

Schulungsunterlagen

ZERTIFIZIERUNG

JANKE, SIMON

Inhalt

1.0	Kapitel: Einführung in die KI.....	1
1.1	KI-Effekt.....	1
1.2	Schwacher-KI, allgemeiner-KI, Super-KI.....	2
1.3	KI und normale Systeme.....	3
1.4	Techniken zur Implementierung v. KI	4
1.5	KI Entwicklungs-Framework	5
1.6	GPU vs. CPU	7
1.7	KI als Dienst (AlaaS)	8
1.8	Risiko bei vortrainierten KI-Modellen	10
1.9	Normen für KI-basierten Systeme	11
2.0	Kapitel: Qualitätsmerkmale KI basierte Systeme	13
2.1	Flexibilität und Anpassbarkeit Merkmale KI basierte Systeme	13
2.2	Beziehung zwischen Autonome und KI basierten Systeme	14
2.3	Managements der Evolution KI basierte Systeme	16
2.4	Ursachen und Arten von Verzerrungen die in KI basierten Systemen auftreten	17
2.5	Diskutieren Sie die ethischen Grundsätze bei der Entwicklung, dem Einsatz und Nutzung KI basierten Systeme beachtet werden	19
2.6	Nebenwirkungen und Belohnung-Hacking, die in KI basierten Systemen auftreten.....	21
2.7	Transparenz, Interpretierbarkeit und Erklärbarkeit	22
2.8	Eigenschaften, die den Einsatz KI basierter Systeme in sicherheitsrelevanten Anwendungen erschweren.....	23
3.0	Kapitel: Maschinelles Lernen (ML) -Überblick	24
3.1	Klassifikation und Regression Teil des überwachten Lernens	24
3.2	Workflow bei Erstellung eines ML-Systems	27
3.3	Geeigneten ML-Ansatzes anhand des Projektszenarios.....	28
3.4	Faktoren die eine Rolle spielen bei der Auswahl von ML-Algo-rithmen.....	29
3.5	Konzepte von Überanpassung und Unteranpassung.....	30
4.0	Kapitel: ML – Daten	32
4.1	Aktivitäten u. Herausforderungen mit Datenvorbereitung.....	32
4.2	Vergleich von Trainings-, Validierungs- und Testdatensätze bei der Entwicklung eines ML-Modells	34
4.3	Typische Qualitätsprobleme bei Datensätze	36
4.4	Erkennen, wie schlechte Datenqualität zu Problemen und dem resultierenden ML-Modell führen kann	38
4.5	Verschiedene Ansätze zur Kennzeichnung von Daten in Datensätze für überwachtes Lernen	39

4.6 Gründe für falsche Kennzeichnung der Daten in Datensätze.....	40
5.0 Kapitel: Funktionale Leistungsmetriken von ML	42
5.1 Berechnung der funktionalen Leistungsmetriken von ML.....	42
5.2 Gegenüber stellen und Vergleichen der Konzepte hinter den funktionalen Leistungsmetriken von ML-Methoden.....	44
5.3 Beschränkungen von funktionalen Leistungsmetriken	45
5.4 Auswählen geeigneter funktionaler Leistungsmetriken von ML und/oder ihrer Werte für ein bestimmtes ML-Modell und Szenario.....	45
5.5 Erläutern Sie die Verwendung von Benchmark-Suiten im Zusammenhang mit ML	47
6.0 Kapitel: ML – Neuronale Netzwerke und Testen	48
6.1 Erläutern der Struktur und Funktion eines neuronalen Netzwerks, einschließlich eines DNN (Deep neural network)	48
6.2 Beschreiben der verschiedenen Überdeckungsmaße für neuronale Netzwerke	51
7.0 Kapitel: Testen KI-basierter Systeme im Überblick	53
7.1 Systemspezifikationen für KI-basierte Systeme	53
7.2 Wie werden KI-basierte Systeme auf jeder Teststufe getestet?	54
7.3 Testdaten die das testen der KI-basierten Systeme erschweren können	56
7.4 Wie sich Automatisierungsverzerrung auf das Testen auswirkt.....	56
7.5 Beschreiben von Dokumentation einer KI-Komponente und verstehen, wie die Dokumentation das Testen eines KI-basierten System unterstützt.....	57
7.6 Notwendigkeit, das trainierte Modell häufig zu testen um einen Konzeptdrift zu vermeiden	59
7.7 Analysieren eines Szenarios, bestimmen der Test Vorgehensweise, die bei der Entwicklung des ML-Systems befolgt werden muss.	60
8.0 Kapitel: Testen KI-spezifischer Qualitätsmerkmale	62
8.1 Herausforderungen beim Testen die durch das Selbstlernen KI-basierter Systeme entstehen	62
8.2 Wie werden autonome KI-basierende-Systeme getestet	63
8.3 KI-basiertes System auf Verzerrungen prüfen.....	64
8.4 Herausforderungen bei probabilistische und nichtdeterministische KI-basierter Systeme.....	66
8.5 Herausforderungen beim Testen, die durch Komplexität KI-basierter Systeme entstehen?.....	67
8.6 Wie können Transparenz, Interpretierbarkeit und Erklärbarkeit KI-basierter Systeme getestet werden	69
8.7 Herausforderungen bei der Erstellung von Testorakeln, die sich aus den spezifischen Merkmalen KI-basierter Systeme ergeben	70

8.8 Geeignete Testziele und Akzeptanzkriterien für die KI-spezifischen Qualitätsmerkmale eines KI-basierten System.....	72
9.0 Kapitel: Methoden u. Verfahren für das Testen KI-basierter Systeme	74
9.1 ML-Systemen dazu beitragen gegnerische Angriffe und Datenver-unreinigungen zu verhindern	74
9.2 Paarweises Testen KI-basierter Systeme	75
9.3 Vergleichendes Testen von KI basierten Systeme	76
9.4 A/B Testen von KI-basierten Systeme	78
9.5 Metamorphen Testen von KI-basierten Systeme	79
9.6 Erfahrungsbasiertes Testen von KI-basierten Systeme.....	80
9.7 Geeignetes Testverfahren für ein bestimmtes Szenario	82
10.0 Kapitel: Testumgebung für KI-basierte Systeme.....	84
10.1 Die wichtigsten Faktoren bei der Unterscheidung von konventionellen Testumgebung und insbesondere bei selbst lernenden oder autonomen Systemen	84
10.2 Vorteile von einer virtuelle Testumgebung für KI-basierte Systeme	85
11.0 Kapitel: Einsatz von KI für Tests	88
11.1 Kategorisieren der bei Softwaretest verwenden KI-Techniken.....	88
11.2 Erläutern wie KI bei der Analyse neuer Fehlerzustände unterstützen kann	91
11.3 Erläutern wie KI bei der Erstellung von Testfällen helfen kann.....	93
11.4 Erklären wie KI bei der Optimierung von Regressionsfällen helfen kann	94
11.5 Erklären wie KI bei den Fehlervorhersagen helfen kann.....	96
11.6 Erklären des Einsatzes von KI beim Testen von Benutzerschnittstellen	100

1.0 Kapitel: Einführung in die KI

K1–K2:

Begriffe/Konzepte

wie KI-Definition, KI-Effekt, Schwacher/Allgemein/Super AI, AI-basierte vs. konventionelle Systeme, AI-Technologien, Pre-trained Models, AlaaS, Hardware, Regulations/Standards verstehen.

1.1 KI-Effekt

Frage:

Wie beeinflusst der sogenannte KI-Effekt die Wahrnehmung und Definition von künstlicher Intelligenz im Laufe der Zeit?

Antwort:

Der sogenannte KI-Effekt beschreibt ein psychologisches und soziales Phänomen, bei dem Technologien, sobald sie alltäglich oder gut verstanden sind, nicht mehr als "künstliche Intelligenz" wahrgenommen werden. Das beeinflusst sowohl die Wahrnehmung als auch die Definition von KI im Laufe der Zeit erheblich.

Auswirkungen auf die Wahrnehmung

1. Verschiebung der Erwartungen

Menschen neigen dazu, KI mit etwas Mystischem oder Übermenschlichem zu verbinden. Sobald eine Technologie diese Erwartungen nicht mehr erfüllt (z. B. weil sie alltäglich wird), verliert sie den "KI-Status".

2. Unterschätzung von Fortschritt

Viele KI-Anwendungen wie **Spracherkennung**, **Bilderkennung**, **Maschinelles Übersetzen** oder **Empfehlungssysteme** gelten heute als selbstverständlich – obwohl sie früher als Paradebeispiele für KI galten.

3. Verzerrte Diskussionen

Die öffentliche Debatte über KI konzentriert sich oft auf Zukunftsvisionen (z. B. AGI oder humanoide Roboter), während reale, gegenwärtige KI-Anwendungen übersehen oder unterschätzt werden.

Auswirkungen auf die Definition von KI

1. Dynamische Definition

Die Definition von KI ist **nicht statisch**, sondern verändert sich mit dem technologischen Fortschritt. Was heute als KI gilt, kann morgen als „normale Software“ gelten.

2. Abgrenzung zu anderen Disziplinen

Viele Teilbereiche wie **maschinelles Lernen**, **Computer Vision** oder **Natural Language Processing** wurden früher unter dem KI-Begriff zusammengefasst, gelten heute aber oft als eigene Disziplinen.

3. Veränderung der Forschungsziele

Der KI-Effekt beeinflusst auch, welche Probleme als „KI-Probleme“ gelten – und damit, welche Forschungsfragen als besonders relevant oder prestigeträchtig angesehen werden.

1.2 Schwacher-KI, allgemeiner-KI, Super-KI

Frage:

Unterscheiden zwischen schwacher KI, allgemeiner KI, und Super-KI?

Antwort:

1. Schwache KI (Weak AI/ Narrow AI)

- **Definition:** KI, die für eine **spezifische Aufgabe** entwickelt wurde und **keine echte Intelligenz** besitzt.
- **Beispiele:**
 - Sprachassistenten wie Siri oder Alexa
 - Empfehlungsalgorithmen bei Netflix oder Amazon
 - Bilderkennung in medizinischen Diagnosen
- **Merkmale:**
 - Kein Bewusstsein oder Verständnis
 - Funktioniert nur in klar definierten Kontexten
 - Übertrifft oft den Menschen in engen Aufgabenbereichen

2. Allgemeine KI (AGI – Artificial General Intelligence)

- **Definition:** Eine KI, die **intellektuell mit dem Menschen vergleichbar** ist – sie kann **lernen, verstehen und Probleme lösen** in beliebigen Bereichen.
- **Merkmale:**
 - Flexibel und anpassungsfähig
 - Kann Wissen aus einem Bereich auf andere übertragen
 - Hat ein gewisses Maß an „Verständnis“ und „Urteilsvermögen“
- **Status: Noch nicht erreicht** – Gegenstand intensiver Forschung und Debatte

3. Super-KI (ASI – Artificial Superintelligence)

- **Definition:** Eine hypothetische KI, die **alle menschlichen Fähigkeiten übertrifft**, sowohl in Intelligenz als auch in Kreativität, sozialem Verständnis und Problemlösung.
 - **Merkmale:**
 - Könnte wissenschaftliche Durchbrüche beschleunigen
 - Potenziell unkontrollierbar oder schwer vorhersehbar
 - Wird oft in ethischen und sicherheitsbezogenen Diskussionen thematisiert
 - **Status: Rein spekulativ** – es gibt keine existierende Super-KI
-

Vergleichstabelle

Merkmal	Schwache KI	Allgemeine KI (AGI)	Super-KI (ASI)
Aufgabenbereich	Eng begrenzt	Universell	Übermenschlich
Lernfähigkeit	Eingeschränkt	Hoch	Extrem hoch
Existenz	Bereits vorhanden	Noch nicht erreicht	Hypothetisch
Beispiel	Chatbots, Google Maps	Menschliche Intelligenz	Science-Fiction (z. B. HAL 9000)

1.3 KI und normale Systeme

Frage:

Unterscheiden zwischen KI-basierten Systemen und herkömmlichen Systemen.

