ISTQB® Certified Tester
Lehrplan KI Test (CT-AI)

Version 2.0 D

International Software Testing Qualifications Board



Deutschsprachige Ausgabe, zur Verfügung gestellt von
German Testing Board e. V. und Austrian 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









© International Software Testing Qualifications Board

Urheberschutzvermerk

© International Software Testing Qualifications Board (im Folgenden ISTQB® genannt)

Übersetzung des englischsprachigen Lehrplans des International Software Testing Qualifications Board (ISTQB®), Originaltitel: Certified Tester AI Testing (CT-AI) Syllabus, Version V2.0 von 2026.

ISTQB® ist eine eingetragene Marke des International Software Testing Qualifications Board.

Urheberrecht © 2026 an diesem Lehrplan haben die Autoren der englischen Originalausgabe: Klaudia Dussa-Zieger (chair), Stuart Reid, Vipul Kocher, Qin Liu, Jarosław Hryszko, Kyle Alexander Siemens und Werner Henschelchen.

Urheberrecht © 2021 an der Version 1.0 haben die Autoren der englischen Originalausgabe: Klaudia Dussa-Zieger (chair), Vipul Kocher, Qin Liu, Stuart Reid, Kyle Alexander Siemens, Werner Henschelchen und Adam Leon Smith.

Urheberrecht © 2026 an der Übersetzung in die deutsche Sprache steht den Mitgliedern der D.A.CH-Task-Force Lokalisierung CT-AI V2.0 zu: Ralf Bongard, Renzo Cerquozzi, Florian Fieber, Jan Giesen, Matthias Hamburg, Michael Humm, Manfred Oberlerchner, Ralf Pichler, Nishan Portoyan, Nils Röttger, Gerhard Runze, Maud Schlich, Erhard Wunderlich und Mario Winter (Leitung).

Dieser Lehrplan KI Test (CT-AI), ISTQB® Version 2.0 D vom 12.11.2026 („das Werk“) ist urheberrechtlich geschützt. Inhaber der ausschließlichen Nutzungsrechte an dem Werk sind das German Testing Board (GTB) und das Austrian Testing Board (ATB).

Alle Rechte vorbehalten. Die Autoren übertragen hiermit das Urheberrecht an das ISTQB®. Die Autoren (als derzeitige Inhaber der Urheberrechte) und das ISTQB® (als künftiger Inhaber der Urheberrechte) haben sich mit den folgenden Nutzungsbedingungen einverstanden erklärt:

- Auszüge aus diesem Dokument dürfen für nicht-kommerzielle Zwecke kopiert werden, wenn die Quelle angegeben wird. Jeder zugelassene Schulungsanbieter darf diesen Lehrplan als Grundlage für einen Schulungskurs verwenden, wenn die Autoren und das ISTQB® als Quelle und Urheberrechtsinhaber des Lehrplans genannt werden und unter der Voraussetzung, dass in der Werbung für einen solchen Schulungskurs der Lehrplan erst dann erwähnt wird, wenn die offizielle Akkreditierung des Schulungsmaterials durch ein vom ISTQB® anerkanntes Mitgliedsboard erfolgt ist.
- Jede Einzelperson oder Gruppe von Einzelpersonen darf diesen Lehrplan als Grundlage für Artikel und Bücher verwenden, wenn die Autoren und das ISTQB® als Quelle und Urheberrechtsinhaber des Lehrplans genannt werden.
- Jede andere Verwendung dieses Lehrplans ist ohne vorherige schriftliche Zustimmung des ISTQB® verboten.
- Jedes vom ISTQB® anerkannte nationale Mitglieds-Board darf diesen Lehrplan übersetzen, sofern es den oben genannten Urheberrechtshinweis in der übersetzten Fassung des Lehrplans wiedergibt.

Zur besseren Lesbarkeit wird im gesamten Dokument auf die gleichzeitige Verwendung männlicher und weiblicher Sprachformen verzichtet. Es wird das generische Maskulinum verwendet, wobei unterschiedliche Geschlechter gleichermaßen gemeint sind.

Änderungsübersicht Originalausgabe

Version	Datum	Bemerkungen
1.0	2021/10/01	Freigabe für GA
2.0 Alpha	2025/08/15	Alpha review
2.0 Beta	2026/01/07	Beta review
2.0	2026/04/17	Release for GA

Änderungsübersicht deutschsprachige Ausgabe

Version	Datum	Bemerkungen
1.0 D	08.12.2022	Veröffentlichung durch GTB e.V. mit ATB und STB
2.0 D	12.11.2026	Veröffentlichung durch GTB e.V. mit ATB

© International Software Testing Qualifications Board

Inhaltsverzeichnis

Urheberschutzvermerk ... 2
Änderungsübersicht Originalausgabe ... 3
Änderungsübersicht deutschsprachige Ausgabe ... 3
Inhaltsverzeichnis ... 4
Danksagung ... 7
0 Einführung ... 8
0.1 Zweck dieses Syllabus ... 8
0.2 Der Certified Tester KI Test (CT-AI) ... 8
0.3 Karriereweg für Tester ... 8
0.4 Geschäftlicher Nutzen ... 9
0.5 Prüfbare Lernziele, Ziele der praktischen Übungen und kognitive Wissensstufen ... 9
0.6 Die Zertifizierungsprüfung zum KI Test (CT-AI) ... 10
0.7 Akkreditierung ... 10
0.8 Umgang mit Normen und Standards ... 10
0.9 Detaillierungsgrad ... 10
0.10 Aufbau des Lehrplans ... 11
1 Einführung in die Künstliche Intelligenz – 120 Minuten ... 13
1.1 Einführung in die KI ... 14
1.1.1 KI-basierte Systeme und konventionelle Systeme ... 14
1.1.2 Schwache KI, allgemeine KI und Super-KI ... 14
1.1.3 Verschiedene Arten von KI-Technologien ... 15
1.1.4 Generative KI ... 16
1.1.5 Hardware für ML-Systeme ... 17
1.1.6 Entwicklung und Hosting von KI-Modellen ... 17
1.1.7 Entwicklungs-Frameworks für maschinelles Lernen ... 18
1.1.8 Regularien und Standards für KI ... 19
2 Qualitätsmerkmale für KI-basierte Systeme – 45 Minuten ... 21
2.1 Qualitätsmerkmale für KI-basierte Systeme ... 22
2.1.1 KI-spezifische Qualitätsmerkmale ... 22
2.1.2 KI und Sicherheit (*safety*) ... 23
2.2 Akzeptanzkriterien für KI-basierte Systeme ... 23
2.2.1 Akzeptanzkriterien für KI-basierte Systeme ... 23
3 Maschinelles Lernen (ML) – 375 Minuten ... 26
3.1 Einführung in das maschinelle Lernen ... 27
3.1.1 Verschiedene Arten maschinellen Lernens ... 27

© International Software Testing Qualifications Board

3.1.2 ML-Workflow ... 28
3.1.3 Praktische Übung zum Ziel HO-3.1.3 (H2): … selbstständig ein ML-Modell erstellen ... 30
3.1.4 Vortrainierte Modelle, Fein-Tuning und Retrieval-augmentierte Generierung ... 30
3.2 Daten für maschinelles Lernen ... 31
3.2.1 Aktivitäten im Rahmen der Datenvorbereitung ... 31
3.2.2 Praktische Übung zum Ziel HO-3.2.2 (H2): … selbstständig die Datenvorbereitung durchführen, um die Erstellung eines ML-Modells zu unterstützen. ... 32
3.2.3 Trainings-, Validierungs- und Testdatensätze ... 32
3.3 Funktionale Leistungsmetriken von ML zur Klassifikation ... 33
3.3.1 Berechnung der funktionalen Leistungsmetriken von ML ... 33
3.3.2 Praktische Übung zum Ziel HO-3.3.2 (H2): … ein ML-Modell anhand ausgewählter funktionaler Leistungsmetriken von ML bewerten. ... 34
3.3.3 Praktische Übung zum Ziel HO-3.3.3 (H2): … die Auswirkungen verschiedener ML-Modelle und Datensatzkombinationen aufzeigen. ... 35
3.4 Neuronale Netze ... 35
3.4.1 Struktur und Funktionsweise eines tiefen neuronalen Netzes ... 36
3.4.2 Praktische Übung zum Ziel HO-3.4.2 (H1): … die Implementierung eines Perzeptrons nachvollziehen ... 37
3.4.3 Überdeckungsmaße für neuronale Netze ... 37
4 Testen KI-basierter Systeme – 195 Minuten ... 38
4.1 Einführung in das Testen KI-basierter Systeme ... 39
4.1.1 Fixierte und adaptive KI-basierte Systeme ... 39
4.1.2 Begründung für einen statistischen Ansatz beim Testen KI-basierter Systeme ... 40
4.1.3 Testorakel für KI-basierte Systeme ... 41
4.2 Testen von generativer KI und Large Language-Modellen ... 41
4.2.1 Testen generativer KI ... 42
4.2.2 Red-Teaming für GenAI-Systeme ... 42
4.2.3 Praktische Übung zum Ziel HO-4.2.3 (H2): … selbstständig explorative Tests eines LLMs durchführen ... 43
4.3 Teststufen und ML-Systeme ... 44
4.3.1 Teststufen für ML-Systeme ... 44
4.3.2 Risikobasierter Test für ML-Systeme ... 44
5 Eingabedatentest für ML-Systeme – 180 Minuten ... 46
5.1 Eingabedatentest für ML-Systeme ... 47
5.1.1 Risiken bei Eingabedaten und ihre Minderung ... 47
5.1.2 Testen auf Verzerrung ... 48
5.1.3 Daten-Pipelinetest ... 49
5.1.4 Datenrepräsentativitätstest ... 50

© International Software Testing Qualifications Board

5.1.5 Datensatzeinschränkungstest ... 51
5.1.6 Kennzeichen-Korrektheitstest ... 51
5.1.7 Praktische Übung zum Ziel HO-5.1.7 (H2): … selbstständig den Eingabedatentest auf ML-Datensätzen durchführen ... 52
6 Modelltest für ML-Systeme – 225 Minuten ... 53
6.1 Modelltest für ML-Systeme ... 54
6.1.1 Risiken und Maßnahmen zur Risikominderung beim maschinellen Lernen ... 54
6.1.2 Dokumentation und Review von ML-Modellen ... 55
6.1.3 Testen der funktionalen Leistung für probabilistische ML-Systeme ... 56
6.1.4 Gegnerischer Test von ML-Systemen ... 57
6.1.5 Metamorpher Test ... 58
6.1.6 Praktische Übung zum Ziel HO-6.1.6 (H2): … selbstständig metamorphe Tests anwenden ... 58
6.1.7 Drift-Test ... 59
6.1.8 Testen auf Überanpassung und Unteranpassung ... 59
6.1.9 A/B-Test ... 60
6.1.10 Back-to-Back-Test ... 61
7 ML-Entwicklungstest – 30 Minuten ... 62
7.1 ML-Entwicklungstest ... 63
7.1.1 Risiken bei der Entwicklung von ML-Systemen und Maßnahmen zu deren Minderung ... 63
7.1.2 Bereitstellungstest für ML-Systeme ... 64
8 Abkürzungen ... 66
9 KI-spezifische und andere Begriffe ... 68
10 Referenzen ... 77
10.1 Normen und Standards ... 77
10.2 ISTQB®-Dokumente ... 77
10.3 Glossar-Referenzen ... 77
10.4 Bücher, Artikel und Web-Seiten ... 78
11 Markenzeichen ... 79
12 Anhang A – Lernziele/Kognitive Wissensstufen ... 80
Stufe 1: Erinnern (K1) ... 80
Stufe 2: Verstehen (K2) ... 80
Stufe 3: Anwenden (K3) ... 81
Stufe 4: Analysieren (K4) ... 81
13 Anhang B – Verfolgbarkeitsmatrix für geschäftliche Nutzen mit den Lernzielen ... 82
14 Anhang C – Versionshinweise ... 89
15 Index ... 90

© International Software Testing Qualifications Board

Danksagung

Der englischsprachige Lehrplan wurde von der Generalversammlung des ISTQB® am 17. April 2026 formell freigegeben.

Er wurde von der ISTQB® taskforce AI (v2.0) erstellt: Klaudia Dussa-Zieger (chair), Stuart Reid, Vipul Kocher, Qin Liu, Jarosław Hryszko, Kyle Alexander Siemens und Werner Henschelchen.

Die folgenden Personen haben an den Reviews und der Kommentierung dieses Lehrplans mitgewirkt: Marina Abratis, Tom Adams, Laura Albert, Abhishek Alladi, Menno van den Berg, Earl Burba, Simeone Chiumarulo, Marco Ciarlitto, Alessandro Collino, Jean-Baptiste Crouigneau, Yara Dalgamoni, Taz Daughtrey, Wim Decoutere, Dmitrii Degtiarenko, Iuliia Emelianova, Lozana Enbah, Tamás Gergely, David Hendrickx, David Janota, Sagar Joshi, Norbert Juhász, Willem Keesman, John Kurowski, Ine Lutterman, Niranjan Maharajh, Rik Marselis, Judy McKay, Gary Mogyorodi, Markus Niehammer, Tauhida Parveen, Arnd Pehl, Lukas Piska, Daniel Polan, Andrew Pollner, Nishan Portoyan, Meile Posthuma, Miroslav Renda, Randall Rice, Piet de Roo, Nicola de Rosa, Mark Rutz, Salvatore Sarno, Klaus Skafte, Giancarlo Tomasig, Yaron Tsubery, Rahul Verma, André Verschelling, Linda Vreeswijk, Mario Winter

Das Team dankt den Arbeitsgruppen 'Exam', 'Glossary' und 'Marketing' für ihre Unterstützung bei der Entwicklung des Lehrplans, und den Mitglieds-Boards für ihre Anregungen und Beiträge.

Die Version 1.0 des englischsprachigen Lehrplans wurde von der ISTQB® taskforce AI (v1.0) erstellt: Klaudia Dussa-Zieger (chair), Vipul Kocher, Qin Liu, Stuart Reid, Adam Leon Smith, Kyle Alexander Siemens, und Werner Henschelchen.

Das Team dankt den Autoren der drei beitragenden 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

Als Reviewer der lokalisierten deutschsprachigen Fassung 2.0 waren aktiv: Isabelle Chariot, Matthias Hamburg, Martin Klonk, Horst Pohlmann, Robert Pucher, Mario Winter

Das Team der D.A.CH-Task-Force Lokalisierung CT-AI V2.0 dankt den Arbeitsgruppen „Prüfung“ und „Glossar“, den Reviewern sowie dem Vorstand des GTB e.V. für ihre Unterstützung bei der Lokalisierung dieses Dokuments.

© International Software Testing Qualifications Board

0 Einführung

0.1 Zweck dieses Syllabus

Dieser Lehrplan bildet die Grundlage für den ISTQB® KI Test (CT-AI). Das ISTQB® stellt diesen Lehrplan wie folgt zur Verfügung:

- 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.
- Zertifizierungsstellen zur Ableitung von Prüfungsfragen in ihre nationale(n) Sprache(n), die an die Lernziele dieses Lehrplans angepasst sind.
- Ausbildungsanbietern zur Erstellung von Kursmaterialien und zur Bestimmung angemessener Lehrmethoden.
- Zertifizierungskandidaten zur Vorbereitung auf die Zertifizierungsprüfung (entweder als Teil eines Seminars oder unabhängig davon).
- 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 KI Test (CT-AI)

Die Qualifikation Certified Tester KI Test (CT-AI) richtet sich an alle in das Thema Softwaretesten involvierten Personen, die ihr Wissen über das Testen KI-basierter Systeme 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, Benutzerabnahme-Tester und Softwareentwickler. Diese Zertifizierung eignet sich auch für alle, die ein grundlegendes Verständnis für das Testen KI-basierter Systeme erlangen möchten, wie z. B. Projektmanager, Qualitätsmanager, Softwareentwicklungsmanager, Business-Analysten, Mitglieder von Operations-Teams, IT-Direktoren und Unternehmensberater.

0.3 Karriereweg für Tester

Das ISTQB®-Schema unterstützt Testexperten in allen Stufen ihrer Karriere und bietet ihnen sowohl eine breite als auch eine tiefe Wissensbasis. Personen, welche die ISTQB®-Zertifizierung Certified Tester – Foundation Level erwerben, könnten auch an den Core Advanced Levels (Test Analyst, Technical Test Analyst, Test Management, Test Automation Engineering und Agile Tester) und anschließend am Expert Level (Testmanagement oder Testprozessverbesserung) interessiert sein. Wer Fähigkeiten in der Testtätigkeit in einer agilen Umgebung entwickeln möchte, könnte die Zertifizierung Agile Test Leadership at Scale (ATLaS) in Betracht ziehen. Die Aufbaustufe Specialist bietet tiefe Einblicke in spezifische Testansätze und Testaktivitäten (z. B. KI-Test, Testen mit generativer KI, Test Automation Strategy oder Testen mobiler Apps) und bündelt das Test-Know-how für bestimmte Branchendomänen (z. B. Automotive oder Gaming).

Aktuelle Informationen über das ISTQB®-Certified-Tester-Schema finden Sie unter www.istqb.org oder auf den Seiten der nationalen Boards, wie z. B. https://www.gtb.de (Deutschland) und https://www.austriantestingboard.at (Österreich)

© International Software Testing Qualifications Board

0.4 Geschäftlicher Nutzen

In diesem Abschnitt werden die acht geschäftlichen Nutzen (*business outcomes*, BO) aufgeführt, die von Kandidaten erwartet werden, welche die Zertifizierung KI Test (CT-AI) erworben haben.

Kandidaten, die die Zertifizierung KI Test (CT-AI) erworben haben, können Folgendes:

BO1	Verstehen des aktuellen Stands der KI, einschließlich generativer KI.
BO2	Sammeln von Erfahrung mit der Implementierung und dem Testen von Modellen des maschinellen Lernens (ML).
BO3	Verstehen der Funktionsweise und des Testens einfacher neuronaler Netze.
BO4	Verstehen der spezifischen Qualitätsmerkmale von KI gemäß ISO/IEC 25059.
BO5	Berechnen und Interpretieren funktionaler Leistungsmetriken von ML.
BO6	Erkennen von Umfang und Bedeutung der beiden Teststufen, die für das Testen von ML-Systemen spezifisch sind.
BO7	Beitragen zur Entwicklung einer effektiven Teststrategie für ML-Systeme.
BO8	Entwerfen und Durchführen von Testfällen für ML-Systeme.

0.5 Prüfbare Lernziele, Ziele der praktischen Übungen und kognitive Wissensstufen

Lernziele (*learning objectives*, LO) unterstützen den geschäftlichen Nutzen und dienen zur Ausarbeitung der Prüfungen für die Zertifizierung Certified Tester „KI Test“.

Die spezifischen Lernziele werden zu Beginn jedes Kapitels aufgeführt und wie folgt klassifiziert:

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

Weitere Details und Beispiele für Lernziele finden Sie in Anhang A – Lernziele/Kognitive Wissensstufen.

Für alle Begriffe, die als Schlüsselbegriffe und KI-spezifische Schlüsselbegriffe direkt unter den Kapitelüberschriften aufgeführt sind, müssen der korrekte Name und die Definition aus dem ISTQB®-Glossar oder aus Kapitel 9 erinnert werden können (K1), auch wenn sie nicht ausdrücklich in den Lernzielen erwähnt werden.

Die spezifischen Ziele der praktischen Übungen (*hands-on objectives*, HO) fokussieren auf die praktische Anwendung der Lernziele und werden am Anfang jedes Kapitels aufgeführt. Die Stufe eines HO-Ziels wird wie folgt klassifiziert:

- H0: Dies kann eine Live-Demonstration einer Übung oder ein aufgezeichnetes Video umfassen. Da die Übung nicht von den Lernenden durchgeführt wird, handelt es sich nicht um eine Übung im engeren Sinne.
- H1: Angeleitete Übung. Die Lernenden folgen einer Abfolge von Schritten, die von der Lehrperson durchgeführt werden.
- H2: Übung mit Hinweisen. Die Lernenden erhalten eine Aufgabe mit entsprechenden Hinweisen, sodass die Aufgabe innerhalb des vorgegebenen Zeitrahmens gelöst werden kann.

© International Software Testing Qualifications Board

0.6 Die Zertifizierungsprüfung zum KI Test (CT-AI)

Die Prüfung zum KI Test (CT-AI) stützt sich auf die Lernziele in diesem Lehrplan. Alle Abschnitte des Lehrplans sind prüfungsrelevant, mit Ausnahme dieser Einführung sowie der praktischen Übungen, Referenzen und Anhänge. Für die Beantwortung der Prüfungsfragen kann es erforderlich sein, Material zu verwenden, das auf mehr als einem Abschnitt dieses Lehrplans basiert. 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.

Weitere Einzelheiten entnehmen Sie bitte dem Dokument „Exam Structures & Rules tables Compatible with Syllabi on Foundation and Advanced Levels, and Specialist Modules".

Die wichtigste Zulassungsvoraussetzung für alle, die an der CT-AI-Prüfung teilnehmen möchten, ist der Erwerb der ISTQB®-Zertifizierung „Certified Tester Foundation Level" [CTFL].

Es wird dringend empfohlen, dass die Kandidaten außerdem:

- über grundlegende Kenntnisse entweder in der Softwareentwicklung oder im Softwaretesten verfügen, beispielsweise durch sechs Monate Berufserfahrung als System- oder Benutzerabnahmetester, Datenwissenschaftler oder Softwareentwickler;
- einen nach ISTQB®-Standards akkreditierten Kurs (durch eines der von ISTQB anerkannten nationalen Mitglieds-Boards) absolvieren.

0.7 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 Quality Board; in Österreich: Austrian Testing Board) beziehen oder bei der Organisation, 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.

0.8 Umgang mit Normen und Standards

Internationale Normungsorganisationen wie IEEE und ISO haben Normen zu Qualitätsmerkmalen und Softwaretests herausgegeben. Der Zweck dieser Verweise besteht darin, einen Rahmen zu bieten (wie bei den Verweisen auf ISO/IEC 25059 und ISO/IEC 25010 bezüglich eines Qualitätsmodells für KI-basierte Systeme) oder dem Leser auf Wunsch zusätzliche Informationen zur Verfügung zu stellen. Zu beachten ist, dass der Lehrplan Normen und Standards als Referenz verwendet. Die Normen- bzw. Standard-dokumente selbst sind nicht für die Prüfung relevant.

0.9 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 KI Test (CT-AI) beschreiben.
- Eine Liste von Begriffen, an die sich die Lernenden erinnern können müssen.
- Lernziele für jeden Wissensbereich, welche die zu erreichenden kognitiven Lernergebnisse beschreiben.

© International Software Testing Qualifications Board

- 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 KI Test (CT-AI) behandelt wird. Er konzentriert sich auf die Einführung in die grundlegenden Konzepte der künstlichen Intelligenz (KI) und des maschinellen Lernens (ML) im Besonderen und darauf, wie Systeme, die auf diesen Techniken basieren, getestet werden können.

Der Lehrplan verwendet für Begriffe des Softwaretestens und der Qualitätssicherung die Terminologie (d. h. Bezeichnungen und Bedeutungen) gemäß ISTQB®-Glossar.

0.10 Aufbau des Lehrplans

Es gibt sieben 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 **19,5** Unterrichtsstunden, die sich wie folgt auf die sieben Kapitel verteilen:

- Kapitel 1: Einführung in die Künstliche Intelligenz – 120 Minuten
  - Verstehen der wesentlichen Unterschiede zwischen KI-basierten Systemen und herkömmlichen Systemen und Erkunden des Spektrums der KI-Fähigkeiten von schwacher KI bis hin zu Super-KI.
  - Erwerb eines grundlegenden Verständnisses von KI-Technologien, einschließlich Generativer KI (GenAI), sowie der Hardware, Hosting-Lösungen und Entwicklungs-Frameworks, die zur Implementierung von Systemen für maschinelles Lernen (MLS) verwendet werden.
  - Erfahren, wie Vorschriften und Standards die Entwicklung und das Testen von KI-basierten Lösungen beeinflussen.
- Kapitel 2: Qualitätsmerkmale für KI-basierte Systeme – 45 Minuten
  - Kennenlernen der spezifischen Qualitätsmerkmale für KI-basierte Systeme, einschließlich der in ISO/IEC 25059 definierten, und Verstehen sicherheitsrelevanter Überlegungen beim Einsatz von KI in kritischen Systemen.
  - Erfahren, wie geeignete Akzeptanzkriterien definiert werden, die auf das besondere Verhalten und die Leistung KI-basierter Lösungen zugeschnitten sind.
- Kapitel 3: Maschinelles Lernen (ML) – 375 Minuten
  - Vertraut machen mit den Arten des maschinellen Lernens, den wichtigsten Schritten im ML-Workflow sowie damit, wie vortrainierte Modelle, Fein-Tuning und Retrieval-augmentierte Generierung zu modernen KI-basierten Systemen beitragen.
  - Mehr erfahren über die Datenvorbereitung, die Rolle von Trainingsdatensätzen, Validierungsdatensätzen und Testdaten sowie darüber, wie diese die Entwicklung und Leistung von ML-Modellen beeinflussen.
  - Auseinandersetzen mit neuronalen Netzen, einschließlich ihrer Struktur und Überdeckung, und Sammeln praktischer Erfahrungen mit Metriken und der Verwendung einer Konfusionsmatrix.
- Kapitel 4: Testen KI-basierter Systeme – 195 Minuten
  - Verstehen der besonderen Herausforderungen beim Testen KI-basierter Systeme, einschließlich der Unterschiede in der Testbarkeit zwischen fixierten und adaptiven KI-Systemen, der Notwendigkeit statistischer Tests und der Schwierigkeiten bei der Definition von Testorakeln.
  - Lernen, wie GenAI und Large Language-Modelle (LLM) getestet werden, unter Verwendung von Techniken wie Red-Teaming und dem explorativen Test eines KI-Systems, das Testaufgaben ausführt.

© International Software Testing Qualifications Board

- Entdecken von Teststrategien für MLS, die verschiedene Teststufen und die Anwendung risikobasierter Tests umfassen.

- Kapitel 5: Eingabedatentest für ML-Systeme – 180 Minuten
  - Erfahren, wie Eingabedaten für MLS zu testen und zu validieren sind, einschließlich Verfahren zur Erkennung von Verzerrungen, zur Überprüfung der Korrektheit von Kennzeichen, zur Bewertung der Datenrepräsentativität und zum Test der Daten-Pipeline.
- Kapitel 6: Modelltest für ML-Systeme – 225 Minuten
  - Entdecken von Testansätzen zur Risikominderung bei ML-Modellen, darunter Reviews der Dokumentation, Test der funktionalen Leistung von ML sowie die Erkennung von Überanpassung, Unteranpassung und Drift.
  - Kennenlernen fortgeschrittener Testverfahren, darunter gegnerischer Test, A/B-Test, Back-to-Back-Test und den Einsatz von Angriffen zur Aufdeckung von Schwachstellen im Modell.
  - Erwerben von praktischem Verständnis des metamorphen Tests, einschließlich der Ableitung und Anwendung von Testfällen, bei denen herkömmliche Testorakel nicht ausreichen.
- Kapitel 7: ML-Entwicklungstest – 30 Minuten
  - Kennenlernen von Testansätzen und Teststrategien, mit denen sich Risiken während der Entwicklung und Bereitstellung von ML-Systemen mindern lassen, um ein robustes Systemverhalten in Produktivumgebungen zu gewährleisten.

© International Software Testing Qualifications Board

1 Einführung in die Künstliche Intelligenz – 120 Minuten

Schlüsselbegriffe
Keine

Spezifische Schlüsselbegriffe für KI
allgemeine KI, KI-basiertes System, künstliche Intelligenz, maschinelles Lernen, ML-Entwicklungs-Framework, schwache KI, Super-KI

Lernziele für Kapitel 1: Die Lernenden können …

1.1 Einführung in die KI

AI-1.1.1 (K2) … zwischen KI-basierten Systemen und konventionellen Systemen unterscheiden.

AI-1.1.2 (K2) … schwache KI, allgemeine KI und Super-KI voneinander unterscheiden.

AI-1.1.3 (K2) … die verschiedenen Arten von KI-Technologien erklären.

AI-1.1.4 (K2) … generative KI erklären.

AI-1.1.5 (K2) … die verfügbaren Hardware-Optionen zur Implementierung von Systemen für maschinelles Lernen vergleichen.

AI-1.1.6 (K2) … die Optionen für die Entwicklung und das Hosting von KI-Modellen vergleichen.

AI-1.1.7 (K2) … die von ML-Entwicklungs-Frameworks bereitgestellten Funktionen zusammenfassen.

AI-1.1.8 (K2) … erklären, wie sich Regulierungen, Normen und Standards auf die Entwicklung und das Testen von KI-basierten Systemen auswirken.

© International Software Testing Qualifications Board

1.1 Einführung in die KI

Dieses Kapitel stellt die wesentlichen Unterschiede zwischen konventionellen und KI-basierten Systemen vor und konzentriert sich dabei auf deren Designansätze, Anpassbarkeit und Erklärbarkeit. Es beschreibt das Spektrum der KI-Fähigkeiten, das von schwacher KI über allgemeine KI bis hin zu Super-KI reicht, und gibt einen Überblick über Kerntechnologien wie ML, Deep Learning und GenAI. Das Kapitel beschreibt gängige Hardware- und Entwicklungsumgebungen für KI-basierte Systeme sowie beliebte ML-Entwicklungs-Frameworks. Abschließend werden wesentliche regulatorische und technische Normen und Standards betrachtet, die für die verantwortungsvolle Entwicklung und den Einsatz von KI in verschiedenen Bereichen als Leitlinien dienen.

1.1.1 KI-basierte Systeme und konventionelle Systeme

Konventionelle Computersysteme werden in der Regel mit imperativen Sprachen programmiert, bei denen menschliche Entwickler Schritt-für-Schritt-Anweisungen explizit definieren, einschließlich Konstrukten wie if-then-else-Anweisungen und Schleifen. Dieser deterministische Ansatz trägt dazu bei, das Verhalten des Systems vorhersehbar und transparent zu machen, wodurch es für Menschen leichter wird zu verstehen, wie Eingaben zu unterschiedlichen Ausgaben führen. Im Gegensatz dazu folgen die meisten KI-basierten Systeme, insbesondere solche, die ML nutzen, keinen vordefinierten Regeln. Stattdessen analysieren sie Muster in den Daten, um zu bestimmen, wie sie auf neue Eingaben reagieren sollen. Ein KI-basiertes System zur Bilderkennung, das darauf trainiert wurde, Katzen zu identifizieren, stützt sich beispielsweise nicht auf explizit codierte Regeln. Stattdessen lernt es aus einem Datensatz mit Katzenbildern und extrahiert Muster, die es später auf unbekannte Bilder anwendet, um diese korrekt zu klassifizieren.

Ein grundlegender Unterschied zwischen konventionellen und KI-basierten Systemen besteht darin, wie sie an die Problemlösung herangehen. Viele KI-basierte Systeme stützen sich auf probabilistisches Schließen, statistische Inferenz und Mustererkennung, um Ergebnisse zu generieren. Dadurch können KI-Modelle komplexe Formen von Unsicherheit und Mehrdeutigkeit effektiver bewältigen, was zu Ergebnissen führt, die nicht immer vorhersehbar sind.

Eine zentrale Herausforderung bei KI-basierten Systemen ist die Erklärbarkeit. Viele KI-Modelle, insbesondere Deep-Learning-Architekturen, können Milliarden von Parametern enthalten, was ihre interne Funktionsweise für Menschen schwer interpretierbar macht. Dieser „Black-Box“-Charakter wirft Bedenken in kritischen Bereichen wie Gesundheitswesen, Finanzen, Verteidigung und Verkehr auf, in denen es entscheidend ist, zu verstehen, warum ein KI-Modell eine bestimmte Entscheidung getroffen hat. Die Gewährleistung von Transparenz (siehe 2.1.1) und Erklärbarkeit bei KI-gesteuerten Entscheidungsprozessen ist zu einem zentralen Schwerpunkt der KI-Regulierung geworden.

Ein weiterer wesentlicher Unterschied ist die Anpassbarkeit. Konventionelle Systeme sind statisch und erfordern in der Regel manuelle Aktualisierungen, um neues Wissen zu integrieren oder auf Veränderungen in der Umgebung zu reagieren. KI-basierte Systeme hingegen können selbstlernend sein und ihre Leistung kontinuierlich verbessern, wenn sie auf neue Daten stoßen. Diese Anpassbarkeit macht KI in dynamischen Umgebungen besonders leistungsfähig. Anpassbarkeit erfordert zudem eine kontinuierliche Überwachung, um die fortwährende Übereinstimmung mit den Kernanforderungen aufrechtzuerhalten.

