten dokumentenbasierte KI, die einen Formularparser und Dokumenten-OCR umfasst. Die Preise richten sich nach der Anzahl der zu verarbeitenden Seiten [R29].
- Amazon CodeGuru: Dies bietet eine statische Analyse von ML-Java-Code, die Entwicklern Empfehlungen zur Verbesserung der Codequalität liefert. Die Preise richten sich nach der Anzahl der analysierten Quellcodezeilen [R30].
- Microsoft Azure Cognitive Search: Bietet KI-Suche in der Cloud. Die Preise basieren auf Sucheinheiten (definiert durch den verwendeten Speicher und Durchsatz) [R31].

1.8 Vortrainierte Modelle

1.8.1 Einführung in vortrainierte Modelle

Das Trainieren von ML-Modellen kann teuer sein (siehe Kapitel 3). Zunächst müssen die Daten vorbereitet werden, und dann muss das Modell trainiert werden. Die erste Aktivität kann große Mengen an menschlichen Ressourcen benötigen, während die zweite Aktivität eine Menge an Computerressourcen benötigen kann. Viele Organisationen haben keinen Zugang zu diesen Ressourcen.

Eine kostengünstigere und oft auch effektivere Alternative ist die Verwendung eines vortrainierten Modells. Dieses bietet eine ähnliche Funktionalität wie das benötigte Modell und wird als Grundlage für die Erstellung eines neuen Modells verwendet, das die Funktionalität des vortrainierten Modells erweitert und/oder fokussiert. Solche Modelle sind nur für eine begrenzte Anzahl von Techniken verfügbar, z. B. für neuronale Netze und Random Forests.

Wenn ein Bildklassifikator benötigt wird, könnte dieser anhand des öffentlich zugänglichen ImageNet-Datensatzes trainiert werden, der über 14 Millionen Bilder enthält, die in über 1000 Kategorien klassifiziert sind [R14]. Dies verringert das Risiko, erhebliche Ressourcen zu verbrauchen, ohne dass eine Erfolgsgarantie besteht. Alternativ könnte ein bestehendes Modell wiederverwendet werden, das bereits auf diesem Datensatz trainiert wurde. Durch die Verwendung eines solchen vortrainierten Modells werden Trainingskosten gespart und das Risiko, dass es nicht funktioniert, weitgehend eliminiert.

Wird ein vortrainiertes Modell ohne Änderungen verwendet, kann es einfach in das KI-basierte System eingebettet oder als Dienst genutzt werden (siehe Abschnitt 1.7).

1.8.2 Transferlernen

Es ist auch möglich, ein bereits trainiertes Modell so zu verändern, dass es eine weitere Aufgabe erfüllt. Dies ist als Transferlernen bekannt und wird bei tiefen neuronalen Netzen verwendet, bei denen die

oberen Schichten (siehe Kapitel 6) des neuronalen Netzes in der Regel recht einfache Aufgaben erfüllen (z. B. die Unterscheidung zwischen geraden und gekrümmten Linien in einem Bildklassifikator), während die tieferen Schichten speziellere Aufgaben erfüllen (z. B. die Unterscheidung zwischen Typen von Gebäudearchitekturen). In diesem Beispiel können alle Schichten eines Bildklassifizierers bis auf die späteren Schichten wiederverwendet werden, so dass die frühen Schichten nicht trainiert werden müssen. Die späteren Schichten werden dann neu trainiert, um die speziellen Anforderungen für einen neuen Klassifikator zu erfüllen. In der Praxis kann das vortrainierte Modell durch zusätzliches Training mit neuen problemspezifischen Daten feinabgestimmt werden.

Die Effektiviät dieses Ansatzes hängt weitgehend ab von der Ähnlichkeit zwischen der Funktion, die das ursprüngliche Modell erfüllt, und der Funktion, die das neue Modell benötigt. Ein Bildklassifikator für Katzenarten wäre beispielsweise weitaus leichter als Bildklassifikator für Hunderassen anzupassen wie als Klassifikator für sprachliche Akzente von Menschen.

Es sind viele vortrainierte Modelle verfügbar, insbesondere von Wissenschaftlern. Einige Beispiele sind ImageNet-Modelle [R14] wie Inception, VGG, AlexNet und MobileNet für die Klassifikation von Bildern und vortrainierte NLP-Modelle wie BERT von Google [R15].

1.8.3 Risiken bei der Verwendung von vortrainierten Modellen und Transferlernen

Die Verwendung von vortrainierten Modellen und das Transferlernen sind gängige Ansätze für den Aufbau KI-basierter Systeme, die jedoch mit einigen Risiken verbunden sind. Dazu gehören:

- Einem vorab trainierten Modell kann es im Vergleich zu einem selbsterstellten Modell an Transparenz fehlen.
- Die Ähnlichkeit zwischen der Funktion, die das vortrainierte Modell ausführt, und der erforderlichen Funktionalität ist möglicherweise nicht ausreichend. Außerdem werden die Unterschiede von Datenwissenschaftlern möglicherweise nicht verstanden.
- Unterschiede in den Datenvorbereitungsschritten (siehe Abschnitt 4.1), die bei der ursprünglichen Entwicklung des vortrainierten Modells verwendet wurden, und den Datenvorbereitungsschritten bei der Verwendung dieses Modells in einem neuen System können sich auf die resultierende funktionale Leistung auswirken.
- Die Unzulänglichkeiten eines vortrainierten Modells werden wahrscheinlich von den Modellen übernommen, die darauf basieren, und sind möglicherweise nicht dokumentiert. So können beispielsweise übernommene Verzerrungen (siehe Abschnitt 2.4) nicht offensichtlich sein, wenn die Dokumentation über die zum Trainieren des Modells verwendeten Daten unzureichend ist. Wenn das trainierte Modell nicht weit verbreitet ist, gibt es wahrscheinlich mehr unbekannte (oder nicht dokumentierte) Fehler, und es können gründlichere Tests erforderlich sein, um dieses Risiko zu mindern.
- Modelle, die durch Transferlernen erstellt werden, sind höchstwahrscheinlich anfällig für dieselben Schwachstellen wie das zuvor trainierte Modell, auf dem sie basieren (z. B. gegnerische Angriffe, wie in Abschnitt 9.1.1 erläutert). Wenn darüber hinaus bekannt ist, dass ein KI-gestütztes System ein bestimmtes vortrainiertes Modell enthält (oder auf einem bestimmten vortrainierten Modell basiert), dann können die damit verbundenen Schwachstellen potenziellen Angreifern bereits bekannt sein.

Zu beachten ist, dass mehrere der oben genannten Risiken durch eine gründliche Dokumentation des vortrainierten Modells (siehe Abschnitt 7.5) abgemildert werden können.

1.9 Normen, Vorschriften und KI

Das Joint Technical Committee of IEC and ISO on information technology (ISO/IEC JTC1) erarbeitet internationale Normen mit Bezug zur KI. So wurde zum Beispiel 2017 ein Unterausschuss für KI (ISO/IEC JTC 1/SC42) eingerichtet. Darüber hinaus hat ISO/IEC JTC1/SC7, das sich mit Software- und Systemtechnik befasst, einen technischen Bericht über das Testen KI-basierter Systeme [S01] veröffentlicht.

Normen zur künstlichen Intelligenz werden auch auf regionaler Ebene (z. B. europäische Normen) und auf nationaler Ebene veröffentlicht.

Die EU-weite Datenschutz-Grundverordnung (DSGVO) trat im Mai 2018 in Kraft und legt Verpflichtungen für die Verantwortlichen für die Datenverarbeitung in Bezug auf personenbezogene Daten und automatisierte Entscheidungsfindung fest [B06]. Die DSGVO enthält Anforderungen zur Bewertung und Verbesserung der funktionalen Leistung von KI-Systemen, einschließlich der Minderung potenzieller Diskriminierung, und zur Gewährleistung der Rechte von Einzelpersonen, keiner automatisierten Entscheidungsfindung unterworfen zu werden. Der wichtigste Aspekt der DSGVO aus der Testperspektive ist, dass personenbezogene Daten (einschließlich Vorhersagen) genau sein sollten. Das bedeutet nicht, dass jede einzelne Vorhersage des Systems richtig sein muss, sondern dass das System für die Zwecke, für die es verwendet wird, genau genug sein sollte.[2]

Das Deutsche Institut für Normung (DIN) hat auch das KI-Qualitäts-Metamodell entwickelt ([S02], [S03]).

Normen für KI werden auch von Branchenverbänden veröffentlicht. So arbeitet beispielsweise das Institute of Electrical and Electronics Engineers (IEEE) an einer Reihe von Normen zu Ethik und KI (The IEEE Global Initiative for Ethical Considerations in Artificial Intelligence and Autonomous Systems). Viele dieser Normen befinden sich zum Zeitpunkt der Erstellung dieses Lehrplans noch in der Entwicklung.

Wenn KI in sicherheitsrelevanten Systemen eingesetzt wird, gelten die entsprechenden Normen, z. B. ISO 26262 [S04] und ISO/PAS 21448 (SOTIF) [S05] für Automobilsysteme. Solche Normen werden in der Regel von staatlichen Stellen vorgeschrieben, und in einigen Ländern wäre es illegal, ein Auto zu verkaufen, wenn die enthaltene Software nicht der ISO 26262 entspräche. Normen an sich sind freiwillig zu befolgende Dokumente, und ihre Anwendung wird normalerweise nur durch Gesetze oder Verträge vorgeschrieben. Viele Anwender von Normen tun dies dennoch, um vom Fachwissen der Autoren zu profitieren und Produkte von höherer Qualität zu schaffen.

[2] Die EU Kommission hat 2021 einen Vorschlag für eine Verordnung des Europäischen Parlaments und des Rates zur Festlegung harmonisierter Vorschriften für Künstliche Intelligenz (Gesetz über Künstliche Intelligenz, AI Act) vorgelegt [R32]. Diese soll 2023 Inkrafttreten. Der EU AI Act basiert auf einem risikobasierten Ansatz und stellt je nach Risikostufe der KI verschiedene Qualitäts- und Testanforderungen.

2 Qualitätsmerkmale für KI-basierte Systeme - 105 Minuten

Schlüsselwörter

Keine

KI-spezifische Schlüsselwörter

Anpassbarkeit, algorithmische Verzerrung, Autonomie, Verzerrung (*bias*), Evolution, Erklärbarkeit, erklärbare KI (*eXplainable AI*, XAI), Flexibilität, (unangemessene) Verzerrung (*bias*), Interpretierbarkeit, ML-System, maschinelles Lernen, Belohnungs-Hacking (*reward hacking*), Robustheit, Stichprobenverzerrung, selbstlernendes System, Nebenwirkungen, Transparenz

Lernziele für Kapitel 2:

2.1 Flexibilität und Anpassbarkeit

AI-2.1.1 K2 Erläutern der Wichtigkeit von Flexibilität und Anpassbarkeit als Merkmale KI-basierter Systeme

2.2 Autonomie

AI-2.2.1 K2 Erläutern der Beziehung zwischen Autonomie und KI-basierten Systemen

2.3 Evolution

AI-2.3.1 K2 Erläutern der Wichtigkeit des Managements der Evolution KI-basierter Systeme

2.4 Verzerrung

AI-2.4.1 K2 Beschreiben der verschiedenen Ursachen und Arten von Verzerrungen, die in KI-basierten Systemen auftreten

2.5 Ethik

AI-2.5.1 K2 Diskutieren der ethischen Grundsätze, die bei der Entwicklung, dem Einsatz und der Nutzung KI-basierter Systeme beachtet werden sollten

2.6 Nebenwirkungen und Belohnungs-Hacking

AI-2.6.1 K2 Erläutern von Nebenwirkungen und Belohnungs-Hacking, die in KI-basierten Systemen auftreten

2.7 Transparenz, Interpretierbarkeit und Erklärbarkeit

AI-2.7.1 K2 Erklären, wie Transparenz, Interpretierbarkeit und Erklärbarkeit auf KI-basierte Systeme anwendbar sind

2.8 Funktionale Sicherheit und KI

AI-2.8.1 K1 Erinnern der Eigenschaften, die den Einsatz KI-basierter Systeme in sicherheitsrelevanten Anwendungen erschweren

2.1 Flexibilität und Anpassbarkeit

Flexibilität und Anpassbarkeit sind eng miteinander verbundene Qualitätsmerkmale. In diesem Lehrplan wird Flexibilität als die Fähigkeit des Systems betrachtet, in Situationen eingesetzt zu werden, die nicht Teil der ursprünglichen Systemanforderungen waren, während Anpassbarkeit als die Leichtigkeit betrachtet wird, mit der das System an neue Situationen, wie z. B. unterschiedliche Hardware und sich ändernde Einsatzumgebungen, angepasst werden kann.

Sowohl Flexibilität als auch Anpassbarkeit sind nützlich, wenn:

- die Einsatzumgebung zum Zeitpunkt der Einführung des Systems nicht vollständig bekannt ist;
- das System mit neuen Einsatzumgebungen zurechtkommen soll;
- das System sich an neue Situationen anpassen soll;
- das System bestimmen muss, wann es sein Verhalten ändern soll.

Von selbstlernenden KI-basierten Systemen wird erwartet, dass sie alle oben genannten Eigenschaften aufweisen. Folglich müssen sie anpassbar sein und das Potenzial haben, flexibel zu sein.

Die Anforderungen an die Flexibilität und Anpassbarkeit eines KI-basierten Systems sollten Einzelheiten zu allen Änderungen der Einsatzumgebung enthalten, an die sich das System anpassen soll. Diese Anforderungen sollten auch Beschränkungen für die Zeit und die Ressourcen enthalten, die das System verwenden kann, um sich anzupassen (z. B. wie viel Zeit es benötigen darf, um sich an die Erkennung eines neuen Objekttyps anzupassen).

2.2 Autonomie

Bei der Definition von Autonomie ist es zunächst wichtig zu erkennen, dass ein vollständig autonomes System völlig unabhängig von menschlicher Aufsicht und Kontrolle wäre. In der Praxis wird eine vollständige Autonomie oft nicht angestrebt. So werden z. B. vollständig selbstfahrende Autos, die im Volksmund als "autonom" bezeichnet werden, offiziell als "voll automatisiertes Fahren" eingestuft [B07].

Viele betrachten autonome Systeme als "smart" oder "intelligent", was darauf hindeutet, dass sie KI-basierte Komponenten enthalten, um bestimmte Funktionen auszuführen. So verwenden autonome Fahrzeuge, die sich der Situation bewusst sein müssen, in der Regel mehrere Sensoren und Bildverarbeitung, um Informationen über die unmittelbare Umgebung des Fahrzeugs zu sammeln. Maschinelles Lernen, und insbesondere Deep Learning (siehe Abschnitt 6.1), hat sich als der effektivste Ansatz zur Erfüllung dieser Funktion erwiesen. Autonome Systeme können auch Entscheidungs- und Kontrollfunktionen beinhalten. Beide können mit KI-basierten Komponenten effektiv durchgeführt werden.

Auch wenn einige KI-basierte Systeme als autonom gelten, trifft dies nicht auf alle KI-basierten Systeme zu. In diesem Lehrplan wird Autonomie als die Fähigkeit des Systems betrachtet, über längere Zeiträume unabhängig von menschlicher Aufsicht und Kontrolle zu arbeiten. Dies kann dabei helfen, die Merkmale eines autonomen Systems zu ermitteln, die spezifiziert und getestet werden müssen. So muss beispielsweise bekannt sein, wie lange ein autonomes System ohne menschliches Eingreifen zufriedenstellend funktionieren soll. Darüber hinaus ist es wichtig, die Ereignisse zu ermitteln, bei denen das autonome System die Kontrolle an seine menschlichen Kontrolleure zurückgeben muss.

2.3 Evolution

In diesem Lehrplan wird Evolution als die Fähigkeit des Systems betrachtet, sich selbst als Reaktion auf veränderte äußere Beschränkungen zu verbessern. Einige KI-Systeme können als selbstlernend

bezeichnet werden, und erfolgreiche selbstlernende KI-basierte Systeme müssen diese Form der Evolution einbeziehen.

KI-basierte Systeme arbeiten oft in einer sich verändernden Umgebung. Wie andere Formen von IT-Systemen muss auch ein KI-basiertes System flexibel und anpassbar genug sein, um mit Veränderungen in seiner Einsatzumgebung fertig zu werden.

Selbstlernende KI-basierte Systeme müssen in der Regel zwei Arten von Veränderungen bewältigen:

- Eine Art der Veränderung besteht darin, dass das System aus seinen eigenen Entscheidungen und seinen Interaktionen mit der Umgebung lernt.
- Die andere Art der Veränderung besteht darin, dass das System aus Änderungen in der Einsatzumgebung des Systems lernt.

In beiden Fällen wird sich das System im Idealfall weiterentwickeln, um seine Effektivität und Effizienz zu verbessern. Diese Evolution muss jedoch eingeschränkt werden, um zu verhindern, dass das System unerwünschte Eigenschaften entwickelt. Jede Evolution muss weiterhin den ursprünglichen Systemanforderungen und -beschränkungen entsprechen. Wo diese fehlen, muss das System so verwaltet werden, dass jede Evolution innerhalb der Grenzen bleibt und stets mit den menschlichen Werten in Einklang gebracht wird. Abschnitt 2.7 enthält Beispiele für die Auswirkungen von Nebenwirkungen und Belohnungs-Hacking auf selbstlernende KI-basierte Systeme.

2.4 Verzerrung

Im Zusammenhang mit KI-basierten Systemen ist Verzerrung ein statistisches Maß für den Abstand zwischen den vom System gelieferten Ergebnissen und dem, was als "faire Ergebnisse" angesehen wird, die keine Bevorzugung einer bestimmten Gruppe aufweisen. Unangemessene Verzerrungen können mit Attributen wie Geschlecht, Rasse, ethnische Zugehörigkeit, sexuelle Orientierung, Einkommensniveau und Alter verbunden sein. Fälle von unangemessener Verzerrung in KI-basierten Systemen wurden beispielsweise für Systeme berichtet, die Empfehlungen für die Kreditvergabe von Banken aussprechen, sowie für Einstellungssysteme und Justizüberwachungssysteme.

Verzerrung kann in viele Arten KI-basierter Systeme eingebracht werden. So ist es beispielsweise schwierig zu verhindern, dass die Verzerrung von Experten in die von einem Expertensystem angewandten Regeln einfließt. Die weite Verbreitung von ML-Systemen führt jedoch dazu, dass ein Großteil der Diskussion über Verzerrung im Zusammenhang mit diesen Systemen geführt wird.

ML-Systeme werden eingesetzt, um Entscheidungen und Vorhersagen zu treffen, wobei Algorithmen verwendet werden, die auf gesammelte Daten zurückgreifen, und diese beiden Komponenten können zu Verzerrungen der Ergebnisse führen:

- Algorithmische Verzerrungen können auftreten, wenn der Lernalgorithmus falsch konfiguriert ist, z. B. wenn er einige Daten im Vergleich zu anderen überbewertet. Diese Quelle der Verzerrung kann durch Tuning der Hyperparameter der ML-Algorithmen verursacht und gesteuert werden (siehe Abschnitt 3.2).
- Stichprobenverzerrungen können auftreten, wenn die Trainingsdaten nicht vollständig repräsentativ für den Datenraum sind, auf den ML angewendet wird.

Eine unangemessene Verzerrung wird häufig durch Stichprobenverzerrungen verursacht, kann aber gelegentlich auch durch algorithmische Verzerrungen verursacht werden.

2.6 Ethik

Ethik wird im Cambridge-Lexikon definiert als:

> *Ein System von akzeptierten Überzeugungen, die das Verhalten kontrollieren, insbesondere solch ein System, das auf Moral beruht.*

KI-basierte Systeme mit verbesserten Fähigkeiten haben vielfältige Auswirkungen auf das Leben der Menschen. Mit der zunehmenden Verbreitung dieser Systeme wurden Bedenken geäußert, ob ihr Einsatz ethisch vertretbar ist.

Was als ethisch angesehen wird, kann sich im Laufe der Zeit ändern und auch von Ort zu Ort und von Kultur zu Kultur unterschiedlich sein. Es muss darauf geachtet werden, dass bei der Einführung eines KI-basierten Systems an einem anderen Standort die unterschiedlichen Werte der Beteiligten berücksichtigt werden.

Nationale und internationale Strategien zur Ethik der KI gibt es in vielen Ländern und Regionen. Die Organisation für wirtschaftliche Zusammenarbeit und Entwicklung (OECD) veröffentlichte 2019 ihre Grundsätze für KI, die ersten internationalen Standards, auf die sich die Regierungen für eine verantwortungsvolle Entwicklung von KI geeinigt haben [B08]. Diese Grundsätze wurden bei ihrer Veröffentlichung von zweiundvierzig Ländern angenommen und werden auch von der Europäischen Kommission unterstützt. Sie enthalten praktische politische Empfehlungen sowie wertebasierte Grundsätze für den "verantwortungsvollen Umgang mit vertrauenswürdiger KI". Diese sind wie folgt zusammengefasst:

- KI-Systeme sollten den Menschen und dem Planeten zugutekommen, indem sie integratives Wachstum, nachhaltige Entwicklung und Wohlstand fördern.
- KI-Systeme sollten die Rechtsstaatlichkeit, die Menschenrechte, die demokratischen Werte und die Diversität respektieren und geeignete Schutzmaßnahmen zur Gewährleistung einer fairen Gesellschaft vorsehen.
- KI-Systeme sollten transparent sein, damit die Menschen die Ergebnisse verstehen und sie in Frage stellen können.
- KI-Systeme müssen während ihres gesamten Lebenszyklus robust, sicher und zuverlässig funktionieren, und die Risiken sollten kontinuierlich bewertet werden.
- Organisationen und Personen, die KI-Systeme entwickeln, einsetzen oder betreiben, sollten rechenschaftspflichtig sein.

2.7 Nebenwirkungen und Belohnungs-Hacking

Nebenwirkungen und Belohnungs-Hacking können dazu führen, dass KI-basierte Systeme unerwartete und sogar schädliche Ergebnisse erzeugen, wenn das System versucht, seine Zielvorgaben zu erreichen [B09].

Negative Nebenwirkungen können entstehen, wenn der Designer eines KI-basierten Systems ein Ziel vorgibt, das "sich auf die Erledigung einiger spezifischer Aufgaben in der Umgebung konzentriert, aber andere Aspekte der (potenziell sehr großen) Umgebung ignoriert und damit implizit Gleichgültigkeit gegenüber Umgebungsvariablen zum Ausdruck bringt, deren Veränderung eigentlich schädlich sein könnte" [B09]. Ein selbstfahrendes Auto mit dem Ziel, "so kraftstoffsparend und sicher wie möglich" an sein Ziel zu gelangen, kann dieses Ziel zwar erreichen, allerdings mit der Nebenwirkung, dass sich die Fahrgäste über die übermäßig lange Fahrzeit extrem ärgern.

Belohnungs-Hacking kann dazu führen, dass ein KI-basiertes System ein bestimmtes Ziel durch eine "clevere" oder "einfache" Lösung erreicht, die "den Geist der Absicht des Designers pervertiert". Das Ziel kann sozusagen "abgezockt" werden. Ein weit verbreitetes Beispiel für Belohnungs-Hacking ist der

Fall, dass ein KI-basiertes System sich selbst beibringt, ein Arcade-Computerspiel zu spielen. Es wird mit dem Ziel konfrontiert, die "höchste Punktzahl" zu erreichen, und um dies zu erreichen, hackt es einfach den Datensatz, in dem die höchste Punktzahl gespeichert ist, anstatt das Spiel zu spielen, um sie zu erreichen.

2.8 Transparenz, Interpretierbarkeit und Erklärbarkeit

KI-basierte Systeme werden in der Regel in Bereichen eingesetzt, in denen die Nutzer diesen Systemen vertrauen müssen. Dies kann aus Sicherheitsgründen der Fall sein, aber auch dort, wo der Schutz der Privatsphäre erforderlich ist und wo sie potenziell lebensverändernde Vorhersagen und Entscheidungen treffen können.

Den meisten Nutzern werden KI-basierte Systeme als "Black Boxes" präsentiert, und die Nutzer wissen kaum, wie diese Systeme zu ihren Ergebnissen kommen. In einigen Fällen gilt diese Unwissenheit sogar für die Datenwissenschaftler, welche die Systeme entwickelt haben. Gelegentlich sind sich die Nutzer nicht einmal bewusst, dass sie mit einem KI-basierten System interagieren.

Die inhärente Komplexität KI-basierter Systeme hat zum Bereich der "erklärbaren KI" (XAI) geführt. Ziel von XAI ist es, dass die Nutzer nachvollziehen können, wie KI-basierte Systeme zu ihren Ergebnissen kommen, um so das Vertrauen der Nutzer in sie zu erhöhen.

Laut The Royal Society [B10] gibt es mehrere Gründe für den Wunsch nach XAI, darunter:

- Vertrauen der Nutzer in das System schaffen
- Schutz vor Verzerrung
- Erfüllung von regulativen Normen und Standards oder politischen Anforderungen
- Verbesserung des Systemdesigns
- Bewertung von Risiko, Robustheit und Schwachstellen
- Verstehen und Verifizierung der Ergebnisse eines Systems
- Autonomie, Handlungsfähigkeit (dem Nutzer das Gefühl geben, selbstbestimmt zu handeln) und Erfüllung sozialer Werte

Daraus ergeben sich die folgenden drei grundlegenden wünschenswerten XAI-Merkmale für KI-basierte Systeme aus der Sicht eines Stakeholders (siehe Abschnitt 8.6):

- Transparenz: Darunter versteht man die Leichtigkeit, mit der der Algorithmus und die Trainingsdaten, die zur Erstellung des Modells verwendet werden, ermittelt werden können.
- Interpretierbarkeit: Darunter versteht man die Verständlichkeit der KI-Technologie für die verschiedenen Beteiligten, einschließlich der Nutzer.
- Erklärbarkeit: Darunter versteht man die Leichtigkeit, mit der die Benutzer feststellen können, wie das KI-basierte System zu einem bestimmten Ergebnis kommt.

2.9 Funktionale Sicherheit und KI

In diesem Lehrplan wird funktionale Sicherheit als die Erwartung betrachtet, dass ein KI-basiertes System keine Schäden für Menschen, Eigentum oder die Umwelt verursacht. KI-basierte Systeme können verwendet werden, um Entscheidungen zu treffen, welche die Sicherheit beeinflussen. KI-basierte Systeme, die in den Bereichen Medizin, Fertigung, Verteidigung, IT-Sicherheit und Verkehr eingesetzt werden, können beispielsweise die Sicherheit beeinflussen.

Zu den Merkmalen KI-basierter Systeme, die es erschweren, ihre Sicherheit zu gewährleisten (z. B. Menschen nicht zu schaden), gehören:

- Komplexität
- Nicht-Determiniertheit
- probabilistischer Charakter
- selbstlernend
- mangelnde Transparenz, Interpretierbarkeit und Erklärbarkeit
- mangelnde Robustheit

Die Herausforderungen bei dem Test mehrerer dieser Merkmale werden in Kapitel 8 behandelt.

3 Maschinelles Lernen (ML) - Überblick - 145 Minuten

Schlüsselwörter

Keine

KI-spezifische Schlüsselwörter

Assoziation, Klassifikation, Clusterbildung, Datenvorbereitung, ML-Algorithmus, ML-Framework, Funktionale Leistungskriterien von ML, ML-Modell, ML-Trainingsdaten, ML-Workflow, ML-Modellevaluierung, ML-Modell-Tuning, Ausreißer, Überanpassung, Regression, bestärkendes Lernen, überwachtes Lernen, Unteranpassung, unüberwachtes Lernen

Lernziele für Kapitel 3:

3.1 Arten von ML

AI-3.1.1 K2 Beschreiben von Klassifikation und Regression als Teil des überwachten Lernens

AI-3.1.2 K2 Beschreiben von Clusterbildung und Assoziation als Teil des unüberwachten Lernens

AI-3.1.3 K2 Beschreiben von bestärkendem Lernen

3.2 ML-Workflow

AI-3.2.1 K2 Zusammenfassen des Workflows bei der Erstellung eines ML-Systems

3.3 Auswahl einer Art von ML

AI-3.3.1 K3 Identifizieren eines geeigneten ML-Ansatzes (Klassifikation, Regression, Clusterbildung, Assoziation oder bestärkendes Lernen) anhand eines Projektszenarios

3.4 Faktoren bei der Auswahl eines ML-Algorithmus

AI-3.4.1 K2 Erklären der Faktoren, die bei der Auswahl von ML-Algorithmen eine Rolle spielen

3.5 Überanpassung und Unteranpassung

AI-3.5.1 K2 Zusammenfassen der Konzepte von Überanpassung und Unteranpassung

HO-3.5.1 H0 Demonstrieren von Überanpassung und Unteranpassung

3.1 Arten von ML

ML-Algorithmen können in folgende Kategorien eingeteilt werden:

- überwachtes Lernen
- unüberwachtes Lernen
- bestärkendes Lernen

3.1.1 Überwachtes Lernen

Bei dieser Art des Lernens erstellt der Algorithmus in der Trainingsphase das ML-Modell aus gekennzeichneten Daten (*labelled data*). Die gekennzeichneten Daten, die in der Regel Paare von Eingaben umfassen (z. B. ein Bild eines Hundes und die Kennzeichnung "Hund"), werden vom Algorithmus verwendet, um die Beziehung zwischen den Eingabedaten (z. B. Bilder von Hunden) und den Ausgabebezeichnungen (z. B. "Hund" und "Katze") während des Trainings abzuleiten. In der Testphase des ML-Modells wird ein neuer Satz bislang unbekannter Daten auf das trainierte Modell angewendet, um die Ausgabe vorherzusagen. Das Modell wird eingesetzt, sobald die Genauigkeit der Ausgabe zufriedenstellend ist.

Probleme, die durch überwachtes Lernen gelöst werden, werden in zwei Kategorien unterteilt:

- Klassifikation: Dies ist der Fall, wenn das Problem erfordert, dass eine Eingabe in eine von wenigen vordefinierten Klassen eingeordnet wird. Gesichtserkennung oder Objekterkennung in einem Bild sind Beispiele für Probleme, bei denen Klassifikation verwendet wird.
- Regression: Dies ist der Fall, wenn das ML-Modell eine numerische Ausgabe mithilfe von Regression vorhersagen soll. Die Vorhersage des Alters einer Person auf der Grundlage von Eingabedaten über ihre Gewohnheiten oder die Vorhersage künftiger Aktienkurse sind Beispiele für Probleme, bei denen Regression eingesetzt wird.

Zu beachten ist, dass sich der Begriff Regression, wie er im Zusammenhang mit einem ML-Problem verwendet wird, von seiner Verwendung in anderen ISTQB®-Lehrplänen unterscheidet, wie z.B. [I01], wo der Begriff Regression verwendet wird, wenn Modifikationen einer Software änderungsbezogene Fehlerzustände verursachen.

3.1.2 Unüberwachtes Lernen

Bei dieser Art des Lernens erstellt der Algorithmus in der Trainingsphase das ML-Modell aus nicht gekennzeichneten Daten. Die nicht gekennzeichneten Daten werden vom Algorithmus verwendet, um während der Trainingsphase Muster in den Eingabedaten abzuleiten und die Eingaben auf der Grundlage ihrer Gemeinsamkeiten verschiedenen Klassen zuzuordnen. In der Testphase wird das trainierte Modell auf einen neuen Satz bislang unbekannter Daten angewendet, um vorherzusagen, welchen Klassen die Eingabedaten zugeordnet werden sollten. Das Modell wird eingesetzt, sobald die Genauigkeit der Ausgabe als zufriedenstellend angesehen wird.

Probleme, die durch unüberwachtes Lernen gelöst werden, lassen sich in zwei Kategorien einteilen:

- Clusterbildung: Dies ist der Fall, wenn das Problem die Identifizierung von Ähnlichkeiten in den Eingabedatenpunkten erfordert, so dass sie auf der Grundlage gemeinsamer Merkmale oder Eigenschaften gruppiert werden können. Clusterbildung wird zum Beispiel verwendet, um verschiedene Arten von Kunden für Marketingzwecke zu kategorisieren.
- Assoziation: Dies ist der Fall, wenn für das Problem interessante Beziehungen oder Abhängigkeiten zwischen Datenattributen ermittelt werden müssen. Zum Beispiel kann ein Produktempfehlungssystem Assoziationen auf der Grundlage des Einkaufsverhaltens der Kunden erkennen.