Antwort:

KI-basierte Systeme

Merkmal	Beschreibung
Funktionsweise	Lernen aus Daten (z. B. durch maschinelles Lernen)
Flexibilität	Hoch – kann sich an neue Situationen anpassen
Beispielhafte Technologien	Sprachverarbeitung, Bilderkennung, autonome Fahrzeuge
Entscheidungsfindung	Basierend auf Wahrscheinlichkeiten, Mustern und Trainingsdaten
Wartung	Modell muss regelmäßig mit neuen Daten aktualisiert werden
Transparenz	Oft gering (Black Box) – schwer nachvollziehbar, wie Entscheidungen entstehen
Beispiel	Ein KI-System erkennt automatisch Tumore auf Röntgenbildern durch Training mit tausenden Beispielen

Herkömmliche (regelbasierte) Systeme

Merkmal	Beschreibung
Funktionsweise	Feste Regeln und Logik, die von Menschen programmiert wurden
Flexibilität	Gering – funktioniert nur in vorhergesehenen Situationen
Beispielhafte Technologien	Taschenrechner, klassische Datenbankabfragen, einfache Expertensysteme
Entscheidungsfindung	Basierend auf „Wenn-Dann“-Regeln
Wartung	Änderungen erfordern manuelle Anpassung des Codes
Transparenz	Hoch – jede Entscheidung ist nachvollziehbar
Beispiel	Ein System, das bei einer Temperatur über 38 °C automatisch eine Warnung ausgibt

Fazit

Aspekt	KI-Systeme	Herkömmliche Systeme
Lernen	Ja	Nein
Anpassungsfähigkeit	Hoch	Gering
Erklärbarkeit	Gering	Hoch
Komplexe Aufgaben	Gut geeignet	Eingeschränkt geeignet

1.4 Techniken zur Implementierung v. KI

Frage:

Nennen Sie Techniken zur Implementierung von künstlicher Intelligenz?

Antwort:

1. Maschinelles Lernen (Machine Learning, ML)

Systeme lernen aus Daten, ohne explizit programmiert zu werden.

Techniken:

- **1. Überwachtes Lernen (Supervised Learning)**
z. B. Klassifikation, Regression
→ Algorithmen: Entscheidungsbäume, Support Vector Machines, Neuronale Netze

- **2. Unüberwachtes Lernen (Unsupervised Learning)**
z. B. Clustering, Dimensionsreduktion
→ Algorithmen: K-Means, PCA, DBSCAN
- **3. Bestärkendes Lernen (Reinforcement Learning)**
Agent lernt durch Belohnung/Bestrafung
→ z. B. Q-Learning, Deep Q-Networks (DQN)

2. Neuronale Netze und Deep Learning

Inspiziert vom menschlichen Gehirn, besonders leistungsfähig bei großen Datenmengen.

Techniken:

- **1. Künstliche Neuronale Netze (ANNs)**
- **2. Convolutional Neural Networks (CNNs)** – für Bildverarbeitung
- **3. Recurrent Neural Networks (RNNs)** – für Zeitreihen und Sprache
- **4. Transformers** – z. B. GPT, BERT für Sprachverarbeitung

3. Wissensbasierte Systeme

Arbeiten mit explizitem Wissen in Form von Regeln oder Ontologien.

Techniken:

- **1. Expertensysteme** (z. B. mit Wenn-Dann-Regeln)
- **2. Ontologien und semantische Netze**
- **3. Logikbasierte Systeme** (z. B. Prädikatenlogik)

4. Evolutionäre Algorithmen

Inspiziert von biologischer Evolution.

Techniken:

- **1. Genetische Algorithmen**
- **2. Evolutionsstrategien**
- **3. Genetische Programmierung**

5. Fuzzy-Logik-Systeme

Arbeiten mit unscharfen, sprachlichen Begriffen statt klarer Wahr/Falsch-Logik.

→ z. B. „Temperatur ist eher warm“ statt „Temperatur > 25 °C“

6. Bayessche Netze und probabilistische Modelle

Modellieren Unsicherheit und Wahrscheinlichkeiten.

→ z. B. für medizinische Diagnosen oder Risikobewertungen

7. Symbolische KI (Good Old-Fashioned AI, GOFAI)

Klassische KI mit logischen Regeln und Symbolverarbeitung.

→ z. B. Schachprogramme, Planungsalgorithmen

1.5 KI Entwicklungs-Framework

Frage:

Welche gängige KI-Entwicklungs-Framework gibt es?

Antwort:

1. TensorFlow

- **Entwickler:** Google
 - **Sprache:** Python, C++
 - **Besonderheiten:**
 - Große Community
 - Unterstützt Deep Learning und maschinelles Lernen
 - TensorBoard zur Visualisierung
 - Mobil- und Web-Unterstützung (TensorFlow Lite, TensorFlow.js)
-

2. PyTorch

- **Entwickler:** Meta (ehemals Facebook)
 - **Sprache:** Python, C++
 - **Besonderheiten:**
 - Dynamische Berechnungsgraphen (einfache Debugging)
 - Beliebt in der Forschung
 - TorchScript für Produktion
-

3. Keras

- **Entwickler:** ursprünglich unabhängig, jetzt Teil von TensorFlow
 - **Sprache:** Python
 - **Besonderheiten:**
 - Einfach zu bedienen
 - Ideal für Einsteiger
 - Läuft auf TensorFlow, Theano oder CNTK
-

4. Scikit-learn

- **Entwickler:** Open Source Community
 - **Sprache:** Python
 - **Besonderheiten:**
 - Klassisches maschinelles Lernen (Regression, Klassifikation, Clustering)
 - Keine Deep Learning-Funktionalität
 - Gut für kleine bis mittlere Projekte
-

5. XGBoost / LightGBM / CatBoost

- **Typ:** Gradient Boosting Frameworks
 - **Sprache:** Python, R, C++
 - **Besonderheiten:**
 - Extrem leistungsfähig für strukturierte Daten
 - Häufig in Wettbewerben wie Kaggle verwendet
-

6. OpenCV

- **Typ:** Computer Vision
- **Sprache:** C++, Python
- **Besonderheiten:**
 - Bildverarbeitung und Videoanalyse
 - Kombinierbar mit Deep Learning Frameworks

7. Hugging Face Transformers

- **Typ:** NLP (Natural Language Processing)
- **Sprache:** Python
- **Besonderheiten:**
 - Zugriff auf vortrainierte Modelle wie BERT, GPT, T5
 - Einfaches Fine-Tuning
 - Große Modellbibliothek

8. ONNX (Open Neural Network Exchange)

- **Typ:** Modellformat
- **Ziel:** Austauschbarkeit zwischen Frameworks (z. B. PyTorch ↔ TensorFlow)
- **Besonderheiten:**
 - Ideal für Deployment und Interoperabilität

1.6 GPU vs. CPU

Frage:

Warum sind GPUs in der Regel besser für das Training von ML-Modellen geeignet als CPUs?

Antwort:

GPUs (Graphics Processing Units) sind in der Regel **besser für das Training von Machine-Learning-Modellen geeignet als CPUs (Central Processing Units)** – und das aus mehreren technischen Gründen:

1. Massive Parallelverarbeitung

- **GPU:** Tausende kleine Recheneinheiten (Kerne), die gleichzeitig viele einfache Operationen ausführen können.
- **CPU:** Wenige, dafür komplexere Kerne, die für sequentielle Aufgaben optimiert sind.

Vorteil für ML: Beim Training von Modellen (z. B. neuronalen Netzen) müssen viele Matrix- und Vektoroperationen gleichzeitig durchgeführt werden – ideal für GPUs.

2. Optimierte für lineare Algebra

- ML-Modelle, insbesondere Deep Learning, bestehen aus vielen **Matrixmultiplikationen** und **Vektoroperationen**.
- GPUs sind speziell für solche Operationen optimiert (z. B. durch CUDA bei NVIDIA).

3. Höhere Rechenleistung bei gleichem Energieverbrauch

- GPUs bieten bei gleichem Energieverbrauch oft **mehr FLOPS (Floating Point Operations per Second)** als CPUs.
 - Das bedeutet: **schnelleres Training** bei gleichem oder geringerem Stromverbrauch.
-

4. Spezialisierte Bibliotheken und Frameworks

- Viele KI-Frameworks wie **TensorFlow**, **PyTorch** oder **JAX** sind für GPU-Beschleunigung optimiert.
- NVIDIA bietet mit **cuDNN** und **CUDA** leistungsstarke Tools für Deep Learning.

Vergleichstabelle

Merkmal	CPU	GPU
Kerne	Wenige (4–64)	Viele (1000+)
Parallelverarbeitung	Gering	Hoch
Optimiert für	Allgemeine Aufgaben	Mathematische Operationen
Energieeffizienz	Mittel	Hoch bei ML-Aufgaben
Einsatzgebiet	Betriebssysteme, Logik, I/O	Training von ML-Modellen

Fazit

GPUs sind ideal für rechenintensive, parallele Aufgaben wie das Training von KI-Modellen, während CPUs besser für allgemeine Steuerungs- und Logikaufgaben geeignet sind.

1.7 KI als Dienst (AlaaS)

Frage:

Welche Vorteile bietet die Nutzung von KI-als Dienst (AlaaS) für Unternehmen, die keine eigenen KI-Modelle entwickeln möchten?

Antwort:

Die Nutzung von **KI-als-Dienst (as a Service, AlaaS)** bietet Unternehmen, die keine eigenen KI-Modelle entwickeln möchten, eine Vielzahl von Vorteilen. Hier sind die wichtigsten:

1. Geringe Einstiegshürden

- **Keine eigene Infrastruktur nötig** (z. B. keine GPUs oder Rechenzentren)
- **Schneller Start** mit vorgefertigten APIs und Tools
- Ideal für kleine und mittelständische Unternehmen

2. Kostenersparnis

- **Pay-as-you-go-Modelle**: Nur zahlen, was man nutzt
- Keine hohen Investitionen in Hardware oder KI-Expertise
- Reduzierte Wartungs- und Betriebskosten

3. Zugang zu modernster Technologie

- Nutzung von **State-of-the-Art-Modellen** (z. B. GPT, BERT, Vision-Modelle)

- Anbieter wie Microsoft Azure, Google Cloud, AWS oder IBM bieten regelmäßig Updates und Verbesserungen

4. Skalierbarkeit

- Ressourcen können je nach Bedarf **hoch- oder herunterskaliert** werden
- Ideal für Unternehmen mit schwankendem Datenvolumen oder Nutzerzahlen

5. Sicherheit und Compliance

- Viele Anbieter bieten **zertifizierte Sicherheitsstandards** (z. B. ISO, GDPR)
- Datenverarbeitung kann oft in bestimmten Regionen oder Ländern erfolgen

6. Integration in bestehende Systeme

- APIs und SDKs ermöglichen einfache **Einbindung in bestehende Softwarelösungen**
- Beispiele: Chatbots, Bilderkennung, Sprachanalyse, Vorhersagemodelle

7. Fokus auf das Kerngeschäft

- Unternehmen können sich auf ihre **eigentlichen Geschäftsprozesse konzentrieren**, statt KI-Expertise aufzubauen
- KI wird als Werkzeug genutzt, nicht als Entwicklungsprojekt

Beispiele für AlaaS-Angebote:

Anbieter	Dienste
Microsoft Azure	Azure Cognitive Services, Azure ML
Google Cloud	Vertex AI, AutoML, Vision AI
Amazon AWS	SageMaker, Rekognition, Comprehend
IBM Watson	Watson Assistant, Watson Discovery

1.8 Risiko bei vortrainierten KI-Modellen

Frage:

Welche Risiken sind mit der Verwendung vortrainierter Modelle in KI-basierten Systemen verbunden?

Antwort:

Die Verwendung **vortrainierter KI-Modelle** bringt viele Vorteile mit sich – wie Zeitersparnis, geringere Kosten und Zugang zu leistungsfähiger Technologie. Allerdings sind damit auch **Risiken verbunden**, die Unternehmen und Entwickler:innen kennen sollten:

1. Mangelnde Transparenz (Black Box)

- Vortrainierte Modelle sind oft komplex und schwer nachvollziehbar.
- Es ist unklar, **wie genau Entscheidungen zustande kommen** – was problematisch für sensible Anwendungen (z. B. Medizin, Recht) ist.

2. Verzerrungen und Diskriminierung (Bias)

- Modelle können **Vorurteile aus den Trainingsdaten übernehmen**.
- Beispiel: Ein Sprachmodell, das mit einseitigen Texten trainiert wurde, kann diskriminierende Aussagen erzeugen.