1.1.2 Schwache KI, allgemeine KI und Super-KI

Schwache KI ist darauf ausgelegt, bestimmte Aufgaben auszuführen, und umfasst alle derzeit eingesetzten KI-Systeme. Auf schwacher KI basierende Systeme arbeiten in einem begrenzten Anwendungsbereich und können bei der Lösung spezialisierter Probleme, wie Bilderkennung, Sprachverarbeitung und Sprachübersetzung, äußerst effizient sein. Ihnen fehlt jedoch die Fähigkeit, über die Funktionen, die sie gelernt haben, hinaus zu verallgemeinern. So kann beispielsweise ein KI-basiertes System, das sich durch hervorragende Gesichtserkennung auszeichnet, keine Sprachübersetzungen durchführen, es sei denn, es wird explizit dafür umtrainiert. Frontier-KI, eine Untergruppe der schwachen KI, stellt die fortschrittlichste

© International Software Testing Qualifications Board

Form dieser Systeme dar und erweitert mit GenAI-Modellen die Grenzen der derzeitigen Fähigkeiten. Frontier-KI umfasst groß angelegte Systeme mit hochgradig autonomen Entscheidungsfähigkeiten; sie bleiben jedoch auf bestimmte Aufgaben beschränkt und haben die Vielseitigkeit einer allgemeinen KI noch nicht erreicht.

Allgemeine KI, auch als starke KI bekannt, bezeichnet ein KI-basiertes System, das in der Lage ist, die meisten intellektuellen Aufgaben auszuführen, die ein Mensch bewältigen kann. Allgemeine KI wäre in der Lage, Wissen über ein breites Spektrum von Aufgaben hinweg zu verstehen, zu lernen und anzuwenden, ohne für jede neue Aufgabe neu trainiert werden zu müssen. Diese Art von KI würde menschenähnliches Denken und Anpassbarkeit zeigen und in der Lage sein, unbekannte Probleme in verschiedenen Bereichen zu lösen, ähnlich wie Menschen es tun. Trotz bedeutender Fortschritte in der KI verfügt heute kein KI-basiertes System über allgemeine Intelligenz.

Super-KI (künstliche Superintelligenz) ist die Art von KI, bei der sich ein KI-basiertes System kontinuierlich selbst verbessert, ohne dass menschliches Eingreifen oder Kontrolle erforderlich ist. Damit Super-KI möglich ist, ist ein Internetzugang nicht zwingend erforderlich, doch ein solcher Zugang könnte ihre Fähigkeiten und ihren Einfluss erheblich erweitern. Sie würde die menschliche Intelligenz und die allgemeine KI übertreffen, was nach Ansicht vieler ein existenzielles Risiko für die Menschheit darstellen würde. Der Zeitpunkt, an dem KI-basierte Systeme von allgemeiner KI zu Super-KI übergehen, falls dies überhaupt geschehen sollte, wird gemeinhin als technologische Singularität bezeichnet.

1.1.3 Verschiedene Arten von KI-Technologien

Künstliche Intelligenz (KI) umfasst eine Reihe von Technologien, die jeweils für bestimmte Aufgaben und Herausforderungen geeignet sind. Einer der Kernzweige der KI ist das maschinelle Lernen (ML), das es Systemen ermöglicht, aus Daten zu lernen und Modelle ohne explizite Programmierung zu erstellen. Während sich manche ML-Systeme während ihrer gesamten Lebensdauer kontinuierlich an neue Daten anpassen und verbessern können, arbeiten andere auf der Grundlage des ursprünglich erlernten Wissens und erfordern ein explizites Umtrainieren, um ihre Fähigkeiten zu aktualisieren. ML umfasst mehrere Ansätze:

- Überwachtes Lernen nutzt gekennzeichnete (*labeled*) Daten und Algorithmen wie lineare Regression und Entscheidungsbäume für Aufgaben wie Vorhersagen und Klassifikation.
- Unüberwachtes Lernen deckt Muster in ungekennzeichneten Daten durch Funktionen wie Clusterbildung (unter Verwendung von Clusterbildungsalgorithmen) auf.
- Bestärkendes Lernen ermöglicht es intelligenten Agenten, optimale Verhaltensweisen zu erlernen, indem sie durch Versuch und Irrtum (Trial-and-Error) mit ihrer Umgebung interagieren.

Zu den wichtigsten ML-Technologien gehören neuronale Netze, Bayes‘sche Modelle, Support-Vektor Maschinen (SVM) und Random Forests. Weitere Informationen zu ML finden Sie im Kapitel 3.

Deep Learning (DL), eine Untergruppe der ML-Methoden, nutzt tiefe neuronale Netze zur Lösung komplexer Probleme. Zum Beispiel:

- Faltende neuronale Netze (*convolutional neural networks*, CNN) sind äußerst effektiv bei der Bilderkennung und Objekterkennung.
- Rückgekoppelte (rekurrente) neuronale Netze (*recurrent neural networks*, RNN) sind auf die Verarbeitung sequenzieller Daten wie Text oder Zeitreihen spezialisiert.
- Transformer verarbeiten weitreichende Abhängigkeiten in Sequenzen und bilden die Grundlage für Modelle zur Verarbeitung natürlicher Sprache sowie für Vision-Transformer für Bilder.

GenAI baut auf diesen Technologien auf, um neue Inhalte zu erstellen, darunter Text, Bilder und Audio (siehe 1.1.4). Dies wird durch Modelle wie LLM vorangetrieben, die tiefe neuronale Netze (*deep neural networks*, DNN) mit der Verarbeitung natürlicher Sprache (*natural language processing*, NLP) kombinieren, um menschenähnliche Sprache zu analysieren und zu generieren.

Weitere spezialisierte KI-Technologien umfassen:

© International Software Testing Qualifications Board

- NLP zur Sprachanalyse, einschließlich Aufgaben wie Stimmungsanalyse und maschinelle Übersetzung.
- Computer Vision zur Analyse visueller Daten und zur Unterstützung von Anwendungen wie Gesichtserkennung und Robotik.
- Fuzzy-Logik für Schlussfolgerungen unter Unsicherheit.
- Suchalgorithmen zur Lösung von Optimierungsproblemen, wie z. B. Navigation und strategische Entscheidungsfindung.
- Regelbasierte Schlussfolgerungssysteme oder Expertensysteme zur strukturierten Entscheidungsunterstützung.

Obwohl die vollständige Integration dieser KI-Technologien nach wie vor begrenzt ist, zeigen neue Entwicklungen wie LLM das Potenzial, verschiedene KI-Technologien zu einheitlichen, intelligenten Systemen zu kombinieren. Die agentenbasierte KI erweitert diese Technologien durch autonome Agenten, die unabhängig planen, schlussfolgern und handeln, um Ziele in dynamischen Umgebungen zu erreichen.

1.1.4 Generative KI

Generative KI (GenAI) bezeichnet KI-basierte Systeme, die auf die Erstellung neuer Inhalte wie Texte, Bilder, Videos, Musik oder komplexer Daten spezialisiert sind, wobei viele davon auch Funktionen für Klassifikation und Vorhersage unterstützen. Diese Systeme lernen aus riesigen Datenmengen, um Ergebnisse zu erzeugen, die ihren Trainingsdaten ähneln, was eine Vielzahl kreativer und praktischer Anwendungen ermöglicht.

Zu den wichtigsten Technologien hinter GenAI gehören generative gegnerische Netze (*generative adversarial networks*, GAN), Diffusionsmodelle und Transformer. GANs nutzen zwei miteinander konkurrierende neuronale Netze, um hochrealistische synthetische Daten zu erzeugen. Diffusionsmodelle generieren Inhalte, indem sie den Daten schrittweise Rauschen hinzufügen und anschließend wieder entfernen, was zu hoher Qualität der Ergebnisse führt. Transformer-Modelle, die LLMs zugrunde liegen, nutzen Selbstaufmerksamkeitsmechanismen, um kohärenten und kontextbezogenen Text zu generieren, und werden zunehmend für multimodale Aufgaben angepasst.

Über ihre technischen Grundlagen hinaus werfen GenAI-Systeme erhebliche gesellschaftliche und ethische Bedenken auf. Missbrauch ist ein großes Problem: Diese Modelle können dazu missbraucht werden, Deepfakes zu erstellen, Falschinformationen zu verbreiten oder überzeugende betrügerische Inhalte zu generieren, wodurch das Vertrauen in digitale Medien und den öffentlichen Diskurs untergraben wird. Die Leichtigkeit, mit der synthetische Inhalte produziert werden können, verstärkt Risiken in Bezug auf Datenschutz, Sicherheit und Manipulation, kann aber auch erhebliche Vorteile mit sich bringen und eröffnet Möglichkeiten für Innovationen in den Bereichen Unterhaltung, Marketing, Bildung und Forschung.

Auch die Auswirkungen auf die Beschäftigung sind bemerkenswert, insbesondere bei Angestellten. Da GenAI Aufgaben wie das Schreiben, das Entwerfen, das Programmieren und sogar die Erstellung juristischer oder medizinischer Dokumente automatisiert, wird zunehmend über einen möglichen Arbeitsplatzverlust diskutiert. Während diese Technologien die Produktivität steigern und neue kreative Möglichkeiten eröffnen können, können sie auch zu Umbrüchen in der Arbeitswelt führen und eine umfassende Umschulung erforderlich machen.

Nachhaltigkeit ist ein weiteres vordringliches Thema. Das Trainieren und Betreiben großer GenAI-Modelle erfordert erhebliche Rechenressourcen, was zu einem hohen Energieverbrauch und einem erheblichen CO2-Fußabdruck führt. Diese Umweltauswirkungen haben Forderungen nach effizienteren Modellentwürfen und einer umweltfreundlicheren Infrastruktur laut werden lassen.

Die meisten praktischen GenAI-Werkzeuge basieren heute auf Foundation-Modellen, die dann für spezifische Anwendungen durch Fein-Tuning angepasst werden. Der Bereich entwickelt sich zudem in Richtung multimodaler Modelle, die Inhalte aus Text, Bildern und Audio verarbeiten und generieren können, was reichhaltigere und flexiblere KI-basierte Systeme ermöglicht. Regulierungsrahmen wie die KI-

© International Software Testing Qualifications Board

Verordnung der EU [EU AI Act] entstehen, um die verantwortungsvolle Entwicklung und Nutzung dieser Technologien zu lenken.

1.1.5 Hardware für ML-Systeme

Für ML-Systeme wird eine Vielzahl von Hardware verwendet. Für Training und Inferenz können unterschiedliche Arten von Hardware zum Einsatz kommen. So kann beispielsweise ein Spracherkennungsmodell auf einem Low-End-Smartphone laufen, obwohl für das Training möglicherweise der Zugriff auf die Rechenleistung von Cloud-Computing erforderlich ist.

ML profitiert in der Regel von Hardware, die Folgendes unterstützt:

- Die Fähigkeit, mit großen Datenstrukturen zu arbeiten.
- Massiv parallele Verarbeitung, beispielsweise zur Unterstützung von Matrixmultiplikationen.
- Arithmetik mit geringer Präzision (Quantisierung): Dieser Ansatz verwendet weniger Bits für die Berechnung (z. B. 4 Bits statt 32 Bits), was zu einer schnelleren Verarbeitung, geringerem Stromverbrauch, kleineren und kostengünstigeren Chips sowie reduzierten Bandbreitenanforderungen führt.

Allgemeine Prozessoren (CPU) bieten Unterstützung für komplexe Operationen mit hoher Präzision, die für ML-Anwendungen in der Regel nicht erforderlich sind, verfügen jedoch meist nur über wenige Kerne. Daher ist ihre Architektur für das ML-Modelltraining und die Ausführung von ML-Modellen weniger effizient als die von Grafikprozessoren (GPUs), die über Tausende von Kernen verfügen und für massiv parallele, aber relativ einfache Grafikverarbeitung entworfen sind. Folglich übertreffen GPUs in ML-Anwendungen in der Regel CPUs, obwohl CPUs meist mit höheren Taktraten laufen. Für kleinere ML-Aufgaben bieten GPUs im Allgemeinen die beste Option.

Bestimmte Hardware ist speziell für KI konzipiert, wie beispielsweise maßgeschneiderte anwendungsspezifische integrierte Schaltkreise (ASICs) und System-on-a-Chip-Bausteine. Diese KI-spezifischen Lösungen zeichnen sich durch mehrere Kerne, eine spezialisierte Datenverwaltung und die Fähigkeit zur In-Memory-Verarbeitung aus. Sie eignen sich am besten für Edge-Computing, während das ML-Modelltraining in der Cloud unter Verwendung spezialisierter Hardware erfolgt.

KI-spezifische Hardwarearchitekturen werden kontinuierlich weiterentwickelt. Dazu gehören neuromorphe Prozessoren, die nicht die herkömmliche Von-Neumann-Architektur verwenden, sondern vom Gehirn inspirierte Architekturen, die neuronale Strukturen nachahmen.

1.1.6 Entwicklung und Hosting von KI-Modellen

KI-basierte Systeme können von Drittanbietern bezogen oder intern innerhalb eines Unternehmens entwickelt werden. KI-basierte Systeme stützen sich in der Regel auf vortrainierte Modelle und können vor Ort oder in der Cloud bereitgestellt werden, wobei die Modelle selbst entweder vor Ort oder in der Cloud gehostet werden; bei cloudbasierten Optionen erfolgt der Zugriff häufig als Dienst. KI-basierte Systeme von Drittanbietern werden in der Regel als vortrainierte Modelle oder als „KI als Dienst“ (*AI as a Service*, AIaaS) angeboten, was eine schnellere Bereitstellung und eine kürzere Markteinführungszeit ermöglicht. Alternativ lassen sich interne KI-basierte Systeme (vor Ort oder in maßgeschneiderten Cloud-Konfigurationen) besser an spezifische Anforderungen anpassen, doch ihre Entwicklung erfordert wahrscheinlich spezielle Fachkenntnisse, die entweder von internen Experten oder von externen Teams bereitgestellt werden müssen.

Die lokale Entwicklung (lokales Programmieren und Trainieren) ermöglicht direkte Kontrolle und Datenschutz. Kleine Modelle, wie Entscheidungsbäume oder kompakte neuronale Netze, können auf PCs entwickelt werden, während mittelgroße Modelle möglicherweise dedizierte GPUs erfordern. Für groß angelegte Modelle werden leistungsstarke Servercluster vor Ort erforderlich, mit den damit verbundenen Kosten für Energie, Kühlung und Hardware.

© International Software Testing Qualifications Board

Die Cloud-Entwicklung bietet erhebliche Flexibilität. Insbesondere öffentliche Clouds stellen vorkonfigurierte Umgebungen mit nutzungsabhängiger Abrechnung bereit, was die anfänglichen Hardware-Investitionen begrenzt und eine einfache Skalierung ermöglicht. Im Gegensatz dazu können private Clouds für Anwendungen, die dies benötigen, erhöhte Sicherheit und Datenschutz bieten, doch diese Kontrolle erfordert eine größere Vorabinvestition in die Infrastruktur.

Viele Unternehmen setzen auf hybride Ansätze, darunter die lokale Entwicklung von Prototypen vor der Skalierung auf die Cloud-Infrastruktur, die lokale Verwaltung sensibler Komponenten, beispielsweise die Aufbereitung privater Daten, und die Nutzung von Cloud-Ressourcen für rechenintensive Aufgaben.

KI-Modelle können in verschiedenen Umgebungen gehostet werden, die von lokalen Installationen bis hin zu cloudbasierten Plattformen reichen. Lokales Hosting beinhaltet die Ausführung kleinerer Modelle auf PCs oder Smartphones, was Datenschutz bietet und Cloud-Lizenzkosten vermeidet, allerdings sind die Hardware-Kapazitäten dabei begrenzt. Für größere KI-Modelle können Unternehmen dedizierte Server einrichten, was zwar eine erhebliche Vorabinvestition erfordert, aber eine verbesserte Kontrolle ermöglicht.

Das Cloud-Hosting von KI-Modellen kann in öffentlichen oder privaten Clouds erfolgen. Öffentliche Cloud-Dienste bieten Skalierbarkeit bei Robustheit und Leistungsstärke; das löst Probleme der Wartung und macht sie für schwankende Arbeitslasten ideal geeignet. Private Clouds bieten ähnliche Vorteile, jedoch mit erhöhter Sicherheit und mehr Anpassungsmöglichkeiten, und werden entweder intern oder durch spezialisierte Anbieter verwaltet, wenn auch zu höheren Kosten.

Hybride Ansätze kombinieren diese Methoden und ermöglichen es Unternehmen, bestimmte Vorgänge lokal auszuführen und gleichzeitig die Flexibilität der Cloud für rechenintensive Aufgaben zu nutzen.

Die optimalen Entwicklungs- und Hosting-Lösungen, über die in der Regel unabhängig voneinander entschieden wird, hängen von Faktoren wie Modellgröße, Komplexität, Leistungsanforderungen, Budgetbeschränkungen, Aspekten der Sicherheit und des Datenschutzes, Bereitstellungsanforderungen und regulatorischen Anforderungen ab. Einige Unternehmen verfolgen mehrstufige Strategien, um Effizienz und Kontrolle in Einklang zu bringen.

1.1.7 Entwicklungs-Frameworks für maschinelles Lernen

ML-Entwicklungs-Frameworks bieten einen Werkzeugkasten zum Erstellen und Trainieren von ML-Modellen. Zu den typischen Funktionen dieser Frameworks gehören:

- Datenhandhabung: Sie unterstützen das Laden, Vorverarbeiten und Verwalten der Daten, die zum Trainieren und Testen des Modells verwendet werden. Dies kann das Bereinigen, Formatieren und Umwandeln der Daten in ein für das gewählte Modell geeignetes Format umfassen.
- Modellbildung: Diese Frameworks bieten Bibliotheken mit ML-Algorithmen und Werkzeuge zum Entwurf der Architektur des zu erstellenden ML-Modells. Dazu gehört die Festlegung des Modelltyps (z. B. neuronales Netz, Entscheidungsbaum), der Anzahl der Schichten und Verbindungen sowie der im Modell durchgeführten mathematischen Operationen.
- Training und Optimierung: Frameworks, die Algorithmen bereitstellen, welche die internen Parameter des Modells iterativ auf der Grundlage der Trainingsdaten und des gewünschten Ergebnisses anpassen. Das Ziel ist es, die Leistung des Modells bei der Erfüllung der gewünschten Aufgabe (z. B. Klassifikation, ML-Regression) zu optimieren. Manche Frameworks können verteiltes Training unterstützen und vortrainierte Modelle erweitern oder per Fein-Tuning optimieren.
- Evaluierung: Sie bieten Werkzeuge, um zu bewerten, wie gut das trainierte Modell bei unbekannten Daten abschneidet. Dies kann die Messung von Genauigkeit, Präzision und Sensitivität bei Aufgaben der Klassifikation oder von Fehlerraten bei Aufgaben der Regression umfassen (siehe 3.1.1).
- Bereitstellung: Manche Frameworks bieten Funktionen für die Bereitstellung des trainierten Modells im realen Betrieb. Dies kann die Konvertierung des Modells in ein Format umfassen, das für die Integration mit Webanwendungen, mobilen Geräten, Edge-Geräten oder eingebetteten Systemen geeignet ist.

© International Software Testing Qualifications Board

Diese Frameworks können auf verschiedenen Abstraktionsebenen arbeiten. Einige bieten eine Anwendungsprogrammierschnittstelle (*application programming interface*, API) auf niedrigerer Ebene, die Entwicklern mehr Kontrolle über die Modellerstellung gewährt, jedoch mehr Programmierkenntnisse erfordert. Andere bieten eine API auf höherer Ebene, welche die Modellerstellung vereinfacht, aber weniger Anpassungsmöglichkeiten bietet.

Verschiedene Frameworks können sich auf unterschiedliche Anwendungsbereiche konzentrieren. Einige sind universell einsetzbar und unterstützen ein breites Spektrum an Anwendungsbereichen. Im Gegensatz dazu sind andere spezialisierter und konzentrieren sich auf bestimmte Bereiche wie Bilderkennung, Spracherkennung und Sprachübersetzung.

Die Auswahl des am besten geeigneten Frameworks kann von verschiedenen Faktoren abhängen, wie zum Beispiel:

- der Anwendungsbereich;
- die Notwendigkeit einer benutzerfreundlichen Benutzungsschnittstelle für Rapid Prototyping;
- Konfigurierbarkeit für komplexe Modelle;
- das Fachwissen der Benutzer;
- Überlegungen zur Bereitstellung, da einige Frameworks besser für ressourcenbeschränkte Umgebungen geeignet sind;
- Umfang der (Community-)Unterstützung;
- Reife des Ökosystems.

1.1.8 Regularien und Standards für KI

KI-Regularien und -Normen sind entscheidend für die verantwortungsvolle Entwicklung, die Bereitstellung und die Nutzung von KI. Das Kernziel besteht darin, Vertrauen in KI zu fördern und dazu beizutragen, die Vorteile von KI zu realisieren und gleichzeitig potenzielle Schäden zu mindern. Im Idealfall sollte die Konformität mit solchen Regularien und Standards gewährleisten, dass KI-basierte Systeme sicher, fair, transparent, nachhaltig, zurechenbar, ethisch und verantwortungsvoll eingesetzt werden.

Auf internationaler Ebene dienen die OECD-KI-Grundsätze [OECD AI] und der UN-Bericht zur KI-Governance für die Menschheit [UN Gov AI] als einflussreiche Soft-Law-Instrumente, die ein gemeinsames Verständnis für einen verantwortungsvollen Umgang mit KI fördern. Sie dienen nationalen Regierungen und Unternehmen als Kompass bei der Formulierung ihrer eigenen KI-Strategien. Diese Grundsätze betonen eine menschenzentrierte KI, ethische Überlegungen und die Bedeutung der internationalen Zusammenarbeit.

Die KI-Verordnung der EU [EU AI Act] stellt einen wegweisenden regulatorischen Schritt dar und verfolgt einen risikobasierten Ansatz zur Regulierung von KI. Durch die Einstufung KI-basierter Systeme nach ihrem Risiko, von minimal bis inakzeptabel, werden die Vorschriften entsprechend abgestuft. Systeme mit hohem Risiko, insbesondere solche, die Grundrechte oder die Sicherheit beeinträchtigen, unterliegen strengen Anforderungen, die rigorose Tests, Datenverwaltung und menschliche Aufsicht umfassen. Die erheblichen finanziellen Sanktionen bei Nichteinhaltung, die sich nach einem Prozentsatz des weltweiten Umsatzes richten, unterstreichen das Engagement der EU für die Durchsetzung. Im Gegensatz dazu verfolgen viele Länder außerhalb der EU einen eher freizügigen Ansatz und bevorzugen eine weniger strenge Regulierung, um Innovationen zu fördern.

Technische Normen, die von Organisationen wie ISO und IEEE entwickelt werden, sind entscheidend, um übergeordnete Erwartungen in die Praxis zu überführen. Sie liefern konkrete technische Spezifikationen und bewährte Verfahren und schließen so die Lücke zwischen Politik und Praxis. So bietet beispielsweise ISO/IEC TR 29119-11 detaillierte Leitlinien für das Testen KI-basierter Systeme, ein entscheidendes Element zum Nachweis der Konformität. Gleichzeitig wird die Normenreihe ISO/IEC 42119 entwickelt, um verschiedene Aspekte des Testens KI-basierter Systeme abzudecken. Darüber hinaus entstehen

© International Software Testing Qualifications Board

branchenspezifische Regularien für Bereiche wie das Gesundheitswesen und den Finanzsektor, die den besonderen Risiken von KI dort Rechnung tragen.

Um sich in dieser sich wandelnden Landschaft der KI-Governance effektiv zurechtzufinden, sind kontinuierlicher Dialog und Zusammenarbeit von größter Bedeutung. Regierungen, Industrie, Wissenschaft und Zivilgesellschaft müssen sich aktiv engagieren, um weltweit einen harmonisierten und wirksamen Ansatz für die KI-Governance zu erreichen. Angesichts der Dynamik der KI müssen darüber hinaus Regularien und Standards regelmäßig überprüft und aktualisiert werden, um weiterhin relevant und wirksam als Leitlinien für die verantwortungsvolle Entwicklung, Bereitstellung und Nutzung von KI zu dienen.

© International Software Testing Qualifications Board

2 Qualitätsmerkmale für KI-basierte Systeme – 45 Minuten

Schlüsselbegriffe

Eingreifbarkeit, funktionale Anpassbarkeit, funktionale KI-Korrektheit, KI-Robustheit, Kontrollierbarkeit durch Benutzer, Minderung gesellschaftlicher und ethischer Risiken, Sicherheit (*safety*), Transparenz

Spezifische Schlüsselbegriffe für KI

Keine

Lernziele für Kapitel 2: Die Lernenden können …

2.1 Qualitätsmerkmale für KI-basierte Systeme

AI-2.1.1 (K2) … das Verhalten von KI-basierten Systemen bezüglich der Qualitätsmerkmale gemäß ISO/IEC 25059 klassifizieren.

AI-2.1.2 (K2) … die speziellen Überlegungen erklären, die beim Einsatz von KI in sicherheitsrelevanten Systemen zu berücksichtigen sind.

2.2 Akzeptanzkriterien für KI-basierte Systeme

AI-2.2.1 (K2) … Beispiele für Akzeptanzkriterien für KI-basierte Systeme nennen.

© International Software Testing Qualifications Board

2.1 Qualitätsmerkmale für KI-basierte Systeme

Dieser Abschnitt behandelt KI-spezifische Qualitätsmerkmale gemäß der Norm ISO/IEC 25059, die traditionelle Softwarequalitätsmodelle erweitert, um die besonderen Eigenschaften KI-basierter Systeme zu berücksichtigen. Es werden neue und angepasste Qualitätsmerkmale eingeführt, darunter funktionale KI-Korrektheit, funktionale Anpassbarkeit, Kontrollierbarkeit durch Benutzer, Transparenz, KI-Robustheit und Eingreifbarkeit sowie die Minderung gesellschaftlicher und ethischer Risiken. Darüber hinaus behandelt dieser Abschnitt zentrale Herausforderungen hinsichtlich der Sicherheit von KI, wie unklare Spezifikationen, Nicht-Determinismus, selbstlernende Systeme, eingeschränkte Erklärbarkeit und sich weiterentwickelnde Normen, wobei deren Auswirkungen auf Testaktivitäten und regulatorische Anforderungen hervorgehoben werden.

2.1.1 KI-spezifische Qualitätsmerkmale

Die ISO/IEC 25059 erweitert das Qualitätsmodell aus der ISO/IEC 25010 um KI-spezifische Aspekte. Diese Erweiterung bewertet KI-basierte Systeme aus zwei Perspektiven: Produkt- und Nutzungsqualität. Aus Sicht des Testens beeinflussen diese Qualitätsmerkmale unmittelbar die Definition von Testzielen, die Formulierung von Akzeptanzkriterien und die Interpretation von Testergebnissen für KI-basierte Systeme. Im Vergleich zur ISO/IEC 25010 neue bzw. angepasste Qualitätsmerkmale sind:

- **Funktionale KI-Korrektheit** (Produktqualität): KI-basierte Systeme, insbesondere solche mit probabilistischem maschinellem Lernen (ML), können keine perfekte Genauigkeit garantieren. Da bei KI-Ausgaben eine gewisse Fehlerquote erwartet wird, wurde das Konzept der funktionalen Korrektheit entsprechend angepasst. ISO/IEC 25059 bewertet die funktionale KI-Korrektheit durch die Berücksichtigung sowohl korrekter als auch fehlerhafter Ergebnisse und durch die Definition akzeptabler Schwellenwerte für fehlerhafte Resultate, wodurch die charakteristische Variabilität der Ausgaben KI-basierter Systeme berücksichtigt wird (siehe 3.3).
- **Funktionale Anpassbarkeit** (Produktqualität): Ein neues Teilmerkmal der funktionalen Eignung. Es beschreibt die Fähigkeit eines KI-basierten Systems, sich nach seiner Bereitstellung selbstständig an Veränderungen seiner Produktivumgebung anzupassen.
- **Kontrollierbarkeit durch Benutzer** (Produktqualität): Ein neues Teilmerkmal der Interaktionsfähigkeit (wobei Interaktionsfähigkeit selbst ein neuer Begriff ist, der in der Version 2023 der ISO/IEC 25010 die Gebrauchstauglichkeit ersetzt). Es beschreibt die Eigenschaft eines KI-basierten Systems, die es einem Menschen oder einem anderen externen Agenten ermöglicht, rechtzeitig in dessen Funktionsweise einzugreifen.
- **Transparenz** (Produkt- und Nutzungsqualität): Ein neues Teilmerkmal der Interaktionsfähigkeit sowie ein neues Teilmerkmal der Zufriedenheit. Es beschreibt den Grad, in dem geeignete Informationen über das KI-basierte System an relevante Stakeholder kommuniziert werden (siehe 6.1.2).
- **KI-Robustheit** (Produktqualität): Ein neues Teilmerkmal der Zuverlässigkeit. Es beschreibt die Fähigkeit eines KI-basierten Systems, sein Niveau an funktionaler KI-Korrektheit unabhängig von den Umständen aufrechtzuerhalten, beispielsweise bei verzerrten, gegnerischen oder ungültigen Eingabedaten, externen Störungen, ungünstigen Umgebungsbedingungen oder Fehlbedienungen durch Benutzer.
- **Eingreifbarkeit** (Produktqualität): Ein neues Teilmerkmal der Sicherheit (*security*). Es beschreibt den Grad, in dem ein Bediener rechtzeitig in die Funktionsweise eines KI-basierten Systems eingreifen kann, um Schäden oder Gefährdungen zu verhindern.
- **Minderung gesellschaftlicher und ethischer Risiken** (Nutzungsqualität): Ein neues Teilmerkmal von „Freiheit von Risiken“ (*freedom from risk*). Es berücksichtigt zahlreiche Aspekte zur Minderung gesellschaftlicher und ethischer Risiken, darunter Zurechenbarkeit, Fairness und Nichtdiskriminierung, berufliche Verantwortung, Förderung menschlicher Werte, Datenschutz, Sicherheit (*safety*) und Sicherheit (*security*), menschliche Kontrolle über Technologie, Einbindung und Entwicklung von Gemeinschaften, menschenzentriertes Design, Einhaltung der Rechtsstaatlichkeit, Respekt vor internationalen Verhaltensnormen, ökologische Nachhaltigkeit und Arbeitspraktiken.

© International Software Testing Qualifications Board

2.1.2 KI und Sicherheit (*safety*)

Sicherheitsrelevante Systeme können Personen, Sachwerte oder die Umwelt schädigen. Die Entwicklung und das Testen klassischer sicherheitsrelevanter Systeme ohne KI erfordern zwar einen hohen Aufwand, sind jedoch grundsätzlich beherrschbar. Für KI-basierte Systeme ergeben sich jedoch zusätzliche Herausforderungen:

- Spezifikationen: Bei traditionellen sicherheitsrelevanten Systemen werden Anforderungen für das Gesamtsystem definiert und so weit verfeinert, bis Entwickler diese in Code umsetzen können. Die Anforderungen vieler KI-basierter Systeme beginnen hingegen häufig mit allgemeinen Zielvorgaben und werden anschließend implizit über Trainingsdaten bereitgestellt, die Muster, Regeln und Zielsetzungen kodieren, ohne jedes Detail im Voraus vollständig zu formalisieren. Dadurch kann die notwendige Verfolgbarkeit von Anforderungen bis zur Implementierung bei KI-basierten Systemen unzureichend sein.
- Nicht-Determinismus: Diese Eigenschaft vieler KI-basierter Systeme erschwert es grundsätzlich, ihr exaktes Verhalten zu garantieren. Selbst intensiv getestete Modelle können aufgrund von Faktoren wie Zufallszahlengenerierung oder geringfügigen Variationen der Eingabewerte unerwartetes Verhalten zeigen.
- Selbstlernende Systeme: Durch gründliche Tests wird vor der Bereitstellung die Sicherheitsintegrität eines Systems nachgewiesen. Bei selbstlernenden KI-basierten Systemen wird dies jedoch dadurch beeinträchtigt, dass sich das Verhalten des Systems zunehmend vom ursprünglich getesteten Verhalten entfernt. Die Kontrolle des Lernprozesses des Modells und der verwendeten Daten kann manchmal helfen, das Entstehen neuer problematischer Verhaltensweisen zu vermeiden. Alternativ können Sicherheitsmechanismen implementiert werden, die verhindern, dass das Modell lernt oder Entscheidungen trifft, welche die Sicherheit gefährden könnten (z. B. eine Komponente zur Inhaltsmoderation, die Prompts filtert).
- Erklärbarkeit und Transparenz: Für sicherheitsrelevante Systeme ist es essenziell zu verstehen, wie und warum sie Entscheidungen treffen. Die Entscheidungsprozesse KI-basierter Systeme sind jedoch häufig nicht transparent. Verfahren der erklärbaren KI (XAI), wie beispielsweise das LIME-Verfahren (*local interpretable model-agnostic explanations*), können Einblicke in die Entscheidungsfindung eines KI-basierten Systems liefern. Diese Verfahren sind jedoch noch nicht weit verbreitet und können die Leistungsfähigkeit des Systems beeinträchtigen.
- Sich weiterentwickelnde Regulierung: Das regulatorische Umfeld für sicherheitsrelevante KI-basierte Systeme entwickelt sich fortlaufend weiter. Der Einsatz von KI ist derzeit in etablierten internationalen Normen zur funktionalen Sicherheit noch nicht berücksichtigt; einige dieser Normen untersagen ihren Einsatz sogar ausdrücklich. Die KI-Verordnung der EU [EU AI Act] (siehe 1.1.8) klassifiziert KI-Systeme, die als Sicherheitskomponenten eingesetzt werden (z. B. in der Luftfahrt, bei Medizinprodukten oder im Automobilbereich), als Hochrisiko-Systeme und stellt strenge Anforderungen an deren Entwicklung und Testen.