3.1.3 Bestärkendes Lernen

Bestärkendes Lernen ist ein Ansatz, bei dem das System (ein "intelligenter Agent") lernt, indem es iterativ mit der Umgebung interagiert und dabei aus der Erfahrung lernt. Beim bestärkenden Lernen werden keine Trainingsdaten verwendet. Der Agent wird belohnt, wenn er eine richtige Entscheidung trifft, und bestraft, wenn er eine falsche Entscheidung trifft.

Die zentralen Herausforderungen bei der Implementierung von bestärkendem Lernen sind die Einrichtung der Umgebung, die Wahl der richtigen Strategie für den Agenten, um das gewünschte Ziel zu erreichen, und der Entwurf einer Belohnungsfunktion. Robotik, autonome Fahrzeuge und Chatbots sind Beispiele für Anwendungen, die bestärkendes Lernen nutzen.

3.2 ML-Workflow

Die Aktivitäten im Workflow des maschinellen Lernens sind:

Verstehen der Ziele

Der Zweck des einzusetzenden ML-Modells muss verstanden und mit den Stakeholdern vereinbart werden, um dessen Übereinstimmung mit den geschäftlichen Prioritäten sicherzustellen. Für das entwickelte Modell sollten Akzeptanzkriterien (einschließlich funktionaler Leistungsmetriken von ML - siehe Kapitel 5) festgelegt werden.

Auswahl eines KI-Entwicklungs-Frameworks

Ein geeignetes KI-Entwicklungs-Framework sollte auf der Grundlage der Ziele, der Akzeptanzkriterien und der geschäftlichen Prioritäten ausgewählt werden (siehe Abschnitt 1.5).

Auswahl und Umsetzung des Algorithmus

Ein ML-Algorithmus wird auf der Grundlage verschiedener Faktoren ausgewählt, darunter die Ziele, Akzeptanzkriterien und die verfügbaren Daten (siehe Abschnitt 3.4). Der Algorithmus kann manuell kodiert werden, häufig wird er jedoch aus einer Bibliothek mit bereits geschriebenem Code abgerufen. Der kodierte Algorithmus wird dann kompiliert, um das Training des Modells vorzubereiten, falls letzteres erforderlich ist.

Vorbereitung und Test der Daten

Die Datenvorbereitung (siehe Abschnitt 4.1) umfasst die Datenbeschaffung, die Datenvorverarbeitung und die Merkmalsermittlung. Parallel zu diesen Aktivitäten kann eine explorative Datenanalyse (EDA) durchgeführt werden.

Die vom Algorithmus und Modell verwendeten Daten richten sich nach den Zielen und werden von allen Aktivitäten der in Abbildung 1 dargestellten Aktivität "Modellgenerierung & Test" genutzt. Handelt es sich bei dem System beispielsweise um ein Echtzeit-Handelssystem, werden die Daten vom Handelsmarkt stammen.

Die Daten, die zum Trainieren, Tuning und Testen des Modells verwendet werden, müssen repräsentativ für die operativen Daten sein, die das Modell verwenden wird. In einigen Fällen ist es möglich, vorab gesammelte Datensätze für das anfängliche Training des Modells zu verwenden (siehe z. B. Kaggle-Datensätze [R16]). Andernfalls müssen die Rohdaten in der Regel vorverarbeitet und Merkmale ermittelt werden.

Die Daten und alle automatisierten Datenvorbereitungsschritte müssen getestet werden. Siehe Abschnitt 7.2.1 für weitere Einzelheiten zum Test der Eingabedaten.

Training des Modells

Der ausgewählte ML-Algorithmus verwendet Trainingsdaten, um das Modell zu trainieren.

Einige Algorithmen, wie z. B. diejenigen, die ein neuronales Netz erzeugen, lesen den Trainingsdatensatz mehrmals. Jede Iteration des Trainings mit dem Trainingsdatensatz wird als Epoche bezeichnet.

Die Parameter zur Definition der Modellstruktur werden an den Algorithmus übergeben (z. B. die Anzahl der Schichten eines neuronalen Netzes oder die Tiefe eines Entscheidungsbaums). Diese Parameter werden als Hyperparameter des Modells bezeichnet.

Parameter, die das Training steuern (z. B. wie viele Epochen beim Training eines neuronalen Netzes verwendet werden sollen), werden ebenfalls an den Algorithmus übergeben. Diese Parameter werden als Hyperparameter des Algorithmus bezeichnet.

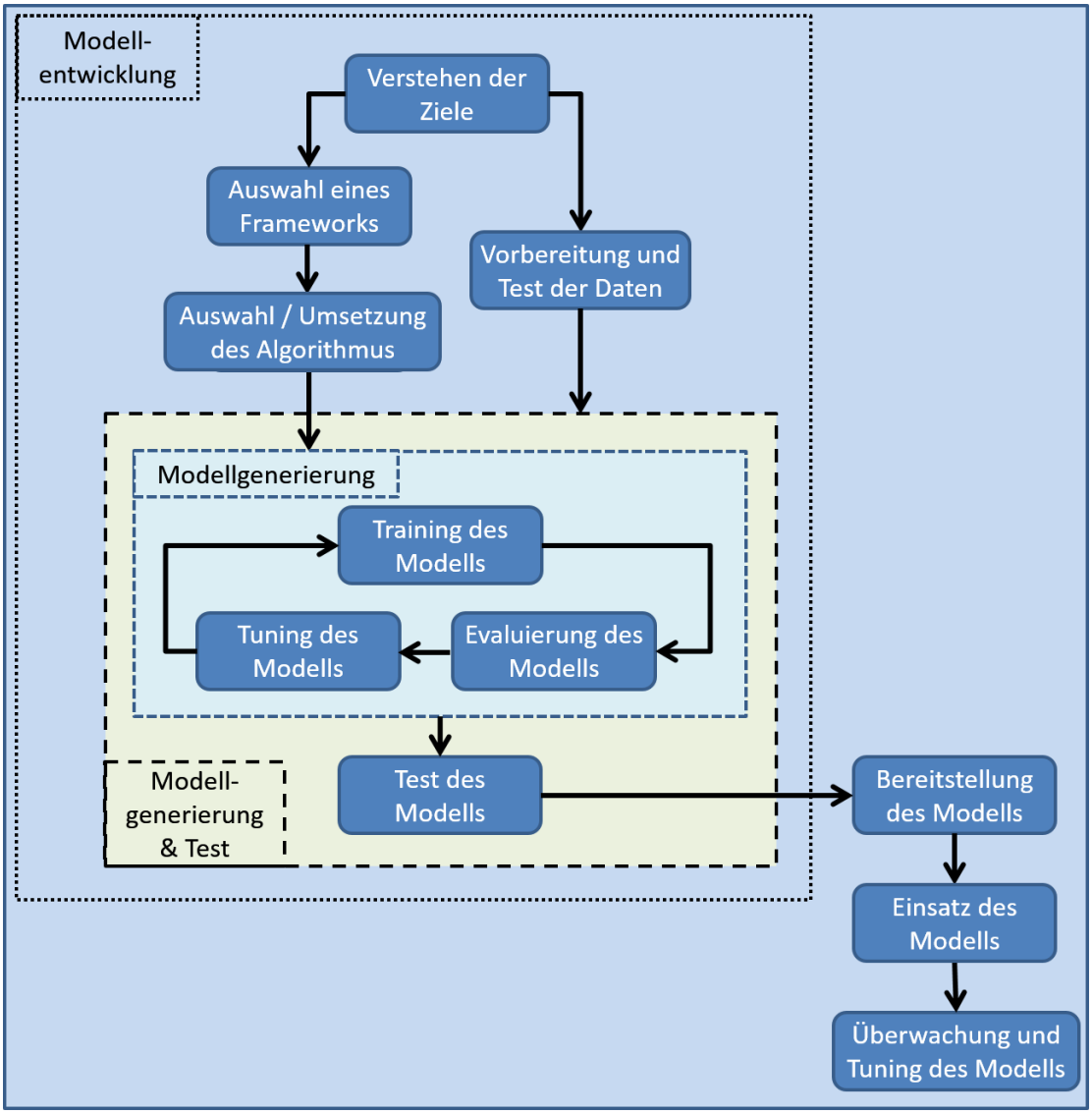

Abbildung 1: ML-Workflow

Evaluierung des Modells

Das Modell wird anhand der vereinbarten funktionalen Leistungsmetriken von ML unter Verwendung des Validierungsdatensatzes (siehe Abschnitt 4.2) evaluiert, und die Ergebnisse werden dann zum Tuning des Modells verwendet. Die Modellevaluierung und -optimierung sollte einem wissenschaftlichen Experiment ähneln, das sorgfältig unter kontrollierten Bedingungen durchgeführt und klar dokumentiert werden muss. In der Praxis werden in der Regel mehrere Modelle mit verschiedenen Algorithmen erstellt und trainiert (z. B. Random Forests, SVM und neuronale Netze), und das beste Modell wird auf der Grundlage der Ergebnisse der Evaluierung und Tuning ausgewählt.

Tuning des Modells

Die Ergebnisse der Evaluierung des Modells anhand der vereinbarten funktionalen Leistungsmetriken von ML werden verwendet, um die Modelleinstellungen an die Daten anzupassen und so die Leistung zu verbessern. Das Modell kann durch Hyperparameter-Tuning abgestimmt werden, wobei die Trainingsaktivität geändert wird (z. B. durch Änderung der Anzahl der Trainingsschritte oder durch Änderung der für das Training verwendeten Datenmenge), oder Attribute des Modells werden aktualisiert (z. B. die Anzahl der Neuronen in einem neuronalen Netz oder die Tiefe eines Entscheidungsbaums).

Die drei Aktivitäten Training, Evaluierung und Tuning können als Modellgenerierung betrachtet werden, wie in Abbildung 1 dargestellt.

Test des Modells

Sobald ein Modell erstellt wurde (d. h. es wurde trainiert, evaluiert und getunt), sollte es anhand eines unabhängigen Testdatensatzes getestet werden, um sicherzustellen, dass die vereinbarten funktionalen Leistungskriterien von ML erfüllt werden (siehe Abschnitt 7.2.2). Die funktionalen Leistungsmetriken von ML aus dem Test werden auch mit denen aus der Evaluierung verglichen, und wenn die Leistung des Modells mit unabhängigen Daten deutlich geringer ist als während der Evaluierung, kann es notwendig sein, ein anderes Modell auszuwählen.

Zusätzlich zu den funktionalen Leistungstests müssen auch nicht-funktionale Tests durchgeführt werden, z. B. für die Zeit zum Trainieren des Modells und die Zeit und den Ressourcenverbrauch für die Erstellung einer Vorhersage. In der Regel werden diese Tests vom Dateningenieur/Wissenschaftler durchgeführt, aber auch Tester mit ausreichenden Kenntnissen des Fachgebiets und Zugang zu den entsprechenden Ressourcen können diese Tests durchführen.

Bereitstellung des Modells

Nach Abschluss der Modellentwicklung, wie in Abbildung 1 dargestellt, muss das getunte Modell in der Regel für die Bereitstellung zusammen mit den zugehörigen Ressourcen, einschließlich der entsprechenden Datenpipeline (siehe Abschnitt 4.1), neu entwickelt bzw. angepasst werden. Dies wird normalerweise durch das KI-Entwicklungs-Framework erreicht. Zu den Zielen können eingebettete Systeme und die Cloud gehören, wo auf das Modell über eine Web-API zugegriffen werden kann.

Einsatz des Modells

Nach der Bereitstellung ist das Modell in der Regel Teil eines größeren KI-basierten Systems und kann operativ eingesetzt werden. Modelle können geplante Batch-Vorhersagen in bestimmten Zeitabständen durchführen oder auf Anfrage in Echtzeit laufen.

Überwachung und Tuning des Modells

Während des Einsatzes des Modells kann sich die Situation verändern und das Modell kann von seiner beabsichtigten Leistung abweichen (siehe Abschnitte 2.3 und 7.6). Um sicherzustellen, dass jede Abweichung erkannt und behandelt wird, sollte das im Betrieb eingesetzte Modell regelmäßig anhand seiner Akzeptanzkriterien evaluiert werden.

Es kann als notwendig erachtet werden, die Modelleinstellungen zu aktualisieren, um das Problem der Abweichungen zu lösen, oder es kann entschieden werden, dass ein erneutes Training mit neuen Daten erforderlich ist, um ein genaueres oder robusteres Modell zu erstellen. In diesem Fall kann ein neues Modell mit aktualisierten Trainingsdaten erstellt und trainiert werden. Das neue Modell kann dann mit dem bestehenden Modell mit einer Art von A/B-Testen verglichen werden (siehe Abschnitt 9.4).

Der in Abbildung 1 dargestellte ML-Workflow ist eine logische Abfolge. In der Praxis wird der Arbeitsablauf so angewandt, dass die Schritte iterativ wiederholt werden (z. B. ist es bei der Auswertung des Modells oft notwendig, zum Trainingsschritt und manchmal zur Datenvorbereitung zurückzukehren).

Die in Abbildung 1 dargestellten Schritte beinhalten nicht die Integration des ML-Modells mit den Nicht-ML-Teilen des Gesamtsystems. ML-Modelle können in der Regel nicht isoliert bereitgestellt werden und müssen mit den Nicht-ML-Teilen integriert werden. In Bildverarbeitungsanwendungen gibt es beispielsweise eine Datenpipeline, die Daten bereinigt und verändert, bevor sie an das ML-Modell weitergeleitet werden. Wenn das Modell Teil eines größeren KI-basierten Systems ist, muss es vor der Bereitstellung in dieses System integriert werden. In diesem Fall können Integrations-, System- und Akzeptanztests durchgeführt werden, wie in Abschnitt 7.2 beschrieben.

3.3 Auswahl einer Art von ML

Für die Auswahl eines geeigneten ML-Ansatzes gelten die folgenden Leitlinien:

- Es sollten ausreichend Trainings- und Testdaten für den gewählten ML-Ansatz zur Verfügung stehen.
- Für das überwachte Lernen ist es erforderlich, dass die Daten korrekt gekennzeichnet sind.
- Wenn es eine Kennzeichen-Ausgabe gibt, kann es sich um überwachtes Lernen handeln.
- Wenn die Ausgabe diskret und kategorisch ist, kann es sich um eine Klassifikation handeln.
- Wenn die Ausgabe numerisch und kontinuierlich ist, kann es sich um eine Regression handeln.
- Ist in dem gegebenen Datensatz keine Ausgabe vorgesehen, kann es sich um unüberwachtes Lernen handeln.
- Wenn das Problem die Gruppierung ähnlicher Daten beinhaltet, kann es sich um Clusterbildung handeln.
- Wenn das Problem beinhaltet, gemeinsam auftretende Daten zu finden, kann es sich um eine Assoziation handeln.
- Das bestärkende Lernen eignet sich besser für Kontexte, in denen eine Interaktion mit der Umwelt stattfindet.
- Wenn das Problem die Vorstellung von mehreren Zuständen beinhaltet und in jedem Zustand Entscheidungen getroffen werden müssen, kann das bestärkende Lernen anwendbar sein.

3.4 Faktoren, die bei der Auswahl von ML-Algorithmen eine Rolle spielen

Für die Auswahl des optimalen ML-Algorithmus, der ML-Modelleinstellungen und der ML-Modell-Hyperparameter gibt es keinen eindeutigen Ansatz. In der Praxis werden diese auf der Grundlage einer Mischung aus den folgenden Faktoren ausgewählt:

- Die erforderliche Funktionalität (z. B. Klassifikation oder Vorhersage eines diskreten Wertes)
- Die geforderten Qualitätsmerkmale, wie
  - Genauigkeit (z. B. können einige Modelle genauer, aber auch langsamer sein)
  - Beschränkungen des verfügbaren Speichers (z. B. bei einem eingebetteten System)
  - die Geschwindigkeit des Trainings (und des erneuten Trainings) des Modells
  - die Geschwindigkeit der Vorhersage (z. B. für Echtzeitsysteme)
  - Anforderungen an Transparenz, Interpretierbarkeit und Erklärbarkeit
- die Art der Daten, die für das Training des Modells zur Verfügung stehen (z. B. können einige Modelle nur mit Bilddaten arbeiten)
- die Menge der Daten, die für das Trainieren und Testen des Modells zur Verfügung stehen (einige Modelle könnten zum Beispiel dazu neigen, bei einer begrenzten Datenmenge in höherem Maße als andere Modelle überangepasst zu sein)

- Die Anzahl der Merkmale in den Eingabedaten, die voraussichtlich vom Modell verwendet werden (z. B. werden andere Faktoren, wie Geschwindigkeit und Genauigkeit, wahrscheinlich direkt von der Anzahl der Merkmale beeinflusst)
- die erwartete Anzahl von Klassen für die Clusterbildung (z. B. können einige Modelle für Probleme mit mehr als einer Klasse ungeeignet sein)
- Frühere Erfahrungen
- Versuch und Irrtum

3.5 Überanpassung und Unteranpassung

3.5.1 Überanpassung

Eine Überanpassung liegt vor, wenn das Modell zu eng an eine Reihe von Datenpunkten angepasst ist und nicht richtig verallgemeinern kann. Ein solches Modell funktioniert sehr gut mit den Daten, mit denen es trainiert wurde, aber es kann Schwierigkeiten haben, genaue Vorhersagen für neue Daten zu machen. Eine Überanpassung kann auftreten, wenn das Modell versucht, sich an jeden Datenpunkt anzupassen, auch an solche Datenpunkte, die als Rauschen oder Ausreißer bezeichnet werden können. Dies kann auch vorkommen, wenn der Trainingsdatensatz nicht genügend Daten enthält.

3.5.2 Unteranpassung

Unteranpassung liegt vor, wenn das Modell nicht ausgereift genug ist, um die Muster in den Trainingsdaten genau zu erfassen. Unterangepasste Modelle sind in der Regel zu simpel und können sowohl für neue Daten als auch für Daten, die den Trainingsdaten sehr ähnlich sind, keine genauen Vorhersagen liefern. Eine Ursache für Unteranpassung kann ein Trainingsdatensatz sein, der keine Merkmale enthält, die wichtige Beziehungen zwischen Eingaben und Ausgaben widerspiegeln. Sie kann auch auftreten, wenn der Algorithmus nicht zu den Daten passt (z. B. Erstellung eines linearen Modells für nicht-lineare Daten).

3.5.3 Praktische Übung: Demonstration von Überanpassung und Unteranpassung

Demonstration der Konzepte der Überanpassung und Unteranpassung eines Modells. Dies könnte anhand eines Datensatzes mit sehr wenigen Daten (Überanpassung) sowie eines Datensatzes mit schlechten Merkmalskorrelationen (Unteranpassung) demonstriert werden.

4 ML - Daten - 230 Minuten

Schlüsselwörter

Keine

KI-spezifische Schlüsselwörter

Annotation, Anreicherung, Klassifikationsmodell, Datenkennzeichnung, Datenvorbereitung, ML-Trainingsdaten, überwachtes Lernen, Testdatensatz, Validierungsdatensatz

Lernziele für Kapitel 4:

4.1 Datenvorbereitung als Teil des ML-Workflows

AI-4.1.1 K2 Beschreiben von Aktivitäten und Herausforderungen im Zusammenhang mit der Datenvorbereitung

HO-4.1.1 H2 Durchführen der Datenvorbereitung zur Unterstützung der Erstellung eines ML-Modells

4.2 Trainings-, Validierungs- und Testdatensätze im ML-Workflow

AI-4.2.1 K2 Vergleichen der Verwendung von Trainings-, Validierungs- und Testdatensätzen bei der Entwicklung eines ML-Modells

HO-4.2.1 H2 Identifizieren von Trainings- und Testdatensätzen und Erstellen eines ML-Modells

4.3 Probleme mit der Datenqualität

AI-4.3.1 K2 Beschreiben von typischen Qualitätsproblemen bei Datensätzen

4.4 Datenqualität und ihre Auswirkungen auf das ML-Modell

AI-4.4.1 K2 Erkennen, wie schlechte Datenqualität zu Problemen mit dem resultierenden ML-Modell führen kann

4.5 Datenkennzeichnung für überwachtes Lernen

AI-4.5.1 K1 Erinnern der verschiedenen Ansätze zur Kennzeichnung von Daten in Datensätzen für überwachtes Lernen

AI-4.5.2 K1 Erinnern der Gründe für eine falsche Kennzeichnung der Daten in Datensätzen

4.1 Datenvorbereitung als Teil des ML-Workflows

Auf die Datenvorbereitung entfallen durchschnittlich 43 % des Aufwands im ML-Workflow, und sie ist wahrscheinlich die ressourcenintensivste Aktivität im ML-Workflow. Im Vergleich dazu entfallen auf die Modellauswahl und -erstellung nur 17 % [R17]. Die Datenvorbereitung ist Teil der Datenpipeline, die Rohdaten aufnimmt und Daten in einer Form ausgibt, die sowohl für das Training eines ML-Modells als auch für die Vorhersage durch ein trainiertes ML-Modell brauchbar ist.

Die Datenvorbereitung kann die folgenden Aktivitäten umfassen:

Datenbeschaffung

- Identifizierung: Die Arten von Daten, die für das Training und die Vorhersagen verwendet werden sollen, werden ermittelt. Für ein selbstfahrendes Auto könnte dies zum Beispiel die Ermittlung des Bedarfs an Radar-, Video- und Lidar-Daten (*Laser imaging, detecting, and ranging*, Lidar) beinhalten.
- Erfassung: Die Datenquelle wird ermittelt und die Mittel zur Datenerhebung werden festgelegt. Dies könnte beispielsweise die Identifizierung des Internationalen Währungsfonds (IWF) als Quelle für Finanzdaten und die Kanäle umfassen, über welche die Daten in das KI-basierte System eingespeist werden sollen.
- Kennzeichnung: Siehe Abschnitt 4.5 Datenkennzeichnung für überwachtes Lernen.

Die erfassten Daten können in verschiedenen Formaten vorliegen (z. B. numerisch, kategorisch, bildlich, tabellarisch, textuell, als Zeitreihe, sensorisch, als räumliche Geodaten, als Video und Audio).

Vorverarbeitung der Daten

- Bereinigung: Werden fehlerhafte Daten, doppelte Daten oder Ausreißer festgestellt, werden sie entweder entfernt oder korrigiert. Darüber hinaus können fehlende Datenwerte durch geschätzte oder vermutete Werte ersetzt werden (z. B. durch Mittelwert, Median und Modus[3]). Auch können personenbezogene Daten entfernt oder anonymisiert werden.
- Umwandlung: Das Format der gegebenen Daten wird geändert (z. B. Zerlegung einer als Zeichenkette gespeicherten Adresse in ihre Bestandteile, Weglassen eines Feldes mit einem zufälligen Bezeichner, Umwandlung kategorischer Daten in numerische Daten, Änderung von Bildformaten). Einige der auf numerische Daten angewandten Transformationen beinhalten eine Skalierung, um sicherzustellen, dass derselbe Bereich verwendet wird. Bei der Standardisierung werden die Daten zum Beispiel so skaliert, dass sie einen Mittelwert von Null und eine Standardabweichung von Eins haben. Durch eine Normierung kann z.B. sichergestellt werden, dass die Daten einen Bereich zwischen Null und Eins aufweisen.
- Anreicherung: Diese wird verwendet, um die Anzahl der Stichproben in einem Datensatz zu erhöhen. Die Anreicherung kann auch verwendet werden, um gegnerische Beispiele in die Trainingsdaten einzubeziehen, was die Robustheit gegen gegnerische Angriffe erhöht (siehe Abschnitt 9.1.
- Stichprobenerhebung: Hierbei wird ein Teil des gesamten verfügbaren Datensatzes ausgewählt, damit Muster im größeren Datensatz beobachtet werden können. Dies geschieht in der Regel, um Kosten und benötigte Zeit für die Erstellung des ML-Modells zu reduzieren.

Zu beachten ist, dass jede Vorverarbeitung das Risiko birgt, nützliche gültige Daten zu verändern oder ungültige Daten hinzuzufügen.

[3] Der Modus (Modalwert) ist der häufigste Wert in einer Stichprobe.

Merkmalsermittlung

- Auswahl von Merkmalen: Ein Merkmal (*feature*) ist ein Attribut/eine Eigenschaft, das/die sich in den Daten widerspiegelt. Die Auswahl von Merkmalen fokussiert auf solche Merkmale, die am ehesten zur Modellbildung und -vorhersage beitragen. In der Praxis bedeutet dies häufig, dass Merkmale entfernt werden, von denen nicht erwartet wird (oder nicht erwünscht wird), dass sie sich auf das resultierende Modell auswirken. Durch die Entfernung irrelevanter Informationen (Rauschen) kann die Merkmalsauswahl die Gesamttrainingszeit verkürzen, eine Überanpassung verhindern (siehe Abschnitt 3.5.1 Überanpassung), die Genauigkeit erhöhen und die Modelle verallgemeinerbarer machen.
- Extraktion von Merkmalen: Dies beinhaltet die Ableitung informativer und nicht redundanter Merkmale aus den vorhandenen Merkmalen. Der daraus resultierende Datensatz ist in der Regel kleiner und kann verwendet werden, um ein ML-Modell mit gleicher Genauigkeit kostengünstiger und schneller zu erstellen.

Parallel zu diesen Datenvorbereitungsaktivitäten wird in der Regel auch eine explorative Datenanalyse (EDA) durchgeführt, um die Gesamtaufgabe der Datenvorbereitung zu unterstützen. Dazu gehört die Durchführung von Datenanalysen zur Entdeckung von Trends in den Daten und die Verwendung von Datenvisualisierung zur Darstellung von Trends in den Daten.

Obwohl die oben genannten Datenvorbereitungsaktivitäten und -unteraktivitäten in einer logischen Reihenfolge dargestellt wurden, können verschiedene Projekte sie anders anordnen oder nur eine Teilmenge von ihnen verwenden. Einige der Datenvorbereitungsschritte, wie z. B. die Identifizierung der Datenquelle, werden nur einmal ausgeführt und können manuell durchgeführt werden. Andere Schritte können Teil der operativen Datenpipeline sein und arbeiten normalerweise mit Live-Daten. Diese Aufgaben sollten automatisiert werden.

4.1.1 Herausforderungen bei der Datenvorbereitung

Einige der Herausforderungen im Zusammenhang mit der Datenvorbereitung sind:

- Der Bedarf an Wissen über:
  - den Anwendungsbereich.
  - die Daten und ihre Eigenschaften.
  - die verschiedenen Techniken der Datenvorbereitung.
- Die Schwierigkeit, qualitativ hochwertige Daten aus verschiedenen Quellen zu erhalten.
- Die Schwierigkeit, die Datenpipeline zu automatisieren und sicherzustellen, dass die Produktionsdatenpipeline sowohl skalierbar ist als auch eine angemessene Performanz aufweist (z. B. die für die Verarbeitung eines Datenelements erforderliche Zeit).
- Die mit der Datenvorbereitung verbundenen Kosten.
- Der Überprüfung auf Fehlerzustände, die während der Datenvorbereitung in die Datenpipeline eingebracht werden, wird nicht genügend Priorität eingeräumt.
- Die Einführung von Stichprobenverzerrungen (siehe Abschnitt 2.4 Verzerrung).

4.1.2 Praktische Übung: Datenvorbereitung für ML

Führen Sie für einen gegebenen Rohdatensatz die in Abschnitt 4.1 Datenvorbereitung als Teil des ML-Workflowsbeschriebenen Schritte zur Datenvorbereitung durch, um einen Datensatz zu erzeugen, der zur Erstellung eines Klassifikationsmodells mit überwachtem Lernen verwendet wird.

Diese Aktivität ist der erste Schritt zur Erstellung eines ML-Modells, das für künftige Übungen verwendet wird.

Für die Durchführung dieser Aktivität werden den Lernenden geeignete (und sprachspezifische) Materialien zur Verfügung gestellt, darunter:

- Bibliotheken
- ML-Frameworks
- Werkzeuge
- Eine Entwicklungsumgebung

4.2 Trainings-, Validierungs- und Testdatensätze im ML-Workflow

Zur Entwicklung eines ML-Modells sind drei Sätze gleichwertiger Daten (d. h. zufällig aus einem einzigen Ausgangsdatensatz ausgewählt) erforderlich:

- Ein Trainingsdatensatz, der zum Trainieren des Modells verwendet wird.
- Ein Validierungsdatensatz, der für die Evaluierung und das nachfolgende Tuning des Modells verwendet wird.
- Ein Testdatensatz (auch als Holdout-Datensatz bezeichnet), der zum Testen des abgestimmten Modells verwendet wird.

Wenn unbegrenzt viele geeignete Daten zur Verfügung stehen, hängt die Menge der Daten, die im ML-Arbeitsablauf für das Training, die Evaluierung und das Testen verwendet werden, in der Regel von den folgenden Faktoren ab:

- Der zum Trainieren des Modells verwendete Algorithmus.
- Die Verfügbarkeit von Ressourcen wie Arbeitsspeicher, Festplattenplatz, Rechenleistung, Netzwerkbandbreite und die verfügbare Zeit.

Da es in der Praxis schwierig ist, genügend geeignete Daten zu beschaffen, werden die Trainings- und Validierungsdatensätze häufig aus einem einzigen kombinierten Datensatz abgeleitet. Der Testdatensatz wird getrennt gehalten und während des Trainings nicht verwendet. Auf diese Weise wird sichergestellt, dass das entwickelte Modell nicht durch die Testdaten beeinflusst wird und die Testergebnisse die Qualität des Modells korrekt widerspiegeln.

Es gibt kein optimales Verhältnis für die Aufteilung des kombinierten Datensatzes in die drei Einzeldatensätze, aber typische Verhältnisse, die als Richtwert verwendet werden können, reichen von 60:20:20 bis 80:10:10 (Training : Validierung : Test). Die Aufteilung der Daten in diese Datensätze erfolgt häufig nach dem Zufallsprinzip, es sei denn, der Datensatz ist klein oder es besteht die Gefahr, dass die resultierenden Datensätze nicht repräsentativ für die erwarteten operativen Daten sind.

Wenn nur begrenzte Daten zur Verfügung stehen, kann die Aufteilung der verfügbaren Daten in drei Datensätze dazu führen, dass nicht genügend Daten für ein effektives Training verfügbar sind. Um dieses Problem zu lösen, können die Trainings- und Validierungsdatensätze kombiniert werden (wobei der Testdatensatz getrennt bleibt) und dann verwendet werden, um mehrfach geteilte (Split) Kombinationen dieses Datensatzes zu erstellen (z. B. 80% Training / 20% Validierung). Die Daten werden dann nach dem Zufallsprinzip den Trainings- und Validierungsdatensätzen zugewiesen. Training, Validierung und Tuning werden unter Verwendung dieser mehrfach geteilten Kombinationen durchgeführt, um mehrere getunte Modelle zu erstellen. Die Gesamtleistung des Modells kann als Durchschnitt über alle Läufe berechnet werden. Es gibt verschiedene Methoden zur Erstellung von mehrfach geteilten Kombinationen, darunter Split-Test, Bootstrap, k-fache Kreuzvalidierung und Leave-One-Out-Kreuzvalidierung (siehe [B02] für weitere Einzelheiten).

4.2.1 Praktische Übung: Identifizieren von Trainings- und Testdaten und Erstellen eines ML-Modells

Teilen Sie die zuvor aufbereiteten Daten (siehe Übung in Abschnitt 4.1.2) in Trainings-, Validierungs- und Testdatensätze auf.

Trainieren und testen Sie ein Klassifikationsmodell mit Hilfe von überwachtem Lernen mit diesen Datensätzen.

Erläutern Sie den Unterschied zwischen Evaluierung/Tuning und Test, indem Sie die mit den Validierungs- und Testdatensätzen erzielte Genauigkeit vergleichen.