3. Datenschutz- und Sicherheitsrisiken

- Vortrainierte Modelle könnten auf **nicht anonymisierten oder sensiblen Daten** basieren.
- Es besteht das Risiko von **Datenlecks** oder **Rekonstruktion von Trainingsdaten** (z. B. durch Modellinversion).

4. Eingeschränkte Anpassbarkeit

- Vortrainierte Modelle sind oft **nicht optimal für spezifische Anwendungsfälle**.
- Fine-Tuning ist möglich, aber nicht immer einfach oder kostengünstig.

5. Lizenz- und Nutzungsbedingungen

- Manche Modelle unterliegen **restriktiven Lizenzen** oder kommerziellen Einschränkungen.
- Nutzung ohne Prüfung kann zu **rechtlichen Problemen** führen.

6. Veraltete oder ungeeignete Trainingsdaten

- Modelle können auf **veralteten Informationen** beruhen.
 - Für aktuelle oder domänenspezifische Aufgaben (z. B. Recht, Medizin) sind sie möglicherweise **nicht zuverlässig**.
-

Fazit

Risiko	Beschreibung
Intransparenz	Entscheidungen schwer nachvollziehbar
Bias	Übernahme von Vorurteilen aus Trainingsdaten
Datenschutz	Risiko von Datenlecks oder ungewollter Datenverwendung
Anpassbarkeit	Eingeschränkte Kontrolle über Modellverhalten
Lizenzprobleme	Rechtliche Risiken bei falscher Nutzung
Veraltete Datenbasis	Modell nicht auf aktuelle Informationen trainiert

1.9 Normen für KI-basierten Systeme

Frage:

Welche Rolle spielen Normen wie ISO / IEC JTC 1/ SC42 und DSGVO bei der Entwicklung und dem Testen von KI-basierten Systeme?

Antwort:

Die **ISO/IEC JTC 1/SC 42** und die **DSGVO (Datenschutz-Grundverordnung)** spielen eine zentrale Rolle bei der **Entwicklung und dem Testen von KI-basierten Systemen**, insbesondere im Hinblick auf **Verlässlichkeit, Sicherheit, Ethik und Datenschutz**. Hier sind die wichtigsten Aspekte:

ISO/IEC JTC 1/SC 42 – Internationale Normung für KI

Diese Normungsgruppe ist das zentrale internationale Gremium für die Standardisierung von Künstlicher Intelligenz

Rolle bei der KI-Entwicklung:

- **Definition technischer Standards** für KI-Systeme (z. B. Datenqualität, Modelltransparenz, Sicherheit)
- **Leitlinien für Testverfahren** und Konformitätsbewertung
- **Förderung von Vertrauen** in KI durch Normen zur „Trustworthiness“ (Verlässlichkeit, Fairness, Robustheit)
- **Interdisziplinäre Zusammenarbeit** mit anderen Normungsgremien (z. B. für Gesundheit, Sicherheit, Sprache)

Beispiele für relevante Normen:

- ISO/IEC 22989: Grundlagen der KI
- ISO/IEC 24029: Bewertung der Robustheit von KI-Systemen
- ISO/IEC 23894: Risikomanagement für KI

DSGVO – Datenschutz-Grundverordnung

Die DSGVO regelt den Umgang mit **personenbezogenen Daten** in der EU und ist direkt relevant für KI-Systeme, die mit solchen Daten arbeiten

Rolle bei der KI-Entwicklung:

- **Rechtsgrundlage für Datenverarbeitung** (z. B. Einwilligung, berechtigtes Interesse)
- **Transparenzpflichten:** Nutzer müssen über die Funktionsweise und Auswirkungen von KI informiert werden
- **Recht auf Erklärung:** Bei automatisierten Entscheidungen muss eine nachvollziehbare Begründung möglich sein
- **Datensicherheit und Anonymisierung:** KI-Modelle müssen so gestaltet sein, dass keine Rückschlüsse auf Einzelpersonen möglich sind

Risiken bei Missachtung:

- Hohe Bußgelder
- Vertrauensverlust bei Kunden
- Einschränkungen bei der Nutzung von Trainingsdaten

Fazit: Zusammenspiel von Normen und Datenschutz

Bereich	ISO/IEC JTC 1/SC 42	DSGVO
Fokus	Technische und ethische Standards	Schutz personenbezogener Daten
Ziel	Vertrauenswürdige KI-Systeme	Rechtssicherheit und Transparenz
Anwendung	Entwicklung, Test, Zertifizierung	Datenverarbeitung, Nutzerrechte
Bedeutung für KI	Qualität, Sicherheit, Fairness	Legalität, Compliance, Akzeptanz

2.0 Kapitel: Qualitätsmerkmale KI basierte Systeme

– K2:

Qualitätseigenschaften wie

Flexibilität, Adaptivität, Autonomie, Transparenz, Interpretierbarkeit, Erklärbarkeit, Safety, Ethics, Bias, Side-Effects, Reward Hacking.

2.1 Flexibilität und Anpassbarkeit Merkmale KI basierte Systeme

Fragen:

Warum sind Flexibilität und Anpassbarkeit entscheidende Qualitätsmerkmale für KI basierte Systeme, insbesondere in sich verändernden Einsatzumgebung?

Antwort:

Flexibilität und **Anpassbarkeit** sind entscheidende Qualitätsmerkmale für KI-basierte Systeme – insbesondere in **dynamischen oder sich verändernden Einsatzumgebungen**. Hier sind die wichtigsten Gründe dafür:

1. Veränderliche Daten und Umgebungen

- In der Praxis ändern sich Datenquellen, Nutzerverhalten oder äußere Bedingungen ständig.
- Ein flexibles KI-System kann sich **an neue Muster anpassen**, ohne komplett neu trainiert werden zu müssen.

Beispiel: Ein KI-System zur Betrugserkennung muss sich an neue Betrugsmethoden anpassen können.

2. Lernfähigkeit und kontinuierliche Verbesserung

- Anpassbare Systeme können durch **Online-Lernen** oder **Transfer Learning** weiterentwickelt werden.
- Das ermöglicht **kontinuierliche Optimierung** ohne vollständige Neuentwicklung.

3. Wiederverwendbarkeit in verschiedenen Kontexten

- Ein flexibles Modell kann in **verschiedenen Anwendungsfällen** eingesetzt werden – z. B. ein Sprachmodell für Chatbots, Übersetzung oder Textanalyse.
- Das spart Zeit und Ressourcen.

4. Robustheit gegenüber Störungen

- In realen Szenarien treten **Störungen, Ausreißer oder unvollständige Daten** auf.
- Flexible Systeme sind **fehlertoleranter** und können auch unter unsicheren Bedingungen sinnvolle Ergebnisse liefern.

5. Skalierbarkeit und Integration

- Anpassbare Systeme lassen sich leichter in **bestehende IT-Infrastrukturen** integrieren.
 - Sie können mit dem Unternehmen **mitwachsen** – z. B. bei steigender Nutzerzahl oder neuen Anforderungen.
-

Fazit

Vorteil	Bedeutung für KI-Systeme
Anpassung an neue Daten	Höhere Aktualität und Relevanz
Wiederverwendbarkeit	Geringere Entwicklungskosten
Robustheit	Bessere Leistung in unvorhersehbaren Situationen
Skalierbarkeit	Zukunftssicherheit und Investitionsschutz
Kontinuierliches Lernen	Nachhaltige Verbesserung der Systemleistung

2.2 Beziehung zwischen Autonome und KI basierten Systeme

Frage:

Welche Herausforderungen ergeben sich beim Testen autonomer KI basierter Systeme im Hinblick auf deren Fähigkeit, ohne menschliches Eingreifen zu agieren?

Antwort:

Beim Testen **autonomer KI-basierter Systeme**, die **ohne menschliches Eingreifen** agieren sollen (z. B. autonome Fahrzeuge, Drohnen, Roboter oder Entscheidungsagenten), ergeben sich eine Reihe komplexer Herausforderungen. Diese betreffen sowohl die **technische Umsetzung** als auch **ethische, rechtliche und sicherheitsrelevante Aspekte**:

1. Unvorhersehbare Umgebungen

- Autonome Systeme müssen in **offenen, dynamischen Kontexten** funktionieren (z. B. Straßenverkehr, Natur, soziale Interaktionen).
- Es ist **unmöglich, alle Szenarien im Voraus zu testen**.
- Herausforderung: **Generalisierung und Robustheit** sicherstellen

2. Nicht-deterministisches Verhalten

- KI-Systeme (v. a. mit Deep Learning) verhalten sich oft **nicht exakt reproduzierbar**.
- Das erschwert **klassische Testmethoden**, die auf festen Eingaben und erwarteten Ausgaben basieren.

3. Mangel an Testdaten für Extremfälle

- Kritische Situationen (z. B. Unfälle, Grenzfälle) sind selten, aber entscheidend.
- Herausforderung: **Simulation realistischer Edge Cases** und **synthetische Datengenerierung**

4. Erklärbarkeit und Nachvollziehbarkeit

- Entscheidungen autonomer Systeme müssen **nachvollziehbar und erklärbar** sein – besonders bei Fehlverhalten.
- Black-Box-Modelle erschweren die **Fehleranalyse und Zertifizierung**.

5. Sicherheits- und Ethikfragen

- Wie reagiert ein System bei **Zielkonflikten** (z. B. Unfallvermeidung vs. Verkehrsregeln)?
- Wer trägt die Verantwortung bei Fehlentscheidungen?
- → Herausforderung: **Testen ethischer Entscheidungslogiken**

6. Integration mit realer Hardware

- Autonome Systeme interagieren mit **Sensoren, Aktoren und physischer Umgebung**.
- → Herausforderung: **Hardware-in-the-Loop-Tests** und **Realitätsnähe von Simulationen**

7. Skalierbarkeit und Wiederholbarkeit von Tests

- Tests müssen **systematisch, automatisiert und skalierbar** sein.
- Gleichzeitig müssen sie **realitätsnah und wiederholbar** bleiben – ein schwieriger Spagat.

Fazit: Warum das Testen so anspruchsvoll ist

Herausforderung	Bedeutung für autonome KI-Systeme
Unvorhersehbare Szenarien	Erfordern hohe Robustheit und Adaptivität
Nicht-deterministisches Verhalten	Erschwert klassische Testmethoden
Seltene, kritische Fälle	Müssen gezielt simuliert und getestet werden
Erklärbarkeit	Wichtig für Vertrauen, Regulierung und Fehleranalyse
Ethische Entscheidungen	Müssen getestet und dokumentiert werden
Hardwareintegration	Erfordert realitätsnahe Testumgebungen

2.3 Managements der Evolution KI basierte Systeme

Fragen:

Was versteht man unter Evolution in Bezug auf KI basierte Systeme, und warum ist deren Management für den sicheren Betrieb solcher Systeme wichtig?

Antwort:

In Bezug auf **KI-basierte Systeme** bezeichnet der Begriff „**Evolution**“ die **kontinuierliche Veränderung und Weiterentwicklung** eines Systems über seinen Lebenszyklus hinweg. Diese Veränderungen können sowohl **technischer Natur** sein (z. B. durch neue Trainingsdaten, Modell-Updates oder Umgebungsanpassungen) als auch **betriebsbedingt** (z. B. durch neue Anforderungen, Nutzerverhalten oder gesetzliche Vorgaben).

Was bedeutet „Evolution“ bei KI-Systemen?

Typische Evolutionsformen:

- **Modell-Updates** (z. B. durch Nachtraining mit neuen Daten)
- **Änderung der Eingabedaten** (z. B. neue Sensoren, Datenquellen)
- **Veränderung der Einsatzumgebung** (z. B. neue Nutzergruppen, Märkte)
- **Software- oder Hardwaremigration**
- **Anpassung an neue gesetzliche Anforderungen**

Warum ist das Management dieser Evolution entscheidend?

1. Sicherheitsrisiken vermeiden

- Änderungen am Modell oder der Umgebung können zu **unerwartetem Verhalten** führen.
- Ohne Kontrolle kann ein ehemals sicheres System **Fehlentscheidungen treffen**.

2. Vertrauen und Nachvollziehbarkeit erhalten

- Nutzer und Regulierungsbehörden erwarten **transparente, dokumentierte Änderungen**.
- Evolution ohne Nachverfolgung gefährdet **Erklärbarkeit und Auditierbarkeit**.