2.2 Akzeptanzkriterien für KI-basierte Systeme

Dieser Abschnitt beschreibt Akzeptanzkriterien in Bezug auf die in der Norm ISO/IEC 25059 definierten Qualitätsmerkmale sowie auf Sicherheit. Für KI-basierte Systeme müssen Akzeptanzkriterien häufig statistisch, probabilistisch oder schwellenwertbasiert formuliert werden, anstatt als einfache Binärwerte. Dadurch entstehen zusätzliche Herausforderungen für das Testen.

2.2.1 Akzeptanzkriterien für KI-basierte Systeme

Bei der Evaluierung der Qualität eines KI-basierten Systems ist es wesentlich, sowohl funktionale als auch nicht-funktionale Qualitätsmerkmale zu berücksichtigen. Dadurch kann bestätigt werden, dass das KI-basierte System wie vorgesehen funktioniert und gleichzeitig übergreifende Qualitätsanforderungen erfüllt. Die Normen ISO/IEC 25010 und ISO/IEC 25059 stellen einen umfassenden Rahmen zur Definition von Softwarequalität bereit. In diesem Abschnitt liegt der Fokus auf den Akzeptanzkriterien, die den KI-

© International Software Testing Qualifications Board

spezifischen Qualitätsmerkmalen (d. h, den in der ISO/IEC 25059 definierten) sowie Sicherheitsaspekten (siehe 2.1.2) zugeordnet sind.

Die folgende Tabelle listet beispielhafte Akzeptanzkriterien für die Sicherheit (*safety*) und für jedes der in der Norm ISO/IEC 25059 definierten Qualitätsmerkmale auf.

Qualitätsmerkmal	Beispiele für Akzeptanzkriterien
Funktionale KI-Korrektheit (siehe Abschnitt 3.3)	• Genauigkeit von 95 % für ein Bilderkennungssystem. • Sensitivität von 90 % für ein System zur Fehlervorhersage.
Funktionale Anpassbarkeit	• Das Motormanagementsystem benötigt maximal 20 Sekunden, um sich beim Überschreiten eines definierten Höhenschwellwertes anzupassen. • Ein Video-Streaming-Dienst passt seine Startseite so an, dass mindestens 40 % Dokumentationsvideos empfohlen werden, nachdem ein Benutzer in einer Sitzung drei Dokumentarfilme vollständig angesehen hat.
Kontrollierbarkeit durch Benutzer	• Ein Supervisor kann innerhalb von 0,5 Sekunden die Kontrolle über eine autonome Drohne übernehmen, wenn diese aufgrund des Verlusts ihrer GPS-Position ein Notsignal sendet. • Das landwirtschaftliche Steuerungssystem benachrichtigt den Landwirt, wenn die visuelle Leistungsfähigkeit des Sensors um mehr als 30 % abnimmt. Überschreitet die Verschlechterung 50 % und erfolgt keine Reaktion des Benutzers, wird das System vollständig deaktiviert.
Transparenz	• Es werden ausreichende Informationen über das eingesetzte ML-Modell eines Drittanbieters sowie über die Herkunft seiner Trainingsdaten bereitgestellt, um die Anforderungen des relevanten Unternehmensstandards zu erfüllen. • Das operative Dashboard und die API des Systems müssen einen Endpunkt bereitstellen, der die eindeutige Versionskennung des aktuell eingesetzten Vorhersagemodells sowie einen Link zu dessen Dokumentation bereitstellt.
KI-Robustheit	• Die Antwortzeit der Vorhersagen eines KI-basierten Sicherheitswarnsystems für Penetrationsangriffe bleibt unter einer Sekunde, wenn der Zugriff auf die zentrale Datenbank mit den Sicherheitslücken für 30 Sekunden unterbrochen ist. • Das Edge-KI-Gerät wechselt automatisch in einen Modus mit reduziertem Funktionsumfang und geringerem Energieverbrauch, anstatt abzustürzen, wenn seine interne Betriebstemperatur für 10 Sekunden über 85 °C liegt.
Eingreifbarkeit	• Wenn ein Roboter seine Sicherheitszone verletzt, kann die Produktionslinie innerhalb von 0,5 Sekunden nach Auslösung der Abschaltung gestoppt werden. • Das Stromnetzmanagementsystem stellt ein Bestätigungsfenster von 30 Sekunden bereit, innerhalb dessen ein Ingenieur gegen jede als „kritisch“ eingestufte von der KI vorgeschlagene Aktion Einspruch erheben kann, bevor sie automatisch ausgeführt wird.

© International Software Testing Qualifications Board

Qualitätsmerkmal	Beispiele für Akzeptanzkriterien
Minderung gesellschaftlicher und ethischer Risiken	• Das automatisierte Strafzumessungssystem diskriminiert gemäß der festgelegten Fairness-Metrik nicht zwischen unterschiedlichen ethnischen Gruppen. • Der Chatbot muss eine interne Bewertung durch das Red-Team mit mindestens 95 % bestehen, um zu zeigen, dass er es ablehnt, Inhalte zu generieren, die Gewalt, Selbstverletzung oder Hassreden fördern.
Sicherheit (*safety*)	• Die Nicht-KI-Komponenten des KI-basierten Lenksystems erfüllen die Anforderungen der ISO 26262-6 auf ASIL-C-Niveau. • 100 % der Beziehungen zwischen Ein- und Ausgaben des ML-Modells eines Kernkraftwerks-Steuerungssystems können mit einer durchschnittlichen Genauigkeit von mindestens 99,9 % durch ein Erklärbarkeits-Werkzeug zugeordnet werden. • Die Analyse und Regulierung von Kontrollsignalen, welche die Sicherheitsgrenzen um mehr als 10 % überschreiten, erfolgt innerhalb von 0,15 Sekunden nach ihrer Erkennung durch das Teilsystem zur Sicherheitsüberwachung.

© International Software Testing Qualifications Board

3 Maschinelles Lernen (ML) – 375 Minuten

Schlüsselbegriffe

funktionale Leistungskriterien von ML, funktionale Leistungsmetriken von ML, k-Multisektions-Neuronenüberdeckung, ML-Modell, Neuronen-Grenzwertüberdeckung, Neuronenüberdeckung, Perzeptron

Spezifische Schlüsselbegriffe für KI

Assoziation, bestärkendes Lernen, Clusterbildung, Datenvorbereitung, Klassifikation, maschinelles Lernen, ML-Algorithmus, ML-Entwicklungs-Framework, ML-Regression, ML-Workflow, überwachtes Lernen, unüberwachtes Lernen, vortrainiertes Modell

Lernziele und Ziele der praktischen Übungen für Kapitel 3: Die Lernenden können …

3.1 Einführung in das maschinelle Lernen

AI-3.1.1 (K2) … zwischen den verschiedenen Formen des maschinellen Lernens unterscheiden.

AI-3.1.2 (K2) … den Workflow zur Erstellung eines ML-Systems zusammenfassen.

HO-3.1.3 (H2) … selbstständig ein ML-Modell erstellen.

AI-3.1.4 (K2) … die Verwendung von vortrainierten Modellen, Fein-Tuning und Retrieval-augmentierter Generierung zusammenfassen.

3.2 Daten für maschinelles Lernen

AI-3.2.1 (K2) … die Aktivitäten im Zusammenhang mit der Datenvorbereitung erklären.

HO-3.2.2 (H2) … selbstständig die Datenvorbereitung durchführen, um die Erstellung eines ML-Modells zu unterstützen.

AI-3.2.3 (K2) … die Verwendung von Trainings-, Validierungs- und Testdatensätzen bei der Entwicklung eines ML-Modells voneinander abgrenzen.

3.3 Funktionale Leistungsmetriken von ML zur Klassifikation

AI-3.3.1 (K3) … gängige funktionale Leistungsmetriken von ML anhand eines gegebenen Satzes von Konfusionsmatrix-Daten berechnen.

HO-3.3.2 (H2) … selbstständig ein ML-Modell anhand ausgewählter funktionaler Leistungsmetriken von ML bewerten.

HO-3.3.3 (H2) … selbstständig die Auswirkungen verschiedener ML-Modelle und Datensatzkombinationen aufzeigen.

3.4 Neuronale Netze

AI-3.4.1 (K2) … den Aufbau und die Funktionsweise eines tiefen neuronalen Netzes erklären.

HO-3.4.2 (H1) … die Implementierung eines Perzeptrons nachvollziehen.

AI-3.4.3 (K2) … die verschiedenen Überdeckungsmaße für neuronale Netze beschreiben.

© International Software Testing Qualifications Board

3.1 Einführung in das maschinelle Lernen

Dieser Abschnitt stellt die Hauptkategorien von ML-Algorithmen vor: überwachtes, unüberwachtes und bestärkendes Lernen, wobei die jeweiligen Problemtypen und typischen Anwendungsbereiche unterschieden werden. Er skizziert den Standard-Workflow für die Entwicklung von ML-Modellen, von der Definition der Ziele und der Datenvorbereitung bis hin zum Training, zur Evaluierung und zum Einsatz der Modelle, wobei der Schwerpunkt auf Iteration und Systemintegration liegt. Praktische Aspekte wie die eigenhändige Modellerstellung, die Verwendung vortrainierter Modelle, das Fein-Tuning und die Retrieval-augmentierte Generierung (RAG-Verfahren) werden ebenfalls behandelt. Hierbei werden Ansätze zur effizienten Anpassung und Verbesserung von KI-Modellen für neue Aufgaben bei gleichzeitiger Bewältigung bestehender Beschränkungen hervorgehoben.

Das Verständnis dieses Workflows ist für Tester unerlässlich, da in den verschiedenen Phasen unterschiedliche Testaktivitäten zum Tragen kommen und Fehler häufig bereits in früheren Schritten wie der Datenvorbereitung oder der Modellauswahl ihren Ursprung haben.

3.1.1 Verschiedene Arten maschinellen Lernens

ML-Algorithmen werden in überwachtes Lernen, unüberwachtes Lernen und bestärkendes Lernen unterteilt.

Beim überwachten Lernen trainieren Algorithmen Modelle anhand von gekennzeichneten Daten, wobei jeder Eingabesatz eine entsprechende Kennzeichnung der Ausgabe aufweist (z. B. Bilder, die als „Hund“ oder „Katze“ gekennzeichnet sind). Das Modell lernt, Eingaben bestimmten Ausgabewerten zuzuordnen, indem es Muster in den Trainingsdaten identifiziert. Überwachtes Lernen wird typischerweise unterteilt in:

- Klassifikation: Dabei werden Eingaben vordefinierten Klassen zugeordnet, beispielsweise die Einstufung von E-Mails als Spam oder Nicht-Spam oder die Objekt-Wiederkennung in Bildern.
- ML-Regression: Hierbei geht es um die Vorhersage kontinuierlicher numerischer Werte, wie beispielsweise die Schätzung des Alters einer Person anhand von Lifestyle-Daten oder die Prognose von Aktienkursen.

  Hierbei ist zu beachten, dass sich der Begriff „ML Regression“ im Kontext des maschinellen Lernens von seiner Verwendung in anderen ISTQB®-Lehrplänen unterscheidet, in denen „Regression“ das Problem beschreibt, wenn Softwareänderungen zu änderungsbedingten Fehlern führen.

Beim unüberwachten Lernen trainiert der Algorithmus Modelle anhand ungekennzeichneter Daten und leitet daraus Muster oder Strukturen ab, ohne dass explizite Ausgabekennzeichnungen vorliegen. Das Modell gruppiert ähnliche Eingaben anhand gemeinsamer Merkmale. Das unüberwachte Lernen wird in der Regel wie folgt unterteilt:

- Clusterbildung: Dabei werden Datenpunkte anhand von Ähnlichkeiten gruppiert, beispielsweise durch die Einteilung von Kunden in verschiedene Gruppen für gezieltes Marketing.
- Assoziation: Dabei werden Beziehungen oder Abhängigkeiten zwischen Datenattributen identifiziert, beispielsweise das Erkennen von Mustern im Kaufverhalten von Kunden, um Produktempfehlungen zu geben.

Beim bestärkenden Lernen lernt das KI-basierte System (ein „intelligenter Agent“) durch Interaktion mit seiner Umgebung. Der Agent erhält je nach Ergebnis seiner Handlungen positives Feedback (Belohnungen) oder negatives Feedback (Strafen), wodurch er aus Erfahrungen statt aus einem Datensatz lernen kann. Zu den Herausforderungen beim bestärkenden Lernen gehören die Einrichtung der Umgebung, die Gestaltung der Belohnungsfunktion und die Auswahl der besten Strategie zur Erreichung des gewünschten Ziels. Zu den Anwendungsbereichen zählen Robotik, autonome Fahrzeuge und adaptive Systeme wie Chatbots.

Jeder ML-Ansatz ist auf unterschiedliche Problemtypen ausgerichtet, wobei die Wahl von der Art der verfügbaren Daten und der jeweiligen Aufgabe abhängt.

© International Software Testing Qualifications Board

3.1.2 ML-Workflow

Die folgenden Aktivitäten im ML-Workflow sind in Abbildung 1 auf S. 30 dargestellt:

Verstehen der Ziele

Der Zweck des ML-Modells wird von den Beteiligten verstanden und abgestimmt, um die Übereinstimmung mit den geschäftlichen Prioritäten zu überprüfen. Für das entwickelte Modell werden Akzeptanzkriterien (einschließlich der funktionalen Leistungsmetriken von ML – siehe 3.3) definiert.

Auswahl eines Frameworks

Auf Basis der Ziele, der Akzeptanzkriterien (siehe 2.2) und der geschäftlichen Prioritäten wird ein geeignetes ML-Entwicklungs-Framework (siehe 1.1.7 für Details zu den bereitgestellten Funktionen) ausgewählt.

Auswahl und Umsetzung des Algorithmus

Ein ML-Algorithmus wird auf der Grundlage verschiedener Faktoren ausgewählt, darunter die Ziele, die Akzeptanzkriterien und die verfügbaren Daten (siehe 3.2). Der Algorithmus kann manuell programmiert werden, wird jedoch häufig aus einer Software-Bibliothek abgerufen. Der Algorithmus wird anschließend bei Bedarf kompiliert.

Vorbereitung und Test der Daten

Die Datenvorbereitung (siehe 3.2) umfasst die Datenbeschaffung, die Datenvorverarbeitung und die Merkmalsermittlung. Parallel zu diesen Aktivitäten kann eine explorative Datenanalyse (EDA) durchgeführt werden.

Die vom Algorithmus und vom Modell verwendeten Daten richten sich nach den Zielen und werden in allen Aktivitäten in der in Abbildung 1 dargestellten Aktivität „Modellgenerierung und -test“ genutzt.

Die Daten zum Trainieren, Bewerten, Optimieren und Testen des Modells müssen repräsentativ für die Daten sein, welche das Modell im operativen Einsatz verarbeiten wird.

Es werden die Daten selbst sowie alle automatisierten Datenvorbereitungsschritte getestet (siehe Kapitel 5).

Training des Modells

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

Parameter, die die Modellstruktur definieren (z. B. die Anzahl der Schichten eines neuronalen Netzes oder die Tiefe eines Entscheidungsbaums), werden an den Algorithmus übergeben. Diese Parameter werden als Modell-Hyperparameter bezeichnet.

Parameter, die das Training steuern (z. B. die Anzahl der Iterationen beim Training eines neuronalen Netzes), werden ebenfalls an den Algorithmus übergeben. Diese Parameter werden als Algorithmus-Hyperparameter bezeichnet.

Evaluierung des Modells

Das Modell wird unter Verwendung des Validierungsdatensatzes anhand der vereinbarten funktionalen Leistungsmetriken von ML (siehe 3.3) evaluiert, und die Ergebnisse werden genutzt, um das Modell in der Aktivität „Tuning des Modells“ zu verbessern. In der Praxis werden in der Regel mehrere Modelle erstellt und unter Verwendung verschiedener Algorithmen (z. B. Random Forests, SVM und neuronale Netze) sowie verschiedener Trainingsdatensätze trainiert, und die beste Kombination wird auf der Grundlage der Evaluierungsergebnisse ausgewählt.

Tuning des Modells

Die Ergebnisse der Evaluierung werden verwendet, um die Hyperparameter des Modells und die Hyperparameter des Algorithmus anzupassen. Das Modell wird anschließend mit diesen angepassten Einstellungen erneut trainiert, um seine funktionale Leistung zu verbessern.

Die drei Aktivitäten Training, Evaluierung und Tuning des Modells bilden zusammen die „Modellgenerierung“, wie in Abbildung 1 dargestellt.

© International Software Testing Qualifications Board

Test des Modells

Sobald durch die Aktivitäten der „Modellgenerierung“ ein akzeptables Modell generiert wurde, wird es anhand eines unabhängigen Testdatensatzes getestet, um zu überprüfen, ob die vereinbarten funktionalen Leistungskriterien von ML erfüllt sind. Die Testergebnisse werden zudem mit denen aus der Evaluierung verglichen. Wenn die Leistung des Modells mit unabhängigen Testdaten deutlich geringer ist als während der Evaluierung, kann es erforderlich sein, zu den Aktivitäten der „Modellgenerierung“ oder sogar zur Aktivität „Vorbereitung und Test der Daten“ zurückzukehren, um ein neues Modell zu trainieren.

Neben den funktionalen Leistungstests können auch nicht-funktionale Tests durchgeführt werden, beispielsweise zur Ermittlung der Zeit, die für die Erstellung einer Vorhersage benötigt wird. In der Regel werden die Tests in diesem Bereich von Dateningenieuren oder -wissenschaftlern durchgeführt; jedoch können auch Tester mit ausreichenden Fachkenntnissen und Zugriff auf die entsprechenden Ressourcen diese Tests durchführen.

Bereitstellung des Modells

Sobald die Aktivität „Modellgenerierung und -test“ abgeschlossen ist, wird das optimierte Modell in der Regel zusammen mit seiner Daten-Pipeline für die Bereitstellung überarbeitet. Dies erfolgt in der Regel über das ML-Entwicklungs-Framework. Zu den Zielplattformen können eingebettete Systeme und die Cloud gehören, wo über eine Web-API auf das Modell zugegriffen werden kann. Das überarbeitete, bereitgestellte Modell wird getestet, um sicherzustellen, dass es weiterhin die Akzeptanzkriterien erfüllt.

Einsatz des Modells

Nach der Bereitstellung wird das Modell in der Regel in ein größeres, KI-basiertes System im Produktivbetrieb integriert. Modelle können in festgelegten Zeitintervallen geplante Vorhersagen im Batch-Betrieb durchführen, oder auf Anfrage in Echtzeit ausgeführt werden.

Überwachung und Tuning des Modells

Während der Nutzung des Modells kann sich seine Situation mit der Zeit ändern, und es kann ein Drift des Modells weg von seiner beabsichtigten Leistung auftreten (siehe 6.1.7). Um sicherzustellen, dass etwaiger Drift erkannt und behandelt wird, ist das operative Modell regelmäßig anhand seiner Akzeptanzkriterien zu bewerten.

Es kann erforderlich sein, ein neues Modell zu erstellen, indem ein erneutes Training mit neuen Daten, neuen Hyperparametern oder beidem durchgeführt wird. Das neueste Modell kann dann unter Nutzung einer Form von A/B-Test (siehe 6.1.9) mit dem bestehenden Modell verglichen werden.

Der in Abbildung 1 dargestellte ML-Workflow stellt eine logische Abfolge dar; in der Praxis wird der Workflow jedoch iterativ angewendet, wobei Schritte wiederholt werden.

Die in Abbildung 1 dargestellten Aktivitäten umfassen nicht die Integration des ML-Modells in die Nicht-ML-Teile des Gesamtsystems. In der Regel können ML-Modelle nicht isoliert eingesetzt werden und müssen in Nicht-ML-Komponenten integriert werden (z. B. wird in Bildverarbeitungsanwendungen eine Daten-Pipeline verwendet, um Daten zu bereinigen und zu modifizieren, bevor sie an das ML-Modell übermittelt werden). Ist das Modell Teil eines großen Systems, muss es vor der Bereitstellung in dieses System integriert werden. In diesem Fall können Integrations-, System- und Abnahmetests durchgeführt werden.

© International Software Testing Qualifications Board

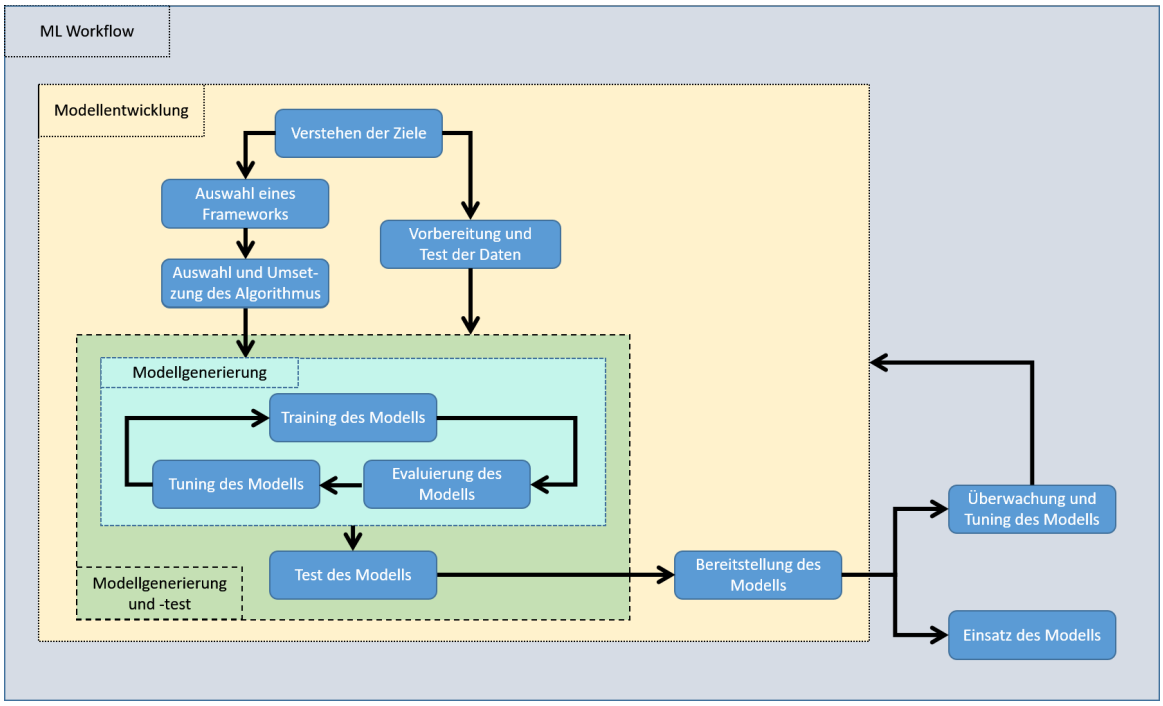


Abbildung 1: ML-Workflow

3.1.3 Praktische Übung zum Ziel HO-3.1.3 (H2): … selbstständig ein ML-Modell erstellen

Wählen Sie ein Modell für die Klassifikation aus, trainieren Sie es und testen Sie es mithilfe von überwachtem Lernen.

Erklären Sie den Unterschied zwischen Evaluierung, Tuning und Test des Modells, indem Sie die mit Validierungsdatensätzen und Testdatensätzen erzielte Genauigkeit vergleichen.

3.1.4 Vortrainierte Modelle, Fein-Tuning und Retrieval-augmentierte Generierung

Das Training eines neuen KI-Modells von Grund auf ist sowohl kostspielig als auch zeitaufwendig. Um diesem Problem zu begegnen, ist das Fein-Tuning eine gängige Lösung, bei der ein vortrainiertes neuronales Netz an eine neue, andere Aufgabe angepasst wird. Einer der Hauptvorteile besteht darin, dass im Vergleich zur Entwicklung eines Modells von Grund auf deutlich weniger Trainingsdaten (und Trainingsaufwand) erforderlich sind.

Das vortrainierte Modell wird durch zusätzliches Training mit Daten, die für die neue Aufgabe spezifisch sind, durch Fein-Tuning angepasst. Das Fein-Tuning kann auf das gesamte neuronale Netz, nur auf bestimmte Schichten (typischerweise nahe dem Ausgabebereich des neuronalen Netzes) oder auf zusätzliche Schichten angewendet werden. Nach dem Training wird die funktionale Leistung von ML des Modells bewertet. Basierend auf diesen Ergebnissen kann ein weiteres Fein-Tuning erfolgen, bis das Modell die erforderlichen Akzeptanzkriterien erfüllt.

Der Erfolg des Fein-Tunings hängt von der Ähnlichkeit zwischen der ursprünglichen und der neuen Aufgabe ab. Hier können bereits kleine Unterschiede zu einem sehr effektiven Fein-Tuning führen, z. B. dürfte ein Fein-Tuning zur Anpassung eines Bild-Klassifikators für Katzenrassen zur Erkennung von Hunderassen gut funktionieren. Die Anpassung an sprachliche Akzente ist hingegen aufgrund des größeren Unterschieds weniger effektiv. Ebenso erfordert das Fein-Tuning eines Large Language-Modells (LLMs) für die von ISTQB® definierte Grenzwertanalyse nur eine geringfügige Änderung am LLM und ist mit guten Trainingsdaten problemlos realisierbar.

© International Software Testing Qualifications Board

Eine Alternative zum Fein-Tuning eines LLMs ist die Retrieval-augmentierte Generierung (RAG), bei der dem LLM Datenquellen bereitgestellt werden, die speziell auf die jeweilige Anforderung zugeschnitten sind. Diese Datenquellen werden in ein durchsuchbares Format umgewandelt, sodass sie mit dem Kontext des Prompts verglichen werden können. Sobald relevante Dokumente identifiziert wurden, werden diese in einen erweiterten Prompt integriert, der an das LLM weitergeleitet wird. Da dem LLM nun relevantere Informationen zur Verfügung stehen, wird seine entsprechende Antwort voraussichtlich eine höhere Präzision aufweisen. Bei RAG werden keine Änderungen am vortrainierten Modell vorgenommen.

Ein vortrainiertes Modell kann RAG, Fein-Tuning oder beides zusammen nutzen, um die Leistung zu verbessern.

In der Regel werden etwaige Verzerrungen oder Sicherheitslücken des vortrainierten Modells auf das neue Modell übertragen, sodass Tests erforderlich sind, um sicherzustellen, dass es bei der neuen Aufgabe zuverlässig und fair arbeitet.

3.2 Daten für maschinelles Lernen

Die Datenvorbereitung gilt als eine der wichtigsten und ressourcenintensivsten Aktivitäten im ML-Workflow. Unterscheiden sich die Produktivdaten erheblich von den Trainingsdaten, gelten möglicherweise die Annahmen zur funktionalen Leistung von ML und zur Sicherheit (*safety*) nicht mehr. Die Datenvorbereitung nimmt in der Regel einen deutlich größeren Anteil des Gesamtaufwands ein als andere Aktivitäten, wie beispielsweise die Modellauswahl und die Modellerstellung. Die Datenvorbereitung ist untrennbar mit der Daten-Pipeline verbunden, welche die Rohdaten verarbeitet und in ein für das Training und die Vorhersage durch ML-Modelle nutzbares Format umwandelt.

3.2.1 Aktivitäten im Rahmen der Datenvorbereitung

Die Datenvorbereitung trägt dazu bei, die Qualität und Eignung der Daten für das Training des Modells sicherzustellen. Sie umfasst mehrere zentrale Aktivitäten:

- Datenbeschaffung:
  - Identifizierung relevanter Datentypen (z. B. numerisch, kategorial, Bild, Text).
  - Erfassung von Daten aus verschiedenen Quellen, wie z. B. Datenbanken, APIs oder Echtzeit-Sensoren.
  - Kennzeichnung von Daten für das überwachte Lernen sowie Überprüfung der Fehlerfreiheit und Konsistenz.

  Die erfassten Daten können verschiedene Formen annehmen (z. B. numerische, kategoriale, Bild, tabellarische, Text, Zeitreihe, sensorische, räumliche/geographische, Video und Audio).
- Datenvorverarbeitung:
  - Bereinigung der Daten, einschließlich:
    - Entfernung von Fehlern, Duplikaten und Ausreißern zur Überprüfung der Fehlerfreiheit und Konsistenz der Daten.
    - Ergänzung (*imputing*) fehlender Werte mithilfe von Verfahren wie Mittelwert, Median oder Modus (auch Modalwert genannt), um die Vollständigkeit der Daten zu gewährleisten.
    - Anonymisierung oder Entfernung personenbezogener Daten zum Schutz der Privatsphäre und zur Einhaltung gesetzlicher Vorschriften.
  - Umwandlung von Datenformaten, Skalierung und Normalisierung zur Gewährleistung der Konsistenz.
  - Erweiterung (*augmentation*) der Daten zur Vergrößerung der Stichprobengröße, Einbeziehung gegnerischer Beispiele zur Verbesserung der Robustheit gegenüber gegnerischen Angriffen sowie Generierung synthetischer Daten.
  - Stichprobenziehung von Teilmengen, um Trainingszeiten und Rechenaufwand zu reduzieren.

© International Software Testing Qualifications Board

- Merkmalsermittlung (*feature engineering*):
  - Auswahl relevanter Merkmale anhand ihres Beitrags zur Leistung des ML-Modells.
  - Extraktion einer Teilmenge informativer und nicht-redundanter Merkmale aus den vorhandenen Merkmalen, um Trainingszeiten und Rechenaufwand zu reduzieren.

Parallel zu diesen Maßnahmen der Datenvorbereitung wird in der Regel auch eine explorative Datenanalyse (EDA) durchgeführt, um Einblicke in die Daten zu gewinnen. Dazu gehören:

- das Aufdecken von Trends, Mustern und Anomalien in den Daten,
- die Datenvisualisierung mithilfe von Diagrammen und Grafiken zum besseren Verständnis.

Die Datenvorbereitung für Trainingsdaten ist in der Regel ein iterativer Prozess, der oft manuell durchgeführt wird. Einzelne Schritte der Datenvorbereitung können je nach den spezifischen Projektanforderungen neu geordnet oder weggelassen werden. Die Produktivdaten sollten den Eigenschaften der Trainingsdaten entsprechen (z. B. Datenverteilungen und Merkmalsbereiche), damit das Modell im Produktivbetrieb wie erwartet funktioniert. Die Schritte der Datenvorbereitung selbst können jedoch im Hinblick auf Effizienz und Skalierbarkeit im Produktivbetrieb angepasst werden.

3.2.2 Praktische Übung zum Ziel HO-3.2.2 (H2): … selbstständig die Datenvorbereitung durchführen, um die Erstellung eines ML-Modells zu unterstützen.

Führen Sie für eine gegebene Datenmenge die dafür geeigneten Aktivitäten im Rahmen der Datenvorbereitung gemäß Abschnitt 3.2.1 durch und erstellen Sie einen Datensatz zur Generierung eines Klassifikationsmodells mittels überwachtem Lernen.

Diese Aktivität dient als erster Schritt zur Erstellung eines ML-Modells, das in weiteren Übungen verwendet wird.

Zur Durchführung dieser Aufgabe erhalten die Lernenden geeignete (und sprachspezifische) Materialien, darunter:

- Bibliotheken
- ML-Entwicklungs-Framework
- Werkzeuge

3.2.3 Trainings-, Validierungs- und Testdatensätze

Logisch betrachtet werden zur Entwicklung eines ML-Modells drei Sätze gleichwertiger Daten benötigt (z. B. zufällig aus einem einzigen repräsentativen Datensatz ausgewählt):

- Ein Trainingsdatensatz wird zum Trainieren des Modells verwendet.
- Ein Validierungsdatensatz dient der Evaluierung und dem anschließenden Tuning des Modells.
- Ein Testdatensatz, auch als Hold-out-Datensatz bezeichnet, wird für den Test des getunten Modells verwendet.