4.3 Probleme mit der Datensatzqualität

Einige der typischen Qualitätsprobleme im Zusammenhang mit den Daten in einem Datensatz sind in der folgenden Tabelle 1 aufgeführt:

Qualitätsaspekt	Beschreibung
Falsche Daten	Die erfassten Daten waren fehlerhaft (z. B. durch einen defekten Sensor) oder wurden falsch eingegeben (z. B. durch Copy-Paste-Fehlhandlungen).
Unvollständige Daten	Es können Datenwerte fehlen (z. B. kann ein Feld in einem Datensatz leer sein, oder die Daten für ein bestimmtes Zeitintervall wurden ausgelassen). Für unvollständige Daten kann es verschiedene Gründe geben, darunter IT-Sicherheitsprobleme, Hardwareprobleme und menschliches Versagen.
Falsch gekennzeichnete Daten	Es gibt mehrere mögliche Gründe für eine falsche Kennzeichnung von Daten (siehe Abschnitt 4.5.2).
Unzureichende Daten	Es stehen nicht genügend Daten zur Verfügung, um Muster durch die verwendeten Lernalgorithmen zu erkennen (zu beachten ist, dass die erforderliche Mindestdatenmenge für verschiedene Algorithmen unterschiedlich ist).
Nicht vorverarbeitete Daten	Die Daten sollten vorverarbeitet werden, um sicherzustellen, dass sie sauber sind, ein einheitliches Format haben und keine unerwünschten Ausreißer enthalten (siehe Abschnitt 4.1).
Überholte Daten	Die Daten, die sowohl für das Lernen als auch für die Vorhersage verwendet werden, sollten so aktuell wie möglich sein (z. B. kann die Verwendung von Finanzdaten, die mehrere Jahre zurückliegen, durchaus zu ungenauen Ergebnissen führen).
Unausgewogene Daten	Unausgewogene Daten können aus unangemessener Verzerrung (z. B. aufgrund von Rasse, Geschlecht oder ethnischer Zugehörigkeit), ungünstiger Platzierung von Sensoren (z. B. Gesichtserkennungskameras in Deckenhöhe), Schwankungen bei der Verfügbarkeit von Datensätzen und unterschiedlichen Motivationen der Datenlieferanten resultieren.

Qualitätsaspekt	Beschreibung
Unfaire Daten	Fairness ist ein subjektives Qualitätsmerkmal, kann aber oft festgestellt werden. Um beispielsweise die Vielfalt oder die Ausgewogenheit zwischen den Geschlechtern zu fördern, können ausgewählte Daten gegenüber Minderheiten oder benachteiligten Gruppen positiv verzerrt sein (zu beachten ist, dass solche Daten zwar als fair, aber nicht als ausgewogen angesehen werden können).
Doppelte Daten	Wiederholte Datensätze können das resultierende ML-Modell übermäßig beeinflussen.
Irrelevante Daten	Daten, die für das zu behandelnde Problem nicht relevant sind, können die Ergebnisse negativ beeinflussen und zu einer Verschwendung von Ressourcen führen.
Fragen des Datenschutzes	Bei jeder Datennutzung sind die einschlägigen Datenschutzgesetze zu beachten (z. B. GDPR in Bezug auf personenbezogene Daten in der Europäischen Union, [R33]).
Fragen der IT-Sicherheit	Betrügerische oder irreführende Daten, die absichtlich in die Trainingsdaten eingefügt wurden, können zur Ungenauigkeit des trainierten Modells führen.

Tabelle 1: Typische Probleme der Datensatzqualität

4.4 Datenqualität und ihre Auswirkungen auf das ML-Modell

Die Qualität des ML-Modells hängt in hohem Maße von der Qualität des Datensatzes ab, aus dem es erstellt wird. Daten von schlechter Qualität können sowohl zu fehlerhaften Modellen als auch zu fehlerhaften Vorhersagen führen.

Die folgenden Kategorien von Fehlerzuständen sind auf Probleme mit der Datenqualität zurückzuführen:

- Verminderte Genauigkeit: Diese Mängel werden durch falsche, unvollständige, falsch gekennzeichnete, unzureichende, überholte oder irrelevante Daten sowie durch Daten verursacht, die nicht vorverarbeitet wurden. Wenn die Daten beispielsweise dazu verwendet werden, ein Modell der zu erwartenden Hauspreise zu erstellen, die Trainingsdaten aber nur wenige oder gar keine Daten über Einfamilienhäuser mit Wintergärten enthalten, dann wären die vorhergesagten Preise für diesen speziellen Haustyp wahrscheinlich ungenau.
- Verzerrtes Modell: Diese Fehler werden durch unvollständige, unausgewogene, unfaire, wenig vielfältige oder doppelte Daten verursacht. Wenn beispielsweise die Daten für ein bestimmtes Merkmal fehlen (z. B. wenn alle medizinischen Daten für die Vorhersage von Krankheiten von Probanden eines bestimmten Geschlechts stammen), wird sich dies wahrscheinlich nachteilig auf das resultierende Modell auswirken (es sei denn, das Modell soll nur dazu verwendet werden, Vorhersagen für dieses Geschlecht zu treffen).
- Kompromittiertes Modell: Diese Fehlerzustände sind auf unzureichenden Datenschutz und IT-Sicherheitsmängel zurückzuführen. Beispielsweise können Datenschutzprobleme in den Daten zu Sicherheitslücken führen, die es Angreifern ermöglichen würden, Informationen aus den Modellen zurückzugewinnen, was wiederum zur Preisgabe persönlicher Daten führen könnte.

4.5 Datenkennzeichnung für überwachtes Lernen

Die Datenkennzeichnung ist die Ergänzung von nicht gekennzeichneten (oder schlecht gekennzeichneten) Daten durch Hinzufügen von Kennzeichnungen, so dass sie für die Verwendung beim überwachten Lernen geeignet sind. Die Datenkennzeichnung ist eine ressourcenintensive Tätigkeit, die Berichten zufolge durchschnittlich 25 % der Zeit in ML-Projekten in Anspruch nimmt [B11].

In ihrer einfachsten Form kann die Kennzeichnung von Daten darin bestehen, dass Bilder oder Textdateien je nach ihrer Klasse in verschiedenen Ordnern abgelegt werden. So werden beispielsweise alle Textdateien mit positiven Produktbewertungen in einem Ordner und alle negativen Bewertungen in einem anderen Ordner abgelegt. Die Kennzeichnung von Objekten in Bildern durch das Zeichnen von Rechtecken um sie herum ist eine weitere gängige Kennzeichnungstechnik, die oft als Annotation bezeichnet wird. Für die Kennzeichnung von 3D-Objekten oder für das Zeichnen von Begrenzungsrahmen (*bounding box*) um unregelmäßige Objekte können komplexere Annotationen erforderlich sein. Datenkennzeichnung und Annotation werden in der Regel durch Werkzeuge unterstützt.

4.5.1 Ansätze zur Datenkennzeichnung

Die Kennzeichnung kann auf verschiedene Weise erfolgen:

- Intern: Die Kennzeichnung wird von Entwicklern, Testern oder einem Team innerhalb des Unternehmens durchgeführt, das für die Datenkennzeichnung eingerichtet wurde.
- Ausgelagert: Die Kennzeichnung wird von einer externen Fachorganisation durchgeführt.
- Crowdsourced: Die Kennzeichnung wird von einer großen Gruppe von Personen vorgenommen. Da es schwierig ist, die Qualität der Kennzeichnung zu kontrollieren, können mehrere Personen gebeten werden, dieselben Daten zu kennzeichnen, und dann wird eine Entscheidung über die zu verwendende Kennzeichnung getroffen.
- KI-unterstützt: KI-basierte Werkzeuge werden verwendet, um Daten zu erkennen und zu annotieren oder um ähnliche Daten in sog. Cluster zu gruppieren. Die Ergebnisse werden dann im Rahmen eines zweistufigen Prozesses von einem Menschen bestätigt oder eventuell ergänzt (z. B. durch Änderung der Begrenzungsrahmen).
- Hybrid: Es könnte eine Kombination der oben genannten Kennzeichnungsansätze verwendet werden. So wird beispielsweise die Crowdsourced Kennzeichnung in der Regel von einer externen Organisation verwaltet, die Zugang zu spezialisierten KI-basierten Crowd-Management-Tools hat.

Gegebenenfalls ist es möglich, einen bereits gekennzeichneten Datensatz wiederzuverwenden, wodurch die Notwendigkeit der Datenkennzeichnung gänzlich entfällt. Viele solcher Datensätze sind öffentlich verfügbar, zum Beispiel bei Kaggle [R16].

4.5.2 Falsch gekennzeichnete Daten in Datensätzen

Beim überwachten Lernen wird davon ausgegangen, dass die Daten von den Datenannotatoren korrekt gekennzeichnet wurden. In der Praxis ist es jedoch selten der Fall, dass alle Elemente in einem Datensatz korrekt gekennzeichnet sind. Daten können aus den folgenden Gründen falsch gekennzeichnet werden:

- Zufällige Fehlhandlungen können von Datenannotatoren gemacht werden (z. B. das Drücken einer falschen Taste).
- Systematische Fehlhandlungen können vollzogen werden (z. B. wenn die Datenannotatoren falsche Anweisungen erhalten oder schlecht ausgebildet sind).
- Vorsätzliche Fehlhandlungen können von böswilligen Datenannotatoren gemacht werden.

- Übersetzungsfehler können dazu führen, dass in einer Sprache korrekt gekennzeichnete Daten in einer anderen falsch gekennzeichnet werden.
- Ist die Auswahl offen für Interpretationen, können die subjektiven Einschätzungen der Datenannotatoren zu widersprüchlichen Datenkennzeichnungen durch verschiedene Datenannotatoren führen.
- Das Fehlen der erforderlichen Fachkenntnisse kann zu einer falschen Kennzeichnung führen.
- Komplexe Klassifikationsaufgaben können dazu führen, dass mehr Fehlhandlungen gemacht werden.
- Die zur Unterstützung der Datenkennzeichnung verwendeten Werkzeuge weisen Fehlerzustände auf, die zu falschen Kennzeichnungen führen.
- ML-basierte Ansätze zur Kennzeichnung sind probabilistisch, und dies kann zu einigen falschen Kennzeichnungen führen.

5 Funktionale Leistungsmetriken von ML - 120 Minuten

Schlüsselwörter

Keine

KI-spezifische Schlüsselwörter

Genauigkeit (*accuracy*), Fläche unter der Kurve (*area under curve*, AUC), Konfusionsmatrix, F1-Wert, Clusterübergreifende Metriken, Clusterinterne Metriken, mittlerer quadratischer Fehler (MQF, *mean square error*, MSE), ML-Benchmark-Suites, Funktionale Leistungsmetriken von ML, Präzision, Sensitivität (*recall*), Betriebskennlinie des Beobachters (*receiver operating characteristic*, ROC), Regressionsmodell, R-Quadrat, Silhouettenkoeffizient

Lernziele für Kapitel 5:

5.1 Konfusionsmatrix

AI-5.1.1 K3 Berechnen der funktionalen Leistungsmetriken von ML aus einem gegebenen Satz von Konfusionsmatrix-Daten

5.2 Zusätzliche funktionale Leistungsmetriken von ML für Klassifikation, Regression und Clusterbildung

AI-5.2.1 K2 Gegenüberstellen und Vergleichen der Konzepte hinter den funktionalen Leistungsmetriken von ML für Klassifikations-, Regressions- und Clusterbildungs-Methoden

5.3 Beschränkungen der funktionalen Leistungsmetriken von ML

AI-5.3.1 K2 Zusammenfassen der Grenzen der Verwendung von funktionalen Leistungsmetriken von ML zur Bestimmung der Qualität des ML-Systems

5.4 Auswahl von funktionalen Leistungsmetriken von ML

AI-5.4.1 K4 Auswählen geeigneter funktionaler Leistungsmetriken von ML und/oder ihrer Werte für ein bestimmtes ML-Modell und Szenario

HO-5.4.1 H2 Evaluieren des erstellten ML-Modells anhand ausgewählter funktionaler Leistungsmetriken von ML

5.5 Benchmark-Suiten für ML

AI-5.5.1 K2 Erläutern der Verwendung von Benchmark-Suiten im Zusammenhang mit ML

5.1 Konfusionsmatrix

Bei einem Klassifikationsproblem wird ein Modell die Ergebnisse nur selten immer richtig vorhersagen. Für ein solches Problem kann eine Konfusionsmatrix mit den folgenden Möglichkeiten erstellt werden:

		Tatsächlich	
		Positiv	Negativ
Vorhergesagt	Positiv	Richtig Positiv (RP)	Falsch Positiv (FP)
	Negativ	Falsch Negativ (FN)	Richtig Negativ (RN)

Abbildung 2: Konfusionsmatrix

Zu beachten ist, dass die in Abbildung 2 dargestellte Konfusionsmatrix unterschiedlich dargestellt werden kann, aber immer Werte für die vier möglichen Situationen von richtig positiv (RP), richtig negativ (RN), falsch positiv (FP) und falsch negativ (FN) erzeugt.

Auf der Grundlage der Konfusionsmatrix werden die folgenden funktionalen Leistungsmetriken von ML definiert:

- Genauigkeit

  Genauigkeit = (RP + RN) / (RP + RN + FP + FN) * 100%

  Die Genauigkeit misst den Prozentsatz aller richtig vorhergesagten Klassifikationen.

- Präzision

  Präzision = RP / (RP + FP) * 100%

  Die Präzision misst den Anteil der vorhergesagten positiven Ergebnisse, die richtig vorhergesagt wurden. Sie ist ein Maß dafür, wie sicher man sich bei positiven Vorhersagen sein kann.

- Sensitivität

  Sensitivität = RP / (RP + FN) * 100%

  Die Sensitivität (auch als richtig-positiv Rate (RPR) oder Recall bezeichnet) misst den Anteil der tatsächlich positiven Fälle, die richtig vorhergesagt wurden. Sie ist ein Maß dafür, wie sicher man sein kann, keine positiven Fälle zu übersehen.

- F1-Wert

  F1-Wert = 2 * (Präzision * Sensitivität) / (Präzision + Sensitivität)

  Der F1-Wert wird als harmonisches Mittel aus Präzision und Sensitivität berechnet. Er hat einen Wert zwischen Null und Eins. Ein Wert nahe bei Eins zeigt an, dass falsche Daten einen geringen Einfluss auf das Ergebnis haben. Ein niedriger F1-Score deutet darauf hin, dass das Modell bei der Erkennung positiver Daten schlecht abschneidet.

5.2 Zusätzliche funktionale Leistungsmetriken von ML für Klassifikation, Regression und Clusterbildung

Es gibt zahlreiche Metriken für verschiedene Arten von ML-Problemen (zusätzlich zu den in Abschnitt 5.1 beschriebenen Metriken für die Klassifikation). Einige der am häufigsten verwendeten Metriken werden im Folgenden beschrieben.

Metriken für überwachte Klassifikation

- Die Betriebskennlinie des Beobachters (*receiver operating characteristic curve*, ROC-Kurve) ist eine grafische Darstellung, welche die Fähigkeit eines binären Klassifikators veranschaulicht, wenn seine Unterscheidungsschwelle variiert wird. Die Methode wurde ursprünglich für militärische Radargeräte entwickelt, woher ihre Bezeichung stammt. Für die ROC-Kurve wird die Sensitivität bzw. richtig positiv Rate (RPR) auf der y-Achse gegen die falsch-positiv Rate (FPR = FP / (RN + FP)) auf der x-Achse aufgetragen.
- Der Flächeninhalt unter der ROC-Kurve wird als “Fläche unter der Kurve” (AUC) bezeichnet. Die AUC stellt den Grad der Trennbarkeit eines Klassifikators dar und zeigt, wie gut das Modell zwischen den Klassen unterscheidet. Je näher die AUC an eins ist, desto besser sind die Vorhersagen des Modells.

Metriken der überwachten Regression

Bei überwachten Regressionsmodellen geben die Metriken an, wie gut die Regressionslinie zu den tatsächlichen Datenpunkten passt.

- Der mittlere quadratische Fehler (MQF, *mean square error* MSE) ist der Durchschnitt der quadrierten Differenzen zwischen dem tatsächlichen Wert und dem vorhergesagten Wert. Der Wert des MQF ist immer positiv, und ein Wert, der näher bei Null liegt, deutet auf ein besseres Regressionsmodell hin. Durch die Quadrierung der einzelnen Differenzen wird sichergestellt, dass sich positive und negative Fehler nicht gegenseitig aufheben.
- Die Metrik R-Quadrat (auch als Bestimmtheitsmaß bezeichnet) beurteilt, wie gut das Regressionsmodell an die abhängigen Variablen angepasst ist.

Metriken für unüberwachte Clusterbildung

Für die unüberwachte Clusterbildung gibt es mehrere Metriken, welche die Abstände zwischen den verschiedenen Clustern und die Nähe der Datenpunkte innerhalb eines bestimmten Clusters darstellen.

- Clusterinterne Metriken messen die Ähnlichkeit von Datenpunkten innerhalb eines Clusters.
- Clusterübergreifende Metriken messen die Ähnlichkeit von Datenpunkten in verschiedenen Clustern.
- Der Silhouettenkoeffizient (auch: Silhouettenwert) ist ein Maß (zwischen -1 und +1), das auf den durchschnittlichen clusterinternen und clusterübergreifenden Abständen basiert. Ein Wert von +1 bedeutet, dass die Cluster gut voneinander separiert sind, ein Wert von Null bedeutet eine zufällige Clusterbildung, und ein Wert von -1 bedeutet, dass die Cluster falsch zugeordnet sind.

5.3 Beschränkungen der funktionalen Leistungsmetriken von ML

Funktionale Leistungsmetriken von ML beschränken sich auf die Messung der Funktionalität des Modells, z. B. in Form von Genauigkeit, Präzision, Sensitivität, MQF, AUC und dem Silhouettenkoeffizienten. Sie messen keine anderen nicht-funktionalen Qualitätsmerkmale, wie die in ISO 25010[S06] definierten (z. B. Performanz) und die in Kapitel 2 beschriebenen (z. B. Erklärbarkeit, Flexibilität und Autonomie). In diesem Lehrplan wird der Begriff "Funktionale Leistungsmetriken von ML" verwendet, da der Begriff “Leistungsmetriken“ ("performance metrics") für diese funktionalen

Metriken weit verbreitet ist. Die Erweiterung zu "Funktionale Leistungsmetrik von ML" unterstreicht, dass diese Metriken spezifisch für das maschinelle Lernen sind und keine Beziehung z.B. zu Performanzmetriken im Sinne von Lastverhalten haben.

Die funktionalen Leistungsmetriken von ML werden durch mehrere Faktoren eingeschränkt:

- Beim überwachten Lernen werden die funktionalen Leistungsmetriken von ML auf der Grundlage von gekennzeichneten Daten berechnet, und die Genauigkeit der resultierenden Metriken hängt von der korrekten Kennzeichnung ab (siehe Abschnitt 4.5).
- Die für die Messung verwendeten Daten sind möglicherweise nicht repräsentativ (z. B. können sie verzerrt sein), und die generierten funktionalen Leistungsmetriken von ML hängen von diesen Daten ab (siehe Abschnitt 2.4).
- Das System kann aus mehreren Komponenten bestehen, aber funktionale Leistungsmetriken von ML gelten nur für das ML-Modell. So wird beispielsweise die Datenpipeline bei der Evaluierung des Modells durch die funktionalen Leistungsmetriken von ML nicht berücksichtigt.
- Die meisten funktionalen Leistungsmetriken von ML können nur mit Hilfe von Werkzeugen gemessen werden.

5.4 Auswahl funktionaler Leistungsmetriken von ML

Normalerweise ist es nicht möglich, ein ML-Modell zu erstellen, das den höchsten Wert für alle aus einer Konfusionsmatrix generierten funktionalen Leistungsmetriken von ML erreicht. Stattdessen werden die am besten geeigneten funktionalen Leistungsmetriken von ML als Akzeptanzkriterien auf der Grundlage der erwarteten Verwendung des Modells ausgewählt (z. B. ist zur Minimierung von falsch positiven Ergebnissen ein hoher Wert der Präzision erforderlich, während zur Minimierung von falsch negativen Ergebnissen die Sensitivität hoch sein sollte). Die folgenden Kriterien können bei der Auswahl der in den Abschnitten 5.1 und 5.2 beschriebenen funktionalen Leistungsmetriken von ML verwendet werden:

- Genauigkeit: Diese Metrik ist voraussichtlich anwendbar, wenn die Datensätze symmetrisch sind (z. B. die Anzahl der falsch positiven und falsch negativen Ergebnisse und Kosten sind ähnlich). Diese Metrik ist ungeeignet, wenn eine Datenklasse gegenüber den anderen dominiert. In diesem Fall sollte der F1-Wert herangezogen werden.
- Präzision: Dies kann eine geeignete Metrik sein, wenn die Kosten für falsch positive Ergebnisse hoch sind und das Vertrauen in positive Ergebnisse hoch sein muss. Ein Spam-Filter (bei dem die Klassifikation einer E-Mail als Spam als positiv angesehen wird) ist ein Beispiel, bei dem eine hohe Präzision erforderlich ist, da es für die meisten Benutzer nicht akzeptabel ist, wenn zu viele E-Mails im Spam-Ordner landen, die eigentlich kein Spam sind. Wenn der Klassifikator mit Situationen zu tun hat, in denen ein sehr großer Prozentsatz der Fälle positiv ist, ist die Verwendung der Präzision allein wahrscheinlich keine gute Wahl.
- Sensitivität: Wenn es von entscheidender Bedeutung ist, dass positive Ergebnisse nicht übersehen werden, ist ein hoher Wert der Sensitivität wichtig. So wäre es beispielsweise inakzeptabel, bei der Krebserkennung richtig-positive Ergebnisse zu übersehen und sie als negativ zu markieren (d. h. es wurde kein Krebs entdeckt).
- F1-Wert - Der F1-Wert ist am nützlichsten, wenn es ein Ungleichgewicht der erwarteten Klassen gibt und wenn die Präzision und die Sensititivtät von ähnlicher Wichtigkeit sind.

Zusätzlich zu den oben genannten Metriken werden in Abschnitt 5.2 beschrieben. Diese können z. B. für bestimmte ML-Probleme anwendbar sein:

- Die AUC für die ROC-Kurve kann für überwachte Klassifikationsprobleme verwendet werden.
- MQF und R-Quadrat können für überwachte Regressionsprobleme verwendet.

- Clusterübergreifende Metriken, clusterinterne Metriken und der Silhouettenkoeffizient können für unüberwachte Clusterbildungs-Probleme verwendet werden.

5.4.1 Praktische Übung: Evaluieren eines erstellten ML-Modells

Berechnen Sie mit dem in der vorherigen Übung trainierten Klassifikationsmodell die Werte für Genauigkeit, Präzision, Sensitivität und F1-Wert und notieren Sie die Ergebnisse.

Verwenden Sie gegebenenfalls die Bibliotheksfunktionen, die von Ihrem Entwicklungs-Framework bereitgestellt werden, um die Berechnungen durchzuführen.

5.5 Benchmark-Suiten für ML

Neue KI-Techniken wie neue Datensätze, Algorithmen, Modelle und Hardware erscheinen regelmäßig, und es kann schwierig sein, die relative Wirksamkeit jeder neuen Technike zu bestimmen.

Um objektive Vergleiche zwischen diesen verschiedenen Techniken zu ermöglichen, gibt es branchenübliche ML-Benchmark-Suiten. Diese decken ein breites Spektrum von Anwendungsbereichen ab und bieten Werkzeuge zur Bewertung von Hardware-Plattformen, Software-Frameworks und Cloud-Plattformen für KI- und ML-Leistung.

ML-Benchmark-Suites können verschiedene Messwerte liefern, darunter Trainingszeiten (z. B. wie schnell ein Framework ein ML-Modell mit einem definierten Trainingsdatensatz bis zu einer bestimmten Zielqualitätsmetrik, wie 75 % Genauigkeit, trainieren kann) und Inferenzzeiten (z. B. wie schnell ein trainiertes ML-Modell Inferenzen durchführen kann).

ML-Benchmark-Suiten werden von verschiedenen Organisationen angeboten, z. B. von

- MLCommons [R18]: Hierbei handelt es sich um eine 2020 gegründete gemeinnützige Organisation, die zuvor ML Perf hieß und Benchmarks für Software-Frameworks, KI-spezifische Prozessoren und ML-Cloud-Plattformen bereitstellt.
- DAWNBench [R19]: Dies ist eine ML-Benchmark-Suite der Stanford University.
- MLMark [R20]: Hierbei handelt es sich um eine ML-Benchmark-Suite zur Messung der Leistung und Genauigkeit eingebetteter Inferenzen des Embedded Microprocessor Benchmark Consortium.

6 ML - Neuronale Netzwerke und Testen - 65 Minuten

Schlüsselwörter

Keine

KI-spezifische Schlüsselwörter

Aktivierungswert, tiefes neuronales Netzwerk (*deep neural network*, DNN), mehrschichtiges Perzeptron, neuronales Netzwerk, Neuronenüberdeckung, Perzeptron, Vorzeichenwechselüberdeckung, Vorzeichen-Vorzeichen-Überdeckung, überwachtes Lernen, Schwellenwertüberdeckung, Trainingsdatensatz, Wertänderungsüberdeckung

Lernziele für Kapitel 6:

6.1 Neuronale Netzwerke

AI-6.1.1 K2 Erläutern der Struktur und Funktion eines neuronalen Netzwerkes, einschließlich eines DNN

HO-6.1.1 H1 Beobachten der Implementierung eines Perzeptrons

6.2 Überdeckungsmaße für neuronale Netzwerke

AI-6.2.1 K2 Beschreiben der verschiedenen Überdeckungsmaße für neuronale Netzwerke

6.1 Neuronale Netzwerke

Künstliche neuronale Netzwerke sollten ursprünglich die Funktionsweise des menschlichen Gehirns imitieren, das man sich als eine Vielzahl miteinander verbundener biologischer Neuronen vorstellen kann. Das einschichtige Perzeptron ist eines der ersten Beispiele für die Implementierung eines künstlichen neuronalen Netzwerkes und umfasst ein neuronales Netzwerk mit nur einer Schicht (d. h. einem einzigen Neuron). Es kann zum überwachten Lernen von Klassifikatoren verwendet werden, die entscheiden, ob eine Eingabe zu einer bestimmten Klasse gehört oder nicht.

Die meisten aktuellen neuronalen Netzwerke gelten als tiefe neuronale Netzwerke, da sie mehrere Schichten haben und als mehrschichtige Perzeptrons betrachtet werden können (siehe Abbildung 3).

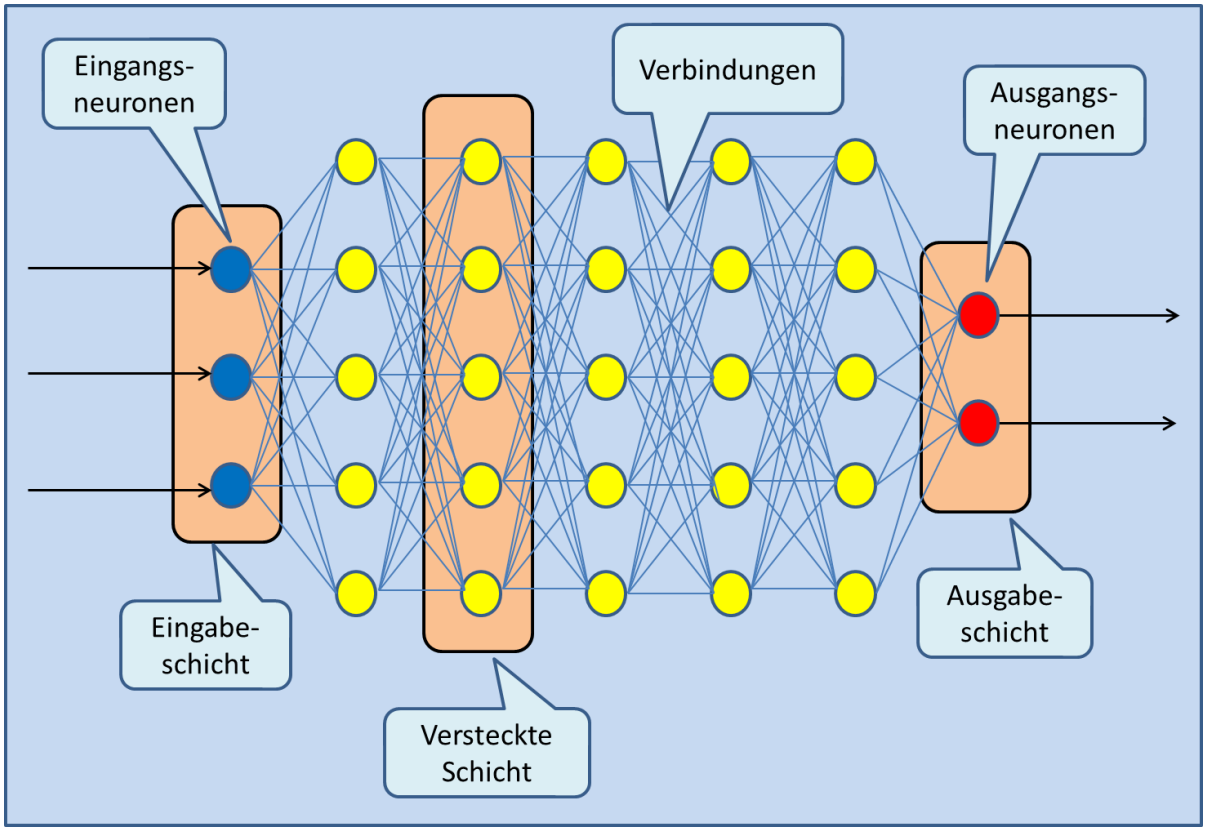

Abbildung 3: Struktur eines tiefen neuronalen Netzwerkes

Ein tiefes neuronales Netzwerk besteht aus drei Arten von Schichten. Die Eingabeschicht empfängt Eingaben, zum Beispiel Pixelwerte von einer Kamera. Die Ausgabeschicht liefert Ergebnisse an die Außenwelt. Dabei kann es sich beispielsweise um einen Wert handeln, der die Wahrscheinlichkeit angibt, dass es sich bei dem Eingabebild um eine Katze handelt.

Zwischen der Eingabe- und der Ausgabeschicht befinden sich versteckte Schichten, die aus künstlichen Neuronen bestehen, die auch als Knoten bezeichnet werden. Die Neuronen in einer Schicht sind mit den Neuronen in der nächsten Schicht verbunden, wobei die Anzahl der Neuronen in jeder aufeinander folgenden Schicht unterschiedlich sein kann. Die Neuronen führen Berechnungen durch und leiten Informationen über das Netzwerk von den Eingangsneuronen zu den Ausgangsneuronen weiter.

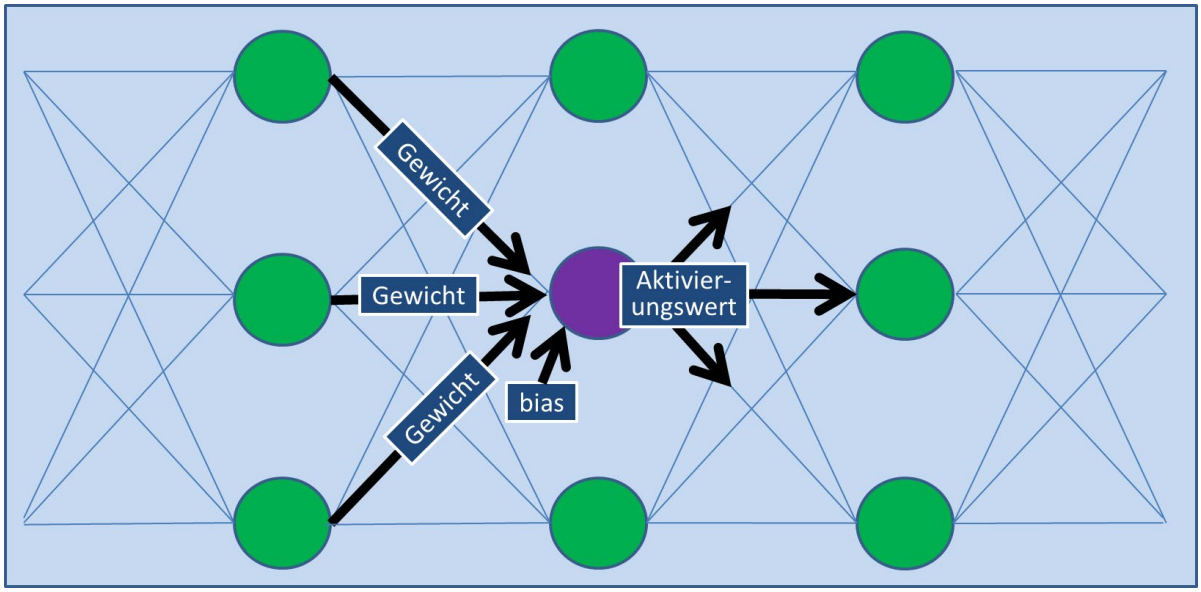

Abbildung 4: Von jedem Neuron durchgeführte Berechnungen

Wie in Abbildung 4 dargestellt, erzeugt die von jedem Neuron (mit Ausnahme der Neuronen in der Eingabeschicht) durchgeführte Berechnung den so genannten Aktivierungswert. Dieser Wert wird berechnet, indem eine Formel (die Aktivierungsfunktion) ausgeführt wird. Diese erhält als Eingaben die Aktivierungswerte aller Neuronen in der vorherigen Schicht, die den Verbindungen zwischen den Neuronen zugewiesenen Gewichte (diese Gewichte ändern sich, wenn das Netzwerk lernt) und den individuellen sog. Bias jedes Neurons.