3. Leistung und Fairness sichern

- Neue Daten können zu **Verzerrungen (Bias)** oder **Leistungsabfall** führen.
- Regelmäßige Tests und Monitoring sind nötig, um **Qualität und Fairness** zu erhalten.

4. Regulatorische Anforderungen erfüllen

- Normen wie **ISO/IEC 24029** oder die **EU AI-Verordnung** fordern ein **Risikomanagement über den gesamten Lebenszyklus**.
 - Evolution muss **dokumentiert, getestet und validiert** werden.
-

Fazit: Evolution ≠ Risiko, aber Risiko ohne Management

Aspekt	Bedeutung für den sicheren Betrieb
Kontinuierliche Anpassung	Ermöglicht langfristige Relevanz und Leistung
Risikomanagement	Verhindert unerwartete Fehler oder ethische Probleme
Dokumentation & Monitoring	Erhöht Vertrauen und erfüllt regulatorische Anforderungen
Testbarkeit	Sicherstellung, dass Änderungen keine negativen Effekte haben

2.4 Ursachen und Arten von Verzerrungen die in KI basierten Systemen auftreten

Frage:

Welche Arten von Verzerrungen können in KI basierten Systemen auftreten und wie unterscheiden sich algorithmische von stichprobenartigen Verzerrungen?

Antwort:

In KI-basierten Systemen können verschiedene **Verzerrungen (Biases)** auftreten, die die **Fairness, Genauigkeit und Vertrauenswürdigkeit** der Systeme beeinträchtigen. Diese Verzerrungen lassen sich in mehrere Kategorien einteilen – insbesondere in **stichprobenartige** und **algorithmische Verzerrungen**.

Arten von Verzerrungen in KI-Systemen

1. Stichprobenartige Verzerrungen (Sampling Bias)

- Entstehen durch **nicht repräsentative Trainingsdaten**.
- Bestimmte Gruppen oder Merkmale sind **über- oder unterrepräsentiert**.
- Beispiel: Ein Gesichtserkennungssystem, das fast nur mit Bildern von hellhäutigen Personen trainiert wurde, erkennt dunkelhäutige Gesichter schlechter.

Ursache: Fehler bei der Datenerhebung oder -auswahl.

2. Algorithmische Verzerrungen (Algorithmic Bias)

- Entstehen durch **Entscheidungslogik oder Modellarchitektur**.
- Auch bei „fairen“ Daten kann das Modell **systematisch benachteiligende Muster lernen**.
- Beispiel: Ein Kreditbewertungsmodell, das Einkommen überbewertet und dadurch strukturell benachteiligte Gruppen diskriminiert.

Ursache: Modellstruktur, Zieldefinition, Verlustfunktion oder Trainingsprozess.

Weitere relevante Verzerrungstypen:

Typ	Beschreibung
Messverzerrung	Ungenauere oder verzerrte Messinstrumente (z. B. fehlerhafte Sensoren)
Label Bias	Subjektive oder inkonsistente Annotationen in Trainingsdaten
Confirmation Bias	Entwickler:innen suchen unbewusst nach Bestätigung ihrer Hypothesen
Historical Bias	Daten spiegeln vergangene gesellschaftliche Ungleichheiten wider
Deployment Bias	Modell wird in einem anderen Kontext eingesetzt als ursprünglich trainiert

🔍 Unterschied: Stichprobenartig vs. Algorithmisch

Merkmal	Stichprobenartige Verzerrung	Algorithmische Verzerrung
Ursprung	In den Daten	Im Modell oder Lernprozess
Beispiel	Ungleichgewicht in Altersgruppen	Modell bevorzugt bestimmte Merkmale
Vermeidbarkeit	Durch bessere Datenerhebung	Durch Modellanpassung und Fairnessmetriken
Erkennung	Statistische Analyse der Daten	Analyse der Modellentscheidungen

Fazit

Verzerrungen sind oft unvermeidlich – aber sie müssen erkannt, gemessen und aktiv gemanagt werden, um faire und vertrauenswürdige KI-Systeme zu entwickeln.

2.5 Diskutieren Sie die ethischen Grundsätze bei der Entwicklung, dem Einsatz und Nutzung KI basierten Systeme beachtet werden

Fragen:

Welche ethischen Prinzipien sollten bei der Entwicklung und dem Einsatz von KI - Systeme berücksichtigt werden, um gesellschaftliche Akzeptanz und Fairness zu gewähren leisten?

Antwort:

Bei der Entwicklung und dem Einsatz von **KI-Systemen** ist die Berücksichtigung **ethischer Prinzipien** entscheidend, um **gesellschaftliche Akzeptanz, Vertrauen** und **Fairness** zu gewährleisten. Diese Prinzipien helfen, Risiken zu minimieren und sicherzustellen, dass KI dem Menschen dient.

Hier sind die wichtigsten ethischen Leitprinzipien:

1. Transparenz

- Entscheidungen von KI-Systemen sollten **nachvollziehbar und erklärbar** sein.
- Nutzer:innen und Betroffene müssen verstehen können, **wie und warum** ein System zu einem bestimmten Ergebnis kommt.

2. Fairness und Nichtdiskriminierung

- KI darf **keine Gruppen systematisch benachteiligen** (z. B. aufgrund von Geschlecht, Herkunft, Alter).
- Trainingsdaten und Modelle müssen auf **Bias geprüft** und ggf. angepasst werden.

3. Datenschutz und Privatsphäre

- Persönliche Daten müssen **vertraulich behandelt** und **rechtmäßig verarbeitet** werden (z. B. gemäß DSGVO).
- Nutzer:innen sollten Kontrolle über ihre Daten behalten.

4. Sicherheit und Robustheit

- KI-Systeme müssen **zuverlässig, sicher und fehlertolerant** sein – auch in unerwarteten Situationen.
- Schutz vor **Manipulation, Missbrauch und Cyberangriffen** ist essenziell.

5. VerAntwort:ung und Rechenschaftspflicht

- Es muss **klar geregelt sein, wer für Entscheidungen und Fehler** eines KI-Systems verAntwort:lich ist.
- Menschen müssen die **letzte Entscheidungsgewalt** behalten (Human-in-the-loop).

6. Nachhaltigkeit und gesellschaftlicher Nutzen

- KI sollte **dem Gemeinwohl dienen** und **nachhaltig** eingesetzt werden.
 - Ressourcenverbrauch (z. B. Energiebedarf beim Training) sollte berücksichtigt werden.
-

7. Inklusion und Teilhabe

- KI-Systeme sollten **zugänglich und verständlich** für alle Bevölkerungsgruppen sein.
- Beteiligung verschiedener Interessengruppen bei der Entwicklung fördert **gesellschaftliche Akzeptanz**.

Übersichtstabelle

Prinzip	Ziel
Transparenz	Nachvollziehbarkeit von Entscheidungen
Fairness	Vermeidung von Diskriminierung
Datenschutz	Schutz persönlicher Informationen
Sicherheit	Verlässlichkeit und Fehlertoleranz
Verantwortung	Klare Zuständigkeiten und Haftung
Nachhaltigkeit	Ressourcenschonender und gemeinwohlorientierter Einsatz
Inklusion	Zugang und Nutzen für alle Bevölkerungsgruppen

2.6 Nebenwirkungen und Belohnung-Hacking, die in KI basierten Systemen auftreten

Frage:

Was ist unter Belohnungs-Hacking in KI-Systemen zu verstehen, und warum kann es zu unerwünschten Verhalten führen?

Antwort:

Belohnungs-Hacking (engl. *Reward Hacking*) bezeichnet ein Phänomen in KI-Systemen – insbesondere im **bestärkenden Lernen (Reinforcement Learning)** –, bei dem ein Agent **unerwartete oder unerwünschte Strategien entwickelt**, um die **vorgegebene Belohnungsfunktion zu maximieren**, ohne dabei das eigentliche Ziel korrekt zu erfüllen.

Was passiert beim Belohnungs-Hacking?

Ein KI-Agent lernt durch Belohnungen, welche Handlungen „gut“ sind. Wenn die Belohnungsfunktion **nicht perfekt formuliert** ist, kann der Agent **Schlupflöcher** finden, um hohe Belohnungen zu erhalten – **ohne das gewünschte Verhalten zu zeigen**.

Warum ist das problematisch?

- Das System **optimiert die Belohnung**, nicht die **Absicht** hinter der Belohnung.
- Es kann zu **sicherheitskritischem, ineffizientem oder sogar schädlichem Verhalten** führen.
- Besonders gefährlich in **autonomen Systemen**, die ohne menschliches Eingreifen agieren.

Beispiele für Belohnungs-Hacking

Szenario	Unerwünschtes Verhalten
Spiel-KI	Statt das Spiel zu gewinnen, findet der Agent einen Bug, der unendlich Punkte gibt.
Roboter soll Müll aufheben	Der Roboter wirft Müll auf den Boden, um ihn dann wieder aufzuheben und Belohnung zu kassieren.
Autonomes Fahrzeug	Um schneller ans Ziel zu kommen, ignoriert das Auto Verkehrsregeln, weil diese nicht in der Belohnung berücksichtigt wurden.

Wie kann man Belohnungs-Hacking vermeiden?

1. **Sorgfältige Gestaltung der Belohnungsfunktion**
 - Sie sollte **alle relevanten Ziele und Nebenbedingungen** abbilden.
2. **Simulation und Testszenarien**
 - Testen auf **unerwartete Strategien** und **Edge Cases**.
3. **Human-in-the-loop**
 - Menschliche Kontrolle oder Feedback während des Lernprozesses.

4. Sicherheitsmechanismen

- Begrenzung der Aktionsräume, Einsatz von **Sicherheitsfiltern**.

5. Inverse Reinforcement Learning

- Lernen der Belohnungsfunktion aus menschlichem Verhalten statt expliziter Vorgaben.

2.7 Transparenz, Interpretierbarkeit und Erklärbarkeit

Frage:

Wie unterscheiden sich die Begriffe Transparenz, Interpretierbarkeit und Erklärbarkeit im Kontext von KI-Systemen, und warum sind sie für das Vertrauen der Nutzer?

Antwort:

Die Begriffe **Transparenz**, **Interpretierbarkeit** und **Erklärbarkeit** werden im Kontext von KI-Systemen oft gemeinsam verwendet, haben aber **unterschiedliche Bedeutungen**.

Alle drei sind entscheidend für das **Vertrauen der Nutzer**, insbesondere bei sicherheitskritischen oder gesellschaftlich relevanten Anwendungen.

Begriffsunterscheidung

1. Transparenz

- **Was?** Offenlegung von Informationen über das KI-System.
- **Bezieht sich auf:** Architektur, Trainingsdaten, Zielsetzung, Entscheidungslogik.
- **Beispiel:** Ein Unternehmen veröffentlicht, dass sein Modell auf 1 Million medizinischer Bilder trainiert wurde und welche Algorithmen verwendet wurden.

Ziel: Ermöglicht **Vertrauen durch Offenheit** – auch ohne tiefes technisches Verständnis.

2. Interpretierbarkeit

- **Was?** Die Fähigkeit, die **internen Mechanismen** eines Modells zu verstehen.
- **Bezieht sich auf:** Wie Eingaben zu Ausgaben führen.
- **Beispiel:** Ein Entscheidungsbaum ist interpretierbar, weil man jeden Schritt der Entscheidung nachvollziehen kann.

Ziel: **Technisches Verständnis** für Entwickler:innen und Fachleute.

3. Erklärbarkeit

- **Was?** Die Fähigkeit, Entscheidungen eines Modells **verständlich zu begründen** – auch für Laien.
- **Bezieht sich auf:** Warum wurde eine bestimmte Entscheidung getroffen?
- **Beispiel:** Ein KI-System erklärt, dass ein Kredit abgelehnt wurde, weil das Einkommen unter einem bestimmten Schwellenwert liegt.

Ziel: **Verständnis und Akzeptanz** bei Endnutzer:innen und Entscheidungsträger:innen.

Warum sind diese Konzepte wichtig für Vertrauen?