Wenn eine hohe Anzahl geeigneter Daten verfügbar ist, hängt die Menge der im ML-Workflow für Training, Evaluierung und Test verwendeten Daten in der Regel von den folgenden Faktoren ab:

- Der erwarteten Komplexität des Modells.
- Dem zum Trainieren des Modells verwendeten Algorithmus.
- Der Verfügbarkeit von Ressourcen wie Hauptspeicher, Speicherplatz, Rechenleistung, Netzwerkbandbreite und der zur Verfügung stehenden Zeit.
- Dem gewünschten Vertrauen in das resultierende Modell.

Bei begrenzten Datenmengen kann eine Dreiteilung (Training, Validierung und Test) dazu führen, dass nicht genügend Daten für ein effektives Modelltraining zur Verfügung stehen, wodurch das Risiko einer Unteranpassung steigt. Um dem entgegenzuwirken, besteht eine gängige Strategie darin, nach Möglichkeit

© International Software Testing Qualifications Board

einen kleinen, abschließenden Testdatensatz abzutrennen. Die verbleibenden Daten (der kombinierte Trainings- und Validierungspool) werden dann für sog. Resampling-Verfahren verwendet, wie z. B. die im Folgenden beschriebene *k*-fache Kreuzvalidierung (wobei *k* eine vom Benutzer festgelegte ganze Zahl ist, üblicherweise 5 oder 10).

Bei der *k*-fachen Kreuzvalidierung werden die Daten in *k* gleichgroße Teilmengen (sogenannte Folds oder Segmente) aufgeteilt. Für jedes Segment wird das Modell auf den anderen *k*-1 Segmenten trainiert und mit den Daten im zurückbehaltenen Segment validiert. Dieser Vorgang wird *k*-mal wiederholt, so dass jedes Segment einmal als Validierungsdatensatz dient. Dies ermöglicht ein robustes Hyperparameter-Tuning und eine Einschätzung der funktionalen Leistung von ML. Die Daten werden in der Regel zufällig den Segmenten zugeordnet, häufig unter Verwendung einer geschichteten Stichprobenziehung, um jedes Segment repräsentativ zu machen, insbesondere bei unausgewogenen Daten oder kleinen Datensätzen.

Die funktionalen Leistungsmetriken von ML (z. B. Genauigkeit, F1-Wert – siehe 3.3.1) aus der Validierung jedes Segments werden anschließend gemittelt, um eine zuverlässigere Schätzung der Generalisierungsfähigkeit des Modells zu erhalten. Nach der Ermittlung der optimalen Hyperparameter durch Kreuzvalidierung wird ein endgültiges Modell in der Regel unter Verwendung dieser Hyperparameter auf dem gesamten Trainings- und Validierungspool (alle Daten mit Ausnahme des zurückbehaltenen Testdatensatzes) trainiert. Dieses endgültige Modell wird anschließend einmalig mit dem Testdatensatz getestet, um eine abschließende, unverzerrte Evaluierung zu erhalten. Ist ein Testdatensatz aufgrund extremer Datenknappheit nicht realisierbar, ist die als Mittelwert aus den Kreuzvalidierungen berechnete funktionale Leistung von ML optimistisch verzerrt und kann nicht als unverzerrte Schätzung dienen.

Weitere Resampling-Methoden für begrenzte Datenmengen sind die „Leave-One-Out"-Validierung (ein Sonderfall der *k*-fachen Kreuzvalidierung, bei welcher *k* der Anzahl der Stichproben entspricht) und das Bootstrap-Verfahren.

3.3 Funktionale Leistungsmetriken von ML zur Klassifikation

Bei Klassifikationsaufgaben (siehe 3.1.1) kann eine Konfusionsmatrix verwendet werden, um die Vorhersagen eines Modells zu bewerten. Dabei werden die Vorhersagen als richtig positiv (RP), richtig negativ (RN), falsch positiv (FP) oder falsch negativ (FN) klassifiziert. Aus der Konfusionsmatrix werden wichtige Metriken wie Genauigkeit, Präzision, Sensitivität und F1-Wert abgeleitet. Diese Metriken messen die Qualität der Klassifikation und verdeutlichen die Stärken und Schwächen eines ML-Modells. In diesem Abschnitt werden die Berechnung und Interpretation dieser Metriken erläutert, um die funktionale Leistung eines ML-Modells zu bewerten.

3.3.1 Berechnung der funktionalen Leistungsmetriken von ML

Bei einem Klassifikationsproblem sagt ein Modell die Ergebnisse nur selten immer korrekt voraus. Dies liegt unter anderem an der probabilistischen Natur von ML-Modellen und am Rauschen in den Daten. Für ein solches Problem kann eine Konfusionsmatrix mit den folgenden Angaben erstellt werden.

© International Software Testing Qualifications Board

		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 zwar unterschiedlich dargestellt werden kann (z. B. mit getauschten Einträgen für „Vorhergesagt“ und „Tatsächlich“), jedoch stets Werte für die vier möglichen Fälle liefert, nämlich richtig positiv (RP), richtig negativ (RN), falsch positiv (FP) und falsch negativ (FN).

Auf Basis der Konfusionsmatrix werden folgende Metriken definiert:

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

  Genauigkeit (*accuracy*) misst den Prozentsatz aller korrekten Klassifikationen.
- Präzision = RP / (RP + FP) * 100 %

  Präzision (*precision*) misst das Verhältnis der korrekt vorhergesagten positiven Ergebnisse zur Gesamtmenge aller positiv vorhergesagten Ergebnisse. Sie gibt an, wie sicher man bei positiven Vorhersagen sein kann.
- Sensitivität = RP / (RP + FN) * 100 %

  Sensitivität (*recall*) misst den Anteil der tatsächlich positiven Fälle, die korrekt vorhergesagt wurden. Diese Metrik ist ein Maß dafür, wie sicher man sein kann, keine positiven Fälle zu übersehen.
- F1-Wert = 2 * (Präzision * Sensitivität) / (Präzision + Sensitivität)

  Der F1-Wert (*F1-Score*) ist das harmonische Mittel von Präzision und Sensitivität. Die Werte liegen zwischen 0 und 100. Ein Wert nahe 100 bedeutet, dass das Modell sowohl eine hohe Präzision als auch eine hohe Sensitivität erreicht, was darauf hindeutet, dass Klassifizierungsfehler (falsch positive und falsch negative Ergebnisse) nur minimale Auswirkungen haben. Dagegen deutet ein niedriger F1-Wert darauf hin, dass das Modell Schwierigkeiten hat, positive Fälle korrekt zu identifizieren, entweder, weil es diese nicht erkennt, oder aber viele Fehlalarme produziert.

3.3.2 Praktische Übung zum Ziel HO-3.3.2 (H2): … ein ML-Modell anhand ausgewählter funktionaler Leistungsmetriken von ML bewerten.

Verwenden Sie das in der vorherigen Übung trainierte Klassifikationsmodell und berechnen sowie visualisieren Sie die Ergebnisse für Genauigkeit, Präzision, Sensitivität und F1-Wert. Nutzen Sie dabei, sofern verfügbar, die Bibliotheksfunktionen Ihres ML-Entwicklungs-Frameworks.

© International Software Testing Qualifications Board

3.3.3 Praktische Übung zum Ziel HO-3.3.3 (H2): … die Auswirkungen verschiedener ML-Modelle und Datensatzkombinationen aufzeigen.

Nach der vorherigen Übung verwenden Sie nun verschiedene ML-Modelle und Datensatzkombinationen, um deren Auswirkungen auf das Training des ML-Modells sowie auf dessen finales Verhalten zu untersuchen. Beobachten Sie dabei die Trainingszeiten und die funktionalen Leistungsmetriken des ML-Modells.

3.4 Neuronale Netze

Künstliche neuronale Netze wurden ursprünglich entwickelt, um die Funktionsweise des menschlichen Gehirns nachzubilden, das als Netz miteinander verbundener biologischer Neuronen betrachtet werden kann.

Das Single-Layer-Perzeptron ist eines der frühesten Beispiele für die Implementierung eines künstlichen neuronalen Netzes. Es besteht aus nur einer Schicht und kann für das überwachte Lernen binärer Klassifikatoren für linear separierbare Probleme verwendet werden, die bestimmen, ob eine Eingabe zu einer bestimmten Klasse gehört oder nicht. Beispielsweise kann ein Perzeptron bei E-Mails unterscheiden, ob es sich um Spam handelt oder nicht, indem es lernt, die Merkmale der beiden Kategorien durch eine lineare Trennlinie im Eingaberaum zu unterteilen.

Die meisten heute eingesetzten neuronalen Netze gelten als tiefe neuronale Netze, da sie aus mehreren Schichten bestehen. Vollständig verbundene Netze können als Mehrschicht-Perzeptrons betrachtet werden (siehe Abbildung 3).

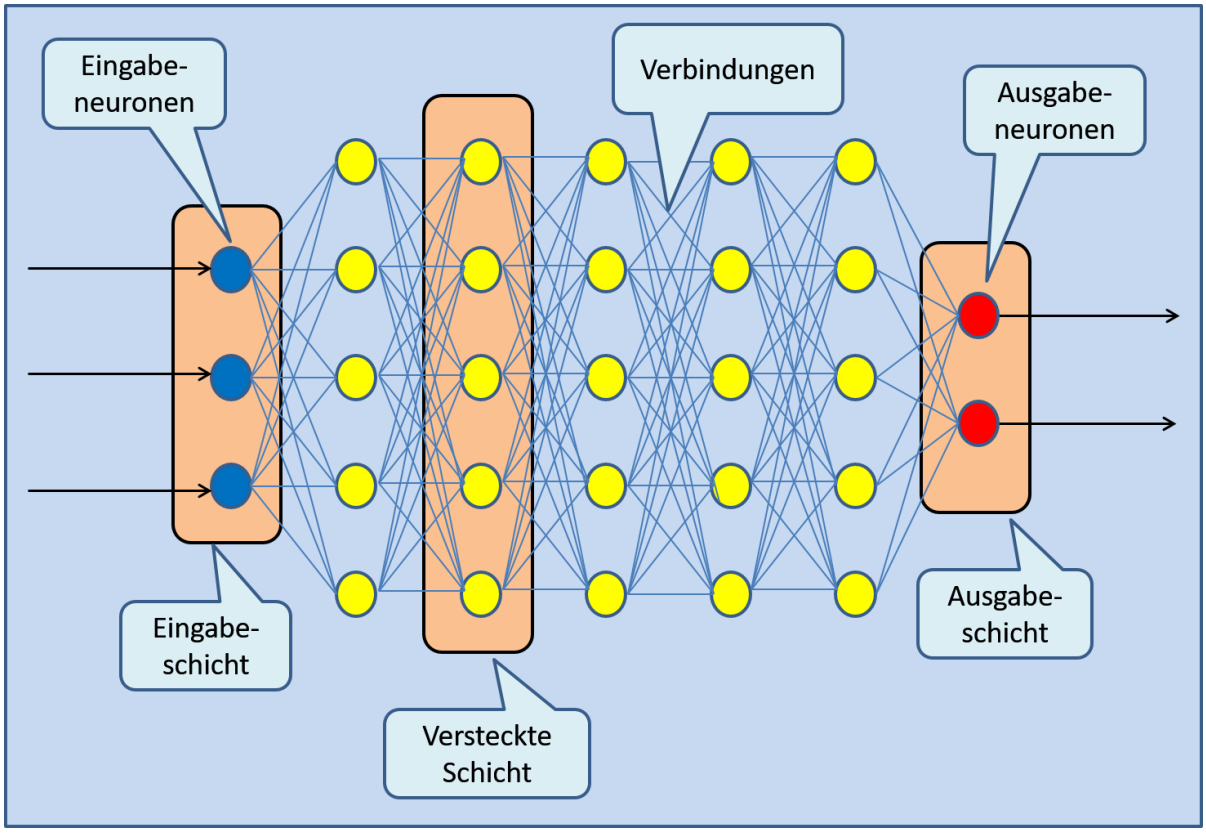


Abbildung 3: Struktur eines tiefen neuronalen Netzes

© International Software Testing Qualifications Board

3.4.1 Struktur und Funktionsweise eines tiefen neuronalen Netzes

Ein tiefes neuronales Netz besteht typischerweise aus drei Haupttypen von Schichten. Die Eingabeschicht empfängt Eingaben, beispielsweise Pixelwerte einer Kamera. Die Ausgabeschicht liefert Ergebnisse an die Außenwelt, beispielsweise die Wahrscheinlichkeit, dass ein Eingabebild eine Katze zeigt. Zwischen Eingabe- und Ausgabeschicht befinden sich versteckte Schichten, die aus künstlichen Neuronen beziehungsweise Knoten bestehen. In vielen Architekturen sind die Neuronen einer Schicht mit allen Neuronen der nachfolgenden Schicht verbunden. Jede Verbindung hat dabei ein eigenes Verbindungsgewicht.

Die Neuronen führen Berechnungen durch und übertragen durch das Netz Informationen von den Eingabe- zu den Ausgabeneuronen. Dabei werden die Eingabedaten schrittweise in immer abstraktere Repräsentationen transformiert.

Die von einem Neuron durchgeführte Berechnung erzeugt einen sogenannten Aktivierungswert. Dieser wird berechnet, indem zunächst die mit den Verbindungsgewichten gewichtete Summe der Aktivierungswerte aller verbundenen Neuronen der vorherigen Schicht gebildet wird. Zu dieser Summe wird dann der individuelle Verschiebungswert[1] (*bias*) des Neurons addiert. Das Ergebnis wird anschließend durch eine nicht-lineare Aktivierungsfunktion verarbeitet. Die Verwendung verschiedener Aktivierungsfunktionen ergibt unterschiedliche Aktivierungswerte.

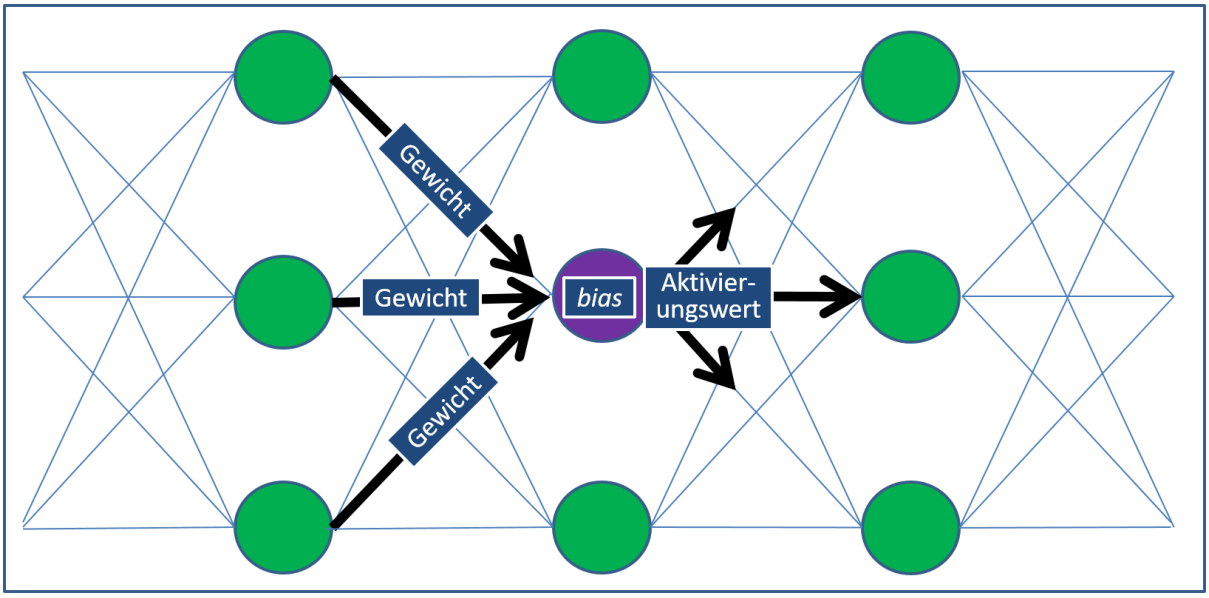


Abbildung 4: Von jedem Neuron durchgeführte Berechnungen

Zu Beginn des Trainings werden die Verbindungsgewichte und der Verschiebungswert jedes Neurons üblicherweise mit kleinen Zufallswerten initialisiert (Verschiebungswerte manchmal zu 0). Die Trainingsdaten durchlaufen dann das Netz, wobei jedes Neuron seine Aktivierungsfunktion ausführt, um letztlich eine Ausgabe zu generieren. Die generierte Ausgabe wird anschließend mit dem bekannten korrekten Ergebnis verglichen. Der daraus resultierende Fehler (oder Verlust), der die Abweichung quantifiziert, wird durch das Netz zurückgeführt, um die Gewichte und Verschiebungswerte anzupassen und den Fehler zu minimieren. Wenn mehr Trainingsdaten das Netz durchlaufen (jeder Durchlauf eines Trainingsdatensatzes wird als Epoche bezeichnet), werden bei jeder Epoche die Gewichte und Verschiebungswerte schrittweise angepasst, sodass das Netz lernt. Nach einiger Zeit wird das erzielte Ergebnis idealerweise als gut genug angesehen, um das Training zu beenden.

[1] Der hier verwendete Begriff “Verschiebungswert (*bias*)” ist nicht mit der in Abschnitt 5.1.2 behandelten Verzerrung (*bias*) zu verwechseln.

© International Software Testing Qualifications Board

3.4.2 Praktische Übung zum Ziel HO-3.4.2 (H1): … die Implementierung eines Perzeptrons nachvollziehen

Die Lernenden werden bei einer Übung angeleitet, die zeigt, wie ein Perzeptron eine einfache Funktion erlernt, beispielsweise die UND-Funktion.

Die Übung soll verdeutlichen, wie ein Perzeptron seine Gewichte und Verschiebungswerte über mehrere Epochen hinweg anpasst, bis der Fehler auf null reduziert wird. Für diese Aktivität können unterschiedliche Hilfsmittel wie Tabellenkalkulationen oder Simulationen eingesetzt werden.

3.4.3 Überdeckungsmaße für neuronale Netze

Strukturelle Überdeckungsmetriken für neuronale Netze wurden entwickelt, um zu bewerten, wie umfassend Testeingaben die internen Mechanismen eines Modells anregen. Da neuronale Netze keinen explizit programmierten Kontrollflüssen folgen, sondern ihr Verhalten durch gelernte Gewichte, Verschiebungswerte und Aktivierungen bestimmt wird, sind spezielle Überdeckungsmaße erforderlich. Diese bewerten, in welchem Umfang verschiedene Bereiche des neuronalen Netzes unter Testbedingungen aktiviert wurden. Typische Ansätze sind [COV_REF]:

- Neuronenüberdeckung: Misst den Anteil der Neuronen im Netz, deren Ausgabe während des Testens einen festgelegten Schwellenwert überschreitet.
- k-Multisektions-Neuronenüberdeckung (*k-Multisection Neuron Coverage*, kMNC): Der mögliche Ausgabebereich jedes Neurons wird in k Abschnitte unterteilt. Die Metrik misst den Anteil dieser Abschnitte, die während des Testens aktiviert werden.
- Neuronen-Grenzwertüberdeckung (*Neuron Boundary Coverage*, NBC): Misst den Anteil der Neuronen, deren Ausgabe während des Testens entweder den im Training erreichten Maximalwert überschreitet oder unter den im Training erreichten Minimalwert fällt.

Diese Überdeckungsmetriken können das KI-Testen unterstützen, indem sie Bereiche eines Modells aufzeigen, die bislang nicht getestet wurden und daher potenziell Defekte oder nicht erlernte Verhaltensweisen verbergen könnten. Wenn beispielsweise bestimmte Neuronen oder Schichten während des Testens niemals aktiviert werden, kann die Leistungsfähigkeit des Modells in den zugehörigen Entscheidungsgrenzen fraglich sein. Solche Erkenntnisse helfen Testern dabei, zusätzliche Testeingaben oder Testbedingungen zu entwerfen, um wenig untersuchte Teile des Netzes zu testen. Derzeit stehen nur wenige kommerzielle Werkzeuge zur Verfügung, die diese spezifischen Überdeckungsmaße unterstützen.

Strukturelle Überdeckung allein garantiert jedoch nicht, dass ein neuronales Netz gut generalisiert oder mit unterschiedlichen Variationen aus der realen Welt umgehen kann. Neuronale Netze können Scheinkorrelationen erlernen und dadurch korrekte Aktivierungen aus falschen Gründen erzeugen. Daher sollten Tester zusätzlich andere Testverfahren einsetzen, wie z. B. gegnerischer Test und metamorpher Test, wie in den Kapiteln 5, 6 und 7 beschrieben.

© International Software Testing Qualifications Board

4 Testen KI-basierter Systeme – 195 Minuten

Schlüsselbegriffe
Angriff, explorativer Test, risikobasierter Test, Testorakel

Spezifische Schlüsselbegriffe für KI
adaptives KI-basiertes System, fixiertes KI-basiertes System, generative KI, KI-basiertes System, Large Language-Modell (LLM)

Lernziele und Ziele der praktischen Übungen für Kapitel 4: Die Lernenden können …

4.1 Einführung in das Testen KI-basierter Systeme

AI-4.1.1 (K2) … die Testbarkeit fixierter und adaptiver KI-basierter Systeme vergleichen.

AI-4.1.2 (K2) … erklären, warum beim Testen KI-basierter Systeme häufig ein statistischer Ansatz erforderlich ist.

AI-4.1.3 (K2) … die Herausforderungen und Lösungen im Zusammenhang mit Testorakeln für KI-basierte Systeme erklären.

4.2 Testen von generativer KI und Large Language-Modellen

AI-4.2.1 (K2) … erklären, wie generative KI getestet werden kann.

AI-4.2.2 (K3) … Red-Teaming für GenAI-Systeme umsetzen.

HO-4.2.3 (H2) … selbstständig explorative Tests eines LLMs durchführen.

4.3 Teststufen und ML-Systeme

AI-4.3.1 (K2) … die Teststufen bei der Entwicklung von ML-Systemen zusammenfassen.

AI-4.3.2 (K2) … erklären, wie der risikobasierte Test bei ML-Systemen angewendet wird.

© International Software Testing Qualifications Board

4.1 Einführung in das Testen KI-basierter Systeme

Im Vergleich zum Testen von konventioneller Software bringt das Testen KI-basierter Systeme besondere Herausforderungen mit sich. KI-basierte Systeme lassen sich grob in fixierte und adaptive Systeme unterteilen. Fixierte KI-basierte Systeme (*locked AI-based systems*; Synonym: statische KI-basierte Systeme), die nach der Inbetriebnahme ein festgelegtes Verhalten aufweisen, sind aufgrund ihrer weitgehend deterministischen Natur einfacher zu testen. Dagegen bringen adaptive Systeme, die sich nach der Inbetriebnahme weiterentwickeln und lernen, eine erhöhte Komplexität mit sich, da sich ihr Verhalten unvorhersehbar ändern kann. Darüber hinaus macht der probabilistische Charakter vieler KI-Modelle häufig einen statistischen Testansatz erforderlich, da deterministische Methoden möglicherweise nicht ausreichen, um Ausgaben zu bewerten, die durch Daten und Wahrscheinlichkeiten beeinflusst werden. Dies erfordert die Bewertung von Leistungsverteilungen und Konfidenzniveaus über verschiedene Szenarien hinweg.

Eine zentrale Herausforderung beim Testen ist das sogenannte Testorakelproblem, also die Überprüfung, ob die tatsächliche Ausgabe des Testobjekts für eine gegebene Eingabe korrekt ist. Bei konventioneller Software machen klar definierte Spezifikationen es relativ einfach, die erwarteten Ergebnisse zu spezifizieren und somit die Korrektheit zu überprüfen. Bei KI-basierten Systemen, insbesondere solchen, die komplexe oder subjektive Aufgaben bewältigen, kann die eindeutige Definition der erwarteten Ergebnisse hingegen schwierig oder sogar unmöglich sein. Diese Unsicherheit verstärkt sich bei Aufgaben, die menschliche Fähigkeiten übersteigen, Unschärfe beinhalten oder bei denen keine objektive Grundwahrheit (*ground truth*) existiert. In solchen Fällen ist die Implementierung automatisierter Testorakel schwierig, da manchmal Expertenwissen oder statistisches Schließen erforderlich sind.

Zusätzlich verschärfen vage oder unvollständige Systemanforderungen das Testorakelproblem beim Testen KI-basierter Systeme. Mögliche Lösungsansätze umfassen den Einsatz statistischer Auswertungen, die Konsultation von Fachexperten oder die Definition einer Grundwahrheit, anhand derer die Ausgaben bewertet werden. Diese Ansätze zielen darauf ab, zuverlässige Erwartungen zu etablieren und KI-basierte Systeme wirksam zu evaluieren.

4.1.1 Fixierte und adaptive KI-basierte Systeme

Viele der heute im Einsatz befindlichen KI-basierten Systeme sind fixierte KI-basierte Systeme, die ihr Verhalten nach der Inbetriebnahme nicht verändern. Ein Beispiel für ein fixiertes KI-basiertes System ist ein in Betrieb genommenes ML-Modell, das auf einem DNN basiert, und dessen Gewichte und Verschiebungswerte in der Entwicklung festgelegt und nur durch ein erneutes Training des DNN änderbar sind. Sicherheitsrelevante Technologie für selbstfahrende Fahrzeuge nutzt für bestimmte Aufgaben, wie die Spurerkennung oder die Erkennung von Verkehrszeichen, häufig fixierte KI-basierte Systeme.

Im Gegensatz dazu kann ein adaptives KI-basiertes System, wie z. B. ein System mit bestärkendem Lernen, sein Verhalten auch nach der Inbetriebnahme anpassen. Die Änderungen können auf einer Belohnungsfunktion basieren oder dazu dienen, sich an eine neue Produktivumgebung anzupassen. Die genauen Details solcher Änderungen lassen sich jedoch nicht im Voraus vorhersagen. So könnte eine E-Commerce-Plattform adaptive KI einsetzen, um Nutzern Produkte auf Basis ihres bisherigen Verhaltens und ihrer sich verändernden Präferenzen zu empfehlen.

Viele GenAI-Systeme, wie LLM-basierte Chatbots, werden zur Laufzeit als fixierte Modelle eingesetzt, aber regelmäßig aktualisiert. Damit nehmen sie eine Position zwischen vollständig fixierten und vollständig adaptiven Systemen ein.

In der Praxis erstrecken sich KI-basierte Systeme auf einem Kontinuum von vollständig deterministischen fixierten Systemen, die für eine gegebene Eingabe dieselbe Ausgabe liefern, bis hin zu absichtlich nicht-deterministischen, selbstlernenden Systemen, die ihr Verhalten anpassen und weiterentwickeln.

Fixierte KI-basierte Systeme sind deutlich einfacher zu testen als adaptive, da sie weitgehend deterministisch sind und sich die erwarteten Ergebnisse daher nicht ändern. Ein aktualisiertes fixiertes KI-basiertes System gilt in der Regel als neues System, das einen neuen Testlauf erfordert. Auch wenn fixierte

© International Software Testing Qualifications Board

KI-basierte Systeme im Allgemeinen als deterministisch gelten, können große neuronale Netze in der Praxis aufgrund von Faktoren wie der Genauigkeit von Gleitkommazahlen und Unterschieden in der Hardware-Ausführung nicht-deterministisches Verhalten zeigen. Dies gilt insbesondere beim Einsatz von Parallelverarbeitung oder GPUs.

Ein adaptives KI-basiertes System kann vor der Inbetriebnahme strengen Tests unterzogen werden (beispielsweise durch die Simulation von Änderungen der Umgebungsbedingungen, das Testen des Lernmechanismus selbst oder das Testen der Fähigkeit des Systems, sich in kontrollierten Szenarien angemessen anzupassen). Allerdings ist dieses Testen komplexer als bei einem fixierten KI-basierten System, da ein adaptives System sein Verhalten während des Testens oder als Folge des Testens ändern kann.

Da neue Verhaltensweisen eines adaptiven KI-basierten Systems nicht immer vorhersehbar sind, können solche unvorhersehbaren Verhaltensweisen weder im Voraus getestet noch Testfälle dafür vorbereitet werden. Eine automatisierte Testsuite, die sich auf die Kernfunktionalität des Systems konzentriert, kann erstellt und bei jeder wesentlichen Änderung des Systems ausgeführt werden, um zu überprüfen, ob die Anpassungen sicher sind.

Testen kann auch dazu eingesetzt werden, zu überprüfen, ob die Leistung des Systems einen bestimmten Schwellenwert nicht unterschreitet. Dieses Testen kann als Reaktion auf wesentliche Systemänderungen oder im Rahmen einer laufenden Überwachung erfolgen.

4.1.2 Begründung für einen statistischen Ansatz beim Testen KI-basierter Systeme

Das Testen KI-basierter Systeme bringt aufgrund ihrer datengetriebenen und probabilistischen Natur besondere Herausforderungen mit sich. Ein statistischer Ansatz ist beim Testen KI-basierter Systeme aus mehreren Gründen erforderlich:

- **Nicht-Determinismus:** Viele KI-basierte Systeme sind grundlegend probabilistisch und zeigen daher häufig nicht-deterministisches Verhalten. Das bedeutet, dass dieselben Eingaben nicht immer zu derselben Ausgabe führen müssen. Dies kann auf stochastische Elemente in ihrer Architektur oder auf die Implementierung probabilistischer Verknüpfungen zurückzuführen sein, die aus Trainingsdaten gelernt wurden. Daher kann ein einzelner Test, beispielsweise ein Fall, in dem ein Modell eine Katze fälschlicherweise als Hund klassifiziert, keine allgemeingültige Aussage zur funktionalen KI-Korrektheit des Modells widerspiegeln. Um sicherzustellen, dass die Testergebnisse diese Unsicherheit zuverlässig überwinden, muss die Testsuite groß genug sein, um statistisch signifikante Ergebnisse zu liefern.
- **Evaluierung der verteilungsspezifischen Leistung**: KI-Modelle werden auf bestimmten Verteilungen der Daten trainiert, die nicht exakt mit ihrer Produktivumgebung übereinstimmen. Um zu beurteilen, wie ein Modell unter realen Bedingungen funktionieren wird, muss eine statistisch signifikante Stichprobe von Szenarien mit relevanten Verteilungen der Daten aus dem Produktivbetrieb getestet werden. Dies stellt sicher, dass die Tests die Variabilität der Daten im Produktivbetrieb und das daraus resultierende Modellverhalten erfassen.
- **Umgang mit Unsicherheit und Verzerrung:** KI-basierte Systeme sind anfällig für Verzerrungen von Daten (*data biases*) und können plausible, aber fehlerhafte Vorhersagen mit hoher Konfidenz liefern. Statistische Tests ermöglichen es, die Genauigkeit, Fairness und KI-Robustheit des Modells anhand von Leistungsmetriken wie Konfidenzintervallen, Hypothesentests und Fehleranalysen zu quantifizieren und zu analysieren.
- **Regulatorischer und sicherheitskritischer Kontext**: In regulierten Branchen (z. B. Gesundheitswesen, Transport) ist es häufig erforderlich nachzuweisen, dass ein KI-basiertes System Sicherheits- oder Fairness-Schwellenwerte mit hoher Konfidenz einhält. Statistische Methoden unterstützen Aussagen über die Zuverlässigkeit eines Modells über ein breites Spektrum von Szenarien hinweg und nicht nur für einzelne Beispiele.

Ein statistischer Ansatz für das Testen probabilistischer ML-Systeme wird in Abschnitt 6.1.3 beschrieben.

© International Software Testing Qualifications Board

4.1.3 Testorakel für KI-basierte Systeme

Das Testen KI-basierter Systeme kann eine Herausforderung sein, vor allem wenn es darum geht, die erwarteten Ergebnisse zu bestimmen (das Testorakelproblem). Diese Schwierigkeiten entstehen durch verschiedene Faktoren, die KI-basierten Systemen inhärent sind:

- **Probabilistische und nicht-deterministische Natur:** Die Ausgaben von KI-Systemen können selbst bei identischen Eingaben variieren. Zwar gibt es bei vielen Systemen (z. B. beim überwachten Lernen) einen einzigen korrekten Zielwert, aber die Ausgaben des Modells sind probabilistisch und das Definieren eines strengen „Bestanden“ / „Fehlgeschlagen“-Orakels erfordert daher häufig die Festlegung von Schwellenwerten oder Toleranzbereichen.
- **Explorative Entwicklung und unvollständige Spezifikationen**: KI-Entwicklung ist häufig explorativ. Systemanforderungen können sich weiterentwickeln, unvollständig oder schlicht nicht vorhanden sein. Es fehlen dann die detaillierten Spezifikationen, die für die Generierung präziser erwarteter Ergebnisse notwendig wären.
- **Komplexität der Aufgaben**: KI-basierte Systeme bewältigen häufig Aufgaben, die für eine unmittelbare Überprüfung durch den Menschen zu komplex sind. Eine manuelle Prüfung der erwarteten Ausgaben ist dann nicht praktikabel.
- **Subjektivität des Verhaltens:** Die Korrektheit des Verhaltens eines KI-basierten Systems kann subjektiv sein. So können beispielsweise die Erwartungen der Nutzer an virtuelle Assistenten stark variieren, was die Festlegung allgemein anerkannter erwarteter Ergebnisse erschwert.
- **Selbstlernende Systeme:** Diese KI-basierten Systeme aktualisieren ihre internen Modelle kontinuierlich auf Basis neuer Daten, die nach der Inbetriebnahme anfallen. Dadurch verändert sich das „korrekte" Verhalten des KI-basierten Systems im Laufe der Zeit, sodass die Systemantworten auch dann wirksam und angemessen bleiben, wenn sich die Definition von „korrekt" weiterentwickelt. Ein anfänglich festgelegter Satz erwarteter Ergebnisse kann dadurch schnell ungültig werden.