Zu beachten ist, dass der Bias eines Neurons ein voreingestellter konstanter Wert ist und nichts mit dem Begriff Verzerrung (Englisch: *bias*) zu tun hat. Die Ausführung verschiedener Aktivierungsfunktionen kann dazu führen, dass unterschiedliche Aktivierungswerte berechnet werden. Diese Werte sind in der Regel um Null zentriert und haben einen Bereich zwischen -1 (was bedeutet, dass das Neuron "desinteressiert" ist) und +1 (was bedeutet, dass das Neuron "sehr interessiert" ist).

Beim Training des neuronalen Netzwerkes wird jedes Neuron auf einen Bias-Wert voreingestellt, und die Trainingsdaten werden durch das Netzwerk geleitet, wobei jedes Neuron die Aktivierungsfunktion ausführt, um schließlich eine Ausgabe zu erzeugen. Die erzeugte Ausgabe wird dann mit dem bekannten korrekten Ergebnis verglichen (in diesem Beispiel des überwachten Lernens werden gekennzeichnete Daten verwendet). Die Differenz zwischen der tatsächlichen Ausgabe und dem bekannten korrekten Ergebnis wird dann durch das Netzwerk zurückgeführt, um die Werte der Gewichte an den Verbindungen zwischen den Neuronen zu ändern, um diese Differenz zu minimieren. Je mehr Trainingsdaten in das Netzwerk eingespeist werden, desto häufiger werden die Gewichte angepasst, während das Netzwerk lernt. Letztendlich werden die erzeugten Ergebnisse als gut genug angesehen, um das Training zu beenden.

6.1.1 Praktische Übung: Implementierung eines einfachen Perzeptrons

Die Lernenden werden durch eine Übung geführt, die zeigt, wie ein Perzeptron eine einfache Funktion lernt, z. B. eine UND-Funktion.

Die Übung sollte zeigen, wie ein Perzeptron lernt, indem es die Gewichte über eine Reihe von Epochen hinweg ändert, bis der Fehler gleich Null ist. Für diese Aufgabe können verschiedene Verfahren verwendet werden (z. B. Tabellenkalkulation, Simulation).

6.2 Überdeckungsmaße für neuronale Netzwerke

Das Erreichen von White-Box-Überdeckungskriterien (z. B. Anweisungs-, Verzweigungs-, modifizierte Bedingungs-/Entscheidungsüberdeckung (MC/DC) [I01]) ist für die Einhaltung einiger für die funktionale Sicherheit relevanter Standards [S07] bei der Verwendung von herkömmlichen imperativen Programmiersprachen obligatorisch. Dieses wird von vielen Testern auch für andere kritische Anwendungen empfohlen. Die Überwachung und Verbesserung der Überdeckung unterstützt den Entwurf neuer Testfälle, was zu einem erhöhten Vertrauen in das Testobjekt führt.

Die Verwendung solcher Maßstäbe zur Messung der Überdeckung eines neuronalen Netzwerkes ist jedoch wenig sinnvoll, da bei jeder Ausführung des neuronalen Netzwerkes in der Regel der gleiche Code ausgeführt wird. Stattdessen wurden Überdeckungsmaße vorgeschlagen, die auf der Überdeckung der Struktur des neuronalen Netzwerkes selbst basieren, genauer gesagt, auf den Neuronen darin. Die meisten dieser Maße beruhen auf den Aktivierungswerten der Neuronen.

Die Überdeckung für neuronale Netzwerke ist ein neues Forschungsgebiet. Akademische Arbeiten wurden erst 2017 veröffentlicht, und daher gibt es nur wenig objektive Evidenz (z. B. doppelte Forschungsergebnisse), die zeigen, dass die vorgeschlagenen Maßnahmen wirksam sind. Allerdings gibt es trotz der Tatsache, dass Anweisungs- und Entscheidungsüberdeckung seit über 50 Jahren verwendet werden, ebenfalls wenig objektive Evidenz für ihre relative Effektivität, obwohl sie für die Messung der Überdeckung von Software in sicherheitsrelevanten Anwendungen, wie medizinischen Geräten und Avioniksystemen, vorgeschrieben worden sind.

Die folgenden Überdeckungskriterien für neuronale Netzwerke wurden von Forschern für eine Vielzahl von Anwendungen vorgeschlagen und angewendet:

- Neuronenüberdeckung: Eine vollständige Neuronenüberdeckung erfordert, dass jedes Neuron im neuronalen Netzwerk einen Aktivierungswert größer als Null erreicht [B12]. Dies ist in der Praxis sehr leicht zu erreichen, und Untersuchungen haben gezeigt, dass bei einer Vielzahl von tiefen neuronalen Netzwerken mit sehr wenigen Testfällen eine fast 100%ige Neuronenüberdeckung erreicht wird. Das Nichterreichen dieses Überdeckungsmaßes kann als Alarmsignal sehr nützlich sein.
- Schwellenwertüberdeckung: Eine vollständige Schwellenwertüberdeckung setzt voraus, dass jedes Neuron im neuronalen Netzwerk einen Aktivierungswert erreicht, der größer als ein bestimmter Schwellenwert ist. Die Forscher, die das DeepXplore-Framework entwickelt haben, schlugen vor, die Neuronenüberdeckung anhand des Aktivierungswertes zu messen, sobald dieser in mindestens einem Testfall einen je nach Situation geänderten Schwellenwert überschreitet. Sie führten ihre Forschungen mit einem Schwellenwert von 0,75 durch und berichteten, dass sie mit diesem White-Box-Ansatz effizient Tausende von falschen Grenzfall-Verhaltensweisen fanden. Diese Art der Neuronenüberdeckung mit änderbarem Schwellenwert wurde hier umbenannt, um sie leichter von der Neuronenüberdeckung mit dem Schwellenwert Null zu unterscheiden, für welche einige Forscher den Begriff "Neuronenüberdeckung" verwenden.
- Vorzeichenwechselüberdeckung: Um eine vollständige Vorzeichenwechselüberdeckung zu erreichen, müssen die Testfälle jedes Neuron dazu bringen, sowohl positive als auch negative Aktivierungswerte zu erreichen [B13].
- Wertänderungsüberdeckung: Um eine vollständige Wertänderungsüberdeckung zu erreichen, müssen die Testfälle jedes Neuron veranlassen, zwei Aktivierungswerte zu erreichen, wobei die Differenz zwischen den beiden Werten einen bestimmten Wert überschreitet [B13].
- Vorzeichen-Vorzeichen-Überdeckung: Diese Überdeckung berücksichtigt Neuronenpaare in benachbarten Schichten und das Vorzeichen ihrer Aktivierungswerte. Damit ein Neuronenpaar als abgedeckt gilt, muss ein Testfall zeigen, dass die Änderung des Vorzeichens eines Neurons in der ersten Schicht dazu führt, dass das Neuron in der zweiten Schicht sein Vorzeichen ändert, während die Vorzeichen aller anderen Neuronen in der zweiten Schicht unverändert

bleiben [B13]. Dies ist ein ähnliches Konzept wie die MC/DC-Überdeckung für imperative Programmiersprachen.

Forscher haben über weitere Überdeckungsmaße berichtet, die auf Schichten basieren (obwohl sie einfacher sind als die Vorzeichen-Vorzeichen-Überdeckung). Ein erfolgreicher Ansatz, der Nächste-Nachbar-Algorithmen verwendet, um sinnvolle Änderungen in benachbarten Neuronengruppen zu identifizieren, wurde in das TensorFuzz-Tool implementiert [B14].

7 Testen KI-basierter Systeme im Überblick - 115 Minuten

Schlüsselwörter

Eingabedatentest, Testen von ML-Modellen

KI-spezifische Schlüsselwörter

KI-Komponente, Automatisierungsverzerrung, Big Data, Konzeptdrift, Datenpipeline, funktionale Leistungsmetriken von ML, Trainingsdaten

Lernziele für Kapitel 7:

7.1 Spezifikation KI-basierter Systeme

AI-7.1.1 K2 Erklären, wie Systemspezifikationen für KI-basierte Systeme zu Herausforderungen beim Testen führen können

7.2 Teststufen für KI-basierte Systeme

AI-7.2.1 K2 Beschreiben, wie KI-basierte Systeme auf jeder Teststufe getestet werden

7.3 Testdaten zum Testen KI-basierter Systeme

AI-7.3.1 K1 Erinnern der Faktoren im Zusammenhang mit Testdaten, die das Testen KI-basierter Systeme erschweren können

7.4 Testen auf Automatisierungsverzerrung in KI-basierten Systemen

AI-7.4.1 K2 Erläutern der Automatisierungsverzerrung und wie sich diese auf das Testen auswirkt

7.5 Dokumentation einer KI-Komponente

AI-7.5.1 K2 Beschreiben der Dokumentation einer KI-Komponente und verstehen, wie die Dokumentation das Testen KI-basierter Systeme unterstützt

7.6 Testen auf Konzeptdrift

AI-7.6.1 K2 Erläutern der Notwendigkeit, das trainierte Modell häufig zu testen, um Konzeptdrift zu vermeiden

7.7 Auswahl einer Testvorgehensweise für ein ML-System

AI-7.7.1 K4 Analysieren eines bestimmten Szenarios und bestimmen einer Testvorgehensweise, die bei der Entwicklung des ML-Systems befolgt werden sollte

7.1 Spezifikation KI-basierter Systeme

Systemanforderungen und Entwurfsspezifikationen sind sowohl für KI-basierte Systeme als auch für herkömmliche Systeme gleichermaßen wichtig. Diese Spezifikationen bilden die Grundlage für Tester, um zu prüfen, ob das tatsächliche Systemverhalten mit den spezifizierten Anforderungen übereinstimmt. Wenn die Spezifikationen jedoch unvollständig und nicht testbar sind, entsteht ein Testorakelproblem (siehe Abschnitt 8.7).

Es gibt mehrere Gründe, warum die Spezifikation KI-basierter Systeme eine besondere Herausforderung darstellen kann:

- In vielen Projekten zur Entwicklung KI-basierter Systeme werden die Anforderungen nur in Form von hochrangigen Geschäftszielen und erforderlichen Vorhersagen spezifiziert. Ein Grund dafür ist der explorative Charakter der KI-basierten Systementwicklung. Häufig beginnen KI-basierte Systemprojekte mit einem Datensatz, und das Ziel besteht darin, zu ermitteln, welche Vorhersagen aus diesen Daten gewonnen werden können. Im Gegensatz dazu wird in herkömmlichen Projekten die erforderliche Logik typischerweise zu Beginn spezifiziert.
- Die Genauigkeit eines KI-basierten Systems ist oft nicht bekannt, bis die Testergebnisse aus unabhängigen Tests vorliegen. Zusammen mit dem explorativen Entwicklungsansatz führt dies häufig zu unzureichenden Spezifikationen, da die Implementierung bereits im Gange ist, wenn die gewünschten Akzeptanzkriterien festgelegt werden.
- Der probabilistische Charakter vieler KI-basierter Systeme kann es erforderlich machen, Toleranzen für einige der erwarteten Qualitätsanforderungen festzulegen, z. B. für die Genauigkeit der Vorhersagen.
- Wenn die Ziele des Systems die Nachahmung menschlichen Verhaltens und nicht die Bereitstellung spezifischer Funktionen erfordern, führt dies häufig zu schlecht spezifizierten Verhaltensanforderungen, die darauf beruhen, dass das System genauso gut oder besser als die menschlichen Aktivitäten ist, die es ersetzen soll. Dies kann die Definition eines Testorakels erschweren, insbesondere dann, wenn die Menschen, die es ersetzen soll, in ihren Fähigkeiten stark variieren.
- Wenn KI bei der Implementierung von Benutzungsschnittstellen eingesetzt wird, z. B. durch Erkennung natürlicher Sprache, Computer Vision oder physische Interaktion mit Menschen, müssen die Systeme eine größere Flexibilität aufweisen. Eine solche Flexibilität kann jedoch auch zu Herausforderungen bei der Identifizierung und Dokumentation all der verschiedenen Arten führen, in denen solche Interaktionen stattfinden können.
- Qualitätsmerkmale für KI-basierte Systeme wie Anpassbarkeit, Flexibilität, Evolution und Autonomie, müssen im Rahmen der Anforderungsspezifikation berücksichtigt und definiert werden (siehe Kapitel 2). Aufgrund der Neuartigkeit dieser Merkmale kann es schwierig sein, sie zu definieren und zu testen.

7.2 Teststufen für KI-basierte Systeme

KI-basierte Systeme bestehen in der Regel sowohl aus KI- als auch aus Nicht-KI-Komponenten. Nicht-KI-Komponenten können mit herkömmlichen Ansätzen getestet werden [I01], während KI-Komponenten und Systeme, die KI-Komponenten enthalten, in mancher Hinsicht anders getestet werden müssen, wie im Folgenden beschrieben. Für alle Teststufen, die das Testen von KI-Komponenten beinhalten, ist es wichtig, das Testen eng durch Dateningenieure/-wissenschaftler und Domänenexperten zu unterstützen.

Ein wesentlicher Unterschied zu den Teststufen für konventionelle Software besteht in der Aufnahme von zwei neuen, spezialisierten Teststufen, die explizit das Testen der Eingabedaten und der in KI-basierten Systemen verwendeten Modelle behandeln [B15]. Der größte Teil dieses Abschnitts ist auf alle KI-basierten Systeme anwendbar, obwohl einige Teile speziell auf ML ausgelegt sind.

7.2.1 Eingabedatentest

Mit dem Eingabedatentest soll sichergestellt werden, dass die vom System für das Training, die Evaluierung, den Test und den Einsatz verwendeten Daten von höchster Qualität sind (siehe Abschnitt 4.3). Dazu gehören die folgenden Punkte:

- Reviews
- Statistische Techniken (z. B. Testen der Daten auf Verzerrungen)
- Explorative Datenanalyse (*explorative data analysis*, EDA) der Trainingsdaten
- Statische und dynamische Tests der Datenpipeline

Die Datenpipeline umfasst in der Regel mehrere Komponenten zur Datenvorbereitung (siehe Abschnitt 4.1), und der Test dieser Komponenten umfasst sowohl Komponententests als auch Integrationstests. Die Datenpipeline für das Training kann sich deutlich von der Datenpipeline unterscheiden, die zur Unterstützung der operationellen Vorhersage verwendet wird. Für das Training kann die Datenpipeline als Prototyp betrachtet werden, im Vergleich zu der vollständig entwickelten, automatisierten Version, die im Betrieb eingesetzt wird. Aus diesem Grund kann der Test dieser beiden Versionen der Datenpipeline recht unterschiedlich ausfallen. Der Test der funktionalen Gleichwertigkeit der beiden Versionen sollte jedoch ebenfalls in Betracht gezogen werden.

7.2.2 ML-Modelltest

Ziel des ML-Modelltests ist es, sicherzustellen, dass das ausgewählte Modell alle festgelegten Leistungskriterien erfüllt. Dies schließt ein:

- Funktionale Leistungskriterien von ML (siehe Abschnitte 5.1 und 5.2)
- Nicht-funktionale Abnahmekriterien von ML, die für das Modell an sich angemessen sind, wie z. B. Trainingsgeschwindigkeit, Vorhersagegeschwindigkeit, verwendete Rechenressourcen, Anpassbarkeit und Transparenz .

ML-Modelltests zielen auch darauf ab, festzustellen, dass die Wahl des ML-Frameworks, des Algorithmus, des Modells, der Modelleinstellungen und der Hyperparameter so nahe wie möglich am Optimum liegt. Gegebenenfalls kann das Testen von ML-Modellen auch Tests zur Erreichung von White-Box-Überdeckungskriterien umfassen (siehe Abschnitt 6.2). Das ausgewählte Modell wird später mit anderen Komponenten (KI und Nicht-KI) integriert.

7.2.3 Komponententest

Der Komponententest ist eine konventionelle Teststufe, die für alle Nicht-Modell-Komponenten, wie Benutzungsschnittstellen und Kommunikationskomponenten, anwendbar ist.

7.2.4 Komponenten-Integrationstest

Der Komponenten-Integrationstest ist eine konventionelle Teststufe, mit der sichergestellt werden soll, dass die Systemkomponenten (sowohl KI als auch Nicht-KI) korrekt zusammenwirken. Es wird getestet, dass die Eingaben aus der Datenpipeline wie erwartet vom Modell empfangen werden und dass alle vom Modell erstellten Vorhersagen mit den relevanten Systemkomponenten (z. B. der Benutzungsschnittstelle) ausgetauscht und von diesen korrekt verwendet werden. Wenn KI als Dienst bereitgestellt wird (siehe Abschnitt 1.7), ist es üblich, API-Tests des bereitgestellten Dienstes als Teil der Komponenten-Integrationstests durchzuführen.

7.2.5 Systemtest

Der Systemtest ist eine konventionelle Teststufe, mit der sichergestellt werden soll, dass das Gesamtsystem aus integrierten Komponenten (sowohl KI als auch Nicht-KI) wie erwartet funktioniert. Dies erfolgt sowohl unter funktionalen als auch unter nicht-funktionalen Gesichtspunkten und in einer Testumgebung, die der Einsatzumgebung sehr ähnlich ist. Je nach System können diese Tests in Form von Feldversuchen in der erwarteten Einsatzumgebung oder in einem Simulator durchgeführt werden (z. B. wenn die Testszenarien gefährlich oder in einer Einsatzumgebung schwer zu reproduzieren sind).

Während des Systemtests werden die funktionalen Leistungskriterien von ML erneut getestet, um sicherzustellen, dass die Testergebnisse des ersten ML-Modelltests nicht beeinträchtigt werden, wenn das Modell in ein vollständiges System eingebettet wird. Dieser Test ist besonders wichtig, wenn die KI-Komponente absichtlich verändert wurde (z. B. Komprimierung eines DNN, um seine Größe zu verringern).

Auf der Teststufe Systemtest werden auch viele der nicht-funktionalen Anforderungen an das System getestet. So können z. B. Robustheitstests durchgeführt werden und das System kann auf seine Erklärbarkeit getestet werden. Gegebenenfalls können die Schnittstellen zu Hardwarekomponenten (z. B. Sensoren) im Rahmen des Systemtests getestet werden.

7.2.6 Abnahmetest

Der Abnahmetest ist eine konventionelle Teststufe und dient der Feststellung, ob das Gesamtsystem für den Kunden akzeptabel ist. Bei KI-basierten Systemen kann die Definition von Abnahmekriterien eine Herausforderung darstellen (siehe Abschnitt 8.8). Wird KI als Dienst erbracht (siehe Abschnitt 1.7), kann ein Abnahmetest erforderlich sein, um festzustellen, ob der Dienst für das vorgesehene System geeignet ist und ob z. B. die funktionalen Leistungskriterien von ML ausreichend erfüllt wurden.

7.3 Testdaten zum Testen KI-basierter Systeme

Je nach Situation und dem System unter Test (SUT) kann die Beschaffung von Testdaten eine Herausforderung darstellen. Es gibt mehrere potenzielle Herausforderungen im Umgang mit Testdaten für KI-basierte Systeme, darunter:

- Die Erstellung und Verwaltung großer Datenmengen (Big Data, Daten mit hohem Volumen, hoher Datenrate und großer Vielfalt) kann schwierig sein. So kann es beispielsweise schwierig sein, repräsentative Testdaten für ein System zu erstellen, das große Mengen an Bildern und Audiodaten mit hoher Geschwindigkeit verbraucht.
- Die Eingabedaten müssen sich möglicherweise im Laufe der Zeit ändern, insbesondere wenn sie Ereignisse in der realen Welt darstellen. Zum Beispiel müssen aufgenommene Fotos zum Testen eines Gesichtserkennungssystems möglicherweise "gealtert" werden, um das Altern von Menschen über mehrere Jahre in der realen Welt darzustellen.
- Personenbezogene oder anderweitig vertrauliche Daten erfordern möglicherweise besondere Techniken zur Bereinigung, Verschlüsselung oder Anonymisierung. Auch kann eine rechtliche Genehmigung für die Verwendung erforderlich sein.
- Wenn die Tester dieselbe Implementierung wie die Datenwissenschaftler für die Datenbeschaffung und die Datenvorverarbeitung verwenden, können Fehlerzustände in diesen Schritten maskiert sein.

7.4 Testen auf Automatisierungsverzerrungen in KI-basierten Systemen

Eine Kategorie KI-basierter Systeme hilft dem Menschen bei der Entscheidungsfindung. Gelegentlich neigt der Mensch jedoch dazu, diesen Systemen zu sehr zu vertrauen. Dieses fehlgeleitete Vertrauen kann entweder als Automatisierungs- oder als Bequemlichkeitsverzerrung bezeichnet werden und hat zwei Formen.

- Die erste Form der Automatisierungs-/Bequemlichkeitsverzerrung besteht darin, dass der Mensch die Empfehlungen des Systems akzeptiert und die Eingaben aus anderen Quellen (einschließlich seiner selbst) nicht berücksichtigt. Beispielsweise könnte ein Verfahren, bei dem ein Mensch Daten in ein Formular eingibt, durch maschinelles Lernen verbessert werden, indem das Formular vorausgefüllt wird und der Mensch dann diese Daten validiert. Es hat sich gezeigt, dass diese Form der Automatisierung die Qualität der getroffenen Entscheidungen typischerweise um 5 % verringert, aber je nach Systemkontext kann dieser Wert noch viel höher sein [B16]. Auch die automatische Korrektur von getipptem Text (z. B. in Mobiltelefonnachrichten) ist oft fehlerhaft und kann die Bedeutung verändern. Die Benutzer bemerken dies oft nicht und berichtigen den Fehler nicht.
- Die zweite Form der Automatisierungs-/Bequemlichkeitsverzerrung besteht darin, dass der Mensch einen Systemfehler übersieht, weil er das System nicht angemessen überwacht. So werden beispielsweise teilautonome Fahrzeuge zunehmend selbstfahrend, sind aber immer noch darauf angewiesen, dass ein Mensch im Falle eines drohenden Unfalls das Steuer übernimmt. Typischerweise wird der menschliche Fahrzeuginsasse allmählich zu vertrauensvoll gegenüber den Fähigkeiten des Systems, das Fahrzeug zu steuern, und beginnt, weniger aufmerksam zu sein. Dies kann dazu führen, dass er nicht mehr in der Lage ist, im Bedarfsfall angemessen zu reagieren.

In beiden Szenarien sollten die Tester verstehen, wie die menschliche Entscheidungsfindung beeinträchtigt werden kann, und sowohl die Qualität der Systemempfehlungen als auch die Qualität der entsprechenden menschlichen Eingaben durch repräsentative Benutzer testen.

7.5 Dokumentieren einer KI-Komponente

Zu den typischen Inhalten der Dokumentation einer KI-Komponente gehören:

- Allgemeines: Bezeichner, Beschreibung, Entwicklerangaben, Hardware-Anforderungen, Lizenzangaben, Version, Datum und Kontaktinformationen.
- Entwurf: Annahmen und technische Entscheidungen.
- Nutzung: Primäre und sekundäre Anwendungsfälle, typische Nutzer, Art des Selbstlernens, bekannte Verzerrungen, ethische Aspekte, Aspekte der Sicherheit, Transparenz, Entscheidungsschwellenwerte, Plattform und Konzeptdrift.
- Datensätze: Merkmale, Sammlung, Verfügbarkeit, Anforderungen an die Vorverarbeitung, Verwendung, Inhalt, Kennzeichnung, Größe, Datenschutz, IT-Sicherheit, Verzerrung/Fairness und Einschränkungen/Beschränkungen.
- Testen: Testdatensatz (Beschreibung und Verfügbarkeit), Unabhängigkeit der Tests, Istergebnisse des Tests, Testvorgehensweise für Robustheit, Erklärbarkeit, Konzeptdrift und Übertragbarkeit.
- Training und funktionale Leistung von ML: ML-Algorithmus, Gewichtung, Validierungsdatensatz, Auswahl von funktionalen Leistungsmetriken von ML, Schwellenwerte für die funktionalen Leistungsmetriken von ML und tatsächliche funktionale Leistungsmetriken von ML.

Eine klare Dokumentation trägt dazu bei, die Tests zu verbessern, indem sie Transparenz über die Implementierung des KI-basierten Systems schafft. Die für das Testen wichtigsten Bereiche der Dokumentation sind:

- Der Zweck des Systems und die Spezifikation der funktionalen und nicht-funktionalen Anforderungen. Diese Arten der Dokumentation sind in der Regel Teil der Testbasis.
- Informationen zur Architektur und zum Design, aus denen hervorgeht, wie die verschiedenen KI- und Nicht-KI-Komponenten zusammenwirken. Dies unterstützt die Identifizierung von Integrationstestzielen und kann eine Grundlage für White-Box-Tests der Systemstruktur bilden.
- Die Spezifikation der Einsatzumgebung. Dies ist erforderlich, um die Autonomie, Flexibilität und Anpassbarkeit des Systems zu testen.
- Die Quelle der Eingabedaten, einschließlich der zugehörigen Metadaten. Dies muss bei dem Test der folgenden Aspekte klar verstanden werden:
    - Funktionale Korrektheit von Eingaben aus nicht vertrauenswürdigen Quellen.
    - Explizite oder implizite Stichprobenverzerrung.
    - Flexibilität, einschließlich des Fehl-Lernens bei unzureichenden Dateneingaben für selbstlernende Systeme.
- Die Art und Weise, in der das System an Veränderungen in seiner Einsatzumgebung anpassbar sein soll. Dies wird als Testbasis für den Test der Anpassbarkeit benötigt.
- Angaben zu den erwarteten Systembenutzern. Dies ist erforderlich, um sicherzustellen, dass die Tests repräsentativ durchgeführt werden können.

7.6 Testen auf Konzeptdrift

Die Einsatzumgebung kann sich im Laufe der Zeit verändern, ohne dass sich das trainierte Modell entsprechend ändert. Dieses Phänomen wird als Konzeptdrift bezeichnet und führt in der Regel dazu, dass die Ergebnisse des Modells immer ungenauer und weniger nützlich werden. So können beispielsweise die Auswirkungen von Marketingkampagnen dazu führen, dass sich das Verhalten potenzieller Kunden im Laufe der Zeit ändert. Bei solchen Veränderungen kann es sich um saisonale oder abrupte Veränderungen aufgrund kultureller, moralischer oder gesellschaftlicher Veränderungen handeln, die außerhalb des Systems liegen. Ein Beispiel für eine solche abrupte Veränderung sind die Auswirkungen der COVID-19-Pandemie und ihre Folgen für die Genauigkeit der Modelle für Umsatzprognosen und Aktienmärkte.

Für eine Konzeptdrift anfällige Systeme sollten regelmäßig anhand der vereinbarten funktionalen Leistungskriterien von ML getestet werden, um sicherzustellen, dass Konzeptdrift früh genug erkannt wird. Typische Abhilfemaßnahmen können die Stilllegung des Systems oder ein erneutes Training des Systems beinhalten. Im Falle eines erneuten Trainings würde dieses mit aktuellen Trainingsdaten durchgeführt, gefolgt von Fehlernachtests, Regressionstests und möglicherweise einer Form von A/B-Tests (siehe Abschnitt 9.4), wobei das aktualisierte B-System das ursprüngliche A-System übertreffen muss.

7.7 Auswahl einer Testvorgehensweise für ein ML-System

Ein KI-basiertes System umfasst in der Regel sowohl KI- als auch Nicht-KI-Komponenten. Die Testvorgehensweise basiert auf einer Risikoanalyse für ein solches System und umfasst sowohl konventionelle Tests als auch spezielle Tests, um die für KI-Komponenten und KI-basierte Systeme spezifischen Faktoren zu berücksichtigen.

Die folgende Tabelle 2 listet einige typische Risiken und entsprechende Abmilderungen auf, die speziell für ML-Systeme gelten. Zu beachten ist, dass diese Liste nur eine begrenzte Anzahl von Beispielen

enthält und dass es noch viele weitere Risiken speziell für ML-Systeme gibt, die durch Tests abgemildert werden müssen.

Risiko-Aspekt	Beschreibung und mögliche Abmilderungen
Die Datenqualität ist möglicherweise geringer als erwartet.	Dieses Risiko kann auf verschiedene Arten zu einem Problem werden, die jeweils auf unterschiedliche Art und Weise verhindert werden können (siehe Abschnitt 4.4). Zu den üblichen Abmilderungen gehören die Verwendung von Reviews, EDA und dynamischen Tests.
Die Betriebsdaten-Pipeline kann fehlerhaft sein.	Dieses Risiko kann durch dynamische Tests der einzelnen Pipelinekomponenten und den Integrationstest der gesamten Pipeline teilweise abgemildert werden.
Der ML-Workflow, der zur Entwicklung des Modells verwendet wurde, kann suboptimal sein. (Siehe Abschnitt 3.1)	Risikoquellen könnten sein: • Fehlende Einigung im Vorfeld über den zu befolgenden ML-Workflow. • Eine schlechte Wahl des Arbeitsablaufs. • Dateningenieure, die sich nicht an den Arbeitsablauf halten. Reviews durch Experten können die Gefahr der Wahl des falschen Arbeitsablaufs verringern, während ein eher praktisches Management (hands-on management) oder Audits die Probleme bei der Einigung auf den Arbeitsablauf und dessen Umsetzung analysieren und lösen könnten.
Die Wahl des ML-Frameworks, des Algorithmus, des Modells, der Modelleinstellungen und/oder der Hyperparameter kann suboptimal sein.	Risikoquellen könnten mangelndes Fachwissen der Entscheidungsträger oder auf eine mangelhafte Umsetzung der Evaluierungs- und Tuning-Schritte (oder Testschritte) des ML-Workflows sein. Reviews durch Experten können die Gefahr von Fehlentscheidungen verringern, und ein besseres Management kann sicherstellen, dass die Evaluierungs- und Tuning-Schritte (und Testschritte) des Arbeitsablaufs eingehalten werden.
Die gewünschten funktionalen Leistungskriterien von ML können unter Umständen nicht erfüllt werden, obwohl die ML-Komponente diese Kriterien für sich genommen erfüllt.	Risikoquellen könnten sein, dass die für das Training und den Test des Modells verwendeten Datensätze nicht repräsentativ für die in der Praxis anfallenden Daten sind. Reviews der ausgewählten Datensätze durch Experten (oder Nutzer) können das Risiko verringern, dass die ausgewählten Daten nicht repräsentativ sind.
Die gewünschten funktionalen Leistungskriterien von ML werden erfüllt, aber die Benutzer sind möglicherweise mit den	Risikoquelle könnte die Wahl der falschen funktionalen Leistungskriterien von ML sein (z. B. wurde eine hohe Trefferquote gewählt, obwohl eine hohe Genauigkeit erforderlich gewesen wäre). Reviews mit Experten können das Risiko der Auswahl der falschen funktionalen Leistungsmetriken von ML mindern.