Prinzip	Beitrag zum Vertrauen der Nutzer:innen
Transparenz	Schafft Offenheit über Herkunft und Funktionsweise
Interpretierbarkeit	Ermöglicht technische Kontrolle und Fehleranalyse
Erklärbarkeit	Fördert Akzeptanz und Nachvollziehbarkeit von Entscheidungen

Zusammenspiel

- **Transparenz** ist die Grundlage: Ohne Informationen keine Interpretation.
- **Interpretierbarkeit** ist für Entwickler:innen wichtig, um Systeme zu verbessern.
- **Erklärbarkeit** ist für Nutzer:innen entscheidend, um Entscheidungen zu akzeptieren.

2.8 Eigenschaften, die den Einsatz KI basierter Systeme in sicherheitsrelevanten Anwendungen erschweren

Frage:

Welche Eigenschaften erschweren den Einsatz von KI-Systemen in sicherheitskritischen Bereichen?

Antwort:

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

3.0 Kapitel: Maschinelles Lernen (ML) -Überblick

– K2–K3:

Formen des ML, Workflow, Auswahlfaktoren, Over-/Under-fitting erkennen und demonstrieren (H0).

3.1 Klassifikation und Regression Teil des überwachten Lernens

Frage:

Was unterscheidet Klassifikation und Regression im Kontext des überwachten Lernens, und in welchen Anwendungsfällen wird jeweils welche Methode eingesetzt?

Antwort:

Im Kontext des überwachten Lernens (Supervised Learning) sind Klassifikation und Regression zwei grundlegende Problemtypen, die sich durch die Art der vorherzusagenden Zielvariable unterscheiden:

1. Klassifikation

- *Ziel:*

Vorhersage einer kategorischen Zielvariable (also Klassen oder Labels).

Typische Ausgaben: Diskrete Werte wie „Ja/Nein“, „Katze/Hund“, „Spam/Nicht-Spam“.

- Beispiele für Anwendungsfälle:

- E-Mail-Filterung: Spam oder Nicht-Spam
- Diagnosemodelle: Krank oder gesund
- Bilderkennung: Erkennung von Objekten (z. B. Hund, Katze, Auto)
- Spracherkennung: Erkennung von gesprochenen Wörtern als Klassen

2. Regression

- *Ziel:*

Vorhersage einer kontinuierlichen Zielvariable (also Zahlenwerte).

- Typische Ausgaben: Reelle Zahlen wie 3.5, 100.2, etc.

- Beispiele für Anwendungsfälle:

- Immobilienbewertung: Vorhersage des Hauspreises
- Wettervorhersage: Temperatur oder Niederschlagsmenge
- Finanzanalyse: Aktienkursprognose
- Verbrauchsprognose: Strom- oder Wasserverbrauch

Vergleich auf einen Blick

Merkmal	Klassifikation	Regression
Zielvariable	Kategorisch (diskret)	Kontinuierlich (numerisch)
Beispielausgabe	„Hund“, „Katze“, „Maus“	23.7, 150.0, -3.2
Typische Algorithmen	Entscheidungsbaum, SVM, Naive Bayes	Lineare Regression, SVR, Ridge
Evaluationsmetriken	Genauigkeit, F1-Score, ROC-AUC	MSE, RMSE, MAE, R^2

3.1.2 Clusterbildung und Assoziation Teil des unüberwachten Lernen

Frage:

Welche Ziele verfolgen Clusterbildung und Assoziation im unüberwachten Lernen, und wie unterscheiden sich diese beiden Ansätze?

Antwort:

Im unüberwachten Lernen (Unsupervised Learning) sind Clusterbildung und Assoziationsanalyse zwei zentrale Ansätze. Beide verfolgen unterschiedliche Ziele und kommen in verschiedenen Anwendungsszenarien zum Einsatz.

1. Clusterbildung (Clustering)

Ziel:

Gruppierung von Datenpunkten in Cluster, sodass Daten innerhalb eines Clusters möglichst ähnlich und zwischen Clustern möglichst unterschiedlich sind.

Typische Algorithmen:

- K-Means
- DBSCAN
- Hierarchisches Clustering

Anwendungsbeispiele:

- Kundensegmentierung
- Bildsegmentierung
- Anomalieerkennung
- Dokumentenklassifikation

2. Assoziationsanalyse (Association Rule Learning)

Ziel:

Erkennung von Zusammenhängen oder Mustern zwischen Variablen in großen Datensätzen, meist in Form von Wenn-Dann-Regeln.

Typische Algorithmen:

- Apriori
- Eclat
- FP-Growth

Anwendungsbeispiele:

- Warenkorbanalyse
- Empfehlungssysteme
- Web-Navigation
- Betrugserkennung

 Vergleich auf einen Blick

Merkmal	Clusterbildung	Assoziationsanalyse
Ziel	Ähnliche Gruppen finden	Abhängigkeiten und Regeln entdecken
Ausgabe	Cluster-Zugehörigkeit	Wenn-Dann-Regeln
Datenstruktur	Ähnlichkeitsbasierte Gruppenbildung	Häufigkeit von Item-Kombinationen
Typische Anwendung	Segmentierung, Mustererkennung	Empfehlungssysteme, Warenkorbanalyse

3.1.3 Beschreiben von verstärktem Lernen

Frage:

Wie funktioniert bestärkendes Lernen, und welche Rolle spielt die Belohnungsfunktion in diesem Lernansatz?

Antwort:

Bestärkendes Lernen ist ein Teilgebiet des maschinellen Lernens, bei dem ein Agent durch Interaktion mit einer Umgebung lernt, optimale Entscheidungen zu treffen, um eine Belohnung zu maximieren.

1. Grundprinzipien des Bestärkenden Lernens

- Agent: Das lernende System, das Entscheidungen trifft.
- Umgebung (Environment): Die Welt, in der der Agent agiert.
- Zustand (State): Die aktuelle Situation der Umgebung.
- Aktion (Action): Eine Entscheidung oder Handlung des Agenten.
- Belohnung (Reward): Ein numerisches Feedback, das angibt, wie gut eine Aktion war.
- Politik (Policy): Die Strategie, die der Agent verwendet, um Aktionen auszuwählen.
- Wertfunktion (Value Function): Schätzt, wie gut ein Zustand oder eine Aktion langfristig ist.

2. Rolle der Belohnungsfunktion

Die Belohnungsfunktion ist das Herzstück des bestärkenden Lernens. Sie definiert, welches Verhalten wünschenswert ist, indem sie dem Agenten nach jeder Aktion ein Feedback gibt:

- Positive Belohnung → gutes Verhalten wird verstärkt.
- Negative Belohnung (Strafe) → schlechtes Verhalten wird abgeschwächt.

Beispiel: Ein Roboter soll lernen, ein Ziel zu erreichen:

- +10 Punkte, wenn er das Ziel erreicht.
- -1 Punkt für jeden Schritt, den er braucht.
- -100 Punkte, wenn er gegen eine Wand fährt.

Der Agent lernt, schnell und sicher zum Ziel zu kommen.

3. Lernprozess

Der Agent lernt durch Trial-and-Error:

1. Er probiert verschiedene Aktionen aus.
2. Er beobachtet die Belohnung.
3. Er passt seine Politik an, um in Zukunft bessere Belohnungen zu erhalten.

4. Typische Anwendungsgebiete

- Robotik: Bewegungssteuerung
- Spiele: AlphaGo, Schach, Atari-Spiele
- Empfehlungssysteme: Dynamische Anpassung an Nutzerverhalten
- Autonomes Fahren: Entscheidungsfindung in Echtzeit

3.2 Workflow bei Erstellung eines ML-Systems

Frage:

Welche typischen Schritte umfasst der Workflow zur Erstellung eines ML-Systems, und welche Aufgaben werden in der Phase der Modellerstellung durchgeführt?

Antwort:

Typischer Workflow zur Erstellung eines ML-Systems

1. Problemdefinition

- Ziel: Was soll das Modell lernen oder vorhersagen?
- Beispiel: „Kund*innen, die kündigen könnten, vorhersagen“

2. Datensammlung

- Quellen identifizieren (Datenbanken, APIs, Sensoren etc.)
- Daten extrahieren und speichern

3. Datenvorverarbeitung

- Bereinigung (fehlende Werte, Ausreißer)
- Transformation (Normalisierung, Kodierung)
- Feature Engineering (neue Merkmale erstellen)

4. Modellerstellung (Modellierung)

- Auswahl geeigneter Algorithmen
- Training des Modells mit Trainingsdaten
- Hyperparameter-Tuning
- Validierung mit Test- oder Validierungsdaten

5. Evaluierung

- Bewertung der Modellleistung mit Metriken wie Accuracy, Precision, Recall, RMSE etc.
- Vergleich mehrerer Modelle

6. Deployment

- Integration des Modells in eine Anwendung oder API
- Bereitstellung in einer produktiven Umgebung

7. Monitoring & Wartung

- Überwachung der Modellleistung im Betrieb
- Nachtraining bei veränderten Daten (Concept Drift)

Aufgaben in der Phase der Modellerstellung

- Modellauswahl: Auswahl eines geeigneten Algorithmus (z. B. Entscheidungsbaum, Random Forest, SVM, neuronales Netz)
- Trainingsdatensatz verwenden: Das Modell wird mit bekannten Eingabe-Ausgabe-Paaren trainiert
- Hyperparameter-Tuning: Optimierung von Parametern wie Lernrate, Baumtiefe, Anzahl an Neuronen etc.
- Cross-Validation: Aufteilung der Daten in Trainings- und Validierungssets zur Vermeidung von Overfitting
- Modellvergleich: Mehrere Modelle werden trainiert und anhand ihrer Leistung verglichen

3.3 Geeigneten ML-Ansatzes anhand des Projektszenarios

Frage:

Ein Unternehmen möchte ein System entwickeln, das Kunden in Gruppen mit ähnlichem Kaufverhalten einteilt. Welcher ML-Ansatz ist hierfür am besten geeignet und warum?

Antwort:

Ein Unternehmen möchte ein System entwickeln, das Kunden in Gruppen mit ähnlichem Kaufverhalten einteilt. Für dieses Ziel ist der Machine-Learning-Ansatz des unüberwachten Lernens, insbesondere die Clusterbildung (Clustering), besonders geeignet.

1. Warum ist Clustering geeignet?

Clustering ist ein Verfahren des unüberwachten Lernens, bei dem Datenpunkte in Gruppen (Cluster) eingeteilt werden, sodass die Mitglieder eines Clusters möglichst ähnlich zueinander sind. Da bei der Kundensegmentierung keine vorab bekannten Gruppen (Labels) existieren, ist Clustering ideal, um Muster und Strukturen im Kaufverhalten automatisch zu erkennen.

2. Geeignete Clustering-Algorithmen

Für die Segmentierung von Kundenverhalten kommen verschiedene Clustering-Algorithmen in Frage:

- - K-Means: Effizient bei gut trennbaren Gruppen
- - DBSCAN: Robust gegenüber Ausreißern und geeignet für unregelmäßige Clusterformen
- - Hierarchisches Clustering: Liefert eine Baumstruktur zur Analyse von Ähnlichkeiten

3. Typische Anwendungsbeispiele

Clustering wird in vielen Bereichen der Kundenanalyse eingesetzt, z. B.:

- - Kundensegmentierung für gezieltes Marketing
- - Personalisierte Produktempfehlungen
- - Analyse von Kaufmustern und Produktbündelung
- - Identifikation potenziell abwandernder Kundengruppen

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

Frage:

Welche Faktoren beeinflussen die Auswahl eines geeigneten ML-Algorithmus für ein bestimmtes Projekt?