Das Testorakelproblem bei KI-basierten Systemen lässt sich auf verschiedene Weisen angehen:

- **Definition von Ausgabegrenzen:** Tester können vereinbarte akzeptable Bereiche, Verteilungen, festgelegte Grenzwerte und Toleranzen verwenden, etwa die maximale Distanz, innerhalb derer ein autonomes Fahrzeug stehen bleiben soll.
- **Definition von Umgebungsgrenzen:** Tester müssen Werte für die Bedingungen der Testumgebung festlegen (z. B. Lichtverhältnisse, Temperatur, Netzwerklatenz), um sicherzustellen, dass die Ausgaben besser vorhersehbar und reproduzierbar sind.
- **Konsultation von Experten**: Fachexperten können dabei helfen, erwartete Ergebnisse zu definieren, auch wenn ihre Einschätzungen voneinander abweichen oder fehlbar sein können.
- **Spezialisierte Testmethoden**: A/B-Tests, Back-to-Back-Tests und metamorphes Testen können KI-basierte Systeme bewerten, indem sie Verhaltensweisen vergleichen oder Eigenschaften verifizieren, oft ohne für jeden Fall explizite erwartete Ergebnisse zu benötigen.
- **Stellvertreter-Orakel** (*proxy oracle*)**:** Sekundäre Systeme oder Modelle (einschließlich anderer KI-Systeme) können eingesetzt werden, um Ausgaben zu bewerten oder zu validieren, wenn direkte erwartete Ergebnisse nicht verfügbar sind. Beispielsweise kann ein Stellvertreter-Modell auf gekennzeichneten Daten trainiert werden, um Vorhersagen für nicht gekennzeichnete Tests zu treffen.

4.2 Testen von generativer KI und Large Language-Modellen

Dieser Abschnitt beschreibt konkrete Methoden zur Validierung von GenAI-Systemen und behandelt die Black-Box-Evaluierung, das Red-Teaming und eine praktische Übung zum explorativen Test. Dabei werden Herausforderungen hervorgehoben, die sich aus vielfältigen Eingaben, justierbaren Parametern und Kontextfenstern ergeben. Außerdem werden sowohl funktionale als auch nicht-funktionale Qualitätsmaße mithilfe von Benchmarks und gezielten Übungen behandelt. Weitere Details zur Verwendung von GenAI-Systemen beim Testen finden sich im CT-GenAI-Lehrplan ([CT-GenAI]).

© International Software Testing Qualifications Board

4.2.1 Testen generativer KI

Beim Testen generativer KI geht es darum, die Korrektheit, Kohärenz und Kreativität der erzeugten Ausgaben zu bewerten, einschließlich der Originalität und Neuartigkeit der generierten Ergebnisse. Auch wird die Erfüllung der festgelegten funktionalen und nicht-funktionalen Anforderungen überprüft. Da GenAI-Systeme Texte, Bilder, Videos und Audio erzeugen können, müssen Teststrategien entsprechend angepasst werden, um die Qualitätsmerkmale solcher Inhalte zu bewerten.

Ein verbreiteter Ansatz ist der Black-Box-Test, bei dem Tester verschiedene Eingaben — Prompts, Bilder, Teile von Daten — in das System einspeisen und die Testergebnisse bewerten (s. a. Red-Teaming in Abschnitt 4.2.2). Dabei werden Faktoren wie Klarheit, Originalität und die Einhaltung domänenspezifischer Regeln berücksichtigt. Dieser Ansatz, sei es manuell oder automatisiert, ist besonders nützlich für Endbenutzeranwendungen wie Chatbots oder Entwurfswerkzeuge, bei denen die Nutzerzufriedenheit von der Nützlichkeit und Plausibilität der generierten Inhalte abhängt.

Eine wesentliche Herausforderung beim Testen eines GenAI-Modells ist das sogenannte „Problem der Eingabe-Explosion“: Eingaben können äußerst vielfältig und schwer kontrollierbar sein. So gibt es beispielsweise optionale System-Prompts und einen Benutzer-Prompt, welche enorme Mengen unterschiedlicher Daten enthalten können, und es kann auch über eine API auf GenAI zugegriffen werden. Auch beeinflussen mehrere Parameter wie Temperatur und maximale Token-Anzahl die Ausgabe. Zusätzlich wirkt sich das Kontextfenster, das frühere Teile einer Konversation speichert, auf die generierte Ausgabe aus.

In vielen Situationen kann die Bewertung der Ausgabe eine manuelle Überprüfung einbeziehen, jedoch hängt unabhängig davon die Feststellung, ob ein Test bestanden oder fehlgeschlagen ist, von qualitativen Evaluierungskriterien ab, die in den Anforderungen definiert sind. Alternativ kann häufig ein zweites GenAI-System eingesetzt werden, um das Testergebnis automatisch zu bestimmen. So kann beispielsweise die Korrektheit eines von einem GenAI-System generierten Bildes durch ein Bilderkennungssystem überprüft werden. Solche Ansätze sollten jedoch mit Bedacht eingesetzt werden, da sie ähnliche Verzerrungen oder Fehler reproduzieren können.

Nicht-funktionales Testen kann für GenAI-Systeme ebenso wichtig sein wie funktionales Testen. Dazu gehört die Evaluierung der Ressourcennutzung sowohl während der Inferenz als auch während des Trainings, um einen effizienten Betrieb und Kosteneffektivität sicherzustellen. Zu den Metriken zählen CPU- und GPU-Auslastung, Speicherverbrauch, Netzwerkbandbreite und Antwortzeiten.

Benchmark-Suiten bieten standardisierte Evaluierungs-Frameworks zur Bewertung der Fähigkeiten von GenAI. Diese kuratierten Datensätze und zugehörigen Aufgaben ermöglichen einen konsistenten Vergleich verschiedener Modelle. Sie messen verschiedene Aspekte, von Sprachverständnis über Schlussfolgerungsvermögen bis hin zu Programmierfähigkeiten. Durch die Messung von GenAI anhand dieser Benchmarks lassen sich Verbesserungspotenziale auf systematische und reproduzierbare Weise identifizieren.

4.2.2 Red-Teaming für GenAI-Systeme

Red-Teaming (RT) ist eine systematische „Black-Box“-Form des Fehlerangriffs, mit der versucht wird, schädliches Verhalten in einem KI-basierten System und vor allem in seinen Ausgaben aufzudecken. In Anlehnung an historische Praktiken wie militärische Kriegsspiele und die „Tiger Teams“ der NASA beinhaltet KI-RT den „Angriff“ auf ein KI-basiertes System mit dem Ziel, das System absichtlich dazu zu bringen, schädliche oder unerwünschte Ergebnisse zu produzieren, wie beispielsweise Datenschutzverletzungen, die Äußerung rassistischer Ansichten oder die Bereitstellung von Anleitungen zur Durchführung chemischer, biologischer, radiologischer oder nuklearer (CBRN) Angriffe. Sobald solche Schwachstellen identifiziert sind, wird auf deren Basis das System aktualisiert und gestärkt, sodass es in Zukunft weniger anfällig für solche Ergebnisse ist. Während RT auf jedes KI-basierte System angewendet werden kann, ist es für GenAI besonders wichtig, da die enorme Bandbreite möglicher Ein- und Ausgaben einen umfangreichen „Angriffsraum“ schafft. RT wird in der Regel auf das gesamte KI-basierte System Ende zu Ende angewendet, kann aber auch nur auf das KI-Modell angewendet werden.

© International Software Testing Qualifications Board

Viele Unternehmen konzentrieren RT auf Sicherheitslücken und Schwachstellen; es kann jedoch auch genutzt werden, um schädliche Fähigkeiten in anderen Bereichen zu erkennen, wie beispielsweise bei der Zuverlässigkeit, dem Datenschutz, der Fairness, der Verzerrung und der Erzeugung von Falschinformationen. Für bestimmte Arten KI-basierter Systeme wird RT zunehmend regulatorisch gefordert, wie dies in Rahmenwerken, beispielsweise der KI-Verordnung der EU [EU AI Act], zu sehen ist. Als adaptiver, dynamischer Evaluierungsansatz, bei dem Prompts als Reaktion auf Ausgaben sofort aktualisiert werden können, ergänzt RT statische Ansätze wie Benchmarking, indem Systeme unter extremen und unerwarteten Bedingungen getestet werden.

Wird RT für Evaluierungen der Sicherheit (*security*) eingesetzt, testet man das System, um Sicherheitslücken gegenüber externen Angriffen zu identifizieren, darunter sowohl nicht-KI-spezifische als auch KI-spezifische Faktoren der Sicherheit (*security*) wie z. B. indirekte promptübergreifende Einschleusungsangriffe und das Verstecken bösartiger Inhalte in Dokumenten, die von RAG verwendet werden. Während sich Sicherheits-RT (*security*-RT) in der Regel auf böswillige Eingaben konzentriert, besteht das Ziel bei Sicherheits- (*safety*) und anderen Evaluierungen oft darin, festzustellen, wie das System unter normalen Nutzungsbedingungen schädliche Ausgaben erzeugen könnte, beispielsweise durch die Generierung unsicherer medizinischer Ratschläge, ohne dass seitens des Benutzers eine gegnerische Absicht vorliegt.

RT ist am effektivsten, wenn es vor der Bereitstellung eines Systems, aber nach Abschluss erster interner Qualitätsbewertungen durchgeführt wird. Die Kernaufgabe ist häufig die interaktive Eingabe von Prompts, bei der Mitglieder des Red-Teams Dialoge in mehreren Runden (z. B. 15–20) führen, um fehlerhaftes oder gegen Richtlinien verstoßendes Verhalten hervorzurufen. Der Ansatz umfasst typischerweise Folgendes:

1. Zusammenstellung eines vielfältigen Teams von Testern, um ein breites Spektrum an Perspektiven und Angriffsvektoren abzudecken.
2. Gewährung des Zugriffs auf das KI-basierte System in einer sicheren Testumgebung.
3. Prompting des Systems zur Identifikation von Sicherheitslücken, durch ergebnisoffene Exploration oder unter Verwendung von Checklisten.
4. Analyse der identifizierten Fehlerwirkungen, um Bedrohungen zu verstehen.
5. Erstellung von Datensätzen aus diesen Bedrohungen, um Abwehrmaßnahmen und Systemverbesserungen zu unterstützen.

Um eine umfassende Überdeckung durch RT zu erreichen, setzen Unternehmen neben kleinen Expertenteams verschiedene Strategien ein. Dazu gehören manuelle Ansätze wie die Crowdsourcing-basierte Generierung von Prompts sowie automatisierte Ansätze, bei denen beispielsweise ein LLM zur Generierung zahlreicher Prompts für Angriffe verwendet wird und die Ergebnisse anschließend von einem anderen LLM überprüft werden. Hybride Ansätze kombinieren die Kreativität menschlicher Tester mit der Skalierbarkeit der Automatisierung.

RT, das proaktiv und auf die Phase vor der Bereitstellung ausgerichtet ist, ergänzt das Blue-Teaming, bei dem es um die fortlaufende, in Echtzeit erfolgende defensive Überwachung und Filterung von Eingaben in ein KI-basiertes System im Produktivbetrieb geht, um dieses vor Angriffen zu schützen. Erkenntnisse aus dem RT können genutzt werden, um die im Blue-Teaming eingesetzten Überwachungsmaßnahmen und Filter zu verbessern.

4.2.3 Praktische Übung zum Ziel HO-4.2.3 (H2): … selbstständig explorative Tests eines LLMs durchführen

Die Lernenden führen explorative Tests an einem LLM durch, das eine Grenzwertanalyse durchführt. Sie erhalten ein Testsitzungsblatt (*session sheet*) für den explorativen Test der Fähigkeit des LLMs, Testfälle mithilfe einer 2-Wert- und 3-Wert-Grenzwertanalyse zu generieren. Im Rahmen einer Nachbesprechung überprüfen sie die Richtigkeit und Vollständigkeit der Ausgaben.

© International Software Testing Qualifications Board

4.3 Teststufen und ML-Systeme

MLS erfordern spezielle Teststufen, um den besonderen Risiken des maschinellen Lernens Rechnung zu tragen. Dies sind der Eingabedatentest und der ML-Modelltest (s. Abschnitt 5.1 und Kapitel 6). Darüber hinaus sind herkömmliche Teststufen weiterhin relevant, darunter Komponententest, Komponentenintegrationstest, Systemtest und Abnahmetest.

4.3.1 Teststufen für ML-Systeme

Die Nicht-KI-Komponenten eines ML-Systems können mithilfe herkömmlicher Teststufen getestet werden. Darüber hinaus erfordern ML-Modelle und ML-Systeme weitere Tests, um die spezifischen Risiken im Zusammenhang mit ML zu berücksichtigen.

Zur Bewältigung ML-spezifischer Risiken werden zwei spezielle Teststufen verwendet:

- **Eingabedatentest** (siehe Kapitel 5)**:** Bezieht sich auf die Trainingsdaten, die für das Training des ML-Modells genutzt werden und die Produktivdaten, die vom MLS genutzt werden, um die Vorhersagen für die Produktivumgebung zu erzeugen.
- **ML-Modelltest** (siehe Kapitel 6)**:** Behandelt das Testen von ML-Modellen, welche die endgültigen Ergebnisse des ML-Workflows sind (siehe 3.1.2).

Die auf ML bezogenen Risiken können nicht alle mit diesen zwei ML-spezifischen Teststufen adressiert werden. Die folgenden Teststufen werden typischerweise ebenfalls, abhängig von den wahrgenommenen Risiken, benötigt:

- **Komponententest:** Ist anwendbar für alle Nicht-KI-Komponenten wie Benutzungsschnittstellen, Daten-Pipelines und Kommunikationskomponenten.
- **Komponentenintegrationstest:** Beinhaltet das Testen, ob die Eingaben aus der Daten-Pipeline wie erwartet vom Modell empfangen werden und ob alle vom Modell erzeugten Vorhersagen mit den relevanten Systemkomponenten (z. B. der Benutzungsschnittstelle) ausgetauscht und korrekt verwendet werden. Wird KI-als-Dienst bereitgestellt (siehe 1.1.6), wird der API-Test des bereitgestellten Dienstes als Teil des Komponentenintegrationstests durchgeführt.
- **Systemtest:** Umfasst Fehlernachtests, um sicherzustellen, dass die funktionale Leistung von ML des Modells aus den anfänglichen ML-Modelltests nicht beeinträchtigt wird, wenn das Modell in ein komplettes System eingebettet wird. Das Testen ist besonders wichtig, wenn das ML-Modell absichtlich verändert wurde (beispielsweise durch die Kompression eines DNN zur Reduktion der Größe). Nicht-funktionale Tests werden ebenfalls abgedeckt, beispielsweise das Testen der Performanz bezogen auf die Zeit, die das vollständige KI-System benötigt, um eine Vorhersage zu liefern.
- **Systemintegrationstest:** Konzentriert sich auf die Überprüfung der Schnittstellen und des Austauschs von Daten zwischen dem KI-System und externen Systemen oder Diensten unter Nutzung einer repräsentativen Umgebung für die Produktivbedingungen.
- **Abnahmetest:** Wird KI-als-Dienst genutzt, kann der Abnahmetest erforderlich sein, um festzustellen, ob der Dienst für den vorgesehenen Zweck geeignet ist und ob die funktionalen Leistungskriterien von ML erfüllt wurden.

4.3.2 Risikobasierter Test für ML-Systeme

Risikobasiertes Testen sollte für alle Systeme angewendet werden, unabhängig davon, ob sie KI-Komponenten beinhalten oder nicht. Die meisten regulatorischen Rahmenwerke (ob publiziert oder in Entwicklung) verlangen einen risikobasierten Ansatz für die Entwicklung und das Management von KI-basierten Systemen. Da KI-basierte Systeme besondere Risiken beinhalten, weicht deren Test von dem nicht-KI-basierter Systeme ab.

Es gibt keinen standardisierten Weg zur Kategorisierung von ML-Risiken, allerdings werden ML-spezifische Risiken sowohl mit der Entwicklung des MLS (Projektrisiken) als auch mit dem MLS selbst (Produktrisiken)

© International Software Testing Qualifications Board

assoziiert. Eine Möglichkeit der Kategorisierung dieser Risiken ist, den Workflow von ML zu nutzen und in drei Hauptbereiche zu unterteilen:

- **Entwicklung:** Umfasst den ML-Algorithmus, die Entwicklung des Modells und das ML-Entwicklungs-Framework. Beispiele für Projektrisiken beinhalten sub-optimale Auswahl von Algorithmen, mangelhafte Auswahl des Evaluierungsansatzes und Sicherheitslücken des Entwicklungs-Frameworks. Siehe Kapitel 7 für weitere Details.
- **Eingabedaten:** Umfasst die Bereitstellung der Trainingsdaten zur Unterstützung des ML und die Bereitstellung der Produktivdaten, die das Modell unter Betriebsbedingungen nutzt. Beispiele für Produktrisiken beinhalten die Verzerrung der Trainingsdaten, Daten-Pipeline-Fehler und nicht-repräsentative Trainingsdaten. Siehe Kapitel 5 für weitere Details.
- **Modell:** Umfasst das generierte ML-Modell. Beispiele für Produktrisiken sind das Nicht-Erreichen der geforderten funktionalen Leistungskriterien von ML, ein überangepasstes Modell und die Anfälligkeit für gegnerische Beispiele. Siehe Kapitel 6 für weitere Details.

Eine andere Möglichkeit zur Kategorisierung der mit KI verbundenen Risiken ist die Nutzung der in ISO/IEC 25059 definierten Qualitätsmerkmale.

Es gibt unterschiedliche Testarten und -verfahren, die diese Risiken adressieren und die speziell für das Testen von MLS entworfen wurden, beispielsweise Daten-Pipelinetest, gegnerischer Test und Review der Angemessenheit von Algorithmus/Modell. Der Lehrplan beschreibt diese in den Kapiteln 5, 6 und 7.

© International Software Testing Qualifications Board

5 Eingabedatentest für ML-Systeme – 180 Minuten

Schlüsselbegriffe

Daten-Pipelinetest, Datenrepräsentativitätstest, Datensatzeinschränkungstest, Eingabedatentest, Kennzeichen-Korrektheitstest, Review, Testen auf Verzerrung

Spezifische Schlüsselbegriffe für KI

Analyse auf mittelbare Diskriminierung, Multiple Annotation

Lernziele und Ziele der praktischen Übungen für Kapitel 5: Die Lernenden können …

5.1 Eingabedatentest für ML-Systeme

AI-5.1.1 (K2) … Beispiele für Testansätze nennen, die zur Risikominderung von Eingabedaten für ML-Systeme Verwendung finden.

AI-5.1.2 (K2) … das Testen auf Verzerrungen erklären.

AI-5.1.3 (K2) … die verschiedenen Arten von Daten-Pipelinetests zusammenfassen.

AI-5.1.4 (K2) … den Datenrepräsentativitätstest erklären.

AI-5.1.5 (K3) … den Datensatzeinschränkungstest anwenden.

AI-5.1.6 (K2) … den Kennzeichen-Korrektheitstest erklären.

HO-5.1.7 (H2) … selbstständig den Eingabedatentest auf ML-Datensätzen durchführen.

© International Software Testing Qualifications Board

5.1 Eingabedatentest für ML-Systeme

Der Eingabedatentest soll sicherstellen, dass die von ML-Systemen verwendeten Daten für Training, Test und Vorhersage von ausreichender Qualität sind (siehe 3.2). Dazu gehören Reviews, statistische Verfahren (z. B. das Testen auf Verzerrungen), die explorative Datenanalyse (EDA) der Trainingsdaten sowie statische und dynamische Tests der Daten-Pipeline.

5.1.1 Risiken bei Eingabedaten und ihre Minderung

Die folgende Tabelle enthält Beispiele für Risiken im Zusammenhang mit Eingabedaten und entsprechende Tests, die zur Risikominderung eingesetzt werden könnten:

Potenzielles Risiko	Mögliche Maßnahme zur Risikominderung
Fehlerhafte Trainingsdaten führen zu verzerrten Modellen. Probleme in Algorithmus, Modell oder ML-Entwicklungs-Framework, die zu systembedingter Unfairness führen.	Testen auf Verzerrungen – siehe 5.1.2.
Trainingsdaten stammen aus einer nicht vertrauenswürdigen Quelle. Daten für Training, Test und Validierung sind mangelhaft verwaltet.	Datenprovenienztest
Trainingsdaten sind verunreinigt.	A/B-Test – siehe 6.1.9 Datenprovenienztest EDA – siehe 3.2.1 Angriffe als Bestandteil von Red-Teaming – siehe 4.2.2
Datensatz ist in sich inkonsistent. Daten außerhalb des zulässigen Bereichs. Falsche Datentypen.	Datensatzeinschränkungstest – siehe 5.1.5.
Suboptimale Merkmalsauswahl.	Merkmalstest (*feature testing*)
Unausgewogener Datensatz aufgrund mangelhafter Überdeckung aller Ziel-Klassen. Durch Datenerweiterung verzerrter Datensatz. Fehlende Daten. Trainingsdaten decken nur einen Teil der gesamten Anwendungsfälle ab. Wertebereiche werden im Datensatz nicht vollständig abgedeckt.	Datenrepräsentativitätstest – siehe 5.1.4.
Mangelhafte Anleitung zur Datenkennzeichnung. Mehrdeutige Daten. Ungenaue oder inkonsistente Kennzeichnungen.	Kennzeichen-Korrektheitstest – siehe 5.1.6.

© International Software Testing Qualifications Board

Potenzielles Risiko	Mögliche Maßnahme zur Risikominderung
Fehlerhafte Daten-Pipeline durch mangelhaften Entwurf oder Integration. Mangelhafte Datenqualität beeinträchtigt die Ausgaben der Daten-Pipeline. Leistungseinbußen beim Einsatz der Daten-Pipeline. Verletzungen der Sicherheit (*security*) oder unkontrollierte Änderungen beeinflussen die Daten-Pipeline.	Daten-Pipelinetest – siehe 5.1.3.

5.1.2 Testen auf Verzerrung

Als Verzerrungen in einem MLS werden nicht-zufällige, unfaire unterschiedliche Behandlungen aufgrund sensibler Attribute bezeichnet, wie etwa Geschlecht, Alter oder ethnische Zugehörigkeit. Solche Unterschiede sind häufig rechtswidrig und führen dazu, dass das System diskriminierend wirkt.

Um Verzerrungen in ML-Systemen zu testen, muss man deren mögliche Ursachen verstehen. Diese sind in erster Linie:

- Fehler in den Trainingsdaten, wie beispielsweise mangelnde Repräsentativität, historische Verzerrungen oder absichtliche Datenverunreinigungen (die zu Datenverzerrungen führen).
- Mängel im Algorithmus, im Modell oder im ML-Entwicklungs-Framework, die zu systembedingter Unfairness führen, wie beispielsweise ein Algorithmus, der in einem System zur Kreditbewilligung eine Entscheidungsschwelle für Bonitätsbewertungen verwendet (algorithmische Verzerrung).

Testansätze zum Erkennen von Verzerrungen beinhalten:

- Reviews des gesamten ML-Workflows sowie insbesondere der Datenvorbereitung, um deren Potenzial zur Einführung von Verzerrungen festzustellen.
- Reviews der Datensatzdokumentation, um potenzielle Ursachen von unfairer Behandlung zu identifizieren und zu mindern, indem die Beschaffung und Annotation der Daten untersucht wird sowie welche Bevölkerungsgruppen sie repräsentieren.
- Statische Analyse der Programme zur Datenvorbereitung sowie des Modellimplementierungscodes, um Anti-Patterns oder den unsachgemäßen Umgang mit sensiblen Attributen zu identifizieren.
- Explorative Datenanalyse (EDA) der Trainingsdaten unter Verwendung von Visualisierungs- und Clusterbildungsmethoden, um unausgewogene Daten, verzerrte Verteilungen oder ungewöhnliche Gruppierungen über verschiedene sensible Attribute hinweg aufzudecken.
- Dynamisches Testen, um das Erkennen von Verzerrungen in einem ML-Modell zu erleichtern, indem ein bekannter, unverzerrter und repräsentativer Datensatz in das System eingegeben wird. Dessen Vorhersagen werden auf statistisch signifikante Unterschiede in den Ergebnissen zwischen verschiedenen sensiblen Gruppen analysiert. Auf diese Weise lassen sich Verzerrungen in den Modellausgaben identifizieren, unabhängig davon, ob diese Verzerrungen aufgrund der Daten während des Trainings oder während der Modellentwicklung entstanden sind.
- Kennzeichen-Korrektheitstest (siehe 5.1.6), um eine fehlerhafte Datenkennzeichnung zu identifizieren, die zum Erlernen falscher Zusammenhänge zwischen spezifischen sensiblen Attributen und Ausgaben führt.
- Analyse auf mittelbare Diskriminierung (*disparate impact analysis*):
  1. Identifikation der sensiblen Attribute für das Modell.
  2. Erzeugung von kontrafaktischen Beispielen für die sensiblen Attribute, zum Beispiel Szenarien, in denen in einem Kreditantrag das Geschlecht von männlich zu weiblich geändert wird.

© International Software Testing Qualifications Board

3. Erzeugung von Modellausgaben durch Verwendung der kontrafaktischen Beispiele als Eingaben in das Modell.
4. Analyse der Ergebnisse mehrerer Tests, um ein statistisch signifikantes Ergebnis zu erzielen. Damit wird festgestellt, ob die Änderung eines sensiblen Attributes zu einer Änderung der Modellausgabe führt, was auf das Vorliegen von Verzerrungen hinweist.

Hinweis: Die Analyse auf mittelbare Diskriminierung kann auch auf Kombinationen sensibler Attribute angewendet werden. Dadurch lassen sich verdeckte Verzerrungen identifizieren, die mit bestimmten Merkmalskombinationen verbunden sind. Dabei ist sorgfältig zu prüfen, dass die kontrafaktischen Beispiele nicht unrealistisch sind, da sonst das Modell möglicherweise eher auf die Realitätsferne der Beispiele reagiert anstatt auf eine zugrundeliegende Verzerrung.

5.1.3 Daten-Pipelinetest

Ein effektives Testen der Daten-Pipeline ist nicht nur für die Bestätigung der Zuverlässigkeit und Leistungsfähigkeit datengetriebener Systeme von entscheidender Bedeutung, sondern auch für die Aufrechterhaltung einer hohen Datenqualität während des gesamten ML-Workflows (siehe 3.1.2).

Reviews in der Entwurfsphase der Daten-Pipeline stehen am Anfang des hier angewendeten mehrstufigen Ansatzes.

Komponententests umfassen dabei Tests von Komponenten zur Datenaufnahme, Transformationsskripten und Sensorschnittstellen. Dabei kommen Code-Reviews, statische Analysen und hardwarespezifische Tests zum Einsatz, um eine zuverlässige Datenerfassung zu verifizieren. Komponententests validieren die Logik von Datentransformationen, überprüfen die Umsetzung von Datenvalidierungsregeln, bestätigen eine robuste Fehlerbehandlung und testen auf Sicherheitslücken, die ausgenutzt werden könnten, um Schadprogramme einzuschleusen oder Daten zu verunreinigen.

Komponentenintegrationstests verifizieren den nahtlosen Datenfluss über interne Schnittstellen hinweg sowie die korrekte Interpretation der Daten während ihres Durchlaufs durch die Pipeline. Dabei werden Fehler erkannt, die durch nicht kompatible Schnittstellen oder falsche Annahmen zwischen den Komponenten entstehen.

Systemtests prüfen die vollständig zusammengesetzte Pipeline, beginnend mit dem Smoke-Testen zur Bestätigung der grundlegenden Funktionalität der Pipeline. Funktionale Tests prüfen die Übereinstimmung der Pipeline mit den festgelegten Anforderungen, einschließlich Datentransformationen und korrekter Weiterleitung der Daten. Nicht-funktionale Tests evaluieren die Leistung der Daten-Pipeline unter Last, ihre Skalierbarkeit zur Bewältigung steigender Datenmengen sowie die Sicherheitsmaßnahmen zum Schutz der Datenintegrität. Tests durch Fehlereinfügungen ermitteln die Robustheit der Pipeline. Dabei wird durch simulierte fehlerhafte Dateneingaben bewertet, inwieweit das System in der Lage ist, die Datenintegrität auch bei unerwarteten oder beschädigten Daten aufrechtzuerhalten. In Back-to-Back-Tests (siehe 6.1.10) wird die Pipeline für die Produktivdaten mit der Pipeline für die Trainingsdaten verglichen, um deren konsistente Funktionalität zu verifizieren.

Systemintegrationstests verifizieren das korrekte Zusammenspiel zwischen der Daten-Pipeline und externen Systemen oder Diensten, wie etwa Datenquellen, Speicherplattformen, Testmonitoren und nachgelagerten Empfängern wie beispielsweise ML-Modellen.

Tests in der Produktivumgebung werden auf dem operativen System durchgeführt. Durch Back-to-Back-Tests wird überprüft, ob die Leistung im Vergleich zu früheren Versionen gleichbleibend ist oder sich verbessert hat. A/B-Tests (siehe 6.1.9) ermöglichen den Vergleich neuer Pipeline-Versionen mit einer Referenz, wodurch Verbesserungen validiert und gleichzeitig sichergestellt wird, dass es zu keiner Verschlechterung im Datenfluss der Produktivumgebung kommt. Es können Werkzeuge integriert werden, um das Modellverhalten, die Leistung und potenzielle Fehler kontinuierlich in Echtzeit zu überwachen und zu beobachten.

Reviews des Konfigurationsmanagements helfen sicherzustellen, dass im Training, Test und in der Produktivumgebung die richtigen Versionen des Pipeline-Codes, die richtigen Konfigurationen und die korrekten Datensätze verwendet werden.

© International Software Testing Qualifications Board

Nicht zuletzt müssen die Teststrategien auf den Zweck der Pipeline abgestimmt sein. Trainingsdaten-Pipelines, bei denen es sich häufig um explorative Prototypen handelt, haben andere Prioritäten als robuste Produktivdaten-Pipelines für die. Der Test von Trainingsdaten-Pipelines kann den Schwerpunkt auf die Datenintegrität legen, während der Test von Produktivdaten-Pipelines die Zuverlässigkeit, Performanz und Wartbarkeit in den Vordergrund stellt.

5.1.4 Datenrepräsentativitätstest

Der Datenrepräsentativitätstest bestimmt, inwieweit die Eigenschaften von Datensätzen, die für das Training, die Validierung und das Testen von ML-Modellen verwendet werden, mit den realen Daten übereinstimmen, denen das ML-Modell voraussichtlich im Produktivbetrieb begegnen wird. Die Tests betrachten verschiedene Formen mangelhafter Repräsentativität, wie etwa verzerrte Datensätze, fehlende Daten, unausgewogene Verteilung von Merkmalen, unzureichende Überdeckung von Betriebsszenarien und unausgewogene Repräsentation von Klassen (siehe 5.1.1).

Diese Tests umfassen in der Regel die folgenden Schritte:

1. Definition der Grundgesamtheit:
   - Erfassen der vorgesehenen Anwendungsfälle und des Produktiv-Kontexts des ML-Systems.
   - Analysieren der Eigenschaften von Endnutzern und Produktivumgebungen.
   - Ermitteln der erwarteten Verteilungen der Daten in der Produktivumgebung sowie kritischer Sonderfälle durch:
     - Einbeziehen von Fachexperten, um reale Datenmuster zu verstehen.
     - Analysieren von Daten aus bestehenden Systemen oder vergleichbaren Anwendungen.
     - Verwenden von Benchmark-Datensätzen aus vertrauenswürdigen Quellen (z. B. NIST, Industrie-Datenbanken).
   - Anwendung einer geschichteten Stichprobenziehung auf die repräsentativen Referenzdaten, um eine Daten-Baseline zu erstellen, die alle relevanten Untergruppen der Zieldomäne abdeckt.
2. Analyse der Daten-Charakteristika:
   - Anwenden der EDA (s. Abschnitt 3.2.1), um:
     - Trainings-/Testdatensätze auf ihre Repräsentativität hin zu überprüfen.
     - den Referenzdatensatz zu untersuchen, der die erwarteten Produktivdaten repräsentiert.
     - durch Histogramme, Streudiagramme und andere grafische Verfahren die Verteilungen in Datensätzen zu visualisieren.
   - Untersuchen von Beziehungen zwischen Merkmalen und insbesondere deren Korrelationen, um erhaltenswerte Muster zu identifizieren.
   - Identifizieren potenzieller Anomalien, Lücken oder ungewöhnlicher Häufungen in den Daten.
3. Anwendung statistischer Bewertungsverfahren
   - Verwendung formaler statistischer Tests wie Chi-Quadrat und Kolmogorov-Smirnov zum Vergleich von Verteilungen [STATS].
   - Überprüfung auf unausgewogene Daten, insbesondere bei Klassifizierungsproblemen.
   - Überprüfung auf angemessene Überdeckung sowohl typischer Szenarien als auch Sonder- und Grenzfälle.