Risiko-Aspekt	Beschreibung und mögliche Abmilderungen
gelieferten Ergebnissen unzufrieden.	Erfahrungsbasierte Tests könnten auch ungeeignete Kriterien identifizieren. Das Risiko könnte auch auf eine Konzeptdrift zurückzuführen sein. In diesem Fall könnte ein häufigeres Testen des operativen Systems das Risiko mindern.
Die gewünschten funktionalen Leistungskriterien von ML werden erfüllt, aber die Nutzer sind möglicherweise mit dem erbrachten Dienst unzufrieden.	Risikoquelle könnte ein mangelnder Fokus auf die nicht-funktionalen Anforderungen des Systems sein. Zu beachten ist, dass die Palette der Qualitätsmerkmale für KI-basierte Systeme über die Qualitätsmerkmale aus ISO/IEC 25010 hinausgeht (siehe Kapitel 2). Die Verwendung eines risikobasierten Ansatzes zur Priorisierung der Qualitätsmerkmale und die Durchführung der entsprechenden nicht-funktionalen Tests könnten dieses Risiko mindern. Alternativ könnte die Risikoquelle eine Kombination von Faktoren sein, die durch erfahrungsbasierte Tests im Systemtest ermittelt werden können. Kapitel 8 enthält Hinweise zum Testen dieser Merkmale.
Das selbstlernende System erbringt möglicherweise nicht die von den Nutzern erwartete Leistung.	Dieses Risiko kann verschiedene Quellen haben, zum Beispiel: • Die vom System zum Selbstlernen verwendeten Daten können ungeeignet sein. In diesem Fall könnte ein Review durch Experten die problematischen Daten identifizieren. • Das System kann versagen, weil die neue selbstgelernte Funktionalität nicht akzeptabel ist. Dies könnte durch automatisierte Regressionstests einschließlich eines Leistungsvergleichs mit der vorherigen Funktionalität abgemildert werden. • Das System kann auf eine Weise lernen, die von den Nutzern nicht erwartet wird, was durch erfahrungsbasierte Tests festgestellt werden könnte.
Die Benutzer können frustriert sein, weil sie nicht verstehen, wie das System seine Entscheidungen trifft.	Risikoquelle könnte ein Mangel an Erklärbarkeit, Interpretierbarkeit und/oder Transparenz sein. Siehe Abschnitt 8.6 Testen der Transparenz, Interpretierbarkeit und Erklärbarkeit KI-basierter Systemefür Einzelheiten zum Testen auf diese Merkmale.
Es kann vorkommen, dass das Modell hervorragende Vorhersagen liefert, wenn die Daten den Trainingsdaten ähnlich sind, aber ansonsten schlechte Ergebnisse liefert.	Risikoquelle kann eine Überanpassung sein (siehe Abschnitt 3.5.1), die durch Testen des Modells mit einem vom Trainingsdatensatz völlig unabhängigen Datensatz oder durch erfahrungsbasiertes Testen aufgedeckt werden kann.

Tabelle 2: Risiko-Aspekte von ML-Systemen

8 Testen KI-spezifischer Qualitätsmerkmale - 150 Minuten

Schlüsselwörter

Testorakel

KI-spezifische Schlüsselwörter

Algorithmische Verzerrung, autonomes System, Autonomie, Expertensystem, Erklärbarkeit, unangemessene Verzerrung, Interpretierbarkeit, LIME-Methode (*Local Interpretable Model-Agnostic Explanations*), ML-Trainingsdaten, nicht-deterministisches System, probabilistisches System, Stichprobenverzerrung, selbstlernendes System, Transparenz

Lernziele für Kapitel 8:

8.1 Herausforderungen beim Testen selbstlernender Systeme

AI-8.1.1 K2 Erläutern der Herausforderungen beim Testen, die durch das Selbstlernen KI-basierter Systeme entstehen

8.2 Test von autonomen KI-basierten Systemen

AI-8.2.1 K2 Beschreiben, wie autonome KI-basierte Systeme getestet werden

8.3 Testen auf algorithmische, stichprobenartige und unangemessene Verzerrungen

AI-8.3.1 K2 Erklären, wie man ein KI-basiertes System auf Verzerrungen prüft

8.4 Herausforderungen beim Testen probabilistischer und nicht-deterministischer KI-basierter Systeme

AI-8.4.1 K2 Erläutern der Herausforderungen beim Testen, die durch probabilistische und nicht-deterministische Eigenschaften KI-basierter Systeme entstehen

8.5 Herausforderungen beim Testen komplexer KI-basierter Systeme

AI-8.5.1 K2 Erläutern der Herausforderungen beim Testen, die durch die Komplexität KI-basierter Systemen entstehen

8.6 Testen der Transparenz, Interpretierbarkeit und Erklärbarkeit KI-basierter Systeme

AI-8.6.1 K2 Beschreiben, wie die Transparenz, Interpretierbarkeit und Erklärbarkeit KI-basierter Systeme getestet werden kann

HO-8.6.1 H2 Verwenden eines Werkzeugs um zu zeigen, wie Erklärbarkeit von Testern genutzt werden kann

8.7 Testorakel für KI-basierte Systeme

AI-8.7.1 K2 Erläutern der Herausforderungen bei der Erstellung von Testorakeln, die sich aus den spezifischen Merkmalen KI-basierter Systeme ergeben

8.8 Testziele und Akzeptanzkriterien

AI-8.8.1 K4 Auswählen geeigneter Testziele und Akzeptanzkriterien für die KI-spezifischen Qualitätsmerkmale eines bestimmten KI-basierten Systems

8.1 Herausforderungen beim Testen selbstlernender Systeme

Bei dem Test selbstlernender Systeme sind mehrere potenzielle Herausforderungen zu bewältigen (siehe Kapitel 2 für weitere Einzelheiten zu diesen Systemen), darunter:

- Unerwartete Änderungen: Die ursprünglichen Anforderungen und Beschränkungen, innerhalb derer das System funktionieren sollte, sind in der Regel bekannt, aber es gibt möglicherweise nur wenige oder gar keine Informationen über die vom System selbst vorgenommenen Änderungen. Normalerweise ist es möglich, anhand der ursprünglichen Anforderungen und des ursprünglichen Entwurfs (und aller spezifizierten Beschränkungen) zu testen, aber wenn das System eine innovative Implementierung gefunden oder eine Lösung durch Experimentieren ermittelt hat (deren Implementierung nicht sichtbar ist), kann es schwierig sein, Tests zu entwickeln, die für diese neue Implementierung geeignet sind. Wenn Systeme sich selbst (und ihre Ergebnisse) ändern, können sich auch die Ergebnisse von zuvor bestandenen Tests ändern. Dies ist eine Herausforderung für den Testentwurf. Sie kann durch den Entwurf geeigneter Tests angegangen werden, die relevant bleiben, wenn das System sein Verhalten ändert, wodurch ein potenzielles Problem bei Regressionstests vermieden wird. Es kann jedoch auch erforderlich sein, neue Tests auf der Grundlage des beobachteten neuen Systemverhaltens zu entwickeln.
- Komplexe Abnahmekriterien: Es kann notwendig sein, die Erwartungen an die Verbesserung des Systems zu definieren, wenn es selbst lernt. So kann man beispielsweise davon ausgehen, dass sich die funktionale Leistung des Systems verbessern sollte, wenn es sich selbst ändert. Außerdem kann es schnell komplex werden, eine nicht-triviale "Verbesserung" zu spezifizieren. So kann beispielsweise eine Mindestverbesserung erwartet werden (und nicht einfach irgendeine Verbesserung), oder die geforderte Verbesserung kann mit Umweltfaktoren verknüpft sein (z. B. ist eine Mindestverbesserung der Funktionalität X um 10 % erforderlich, wenn sich der Umweltfaktor F um mehr als Y ändert). Diese Probleme können durch die Spezifikation und den Test gegen komplexere Akzeptanzkriterien sowie durch die kontinuierliche Aufzeichnung der aktuellen funktionalen Leistung des Basis-Systems gelöst werden.
- Unzureichende Testzeit: Es kann notwendig sein zu wissen, wie schnell das System in verschiedenen Szenarien lernen und sich anpassen soll. Diese Akzeptanzkriterien können schwer zu spezifizieren und zu erfassen sein. Wenn sich ein System schnell anpasst, könnte die Zeit nicht ausreichen, um nach jeder Änderung manuell neue Tests auszuführen, so dass Tests zu schreiben sind, die automatisch ausgeführt werden können, wenn sich das System selbst ändert. Diese Herausforderungen können durch die Festlegung geeigneter Akzeptanzkriterien (siehe Abschnitt 8.8) und automatisiertes kontinuierliches Testen bewältigt werden.
- Anforderungen an die Ressourcen: Die Systemanforderungen können Akzeptanzkriterien hinsichtlich der Ressourcen enthalten, welche das System bei der Durchführung von Selbstlern- oder Anpassungsprozessen verwenden darf. Dies kann z. B. den Umfang der Verarbeitungszeit und des Arbeitsspeichers umfassen, der zur Verbesserung verwendet werden darf. Darüber hinaus ist zu überlegen, ob diese Ressourcennutzung mit einer messbaren Verbesserung der Funktionalität oder Genauigkeit verbunden sein sollte. Diese Herausforderung wirkt sich auf die Festlegung von Akzeptanzkriterien aus.
- Unzureichende Spezifikationen der Einsatzumgebung: Ein selbstlernendes System kann sich verändern, wenn die Eingaben aus der Umgebung, die es erhält, außerhalb des erwarteten Bereichs liegen oder wenn sie nicht in den Trainingsdaten enthalten waren. Diese Eingaben können Angriffe in Form von Datenverunreinigung sein (siehe Abschnitt 9.1.2). Es kann schwierig sein, das gesamte Spektrum der Einsatzumgebungen und Umgebungsveränderungen vorherzusagen. Daher ist es ebenso schwierig, einen vollständigen Satz repräsentativer Testfälle und Umgebungsanforderungen zu ermitteln. Im Idealfall wird der gesamte Umfang möglicher Änderungen in der Einsatzumgebung, auf die das System reagieren soll, als Akzeptanzkriterien definiert.

- Komplexe Testumgebung: Die Verwaltung einer Testumgebung, die alle potenziellen hochriskanten Änderungen der Einsatzumgebung nachahmen kann, ist eine Herausforderung und kann den Einsatz von Testwerkzeugen (z. B. ein Fehlereinfügungswerkzeug) erfordern. Je nach Art der Einsatzumgebung kann der Test durch Manipulation von Eingängen und Sensoren erfolgen oder durch Zugang zu verschiedenen physischen Umgebungen, in denen das System getestet werden kann.
- Unerwünschte Verhaltensänderungen: Ein selbstlernendes System ändert sein Verhalten auf der Grundlage seiner Eingaben. Es kann sein, dass die Tester dies nicht verhindern können, beispielsweise wenn ein System eines Drittanbieters verwendet wird oder wenn das Produktionssystem getestet wird. Durch die Wiederholung der gleichen Tests kann ein selbstlernendes System effektiver auf diese Tests reagieren, was dann das langfristige Verhalten des Systems beeinflussen kann. Es ist daher wichtig zu verhindern, dass die Tests das Verhalten eines selbstlernenden Systems ungünstig beeinflussen. Dies ist eine Herausforderung für den Testfallentwurf und das Testmanagement.

8.2 Test autonomer KI-basierter Systeme

Autonome Systeme müssen in der Lage sein zu entscheiden, wann sie ein menschliches Eingreifen benötigen und wann nicht. Um die Autonomie KI-basierter Systeme zu testen, müssen daher Bedingungen geschaffen werden, unter denen das System übt, diese Entscheidungen zu treffen.

Der Test auf Autonomie kann erfordern:

- Testen, ob das System für ein bestimmtes Szenario einen menschlichen Eingriff verlangt, wenn das System die Kontrolle abgeben sollte. Solche Szenarien könnten eine Änderung der Einsatzumgebung oder eine Überschreitung der Autonomiegrenzen des Systems sein.
- Testen, ob das System den Menschen zum Eingreifen auffordert, wenn es die Kontrolle nach einer bestimmten Zeitspanne abgeben sollte.
- Testen, ob das System unnötigerweise einen menschlichen Eingriff anfordert, obwohl es noch autonom arbeiten sollte.

Es kann hilfreich sein, die Grenzwertanalyse auf die Einsatzumgebung anzuwenden, um die notwendigen Bedingungen für diese Tests zu schaffen. Es kann eine Herausforderung sein, zu definieren, wie sich die Parameter zur Bestimmung der Autonomie in der Einsatzumgebung manifestieren, und die Testszenarien zu erstellen, die von dem Grad der Autonomie abhängen.

8.3 Testen auf algorithmische, stichprobenartige und unangemessene Verzerrungen

Ein ML-System sollte anhand der verschiedenen Verzerrungen evaluiert werden, und es sollten Maßnahmen zur Beseitigung unangemessener Verzerrungen getroffen werden. Dies kann bedeuten, dass absichtlich eine positive Verzerrung eingeführt wird, um der unangemessenen Verzerrung entgegenzuwirken.

Beim Testen mit einem unabhängigen Datensatz können Verzerrungen oft erkannt werden. Es kann jedoch schwierig sein, alle Daten zu identifizieren, die eine Verzerrung verursachen, da der ML-Algorithmus Kombinationen von scheinbar unverbundenen Merkmalen verwenden kann, um unerwünschte Verzerrungen zu erzeugen.

KI-basierte Systeme sollten auf algorithmische Verzerrungen, Stichprobenverzerrungen und unangemessene Verzerrungen getestet werden (siehe Abschnitt 2.4). Dies kann Folgendes beinhalten:

- Analyse während der Trainings-, Evaluierungs- und Optimierungsaktivitäten des Modells, um festzustellen, ob eine algorithmische Verzerrung vorliegt.

- Review der Quelle der Trainingsdaten und der zu ihrer Gewinnung verwendeten Verfahren, damit das Vorhandensein von Stichprobenverzerrungen erkannt werden kann.
- Review der Vorverarbeitung von Daten als Teil des ML-Workflows, um festzustellen, ob die Daten in einer Weise beeinflusst wurden, die zu einer Verzerrung der Stichprobe führen könnte.
- Messung, wie sich Änderungen der Systemeingaben auf die Systemausgaben über eine große Anzahl von Interaktionen auswirken, und Untersuchung der Ergebnisse auf der Grundlage der Gruppen von Personen oder Objekten, für oder gegen die das System möglicherweise unangemessen verzerrt ist. Dies ähnelt der in Abschnitt 8.6 beschriebenen LIME-Methode und kann sowohl in einer Produktionsumgebung als auch im Rahmen von Tests vor der Freigabe durchgeführt werden.
- Beschaffung zusätzlicher Informationen über die Attribute der Eingabedaten, die möglicherweise mit der Verzerrung zusammenhängen, und deren Korrelation mit den Ergebnissen. Dies könnte sich zum Beispiel auf demografische Daten beziehen, die bei Tests auf unangemessene Verzerrungen angebracht sein könnten, die Gruppen von Menschen betreffen, wenn die Zugehörigkeit zu einer Gruppe für die Bewertung der Verzerrung relevant ist, selbst aber keine Eingabe für das Modell darstellt. Dies liegt daran, dass die Verzerrung auf "versteckten" Variablen beruhen kann, die in den Eingabedaten nicht explizit vorhanden sind, sondern vom Algorithmus abgeleitet werden.

8.4 Herausforderungen beim Testen probabilistischer und nicht-deterministischer KI-basierter Systeme

Die meisten probabilistischen Systeme sind auch nicht-deterministisch, so dass die folgende Liste von Test-Herausforderungen typischerweise für KI-basierte Systeme auch mit nur einem dieser Attribute gilt:

- Es kann mehrere gültige Ergebnisse eines Tests mit demselben Satz von Vorbedingungen und Eingaben geben. Dies macht die Definition der erwarteten Ergebnisse schwieriger und kann zu Problemen führen:
  - Wenn Tests für Fehlernachtests wiederverwendet werden.
  - Wenn Tests für Regressionstests wiederverwendet werden.
  - Wo die Reproduzierbarkeit der Tests wichtig ist.
  - Wenn die Tests automatisiert werden.
- Tester benötigen in der Regel eine tiefere Kenntnis des geforderten Systemverhaltens, damit sie vernünftige Bedingungen für das Bestehen des Tests entwickeln können, anstatt einfach einen exakten Wert für das erwartete Testergebnis anzugeben. Im Vergleich zu konventionellen Systemen müssen Tester beispielsweise differenziertere erwartete Ergebnisse definieren. Diese erwarteten Testergebnisse können Toleranzen enthalten (z. B. "liegt das tatsächliche Ergebnis innerhalb von 2 % der optimalen Lösung?").
- Wenn ein einzelnes endgültiges Ergebnis eines Tests aufgrund der probabilistischen Eigenschaft des Systems nicht möglich ist, müssen Tester einen Test oft mehrmals durchführen, um ein statistisch gültiges Testergebnis zu erhalten.

8.5 Herausforderungen beim Testen komplexer KI-basierter Systeme

KI-gestützte Systeme werden häufig für die Durchführung von Aufgaben eingesetzt, die für Menschen zu komplex sind. Dies kann zu einem Testorakel-Problem führen, da die Tester nicht in der Lage sind, die erwarteten Ergebnisse zu ermitteln, wie sie es normalerweise tun würden (siehe Abschnitt 8.7). So werden KI-basierte Systeme zum Beispiel häufig eingesetzt, um Muster in großen Datenmengen zu erkennen. Sie werden dort eingesetzt, weil sie Muster finden können, die Menschen selbst nach

intensiver Analyse manuell nicht finden können. Somit kann es eine Herausforderung sein, das erforderliche Verhalten solcher Systeme in ausreichender Tiefe zu verstehen, um die erwarteten Ergebnisse erzeugen zu können.

Ein ähnliches Problem ergibt sich, wenn die interne Struktur eines KI-basierten Systems durch Software generiert wird, so dass sie für Menschen zu komplex ist. Dies führt dazu, dass das KI-basierte System nur als Black-Box getestet werden kann. Selbst wenn die interne Struktur sichtbar ist, liefert dies keine zusätzlichen nützlichen Informationen, die beim Testen helfen.

Die Komplexität KI-basierter Systeme nimmt zu, wenn sie probabilistische Ergebnisse liefern und nicht-deterministisch sind (siehe Abschnitt 8.4).

Die Probleme mit nicht-deterministischen Systemen werden noch verschärft, wenn ein KI-basiertes System aus mehreren interagierenden Komponenten besteht, die jeweils probabilistische Ergebnisse liefern. Ein System zur Gesichtserkennung zum Beispiel verwendet wahrscheinlich ein Modell, um ein Gesicht in einem Bild zu identifizieren, und ein zweites Modell, um zu erkennen, welches Gesicht identifiziert wurde. Die Interaktionen zwischen den KI-Komponenten können komplex und schwer zu verstehen sein, was es schwierig macht, alle Risiken zu erkennen und Tests zu entwerfen, die das System angemessen verifizieren.

8.6 Testen der Transparenz, Interpretierbarkeit und Erklärbarkeit KI-basierter Systeme

Informationen darüber, wie das System implementiert wurde, können von den Systementwicklern bereitgestellt werden. Dazu gehören die Quellen der Trainingsdaten, die Art und Weise, wie die Kennzeichnung durchgeführt wurde, und wie die Systemkomponenten konzipiert wurden. Wenn diese Informationen nicht zur Verfügung stehen, kann dies die Entwicklung von Tests erschweren. Wenn beispielsweise keine Informationen über Trainingsdaten verfügbar sind, wird es schwierig, mögliche Lücken in diesen Daten zu identifizieren und die Auswirkungen dieser Lücken zu testen. Diese Situation kann mit Black-Box- und White-Box-Tests verglichen werden und hat ähnliche Vor- und Nachteile. Die Transparenz kann getestet werden, indem man die dokumentierten Informationen über die Daten und den Algorithmus mit der tatsächlichen Implementierung vergleicht und feststellt, wie gut sie übereinstimmen.

Bei ML kann es oft schwieriger sein, den Zusammenhang zwischen einer bestimmten Eingabe und einer bestimmten Ausgabe zu erklären als bei herkömmlichen Systemen. Diese geringe Erklärbarkeit ist in erster Linie darauf zurückzuführen, dass das Modell, welches die Ausgabe generiert, selbst durch Code (den Algorithmus) erzeugt wird und nicht die Art und Weise widerspiegelt, wie Menschen über ein Problem denken. Verschiedene ML-Modelle bieten unterschiedliche Grade an Erklärbarkeit und sollten auf der Grundlage der Anforderungen an das System ausgewählt werden, zu denen auch Erklärbarkeit und Testbarkeit gehören können.

Eine Methode zum Verständnis der Erklärbarkeit ist das dynamische Testen des ML-Modells, indem man die Testdaten durch Störungen verändert. Es gibt Methoden, um die Erklärbarkeit auf diese Weise zu quantifizieren und visuelle Erklärungen dafür zu liefern. Einige dieser Methoden sind modellunabhängig, während andere spezifisch für einen bestimmten Modelltyp sind und den Zugang zu diesem erfordern. Auch exploratives Testen kann verwendet werden, um die Beziehung zwischen den Eingaben und Ausgaben eines Modells besser zu verstehen.

Die LIME-Methode ist modellunabhängig und verwendet dynamisch eingefügte Störungen der Eingaben und die Analyse der Ausgaben, um den Testern einen Überblick über die Beziehung zwischen Eingaben und Ausgaben zu geben. Dies kann eine effektive Methode sein, um die Erklärbarkeit des Modells zu gewährleisten. Sie beschränkt sich jedoch darauf, mögliche Gründe für

die Ausgaben zu liefern, anstatt einen endgültigen Grund zu nennen, und ist nicht für alle Arten von Algorithmen geeignet.

Die Interpretierbarkeit eines KI-basierten Systems hängt stark davon ab, wer es einsetzt. Unterschiedliche Interessengruppen haben möglicherweise unterschiedliche Anforderungen in Bezug darauf, wie gut sie die zugrundeliegende Technike wie z.B. den ML-Algorithmus begreifen müssen.

Das Messen und Testen des Verständnisses sowohl für die Interpretierbarkeit als auch für die Erklärbarkeit kann eine Herausforderung darstellen, da die Beteiligten unterschiedliche Fähigkeiten haben und sich möglicherweise nicht einig sind. Darüber hinaus kann es bei vielen Systemtypen schwierig sein, das Profil typischer Interessengruppen zu ermitteln. Wo solche Tests und Messungen durchgeführt werden, geschieht dies in der Regel in Form von Benutzerumfragen und/oder Fragebögen.

8.6.1 Praktische Übung: Modell-Erklärbarkeit

Verwenden Sie ein geeignetes Werkzeug, um die Erklärbarkeit für das zuvor erstellte Modell herzustellen. Für ein Bild-Klassifikationsmodell oder ein Text-Klassifikationsmodell kann zum Beispiel eine modellunabhängige Methode wie LIME geeignet sein.

Die Lernenden sollten das Tool nutzen, um Erklärungen zu den Modellentscheidungen zu erstellen, insbesondere dazu, wie die Merkmale der Eingaben die Ausgaben beeinflussen.

8.7 Testorakel für KI-basierte Systeme

Ein großes Problem beim Testen KI-basierter Systeme kann die Spezifikation der erwarteten Ergebnisse sein. Ein Testorakel ist eine Informationsquelle zur Ermittlung des erwarteten Ergebnisses, um es mit dem tatsächlichen Ergebnis eines Systems unter Test zu vergleichen. [I01]. Die Herausforderung bei der Bestimmung der erwarteten Ergebnisse ist als Testorakelproblem bekannt.

Bei komplexen, nicht-deterministischen oder probabilistischen Systemen kann es schwierig sein, ein Testorakel zu erstellen, ohne die "Grundwahrheit" zu kennen (d. h. das tatsächliche Ergebnis in der realen Welt, welches das KI-basierte System vorherzusagen versucht). Diese "Grundwahrheit" unterscheidet sich von einem Testorakel insofern, als ein Testorakel nicht unbedingt einen erwarteten Wert liefert, sondern nur einen Mechanismus, mit dem sich feststellen lässt, ob das System korrekt arbeitet oder nicht.

KI-basierte Systeme können sich weiterentwickeln (siehe Abschnitt 2.3), und das Testen selbstlernender Systeme (siehe Abschnitt 8.1) kann ebenfalls unter Testorakelproblemen leiden, da sie sich selbst verändern und dadurch eine häufige Aktualisierung der funktionalen Erwartungen an das System erforderlich machen können.

Ein weiterer Grund für die Schwierigkeit, ein effektives Testorakel zu erhalten, ist, dass die Beurteilung des Softwareverhaltens in vielen Fällen subjektiv ist. Virtuelle Assistenten (z. B. Siri und Alexa) sind ein Beispiel für dieses Problem, da verschiedene Benutzer oft ganz unterschiedliche Erwartungen haben und je nach Wortwahl und Deutlichkeit der Sprache unterschiedliche Ergebnisse erzielen können.

In einigen Situationen kann es möglich sein, das erwartete Ergebnis mit Grenzwerten oder Toleranzen zu definieren. So könnte beispielsweise der Haltepunkt eines autonomen Fahrzeugs auf eine maximale Entfernung zu einem bestimmten Punkt festgelegt werden. Im Zusammenhang mit Expertensystemen kann die Bestimmung der erwarteten Ergebnisse durch die Konsultation eines Experten erfolgen (wobei zu beachten ist, dass die Meinung des Experten auch falsch sein kann). In diesem Zusammenhang sind mehrere wichtige Faktoren zu berücksichtigen:

- Menschliche Experten sind unterschiedlich kompetent. Die beteiligten Experten müssen mindestens so kompetent sein wie die Experten, die durch das System ersetzt werden sollen.
- Experten sind nicht immer einer Meinung, selbst dann, wenn ihnen dieselben Informationen vorgelegt werden.
- Es kann vorkommen, dass menschliche Experten die Automatisierung ihrer Beurteilung nicht gutheißen. In solchen Fällen sollte ihre Bewertung potenzieller Ergebnisse doppelblind erfolgen (d. h. weder die Experten noch die Evaluierer der Ergebnisse wissen, welche Bewertungen automatisiert wurden).
- Menschen neigen eher dazu, ihre Antworten mit Vorbehalten zu versehen (z. B. mit Sätzen wie "Ich bin mir nicht sicher, aber..."). Wenn diese Art von Vorbehalt dem KI-basierten System nicht zur Verfügung steht, sollte dies beim Vergleich der Antworten berücksichtigt werden.

Es gibt Testverfahren, die das Testorakelproblem mildern können, wie vergleichendes Testen (*back-to-back-testing*, siehe Abschnitt 9.3), A/B-Testen (siehe Abschnitt 9.4) und metamorphes Testen (siehe Abschnitt 9.5).

8.8 Testziele und Akzeptanzkriterien

Die Testziele und Akzeptanzkriterien für ein System müssen auf den wahrgenommenen Produktrisiken basieren. Diese Risiken lassen sich häufig durch eine Analyse der erforderlichen Qualitätsmerkmale ermitteln. Qualitätsmerkmale für KI-basierte Systeme sind die traditionell in ISO/IEC 25010 [S06] berücksichtigten Merkmale (d. h. funktionale Eignung, Performanz, Kompatibilität, Gebrauchstauglichkeit, Zuverlässigkeit, IT-Sicherheit, Wartbarkeit und Übertragbarkeit). Zusätzlich sollten auch die folgenden Qualitätsmerkmale für KI-basierte Systeme berücksichtigt werden (siehe Kapitel 2):

Aspekt	**Kriterien für die Akzeptanz**
Anpassbarkeit	• Überprüfen Sie, ob das System immer noch korrekt funktioniert und die nicht-funktionalen Anforderungen erfüllt, wenn es sich an eine Änderung in seiner Umgebung anpasst. Dies kann als eine Form von automatisierten Regressionstests durchgeführt werden. • Überprüfen Sie die Zeit, die das System benötigt, um sich an eine Veränderung in seiner Umgebung anzupassen. • Überprüfen Sie die Ressourcen, die verwendet werden, wenn sich das System an eine Veränderung in seiner Umgebung anpasst.
Flexibilität	• Überprüfen Sie, wie das System in Kontexten außerhalb der ursprünglichen Spezifikation funktioniert. Dies kann in Form von automatisierten Regressionstests erfolgen, die in der veränderten Einsatzumgebung durchgeführt werden. • Überprüfen Sie, wie viel Zeit und/oder Ressourcen das System benötigt, um sich auf einen neuen Kontext einzustellen.
Evolution	• Überprüfen Sie, wie gut das System aus seinen eigenen Erfahrungen lernt. • Überprüfen Sie, wie gut das System mit der Veränderung des Datenprofils (d. h. der Konzeptdrift) zurechtkommt.

Aspekt	Kriterien für die Akzeptanz
Autonomie	• Überprüfen Sie, wie das System reagiert, wenn es aus dem Betriebsbereich gezwungen wird, in dem es völlig autonom sein soll. • Überprüfen Sie, ob das System "überredet" werden kann, einen menschlichen Eingriff anzufordern, obwohl es völlig autonom sein sollte.
Transparenz, Interpretierbarkeit und Erklärbarkeit	• Überprüfen Sie die Transparenz, indem Sie die Zugänglichkeit des Algorithmus und des Datensatzes reviewen. • Überprüfen Sie die Interpretierbarkeit und Erklärbarkeit, indem Sie die Systembenutzer befragen, oder, falls die tatsächlichen Systembenutzer nicht verfügbar sind, Personen mit einem ähnlichen Hintergrund.
Freiheit von unangemessener Verzerrung	• Bei Systemen, bei denen eine Beeinflussung durch Verzerrungen wahrscheinlich ist, kann dies mit einer unabhängigen Testsuite ohne Verzerrungen oder durch den Einsatz von Sachverständigen geprüft werden. • Vergleichen Sie die Testergebnisse mit externen Daten wie Volkszählungsdaten, um zu prüfen, ob die abgeleiteten Variablen unerwünscht verzerrt sind (externe Validitätstests).
Ethik	• Überprüfen Sie das System anhand einer geeigneten Checkliste, z. B. der EG-Bewertungsliste für vertrauenswürdige künstliche Intelligenz [R21], welche die wichtigsten Anforderungen der Ethikrichtlinien für vertrauenswürdige künstliche Intelligenz (KI) [R22] unterstützt.
Probabilistische Systeme und nichtdeterministische Systeme	• Dies kann nicht mit genauen Akzeptanzkriterien evaluiert werden. Wenn das System korrekt arbeitet, kann es bei denselben Tests zu leicht unterschiedlichen Ergebnissen kommen.
Nebenwirkungen	• Identifizieren Sie potenziell schädliche Nebenwirkungen und versuchen Sie, Tests zu erstellen, die das System zu diesen Nebenwirkungen veranlassen.
Belohnungs-Hacking	• Unabhängige Tests können Belohnungs-Hacking aufdecken, wenn diese Tests ein anderes Mittel zur Erfolgsmessung verwenden als der getestete intelligente Agent.
Funktionale Sicherheit	• Dies muss sorgfältig geprüft werden, vielleicht in einer virtuellen Testumgebung (siehe Abschnitt 10.2). Dazu könnten Versuche gehören, ein System zu zwingen, sich selbst Schaden zuzufügen.

Tabelle 3: Qualitätsmerkmale und Akzeptanzkriterien für KI

Für ML-Systeme sollten die erforderlichen funktionalen Leistungsmetriken von ML festgelegt werden (siehe Kapitel 5).

9 Methoden und Verfahren für das Testen KI-basierter Systeme - 245 Minuten

Schlüsselwörter

A/B-Testen, gegnerisches Testen (*adversarial testing*), vergleichendes Testen, intuitive Testfallermittlung, erfahrungsbasiertes Testen, exploratives Testen, metamorphe Relation (MR), metamorphes Testen (MT), paarweises Testen, Pseudo-Orakel, Testorakelproblem, Tour

KI-spezifische Schlüsselwörter

gegnerischer Angriff, gegnerisches Beispiel, Datenverunreinigung, ML-System, trainiertes Modell

Lernziele für Kapitel 9:

9.1 Gegnerische Angriffe und Datenverunreinigung

AI-9.1.1 K2 Erläutern, wie das Testen von ML-Systemen dazu beitragen kann, gegnerische Angriffe und Datenverunreinigung zu verhindern

9.2 Paarweises Testen

AI-9.2.1 K2 Erklären, wie paarweises Testen auf das Testen KI-basierter Systeme angewendet wird

HO-9.2.1 H2 Anwenden von paarweisem Testen für Entwurf und Ausführung von Testfällen für ein KI-basiertes System

9.3 Vergleichendes Testen

AI-9.3.1 K2 Erklären, wie vergleichendes Testen auf das Testen KI-basierter Systeme angewendet wird

9.4 A/B-Testen

AI-9.4.1 K2 Erläutern, wie A/B-Testen auf das Testen KI-basierter Systeme angewendet wird

9.5 Metamorphes Testen

AI-9.5.1 K3 Anwenden von metamorphem Testen für das Testen KI-basierter Systeme

HO-9.5.1 H2 Anwenden von metamorphem Testen, um Testfälle für ein bestimmtes Szenario zu entwerfen und auszuführen

9.6 Erfahrungsbasiertes Testen KI-basierter Systeme

AI-9.6.1 K2 Erklären, wie erfahrungsbasiertes Testen auf das Testen KI-basierter Systeme angewendet wird

HO-9.6.1 H2 Anwenden von explorativem Testen auf ein KI-basiertes System

9.7 Auswahl von Testverfahren für KI-basierte Systeme

AI-9.7.1 K4 Auswählen geeigneter Testverfahren für ein bestimmtes Szenario, wenn Sie ein KI-basiertes System testen

9.1 Gegnerische Angriffe und Datenverunreinigung

9.1.1 Gegnerische Angriffe

Bei einem gegnerischen Angriff verändert ein Angreifer auf subtile Weise gültige Eingaben, die an das trainierte Modell weitergegeben werden, so dass es falsche Vorhersagen macht. Diese gestörten Eingaben, die als gegnerische Beispiele bekannt sind, wurden zuerst bei Spam-Filtern bemerkt, die durch eine geringfügige Änderung einer Spam-E-Mail ausgetrickst werden konnten, ohne die Lesbarkeit zu verlieren. In letzter Zeit werden sie verstärkt mit Bildklassifizierern in Verbindung gebracht. Durch die einfache Änderung einiger weniger Pixel, die für das menschliche Auge unsichtbar sind, kann ein neuronales Netz dazu gebracht werden, seine Bildklassifikation auf ein ganz anderes Objekt zu ändern, und zwar mit einem hohen Maß an Vertrauen.