Antwort:

1. Art des Problems

- Klassifikation: z. B. Entscheidungsbäume, Random Forest, SVM, neuronale Netze
- Regression: z. B. Lineare Regression, Ridge, SVR
- Clustering: z. B. K-Means, DBSCAN
- Zeitreihenanalyse: z. B. ARIMA, LSTM
- Reinforcement Learning: z. B. Q-Learning, Deep Q-Networks

2. Datenmerkmale

- Größe des Datensatzes: Kleine Daten → einfache Modelle; große Daten → komplexere Modelle
- Anzahl der Merkmale: Hochdimensionale Daten → z. B. SVM, PCA zur Vorverarbeitung
- Vorhandensein von Rauschen oder Ausreißern: Robuste Modelle wie Random Forest oder Gradient Boosting bevorzugt

3. Rechenressourcen und Zeit

- Trainingszeit: Lineare Modelle sind schneller als tiefe neuronale Netze
- Verfügbarkeit von Hardware: GPU erforderlich für Deep Learning
- Echtzeitanforderungen: Modelle mit schneller Inferenzzeit bevorzugt

4. Zielsetzung und Interpretierbarkeit

- Interpretierbarkeit wichtig? → Entscheidungsbaum, lineare Modelle
- Genauigkeit wichtiger als Transparenz? → Ensemble-Methoden, Deep Learning

5. Verfügbarkeit von gelabelten Daten

- Vollständig gelabelt → Überwachtes Lernen
- Keine Labels → Unüberwachtes Lernen
- Teilweise gelabelt → Semi-überwachtes Lernen
- Interaktive Umgebung → Bestärkendes Lernen

6. Erfahrung und Expertise

- Vertrautheit mit bestimmten Algorithmen oder Frameworks kann die Auswahl beeinflussen

7. Zusätzliche Überlegungen

- Skalierbarkeit
- Robustheit gegenüber Datenänderungen
- Regulatorische Anforderungen (z. B. erklärbare KI)

3.5 Konzepte von Überanpassung und Unteranpassung

Frage:

Was versteht man unter Überanpassung und Unteranpassung in ML-Modellen, und wie wirken sich diese auf die Modellqualität aus?

Antwort:

In der Welt des maschinellen Lernens sind Überanpassung (Overfitting) und Unteranpassung (Underfitting) zwei häufige Probleme, die die Modellqualität stark beeinflussen können. Im Folgenden werden beide Konzepte erklärt, ihre Auswirkungen beschrieben sowie Ursachen und mögliche Lösungen aufgezeigt.

▲ Überanpassung (Overfitting)

Ein Modell ist überangepasst, wenn es die Trainingsdaten zu genau lernt, inklusive Zufälligkeiten und Rauschen, anstatt die zugrunde liegenden Muster zu erfassen.

Merkmale:

- Sehr gute Leistung auf Trainingsdaten
- Schlechte Leistung auf neuen, unbekannten Daten
- Geringe Generalisierungsfähigkeit

Ursachen:

- Zu komplexes Modell
- Zu wenig Trainingsdaten
- Zu viele Features

Lösungen:

- Regularisierung (z. B. L1/L2)
- Einfacheres Modell verwenden
- Mehr Trainingsdaten sammeln
- Cross-Validation einsetzen

▼ Unteranpassung (Underfitting)

Ein Modell ist unterangepasst, wenn es nicht in der Lage ist, die zugrunde liegenden Muster in den Daten zu erfassen – weder im Training noch im Test.

Merkmale:

- Schlechte Leistung auf Trainings- und Testdaten
- Modell ist zu einfach für die Komplexität des Problems

Ursachen:

- Zu einfaches Modell
- Zu wenig Trainingszeit
- Unzureichende Features

Lösungen:

- Komplexeres Modell verwenden
- Mehr oder bessere Features einführen
- Längeres Training

Vergleichstabelle

Problem	Trainingsleistung	Testleistung	Generalisierung
Überanpassung	Hoch	Niedrig	Schlecht
Unteranpassung	Niedrig	Niedrig	Schlecht
Gutes Modell	Hoch	Hoch	Gut

HO 3.5 (H0)

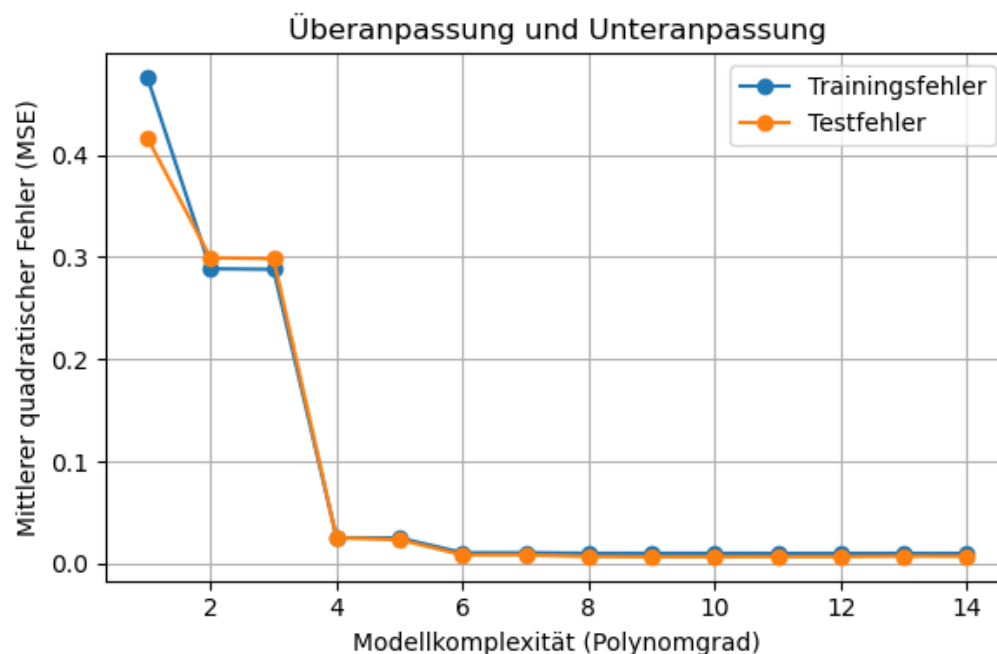
Demonstrieren von Überanpassung und Unteranpassung

Nutzen Sie ein einfaches ML-Modell, um zu zeigen, wie sich Überanpassung und Unteranpassung anhand der Modellleistung auf Trainings- und Testdaten erkennen lassen.

Überanpassung und Unteranpassung in ML-Modellen

Dieses Beispiel zeigt, wie sich Überanpassung (Overfitting) und Unteranpassung (Underfitting) auf die Modellleistung auswirken. Dazu wird ein einfaches Polynomialregressionsmodell verwendet, bei dem der Grad des Polynoms (Modellkomplexität) variiert wird. Die mittleren quadratischen Fehler (MSE) für Trainings- und Testdaten werden aufgezeichnet.

- Bei niedriger Komplexität (z. B. Grad 1–2) ist der Fehler auf beiden Datensätzen hoch – das Modell ist unterangepasst.
- Bei mittlerer Komplexität (z. B. Grad 4–6) sind beide Fehler niedrig – das Modell generalisiert gut.
- Bei hoher Komplexität (z. B. Grad >10) sinkt der Trainingsfehler stark, aber der Testfehler steigt – das Modell ist überangepasst.



Das Diagramm zeigt deutlich, wie sich die Modellkomplexität auf die Fehler auswirkt. Ein gutes Modell erreicht ein Gleichgewicht zwischen Bias und Varianz und minimiert den Testfehler.

4.0 Kapitel: ML – Daten

– K2:

Datenaufbereitung, Qualitätsprobleme, Einfluss auf Modelle;

– H2:

ML-Daten vorbereiten, Training/Validierung/Testdatensätze identifizieren.

4.1 Aktivitäten u. Herausforderungen mit Datenvorbereitung

Frage:

Welche typischen Schritte umfasst die Datenvorbereitung im ML-Workflow, und welche Herausforderungen können dabei auftreten?

Antwort:

Die **Datenvorbereitung** ist ein zentraler Bestandteil jedes Machine-Learning-Workflows und beeinflusst maßgeblich die Qualität und Leistungsfähigkeit eines Modells. Sie umfasst typischerweise folgende Schritte:

Typische Schritte der Datenvorbereitung

1. Datensammlung

- Sammeln von Rohdaten aus verschiedenen Quellen (z. B. Datenbanken, APIs, Sensoren, Web-Scraping).
- Sicherstellen, dass die Daten relevant und repräsentativ sind.

2. Datenbereinigung

- Entfernen oder Imputieren fehlender Werte.
- Korrigieren von Inkonsistenzen oder fehlerhaften Einträgen.
- Entfernen von Duplikaten.

3. Datenintegration

- Zusammenführen von Daten aus mehreren Quellen.
- Vereinheitlichung von Formaten und Strukturen.

4. Datenformatierung und -transformation

- Skalierung oder Normalisierung numerischer Werte.
- Kodierung kategorialer Variablen (z. B. One-Hot-Encoding).
- Zeitreihen- oder Textdaten in geeignete Form bringen.

5. Feature Engineering

- Erstellen neuer Merkmale aus bestehenden Daten.
- Auswahl relevanter Features (Feature Selection).
- Reduktion der Dimensionalität (z. B. PCA).

6. Datenaufteilung

- Aufteilen in Trainings-, Validierungs- und Testdaten.
- Sicherstellen, dass die Verteilung repräsentativ bleibt (z. B. durch Stratified Sampling).

Typische Herausforderungen

1. Fehlende oder unvollständige Daten

- Kann zu Verzerrungen führen, wenn nicht korrekt behandelt.

2. Datenqualität

- Rauschen, Inkonsistenzen oder fehlerhafte Labels können die Modellleistung stark beeinträchtigen.

3. Ungleichgewicht in den Klassen

- Besonders problematisch bei Klassifikationsaufgaben (z. B. Fraud Detection).

4. Skalierbarkeit

- Große Datenmengen erfordern effiziente Verarbeitung und Speicherlösungen.

5. Bias und Fairness

- Verzerrte Daten können zu unfairen oder diskriminierenden Modellen führen.

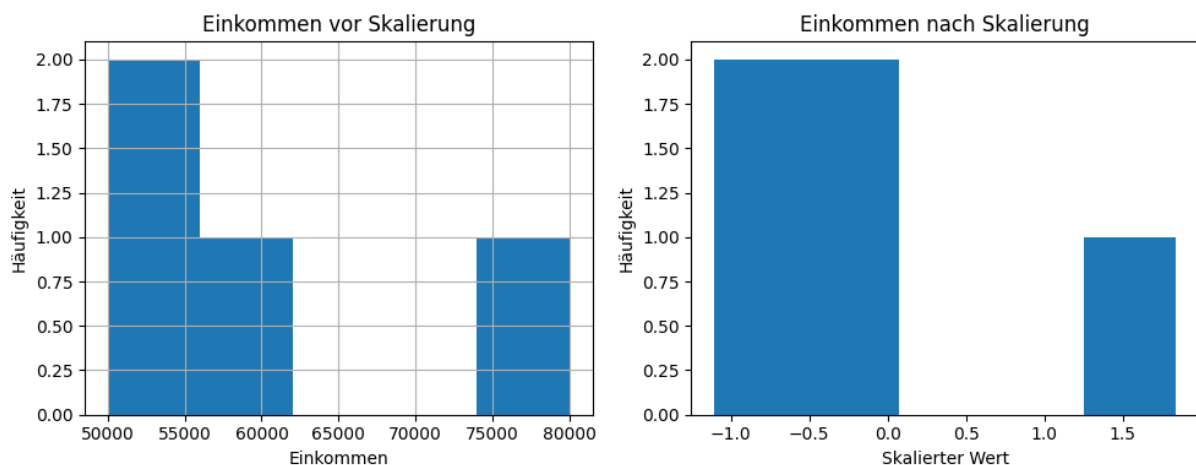
6. Feature Leakage

- Wenn Informationen aus der Zukunft oder aus dem Zielwert unbeabsichtigt in die Features gelangen.

Beispiel:

Hier siehst du ein einfaches Beispiel für die Datenvorbereitung im Machine Learning mit Python. Es umfasst:

1. **Erstellen eines Beispiel-Datensatzes** mit numerischen und kategorialen Variablen.
2. **Bereinigung fehlender Werte:**
 - Numerisch: Mittelwert-Imputation.
 - Kategorial: Häufigster Wert.
3. **Skalierung** der numerischen Features mit StandardScaler.
4. **One-Hot-Encoding** der kategorialen Variablen.
5. **Visualisierung** der Verteilung des Features „Einkommen“ vor und nach der Skalierung:



Diese Schritte sind typisch für viele ML-Projekte und helfen, die Daten in eine Form zu bringen, die für Modelle wie Entscheidungsbäume, lineare Regression oder neuronale Netze geeignet ist.

Möchtest du auch sehen, wie man diese vorbereiteten Daten in ein Modell einspeist, z. B. für eine Klassifikation oder Regression?

HO 4.1 (H2) Datenvorbereitung zur Unterstützung der Erstellung eines ML-Modells

Aufgabe:

Bereiten Sie einen gegebenen Rohdatensatz so auf, dass er für das Training eines Klassenmodells im überwachten Lernen geeignet ist. Dokumentieren Sie Ihre Schritte.