Datenrepräsentativitätstests sollten vor dem Modelltraining stattfinden, um zu verhindern, dass Modelle auf der Grundlage nicht repräsentativer Daten erstellt werden. Nach der Inbetriebnahme sollten die Eigenschaften der Eingabedaten in der Produktivumgebung kontinuierlich überwacht werden, um Veränderungen zu erkennen, die auf einen Datendrift gegenüber den ursprünglichen Verteilungen der Trainingsdaten hindeuten könnten (s. Abschnitt 6.1.7 zu Drift-Test).

© International Software Testing Qualifications Board

5.1.5 Datensatzeinschränkungstest

Der Datensatzeinschränkungstest verifiziert, ob die Daten in einem Datensatz vordefinierten Regeln oder Beschränkungen entsprechen. Ziel ist es, die Integrität und Konsistenz der Daten für das maschinelle Lernen sicherzustellen. So könnte beispielsweise eine Prüfung der Konsistenz der Werte in einem Datensatz durchgeführt werden.

Einschränkungen von Daten finden sich typischerweise in Datenbankschemata. Ein Datenbankschema definiert die Struktur, Typen und Beziehungen der Daten zueinander. In ähnlicher Weise lässt sich für ML-Datensätze eine Menge von Einschränkungen definieren. Diese dienen als logisches Modell der Daten des Datensatzes und sollten für korrekte Daten erfüllt sein.

Es gibt verschiedene Möglichkeiten, Einschränkungen für Datensätze zu kategorisieren. Eine Einschränkung kann auf einen einzelnen Wert eines Attributs angewendet werden, der in einer einzelnen Instanz vorkommt (Einzelwert-Einschränkung), wie zum Beispiel:

- Fehlend – testet auf fehlende Werte und fehlende Attribute.
- Bereich – testet, ob ein Wert in einem vorgegebenen Bereich liegt.
- Typ – testet, ob der Wert eines Attributs dem angegebenen Typ entspricht (z. B. für ein als Ganzzahl spezifiziertes Attribut ist der vorliegende Wert keine Zeichenkette oder reelle Zahl).

Alternativ kann eine Einschränkung für mehrere Werte gelten, wobei in der Regel die Werte eines einzelnen Attributs über mehrere Instanzen hinweg berücksichtigt werden (Multiwert-Einschränkung), wie zum Beispiel:

- Summe – testet, ob die Summe aller Werte einem bestimmten Wert entspricht, diesen übersteigt oder nicht übersteigt (z. B. darf die Gesamtpunktzahl der in einem Formel-1-Rennen vergebenen Punkte 102 nicht überschreiten, muss jedoch über 50,5 Punkte liegen).
- Anzahl – testet, ob die Anzahl aller nicht leeren Werte für Attribute oder Instanzen einem angegebenen Wert entspricht, diesen übersteigt oder nicht übersteigt.
- Duplikate – testet, ob in einem Datensatz identische oder nahezu identische Attributwerte oder Instanzen vorhanden sind, und legt eine maximal tolerierte Anzahl als Schwelle fest (häufig null).
- Nützlichkeit – testet, ob ein Attribut zumindest einige mehrfach vorkommende Werte enthält. Wenn jeder Wert nur einmal vorkommt (wie bei IDs oder Zeitstempeln), liefert das Attribut normalerweise keine nützlichen Muster, aus denen ein ML-Modell lernen kann.
- Ausreißer – identifiziert alle Werte, die als statistische Ausreißer angesehen werden könnten.

Eine besondere Form der Multiwert-Einschränkung ermöglicht den Vergleich der Werte verschiedener Attribute miteinander (Vergleichs-Einschränkung), wie zum Beispiel:

- Größer als – testet, ob ein Wert für ein Attribut größer ist als ein Wert für ein zweites Attribut (z. B. ob für ein Programm die Zahl der Codezeilen die Anzahl der Codezeilen mit Fehlern übersteigt).
- Korrelation – testet, ob die Werte eines Attributs mit den Werten eines zweiten Attributs zusammenhängen (z. B. haben alle Studierenden, deren Wert für das Attribut „Punkte“ mindestens 1,33 Standardabweichungen über dem Mittelwert liegt, auch den Wert „A“ für das Attribut „Note“).

Das Testen der Daten gegen die definierten Einschränkungen kann manuell erfolgen. Angesichts des Umfangs dieser Aufgabe und der in der Regel großen Datenmengen ist jedoch meistens deren Automatisierung im Rahmen der Daten-Pipeline erforderlich. Bei einer Implementierung als Teil der Pipeline kann das Werkzeug zur Durchführung der Datensatzeinschränkungstests direkt Berichte für Datenwissenschaftler (bei Anomalien in den Trainingsdaten) oder für das Betriebspersonal (bei Problemen mit den Produktivdaten) bereitstellen.

5.1.6 Kennzeichen-Korrektheitstest

Die Korrektheit der Datenkennzeichen ist beim überwachten Lernen von entscheidender Bedeutung. Ungenaue oder inkonsistente Kennzeichen beeinträchtigen unmittelbar die funktionale Leistung und die Generalisierungsfähigkeit von ML-Modellen.

© International Software Testing Qualifications Board

Zu den gängigen Ansätzen zur Überprüfung der Korrektheit von Daten-Kennzeichen gehören:

- **Experten-Review:** Fachexperten oder geschulte Annotatoren prüfen manuell eine Stichprobe der gekennzeichneten Daten. Sie bewerten die Fehlerfreiheit der Kennzeichnungen anhand ihres Fachwissens und von Richtlinien.
- **Multiple Annotation:** Datenpunkte werden von mehreren Annotatoren unabhängig voneinander gekennzeichnet und durch eine Form von Back-to-Back-Test bewertet (siehe 6.1.10). Diskrepanzen weisen auf Fehler hin, die untersucht und behoben werden müssen. Die Übereinstimmung zwischen den verschiedenen Annotatoren (Inter-Annotator Übereinstimmung, *inter-annotator agreement*, IAA) lässt sich anhand von Metriken wie dem Cohen-Kappa-Wert oder einer einfachen prozentualen Übereinstimmung messen [STATS]. Niedrige IAA-Werte können auf Mängel in den Kennzeichnungsrichtlinien, mehrdeutige Daten oder eine mangelhafte Kennzeichnung selbst hindeuten.
- **Risikobasierte Priorisierung für Reviews und Annotationen:** Sowohl Experten-Reviews als auch Multiple Annotationen können mithilfe eines risikobasierten Ansatzes fokussiert werden, um Stichproben für Reviews und Multiple Annotationen zu identifizieren. Die Priorisierung kann auf der Wahrscheinlichkeit einer Fehlkennzeichnung basieren, etwa bei mehrdeutigen Datenpunkten (z. B., wenn unklar ist, zu welcher Klasse sie gehören, oder wenn sie sich nahe an einer Grenze zwischen Klassen befinden), sowie auf den Daten, welche den Erfolg oder die Sicherheit (*safety*) einer Anwendung am ehesten beeinflussen.
- **Analyse von Verteilungen der Daten:** Liegen vergleichbare Datensätze vor, kann eine Gegenüberstellung der Verteilungen von Klassen des zu testenden Datensatzes mit denen ähnlicher Datensätze Anomalien und möglicherweise fehlerhafte Kennzeichnungen aufdecken.
- **Automatisierte, regelbasierte Tests:** Für bestimmte Aufgaben können automatisierte Tests auf der Grundlage vordefinierter Kennzeichnungs-Regeln oder -Einschränkungen implementiert werden. Beispielsweise kann überprüft werden, ob Begrenzungsrahmen (Rechtecke, die zur Lokalisierung von Objekten in Bildern dienen) sich nicht überlappen oder über die Bildränder hinausragen.
- **Modell-Verlustanalyse:** Datenpunkte, die während des Modelltrainings einen hohen Verlust aufzeigen, d. h., bei denen die Vorhersagen des Modells erheblich von ihren tatsächlichen Kennzeichen abweichen, können auf fehlerhafte Kennzeichnung hindeuten. Ein hoher Verlust spiegelt einen signifikanten Fehler wider, der darauf hindeutet, dass das Modell Schwierigkeiten hat, das zugewiesene Kennzeichen zu lernen, was auf einen möglichen Fehlerzustand hinweist.
- **Modell-Konfidenzwertanalyse:** Datenpunkte, die bei trainierten Modellen einen geringen Konfidenzwert in den Vorhersagen aufweisen, können falsch gekennzeichnet oder mehrdeutig sein oder aber außerhalb der Verteilung der Trainingsdaten des Modells liegen.

Oftmals ist eine Kombination verschiedener Ansätze am effektivsten. So sind beispielsweise Experten-Reviews wertvoll, um frühzeitig klare Richtlinien für die Annotation festzulegen. IAA-Metriken aus mehreren Annotationen können den Kennzeichnungsprozess weiter verfeinern. Im weiteren Verlauf können modellbasierte Ansätze zudem potenzielle Fehlerzustände in den Kennzeichen identifizieren, während das Modell weiterentwickelt wird.

5.1.7 Praktische Übung zum Ziel HO-5.1.7 (H2): … selbstständig den Eingabedatentest auf ML-Datensätzen durchführen

Führen Sie für einen bestimmten Datensatz (z. B. strukturierte, tabellarische Daten) einen Test der Eingabedaten etwa auf fehlende Werte, Duplikate und Ausreißer durch.

© International Software Testing Qualifications Board

6 Modelltest für ML-Systeme – 225 Minuten

Schlüsselbegriffe

A/B-Test, Back-to-Back-Test, Datendrift, Drift-Test, funktionale Leistung von ML, gegnerischer Test, Konzeptdrift, metamorpher Test, ML-Modelltest, Review

Spezifische Schlüsselbegriffe für KI

Überanpassung, Unteranpassung

Lernziele und Ziele der praktischen Übungen für Kapitel 6: Die Lernenden können …

6.1 Modelltest für ML-Systeme

AI-6.1.1 (K2) … Beispiele für Testansätze zur Risikominderung bei ML-Modellen nennen.

AI-6.1.2 (K2) … den Zweck und den Schwerpunkt von Reviews der Dokumentation von ML-Modellen erklären.

AI-6.1.3 (K2) … erklären, wie der Test der funktionalen Leistung für probabilistische ML-Systeme durchgeführt wird.

AI-6.1.4 (K2) … den gegnerischen Test von ML-Systemen zusammenfassen.

AI-6.1.5 (K3) … den metamorphen Test verwenden, um Testfälle für ein bestimmtes Szenario abzuleiten.

HO-6.1.6 (H2) … selbstständig metamorphe Tests anwenden.

AI-6.1.7 (K2) … erklären, wie Drift-Tests bei ML-Systemen im Produktivbetrieb eingesetzt werden.

AI-6.1.8 (K2) … erklären, wie Überanpassung und Unteranpassung durch Testen erkannt werden.

AI-6.1.9 (K2) … erklären, wie A/B-Tests im Zusammenhang mit ML-Systemen eingesetzt werden.

AI-6.1.10 (K2) … erklären, wie Back-to-Back-Tests im Zusammenhang mit ML-Systemen eingesetzt werden.

© International Software Testing Qualifications Board

6.1 Modelltest für ML-Systeme

Der ML-Modelltest umfasst die Berücksichtigung spezifischer Risiken. Dazu gehören funktionale Risiken wie Verzerrungen, Überanpassung und Sicherheitslücken gegenüber gegnerischen Angriffen sowie nicht-funktionale Risiken wie mangelnde Robustheit und Performanz der KI sowie Risiken bei der Bereitstellung.

6.1.1 Risiken und Maßnahmen zur Risikominderung beim maschinellen Lernen

Die folgende Tabelle listet Beispiele für Risiken bei ML-Modellen und entsprechende Verfahren zum Testen auf, die zur Risikominderung eingesetzt werden könnten:

Potenzielles Risiko	Mögliche Maßnahme zur Risikominderung
Verzerrtes oder unfaires ML-Modell	Testen auf Verzerrung – siehe 5.1.2
Unethisches Modell	Ethischer Systemtest
Gegnerische Beispiele	Gegnerischer Test – siehe 6.1.4
Überangepasstes Modell	Testen auf Überanpassung – siehe 6.1.8
Unterangepasstes Modell	Testen auf Unteranpassung – siehe 6.1.8
Inakzeptabler Datendrift Inakzeptabler Konzeptdrift	Drift-Test – siehe 6.1.7
Das Modell verursacht Seiteneffekte	Seiteneffekttest
Das Modell weist Belohnungs-Hacking auf	Testen des Belohnungs-Hackings
Fehlerzustand der Modell-API	API-Test – siehe 7.1.2
Erforderliche Leistungsmetriken von ML werden nicht erreicht (z. B. mangelnde Genauigkeit, mangelnde Sensitivität *(recall)*)	Testen der funktionalen Leistung von ML – siehe 6.1.3
Funktionale Fehler Nicht-funktionale Fehlerzustände	Metamorpher Test – siehe 6.1.5
Testorakelproblem	Metamorpher Test – siehe 6.1.5 Back-to-Back-Test – siehe 6.1.10 A/B-Test – siehe 6.1.9
Unzureichende Systemanforderungen	Anforderungs-Review Red-Teaming – siehe 4.2.2 Explorativer Test
Mangelnde KI-Robustheit des Modells gegenüber unerwarteten Eingaben	Gegnerischer Test – siehe 6.1.4 Fuzzing
Unzureichende Performanz des ML-Modells	Performanztest
Mangelhafte Modelldokumentation (z. B. Funktion, Genauigkeit, Schnittstelle)	Review der Modelldokumentation – siehe 6.1.2

© International Software Testing Qualifications Board

Potenzielles Risiko	Mögliche Maßnahme zur Risikominderung
Modellaktualisierungen führen zu Fehlerzuständen	Back-to-Back-Test – siehe 6.1.10
Modellaktualisierungen verringern die funktionale Leistung des ML-Modells	A/B-Test – siehe 6.1.9
Die Bereitstellung des aktualisierten Modells führt zu einer sofortigen Fehlerwirkung	Smoke-Test
Die Bereitstellung des aktualisierten Modells führt zu einer Regression	Regressionstest
Sicherheitslücken (*security*) Schwachstellen der Sicherheit (*safety*) Verletzungen der Privatsphäre Schädliche oder unerwünschte Ergebnisse (z. B. rassistische Ansichten, gefährliche Anleitungen)	Red-Teaming – siehe 4.2.2

6.1.2 Dokumentation und Review von ML-Modellen

Eine umfassende Dokumentation von ML-Modellen ist nicht nur eine Formalität, sondern eine unverzichtbare Ressource. Im Gegensatz zu Systemen, bei denen der Quellcode direkt inspiziert werden kann, stellen ML-Systeme aufgrund der eingeschränkten Nachvollziehbarkeit maschinengenerierten Codes, des ihnen innewohnenden „Black-Box“-Charakters und ihrer Abhängigkeit von Daten besondere Herausforderungen dar. Da Modelle zudem häufig aktualisiert werden, wird die Dokumentation zum wichtigsten Instrument für Entwickler, Tester und Aufsichtsbehörden, um KI-basierte Systeme zu verstehen, zu bewerten und ihnen zu vertrauen. Transparenz spielt eine zentrale Rolle bei der Ermöglichung dieses Verständnisses, indem sie es den Stakeholdern erlaubt, das Verhalten des Modells, die Entscheidungslogik und die Datenherkunft während des gesamten KI-Lebenszyklus zu verfolgen.

Standardisierte Dokumentation verbessert die Kommunikation, unterstützt fundierte Entscheidungsfindung und die Überprüfung der Qualität und Wartbarkeit von ML-Modellen. Sie gewinnt für die regulatorische Konformität zunehmend an Bedeutung, wie Rahmenwerke wie die KI-Verordnung der EU [EU AI Act] es zeigen, die Transparenzverpflichtungen zur Geltung bringen, welche eine klare Dokumentation von Modellentscheidungen, Einschränkungen und Maßnahmen zur Interpretierbarkeit erfordern. Bei Systemen mit hohem Risiko ist das Bestehen eines Audits oft eine Voraussetzung für die Bereitstellung, während bei allen Systemen gründliche Reviews zur Sicherstellung der Qualität beitragen.

Es gibt mehrere Dokumentations-Frameworks, darunter Modellkarten, die einen kompakten Überblick über die beabsichtigten Verwendungszwecke, die Evaluierungsergebnisse und ethische Überlegungen eines Modells bieten [MODEL_DOC], sowie Datenblätter für Datensätze, die standardisierte Formate zur Beschreibung von Datensätzen bereitstellen, einschließlich ihrer Motivation, Zusammensetzung, ihres Erhebungsprozesses und ihrer Nutzung [DATA_DOC].

Eine effektive Dokumentation dient Entwicklern und Testern als Checkliste. Die folgende Liste umreißt die typischen Inhalte, die für eine umfassende Dokumentation des ML-Modells erwartet werden:

- **Allgemeines:** Identifikatoren, Beschreibung, Entwickler, Version, Datum, Kontakt, Lizenz, Hardwareanforderungen.
- **Entwurf:** Annahmen, technische Entscheidungen, ML-Algorithmus.
- **Nutzung:** Verwendungszweck, primäre/sekundäre Nutzungen, Benutzer, selbstlernender Ansatz, Verzerrungen, Ethik, Sicherheit, Transparenz, Schwellenwerte, Plattform, Datendrift, Konzeptdrift.

© International Software Testing Qualifications Board

- **Datensätze:** Merkmale, Quelle, Erfassung, Verfügbarkeit, Vorverarbeitung, Verwendung, Inhalt, Kennzeichen, Größe, Datenschutz, Sicherheit (*security*), Verzerrung/Fairness, Einschränkungen.
- **Testen:** Details zum Testdatensatz, Unabhängigkeit des Testens, Testergebnisse, Testarten und -verfahren (z. B. funktionale Tests, gegnerische Tests).
- **Funktional:** Kennzahlen, Validierungsdatensatz, Schwellenwerte, tatsächliche Leistung.
- **Nicht-funktional:** Skalierbarkeit, Zuverlässigkeit, Verfügbarkeit, Performanz (z. B. Latenz, Ressourcennutzung), Wartbarkeit, Robustheit.
- **Betrieblich:** Bereitstellungsplan, Bereitstellungsumgebung, Rechenressourcen, Monitoring von Metriken/Warnmeldungen, Strategie zur Nachtrainierung, Plan zur Modellaktualisierung/zum Rollback, Abkündigungsplan, Sicherheit (*security*) (gegnerische Risiken), Methoden zur Erklärbarkeit.

Das Review der Dokumentation anhand solcher Checklisten ist eine zentrale Testaktivität mit folgenden Zielen:

- Identifizieren von fehlenden Informationen, Ungenauigkeiten und Inkonsistenzen.
- Verbessern der Klarheit und Lesbarkeit.
- Verbesserung der Wartbarkeit, indem verbesserungswürdige Stellen in der Dokumentation identifiziert werden.
- Sicherstellung ausreichender Informationen für Test- und Bereitstellungsaktivitäten.
- Überprüfung, ob alle relevanten regulatorischen Anforderungen erfüllt wurden.

6.1.3 Testen der funktionalen Leistung für probabilistische ML-Systeme

Funktionale Leistungstests bewerten, wie gut ein ML-Modell die funktionale Leistung von ML erfüllt. Dazu werden Metriken wie z. B. Genauigkeit, Sensitivität, Präzision und F1-Wert (siehe Abschnitt 3.3) gemessen und die Ergebnisse mit definierten Akzeptanzkriterien verglichen.

Diese Tests für probabilistische ML-Systeme gehen über einzelne Prüfungen mit „Bestanden/Fehlgeschlagen“ hinaus, um statistisch die Leistung eines Modells gegen seine Akzeptanzkriterien [STATS] zu messen. Dieser Ansatz ist notwendig, um den inhärenten Nicht-Determinismus in ML-Systemen zu berücksichtigen, indem das Verhalten anhand eines großen, repräsentativen Datensatzes evaluiert wird (siehe 4.1.2).

Eine Vorbedingung für diese Art von Testen ist die Definition statistischer Akzeptanzkriterien. Anstelle eines einfachen Zielwerts kann eine Anforderung eine Leistungsmetrik, eine Fehlerspanne (*margin of error*, MoE) und ein Konfidenzniveau (*confidence level*, CL) festlegen. Eine solche Anforderung kann zur Bestimmung der erforderlichen Mindestanzahl an Tests herangezogen werden. Mit steigender Anzahl von Tests gibt es zwei Möglichkeiten:

- Legt man CL fest, verringert sich MoE. Das bedeutet, dass das Testergebnis präziser wird.
- Legt man MoE fest, steigt CL. Das bedeutet, dass das Testergebnis sicherer wird.

Ein Kriterium könnte beispielsweise lauten: „98 % Genauigkeit mit einer MoE von ± 4 % bei einem CL von 95 %“. Um zu bestätigen, dass die gemessene Genauigkeit maximal eine MoE von ± 4 % bei einem CL von 95% aufweist, ist eine Stichprobengröße von 601 Testfällen erforderlich. Diese wird anhand der Formel zur Berechnung der Stichprobengröße für die Schätzung eines Populationsanteils ermittelt. Diese Stichprobengröße basiert auf der konservativen Annahme, dass die tatsächliche Genauigkeit zwischen 0 % und 100 % liegen könnte, was die größtmögliche Unsicherheit darstellt, aber dennoch unter allen Bedingungen eine MoE von ± 4% garantiert.

Um eine geforderte Genauigkeit von mindestens 98% zu erreichen, müssen mindestens 589 der 601 Testfälle bestanden sein. Wenn nach der Ausführung aller 601 Testfälle die beobachtete Genauigkeit 98 % beträgt, ist die gemessene MoE bei einem CL von 95 % geringer als ± 4 % (etwa ± 1,1 %), da die Varianz bei hohen Genauigkeitsniveaus geringer ist. Die MoE wird anhand der Fehlerspannen-Formel für einen Stichprobenanteil berechnet. Das bedeutet, dass es beim Testen nicht immer notwendig ist, alle 601

© International Software Testing Qualifications Board

Testfälle auszuführen. Das sequenzielle Testen bietet einen formalen statistischen Rahmen für dieses vorzeitige Beenden. Anstatt stets die gesamte feste Stichprobe auszuführen, analysieren diese Ansätze die Ergebnisse fortlaufend und brechen vorzeitig ab, sobald ausreichende Belege das Ziel der Genauigkeit stützen (oder es verwerfen). Bleibt die beobachtete Genauigkeit durchgehend hoch (z. B. nicht unter 98 %), so nimmt die statistische Unsicherheit mit zunehmender Anzahl durchgeführter Tests ab. In diesem Fall wird der erforderliche MoE von ± 4 % bei einem CL von 95% nach etwa 170 Testfällen erreicht, sodass das Testen vorzeitig beendet werden kann.

> HINWEIS: Die Formeln und numerischen Berechnungen in diesem Beispiel (für Stichprobengröße, Fehlerspanne und Konfidenzniveau) dienen lediglich der Veranschaulichung und dem Verständnis; die Prüfungsteilnehmer müssen diese Werte in der Zertifizierungsprüfung nicht anhand statistischer Formeln ableiten oder berechnen.

Für sicherheitskritische Systeme kann eine strengere Anforderung an die Zuverlässigkeit gelten, beispielsweise „99 % Zuverlässigkeit bei 95 % Konfidenz“, was 299 Testfälle erfordern würde, die alle bestanden werden müssen, um die Kriterien zu erfüllen.

Um diese Kriterien zu validieren, erfordert die Prüfung einen umfangreichen Testdatensatz, der gänzlich unabhängig von den Trainings- und Validierungsdatensätzen ist. Dieser Testdatensatz muss eine repräsentative Stichprobe des operativen Eingabebereichs (siehe 5.1.4) darstellen, um sicherzustellen, dass die Evaluierung reale Bedingungen widerspiegelt. Die Testfälle werden dann unter Verwendung des ML-Modells innerhalb eines ML-Entwicklungs-Frameworks ausgeführt, das statistische Analysen durchführen kann.

Schließlich werden die aggregierten Ergebnisse mit statistischer Konfidenz interpretiert und berichtet, nicht als einfaches Verhältnis von „Bestanden“ zu „Fehlgeschlagen“. Ein abschließender Testbericht würde beispielsweise lauten: „Das Modell erreichte eine Genauigkeit von 94 % ±4 % bei einem CL von 95 %.“ Dies ermöglicht es den Stakeholdern, die Bandbreite der zu erwartenden Leistung im Produktivbetrieb nachzuvollziehen und eine fundierte Entscheidung darüber zu treffen, ob die funktionale Leistung des Modells für die Bereitstellung akzeptabel ist.

6.1.4 Gegnerischer Test von ML-Systemen

Beim gegnerischen Test (*adversarial testing*) werden dem Modell gezielt Störgrößen in den Eingabedaten zugeführt, die für den Menschen oft nicht wahrnehmbar sind. Diese Eingaben sind so entworfen, dass sie das Modell zu falschen Vorhersagen veranlassen; sind sie erfolgreich, werden sie als gegnerische Beispiele bezeichnet. Daher bestehen die Testeingaben für den gegnerischen Test und somit potenzielle gegnerische Beispiele oft aus leicht modifizierten Versionen legitimer Eingaben, die dazu führen, dass das Modell sie falsch klassifiziert.

Die Identifizierung von Sicherheitslücken durch gegnerische Tests ermöglicht es Entwicklern, Schutzmaßnahmen zu integrieren und die Robustheit des Modells gegenüber gegnerischen Beispielen zu erhöhen. Dabei kann es sich um zufällige gegnerische Beispiele handeln, auf die das Modell stößt, oder um gegnerische Angriffe, bei denen gegnerische Beispiele böswillig eingesetzt werden. Die Generierung effektiver Test-Eingaben für gegnerische Tests ist technisch komplex, und es ist eine ständige Herausforderung, mit den sich weiterentwickelnden Angriffstechniken Schritt zu halten.

Gegnerische Tests können mittels Black-Box-Tests durchgeführt werden, wobei der Fokus auf dem Eingabe- und Ausgabeverhalten des Modells liegt, ohne dass Kenntnisse über dessen interne Funktionsweise erforderlich sind. Dies lässt sich erreichen, indem ein äquivalentes Modell erstellt wird, dessen interne Arbeitsweisen bekannt sind, und aus dem anhand dieser Kenntnisse gegnerische Test-Eingaben generiert werden können. Basierend auf der Annahme, dass äquivalente Modelle gemeinsame Klassifikationsgrenzen aufweisen (d. h. Übertragbarkeit), können die gegnerischen Tests dann auf das ursprüngliche Modell angewendet werden. Im Gegensatz dazu kann ein Brute-Force-Ansatz mit einer großen Anzahl von Tests verwendet werden, in der Hoffnung, dass einige zufällige Tests mit einem gegnerischen Beispiel übereinstimmen.

© International Software Testing Qualifications Board

Gegnerische Tests können auch mithilfe von White-Box-Tests durchgeführt werden. Das Verständnis der internen Abläufe eines ML-Modells (Architektur, Parameter, Trainingsprozess) erleichtert in der Regel die Erstellung gegnerischer Beispiele.

Gegnerische Tests können manuell durch die Erstellung spezifischer gegnerischer Beispiele oder mithilfe automatisierter Algorithmen durchgeführt werden, die eine große Anzahl von Variationen generieren, um wirksame gegnerische Eingaben zu finden.

6.1.5 Metamorpher Test

Der metamorphe Test (MT) ist ein Testverfahren, bei dem neue (Folge-)Testfälle aus einem zuvor bestandenen Ausgangstestfall abgeleitet werden. Ein oder mehrere Folgetestfälle werden generiert, indem der Ausgangstestfall mithilfe einer metamorphen Relation (MR, Plural: MRs) verändert (metamorphisiert) wird. Die MR basiert auf einer Eigenschaft einer geforderten Funktion des Testobjekts und beschreibt, wie sich eine Änderung der Eingaben eines Testfalls in den erwarteten Ergebnissen desselben Testfalls widerspiegelt.

MT kann für die meisten Testobjekte verwendet werden und lässt sich sowohl auf funktionale als auch auf nicht-funktionale Tests anwenden. Tester überprüfen Testziele wie Konsistenz (die Ausgaben stimmen bei ähnlichen Eingaben überein), Monotonie (die Ausgaben ändern sich in Richtung der Eingabe) und Invarianz (die Ausgaben bleiben unter Störungen stabil). MT ist besonders nützlich, wenn die Generierung erwarteter Ergebnisse problematisch ist, weil kein erschwingliches Testorakel verfügbar ist. Dies ist bei einigen ML-Systemen der Fall, die Big Data nutzen, oder bei Systemen, bei denen Tester nicht genügend Klarheit darüber haben, wie das ML-Modell seine Vorhersagen ableitet, was häufig der Fall ist. Im Bereich der KI und des ML wurde der metamorphe Test bereits zum Testen von Bilderkennung, Suchmaschinen, Routenoptimierung und Spracherkennung eingesetzt.

MT wird in der Regel gegenüber herkömmlichen orakelbasierten Tests bevorzugt, wenn:

- aufgrund der Undurchsichtigkeit des Modells oder des Datenumfangs keine zuverlässigen erwarteten Ergebnisse vorliegen;
- das System eine Black Box ist; oder
- relationale Eigenschaften (keine absoluten Werte) für das Vertrauen in das Testergebnis ausreichen.

MT basiert häufig auf einem bestandenen Ausgangstestfall und ist auch dann nützlich, wenn die Generierung der erwarteten Ergebnisse nicht möglich ist. So kann z. B. ein Programm eine Funktion implementieren, die für einen menschlichen Tester zu komplex ist, um sie zu replizieren und als Test-Orakel zu verwenden, so wie es bei einigen komplexen ML-Systemen der Fall ist. In dieser Situation kann MT benutzt werden, um eine Reihe von Ausgabewerten zu erzeugen, deren Beziehungen zueinander auf Gültigkeit überprüft werden anstatt der tatsächlichen Ergebnisse. Wenn diese Beziehungen zutreffen, dann wird mit dieser Form des MT ein höheres Vertrauen in das Programm erreicht. Ein Beispiel dafür ist ein ML-System zur Risikobewertung, das die Lebenserwartung vorhersagt und bei steigender Anzahl geraucher Zigaretten eine fallende Lebenserwartung prognostizieren sollte (Monotonie).

Tester leiten MRs aus Fachkenntnissen, Anforderungen oder Domänen-Eigenschaften (z. B. physikalischen Gesetzen) ab. MRs können durch ein Experten-Review validiert werden, indem sie auf Referenzmodellen ausgeführt und die Überdeckung der Sonderfälle überprüft wird.

Falsche MRs (z. B. das Übersehen komplexer Wechselwirkungen zwischen Variablen) oder unvollständige MR-Sätze können zu falschem Vertrauen führen. MT erkennt relationale Fehler, jedoch nicht alle absoluten Fehler; daher sollte es in Kombination mit anderen Testverfahren eingesetzt werden.

6.1.6 Praktische Übung zum Ziel HO-6.1.6 (H2): … selbstständig metamorphe Tests anwenden

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

© International Software Testing Qualifications Board

- Ableitung mehrerer MRs für ein gegebenes MLS. Diese sollten sowohl MRs enthalten, bei denen die erwarteten Ergebnisse der Ausgangstestfälle und der Folgetestfälle identisch sind, als auch solche, bei denen sie voneinander abweichen.
- Erstellen von Ausgangstestfällen für das MLS. Es muss nicht garantiert sein, dass diese bestanden werden, aber die Lernenden sollten an die Grenzen des MT erinnert werden, wenn keine bestandenen Ausgangstestfälle verfügbar sind.
- Verwendung der abgeleiteten MRs und der generierten Ausgangstestfälle zur Ableitung von Folgetestfällen.
- Ausführen der Folgetestfälle.