Gegnerische Beispiele sind im Allgemeinen übertragbar [B17], was bedeutet, dass ein gegnerisches Beispiel, das ein ML-System zum Scheitern bringt, oft auch ein anderes ML-System zum Scheitern bringt, das für dieselbe Aufgabe trainiert wurde. Selbst wenn das zweite ML-System mit anderen Daten trainiert wurde und auf anderen Architekturen basiert, ist es oft immer noch anfällig für Versagen bei denselben gegnerischen Beispielen.

Bei gegnerischen White-Box-Angriffen weiß der Angreifer, welcher Algorithmus zum Trainieren des Modells verwendet wurde und welche Modelleinstellungen und Parameter verwendet wurden (es besteht ein angemessenes Maß an Transparenz). Der Angreifer nutzt dieses Wissen, um gegnerische Beispiele zu generieren, indem er z. B. kleine Änderungen an den Eingaben vornimmt und beobachtet, welche davon große Änderungen an den Modellausgaben verursachen.

Bei gegnerischen Black-Box-Angriffen untersucht der Angreifer das Modell, um dessen Funktionalität zu ermitteln, und erstellt dann ein Duplikat des Modells, das eine ähnliche Funktionalität bietet. Der Angreifer wendet dann einen White-Box-Ansatz an, um gegnerische Beispiele für dieses Duplikatmodell zu finden. Da die gegnerischen Beispiele im Allgemeinen übertragbar sind, funktionieren dieselben Gegenbeispiele normalerweise auch für das ursprüngliche Modell.

Wenn es nicht möglich ist, ein Duplikatmodell zu erstellen, kann es möglich sein, automatisierte Tests in großem Umfang durchzuführen, um verschiedene gegnerische Beispiele zu entdecken und die Ergebnisse zu beobachten.

Beim gegnerischen Testen werden einfach gegnerische Angriffe mit dem Ziel durchgeführt, Schwachstellen zu erkennen, damit Präventivmaßnahmen zum Schutz vor künftigen Fehlerwirkungen ergriffen werden können. Die identifizierten gegnerischen Beispiele werden zu den Trainingsdaten hinzugefügt, damit das Modell darauf trainiert wird, sie korrekt zu erkennen.

9.1.2 Datenverunreinigung

Bei Datenverunreinigungs-Angriffen manipuliert ein Angreifer die Trainingsdaten, um eines der folgenden beiden Ergebnissen zu erzielen. Der Angreifer kann einerseits Hintertüren oder Trojaner für neuronale Netze einfügen, um künftige Einbrüche zu erleichtern. Andererseits und häufiger verwenden Angreifer verfälschte Trainingsdaten (z. B. falsch gekennzeichnete Daten), um das trainierte Modell dazu zu bringen, falsche Vorhersagen zu machen.

Verunreinigungsangriffe können also u.A. darauf abzielen, das ML-System in bestimmten Situationen zu Fehlklassifikationen zu veranlassen. Sie können aber auch wahllos erfolgen, wie z. B. bei einem Denial-of-Service-Angriff. Ein bekanntes Beispiel für einen Verunreinigungsangriff war die Beschädigung des Microsoft Tay-Chatbots, bei der eine relativ kleine Anzahl schädlicher Twitter-Konversationen das System durch Rückmeldungen darauf trainierte, in Zukunft anstößige Konversationen zu liefern. Eine weit verbreitete Form des Datenverunreinigungsangriffs besteht darin,

dass Millionen von Spam-E-Mails fälschlicherweise als Nicht-Spam gemeldet werden, um die Spam-Filter-Software zu beeinflussen. Ein Bereich, der bei Datenverunreinigung Anlass zur Sorge gibt, ist die Möglichkeit, dass öffentliche, weit verbreitete KI-Datensätze verunreinigt werden.

Tests zur Erkennung von Datenverunreinigung sind mit explorativer Datenanalyse (EDA) möglich, da vergiftete Daten als Ausreißer auftauchen können. Darüber hinaus können die Datenbeschaffungsrichtlinien überprüft werden, um die Herkunft der Trainingsdaten sicherzustellen. Wenn ein betriebsbereites ML-System durch Einschleusung verunreinigter Daten angegriffen werden kann, könnte mit Hilfe von A/B-Tests (siehe Abschnitt 9.4) überprüft werden, ob die aktualisierte Version des Systems noch eng mit der vorherigen Version übereinstimmt. Alternativ kann auch durch Regressionstests eines aktualisierten Systems mit einer vertrauenswürdigen Testsuite festgestellt werden, ob ein System verunreinigt wurde.

9.2 Paarweises Testen

Die Anzahl der Parameter, die für ein KI-basiertes System von Interesse sind, kann extrem hoch sein, insbesondere wenn das System große Datenmengen verwendet oder mit der Außenwelt interagiert, wie z. B. bei einem selbstfahrenden Auto. Bei einem erschöpfenden Test müssten alle möglichen Kombinationen dieser Parameter auf alle möglichen Werte eingestellt und getestet werden. Da dies jedoch zu einer praktisch unendlichen Anzahl von Tests führen würde, werden Testverfahren eingesetzt, um eine Teilmenge auszuwählen, die in der begrenzten verfügbaren Zeit durchgeführt werden kann.

Wenn es möglich ist, zahlreiche Parameter zu kombinieren, von denen jeder viele diskrete Werte haben kann, kann kombinatorisches Testen angewendet werden, um die Anzahl der erforderlichen Testfälle erheblich zu reduzieren, idealerweise ohne die Fähigkeit der Testsuite zur Fehlerfindung zu beeinträchtigen. Es gibt mehrere kombinatorische Testverfahren (siehe [I02] und [S08]). In der Praxis ist jedoch das paarweise Testen das am weitesten verbreitete Verfahren, da es einfach zu verstehen ist und über eine umfangreiche Werkzeugunterstützung verfügt. Darüber hinaus hat die Forschung gezeigt, dass die meisten Fehlerzustände durch Interaktionen weniger Parameter verursacht werden [B33].

Das paarweise Testen wird verwendet, wenn mehrere Parameter mit jeweils mehreren möglichen Werten in Kombination getestet werden müssen, wodurch mehr Kombinationen entstehen, als in der verfügbaren Zeit getestet werden können. Die Parameter könnten insofern unabhängig sein, dass jede Option für jeden beliebigen Faktor (d.h. jeder ausgewählte Wert für einen beliebigen Parameter) mit jeder Option eines beliebigen anderen Faktors kombiniert werden kann, dies ist jedoch nicht immer der Fall. Die Kombination eines bestimmten Parameters (Variable oder Faktor) mit einem bestimmten Wert dieses Parameters wird als Parameter-Wert-Paar bezeichnet (wenn z.B. 'Farbe' ein Parameter mit sieben zulässigen Werten einschließlich 'rot‘ ist, dann könnte 'Farbe = rot' ein Parameter-Wert-Paar sein). Das paarweise Testen verwendet kombinatorische Verfahren, um sicherzustellen, dass jedes Parameter-Wert-Paar einmal gegen jedes Parameter-Wert-Paar jedes anderen Parameters getestet wird (d.h. es werden 'alle Paare' von Parameter-Wert-Paaren für zwei beliebige unterschiedliche Parameter getestet). Dabei wird vermieden, dass alle Kombinationen von Parameter-Wert-Paaren getestet werden.

In der Praxis kann selbst der Einsatz paarweises Testens bei einigen Systemen zu umfangreichen Testsuiten führen, und es wird oft notwendig, Automatisierung und virtuelle Testumgebungen einzusetzen (siehe Abschnitt 10.2), um die erforderliche Anzahl von Tests durchführen zu können. Wenn man beispielsweise selbstfahrende Autos in Betracht zieht, müssen abstrakte Testszenarien für Systemtests sowohl die verschiedenen Umgebungen, in denen die Autos voraussichtlich eingesetzt werden, als auch die verschiedenen Fahrzeugfunktionen testen. So müssen die Parameter die verschiedenen Umgebungsbeschränkungen (z. B. Straßentypen und -oberflächen, Wetter- und

Verkehrsbedingungen sowie Sichtverhältnisse) und die verschiedenen Selbstfahrfunktionen (z. B. adaptiver Tempomat, Spurhalteassistent und Spurwechselassistent) umfassen. Zusätzlich zu diesen Parametern könnten die Eingaben von Sensoren mit unterschiedlichem Wirkungsgrad berücksichtigt werden (z. B. werden die Eingaben einer Videokamera mit fortschreitender Fahrt und zunehmender Verschmutzung schlechter).

Die Forschung ist sich derzeit nicht im Klaren darüber, welches Maß an Strenge für den Einsatz kombinatorischen Testens bei sicherheitskritischen KI-basierten Systemen wie selbstfahrenden Autos erforderlich wäre. Auch wenn paarweises Testen möglicherweise nicht ausreicht, ist bekannt, dass der Ansatz bei der Fehlerfindung wirksam ist.

9.2.1 Praktische Übung: Paarweises Testen

Verwenden Sie für ein implementiertes KI-basiertes System mit mindestens fünf Parametern und mindestens fünfhundert möglichen Kombinationen ein Tool für paarweises Testen, um eine reduzierte Menge paarweiser Kombinationen zu ermitteln und Tests für diese Kombinationen durchzuführen.

Vergleichen Sie die Anzahl der getesteten paarweisen Kombinationen mit der Anzahl, die erforderlich wäre, wenn alle theoretisch möglichen Kombinationen getestet werden würden.

9.3 Vergleichendes Testen

Eine der möglichen Lösungen für das Testorakelproblem (siehe Abschnitt 8.7) beim Testen KI-basierter Systeme ist die Verwendung von vergleichendem Testen (*back-to-back testing*). Dies wird auch als differenzieller Test bezeichnet. Bei vergleichenden Tests wird eine alternative Version des Systems als Pseudo-Orakel verwendet und dessen Ausgaben mit den vom SUT erzeugten Testergebnissen verglichen. Bei dem Pseudo-Orakel kann es sich um ein bestehendes System handeln oder es wird von einem anderen Team entwickelt, möglicherweise auf einer anderen Plattform, mit einer anderen Architektur und in einer anderen Programmiersprache. Beim Testen der funktionalen Eignung (im Gegensatz zu den nicht-funktionalen Anforderungen) ist das als Pseudo-Orakel verwendete System nicht gezwungen, die gleichen nicht-funktionalen Akzeptanzkriterien zu erfüllen wie das SUT. Es muss beispielsweise nicht so schnell ausgeführt werden und kann in diesem Fall wesentlich kostengünstiger erstellt werden.

Im Zusammenhang mit ML ist es möglich, verschiedene Frameworks, Algorithmen und Modelleinstellungen zu verwenden, um ein ML-Pseudo-Orakel zu erstellen. In einigen Situationen kann es auch möglich sein, ein Pseudo-Orakel mit herkömmlicher, nicht KI-gestützter Software zu erstellen.

Damit Pseudo-Orakel bei der Fehlerfindung effektiv sind, sollte es keine gemeinsame Software im Pseudo-Orakel und im SUT geben. Andernfalls wäre es möglich, dass ein und derselbe Fehlerzustand in beiden Tests dazu führt, dass die beiden Testergebnisse übereinstimmen, wenn beide fehlerhaft sind. Bei der Vielzahl unausgereifter, wiederverwendbarer, quelloffener KI-Software, die zur Entwicklung KI-basierter Systeme verwendet wird, kann die Wiederverwendung von Code zwischen dem Pseudo-Orakel und dem SUT das Pseudo-Orakel gefährden. Eine unzureichende Dokumentation von wiederverwendbaren KI-Lösungen kann es den Testern zusätzlich erschweren, dieses Problem zu erkennen.

9.4 A/B-Testen

A/B-Testen ist eine Methode, bei der die Reaktion von zwei Varianten des Programms (A und B) auf dieselben Eingaben verglichen wird, um zu ermitteln, welche der beiden Varianten besser ist. Es handelt sich um eine statistische Testvorgehensweise, die in der Regel den Vergleich von

Testergebnissen aus mehreren Testläufen erfordert, um den Unterschied zwischen den Programmen zu ermitteln.

Ein einfaches Beispiel für diese Methode ist das Versenden von zwei Werbeangeboten per E-Mail an eine in zwei Gruppen aufgeteilte Marketingliste. Die Hälfte der Liste erhält Angebot A, die andere Hälfte Angebot B, und der Erfolg der beiden Angebote hilft bei der Entscheidung, welches in Zukunft verwendet werden soll. Viele E-Commerce- und webbasierte Unternehmen setzen A/B-Testen im Produktivbetrieb ein, indem sie verschiedene Verbraucher auf unterschiedliche Funktionen umleiten, um die Präferenzen der Verbraucher zu ermitteln.

A/B-Testen ist ein Ansatz zur Lösung des Testorakelproblems, bei dem das bestehende System als Teilorakel verwendet wird. Beim A/B-Testen werden keine Testfälle generiert, und es gibt keine Hinweise darauf, wie die Tests gestaltet werden sollten, obwohl in den Tests häufig Eingaben aus dem Produktivbetrieb verwendet werden.

A/B-Testen kann verwendet werden, um Aktualisierungen eines KI-basierten Systems zu testen, für das es vereinbarte Akzeptanzkriterien gibt, wie z. B. die in Kapitel 5 beschriebenen funktionalen Leistungsmetriken von ML. Bei jeder Aktualisierung des Systems wird mit A/B-Testen geprüft, ob die aktualisierte Variante genauso gut wie oder besser als die vorherige Variante abschneidet. Ein solcher Ansatz kann für einen einfachen Klassifikator, aber auch für das Testen weitaus komplexerer Systeme verwendet werden. So kann beispielsweise eine Aktualisierung zur Verbesserung der Effektivität eines intelligenten städtischen Verkehrsleitsystems auch mit A/B-Testen getestet werden (z. B. durch den Vergleich der durchschnittlichen Pendelzeiten für zwei Varianten des Systems in aufeinander folgenden Wochen).

A/B-Testen kann auch zum Testen selbstlernender Systeme verwendet werden. Wenn das System eine Änderung vornimmt, werden automatisierte Tests durchgeführt und die resultierenden Systemeigenschaften mit denen vor der Änderung verglichen. Wurde das System durch die Änderung verbessert, wird die Änderung akzeptiert, andernfalls kehrt das System in seinen vorherigen Zustand zurück.

Ein wesentlicher Unterschied zwischen A/B-Testen und vergleichendem Test besteht darin, dass bei A/B-Testen in der Regel zwei Varianten desselben Systems miteinander verglichen werden, während vergleichende Tests eher zur Aufdeckung von Fehlerzuständen eingesetzt werden.

9.5 Metamorphes Testen (MT)

Metamorphes Testen [B18] ist ein Verfahren zur Erzeugung von Testfällen, die auf einem bestandenen Ausgangstestfall basieren. Ein oder mehrere Folgetestfälle werden erzeugt, indem der Ausgangstestfall auf der Grundlage einer metamorphen Relation (MR) verändert (metamorphisiert) wird. Die MR basiert auf einer Eigenschaft einer geforderten Funktion des Testobjekts und beschreibt, wie sich eine Änderung der Testeingaben eines Testfalls in den erwarteten Ergebnissen desselben Testfalls niederschlägt.

Ein Beispiel wäre ein Programm, das den Durchschnitt einer Reihe von Zahlen bestimmt. Es wird ein Ausgangstestfall generiert, der eine Reihe von Zahlen und einen erwarteten Durchschnitt enthält. Der Ausgangstestfall wird ausgeführt, um zu bestätigen, dass er bestanden wird (d.h. der Testfall ist korrekt und hat keine Fehlerwirkung aufgedeckt). Es ist nun möglich, Folgetestfälle zu generieren, die auf dem basieren, was über die Durchschnittsfunktion des Programms bekannt ist. Zu Beginn kann einfach die Reihenfolge der zu mittelnden Zahlen geändert werden. Angesichts der Durchschnittsfunktion kann man vorhersagen, dass das erwartete Ergebnis dasselbe bleibt. So kann ein Folgetestfall mit den Zahlen in einer anderen Reihenfolge generiert werden, ohne dass das erwartete Ergebnis berechnet werden muss. Bei einem großen Zahlensatz könnte dies z. B. dazu führen, dass eine große Anzahl verschiedener Zahlensätze erzeugt wird, in denen dieselben Zahlen in unterschiedlicher Reihenfolge

verwendet werden, und jeder dieser Sätze könnte zur Erstellung eines eigenen Folgetestfalls verwendet werden. Alle diese Testfälle würden auf demselben Ausgangstestfall basieren und das gleiche erwartete Ergebnis haben.

Es ist üblich, MRs und Folgetestfälle zu haben, bei denen sich das erwartete Ergebnis von dem ursprünglich erwarteten Ergebnis des Ausgangstestfalls unterscheidet. Beispielsweise kann unter Verwendung der gleichen Durchschnittsfunktion eine MR abgeleitet werden, bei der jedes Element der Eingabemenge mit zwei multipliziert wird. Das erwartete Ergebnis für eine solche Menge ist einfach das ursprüngliche erwartete Ergebnis multipliziert mit zwei. In ähnlicher Weise könnte jeder andere Wert als Multiplikator verwendet werden, um potenziell eine unendliche Anzahl von Folgetestfällen auf der Grundlage dieser MR zu erzeugen.

MT kann für die meisten Testobjekte verwendet und sowohl für funktionale als auch für nicht-funktionale Tests eingesetzt werden (z. B. umfasst der Installationstest verschiedene Zielkonfigurationen, bei denen die Installationsparameter in unterschiedlicher Reihenfolge ausgewählt werden können). Das ist besonders nützlich, wenn die Generierung der erwarteten Ergebnisse problematisch ist, weil es kein einfach zu erstellendes Testorakel gibt. Dies ist z. B. bei einigen KI-basierten Systemen der Fall, die auf der Analyse großer Datenmengen beruhen, oder bei solchen, bei denen den Testern nicht klar ist, wie der ML-Algorithmus seine Vorhersagen ableitet. Im Bereich der KI wurde MT u. a. zum Testen von Bilderkennung, Suchmaschinen, Routenoptimierung und Spracherkennung eingesetzt.

Wie oben erläutert, kann MT auf einem bestandenen Ausgangstestfall basieren, aber ML ist auch nützlich, wenn die Überprüfung, ob ein Ausgangstestfall korrekt ist, nicht möglich ist. Dies kann zum Beispiel der Fall sein, wenn das Programm eine Funktion implementiert, die für einen menschlichen Tester zu komplex ist, um sie nachzubilden und als Testorakel zu verwenden, wie dies bei einigen KI-basierten Systemen der Fall ist. In diesem Fall kann MT verwendet werden, um einen oder mehrere Testfälle zu generieren, die ausgeführt werden und eine Reihe von Ausgaben erzeugen, bei denen dann die Beziehungen zwischen den Ausgaben auf ihre Gültigkeit überprüft werden können. Bei dieser Form der MT ist also nicht bekannt, ob die einzelnen Tests korrekt sind, aber die Beziehungen zwischen ihnen müssen zutreffen, was das Vertrauen in das Programm erhöht. Ein Beispiel wäre ein KI-basiertes versicherungsmathematisches Programm, welches das Sterbealter auf der Grundlage eines großen Datensatzes vorhersagt, wobei beispielsweise bekannt ist, dass das vorhergesagte Sterbealter sinken (oder zumindest gleichbleiben) sollte, wenn die Anzahl der gerauchten Zigaretten erhöht wird.

MT ist ein vergleichsweise neues Testverfahren, das erstmals 1998 vorgeschlagen wurde. Es unterscheidet sich von den traditionellen Testverfahren dadurch, dass die erwarteten Ergebnisse der Folgetestfälle nicht in Form von absoluten Werten beschrieben werden, sondern relativ zu den erwarteten Ergebnissen im Ausgangstestfall sind. Es basiert auf einem leicht verständlichen Konzept, kann von Testern angewandt werden, die wenig Erfahrung mit der Anwendung des Verfahrens haben, aber die Anwendungsdomäne verstehen, und hat ähnliche Kosten wie die traditionellen Verfahren. Untersuchungen haben gezeigt, dass mit nur drei bis sechs verschiedenen MRs mehr als 90 % der Fehlerzustände aufgedeckt werden können, die mit Verfahren auf der Grundlage eines traditionellen Testorakels entdeckt werden könnten [B19]. Es ist möglich, aus gut spezifizierten MRs und einem Ausgangstestfall automatisch Folgetestfälle zu generieren. Kommerzielle Werkzeuge sind derzeit (Stand 2021) nicht verfügbar, obwohl Google bereits automatisierte MT zum Testen von Android-Grafiktreibern mit dem Tool GraphicsFuzz einsetzt, das als Open Source zur Verfügung steht (siehe [R23]).

9.5.1 Praktische Übung: Metamorphes Testen

In dieser Übung werden die Lernenden praktische Erfahrungen mit den folgenden Themen sammeln:

- Ableitung mehrerer metamorpher Relationen (MRs) für eine bestimmte KI-basierte Anwendung oder ein Programm. Bei einigen dieser MRs sollten die erwarteten Ergebnisse der Ausgangs- und Folgetestfälle gleich sein und bei einigen sollten sie sich unterscheiden.
- Entwerfen von Ausgangstestfällen für die KI-basierte Anwendung oder das Programm. Diese müssen nicht garantiert erfolgreich sein, aber die Lernenden sollten an die Grenzen von MT erinnert werden, wenn kein solcher "Goldstandard" verfügbar ist.
- Verwendung der abgeleiteten MRs und der entworfenen Ausgangstestfälle zur Ableitung von Folgetestfällen.
- Durchführung der Folgetestfälle.

9.6 Erfahrungsbasiertes Testen KI-basierter Systeme

Erfahrungsbasiertes Testen umfasst intuitive Testfallermittlung, exploratives Testen und checklistenbasiertes Testen [I01], die alle auf das Testen KI-basierter Systeme angewendet werden können.

Intuitive Testfallermittlung basiert in der Regel auf dem Wissen von Testern über typische Entwicklerfehlhandlungen und Fehlerwirkungen in ähnlichen Systemen (oder früheren Versionen). Ein Beispiel für die intuitive Testfallermittlung bei KI-basierten Systemen könnte die Verwendung von Wissen darüber sein, wie ML-Systeme in der Vergangenheit aufgrund der Verwendung von systematisch verzerrten Trainingsdaten versagt haben.

Bei explorativen Tests werden Tests iterativ entworfen und durchgeführt, wobei die Möglichkeit besteht, spätere Tests auf der Grundlage der Testergebnisse früherer Tests abzuleiten. Exploratives Testen ist besonders nützlich, wenn es schlechte Spezifikationen oder Probleme mit dem Testorakel gibt, was bei KI-basierten Systemen häufig der Fall ist. Daher werden explorative Tests in diesem Zusammenhang häufig eingesetzt und sollten als Ergänzung zu den systematischeren Tests auf der Grundlage von Verfahren wie dem metamorphen Testen (siehe Abschnitt 9.5) verwendet werden.

Eine Tour ist eine Metapher, die für eine Reihe von Strategien und Zielen verwendet wird, auf die sich Tester beziehen können, wenn explorative Tests um einen bestimmten Schwerpunkt herum organisiert sind [B20]. Typische Touren für das explorative Testen KI-basierter Systeme könnten sich auf die Konzepte der Verzerrung, Unteranpassung und Überanpassung in ML-Systemen konzentrieren. Zum Beispiel könnte eine Datentour zum Testen des Modells verwendet werden. Bei dieser Tour könnte der Tester verschiedene Arten von Daten, die für das Training verwendet wurden, ihre Verteilung, ihre Variationen, ihr Format und ihre Bereiche usw. identifizieren und dann diese Arten von Daten zum Testen des Modells verwenden.

ML-Systeme sind in hohem Maße von der Qualität der Trainingsdaten abhängig, und der bestehende Bereich der EDA ist eng mit dem Ansatz des explorativen Testens verbunden. Bei der EDA werden Daten auf Muster, Beziehungen, Trends und Ausreißer untersucht. Sie beinhaltet die interaktive, hypothesengesteuerte Untersuchung von Daten und wird in [B21] wie folgt beschrieben: "Wir untersuchen Daten mit Erwartungen. Wir revidieren unsere Erwartungen auf der Grundlage dessen, was wir in den Daten sehen. Und wir iterieren diesen Prozess." EDA erfordert in der Regel Werkzeugunterstützung in zwei Bereichen: für die Interaktion mit den Daten, damit die Analysten komplexe Daten besser verstehen können, und für die Datenvisualisierung, damit sie die Analyseergebnisse einfach darstellen können. Der Einsatz explorativer Verfahren, die in erster Linie durch die Datenvisualisierung angetrieben werden, kann dazu beitragen, den verwendeten ML-Algorithmus zu validieren, Änderungen zu identifizieren, die zu effizienten Modellen führen, und Fachwissen zu nutzen [B22].

Google hat eine Liste mit achtundzwanzig ML-Tests in Form von Aussagen in den Bereichen Daten, Modellentwicklung, Infrastruktur und Überwachung, die als Test-Checkliste bei Google für ML-Systeme

verwendet werden [B23]. Die "ML-Test-Checkliste" von Google wird hier in der von Google veröffentlichten Form vorgestellt:

ML-Daten:

1. Die Erwartungen an ein Merkmal werden in einem Schema festgehalten.
2. Alle Merkmale sind vorteilhaft.
3. Kein Merkmal ist zu teuer.
4. Die Merkmale halten die Anforderungen auf einer Metaebene ein.
5. Die Daten-Pipeline verfügt über angemessene Datenschutzkontrollen.
6. Neue Merkmale können schnell hinzugefügt werden.
7. Der gesamte eingegebene Merkmalscode wird getestet.

Modellentwicklung:

1. Die Modellspezifikationen werden geprüft und vorgelegt.
2. Die Offline- und Online-Metriken korrelieren.
3. Alle Hyperparameter wurden abgestimmt.
4. Die Auswirkungen unzureichender Aktualität des Modells sind bekannt.
5. Ein einfacheres Modell ist nicht besser.
6. Die Qualität des Modells ist in wichtigen Datenbereichen ausreichend.
7. Das Modell wird auf Inklusionsaspekte geprüft.

ML-Infrastruktur:

1. Das Training ist reproduzierbar.
2. Die Modellspezifikationen sind mit Unittests geprüft.
3. Die ML-Pipeline wird einem Integrationstest unterzogen.
4. Die Qualität des Modells wird vor seinem Einsatz überprüft.
5. Das Modell kann auf Fehlerzustände untersucht werden.
6. Die Modelle werden vor dem Einsatz validiert.
7. Eingesetzte Modelle können in den Ausgangszustand gebracht werden

Überwachungstests:

1. Änderungen der Abhängigkeiten führen zu einer Benachrichtigung.
2. Die Dateninvarianzen gelten für die Eingaben.
3. Das Training und der Einsatz sind nicht verzerrt.
4. Die Modelle sind nicht zu alt.
5. Die Modelle sind nummerisch stabil.
6. Die Rechenleistung hat sich nicht verschlechtert.
7. Die Qualität der Vorhersagen hat sich nicht verschlechtert.

9.6.1 Praktische Übung: Exploratives Testen und explorative Datenanalyse (EDA)

Für ein ausgewähltes Modell und einen Datensatz führen die Lernenden eine Datentour durch, bei der sie verschiedene Datentypen und deren Verteilung für verschiedene Parameter berücksichtigen.

Die Lernenden führen eine EDA mit den Daten durch, um fehlende Daten und/oder mögliche Verzerrungen in den Daten zu identifizieren.

9.7 Auswahl von Testverfahren für KI-basierte Systeme

Ein KI-basiertes System umfasst typischerweise sowohl KI- als auch Nicht-KI-Komponenten. Die Auswahl der Testverfahren für das Testen der Nicht-KI-Komponenten ist im Allgemeinen dieselbe wie bei konventionellem Testen. Für die KI-basierten Komponenten kann die Auswahl eingeschränkter sein. Liegt beispielsweise ein Testorakelproblem vor (d. h., die Generierung der erwarteten Ergebnisse ist schwierig), so kann dieses Problem auf der Grundlage der wahrgenommenen Risiken durch den Einsatz des Folgenden abgemildert werden:

- Vergleichendes Testen: Hierfür müssen Testfälle verfügbar sein oder generiert werden, und ein gleichwertiges System muss als Pseudo-Orakel dienen, das für Regressionstests eine frühere Version des Systems sein kann. Zur effektiven Erkennung von Fehlerzuständen kann ein unabhängig entwickeltes System erforderlich sein.
- A/B-Testen: Hierbei werden häufig Eingaben aus dem Produktivbetrieb als Testfälle verwendet und normalerweise zwei Varianten desselben Systems mit Hilfe statistischer Analysen verglichen. A/B-Testen kann verwendet werden, um zu prüfen, ob eine neue Variante datenverunreinigt ist, oder um automatisierte Regressionstests für ein selbstlernendes System durchzuführen.
- Metamorphes Testen: Dies kann von unerfahrenen Testern verwendet werden, um kostengünstig Fehlerzustände zu finden, obwohl sie die Anwendungsdomäne verstehen müssen. MT ist nicht geeignet, um endgültige Ergebnisse zu liefern, da die erwarteten Ergebnisse nicht absolut sind, sondern relativ zum Ausgangstestfall. Kommerzielle Werkzeuge sind derzeit nicht verfügbar, aber viele Tests können manuell erstellt werden.

Gegnerisches Testen ist in der Regel für ML-Modelle geeignet, bei denen der falsche Umgang mit gegnerischen Beispielen erhebliche Auswirkungen haben könnte, oder bei denen das System angegriffen werden kann. In ähnlicher Weise kann das Testen auf Datenverunreinigung für ML-Systeme geeignet sein, wenn das System angegriffen werden kann.

Für komplexe KI-basierte Systeme mit mehreren Parametern ist paarweises Testen oft angebracht.

Erfahrungsbasiertes Testen eignet sich häufig für das Testen KI-basierter Systeme, insbesondere wenn es um die Berücksichtigung der für das Training verwendeten Daten und der Daten aus dem Produktivbetrieb geht. EDA kann zur Validierung des verwendeten ML-Algorithmus, zur Ermittlung von Effizienzsteigerungen und zur Nutzung von Fachwissen verwendet werden. Google hat festgestellt, dass seine ML-Test-Checkliste ein effektiver Ansatz für ML-Systeme ist.

Im speziellen Bereich der neuronalen Netze ist die Überdeckung des Netzes oft für unternehmenskritische Systeme geeignet, wobei einige Überdeckungskriterien eine gründlichere Überdeckung erfordern als andere.