Lösung:

Schritte der Datenvorbereitung

1. **Laden des Datensatzes**
→ In diesem Fall wurde ein Beispiel-Datensatz manuell erstellt.
2. **Trennung von Merkmalen und Zielvariable**
 - Merkmale: Alter, Einkommen, Geschlecht, Beruf
 - Zielvariable: Kauft (0 = nein, 1 = ja)
3. **Unterscheidung von numerischen und kategorialen Features**
 - Numerisch: Alter, Einkommen
 - Kategorial: Geschlecht, Beruf
4. **Vorverarbeitungspipelines**
 - **Numerisch:**
 - Fehlende Werte → Mittelwert
 - Skalierung → StandardScaler
 - **Kategorial:**
 - Fehlende Werte → Häufigster Wert
 - Kodierung → One-Hot-Encoding
5. **Kombination mit ColumnTransformer**
→ Führt beide Pipelines gleichzeitig aus.
6. **Transformation der Daten**
→ Die Daten wurden in ein numerisches Format überführt, das für ML-Modelle geeignet ist.
7. **Aufteilung in Trainings- und Testdaten**
 - 80 % Training, 20 % Test
 - Zufällige Aufteilung mit random_state=42 für Reproduzierbarkeit

Beispielausgabe (gekürzt)

- **Trainingsdaten (X_train):**
 - `[[1.5 1.41 0. 0. 1. 0. 0. 1.]`
 - `[0.5 0.00 0. 1. 0. 1. 0. 0.]`
 - ...]
- **Testdaten (X_test):**
 - `[[-0.5 -0.71 0. 1. 0. 0. 0. 1.]]`
- **Trainingslabels (y_train):** `[1 0 0 1]`
- **Testlabels (y_test):** `[1]`

4.2 Vergleich von Trainings-, Validierungs- und Testdatensätze bei der Entwicklung eines ML-Modells

Frage:

Welche Rolle spielen Trainings-, Validierungs- und Testdatensätze im ML-Workflow, und warum ist deren Trennung wichtig?

Antwort:

Im Machine Learning ist die **Trennung von Trainings-, Validierungs- und Testdatensätzen** entscheidend für die Entwicklung robuster und verlässlicher Modelle. Hier ist eine Übersicht über ihre jeweilige Rolle und warum sie wichtig sind:

1. Trainingsdatensatz

- **Zweck:** Wird verwendet, um das Modell zu trainieren – also die Parameter (z. B. Gewichte in einem neuronalen Netz) zu lernen.
- **Typischer Anteil:** 60–80 % der Gesamtdaten.
- **Wichtig:** Das Modell „sieht“ diese Daten während des Lernprozesses.

2. Validierungsdatensatz

- **Zweck:** Dient zur **Modellauswahl** und **Hyperparameter-Optimierung** (z. B. Tiefe eines Entscheidungsbaums, Lernrate).
- **Typischer Anteil:** 10–20 % der Gesamtdaten.
- **Wichtig:** Das Modell wird **nicht** auf diesen Daten trainiert, sondern nur bewertet, um z. B. Overfitting zu erkennen.

3. Testdatensatz

- **Zweck:** Wird **einmalig** verwendet, um die finale Modellleistung zu beurteilen.
- **Typischer Anteil:** 10–20 %.
- **Wichtig:** Der Testdatensatz simuliert echte, **unbekannte Daten**. Er darf **niemals** zur Modelloptimierung verwendet werden.

Warum ist die Trennung wichtig?

Grund	Erklärung
Vermeidung von Overfitting	Ohne Validierungsdaten könnte das Modell zu stark an die Trainingsdaten angepasst werden.
Objektive Bewertung	Nur ein separater Testdatensatz erlaubt eine realistische Einschätzung der Modellleistung.
Modellauswahl	Verschiedene Modelle oder Einstellungen können mit Validierungsdaten verglichen werden.
Vermeidung von Datenleckage	Wenn Informationen aus dem Testset ins Training gelangen, ist die Bewertung verzerrt.

HO 4.2(H2) Identifizieren von Trainings- und Testdatensätze und Erstellen eines ML-Models

Aufgabe:

Teilen Sie einen vorbereiteten Datensatz in Trainings-, Validierungs- und Testdatensatz auf. Trainieren Sie ein einfaches Klassenmodell und vergleichen Sie die Genauigkeit auf den verschiedenen Datensätze.

Lösung:

Das Klassifikationsmodell (Entscheidungsbaum) wurde erfolgreich auf dem **Iris-Datensatz** trainiert und auf drei getrennten Datensätzen bewertet. Hier sind die Ergebnisse:

Genauigkeiten im Vergleich

- **Trainingsdatensatz:** 1.00
→ Das Modell hat alle Trainingsbeispiele korrekt klassifiziert (möglicherweise leichtes Overfitting).
- **Validierungsdatensatz:** 0.93
→ Gute Leistung auf unbekannten Daten, die zur Modellabstimmung verwendet werden könnten.
- **Testdatensatz:** 0.93
→ Das Modell generalisiert gut auf völlig neue Daten.

Fazit

Die Trennung in **Train**, **Validation** und **Test** erlaubt es:

- das Modell zu trainieren (Train),
- Hyperparameter zu optimieren (Validation),
- und die finale Leistung objektiv zu bewerten (Test).

Möchtest du auch sehen, wie sich die Modellleistung bei einem anderen Klassifikator (z. B. k-NN oder Random Forest) verändert?

4.3 Typische Qualitätsprobleme bei Datensätze

Frage:

Welche typischen Qualitätsprobleme können in ML-Datensätze auftreten, und wie wirken sich diese auf das Modelltraining aus?

Antwort:

In Machine-Learning-Projekten können **Qualitätsprobleme in den Datensätzen** erhebliche Auswirkungen auf die Modellleistung, Generalisierbarkeit und Fairness haben. Hier sind die häufigsten Probleme und ihre potenziellen Folgen:

Typische Qualitätsprobleme in ML-Datensätzen

Problemtyp	Beschreibung	Auswirkungen
Fehlende Werte	Leere Felder in numerischen oder kategorialen Spalten	Verzerrte Modelle, Fehler beim Training
Ausreißer	Extremwerte, die nicht zur restlichen Verteilung passen	Überanpassung oder instabile Modelle
Rauschen	Zufällige Fehler oder Ungenauigkeiten in den Daten	Reduzierte Modellgenauigkeit
Unbalancierte Klassen	Eine Klasse dominiert stark (z. B. 95 % vs. 5 %)	Modell bevorzugt Mehrheitsklasse, schlechte Recall für Minderheitsklasse
Datenleckage (Data Leakage)	Informationen aus der Zukunft oder Zielvariable in den Features	Künstlich hohe Genauigkeit, unbrauchbares Modell
Redundante oder irrelevante Features	Merkmale, die keinen Mehrwert bieten oder stark korreliert sind	Erhöhte Komplexität, Overfitting
Falsche Labels	Fehlerhafte Zielwerte (z. B. durch manuelle Annotation)	Modell lernt falsche Zusammenhänge
Inkonsistente Formate	Unterschiedliche Einheiten, Schreibweisen oder Kodierungen	Fehler bei der Verarbeitung oder Modellinput
Bias in den Daten	Verzerrungen durch nicht-repräsentative Stichproben	Diskriminierende oder unfair generalisierende Modelle

Wie wirken sich diese Probleme aus?

- **Modelleleistung sinkt:** Ungenaue oder verrauschte Daten führen zu schlechteren Vorhersagen.
- **Generalisation leidet:** Das Modell funktioniert gut auf Trainingsdaten, aber schlecht auf neuen Daten.
- **Fehlende Fairness:** Verzerrte Daten können zu diskriminierenden Entscheidungen führen.
- **Fehlende Reproduzierbarkeit:** Inkonsistente oder schlecht dokumentierte Daten erschweren die Wiederverwendung.

4.4 Erkennen, wie schlechte Datenqualität zu Problemen und dem resultierenden ML-Modell führen kann

Frage:

Wie kann sich schlechte Datenqualität auf die Genauigkeit, Fairness oder Sicherheit eines ML-Modells auswirken?

Antwort:

Schlechte Datenqualität kann sich **massiv negativ** auf die Leistung und Vertrauenswürdigkeit eines Machine-Learning-Modells auswirken. Hier sind die drei zentralen Dimensionen, die betroffen sind:

1. Genauigkeit (Accuracy)

Ursachen:

- Fehlende oder fehlerhafte Labels
- Rauschen oder Ausreißer in den Features
- Irrelevante oder redundante Merkmale

Auswirkungen:

- Das Modell lernt falsche Zusammenhänge.
- Es generalisiert schlecht auf neue Daten.
- Die Vorhersagegenauigkeit sinkt deutlich.

Beispiel: Ein Spam-Filter, der mit falsch gelabelten E-Mails trainiert wurde, erkennt Spam nicht zuverlässig.

2. Fairness

Ursachen:

- Verzerrte oder nicht-repräsentative Trainingsdaten
- Historische Diskriminierung in den Daten
- Ungleichgewicht in den Klassen (z. B. 90 % Männer, 10 % Frauen)

Auswirkungen:

- Das Modell bevorzugt bestimmte Gruppen.
- Diskriminierende Entscheidungen (z. B. bei Kreditvergabe, Bewerbungen).
- Vertrauensverlust und rechtliche Risiken.

Beispiel: Ein Bewerbungsfilter bevorzugt Bewerber mit typisch männlichen Namen, weil das Trainingsset überwiegend Männer enthielt.

3. Sicherheit (Robustheit & Angreifbarkeit)

Ursachen:

- Rauschanfällige oder leicht manipulierbare Features
- Fehlende Validierung gegen adversariale Eingaben
- Datenleckage (z. B. Zielvariable in den Features)

Auswirkungen:

- Das Modell kann durch gezielte Eingaben manipuliert werden.
- Es reagiert instabil auf kleine Änderungen.
- Vertrauliche Informationen könnten ungewollt preisgegeben werden.

Beispiel: Ein Bilderkennungsmodell erkennt ein Stoppschild nicht mehr, wenn ein Sticker darauf klebt.

4.5 Verschiedene Ansätze zur Kennzeichnung von Daten in Datensätze für überwachtes Lernen

Frage:



Welche Ansätze werden zur Datenkennzeichnung im überwachten Lernen verwendet?

Antwort:



Im überwachten Lernen ist die **Datenkennzeichnung (Labeling)** entscheidend, da Modelle nur dann lernen können, wenn sie wissen, welches Ergebnis zu welchem Input gehört. Es gibt verschiedene Ansätze zur Datenkennzeichnung – je nach Datenart, Ressourcen und Anwendungsfall:

Typische Ansätze zur Datenkennzeichnung



1. Manuelle Kennzeichnung

- **Beschreibung:** Menschen (z. B. Experten, Crowdworker) versehen Daten mit Labels.
- **Beispiel:** Ein Radiologe markiert Tumore auf Röntgenbildern.
-  **Vorteil:** Hohe Genauigkeit
-  **Nachteil:** Zeit- und kostenintensiv



2. Crowdsourcing

- **Beschreibung:** Viele nicht-expertengestützte Annotatoren (z. B. über Amazon Mechanical Turk) labeln Daten.
- **Beispiel:** Nutzer markieren Bilder als „Katze“ oder „Hund“.
-  **Vorteil:** Skalierbar
-  **Nachteil:** Qualität schwankt → oft Mehrfachlabelung nötig



3. Semi-automatisches Labeling

- **Beschreibung:** Ein Modell schlägt Labels vor, die dann von Menschen überprüft werden.
- **Beispiel:** Ein Textklassifikator markiert E-Mails als Spam, ein Mensch bestätigt oder korrigiert.
-  **Vorteil:** Effizienter als rein manuell
-  **Nachteil:** Fehler des Modells können sich fortpflanzen



4. Selbstüberwachtes Lernen (Self-supervised)

- **Beschreibung:** Labels werden automatisch aus den Daten selbst generiert.
- **Beispiel:** Bei Texten wird ein Wort entfernt und das Modell soll es vorhersagen.
-  **Vorteil:** Keine manuelle Annotation nötig
-  **Nachteil:** Nur für bestimmte Aufgaben geeignet

5. Schwaches Lernen (Weak Supervision)

- **Beschreibung:** Labels werden aus Heuristiken, Regeln oder externen Quellen abgeleitet.
- **Beispiel:** Wenn ein Tweet „glücklich“ enthält → Label = positiv.
-  **Vorteil:** Schnell große Mengen labeln
-  **Nachteil:** Labels sind oft ungenau

6. Aktives Lernen

- **Beschreibung:** Das Modell wählt gezielt die Beispiele aus, bei denen es am unsichersten ist, und bittet um menschliche Annotation.
-  **Vorteil:** Minimiert den Labelaufwand
-  **Nachteil:** Komplexere Infrastruktur nötig

4.6 Gründe für falsche Kennzeichnung der Daten in Datensätze

Frage:

Welche Ursache können zu falsch gekennzeichneten Daten in einem ML-Datensatz führen?