6.1.7 Drift-Test

Drift-Tests können verwendet werden, um zwei Formen von Drift in MLS im Produktivbetrieb zu identifizieren:

- Datendrift: tritt auf, wenn sich die statistischen Eigenschaften der Eingabedaten im Produktivbetrieb im Laufe der Zeit ändern. Beispielsweise können Faktoren wie Veränderungen im Benutzerverhalten oder saisonale Schwankungen erhebliche Unterschiede der aktuellen Eingabedaten von den Daten, mit denen das Modell trainiert wurde, bedeuten. So stößt ein Spamfilter beispielsweise auf neue Arten von Phishing-Angriffen, die zum Zeitpunkt seines Trainings noch nicht existierten.
- Konzeptdrift: tritt auf, wenn sich die Beziehung zwischen den Eingabedaten und der korrekten Ausgabe im Laufe der Zeit verändert. Das bedeutet, dass die ursprünglich vom Modell erlernten Muster oder Regeln die aktuelle Realität nicht mehr widerspiegeln. Beispielsweise könnte aufgrund neuer Finanzvorschriften eine Transaktionsart, die zuvor als „risikoarm“ galt, nun als „risikoreich“ eingestuft werden. Die Bedeutung der Daten hat sich geändert, wodurch die vom Modell erlernten Entscheidungsgrenzen veraltet sind, was zu einem Rückgang der Genauigkeit der Vorhersagen führt.

Dynamische Drift-Tests sind auf die Verfügbarkeit von Benutzer-Feedback angewiesen, welches die aktuelle Grundwahrheit liefert. Diese wird mit der Ausgabe des Modells verglichen, wobei die Differenz zwischen beiden ermittelt und mit einem Schwellenwert abgeglichen wird. Benutzer-Feedback kann direkt oder indirekt sein. Bei einem System, das Filme empfiehlt, kann zum Beispiel ein direktes Feedback der Benutzer erfolgen, indem diese eine Empfehlung bewerten. Ein Beispiel für indirektes Feedback für dasselbe System würde aus den Daten zu den Filmen gewonnen, welche die Benutzer angesehen haben.

Der statische Drift-Test ist nicht von der aktuellen Grundwahrheit abhängig, sondern vergleicht stattdessen die statistischen Eigenschaften der Verteilungen der Eingabe- und der vorhergesagten Ausgabedaten mithilfe eines Tests wie dem Kolmogorov-Smirnov-Test [STATS]. Ein signifikanter Unterschied in einer dieser Verteilungen ist ein Indikator dafür, dass Drift aufgetreten ist.

6.1.8 Testen auf Überanpassung und Unteranpassung

Überanpassung und Unteranpassung sind zwei der drei möglichen Ergebnisse bei ML-Modellen, wobei das dritte ein „richtig angepasstes“ Modell ist. Das Testen auf Über- und Unteranpassung sollte während des Trainings, der Evaluierung und des Modell-Tunings erfolgen.

Überanpassung tritt auf, wenn ein Modell die Trainingsdaten zu gut lernt, sodass es eher das Rauschen in den Daten als das zugrundeliegende Muster erfasst. Dies führt zu einer schlechten Generalisierung auf neue, bisher unbekannte Daten.

Um auf Überanpassung zu prüfen, wird die funktionale Leistung von ML des Modells anhand eines separaten Testdatensatzes bewertet, der während des Trainings nicht verwendet wurde. Dieser Testdatensatz sollte einige weniger häufige Beispiele enthalten, die während des Trainings wahrscheinlich nicht verwendet wurden. Das Modell könnte überangepasst sein, wenn es auf dem Testdatensatz deutlich schlechter abschneidet als auf dem Validierungsdatensatz.

© International Software Testing Qualifications Board

Unteranpassung tritt auf, wenn ein Modell zu einfach ist, um die zugrundeliegende Struktur der Daten zu erfassen, oder wenn der Trainingsdatensatz keine Merkmale enthält, die eine wichtige Beziehung zwischen Eingaben und Ausgaben widerspiegeln. Dies führt zu einer schlechten funktionalen Leistung von ML sowohl auf dem Trainingsdatensatz als auch auf dem Validierungsdatensatz.

Während des Tests lässt sich eine Unteranpassung durch die Auswertung der funktionalen Leistungsmetriken von ML erkennen, wie beispielsweise Genauigkeit, Präzision, Sensitivität oder F1-Wert. Sind diese Metriken sowohl im Trainings- als auch im Validierungsdatensatz durchweg niedrig, deutet dies auf eine Unteranpassung des Modells hin.

Auch die visuelle Inspektion der Lernkurven des Modells kann dabei helfen, eine Unteranpassung zu erkennen. Wenn die Trainings- und Validierungsfehler hoch bleiben und relativ nahe beieinanderliegen, ohne dass im Verlauf des Trainings eine signifikante Verbesserung eintritt, deutet dies auf eine Unteranpassung hin.

Zusammenfassend gehören zur Aufdeckung von Überanpassung und Unteranpassung während des Tests die Evaluierung der funktionalen Leistung von ML des Modells anhand von Validierungsdaten, die Analyse der funktionalen Leistungsmetriken von ML und die Untersuchung der Lernkurven.

6.1.9 A/B-Test

Der A/B-Test ist ein Ansatz, bei dem die Reaktion zweier Varianten des Programms (A und B) auf dieselben Eingaben verglichen wird, um festzustellen, welche der beiden Varianten besser ist. Es handelt sich um einen statistischen Testansatz, der in der Regel den Vergleich von Testergebnissen aus mehreren Testläufen erfordert, um Unterschiede zwischen den Programmen zu ermitteln.

Ein einfaches Beispiel für diesen Ansatz ist der Versand von zwei Werbeangeboten per E-Mail an eine in zwei Gruppen unterteilte Marketingliste. Die Hälfte der Liste erhält Angebot A, die andere Hälfte Angebot B; der Erfolg der einzelnen Angebote hilft bei der Entscheidung, welches in Zukunft verwendet werden soll. Viele E-Commerce- und webbasierte Unternehmen setzen A/B-Tests in Produktion ein, indem sie verschiedene Verbraucher auf unterschiedliche Funktionen umleiten, um die Präferenzen der Verbraucher zu ermitteln.

A/B-Tests sind ein Ansatz zur Lösung des Testorakelproblems, bei dem in der Regel das bestehende System als partielles Testorakel genutzt wird. A/B-Tests generieren keine Testfälle und liefern keine Anleitung dazu, wie die Tests entworfen werden sollten, obwohl betriebliche Eingaben oft in die Tests einfließen.

A/B-Tests können zum Testen von Aktualisierungen eines KI-basierten Systems verwendet werden, vorausgesetzt, es gibt vereinbarte Akzeptanzkriterien, wie beispielsweise funktionale Leistungsmetriken von ML, wie in Abschnitt 3.3 beschrieben. Bei jeder Aktualisierung des Systems wird mittels A/B-Tests festgestellt, ob die aktualisierte Variante mindestens genauso gut oder besser funktioniert als die vorherige Variante.

Ein solcher Testansatz eignet sich für einen einfachen Klassifikator, kann aber auch zum Testen weitaus komplexerer Systeme verwendet werden. So lässt sich beispielsweise eine Aktualisierung zur Verbesserung der Effektivität eines Routenplanungssystems für den Verkehr in einer Smart City mithilfe von A/B-Tests prüfen. Dies geschieht beispielsweise durch den Vergleich der durchschnittlichen Pendelzeiten für zwei Varianten des Systems in aufeinanderfolgenden Wochen.

A/B-Tests können auch zum Testen selbstlernender Systeme eingesetzt werden. Wenn das System eine Änderung vornimmt, werden automatisierte Tests durchgeführt, und die daraus resultierenden Systemeigenschaften werden mit denen vor der Änderung verglichen. Wenn das System verbessert wurde, wird die Änderung übernommen; andernfalls kehrt das System in seinen vorherigen Zustand zurück.

Die gängigsten statistischen Testverfahren für A/B-Tests sind der t-Test, der z-Test, der Chi-Quadrat-Test und der Mann-Whitney-U-Test [STATS].

© International Software Testing Qualifications Board

6.1.10 Back-to-Back-Test

Der Back-to-Back-Test bietet eine praktische Lösung für das Testorakelproblem.

Beim Back-to-Back-Test wird eine alternative Version des Systems als Referenzpunkt (ein Pseudo-Orakel) verwendet und deren Ausgaben mit denen des zu testenden Systems verglichen, wenn beide dieselben Eingaben erhalten. Dieses Pseudo-Orakel kann ein bestehendes System sein oder eines, das speziell für Testzwecke entwickelt wurde, was jedoch mit Kosten verbunden ist.

Im Idealfall sollten das Pseudo-Orakel und das zu testende System keine gemeinsamen Softwarekomponenten aufweisen. Andernfalls könnten beide Systeme denselben Fehlerzustand enthalten, was dazu führen würde, dass ihre Ausgaben übereinstimmen, selbst dann, wenn beide falsch sind. Dies kann angesichts der weit verbreiteten Nutzung wiederverwendbarer Open-Source-KI-Komponenten in der ML-Entwicklung besonders problematisch sein. Aus diesem Grund wird das Pseudo-Orakel oft von einem anderen und im Idealfall unabhängigen Team entwickelt, das möglicherweise andere ML-Entwicklungs-Frameworks, Algorithmen oder Modelleinstellungen verwendet. Manchmal kann herkömmliche Software als Pseudo-Orakel dienen, wenn sie dasselbe Problem löst.

Bei der Durchführung von funktionalen Back-to-Back-Tests muss das Pseudo-Orakel lediglich das funktionale Verhalten nachbilden. Es muss nicht dieselben nicht-funktionalen Anforderungen erfüllen wie das zu testende System, was seine Entwicklung potenziell kostengünstiger macht.

Dieser Testansatz erfordert lediglich die Generierung von Testeingaben, nicht jedoch von erwarteten Ergebnissen, da das Pseudo-Orakel den Vergleichspunkt liefert. Diese Eingaben können aus bestehenden Testfällen stammen, wie beispielsweise Regressionstestsuiten, oder automatisch aus Trainingsdaten generiert werden, was die Durchführung einer großen Anzahl von Tests ermöglicht, sofern die automatisierte Testdurchführung unterstützt wird.

Back-to-Back-Tests bieten einen erheblichen Mehrwert bei der Migration eines MLS in eine neue Umgebung, beispielsweise beim Übergang von der Entwicklung in die Produktion, sowie beim Vergleich von Testergebnissen über verschiedene Umgebungen hinweg. Darüber hinaus kann dieser Testansatz subtile Fehler im Modellverhalten aufdecken, die durch andere Testansätze möglicherweise nicht erkennbar wären, insbesondere beim Vergleich von Reaktionen über ein breites Spektrum von Sonderfällen oder ungewöhnlichen Eingaben hinweg.

Ein wesentlicher Unterschied zwischen A/B-Tests (siehe 6.1.9) und Back-to-Back-Tests besteht darin, dass A/B-Tests zum Vergleich zweier Varianten desselben MLS unter Verwendung von funktionalen Leistungsmetriken von ML und statistischen Verfahren eingesetzt werden, während Back-to-Back-Tests der Fehlererkennung dienen.

© International Software Testing Qualifications Board

7 ML-Entwicklungstest – 30 Minuten

Schlüsselbegriffe
funktionale Leistung von ML, ML-Entwicklungstest, Shadow-Test

Spezifische Schlüsselbegriffe für KI
Keine

Lernziele für Kapitel 7: Die Lernenden können …

7.1 ML-Entwicklungstest

AI-7.1.1 (K2) … Beispiele für Ansätze nennen, die zur Verminderung von Risiken bei der Entwicklung von ML-Systemen verwendet werden.

AI-7.1.2 (K2) … die verschiedenen Formen des Bereitstellungstests für ML-Systeme erklären.

© International Software Testing Qualifications Board

7.1 ML-Entwicklungstest

Dieses Kapitel konzentriert sich speziell auf die Behandlung von Risiken, die durch ML-Entwicklungswerkzeuge, Konfigurationsentscheidungen und Bereitstellungsmechanismen eingeführt werden, und nicht auf das ML-Modell selbst. Es behandelt Testansätze und Testarten wie API-Test, Testen der funktionalen Leistung von ML, A/B-Test, Back-to-Back-Test und Reviews, um KI-Robustheit und Effizienz zu verifizieren.

7.1.1 Risiken bei der Entwicklung von ML-Systemen und Maßnahmen zu deren Minderung

Die folgende Tabelle enthält Beispiele für Risiken bei der Entwicklung von ML-Systemen sowie entsprechende Testmaßnahmen, die zur Risikominderung eingesetzt werden könnten.

Potenzielles Risiko	Mögliche Maßnahme zur Risikominderung
Fehlerhafte oder unbeabsichtigte Verwendung von Bibliotheks- oder Framework-APIs (z. B. TensorFlow, PyTorch)	API-Test – siehe 7.1.2
Ungeeignete Auswahl eines Frameworks	Review der Framework-Eignung
Probleme innerhalb des Algorithmus, Modells oder Entwicklungsframeworks, die zu systembedingter Unfairness führen	Testen auf Verzerrung – siehe 5.1.2
Fehlerhafte Installation oder fehlerhafter Build des Frameworks	Smoke-Testen
Fehlerhafte Implementierung der Auswertung durch das Framework	Reviews des Framework-Auswertungscodes Gegenprüfung (*cross-check*) der Auswertungsergebnisse des Frameworks (z. B. gegen manuelle Referenzwerte)
Unzureichende Performanz (z. B. langsame Reaktionszeiten des Frameworks)	Performanztest
Mangelhafte Gebrauchstauglichkeit des Frameworks	Gebrauchstauglichkeitstest
Defekt in einer vom Framework verwendeten Bibliothek (z. B. Defekt in PyTorch) Fehlerhafte Implementierung eines Algorithmus	Testen der funktionalen Leistung von ML – siehe 6.1.3 Back-to-Back-Test – siehe 6.1.10
Sicherheitslücken im Framework	Sicherheitstest (*security testing*)
Unzureichende Benutzerdokumentation des Frameworks	Review der Framework-Dokumentation
Ungeeignete Auswahl eines Algorithmus	Review der Algorithmus-Eignung A/B-Test – siehe 6.1.9
Ungeeignete Auswahl von Hyperparametern (z. B. Netzstruktur, Lernrate)	Testen der funktionalen Leistung von ML – siehe 6.1.3

© International Software Testing Qualifications Board

Potenzielles Risiko	Mögliche Maßnahme zur Risikominderung
	A/B-Test – siehe 6.1.9
Fehlerhafte Aufteilung der Daten auf Trainings-, Validierungs- und Testdatensätze	Review der Datenaufteilung
Ungeeignete Auswahl des Evaluierungsverfahrens (z. B. k-fache Kreuzvalidierung)	Testen der funktionalen Leistung von ML – siehe 6.1.3
Fehlinterpretation von Testergebnissen aufgrund der stochastischen Natur des Lernprozesses	Testen der funktionalen Leistung von ML – siehe 6.1.3
Bereitstellungsfehler (z. B. bei der Erzeugung einer angepassten Version für eine Zielplattform)	Smoke-Testen Testen der funktionalen Leistung von ML – siehe 6.1.3 A/B-Test – siehe 6.1.9
Das bereitgestellte Modell ist nicht mit der Produktivumgebung kompatibel	Smoke-Testen Testen der Bereitstellung von ML-Systemen – siehe 7.1.2
Das bereitgestellte Modell stellt keine Verbesserung gegenüber dem aktuellen Modell dar	Shadow-Test – siehe 7.1.2

7.1.2 Bereitstellungstest für ML-Systeme

Die Bereitstellung von ML-Systemen umfasst mehrere wichtige Testaktivitäten, die darauf ausgerichtet sind zu verifizieren, dass das KI-basierte System in seiner Zielumgebung (z. B. Cloud, Edge-Gerät oder mobiles Gerät) korrekt und zuverlässig funktioniert. Jede der folgenden Testarten adressiert spezifische Risiken während der Bereitstellung.

- **Installierbarkeitstest:** Verifiziert, dass das ML-System in allen unterstützten Umgebungen erfolgreich installiert, konfiguriert und anschließend wieder deinstalliert werden kann. Dies umfasst das Testen von Systemabhängigkeiten (z. B. GPU-Treibern), der Kompatibilität mit Frameworks sowie die erfolgreiche Ausführung von Installationsskripten.
- **Rollback-Test:** Verifiziert die Fähigkeit des Systems, nach einer unzureichenden oder fehlgeschlagenen Bereitstellung erfolgreich zu einem zuvor stabilen und funktionsfähigen Zustand zurückzukehren. Dies kann sich ausschließlich auf das Modell beziehen oder das gesamte System (einschließlich der Daten-Pipeline) umfassen. Zu beachten ist, dass der Rollback-Test vor der Bereitstellung durchgeführt werden muss, um die Bereitschaft für ein Rollback nachzuweisen.
- **Kanarien-Test**: Validiert neue Bereitstellungen, indem ein aktualisiertes Modell zunächst nur für einen kleinen Teil der Benutzer in der Produktivumgebung (z. B. 5 % der Benutzer) freigegeben wird. Echtzeitmetriken wie Latenz, Genauigkeit und Fehlerraten werden überwacht, um Regressionen vor einer vollständigen Ausrollung zu erkennen.
- **Shadow-Test**: Führt ein neues Modell parallel zum aktuellen Produktivmodell in Echtzeit aus, wobei dieselben Anfragen an beide Systeme weitergeleitet werden, ohne die Antworten des Produktivmodells zu beeinflussen. Dadurch können neues und bestehendes Modell anhand realer Daten in einer kontrollierten und risikoarmen Umgebung verglichen werden. So lassen sich Fehler wie Leistungsregressionen und Datendrift vor der vollständigen Bereitstellung erkennen.

© International Software Testing Qualifications Board

- **Modellkonvertierungstest:** Verifiziert nach der Konvertierung eines ML-Modells vom ursprünglichen Trainingsformat in ein für die Ziel-Produktivumgebung geeignetes Bereitstellungsformat, dass es eine akzeptable Vorhersagegenauigkeit, ein konsistentes Verhalten und eine angemessene Performanz im Produktivbetrieb (z. B. Inferenzgeschwindigkeit und Speicherverbrauch) beibehält.
- **Geräteübergreifender Test:** Verifiziert, dass das ML-System auf allen vorgesehenen Bereitstellungszielen korrekt funktioniert – von mobilen Geräten und Edge-Geräten bis hin zu Cloud-Servern.
- **API-Test:** Verifiziert, dass das ML-System klar definierte und standardkonforme Schnittstellen bereitstellt. Validiert werden die korrekte Verarbeitung von Ein- und Ausgaben, Fehlermeldungen sowie Integrationsabläufe mit externen Systemen wie Datenquellen, Clients und der Daten-Pipeline.

© International Software Testing Qualifications Board

8 Abkürzungen

Abkürzung	Beschreibung
AIaaS	KI als Dienst (*AI as a Service*)
API	Anwendungsprogrammierschnittstelle (*application programming interface*)
CL	Konfidenzniveau (*confidence level*)
CNN	Faltendes neuronales Netz (*convolutional neural network*)
CPU	Zentraler Prozessor (*central processing unit*)
DL	Deep Learning
DNN	Tiefes neuronales Netz (*deep neural network*)
EDA	Explorative Datenanalyse
EU	Europäische Union
FN	Falsch negativ (*false negative*)
FP	Falsch positiv (*false positive*)
GAN	Generatives gegnerisches Netz (*generative adversarial network*)
GenAI	Generative KI (*generative AI*)
GPU	Grafikprozessor (*graphics processing unit*)
HO	Spezifisches Ziel einer praktischen Übung (*hands-on objective*)
IAA	Inter-Annotator Übereinstimmung (*inter-annotator agreement*)
KI	Künstliche Intelligenz (*artificial intelligence*, AI)
kMNC	k-Multisektions-Neuronenüberdeckung (*k-multisection neuron coverage*)
LIME	Lokal interpretierbare modell-agnostische Erklärungen (*Local Interpretable Model-Agnostic Explanations*)
LLM	Large Language-Modell (*large language model*)
LO	Lernziel (*learning objective*)
ML	Maschinelles Lernen
MLS	ML-System
MoE	Fehlerspanne (*margin of error*)
MR	Metamorphe Relation (*metamorphic relation*)
MT	Metamorpher Test
NBC	Neuronen-Grenzwertüberdeckung (*neuron boundary coverage*)
NLP	Verarbeitung natürlicher Sprache (*natural language processing*)
RAG	Retrieval-augmentierte Generierung (*retrieval-augmented generation*)
RN	Richtig negativ (*true negative*, TN)

© International Software Testing Qualifications Board

Abkürzung	Beschreibung
RNN	Rückgekoppeltes (rekurrentes) neuronales Netz (*recurrent neural network*)
RP	Richtig positiv (*true positive*, TP)
RT	Red Teaming
SVM	Support-Vektor Maschine (*support vector machine*)
XAI	Erklärbare KI (*explainable AI*)

© International Software Testing Qualifications Board

9 KI-spezifische und andere Begriffe[2]

Begriff (**Fett:** KI-spezifischer Begriff)	**Definition**
Adaptives KI-basiertes System	Ein KI-basiertes System, das sein Verhalten als Reaktion auf Veränderungen in seiner Produktivumgebung anpasst.
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 KI (*general AI*)	KI, die über das gesamte Spektrum der kognitiven Fähigkeiten ein dem Menschen vergleichbares intelligentes Verhalten zeigt. (nach [ISO/IEC TR 29119-11])
Analyse auf mittelbare Diskriminierung *(disparate impact analysis)*	Ein Verfahren zur Erkennung von Verzerrungen durch den Vergleich von Entscheidungen in ursprünglichen Szenarien mit abgewandelten Versionen, in denen sensible Attribute vertauscht wurden.
Annotation	Die Identifizierung von Objekten in Bildern mit Begrenzungsrahmen, um gekennzeichnete Daten für die Klassifikation zu erhalten.
Assoziation	Eine unüberwachte ML-Lerntechnik, die Beziehungen und Abhängigkeiten zwischen Stichproben identifiziert.
Ausreißer (*outlier*)	Eine Beobachtung, die außerhalb des allgemeinen Musters der Datenverteilung liegt.
Bayes'sches Modell	Ein statistisches Modell, das die Unsicherheit von Modell-Eingaben und -Ausgaben mit Hilfe von Wahrscheinlichkeiten darstellt.
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)*	Ein Ansatz, bei dem ein ML-Modell, genannt Agent, durch die Nutzung einer Versuch-und-Belohnung-Feedbackschleife mit seiner Umgebung lernt, um spezifische Ziele zu erreichen.

[2] Die Definitionen aller Schlüsselbegriffe sind im ISTQB®-Glossar aufgeführt (https://glossary.istqb.org/).

© International Software Testing Qualifications Board

Begriff (**Fett:** KI-spezifischer Begriff)	**Definition**
Big Data	Umfangreiche Datensätze, deren Merkmale in Bezug auf Volumen, Vielfalt, Geschwindigkeit und/oder Variabilität spezielle Technologien sowie Techniken zur Verarbeitung erfordern.
Bootstrap-Verfahren	Ein Resampling-Verfahren, bei dem wiederholt Stichproben mit Zurücklegen aus einem Trainingsdatensatz gezogen werden, um die funktionalen Leistungskriterien von ML eines ML-Modells zu schätzen.
Chatbot	Eine Anwendung, die dazu dient, ein Gespräch über Text od Text-to-Speech zu führen.
Clusterbildung (*clustering*)	Eine ML-Funktion, die ähnliche Datenpunkte zusammenfasst.
Clusterbildungsalgorithmus	Eine Art von ML-Algorithmus zur Gruppierung ähnlicher Objekte in Clustern.
Datenbeschaffung (*data acquisition*)	Die Beschaffung von Daten, die für das von einem ML-Modell zu lösende Geschäftsproblem relevant sind.
Datendrift	Eine im Laufe der Zeit eintretende Veränderung der Verteilung der Eingabedaten, die sich negativ auf die funktionale Leistung eines im Einsatz befindlichen ML-Modells auswirkt.
Daten-Pipeline	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.
Datenverzerrung (*data bias*)	Ein systematischer Fehler, der durch ungenaue, unvollständige oder nicht-repräsentative Trainingsdaten verursacht wird und zu unfairen Ergebnissen in ML-Modellen führt.
Datenvorbereitung *(data preparation)*	Die Aktivitäten der Datenbeschaffung, der Datenvorverarbeitung und der Merkmalsermittlung im ML-Workflow.
Datenvorverarbeitung (*data preprocessing*)	Die Aktivitäten der Datenbereinigung, Datentransformation, Datenerweiterung und Daten-Stichprobenziehung im ML-Arbeitsablauf.
Deep Learning	Maschinelles Lernen unter Verwendung tiefer neuronaler Netze, um komplexe Merkmale und Darstellungen aus großen Datensätzen automatisch zu erlernen.

© International Software Testing Qualifications Board

Begriff (**Fett:** KI-spezifischer Begriff)	**Definition**
Deepfake	Synthetische Medien wie Video, Audio oder Bilder, die mithilfe von KI erstellt oder bearbeitet wurden, um reale Personen oder Ereignisse überzeugend nachzuahmen oder sich als solche auszugeben.
Deterministisch	Ein Verhalten, bei dem Ausgaben und Endzustand durch d Anfangszustand und die Eingaben eindeutig bestimmt sind.
Edge-Computing	Der Teil einer verteilten Architektur, in dem Informationsverarbeitung in der Nähe des Ortes erfolgt, an de die Informationen verwendet werden.
Edge-KI	Die Bereitstellung von KI-Modellen auf lokalen Edge-Geräten, was eine Echtzeitverarbeitung nahe der Datenquelle ermöglicht, ohne auf cloudbasierte Systeme angewiesen zu sein.
Entscheidungsbaum	Ein baumartiges ML-Modell, dessen Knoten Entscheidung und dessen Äste mögliche Ergebnisse darstellen.
Epoche	Eine Iteration des ML-Trainings mit dem gesamten Trainingsdatensatz.
Erklärbare KI (*explainable AI*)	Forschungsbereich, der sich mit dem Verständnis der Faktoren befasst, welche die Ergebnisse von KI-Systemen beeinflussen.
Erweiterung (*augmentation*)	Die Erstellung neuer Datenpunkte auf der Grundlage eines bestehenden Datensatzes.
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	Ein interaktiver, visueller und hypothesengesteuerter Prozess zum Zusammenfassen, Erkunden und Verstehen der wesentlichen Merkmale und Muster von Daten.
F1-Wert (*F1-Score*)	Funktionale Leistungsmetrik von ML, die zur Evaluierung eines Klassifikators verwendet wird und ein Gleichgewicht zwischen Sensitivität und Präzision herstellt.
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.
Faltendes neuronales Netz (*convolutional neural network*)	Eine Art von Deep-Learning-Modell, das für die Verarbeitung rasterähnlicher Daten wie Bilder entworfen ist, und durch schichtweise Operationen räumliche Muster und Merkmale erkennen kann.
Fehlervorhersage (*defect prediction*)	Ein Verfahren zur Vorhersage der Bereiche innerhalb des Testobjekts, in denen Fehler auftreten werden, oder der Menge der vorhandenen Fehler.

© International Software Testing Qualifications Board

Begriff (**Fett:** KI-spezifischer Begriff)	**Definition**
Fixiertes KI-basiertes System *(locked AI-based system)*	Ein deterministisches KI-basiertes System mit einem festen, unveränderlichen Modell, das sein Verhalten nach der Bereitstellung nicht mehr ändert.
Foundation-Modell	Ein Allzweck-ML-Modell, das mithilfe von selbstüberwachtem Lernen auf großen Datensätzen trainiert wurde und als vielseitige Basis konzipiert ist, die für eine breite Palette von Aufgaben in verschiedenen Domänen fein-getuned oder angepasst werden kann. Synonym: Basis-ML-Modell
Frontier-KI	Ein universell einsetzbares KI-basiertes System, das die Fähigkeiten der derzeit fortschrittlichsten KI-Systeme übertrifft.
Fuzzy-Logik	Eine Art von Logik, die auf dem Konzept der teilweisen Wahrheit basiert und durch Sicherheitsfaktoren zwischen 0 und 1 dargestellt wird.
Gegnerischer Angriff (*adversarial attack*)	Der absichtliche Einsatz von gegnerischen Beispielen, um ein ML-Modell fehlschlagen zu lassen.
Genauigkeit (*accuracy*)	Funktionale Leistungsmetrik von ML zur Evaluierung eines Klassifikators, die den Prozentsatz aller richtig vorher-gesagten Klassifikationen misst. (nach [ISO/IEC TR 29119-11])
Generative KI	Eine Art von KI, die neue Inhalte erstellt, indem sie Muster aus vorhandenen Daten lernt.
Geschichtete Stichprobenziehung (*stratified sampling*)	Eine Methode, um zu überprüfen, ob eine Stichprobe verschiedene Teilpopulationen innerhalb der Gesamtpopulation proportional repräsentiert.
Gewicht (*weight*)	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	Ein anwendungsspezifischer integrierter Schaltkreis 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 (*ground truth*)	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.
Hyperparameter-Tuning	Die Bestimmung der optimalen Hyperparameter des Modells oder des Algorithmus auf der Grundlage bestimmter Ziele.

© International Software Testing Qualifications Board

Begriff (**Fett:** KI-spezifischer Begriff)	**Definition**
Intelligenter Agent	Ein autonomes Programm, das seine Aktivitäten anhand von Beobachtungen und Aktionen auf das Erreichen von Zielen ausrichtet.
Inter-Annotator Übereinstimmung (*inter-annotator agreement*)	Der Grad der Übereinstimmung oder Ähnlichkeit zwischen den Annotationen verschiedener Annotatoren zu denselben Daten. (ISO/IEC TS 12791)
KI-als-Dienst (*AI as a Service*)	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-Komponente	Eine Komponente, die KI-Funktionalität bietet.
KI-Modell	Ein Computerprogramm, das KI implementiert.
Klassifikation	Eine ML-Funktion, welche die Ausgabeklasse für eine gegebene Eingabe vorhersagt. (nach [ISO/IEC TR 29119-11])
Klassifikator	Ein ML-Modell für die Klassifikation. Synonym: Klassifikationsmodell
Konfusionsmatrix	Ein Verfahren zur Zusammenfassung der funktionalen Leistung von ML eines Klassifikationsalgorithmus.
Konzeptdrift	Eine im Laufe der Zeit eintretende Veränderung der wahrgenommenen funktionalen Leistung eines ML-Modells, verursacht durch Veränderungen der Benutzererwartungen, des Benutzerverhaltens und der Produktivumgebung.
Künstliche Intelligenz (*artificial intelligence*)	Die Fähigkeit eines technischen Systems, Wissen und Fähigkeiten zu erwerben, zu verarbeiten, zu erzeugen und anzuwenden. (nach [ISO/IEC TR 29119-11])
Large Language-Modell	Ein generatives Text-zu-Text-KI-System, das auf sehr großen Sprachdatensammlungen trainiert wurde.
Lineare Regression	Ein statistisches Verfahren, das die Beziehung zwischen Variablen modelliert, indem eine lineare Gleichung an die beobachteten Daten angepasst wird, wenn die Zielvariable numerisch ist.
Maschinelles Lernen (*machine learning*)	Der Prozess, bei dem mithilfe von Berechnungstechniken Systeme in die Lage versetzt werden, aus Daten oder Erfahrungen zu lernen. (nach [ISO/IEC TR 29119-11])
Merkmal (*feature*)	Ein einzeln messbares Attribut der Eingabedaten, das von einem ML-Algorithmus für das Training und von einem ML-Modell für die Vorhersage verwendet wird.

© International Software Testing Qualifications Board

Begriff (**Fett:** KI-spezifischer Begriff)	**Definition**
Merkmalsermittlung (*feature engineering*)	Aktivität zur Identifizierung der Attribute in den Rohdaten, welche die zugrundeliegenden Beziehungen, die im ML-Modell erscheinen sollen, am besten repräsentieren, um diese in den Trainingsdaten zu verwenden. (nach [ISO/IEC TR 29119-11])
Merkmalstest (*feature testing*)	Eine Testart zur Bestimmung, ob ein Trainingsdatensatz für ein KI-Modell einen geeigneten Satz von Merkmalen enthält.
ML-Algorithmus	Ein Algorithmus, der zur Erstellung eines ML-Modells aus einem Trainingsdatensatz verwendet wird.
ML-Entwicklungs-Framework	Eine Software-Plattform mit Werkzeugen und Bibliotheken zur Erstellung, zum Training und zur Bereitstellung von ML-Modellen.
ML-Funktion	Von einem ML-Modell implementierte Funktionen, wie Klassifikation, Regression oder Clusterbildung.
ML-Regression	Eine Art von ML-Funktion, die für eine bestimmte Eingabe einen numerischen oder kontinuierlichen Ausgabewert liefert. (nach [ISO/IEC TR 29119-11])
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.
Modell-Konfidenzwertanalyse (*model confidence score analysis*)	Ein Verfahren, mit dem Datenpunkte aus dem Training identifiziert werden, die niedrige Konfidenzwerte aufweisen und damit selbst als möglicherweise falsch gekennzeichnet gelten.
Modell-Verlustanalyse (*model loss analysis*)	Ein Verfahren, mit dem Datenpunkte mit hohen Verlustwerten während des Trainings identifiziert werden, die damit selbst als möglicherweise falsch gekennzeichnet gelten.
Multimodales Modell	Ein ML-Modell, das für die Verarbeitung mehrerer Datenmodalitäten wie Text, Bilder, Audio und Video ausgelegt ist.
Multiple Annotation	Ein Ansatz, bei dem von mehreren Annotatoren gekennzeichnete Datenpunkte miteinander verglichen werden.
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.