10 Testumgebungen für KI-basierte Systeme - 30 Minuten

Schlüsselwörter

Virtuelle Testumgebung

KI-spezifische Schlüsselwörter

KI-spezifischer Prozessor, autonomes System, Big Data, Erklärbarkeit, Multiagentensystem, selbstlernendes System

Lernziele für Kapitel 10:

10.1 Testumgebungen für KI-basierte Systeme

AI-10.1.1 K2 Beschreiben der wichtigsten Faktoren, durch die sich die Testumgebungen für KI-basierte Systeme von denen für konventionelle Systeme unterscheiden

10.2 Virtuelle Testumgebungen für KI-basierte Systeme

AI-10.2.1 K2 Beschreiben der Vorteile einer virtuellen Testumgebung für das Testen KI-basierter Systeme

10.1 Testumgebungen für KI-basierte Systeme

KI-basierte Systeme können in einer Vielzahl von Einsatzumgebungen eingesetzt werden, was bedeutet, dass auch die Testumgebungen ähnlich vielfältig sind. Zu den Merkmalen KI-basierter Systeme, die dazu führen können, dass sich die Testumgebungen von denen konventioneller Systeme unterscheiden, gehören:

- Selbstlernend: Von selbstlernenden Systemen und einigen autonomen Systemen wird erwartet, dass sie sich an sich ändernde Einsatzumgebungen anpassen, die bei der ursprünglichen Einführung des Systems möglicherweise noch nicht vollständig definiert waren (siehe Abschnitt 2.1). Folglich ist die Definition von Testumgebungen, die diese nicht definierten Umgebungsveränderungen nachahmen können, von Natur aus schwierig und kann sowohl die Vorstellungskraft der Tester als auch ein in die Testumgebung eingebautes Maß an Zufälligkeit erfordern.
- Autonomie: Von autonomen Systemen wird erwartet, dass sie auf Veränderungen in ihrer Umgebung ohne menschliches Eingreifen reagieren und auch Situationen erkennen, in denen die Autonomie an das menschliche Bedienpersonal zurückgegeben werden sollte (siehe Abschnitt 2.22). Bei einigen Systemen kann die Ermittlung und Nachahmung der Umstände, unter denen die Autonomie abgegeben werden muss, erfordern, dass die Testumgebungen die Systeme an ihre Grenzen bringen. Einige autonome Systeme sollen in gefährlichen Umgebungen arbeiten, und die Einrichtung repräsentativer, gefährlicher Testumgebungen kann eine Herausforderung sein.
- Multiagenten: Wenn KI-basierte Multiagentensysteme mit anderen KI-basierten Systemen zusammenarbeiten sollen, muss die Testumgebung möglicherweise ein gewisses Maß an Nicht-Determiniertheit enthalten, damit sie die Nicht-Determiniertheit der KI-basierten Systeme nachahmen kann, mit denen das SUT interagiert.
- Erklärbarkeit: Bei einigen KI-basierten Systemen kann es schwierig sein, festzustellen, wie das System seine Entscheidungen getroffen hat (siehe Abschnitt 2.87). Wenn es wichtig ist, dies vor dem Einsatz zu verstehen, muss die Testumgebung möglicherweise Werkzeuge enthalten, die erklären, wie die Entscheidungen getroffen werden.
- Hardware: Einige Hardware, die für KI-basierte Systeme verwendet wird, ist speziell für diesen Zweck konzipiert, wie z. B. KI-spezifische Prozessoren (siehe Abschnitt 1.6). Die Notwendigkeit, solche Hardware in die Testumgebung einzubeziehen, sollte im Rahmen der entsprechenden Testplanung berücksichtigt werden.
- Big Data: Wenn von einem KI-basierten System erwartet wird, dass es große Datenmengen verbraucht (z. B. Daten mit hohem Volumen, hoher Datenrate und/oder hoher Varianz), muss die Einrichtung dieser Daten als Teil einer Testumgebung sorgfältig geplant und umgesetzt werden (siehe Abschnitt 7.3).

10.2 Virtuelle Testumgebungen für KI-basierte Systeme

Der Einsatz einer virtuellen Testumgebung beim Testen eines KI-basierten Systems bringt folgende Vorteile mit sich:

- Gefährliche Szenarien: Diese können getestet werden, ohne das SUT, andere interagierende Systeme, einschließlich Menschen, oder die Einsatzumgebung (z.B. Bäume, Gebäude) zu gefährden.
- Ungewöhnliche Szenarien: Diese können getestet werden, wenn es andernfalls sehr zeitaufwändig oder teuer wäre, diese Szenarien für den realen Betrieb einzurichten (z. B. Warten auf ein seltenes Ereignis, wie eine vollständige Sonnenfinsternis oder vier Busse, die gleichzeitig in dieselbe Straßenkreuzung einfahren). Ebenso können Grenzfälle, die in der realen Welt nur schwer zu realisieren sind, in einer virtuellen Testumgebung leichter, wiederholbar und reproduzierbar erstellt werden.

- Extreme Szenarien: Diese können getestet werden, wenn es teuer oder unmöglich wäre, sie in der Realität einzurichten (z. B. bei einer Nuklearkatastrophe oder der Erforschung des Weltraums).
- Zeitintensive Szenarien: Diese können in einer virtuellen Umgebung in kürzeren Zeitabständen (z. B. mehrmals pro Sekunde) getestet werden. Im Gegensatz dazu kann es Stunden oder Tage dauern, sie einzurichten und in Echtzeit auszuführen. Ein weiterer Vorteil ist, dass mehrere virtuelle Testumgebungen parallel betrieben werden können. Dies geschieht in der Regel in der Cloud und ermöglicht die gleichzeitige Ausführung vieler Szenarien, was mit der tatsächlichen Systemhardware möglicherweise nicht möglich ist.
- Beobachtbarkeit und Steuerbarkeit: Virtuelle Testumgebungen bieten eine weitaus bessere Steuerbarkeit der Testumgebung. Sie können beispielsweise sicherstellen, dass ungewöhnliche Bedingungen im Finanzhandel nachgebildet werden. Darüber hinaus bieten sie eine weitaus bessere Beobachtbarkeit, da alle digital bereitgestellten Teile der Umgebung kontinuierlich überwacht und aufgezeichnet werden können.
- Verfügbarkeit: Die Simulation von Hardware durch virtuelle Testumgebungen ermöglicht es, Systeme mit (simulierten) Hardwarekomponenten zu testen, die andernfalls nicht zur Verfügung stünden, etwa weil sie noch nicht entwickelt wurden oder zu teuer sind.

Virtuelle Testumgebungen können speziell für ein bestimmtes System erstellt werden, sie können generisch sein oder zur Unterstützung bestimmter Anwendungsbereiche entwickelt werden. Zum Testen KI-basierter Systeme sind u. A. die folgenden kommerziellen wie auch quelloffenen virtuellen Testumgebungen verfügbar:

- Morse: Die Modular Open Robots Simulation Engine ist ein Simulator für die generische Simulation von einzelnen oder mehreren mobilen Robotern, basierend auf der Blender Game Engine [R24].
- AI Habitat: Hierbei handelt es sich um eine von Facebook KI entwickelte Simulationsplattform, mit der verkörperte Agenten (z. B. virtuelle Roboter) in fotorealistischen 3D-Umgebungen trainiert werden können [R25].
- DRIVE Constellation: Dies ist eine offene und skalierbare Plattform für selbstfahrende Autos von NVIDIA. Sie basiert auf einer Cloud-basierten Plattform und ist in der Lage, Milliarden von Kilometern an autonomen Fahrzeugtests zu generieren [R26].
- MATLAB und Simulink: Sie ermöglichen die Aufbereitung von Trainingsdaten, die Erstellung von ML-Modellen und die Simulation der Ausführung KI-basierter Systeme einschließlich der Modelle anhand synthetischer Daten [R27].

11 Einsatz von KI für Tests - 195 Minuten

Schlüsselwörter

visuelles Testen

KI-spezifische Schlüsselwörter

Bayessches Verfahren, Klassifikation, Clusterbildungsalgorithmus, Fehlervorhersage, grafische Benutzungsschnittstelle (*graphical user interface*, GUI)

Lernziele für Kapitel 11:

11.1 KI-Techniken für das Testen

AI-11.1.1 K2 Kategorisieren der bei Softwaretests verwendeten KI-Techniken

HO-11.1.1 H2 Erläutern von Tätigkeiten im Testbereich, bei denen der Einsatz von KI weniger wahrscheinlich ist, an Hand von Beispielen

11.2 Anwendung von KI zur Analyse gemeldeter Fehlerzustände

AI-11.2.1 K2 Erläutern, wie KI bei der Analyse neuer Fehlerzustände unterstützen kann

11.3 Einsatz von KI für die Testfallerstellung

AI-11.3.1 K2 Erläutern, wie KI bei der Erstellung von Testfällen helfen kann

11.4 Einsatz von KI für die Optimierung von Regressionstestsuiten

AI-11.4.1 K2 Erklären, wie KI bei der Optimierung von Regressionstestsuiten helfen kann

11.5 Einsatz von KI für die Fehlervorhersage

AI-11.5.1 K2 Erklären, wie KI bei der Fehlervorhersage helfen kann

HO-11.5.1 H2 Implementieren eines einfachen KI-basierten Fehlerprognosesystems

11.6 Einsatz von KI zum Testen von Benutzungsschnittstellen

AI-11.6.1 K2 Erklären des Einsatzes von KI beim Testen von Benutzungsschnittstellen

11.1 KI-Techniken für das Testen

In Abschnitt 1.4 werden mehrere KI-Techniken aufgeführt, die alle zur Unterstützung eines bestimmten Aspekts des Softwaretests eingesetzt werden können. Laut Harman [B24] verwendet die Softwareentwicklungs-Community drei große Bereiche von KI-Techniken:

- Fuzzy-Logik und probabilistische Methoden: Hier geht es um den Einsatz von KI-Techniken zur Bewältigung von probabilistischen Problemen der realen Welt. So kann KI beispielsweise zur Analyse und Vorhersage möglicher Systemausfälle unter Verwendung des bayesschen Verfahren eingesetzt werden. Diese können die Wahrscheinlichkeit des Ausfalls von Komponenten oder Funktionen abschätzen oder die potenziell zufällige Natur der menschlichen Interaktionen mit dem System widerspiegeln.
- Klassifikation, Lernen und Vorhersage: Dies kann für verschiedene Anwendungsfälle genutzt werden, z. B. für die Vorhersage von Kosten im Rahmen der Projektplanung oder der Vorhersage von Fehlern. In der Form von ML wird dieser Bereich für viele Softwaretests verwendet, einschließlich Fehlermanagement (siehe Abschnitt 11.2), Fehlervorhersage (siehe Abschnitt 11.5) und Testen von Benutzungsschnittstellen (siehe Abschnitt 11.6).
- Computergestützte Such- und Optimierungsverfahren: Diese können zur Lösung von Optimierungsproblemen durch rechnerische Suche in potenziell großen und komplexen Suchräumen (z. B. mit Suchalgorithmen) eingesetzt werden. Beispiele hierfür sind die Generierung von Testfällen (siehe Abschnitt 11.3), die Identifizierung der kleinsten Anzahl von Testfällen, die ein bestimmtes Überdeckungskriterium erfüllen, und die Optimierung von Regressionstestfällen (siehe Abschnitt 11.4).
- Die obige Kategorisierung ist notwendigerweise weit gefasst, da es beträchtliche Überschneidungen zwischen den Testaufgaben, die durch KI durchgeführt werden können, und den verschiedenen KI-Techniken gibt. Es handelt sich auch nur um eine mögliche Kategorisierung, andere sind genauso denkbar.

11.1.1 Praktische Übung: Der Einsatz von KI bei Tests

Im Rahmen einer Diskussion sollen die Lernenden Testaktivitäten und -aufgaben identifizieren, die derzeit für die Implementierung von KI unpraktikabel sind. Dazu könnten gehören:

- Spezifikation des Testorakels.
- Kommunikation mit den Beteiligten, um Unklarheiten zu klären und fehlende Informationen zu erhalten.
- Vorschläge zur Verbesserung des Benutzererlebnisses.
- Annahmen der Beteiligten hinterfragen und unbequeme Fragen stellen.
- Verständnis für die Bedürfnisse der Nutzer.

Es ist zu unterscheiden zwischen schwacher KI, die für einige begrenzte Aufgaben eingesetzt werden könnte, und allgemeiner KI, die derzeit nicht verfügbar ist (siehe Abschnitt 1.2).

11.2 Einsatz von KI zur Analyse gemeldeter Fehler

Gemeldete Fehler werden in der Regel kategorisiert, nach Prioritäten geordnet und eventuelle Duplikate werden identifiziert. Diese Tätigkeit wird oft als Fehler-Triage oder -analyse bezeichnet und soll die für die Fehlerbehebung aufgewendete Zeit optimieren. KI kann zur Unterstützung dieser Tätigkeit auf verschiedene Weise eingesetzt werden, z. B.:

- Kategorisierung: Die Verarbeitung natürlicher Sprache (*natural language processing*, NLP) [B25] kann verwendet werden, um Text in Fehlerberichten zu analysieren und Themen zu extrahieren, wie z. B. den Bereich der betroffenen Funktionalität, die dann zusammen mit

anderen Metadaten an Clusterbildungsalgorithmen wie k-nächster Nachbar oder Stützvektormaschinen weitergegeben werden können. Diese Algorithmen können geeignete Fehlerkategorien identifizieren und ähnliche oder doppelte Fehlermeldungen hervorheben. Die KI-basierte Kategorisierung ist besonders nützlich für automatisierte Fehlermeldesysteme (z. B. für Microsoft Windows und für Firefox) und für große Projekte mit vielen Softwareingenieuren.

- Kritikalität: Anhand von ML-Modellen, die auf den Merkmalen der kritischsten Fehler trainiert wurden, können die Fehler identifiziert werden, die am wahrscheinlichsten die Systemausfälle verursachen, die einen großen Anteil der gemeldeten Fehler ausmachen [B26].
- Zuweisung: ML-Modelle können vorschlagen, welche Entwickler am besten geeignet sind, bestimmte Fehler zu beheben, und zwar auf der Grundlage des Inhalts der Fehlermeldung und früherer Entwicklerzuordnungen.

11.3 Einsatz von KI für die Testfallgenerierung

Der Einsatz von KI zur Testfallgenerierung kann ein sehr effektives Verfahren sein, um schnell Testressourcen zu erstellen und die Überdeckung zu maximieren (z. B. Code- oder Anforderungsüberdeckung). Die Grundlage für die Erstellung dieser Testfälle bilden der Quellcode, die Benutzungsschnittstelle und ein maschinenlesbares Testmodell. Einige Werkzeuge basieren die Testfälle auch auf der Beobachtung des Low-Level-Verhaltens des Systems durch Instrumentierung oder durch Logdateien [B27].

Wenn jedoch kein Testmodell, das die erforderlichen Verhaltensweisen definiert, als Grundlage für die Testfälle verwendet wird, leidet diese Form der Testfallgenerierung im Allgemeinen unter einem Testorakelproblem, da das KI-basierte Werkzeug nicht weiß, welche Ergebnisse für einen bestimmten Satz von Testdaten zu erwarten sind. Eine Lösung ist die Verwendung von vergleichendem Testen (*back-to-back testing*) , wenn ein geeignetes System zur Verfügung steht, das als Pseudo-Orakel verwendet werden kann (siehe Abschnitt 9.3). Alternativ könnten Tests mit dem erwarteten Ergebnis durchgeführt werden, dass weder eine "nicht reagierende Anwendung" noch ein Systemabsturz oder andere ähnlich einfache Fehlerindikatoren auftreten.

Forschungen, die KI-basierte Testfallgenerierungswerkzeuge mit ähnlichen, nicht auf KI basierenden Fuzz-Testing-Werkzeugen vergleichen, zeigen, dass die KI-basierten Werkzeuge einen gleichwertigen Überdeckungsgrad erreichen und mehr Fehlerzustände finden können, während sie die durchschnittliche Abfolge der Schritte, die für eine Fehlerwirkung erforderlich sind, von durchschnittlich 15.000 Schritten auf etwa 100 Schritte reduzieren. Dies macht das Debugging wesentlich einfacher [B27].

11.4 Einsatz von KI für die Optimierung von Regressionstestsuiten

Wenn Änderungen an einem System vorgenommen werden, werden neue Tests erstellt, ausgeführt und kommen für eine Regressionstestsuite in Frage. Um zu verhindern, dass Regressionstestsuiten zu groß werden, sollten sie häufig optimiert werden. Dabei werden Testfälle ausgewählt, priorisiert und sogar erweitert, um eine effektivere und effizientere Regressionstestsuite zu erstellen.

Ein KI-basiertes Tool kann die Regressionstestsuite optimieren, indem es z. B. die Informationen aus früheren Testergebnissen, die zugehörigen Fehlerzustände und die letzten Änderungen analysiert, z. B. welche Merkmale häufiger fehlerhaft sind und welche Tests den von den letzten Änderungen betroffenen Code testen.

Untersuchungen zeigen, dass die Größe einer Regressionstestsuite um 50 % reduziert werden kann, während die meisten Fehlerzustände weiterhin erkannt werden [B28], und dass bei kontinuierlichen

Integrationstests eine Verringerung der Testausführungsdauer um 40 % erreicht werden kann, ohne dass die Fehlerzustandserkennung wesentlich beeinträchtigt wird [B29].

11.5 Einsatz von KI für die Fehlervorhersage

Mit Hilfe der Fehlervorhersage lässt sich vorhersagen, ob ein Fehlerzustand vorhanden ist, wie viele Fehlerzustände vorhanden sind oder ob Fehlerzustände gefunden werden können. Diese Fähigkeit hängt von der Ausgereiftheit des verwendeten Werkzeugs ab. Die Ergebnisse werden in der Regel dazu verwendet, Prioritäten für die Tests zu setzen (z. B. mehr Tests für die Komponenten, für die mehr Fehlerzustände vorhergesagt werden).

Die Fehlervorhersage basiert in der Regel auf Quellcode-Metriken, Prozess-Metriken und/oder menschlichen und organisatorischen Metriken. Da es so viele potenzielle Faktoren zu berücksichtigen gilt, übersteigt die Bestimmung der Beziehung zwischen diesen Faktoren und der Wahrscheinlichkeit von Fehlerzuständen die menschlichen Fähigkeiten. Daher ist die Verwendung eines KI-basierten Ansatzes, der in der Regel ML verwendet, eine Notwendigkeit. Die Vorhersage von Fehlerzuständen ist am effektivsten, wenn sie auf früheren Erfahrungen in einer ähnlichen Situation beruht (z. B. mit derselben Codebasis und/oder denselben Entwicklern).

Die Fehlervorhersage mit Hilfe von ML wurde bereits in verschiedenen Situationen erfolgreich eingesetzt (z. B. [B30] und [B31]). Es hat sich gezeigt, dass die besten Prädiktoren Personen und organisatorische Maße sind und nicht die weit verbreiteten Quellcode-Metriken wie die Anzahl der Codezeilen und die zyklomatische Komplexität [B32].

11.5.1 Praktische Übung: Aufbau eines Fehlervorhersagesystems

Die Lernenden verwenden einen geeigneten Datensatz (z. B. mit Quellcode-Metriken und entsprechenden Fehlerdaten), um ein einfaches Modell zur Fehlervorhersage zu erstellen, und verwenden es, um die Wahrscheinlichkeit von Fehlerzuständen anhand von Quellcode-Metriken aus ähnlichem Code vorherzusagen.

Das Modell sollte mindestens vier Merkmale aus dem Datensatz verwenden, und die Lernenden sollten die Ergebnisse mit verschiedenen Merkmalen untersuchen, um deutlich zu machen, wie sich die Ergebnisse je nach den ausgewählten Merkmalen ändern.

11.6 Einsatz von KI zum Testen von Benutzungsschnittstellen

11.6.1 Einsatz von KI zum Testen über die grafische Benutzungsschnittstelle (GUI)

Das Testen über die grafische Benutzungsschnittstelle ist der typische Ansatz für manuelle Tests (mit Ausnahme von Komponententests) und bildet häufig den Ausgangspunkt für Initiativen zur Testautomatisierung. Die daraus resultierenden Tests bilden die menschliche Interaktion mit dem Testobjekt nach. Diese skriptgesteuerte Testautomatisierung kann durch Anwendung eines Capture/Playback-Ansatzes implementiert werden, wobei entweder die tatsächlichen Koordinaten der Elemente der Benutzungsschnittstelle oder die softwaredefinierten Objekte/Widgets der Schnittstelle verwendet werden. Dieser Ansatz hat jedoch mehrere Nachteile bei der Objektidentifizierung, einschließlich der Empfindlichkeit gegenüber Änderungen an der Schnittstelle, Codeänderungen und Plattformänderungen.

Mit Hilfe von KI kann die Empfindlichkeit dieses Ansatzes verringert werden, indem KI-basierte Tools eingesetzt werden, um die richtigen Objekte anhand verschiedener Kriterien (z. B. XPath, Label, ID,

Klasse, X/Y-Koordinaten) zu identifizieren und die historisch stabilsten Identifikationskriterien auszuwählen. So kann sich beispielsweise die ID einer Schaltfläche in einem bestimmten Bereich der Anwendung mit jeder Version ändern, so dass das KI-basierte Tool dieser ID im Laufe der Zeit eine geringere Wichtigkeit beimisst und sich stärker auf andere Kriterien verlässt. Bei diesem Ansatz werden die Objekte in der Benutzungsschnittstelle als zu dem Test passend oder nicht passend klassifiziert.

Alternativ dazu wird bei visuellen Tests die Bilderkennung verwendet, um mit GUI-Objekten über dieselbe Schnittstelle wie ein tatsächlicher Benutzer zu interagieren, so dass kein Zugriff auf den zugrunde liegenden Code und die Schnittstellendefinitionen erforderlich ist. Dies macht sie völlig nicht-intrusiv und unabhängig von der zugrundeliegenden Technike. Die Skripte müssen nur mit der sichtbaren Benutzungsschnittstelle arbeiten. Dieser Ansatz ermöglicht es dem Tester, Skripte zu erstellen, die direkt mit den Bildern, Schaltflächen und Textfeldern auf dem Bildschirm interagieren, und zwar auf die gleiche Weise wie ein menschlicher Benutzer, ohne vom allgemeinen Bildschirmlayout beeinflusst zu werden. Die Verwendung von Bilderkennung in der Testautomatisierung kann durch die benötigten Computerressourcen eingeschränkt werden. Dank der Verfügbarkeit erschwinglicher KI, die eine hochentwickelte Bilderkennung unterstützt, ist dieser Ansatz nun jedoch für den allgemeinen Einsatz geeignet.

11.6.2 Einsatz von KI zum Testen der GUI

ML-Modelle können verwendet werden, um die Annehmbarkeit von Fenstern der Benutzungsschnittstelle zu bestimmen (z. B. durch Heuristik und überwachtes Lernen). Tools, die auf diesen Modellen basieren, können falsch gerenderte Elemente identifizieren, feststellen, ob einige Objekte unzugänglich oder schwer zu erkennen sind, und verschiedene andere Probleme mit dem visuellen Erscheinungsbild der GUI erkennen.

Neben der Bilderkennung können auch andere Formen des KI-basierten maschinellen Sehens verwendet werden, um Bilder (z. B. Screenshots) zu vergleichen und unbeabsichtigte Änderungen des Layouts, der Größe, Position, Farbe, Schriftart oder anderer sichtbarer Attribute von Objekten zu identifizieren. Die Ergebnisse dieser Vergleiche können zur Unterstützung von Regressionstests herangezogen werden, um zu prüfen, ob sich Änderungen am Testobjekt negativ auf die Benutzungsschnittstelle ausgewirkt haben.

Die Technik zur Überprüfung der Annehmbarkeit von Fenstern kann mit Vergleichstools kombiniert werden, um anspruchsvollere KI-basierte Regressionstests zu erstellen, die abschätzen können, ob erkannte Änderungen an der Benutzungsschnittstelle für die Benutzer akzeptabel sind oder ob diese Änderungen zur Überprüfung durch einen Menschen gekennzeichnet werden sollten. Solche KI-basierten Tools können auch zur Unterstützung von Kompatibilitätstests auf verschiedenen Browsern, Geräten oder Plattformen eingesetzt werden, um zu prüfen, ob die Benutzungsschnittstelle für dieselbe Anwendung auf verschiedenen Browsern, Geräten oder Plattformen korrekt funktioniert.

12 Referenzen

12.1 Normen und Standards [S]

[S01] ISO/IEC TR 29119-11:2020, Software and systems engineering — Software testing — Part 11 Guidelines on the testing of AI-based systems, https://www.iso.org/standard/79016.html (Zugriff 2021)

[S02] DIN SPEC 92001-1, Künstliche Intelligenz - Lebenszyklus-Prozesse und Qualitätsanforderungen - Teil 1: Qualitäts-Metamodell, https://www.din.de/en/wdc-beuth:din21:303650673 (Zugriff 2021).

[S03] DIN SPEC 92001-2, Künstliche Intelligenz - Lebenszyklusprozesse und Qualitätsanforderungen - Teil 2: Technische und organisatorische Anforderungen, https://www.din.de/en/innovation-and-research/din-spec-en/projects/wdc-proj:din21:298702628 (Zugriff 2021).

[S04] ISO 26262:2018, Road Vehicles – Functional Safety Part 1: Vocabulary https://www.iso.org/standard/68383.html (Zugriff 2021)

[S05] ISO/PAS 21448:2019, Road vehicles — Safety of the intended functionality (SOTIF) - https://www.iso.org/standard/70939.html (Zugriff 2021)

[S06] ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models https://www.iso.org/standard/35733.html (Zugriff 2021)

[S07] ISO 26262-6:2018, Road vehicles — Functional safety — Part 6: Product development at the software level https://www.iso.org/standard/68388.html (Zugriff 2021)

[S08] ISO/IEC/IEEE 29119-4:2021, Software and systems engineering — Software testing — Part 4: Test techniques https://www.iso.org/standard/79430.html (Zugriff 2021)

12.2 ISTQB®-Dokumente [I]

[I01] ISTQB® Certified Tester Foundation Level Syllabus, Version 2018 V3.1 https://www.istqb.org/downloads/category/2-foundation-level-documents.html (Zugriff 2021).

[I02] ISTQB® Certified Tester Advanced Level Test Analyst Syllabus, Version 3.1, Abschnitt 3.2.6 https://www.istqb.org/downloads/category/75-advanced-level-test-analyst-v3-1.html (Zugriff im August 2021).

[I03] ISTQB® Certified Tester AI Testing, Übersicht über den Lehrplan, Version 1.0

12.3 Bücher und Artikel [B]

[B01] Cadwalladr, Carole (2014), "Are the robots about to rise? Google's new director of engineering thinks so …" The Guardian. Guardian News and Media Limited, https://www.theguardian.com/technology/2014/feb/22/robots-google-ray-kurzweil-terminator-singularity-artificial-intelligence (Zugriff 2021).

[B02] Stuart Russell und Peter Norvig, Artificial Intelligence: A Modern Approach, 4. Auflage, Pearson, 2020.

[B03] M. Davies et al., "Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook," Proceedings of the IEEE, vol. 109, no. 5, pp. 911-934, Mai 2021, doi: 10.1109/JPROC.2021.3067593.

[B04] Chris Wiltz, Can Apple Use Its Latest AI Chip for More Than Photos?, Electronics & Test, Artificial Intelligence, https://www.designnews.com/electronics-test/can-apple-use-its-latest-ai-chip-more-photos/153617253461497 (Zugriff 2021).

[B05] HUAWEI Reveals the Future of Mobile AI at IFA 2017, Huawei Press Release, https://consumer.huawei.com/en/press/news/2017/ifa2017-kirin970/ (Zugriff 2021).

[B06] VERORDNUNG (EU) 2016/679 DES EUROPÄISCHEN PARLAMENTS UND DES RATES zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG (Datenschutz-Grundverordnung), April 2016, https://eur-lex.europa.eu/eli/reg/2016/679/oj (Zugriff Mai 2021)

[B07] Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016_201806, SAE, ,https://www.sae.org/standards/content/j3016_201806/ (Zugriff 2021).

[B08] G20-Ministererklärung zu Handel und digitaler Wirtschaft: Annex. Verfügbar unter: https://www.mofa.go.jp/files/000486596.pdf (Zugriff im 2021).

[B09] Concrete Problems in AI Safety, Dario Amodei (Google Brain), Chris Olah (Google Brain), Jacob Steinhardt (Stanford University), Paul Christiano (UC Berkeley), John Schulman (OpenAI), Dan Man'e (Google Brain), März 2016. https://arxiv.org/pdf/1606.06565 (Zugriff 2021).

[B10] Explainable AI: the basics, Policy briefing, Issued: November 2019 DES6051, ISBN: 978-1-78252-433-5, The Royal Society.

[B11] The Ultimate Guide to Data Labeling for Machine Learning, www.cloudfactory.com/data-labeling-guide (Zugriff 2021).

[B12] Pei et al, DeepXplore: Automated Whitebox Testing of Deep Learning Systems, Proceedings of ACM Symposium on Operating Systems Principles (SOSP '17), Jan 2017.

[B13] Sun et al, Testing Deep Neural Networks, https://www.researchgate.net/publication/323747173_Testing_Deep_Neural_Networks (Zugriff 2021).

[B14] A. Odena und I. Goodfellow, TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing, ArXiv e-prints, Jul. 2018, https://arxiv.org/pdf/1807.10875 (Zugriff 2021)

[B15] Riccio, V. et al, Testing Machine Learning based Systems: A Systematic Mapping. Empirical Software Engineering, https://link.springer.com/article/10.1007/s10664-020-09881-0 (Zugriff 2021)

[B16] Baudel, Thomas et al, Addressing Cognitive Biases in Augmented Business Decision Systems. https://arxiv.org/abs/2009.08127 (Zugriff 2021)

[B17] Papernot, N. et al, Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, arXiv preprint arXiv:1605.07277, 2016. https://arxiv.org/pdf/1605.07277 (Zugriff 2021).

[B18] Chen et al, Metamorphic Testing: A Review of Challenges and Opportunities, ACM Comput. Surv. 51, 1, Article 4, January 2018. https://www.researchgate.net/publication/322261865_Metamorphic_Testing_A_Review_of_Challenges_and_Opportunities (Zugriff 2021).

[B19] Huai Liu, Fei-Ching Kuo, Dave Towey, and Tsong Yueh Chen, How effectively does metamorphic testing alleviate the oracle problem?, IEEE Transactions on Software Engineering 40, 1, 4-22, 2014

[B20] James Whittaker, Exploratory Software Testing: Tipps, Tricks, Touren und Techniken für die Testentwicklung, 1. Auflage, Addison-Wesley Professional, 2009.

[B21] L. Wilkinson, A. Anand, und R. Grossman. High-Dimensional Visual Analytics: Interactive Exploration Guided by Pairwise Views of Point Distributions. IEEE Transactions on Visualization and Computer Graphics, 12(6):1363-1372, 2006, https://www.cs.uic.edu/~wilkinson/Publications/sorting.pdf (Zugriff 2021).

[B22] Ryan Hafen und Terence Critchlow, EDA and ML - A Perfect Pair for Large-Scale Data Analysis, IEEE 27th International Symposium on Parallel and Distributed Processing, 2013, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6651091 (Zugriff 2021).

[B23] Breck, Eric, Shanqing Cai, Eric Nielsen, Michael Salib, and D. Sculley, The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction, IEEE International Conference on Big Data (Big Data), 2017, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8258038 (Zugriff 2021).

[B24] Harman, The Role of Artificial Intelligence in Software Engineering, In First International Workshop on Realizing AI Synergies in Software Engineering (RAISE), S. 1-6. IEEE, Juni 2012, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6227961 (Zugriff 2021).

[B25] Nilambri et al, A Survey on Automated Duplicate Detection in a Bug Repository, International Journal of Engineering Research & Technology (IJERT), 2014, https://www.ijert.org/research/a-survey-on-automated-duplicate-detection-in-a-bug-repository-IJERTV3IS041769.pdf (Zugriff 2021)

[B26] Kim, D.; Wang, X.; Kim, S.; Zeller, A.; Cheung, S.C.; Park, S. (2011). "Which Crashes Should I Fix First? Predicting Top Crashes at an Early Stage to Prioritize Debugging

Efforts," in IEEE Transactions on Software Engineering, Band 37, https://ieeexplore.ieee.org/document/5711013 (Zugriff 2021).

[B27] Mao et al, Sapienz: multi-objective automated testing for Android applications, Proceedings of the 25th International Symposium on Software Testing and Analysis, Juli 2016, http://www0.cs.ucl.ac.uk/staff/K.Mao/archive/p_issta16_sapienz.pdf (Zugriff 2021).

[B28] Rai et al, Regression Test Case Optimization Using Honey Bee Mating Optimization Algorithm with Fuzzy Rule Base, World Applied Sciences Journal 31 (4): 654-662, 2014, https://www.researchgate.net/publication/336133351_Regression_Test_Case_Optimization_Using_Honey_Bee_Mating_Optimization_Algorithm_with_Fuzzy_Rule_Base (Zugriff 2021).

[B29] Dusica Marijan, Arnaud Gotlieb, Marius Liaaen. A learning algorithm for optimizing continuous integration development and testing practice, Journal of Software : Practice and Experience, Nov 2018.

[B30] Tosun et al, KI-Based Software Defect Predictors: Applications and Benefits in a Case Study, Proceedings of the Twenty-Second Innovative Applications of Artificial Intelligence Conference (IAAI-10), 2010.

[B31] Kim et al, Predicting Faults from Cached History, 29th International Conference on Software Engineering (ICSE'07), 2007.

[B32] Nagappan 2008 Nagappan et al, The Influence of Organizational Structure on Software Quality: An Empirical Case Study, Proceedings of the 30th international conference on Software engineering (ICSE'08), Mai 2008.

[B33] Kuhn et al, Software Fault Interactions and Implications for Software Testing, IEEE Transactions on Software Engineering vol. 30, no. 6, (June 2004) pp. 418-421.

12.4 Andere Referenzen [R]

Die folgenden Verweise verweisen auf Informationen, die im Internet verfügbar sind. Auch wenn diese Verweise zum Zeitpunkt der Veröffentlichung geprüft wurden, kann das ISTQB® nicht dafür verantwortlich gemacht werden, wenn die Verweise nicht mehr verfügbar sind.

[R01] Wikipedia contributors, "AI effect," Wikipedia, https://en.wikipedia.org/wiki/AI_effect (Zugriff im Mai 2021).

[R02] Apache mxnet, https://mxnet.apache.org (Zugriff im Mai 2021).

[R03] Microsoft Cognitive Toolkit, https://docs.microsoft.com/en-us/cognitive-toolkit/ (Zugriff im Mai 2021).

[R04] IBM Watson, https://www.ibm.com/watson/ai-services

[R05] Google Tensorflow https://www.tensorflow.org/ (Zugriff im Mai 2021).

[R06] Keras https://keras.io/ (Zugriff im Mai 2021).

[R07] Pytorch https://pytorch.org/ (Zugriff im Mai 2021).

[R08] https://scikit-learn.org/stable/whats_new/v0.23.html (Zugriff im Mai 2021).

[R09] NVIDIA VOLTA, https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/ (Zugriff im Mai 2021).

[R10] Cloud TPU, https://cloud.google.com/tpu/ (Zugriff im Mai 2021).

[R11] Edge TPU, https://cloud.google.com/edge-tpu/ (Zugriff im Mai 2021).

[R12] Intel® Nervana™ Neural Network Prozessoren liefern die Skalierung und Effizienz, die für die Entwicklung von Deep-Learning-Modellen erforderlich sind, https://www.intel.ai/nervana-nnp/ (Zugriff im Mai 2021).

[R13] Die Entwicklung von EyeQ, https://www.mobileye.com/our-technology/evolution-eyeq-chip/ (Zugriff im Mai 2021).

[R14] ImageNet - http://www.image-net.org/ (Zugriff im Mai 2021).

[R15] Googles BERT - https://github.com/google-research/bert (Zugriff im Mai 2021).

[R16] https://www.kaggle.com/datasets (Zugriff im Mai 2021).

[R17] https://www.kaggle.com/paultimothymooney/2018-kaggle-machine-learning-data-science-survey (Zugriff im Mai 2021).

[R18] MLCommons - https://mlcommons.org/ (Zugriff im Mai 2021).

[R19] DAWNBench - https://dawn.cs.stanford.edu/benchmark (Zugriff im Mai 2021).

[R20] MLMark - https://www.eembc.org/mlmark (Zugriff im Mai 2021).

[R21] https://digital-strategy.ec.europa.eu/en/library/assessment-list-trustworthy-artificial-intelligence-altai-self-assessment| Die digitale Zukunft Europas gestalten (europa.eu) (Zugriff August 2021)

[R22] https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai (Zugriff im August 2021)

[R23] Google GraphicsFuzz, https://github.com/google/graphicsfuzz (Zugriff im Mai 2021).

[R24] http://www.openrobots.org/morse/doc/0.2.1/morse.html (Zugriff im Mai 2021).

[R25] https://ai.facebook.com/blog/open-sourcing-ai-habitat-a-simulation-platform-for-embodied-ai-research/ (Zugriff im Mai 2021).

[R26] https://www.nvidia.com/en-gb/self-driving-cars/drive-constellation/ (Zugriff im Mai 2021).

[R27] https://uk.mathworks.com/discovery/artificial-intelligence.html#ai-with-matlab (Zugriff im Mai 2021).

Weitere (im EN-Lehrplan nur im Text oder gar nicht genannt, da EU-Spezifisch)

[R28] IBM Watson Assistant: https://www.ibm.com/ai/assistant (Zugriff im Mai 2022).

[R29] Google Cloud KI und ML Produkte: https://cloud.google.com/products/ai (Zugriff im Mai 2022).

[R30] Amazon CodeGuru: https://aws.amazon.com/codeguru/ (Zugriff im Mai 2022).

[R31] Microsoft Azure Cognitive Search: https://azure.microsoft.com/en-us/services/search/ (Zugriff im Mai 2022).

[R32] Proposal for a regulation of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act, AI Act) and amending certain union legislative acts: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206 (Zugriff im Juni 2022).

[R33] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation, GDPR): http://data.europa.eu/eli/reg/2016/679/2016-05-04 (Zugriff im Juni 2022).

Anhang A - Abkürzungen

Abkürzung	Beschreibung
KI	Künstliche Intelligenz (*artificial intelligence*, AI)
AIaaS	KI als Dienst (*AI-as-a-Service*)
API	Schnittstelle zur Anwendungsprogrammierung (*application programming interface*)
AUC	Fläche unter der Kurve (*area under curve*)
DL	Tiefes Lernen (*deep learning*)
DNN	Tiefes neuronales Netzwerk (*deep neural network*)
EDA	Explorative Datenanalyse
EU	Europäische Union
FN	Falsch negativ (*false negative*)
FP	Falsch positiv (*false positive*)
DSGVO (GDPR)	Datenschutz-Grundverordnung (*General Data Protection Regulation)*
GPU	Grafische Verarbeitungseinheit (*graphical processing unit*)
GUI	Grafische Benutzungsschnittstelle (*graphical user interface*)
LIME	Lokal interpretierbare modell-agonistische Erklärungen (*Local Interpretable Model-Agnostic Explanations*)
MC/DC	modifizierte Bedingungs-/Entscheidungsüberdeckung
ML	Maschinelles Lernen
MR	Metamorphe Beziehung (*metamorphic relationship*)
MSE	Mittlerer quadratischer Fehler (*mean square error*)
MT	Metamorpher Test
NLP	Verarbeitung natürlicher Sprache (*natural language processing*)
ROC	Empfänger-Betriebskennlinie (*receiver operation characteristic*)
SUT	Zu testendes System (*system under test*)
SVM	Support-Vektor-Maschine
RN	Richtig negativ (*true negative*, TN)
RP	Richtig positiv (*truepositive*, TP)
XAI	Erklärbare KI (*explainable AI*)

Anhang B - KI-spezifische und andere Begriffe

Begriff	Definition
Aktivierungsfunktion	Die einem Neuron in einem neuronalen Netz zugeordnete Formel, welche die Ausgabe des Neurons aus den Eingaben des Neurons bestimmt
Aktivierungswert	Die Ausgabe einer Aktivierungsfunktion eines Neurons in einem neuronalen Netz
Algorithmische Verzerrung (*algorithmic bias*)	Eine Art von Verzerrung, die durch den ML-Algorithmus verursacht wird
Allgemeine Datenschutzverordnung (*general data protection regulation*, GDPR)	Verordnung der Europäischen Union (EU) zum Schutz natürlicher Personen bei der Verarbeitung personen-bezogener Daten, gilt für die Daten der Bürger der EU und des Europäischen Wirtschaftsraums
Allgemeine KI (*general AI*)	KI, die über das gesamte Spektrum der kognitiven Fähigkeiten ein dem Menschen vergleichbares intelligentes Verhalten zeigt (ISO/IEC TR 29119-11) Synonym: starke KI
Annotation	Die Identifizierung von Objekten in Bildern mit Begrenzungsrahmen, um gekennzeichnete Daten für die Klassifikation zu erhalten
Anreicherung (*augmentation*)	Die Erstellung neuer Datenpunkte auf der Grundlage eines bestehenden Datensatzes
Argumentationsverfahren	KI, die mit Hilfe logischer Techniken Schlussfolgerungen aus verfügbaren Informationen zieht (ISO/IEC TR 29119-11)
Assoziation	Eine unüberwachte Lerntechnik, die Beziehungen und Abhängigkeiten zwischen Stichproben identifiziert
Ausreißer	Eine Beobachtung, die außerhalb des allgemeinen Musters der Datenverteilung liegt
Automatisierungsverzerrung	Eine Art von Verzerrung, die dadurch entsteht, dass eine Person die Empfehlungen eines automatisierten Entscheidungssystems gegenüber anderen Quellen bevorzugt, Synonym: Bequemlichkeitsverzerrung
Autonomes System	Ein System, das über längere Zeiträume ohne menschliches Zutun funktioniert
Autonomie	Die Fähigkeit eines Systems, über einen längeren Zeitraum ohne menschliches Eingreifen zu funktionieren (ISO/IEC TR 29119-11)
Bayessches Modell	Ein statistisches Modell, das die Unbestimmtheit von Modell-Inputs und -Outputs mit Hilfe von Wahrscheinlichkeiten darstellt.
Bayessches Verfahren	Eine Technik, die Vorher-Nachher-Wahrscheinlichkeits-verteilungen als Parameter eines statistischen Modells berücksichtigt

Begriff	Definition
Belohnungsfunktion *(reward function)*	Eine Funktion, die den Erfolg des bestärkenden Lernens definiert
Belohnungs-Hacking (*reward hacking*)	Die von einem intelligenten Agenten ausgeführte Tätigkeit zur Maximierung seiner Belohnungsfunktion auf Kosten der Erfüllung des ursprünglichen Ziels (nach ISO/IEC TR 29119-11)
Bestärkendes Lernen (*reinforcement learning*)	Die Erstellung eines ML-Modells durch einen Prozess des Ausprobierens und Belohnens, um ein Ziel zu erreichen (ISO/IEC TR 29119-11)
Big Data	Umfangreiche Datensätze, deren Merkmale in Bezug auf Volumen, Vielfalt, Geschwindigkeit und/oder Variabilität spezielle Techniken und Techniken zur Verarbeitung erfordern
Chatbot	Eine Anwendung, die dazu dient, ein Gespräch über Text oder Text-to-Speech zu führen
Clustering	Eine Art von ML-Algorithmus zur Gruppierung ähnlicher Objekte in Clustern
Clusterbildung	Eine Art von ML-Funktion, die ähnliche Datenpunkte zusammenfasst
Datenkennzeichnung (*labelling*)	Das Hinzufügen aussagekräftiger Tags zu Objekten in Rohdaten zur Unterstützung der Klassifikation in ML (Synonym: Kennzeichnung)
Datenerfassung	Die Beschaffung von Daten, die für das von einem ML-Modell zu lösende Geschäftsproblem relevant sind
Datenpipeline	Die Durchführung von Datenvorbereitungstätigkeiten zur Bereitstellung von Eingabedaten zur Unterstützung des Trainings durch einen ML-Algorithmus oder der Vorhersage durch ein ML-Modell
Datenpunkt	Ein Satz von einer oder mehreren Messungen, der eine einzelne Beobachtung umfasst und als Teil eines Datensatzes verwendet wird
Datensatz	Eine Sammlung von Daten, die für das Training, die Evaluierung, das Testen und die Vorhersage in ML verwendet werden
Datenverunreinigung (*data poisoning*)	Die absichtliche und böswillige Manipulation von Trainings- oder Eingabedaten für ein ML-Modell
Datenvisualisierung	Eine Technik zur grafischen Darstellung von Datenbeziehungen, Trends und Mustern
Datenvorbereitung	Die Aktivitäten der Datenbeschaffung, der Datenvorverarbeitung und der Merkmalsermittlung im ML-Workflow
Datenvorverarbeitung	Die Aktivitäten der Datenbereinigung, Datenumwandlung, Datenerweiterung und Datenabfrage im ML-Arbeitsablauf

Begriff	Definition
Deduktiver Klassifikator	Ein Klassifikator, der auf der Anwendung von Schlussfolgerungen und Logik auf Eingabedaten basiert
Deterministisches System	Ein System, das aus einer gegebenen Menge von Eingaben und einem gegebenen Ausgangszustand dieselbe Menge von Ausgaben und denselben Endzustand erzeugt
Edge-Computing	Der Teil einer verteilten Architektur, in dem die Informationsverarbeitung in der Nähe des Ortes erfolgt, an dem die Informationen verwendet werden.
Entscheidungsbaum	Ein baumartiges ML-Modell, dessen Knoten Entscheidungen und dessen Äste mögliche Ergebnisse darstellen
Entscheidungsschwelle	Ein Wert, der das Ergebnis einer Vorhersagefunktion in ein binäres Ergebnis von entweder über oder unter dem Wert umwandelt Synonym: Diskriminierungsschwelle
Epoche	Eine Iteration des ML-Trainings mit dem gesamten Trainingsdatensatz
Erklärbare KI (*explainable AI*, XAI)	Der Bereich der Studie, der sich mit dem Verständnis der Faktoren befasst, welche die Ergebnisse von KI-Systemen beeinflussen
Erklärbarkeit	Der Grad des Verständnisses, wie das KI-basierte System zu einem bestimmten Ergebnis gekommen ist (ISO/IEC TR 29119-11)
Evolution	Der Prozess der kontinuierlichen Veränderung von einem niedrigeren, einfacheren oder schlechteren Zustand zu einem höheren, komplexeren oder besseren Zustand
Expertensystem	Ein KI-gestütztes System zur Lösung von Problemen in einem bestimmten Bereich oder Anwendungsgebiet, indem es Schlussfolgerungen aus einer Wissensbasis zieht, die auf menschlichem Fachwissen beruht
Explorative Datenanalyse (EDA)	Die interaktive, hypothesengesteuerte und visuelle Erkundung von Daten zur Unterstützung der Merkmalsermittlung
F1-Wert (*F1-Score*)	Eine funktionale Leistungsmetrik von ML, die zur Evaluierung eines Klassifizierers verwendet wird und ein Gleichgewicht zwischen Recall und Präzision herstellt
Fallbezogene Argumentation (*case-based reasoning*)	Die Technik, ein neues Problem auf der Grundlage der Lösungen ähnlicher früherer Probleme zu lösen (Synonym: Fallbasiertes Schließen)
Falsch negativ (FN)	Eine ML-Modellvorhersage, bei der das Modell fälschlicherweise die negative Klasse vorhersagt
Falsch positiv (FP)	Eine ML-Modellvorhersage, bei der das Modell fälschlicherweise die positive Klasse vorhersagt

Begriff	Definition
Merkmalsermittlung (*Feature-Engineering*)	Die Aktivität, bei der die Attribute in den Rohdaten, welche die zugrunde liegenden Beziehungen, die im ML-Modell erscheinen sollen, am besten repräsentieren, zur Verwendung in den Trainingsdaten identifiziert werden (ISO/IEC TR 29119-11)
Fehlervorhersage	Eine Technik zur Vorhersage der Bereiche innerhalb des Testobjekts, in denen Fehler auftreten werden, oder der Menge der vorhandenen Fehler
Feindlicher Angriff	Die absichtliche Verwendung von Negativbeispielen, um ein ML-Modell zum Scheitern zu bringen
Fläche unter der Kurve (*area under curve*, AUC)	Ein Maß dafür, wie gut ein Klassifikator zwischen zwei Klassen unterscheiden kann.
Flexibilität	Die Fähigkeit eines Systems, in Kontexten außerhalb seiner ursprünglichen Spezifikation zu arbeiten (ISO/IEC TR 29119-11)
Fuzzy-Logik	Eine Art von Logik, die auf dem Konzept der Teilwahrheit basiert und durch Sicherheitsfaktoren zwischen 0 und 1 dargestellt wird
Gegnerisches Beispiel (*adversarial example*)	Eine Eingabe für ein ML-Modell, die durch geringfügige Störungen eines funktionierenden Beispiels erzeugt wird und dazu führt, dass das Modell mit hoher Wahrscheinlichkeit ein falsches Ergebnis ausgibt (ISO/IEC TR 29119-11), Synonym: Kontradiktorisches Beispiel
Gegnerisches Testen (*adversarial testing*)	Ein Testverfahren, das auf der Erstellung und Ausführung von gegnerischen Beispielen basiert, um Fehlerzustände in einem ML-Modell zu identifizieren (ISO/IEC TR 29119-11), Synonym: Kontradiktorisches Testen
Genauigkeit (*Accuracy*)	Funktionale Leistungsmetrik von ML zur Evaluierung eines Klassifikators, die den Prozentsatz aller richtig vorhergesagten Klassifikationen misst (ISO/IEC TR 29119-11)
Gewicht	Eine interne Variable einer Verbindung zwischen Neuronen in einem neuronalen Netz, die sich darauf auswirkt, wie es seine Ausgaben berechnet, und die sich ändert, wenn das neuronale Netz trainiert wird
Grafikprozessor (GPU)	Eine anwendungsspezifische integrierte Schaltung zur Manipulation und Änderung des Speichers, um die Erzeugung von Bildern in einem Bildpuffer zu beschleunigen, der für die Ausgabe an ein Anzeigegerät bestimmt ist
Grundwahrheit	Informationen, die durch direkte Beobachtung und Messung gewonnen werden und von denen man weiß, dass sie real oder wahr sind
Hyperparameter	Ein Parameter, der entweder zur Steuerung des Trainings eines ML-Modells oder zur Festlegung der Konfiguration eines ML-Modells verwendet wird

Begriff	Definition
Hyperparameter-Tuning	Die Bestimmung der optimalen Hyperparameter des Modells oder des Algorithmus auf der Grundlage bestimmter Ziele
Inferenz	Verfahren zum logischen Schließen bzw. Schlussfolgern
Intelligenter Agent	Ein autonomes Programm, das seine Aktivitäten anhand von Beobachtungen und Aktionen auf das Erreichen von Zielen ausrichtet
Inter-Cluster-Metrik	Eine Metrik, welche die Ähnlichkeit von Datenpunkten in verschiedenen Clustern misst
Interpretierbarkeit	Der Grad des Verständnisses, wie die zugrunde liegende KI-Technik funktioniert (ISO/IEC TR 29119-11)
Intra-Cluster-Metrik	Eine Metrik, welche die Ähnlichkeit von Datenpunkten innerhalb eines Clusters misst
KI-als- Dienst (AIaaS)	Ein Software-Lizenzierungs- und Bereitstellungsmodell, bei dem KI und KI-Entwicklungsdienste zentral gehostet werden
KI-basiertes System	Ein System, das eine oder mehrere KI-Komponenten integriert
KI-Effekt	Die Situation, dass ein zuvor als KI gekennzeichnetes System aufgrund des technologischen Fortschritts nicht mehr als KI angesehen wird (ISO/IEC TR 29119-11)
KI-Komponente	Eine Komponente, die KI-Funktionalität bietet
KI-spezifischer Prozessor	Eine Art spezialisierte Hardware zur Beschleunigung von KI-Anwendungen
Klassifikation	Eine Art von ML-Funktion, welche die Ausgabeklasse für eine gegebene Eingabe vorhersagt (ISO/IEC TR 29119-11)
Klassifikator	Ein ML-Modell für die Klassifikation Synonym: Klassifikationsmodell
k-nächster Nachbar	Ein Klassifikationsansatz, der die Wahrscheinlichkeit der Gruppenzugehörigkeit eines Datenpunkts in Abhängigkeit von der Gruppenzugehörigkeit der nächstgelegenen Datenpunkte schätzt
Konfusionsmatrix	Ein Verfahren zur Zusammenfassung der ML-funktionalen Leistung von ML eines Klassifikationsalgorithmus
Konzeptdrift	Eine Veränderung in der wahrgenommenen Genauigkeit der Vorhersagen eines ML-Modells im Laufe der Zeit, die durch Veränderungen in den Erwartungen und im Verhalten der Nutzer sowie in der betrieblichen Einsatzumgebung verursacht wird.
Künstliche Intelligenz (KI)	Die Fähigkeit eines technischen Systems, Wissen und Fähigkeiten zu erwerben, zu verarbeiten, zu erzeugen und anzuwenden (ISO/IEC TR 29119-11)

Begriff	Definition
Lernalgorithmus	Ein Programm, das ein ML-Modell auf der Grundlage der Eigenschaften des Trainingsdatensatzes erstellt
LIME-Methode	Der Algorithmus *Local Interpretable Model-Agnostic Explanations* zur Erklärung der Vorhersagen aus einem ML-Modell
Lineare Regression	Ein statistisches Verfahren, das die Beziehung zwischen Variablen modelliert, indem eine lineare Gleichung an die beobachteten Daten angepasst wird, wenn die Zielvariable nummerisch ist.
Logistische Regression	Ein statistisches Verfahren, das die Beziehung zwischen Variablen modelliert, wenn die Zielvariable und/oder die Eingangsvariablen kategorisch und nicht-nummerisch ist
Maschinelles Lernen (*machine learning*, ML)	Der Prozess, bei dem computergestützte Techniken eingesetzt werden, um Systeme in die Lage zu versetzen, aus Daten oder Erfahrungen zu lernen (ISO/IEC TR 29119-11)
Merkmal (*feature*)	Eine einzeln messbares Attribut der Eingabedaten, das von einem ML-Algorithmus für das Training und von einem ML-Modell für die Vorhersage verwendet wird
Mittlerer quadratischer Fehler (*mean square error*, MSE)	Das statistische Maß für die durchschnittliche quadratische Differenz zwischen den geschätzten Werten und dem tatsächlichen Wert
ML-Algorithmus	Ein Algorithmus, der zur Erstellung eines ML-Modells aus einem Trainingsdatensatz verwendet wird
ML-Benchmark-Suite	Ein Datensatz, der zum Vergleich von ML-Modellen und ML-Algorithmen über eine Reihe von Evaluierungsmetriken verwendet wird
ML-Funktion	Von einem ML-Modell implementierte Funktionen, wie Klassifikation, Regression oder Clusterbildung
ML-Modelltuning	Der Prozess des Testens von Hyperparametern, um eine optimale Leistung zu erreichen
ML-Modellevaluierung	Der Prozess des Vergleichs der erzielten funktionalen Leistungsmetriken von ML mit den erforderlichen Kriterien und denen anderer ML-Modelle
ML-Modelltraining	Der Prozess der Anwendung des ML-Algorithmus auf den Trainingsdatensatz zur Erstellung eines ML-Modells
ML-Framework	Ein Werkzeug oder eine Bibliothek, welche die Erstellung eines ML-Modells unterstützt
ML-System	Ein System, das ein oder mehrere ML-Modelle integriert
ML-Workflow	Eine Abfolge von Aktivitäten zur Verwaltung der Entwicklung und des Einsatzes eines ML-Modells
Multi-Agenten-System	Ein System, das mehrere intelligente Agenten umfasst

Begriff	Definition
Neuromorpher Prozessor	Ein integrierter Schaltkreis, der die biologischen Neuronen des menschlichen Gehirns nachbilden soll
Neuron	Ein Knoten in einem neuronalen Netz, der normalerweise mehrere Eingangswerte empfängt und einen Aktivierungswert erzeugt
Neuronaler Netzwerktrojaner	Eine Schwachstelle, die mittels eines Datenverunreinigungs-Angriffs in ein neuronales Netz eingeschleust wird, um sie später auszunutzen
Neuronales Netz	Ein Netzwerk von primitiven Verarbeitungselementen, die durch gewichtete Verbindungen mit einstellbaren Gewichten verbunden sind, in dem jedes Element einen Wert durch Anwendung einer nichtlinearen Funktion auf seine Eingangswerte erzeugt und ihn an andere Elemente weitergibt oder als Ausgangswert präsentiert (ISO/IEC 2382) Synonym: künstliches neuronales Netz
Nicht-deterministisches System	Ein System, das bei einer bestimmten Menge von Eingaben und einem bestimmten Ausgangszustand nicht immer dieselbe Menge von Ausgaben und denselben Endzustand erzeugt
Perzeptron	Ein neuronales Netz mit nur einer Schicht und einem Neuron
Präzision (*precision*)	Eine funktionale Leistungsmetrik von ML, die zur Evaluierung eines Klassifikators verwendet wird; misst den Anteil der korrekten Ergebnisse unter den als positiv vorhergesagten Ergebnissen (ISO/IEC TR 29119-11)
Probabilistisches System	Ein System, dessen Verhalten in Form von Wahrscheinlichkeiten beschrieben wird und dessen Ergebnisse daher nicht genau vorhergesagt werden können.
Prozedurale Argumentation (*procedural reasoning*)	KI-Technik für die Entwicklung von Echtzeit-Schluss-folgerungssystemen, die komplexe Aufgaben in dynamischen Umgebungen ausführen können Synonym: Prozedurales Schließen
Random Forest	ML-Technik für Klassifikation, Regression und andere Aufgaben, bei der viele Entscheidungsbäume erstellt und ausgeführt werden und dann entweder der Modus der Klasse oder die mittlere Vorhersage der einzelnen Bäume ausgegeben wird
Rauschen	Eine Störung oder Verfälschung von Daten
Receiver-Operating-Characteristic (ROC)-Kurve	Eine grafische Darstellung, welche die Fähigkeit eines binären Klassifizierers veranschaulicht, wenn sein Unterscheidungs-Schwellenwert variiert wird
Regelmaschine	Eine Reihe von Regeln, die festlegen, welche Aktionen stattfinden sollen, wenn bestimmte Bedingungen erfüllt sind

Begriff	Definition
Regression	Eine Art von ML-Funktion, die zu einem numerischen oder kontinuierlichen Ausgabewert für eine gegebene Eingabe führt (ISO/IEC TR 29119-11)
Regressionsmodell	Ein ML-Modell, dessen erwartete Ausgabe für eine gegebene numerische Eingabe eine kontinuierliche Variable ist (ISO/IEC DIS 23053)
Richtig Negativ (RN)	Eine Vorhersage, bei der das Modell die negative Klasse korrekt vorhersagt
Richtig Positiv (RP)	Eine Vorhersage, bei der das Modell die positive Klasse korrekt vorhersagt
R-Quadrat	Ein statistisches Maß dafür, wie nahe die Datenpunkte an der angepassten Regressionslinie liegen. Synonym: Bestimmungskoeffizient
Schwache KI (*narrow AI*)	KI konzentriert sich auf eine einzelne, klar definierte Aufgabe, um ein spezifisches Problem zu lösen (ISO/IEC TR 29119-11, Synonym: enge KI
Selbstlernendes System	Ein adaptives System, das sein Verhalten auf der Grundlage von Lernen durch Versuch und Irrtum ändert (ISO/IEC TR 29119-11)
Sensitivität (*recall*)	Eine funktionale Leistungsmetrik von ML zur Evaluierung eines Klassifikators; misst den Anteil der korrekt vorhergesagten Ergebnisse unter den tatsächlich positiven Fällen (ISO/IEC TR 29119-11) Synonym: Empfindlichkeit
(Funktionale) Sicherheit	Die Erwartung, dass ein System unter definierten Bedingungen nicht zu einem Zustand führt, in dem menschliches Leben, Gesundheit, Eigentum oder die Umwelt gefährdet sind (ISO/IEC/IEEE 12207)
Silhouettenkoeffizient	Ein Clustermaß zwischen -1 und +1, dass auf den durchschnittlichen Unterschieden zwischen den Clustern und innerhalb der Cluster basiert Synonym: Silhouettenwertung
Stichprobenverzerrung (*sample bias*)	Eine Art von Verzerrung, bei der der Datensatz nicht vollständig repräsentativ für den Datenraum ist, auf den ML angewendet wird
Suchalgorithmus	Ein Algorithmus, der systematisch eine Teilmenge aller möglichen Zustände oder Strukturen besucht, bis der Zielzustand oder die Zielstruktur erreicht ist (ISO/IEC TR 29119-11)
Super-KI	Ein auf künstlicher Intelligenz basierendes System, das die menschlichen Fähigkeiten weit übertrifft

Begriff	Definition
Support-Vektor-Maschine (SVM)	Eine ML-Technik, bei der die Datenpunkte als Vektoren im mehrdimensionalen Raum betrachtet werden, die durch eine Hyperebene getrennt sind
Technologische Singularität	Ein Punkt in der Zukunft, an dem der technologische Fortschritt nicht mehr vom Menschen kontrolliert werden kann (ISO/IEC TR 29119-11)
Automatisierungsverzerrung	Eine Art von Verzerrung, die dadurch entsteht, dass eine Person die Empfehlungen eines automatisierten Entscheidungssystems gegenüber anderen Quellen bevorzugt Synonym: Bequemlichkeitsverzerrung
Testdatensatz	Vom Trainingsdatensatz unabhängiger Datensatz für die Evaluierung, ob die vereinbarten funktionalen Leistungskriterien von ML erfüllt werden Synonym: Holdout-Datensatz
Testorakel-Problem	Die Herausforderung, festzustellen, ob ein Test bei einer bestimmten Anzahl von Testeingaben und einem bestimmten Zustand bestanden oder nicht bestanden wurde
Tiefes Lernen (*deep learning*, DL)	ML mit neuronalen Netzen mit mehreren Schichten
Tiefes neuronales Netzwerk (*deep neural network*, DNN)	Ein neuronales Netz, das aus mehreren Schichten von Neuronen besteht Synonym: Mehrschichtiges Perzeptron
Trainingsdatensatz	Ein Datensatz, der zum Trainieren eines ML-Modells verwendet wird
Transfer-Lernen	Eine Technik zur Modifizierung eines vortrainierten ML-Modells, um eine andere verwandte Aufgabe zu erfüllen
Transparenz	Der Grad der Sichtbarkeit des Algorithmus und der Daten, die von dem KI-basierten System verwendet werden (ISO/IEC TR 29119-11)
Überanpassung	Die Erstellung eines ML-Modells, das zu sehr dem Trainingsdatensatz entspricht, was zu einem Modell führt, das nur schwer auf neue Daten verallgemeinert werden kann (ISO/IEC TR 29119-11)
Überwachtes Lernen	Trainieren eines ML-Modells aus Eingabedaten und den entsprechenden Kennzeichnungen
Unangemessene Verzerrung	Eine Art von Verzerrung, die dazu führt, dass ein System Ergebnisse produziert, die für eine bestimmte Gruppe nachteilige Auswirkungen haben
Unteranpassung	Die Erstellung eines ML-Modells, das den zugrunde liegenden Trend des Trainingsdatensatzes nicht widerspiegelt, was zu einem Modell führt, das nur schwer genaue Vorhersagen machen kann (ISO/IEC TR 29119-11)

Begriff	Definition
Unüberwachtes Lernen	Trainieren eines ML-Modells anhand von Eingabedaten unter Verwendung eines nicht beschrifteten Datensatzes
Validierungsdatensatz	Ein Datensatz zur Evaluierung eines trainierten ML-Modells mit dem Ziel, das Modell zu optimieren
Verarbeitung natürlicher Sprache (*natural language processing*, NLP)	Ein Bereich der Informatik, der die Fähigkeit bietet, natürliche Sprachen zu erkennen, zu verstehen, zu manipulieren und aus ihnen eine Bedeutung abzuleiten
Verzerrung (*bias*)	Die systematische Ungleichbehandlung von bestimmten Objekten, Personen oder Gruppen im Vergleich zu anderen (ISO/IEC DIS 22989) Synonym: Bias
Von-Neumann-Architektur	Eine Computerarchitektur, die aus fünf Hauptkomponenten besteht: Speicher, zentrale Verarbeitungseinheit, Steuereinheit, Ein- und Ausgabe
Vortrainiertes Modell	Ein ML-Modell, das bereits trainiert wurde, als es erhalten wurde

Index