© International Software Testing Qualifications Board

Begriff (**Fett:** KI-spezifischer Begriff)	**Definition**
Neuronales Netz	Ein Netz 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-deterministisch	Eine Eigenschaft eines Systems oder Prozesses, bei der ein Ergebnis nicht eindeutig durch seine Anfangsbedingungen bestimmt wird.
Perzeptron	Ein neuronales Netz mit lediglich einer Schicht und einem Neuron.
Präzision (*precision*)	Eine funktionale Leistungsmetrik von ML zur Evaluierung eines Klassifikators; misst den Anteil der korrekten Ergebnisse unter den als positiv vorhergesagten Ergebnissen. (nach [ISO/IEC TR 29119-11])
Probabilistisch	Verhalten, das in Form von Wahrscheinlichkeiten beschrieben wird und dessen Ergebnisse ungewiss sind und eher durch Plausibilität als durch Sicherheit beschrieben werden.
Promptübergreifender Einschleusungsangriff (*cross-prompt injection attack*)	Ein Angriff, bei dem bösartige Anweisungen in einem Prompt oder Kontextsegment das Verhalten eines KI-Modells in einem anderen Prompt, Dialogschritt oder Kontextsegment stören.
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 *(noise)*	Eine Störung oder Verfälschung von Daten.
Retrieval-augmentierte Generierung	Ein ML-Verfahren, bei dem ein GenAI-System seine Ausgabe dynamisch verbessert, indem es relevante externe Informationen abruft, um sein internes Wissen zu ergänzen.
Richtig Negativ (RN) (*true negative, TN*)	Eine Vorhersage, bei der das Modell die negative Klasse korrekt vorhersagt.
Richtig Positiv (RP) (*true positive, TP*)	Eine Vorhersage, bei der das Modell die positive Klasse korrekt vorhersagt.
Rückgekoppeltes neuronales Netz (*recurrent neural network*)	Eine Art Deep-Learning-Modell, das für die Verarbeitung sequenzieller Daten entworfen ist, was ihm ermöglicht, Muster und Abhängigkeiten im Zeitverlauf zu erkennen.
Schwache KI (*narrow AI*)	KI, die sich auf eine einzelne, klar definierte Aufgabe konzentriert, um ein spezifisches Problem zu lösen. (nach [ISO/IEC TR 29119-11]) Synonym: enge KI

© International Software Testing Qualifications Board

Begriff (**Fett:** KI-spezifischer Begriff)	**Definition**
Selbstlernendes System (*self-learning system*)	Ein adaptives System, das sein Verhalten auf der Grundlage von Lernen durch Versuch und Irrtum ändert. (ISO/IEC TR 29119-11)
Sensible Attribute (*sensitive attributes*)	Eigenschaften, die rechtlich oder ethisch geschützte Gruppen und Personen definieren und kontrolliert werden müssen, um Diskriminierung zu verhindern.
Sensitivität (*recall*)	Eine funktionale Leistungsmetrik von ML zur Evaluierung eines Klassifikators; misst den Anteil der korrekt vorhergesagten Ergebnisse unter den tatsächlich positiven Fällen. (nach [ISO/IEC TR 29119-11]) Synonym: Empfindlichkeit
Stimmungsanalyse (*sentiment analysis*)	Der Einsatz von natürlicher Sprachverarbeitung und des maschinellen Lernens zur Identifizierung und Klassifizierung des in einem Text zum Ausdruck gebrachten emotionalen Tons.
Suchalgorithmus	Ein Algorithmus, der systematisch eine Teilmenge aller möglichen Zustände oder Strukturen besucht, bis der Zielzustand oder die Zielstruktur erreicht ist. (nach [ISO/IEC TR 29119-11])
Super-KI	Ein Art künstlicher Intelligenz System, welche die menschlichen Fähigkeiten weit übertrifft.
Support-Vektor Maschine	Ein überwachter ML-Algorithmus, der die optimale Hyperebene zwischen Datenpunkten für die Klassifikation oder ML-Regression ermittelt.
Technologische Singularität	Ein Punkt in der Zukunft, an dem der technologische Fortschritt nicht mehr vom Menschen kontrolliert werden kann. (nach [ISO/IEC TR 29119-11])
Temperatur	Eine Einstellung, die die Zufälligkeit der GenAI-Ausgaben steuert, wobei niedrigere Werte vorhersehbarere Ergebnisse und höhere Werte kreativere Ergebnisse erzeugen.
Testorakelproblem	Die Herausforderung, festzustellen, ob ein Test bei einer bestimmten Anzahl von Testeingaben und einem bestimmten Zustand bestanden oder fehlgeschlagen ist.
Tiefes neuronales Netz (*deep neural network*)	Ein neuronales Netz, das aus mehreren Schichten von Neuronen besteht. Synonym: Mehrschichtiges Perzeptron
Trainingsdatensatz	Ein Datensatz, der zum Trainieren eines ML-Modells verwendet wird.
Transformer	Eine Architektur für neuronale Netze, die sequenzielle Daten verarbeitet, um weitreichende Abhängigkeiten zu erfassen, und NLP-Aufgaben, Computer Vision sowie multimodale Anwendungen unterstützt.

© International Software Testing Qualifications Board

Begriff (**Fett:** KI-spezifischer Begriff)	**Definition**
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. (nach [ISO/IEC TR 29119-11])
Überwachtes Lernen	Ein Ansatz zum Trainieren eines ML-Modells unter Verwendung eines gekennzeichneten Datensatzes.
Unteranpassung	Die Erstellung eines ML-Modells, das den zugrundeliegenden Trend des Trainingsdatensatzes nicht widerspiegelt, was zu einem Modell führt, das ungenaue Vorhersagen macht. (nach [ISO/IEC TR 29119-11])
Unüberwachtes Lernen	Ein Ansatz zum Trainieren eines ML-Modells unter Verwendung eines nicht gekennzeichneten Datensatzes.
Validierungsdatensatz	Ein Datensatz, der zur Evaluierung eines trainierten ML-Modells für das Modell-Tuning verwendet wird.
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. (nach [ISO/IEC 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 anhand eines großen, allgemein einsetzbaren Datensatzes trainiert wurde und für bestimmte Aufgaben wiederverwendet oder durch Fein-Tuning angepasst werden kann.

© International Software Testing Qualifications Board

10 Referenzen

10.1 Normen und Standards

- [ISO/IEC/IEEE 12207] (2017) Systems and software engineering — Software life cycle processes
- [ISO/IEC 2382] (2015) Information technology — Vocabulary
- [ISO/IEC 22989] (2022) Information technology — Artificial intelligence — Artificial intelligence concepts and terminology
- [ISO/IEC TR 24027] (2021) Information technology — Artificial intelligence (AI) — Bias in AI systems and AI aided decision making
- [ISO/IEC 25010] (2023) Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) - Product quality model
- [ISO/IEC 25059] (2023) Software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality model for AI systems
- [ISO 26262-6] (2018) Road vehicles — Functional safety — Part 6: Product development at the software level
- [ISO/IEC TR 29119-11] (2020) Software and systems engineering — Software testing — Part 11: Guidelines on the testing of AI-based systems
- [ISO/IEC TS 42119-2] (2025) Artificial intelligence — Testing of AI – Part 2: Overview of testing AI systems
- [ISO/IEC TS 42119-3] (in Entwicklung) Artificial intelligence — Testing of AI – Part 3: Verification and validation analysis of AI systems
- [ISO/IEC TS 42119-7] (in Entwicklung) Artificial intelligence — Testing of AI – Part 7: Red teaming
- [ISO/IEC TS 42119-8] (in Entwicklung) Artificial intelligence — Testing of AI – Part 8: Quality assessment of prompt-based text-to-text systems that utilize generative AI

10.2 ISTQB®-Dokumente

[CTFL] ISTQB® Certified Tester Foundation Level v4.0.1, https://istqb.org/wp-content/uploads/2024/11/ISTQB_CTFL_Syllabus_v4.0.1.pdf (Zugriff 12.10.2026).

[CT-GenAI] ISTQB® Certified Tester – Testing with Generative AI (CT-GenAI), https://istqb.org/certifications/gen-ai/ (Zugriff 12.10.2026)

10.3 Glossar-Referenzen

Referenz für die im Lehrplan genutzten Begriffe:

[ISTQB® Glossary] https://glossary.istqb.org/

© International Software Testing Qualifications Board

10.4 Bücher, Artikel und Web-Seiten

[COV_REF] An Overview of Structural Coverage Metrics for Testing Neural Networks, Usman et al, Aug 2022, arXiv:2208.03407 (Zugriff 29.07.2025)

[DATA_DOC] Gebru, T., Morgenstern, J., Vecchione, B., et al. (2021). Datasheets for Datasets. Communications of the ACM, 64(12), 86-92, https://dl.acm.org/doi/pdf/10.1145/3502158 (Zugriff 07.10.2025)

[EU AI Act] EU Artificial Intelligence Act, July 2024, https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L_202401689 (Zugriff 30.07.2025)

[MODEL_DOC] Mitchell, M., Wu, S., Varma, R., et al. (2019). Model Cards for Model Reporting. Proceedings of the Conference on Fairness, Accountability, and Transparency, https://dl.acm.org/doi/10.1145/3287560.3287596 (Zugriff 07.10.2025)

[OECD AI] OECD Recommendation of the Council on Artificial Intelligence, May 2024, Organisation for Economic Co-operation and Development (OECD), https://legalinstruments.oecd.org/en/instruments/oecd-legal-0449 (Zugriff 07.10.2025)

[UN Gov AI] Governing AI for Humanity: Final Report, United Nations, September 2024. https://www.un.org/sites/un2.un.org/files/governing_ai_for_humanity_final_report_en.pdf (Zugriff 29.07.2025)

[STATS] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani: An Introduction to Statistical Learning - With Applications in R (2nd ed.), Springer Nature, NY, 2021. https://www.statlearning.com/ (Zugriff 10.06.2026)

© International Software Testing Qualifications Board

11 Markenzeichen

- ISTQB® ist ein registriertes Markenzeichen des International Software Testing Qualifications Board.

© International Software Testing Qualifications Board

12 Anhang A – Lernziele/Kognitive Wissensstufen

Die spezifischen Lernziele, die für diesen Lehrplan gelten, sind am Anfang jedes Kapitels aufgeführt. Jedes Thema im Lehrplan wird anhand des dafür festgelegten Lernziels geprüft.

Die Lernziele beginnen mit einem Aktionsverb, das der kognitiven Wissensstufe entspricht, wie unten aufgeführt.

Stufe 1: Erinnern (K1)

Der Kandidat erinnert sich an einen Begriff oder ein Konzept, erkennt ihn/es wieder und ruft ihn/es ab.

Aktionsverben: Erinnern, erkennen

Beispiele
Erinnern Sie sich an die Konzepte der Testpyramide.
Erkennen Sie die typischen Ziele von Testen.

Stufe 2: Verstehen (K2)

Der Kandidat kann die Gründe oder Erklärungen für Aussagen zum Thema auswählen und das Testkonzept zusammenfassen, vergleichen, klassifizieren und Beispiele dafür nennen.

Aktionsverben: Klassifizieren, vergleichen, unterscheiden, erklären, Beispiele nennen, interpretieren, zusammenfassen

Beispiele	Anmerkungen
Klassifizieren Sie Testwerkzeuge nach ihrem Zweck und den Testaktivitäten, die sie unterstützen.	
Vergleichen Sie die verschiedenen Teststufen.	Kann verwendet werden, um nach Ähnlichkeiten, Unterschieden oder beidem zu suchen.
Differenzieren Sie zwischen Testen und Debugging.	Sucht nach Unterschieden zwischen Konzepten.
Unterscheiden Sie zwischen Projektrisiko und Produktrisiko.	Ermöglicht die separate Klassifizierung von zwei (oder mehr) Konzepten.
Erklären Sie die Auswirkungen des Kontexts auf den Testprozess.	
Geben Sie Beispiele dafür, warum Testen notwendig ist.	
Leiten Sie anhand eines gegebenen Profils von Fehlerwirkungen die Grundursache von Fehlerzuständen ab.	
Fassen Sie die Aktivitäten des Reviewprozesses für Arbeitsergebnisse zusammen.	

© International Software Testing Qualifications Board

Stufe 3: Anwenden (K3)

Der Kandidat kann ein Verfahren ausführen, wenn er mit einer vertrauten Aufgabe konfrontiert wird, oder das richtige Verfahren auswählen und es auf einen gegebenen Kontext anwenden.

Aktionsverben: Anwenden, implementieren, vorbereiten, nutzen

Beispiele	Anmerkungen
Wenden Sie die Grenzwertanalyse an, um aus vorgegebenen Anforderungen Testfälle abzuleiten.	Sollte sich auf ein Verfahren / eine Technik / einen Prozess usw. beziehen.
Implementieren Sie Methoden zur Metrik-Erfassung, um technische Anforderungen und Anforderungen des Managements zu unterstützen.	
Bereiten Sie Tests zur Installierbarkeit für mobile Apps vor.	
Nutzen Sie Verfolgbarkeit, um den Testfortschritt auf Vollständigkeit und Übereinstimmung mit den Testzielen, der Teststrategie und dem Testkonzept zu überwachen.	Könnte in einem LO verwendet werden, das von Kandidaten die Anwendung einer Technik oder eines Verfahrens erwartet. Ähnlich wie „anwenden".

Stufe 4: Analysieren (K4)

Der Kandidat kann Informationen, die sich auf ein Verfahren beziehen, zum besseren Verständnis in ihre Bestandteile zerlegen und kann zwischen Fakten und Schlussfolgerungen unterscheiden. Eine typische Anwendung ist die Analyse eines Dokuments, einer Software oder einer Projektsituation und das Vorschlagen geeigneter Maßnahmen, um ein Problem zu lösen oder eine Aufgabe zu erfüllen.

Aktionsverben: Analysieren, dekonstruieren, skizzieren, Prioritäten setzen, auswählen

Beispiele	Anmerkungen
Analysieren Sie eine gegebene Projektsituation, um festzustellen, welche Black-Box-Testverfahren oder erfahrungsbasierten Testverfahren angewendet werden sollten, um bestimmte Ziele zu erreichen.	Nur in Verbindung mit einem messbaren Ziel der Analyse prüfbar. Sollte von der Form 'Analysiere xxxx um yyyy' (oder ähnlich) sein.
Priorisieren Sie Testfälle in einer bestimmten Testsuite für die Ausführung auf Grundlage der damit verbundenen Produktrisiken.	
Wählen Sie die geeigneten Teststufen und Testarten aus, um eine bestimmte Menge von Anforderungen zu überprüfen.	Wird benötigt, wenn die Auswahl eine Analyse erfordert.

Referenz (für die kognitiven Wissensstufen der Lernziele):

Anderson, L. W., Krathwohl, D. R. (Hrsg.) (2001). A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon.

© International Software Testing Qualifications Board

13 Anhang B – Verfolgbarkeitsmatrix für geschäftliche Nutzen mit den Lernzielen

In diesem Abschnitt wird die Verfolgbarkeit zwischen den geschäftlichen Nutzen und den Lernzielen von „Certified Tester – KI Test 2.0 D“ aufgeführt. Der erste Teil der Tabelle zeigt die Anzahl der Lernziele pro geschäftlichem Nutzen, der zweite Teil zeigt, welche Lernziele mit jedem Geschäftsergebnis verbunden sind.

Geschäftlicher Nutzen KI Test: Personen, welche die Zertifizierung KI Test erworben haben, können Folgendes		BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
BO1	Verstehen des aktuellen Stands der KI, einschließlich generativer KI.	8							
BO2	Sammeln von Erfahrung mit der Implementierung und dem Testen von Modellen des maschinellen Lernens (ML).		10						
BO3	Verstehen der Funktionsweise und des Testens einfacher neuronaler Netze.			2					
BO4	Verstehen der spezifischen Qualitätsmerkmale von KI gemäß ISO/IEC 25059.				3				
BO5	Berechnen und Interpretieren funktionaler Leistungsmetriken von ML.					1			
BO6	Erkennen von Umfang und Bedeutung der beiden Teststufen, die für das Testen von ML-Systemen spezifisch sind.						3		
BO7	Beitragen zur Entwicklung einer effektiven Teststrategie für ML-Systeme.							5	
BO8	Entwerfen und Durchführen von Testfällen für ML-Systeme.								14

© International Software Testing Qualifications Board

Geschäftlicher Nutzen KI Test: Personen, welche die Zertifizierung KI Test erworben haben, können Folgendes			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
LO-Nr	Lernziel: Die Lernenden können …	K-Ebene								
1	Einführung in die Künstliche Intelligenz – 120 Minuten									
1.1	Einführung in die KI									
AI-1.1.1	… zwischen KI-basierten Systemen und konventionellen Systemen unterscheiden.	K2	X							
AI-1.1.2	… schwache KI, allgemeine KI und Super-KI voneinander unterscheiden.	K2	X							
AI-1.1.3	… die verschiedenen Arten von KI-Technologien erklären.	K2	X							
AI-1.1.4	… generative KI erklären.	K2	X							
AI-1.1.5	… die verfügbaren Hardware-Optionen zur Implementierung von Systemen für maschinelles Lernen vergleichen.	K2	X							
AI-1.1.6	… die Optionen für die Entwicklung und das Hosting von KI-Modellen vergleichen.	K2	X							
AI-1.1.7	… die von ML-Entwicklungs-Frameworks bereitgestellten Funktionen zusammenfassen.	K2	X							
AI-1.1.8	… erklären, wie sich Regulierungen, Normen und Standards auf die Entwicklung und das Testen von KI-basierten Systemen auswirken.	K2	X							

© International Software Testing Qualifications Board

Geschäftlicher Nutzen KI Test: Personen, welche die Zertifizierung KI Test erworben haben, können Folgendes			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
2	Qualitätsmerkmale für KI-basierte Systeme – 45 Minuten									
2.1	Qualitätsmerkmale für KI-basierte Systeme									
AI-2.1.1	… das Verhalten von KI-basierten Systemen bezüglich der Qualitätsmerkmale gemäß ISO/IEC 25059 klassifizieren.	K2				X				
AI-2.1.2	… die speziellen Überlegungen erklären, die beim Einsatz von KI in sicherheitsrelevanten Systemen zu berücksichtigen sind.	K2				X				
2.2	Akzeptanzkriterien für KI-basierte Systeme									
AI-2.2.1	… Beispiele für Akzeptanzkriterien für KI-basierte Systeme nennen.	K2				X				
3	Maschinelles Lernen (ML) – 375 Minuten									
3.1	Einführung in das maschinelle Lernen									
AI-3.1.1	… zwischen den verschiedenen Formen des maschinellen Lernens unterscheiden.	K2		X						
AI-3.1.2	… den Workflow zur Erstellung eines ML-Systems zusammenfassen.	K2		X						
AI-3.1.4	… die Verwendung von vortrainierten Modellen, Fein-Tuning und Retrieval-augmentierter Generierung zusammenfassen.	K2		X						
3.2	Daten für maschinelles Lernen									
AI-3.2.1	… die Aktivitäten im Zusammenhang mit der Datenvorbereitung erklären.	K2		X						

© International Software Testing Qualifications Board

Geschäftlicher Nutzen KI Test: Personen, welche die Zertifizierung KI Test erworben haben, können Folgendes			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
AI-3.2.3	… die Verwendung von Trainings-, Validierungs- und Testdatensätzen bei der Entwicklung eines ML-Modells voneinander abgrenzen.	K2		X						
3.3	Funktionale Leistungsmetriken von ML zur Klassifikation									
AI-3.3.1	… gängige funktionale Leistungsmetriken von ML anhand eines gegebenen Satzes von Konfusionsmatrix-Daten berechnen.	K3					X			
3.4	Neuronale Netze									
AI-3.4.1	… den Aufbau und die Funktionsweise eines tiefen neuronalen Netzes erklären.	K2			X					
AI-3.4.3	… die verschiedenen Überdeckungsmaße für neuronale Netze beschreiben.	K2			X					
4	Testen KI-basierter Systeme – 195 Minuten									
4.1	Einführung in das Testen KI-basierter Systeme									
AI-4.1.1	… die Testbarkeit fixierter und adaptiver KI-basierter Systeme vergleichen.	K2		X						
AI-4.1.2	… erklären, warum beim Testen KI-basierter Systeme häufig ein statistischer Ansatz erforderlich ist.	K2		X						
AI-4.1.3	… die Herausforderungen und Lösungen im Zusammenhang mit Testorakeln für KI-basierte Systeme erklären.	K2		X						

© International Software Testing Qualifications Board

Geschäftlicher Nutzen KI Test: Personen, welche die Zertifizierung KI Test erworben haben, können Folgendes			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
4.2	Testen von generativer KI und Large Language-Modellen									
AI-4.2.1	… erklären, wie generative KI getestet werden kann.	K2		X						
AI-4.2.2	… Red-Teaming für GenAI-Systeme umsetzen.	K3		X						
4.3	Teststufen und ML-Systeme									
AI-4.3.1	… die Teststufen bei der Entwicklung von ML-Systemen zusammenfassen.	K2							X	
AI-4.3.2	… erklären, wie der risikobasierte Test bei ML-Systemen angewendet wird.	K2							X	
5	Eingabedatentest für ML-Systeme – 180 Minuten									
5.1	Eingabedatentest für ML-Systeme									
AI-5.1.1	… Beispiele für Testansätze nennen, die zur Risikominderung von Eingabedaten für ML-Systeme Verwendung finden.	K2						X	X	
AI-5.1.2	… das Testen auf Verzerrungen erklären.	K2								X
AI-5.1.3	… die verschiedenen Arten von Daten-Pipelinetests zusammenfassen.	K2								X
AI-5.1.4	… den Datenrepräsentativitätstest erklären.	K2								X
AI-5.1.5	… den Datensatzeinschränkungstest anwenden.	K3								X

© International Software Testing Qualifications Board

Geschäftlicher Nutzen KI Test: Personen, welche die Zertifizierung KI Test erworben haben, können Folgendes			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
AI-5.1.6	… den Kennzeichen-Korrektheitstest erklären.	K2								X
6	Modelltest für ML-Systeme – 225 Minuten									
6.1	Modelltest für ML-Systeme									
AI-6.1.1	… Beispiele für Testansätze zur Risikominderung bei ML-Modellen nennen.	K2						X	X	
AI-6.1.2	… den Zweck und den Schwerpunkt von Reviews der Dokumentation von ML-Modellen erklären.	K2								X
AI-6.1.3	… erklären, wie der Test der funktionalen Leistung für probabilistische ML-Systeme durchgeführt wird.	K2								X
AI-6.1.4	… den gegnerischen Test von ML-Systemen zusammenfassen.	K2								X
AI-6.1.5	… den metamorphen Test verwenden, um Testfälle für ein bestimmtes Szenario abzuleiten.	K3								X
AI-6.1.7	… erklären, wie Drift-Tests bei ML-Systemen im Produktivbetrieb eingesetzt werden.	K2								X
AI-6.1.8	… erklären, wie Überanpassung und Unteranpassung durch Testen erkannt werden.	K2								X
AI-6.1.9	… erklären, wie A/B-Tests im Zusammenhang mit ML-Systemen eingesetzt werden.	K2								X

© International Software Testing Qualifications Board

Geschäftlicher Nutzen KI Test: Personen, welche die Zertifizierung KI Test erworben haben, können Folgendes			BO1	BO2	BO3	BO4	BO5	BO6	BO7	BO8
AI-6.1.10	… erklären, wie Back-to-Back-Tests im Zusammenhang mit ML-Systemen eingesetzt werden.	K2								X
7	ML-Entwicklungstest – 30 Minuten									
7.1	ML-Entwicklungstest									
AI-7.1.1	… Beispiele für Ansätze nennen, die zur Verminderung von Risiken bei der Entwicklung von ML-Systemen verwendet werden.	K2						X	X	
AI-7.1.2	… die verschiedenen Formen des Bereitstellungstests für ML-Systeme erklären.	K2								X

© International Software Testing Qualifications Board

14 Anhang C – Versionshinweise

Der ISTQB® CT-AI v2.0 ist eine umfassende Aktualisierung und Überarbeitung von v1.0, denn die sich rasch weiterentwickelnde KI-Technologie erforderte ein neues Major Release. Der Schwerpunkt von v2.0 liegt eindeutig auf dem Testen KI-basierter Systeme. Aufgrund der Veröffentlichung des ISTQB® Certified Tester „Testen mit Generativer KI“ ist das Kapitel zum Testen mit KI vollständig entfallen.

Mit diesem Major Release wurden folgende Änderungen vorgenommen:

- Verkürzte Einführung in die KI allgemein
- Einbeziehung generativer KI und des Tests von generativer KI
- Zusammenfassung der Qualitätsmerkmale von KI und der damit verbundenen Herausforderungen
- Geringerer Fokus auf funktionale Leistungsmetriken von ML
- Verfeinerte Teststufen: Eingabedatentest und ML-Modelltest
- Verfeinerte Testarten
- Entfall des Tests mit KI
- Entfall der Inhalte zu Testumgebungen für das Testen KI-basierter Systeme
- Die Anforderung an die Mindestschulungsdauer wurde von 4 Tagen auf 3 Tage verkürzt

In der deutschsprachigen Version 2.0 D wurden im Vergleich zu Version 1.0 D folgende Umbenennungen bzw. Änderungen vorgenommen[3]:

- *AI as a Service* von „KI-als-Dienst“ in „KI als Dienst“
- *application programming interface* (API) von „Schnittstelle zur Anwendungsprogrammierung“ in „Anwendungsprogrammierschnittstelle“
- *(data) augmentation* von „(Daten-) Anreicherung“ in „(Daten-) Erweiterung“
- (*) *Bayesian model* von „Bayessches Modell“ in „Bayes’sches Modell“
- *bias* (eines Neurons) von Bias in Verschiebungswert (*bias*)
- (*) *clustering* von „clustering“ in „Clusterbildung“
- (*) *data preprocessing* unverändert „Datenvorverarbeitung“
- *data transformation* von „Datenumwandlung (*data transformation*)“ in „Datentransformation“
- (*) exploratory data analysis unverändert „Explorative Datenanalyse”
- (*) *feature engineering* unverändert „Merkmalsermittlung“
- (*) *fuzzy logic* unverändert „Fuzzy Logik“
- (*) *concept drift* unverändert „Konzeptdrift“
- *operation* von „in Produktion“ in „im Produktivbetrieb“
- *operational environment* von „Produktionsumgebung“ in „Produktivumgebung“
- (*) *pretrained model* unverändert „Vortrainiertes Modell“
- (*) *probabilistic (system)* von „Probabilistisches System“ in „Probabilistisch“
- (*) *reinforcement learning* unverändert „Bestärkendes Lernen“
- (*) *super AI* unverändert „Super-KI“
- (*) *supervised learning* unverändert „Überwachtes Lernen“
- (*) *unsupervised learning* unverändert „Unüberwachtes Lernen“

[3] (*) = Definition in CT-AI 2.0 geändert, s. Kapitel 9. Die Schlüsselbegriffe sind gelistet in [ISTQB® Glossary].

© International Software Testing Qualifications Board

15 Index

Alle Test-Begriffe und spezifischen Begriffe sind im ISTQB®-Glossar definiert (https://glossary.istqb.org/).

A/B-Test 29, 60
adaptives KI-basiertes System 39
Aktivierungsfunktion 36
allgemeine KI 15
Analyse auf mittelbare Diskriminierung 48
Angriff 31, 42, 59
Anwendungsprogrammierschnittstelle 19
API *Siehe* Anwendungsprogrammierschnittstelle
API-Test 44, 63, 65
Assoziation 27
Ausgangstestfall 58
Back-to-Back-Test 49, 52, 61
Benchmark-Suite 42
Bereitstellung 18, 54
bestärkendes Lernen 15, 27
Blue-Teaming 43
Clusterbildung 27
Datenbeschaffung 28, 31
Datendrift 59
Datenhandhabung 18
Daten-Pipelinetest 49
Datenrepräsentativitätstest 50
Datensatzeinschränkungstest 51
Datenvorbereitung 28, 31
Datenvorverarbeitung 28, 31
Deep Learning 15
Drift 29
Drift-Test 59
Dynamisches Testen 48
EDA *Siehe* Explorative Datenanalyse
Eingabedatentest 44, 47
Eingreifbarkeit 22
Erklärbarkeit 14, 23
Erweiterung (der Daten) 31
Evaluierung 18
Explorative Datenanalyse 32, 48
explorativer Test 43
F1-Wert 34
faltendes neuronales Netz 15
Fehlerangriff 42
Fein-Tuning 18, 30
fixiertes KI-basiertes System 39
Folgetestfall 58
Foundation-Modell 16
Frontier-KI 14
Funktionale Anpassbarkeit 22
Funktionale KI-Korrektheit 22
funktionale Leistung von ML 30, 33, 44, 56, 59
funktionale Leistungsmetriken von ML 33, 85
gegnerischer Angriff 54, 57
Gegnerischer Test 57
GenAI *Siehe* generative KI
Genauigkeit 34
generative KI 16, 42
Geräteübergreifender Test 65
Grenzwertanalyse 43
Grundwahrheit 39, 59
Hold-out-Datensatz *Siehe* Testdatensatz
Installierbarkeitstest 64
Inter-Annotator Übereinstimmung 52
Kanarien-Test 64

© International Software Testing Qualifications Board

Kennzeichen-Korrektheitstest 48, 51
k-fache Kreuzvalidierung 33
KI als Dienst 17
KI-basiertes System 14, 39
KI-Governance 20
KI-Robustheit 22
KI-Verordnung der EU 19, 23, 43
Klassifikation 18, 27, 33
k-Multisektions-Neuronenüberdeckung 37
Konfusionsmatrix 33
Kontrollierbarkeit durch Benutzer 22
Konzeptdrift 59
künstliche Intelligenz 14
Large Language-Modell 30, 41
LLM *Siehe* Large Language-Modell
maschinelles Lernen 14, 15
Merkmalsermittlung 28, 32
metamorphe Relation 58
Metamorpher Test 58
Minderung gesellschaftlicher und ethischer Risiken 22
ML-Algorithmus 27, 28, 68, 69, 72
ML-Entwicklungs-Framework 18, 28
ML-Entwicklungstest 63
ML-Modell 17, 18, 28, 32, 35
ML-Modelldokumentation 55
ML-Modelltest 44, 54
ML-Regression 18, 27
ML-Workflow 28
Modellbildung 18
Modell-Konfidenzwertanalyse 52
Modellkonvertierungstest 65
Modell-Verlustanalyse 52
Multiple Annotation 52
Neuron 36
neuronales Netz 35
Neuronen-Grenzwertüberdeckung 37
Neuronenüberdeckung 37
Nicht-Determinismus 23
nicht-funktionaler Test 29, 44, 49, 58
Performanz 54
Perzeptron 35
Präzision 34
Promptübergreifender Einschleusungsangriff 43
Qualitätsmaß 41
Qualitätsmerkmal 21
Red-Teaming 42
Regulierung 19, 23
Resampling 33
Retrieval-augmentierte Generierung 31
Review 48, 49
Risikobasierter Test 44
Robustheit 54
Rollback-Test 64
rückgekoppeltes (rekurrentes) neuronales Netz 15
schwache KI 14
Selbstlernendes System 23
Sensitivität 34
Shadow-Test 64
Sicherheit (*safety*) 23, 43
Sicherheit (*security*) 43
starke KI *Siehe* allgemeine KI
Statische Analyse 48
Super-KI 15
Support-Vektor Maschine 15
Testdatensatz 32, 33
Testen auf Überanpassung 59
Testen auf Unteranpassung 59
Testen auf Verzerrung 48
Testorakel 39

© International Software Testing Qualifications Board

Testorakelproblem 39, 41
Teststufen 44
Training 18, 28, 35
Trainingsdatensatz 32
Transformer 15
Transparenz 22, 23
Überanpassung 54, 59
überwachtes Lernen 15, 27
Unteranpassung 60
unüberwachtes Lernen 15, 27
Validierungsdatensatz 32
Verschiebungswert 39
Verschiebungswert (eines Neurons) 36
Verzerrung 48, 54
vortrainiertes Modell 30

© International Software Testing Qualifications Board