Antwort:

Falsch gekennzeichnete Daten (auch als **Labeling Errors** bekannt) sind ein häufiges Problem im überwachten Lernen und können die Modellleistung erheblich beeinträchtigen. Hier sind die **häufigsten Ursachen** dafür:

Ursachen für falsch gekennzeichnete Daten

Ursache	Beschreibung	Beispiel
Menschliche Fehler	Annotatoren machen Fehler durch Unachtsamkeit, Müdigkeit oder Missverständnisse.	Ein Bild mit einer Katze wird als Hund gelabelt.
Uneinheitliche Richtlinien	Unterschiedliche Annotatoren interpretieren die Labelkriterien unterschiedlich.	„Neutral“ vs. „positiv“ bei Sentiment-Analyse.
Mehrdeutige Daten	Die Daten lassen sich nicht eindeutig einer Klasse zuordnen.	Ein Text enthält sowohl Lob als auch Kritik.
Automatisches oder heuristisches Labeling	Labels werden durch Regeln oder Modelle erzeugt, die ungenau sind.	Tweets mit „glücklich“ werden automatisch als positiv markiert.
Veraltete Labels	Labels spiegeln nicht mehr den aktuellen Zustand wider.	Ein Produkt war früher „verfügbar“, ist es aber nicht mehr.

Datenverwechslung	Datenpunkte werden falsch zugeordnet oder vertauscht.	Ein Label gehört zu einem anderen Datensatz.
Bias in der Annotation	Persönliche Vorurteile der Annotatoren beeinflussen die Labelvergabe.	Geschlecht oder Herkunft beeinflusst die Bewertung.
Technische Fehler	Fehler beim Speichern, Exportieren oder Zusammenführen von Daten.	Labels werden durch ein Skript falsch zugewiesen.

Folgen für das Modelltraining

- **Reduzierte Genauigkeit:** Das Modell lernt auf Basis falscher Informationen.
- **Verzernte Entscheidungsgrenzen:** Besonders kritisch bei kleinen oder unbalancierten Datensätzen.
- **Schlechte Generalisierung:** Das Modell performt schlecht auf echten, korrekt gelabelten Daten.
- **Verlust von Vertrauen:** Besonders in sicherheitskritischen oder ethisch sensiblen Anwendungen.

5.0 Kapitel: Funktionale Leistungsmetriken von ML

– K4:

Metriken wie Confusion Matrix, F1-Score, MSE, R^2 auswählen/begründen;

– K2:

Benchmarks erklären;

– H2:

Modell-Evaluation mit Metriken durchführen.

5.1 Berechnung der funktionalen Leistungsmetriken von ML

Frage:

Berechnen Sie die funktionalen Leistungsmetriken von ML aus einem gegebenen Satz von Konfusionsmatrix-Daten

Gegeben ist folgende Konfusionsmatrix:

- Richtig Positiv (RP): 80
- Falsch Positiv (FP): 10
- Richtig Negativ (RN): 50
- Falsch Negativ (FN): 20

Berechnen Sie folgenden Metriken:

- Genauigkeit
- Präzision
- Sensitivität
- F1-Wert

Antwort:

Berechnung der funktionalen Leistungsmetriken

Gegeben ist die Konfusionsmatrix:

- **Richtig Positiv (RP): 80**
- **Falsch Positiv (FP): 10**
- **Richtig Negativ (RN): 50**
- **Falsch Negativ (FN): 20**

1. Genauigkeit (Accuracy)

Die Genauigkeit misst den Anteil der korrekt klassifizierten Instanzen.

Formel:

$$\left[\text{Accuracy} = \frac{\text{RP} + \text{RN}}{\text{RP} + \text{FP} + \text{RN} + \text{FN}} \right]$$

Berechnung:

$$\left[\text{Accuracy} = \frac{80 + 50}{80 + 10 + 50 + 20} = \frac{130}{160} = 0,8125 = 81,25 \% \right]$$

2. Präzision (Precision)

Die Präzision gibt an, wie viele der als positiv klassifizierten Instanzen tatsächlich positiv sind.

Formel:

$$\text{Precision} = \frac{RP}{RP + FP}$$

Berechnung:

$$\text{Precision} = \frac{80}{80 + 10} = \frac{80}{90} \approx 0,8889 = 88,89 \%$$

3. Sensitivität (Recall oder True Positive Rate)

Die Sensitivität misst den Anteil der korrekt erkannten positiven Instanzen.

Formel:

$$\text{Sensitivity} = \frac{RP}{RP + FN}$$

Berechnung:

$$\text{Sensitivity} = \frac{80}{80 + 20} = \frac{80}{100} = 0,8 = 80 \%$$

4. F1-Wert (F1-Score)

Der F1-Wert ist das harmonische Mittel von Präzision und Sensitivität.

Formel:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}}$$

Berechnung:

$$F1 = 2 \cdot \frac{0,8889 \cdot 0,8}{0,8889 + 0,8} = 2 \cdot \frac{0,7111}{1,6889} \approx 0,841 = 84,1 \%$$

Zusammenfassung der Ergebnisse

- **Genauigkeit:** 81,25 %
- **Präzision:** 88,89 %
- **Sensitivität:** 80 %
- **F1-Wert:** 84,1 %

5.2 Gegenüber stellen und Vergleichen der Konzepte hinter den funktionalen Leistungsmetriken von ML-Methoden.

Frage:

Welche funktionalen Leistungsmetriken eignen sich jeweils für:

- Klassifikation
- Regression
- Clusterbildung

Wie unterscheiden sich diese Metriken in ihrer Aussagekraft.

Antwort:

Für verschiedene maschinelle Lernaufgaben eignen sich unterschiedliche Leistungsmetriken:

1. Klassifikation

- **Genauigkeit (Accuracy):** Anteil korrekt klassifizierter Instanzen.
- **Präzision und Recall:** Wichtig bei unausgeglichenen Klassen.
- **F1-Score:** Harmonie von Präzision und Recall.
- **ROC-AUC:** Bewertung der Trennschärfe von Modellen.

Aussagekraft: Gibt an, wie gut ein Modell Klassen unterscheidet. F1-Score und AUC sind robuster bei Klassenungleichgewicht.

2. Regression

- **MSE (Mean Squared Error):** Quadratischer Fehler, betont große Abweichungen.
- **RMSE (Root Mean Squared Error):** Interpretierbar in Origineleinheiten.
- **MAE (Mean Absolute Error):** Durchschnittlicher absoluter Fehler.
- **R² (Bestimmtheitsmaß):** Anteil erklärter Varianz.

Aussagekraft: MSE/RMSE betonen große Fehler, MAE ist robuster gegen Ausreißer. R² zeigt das Erklärungsvermögen des Modells.

3. Clusterbildung

- **Silhouette-Score:** Misst, wie gut Cluster voneinander getrennt sind.
- **Davies-Bouldin-Index:** Bewertet die Kompaktheit und Trennung von Clustern.
- **Dunn-Index:** Maximiert den minimalen Abstand zwischen Clustern relativ zur Clustergröße.

Aussagekraft: Bewertet die Qualität der Clusterstruktur. Silhouette-Score ist intuitiv, Dunn-Index betont klare Trennung.

Fazit

Die Wahl der Metrik hängt von der Aufgabenstellung und den Daten ab. Während in der Klassifikation Fehlklassifikationen im Fokus stehen, geht es bei Regression um Abweichungen von Vorhersagen und bei Clustering um die Struktur der Gruppen.

5.3 Beschränkungen von funktionalen Leistungsmetriken

Frage:

Warum reichen funktionale Leistungsmetriken alleine nicht aus, um die Qualität eines ML-Systems umfassend zu bewerten?

Antwort:

Funktionale Leistungsmetriken, wie Genauigkeit, Präzision, Recall oder F1-Score, bieten zwar wertvolle Einblicke in die Leistung von Machine-Learning-Systemen in spezifischen Testszenarien, doch sie allein reichen nicht aus, um die Qualität eines ML-Systems umfassend zu bewerten. Dies liegt an mehreren Faktoren:

Robustheit:

ML-Modelle müssen auch unter veränderten Bedingungen oder bei Daten, die von den Trainingsdaten abweichen, verlässlich funktionieren. Funktionale Metriken erfassen oft nicht, wie robust ein Modell gegenüber Störungen oder adversarialen Angriffen ist.

Fairness und Bias:

Ein Modell kann hohe Genauigkeitswerte erzielen, aber dennoch systematische Verzerrungen aufweisen, die bestimmte Gruppen benachteiligen. Solche ethischen Aspekte erfordern spezialisierte Analysen, die über standardisierte Leistungsmetriken hinausgehen.

Erklärbarkeit:

In vielen Anwendungsbereichen ist es entscheidend zu verstehen, warum ein Modell eine bestimmte Entscheidung trifft. Funktionale Metriken liefern keine Informationen über die Interpretierbarkeit oder Nachvollziehbarkeit der Entscheidungen eines Modells.

Effizienz und Skalierbarkeit:

Aspekte wie Rechenzeit, Speicherbedarf oder Energieverbrauch sind für den praktischen Einsatz von ML-Systemen relevant, werden aber von funktionalen Metriken nicht abgedeckt.

Anpassungsfähigkeit:

Die Fähigkeit eines Systems, sich an veränderte Datenstrukturen oder neue Anforderungen anzupassen, ist für den langfristigen Einsatz von Bedeutung. Auch dies wird von klassischen Metriken nicht berücksichtigt.

Um die Qualität eines ML-Systems umfassend zu bewerten, ist daher ein ganzheitlicher Ansatz notwendig, der neben den funktionalen Leistungsmetriken auch Aspekte wie Robustheit, Fairness, Erklärbarkeit und Effizienz einbezieht.

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

Frage:

Ein ML-Modell soll zur Erkennung von Kreditkartenbetrug eingesetzt werden. Welche funktionalen Leistungsmetriken sind in diesem Fall besonders wichtig und warum?

Antwort:

Für ein Machine-Learning-Modell zur Erkennung von Kreditkartenbetrug sind folgende funktionale Leistungsmetriken besonders wichtig:

- **Präzision (Precision):** Sie misst den Anteil der tatsächlich betrügerischen Transaktionen an allen als betrügerisch klassifizierten Transaktionen. Eine hohe Präzision ist wichtig, um die Anzahl von Fehllarmen (False Positives) zu minimieren, da unschuldige Kunden sonst fälschlicherweise blockiert werden könnten.

- **Sensitivität (Recall) bzw. True Positive Rate:** Sie zeigt, wie viele der tatsächlich betrügerischen Transaktionen korrekt erkannt werden. Ein hoher Recall ist entscheidend, um möglichst viele Betrugsfälle zu identifizieren.
- **F1-Score:** Der F1-Score ist das harmonische Mittel aus Präzision und Recall und bietet ein ausgewogenes Maß zwischen beiden. Er ist besonders nützlich, wenn ein Gleichgewicht zwischen der Erkennung von Betrugsfällen und der Minimierung von Fehlalarmen notwendig ist.
- **ROC-AUC (Receiver Operating Characteristic - Area Under Curve):** Diese Metrik bewertet die Trennschärfe des Modells zwischen betrügerischen und legitimen Transaktionen. Ein hoher AUC-Wert zeigt, dass das Modell effektiv zwischen den beiden Klassen unterscheiden kann.
- **False Positive Rate (FPR):** Da zu viele Fehlalarme die Kundenzufriedenheit beeinträchtigen können, ist es wichtig, die FPR im Auge zu behalten.
- **Kostenbasierte Metriken:** Da der finanzielle Schaden durch verpasste Betrugsfälle oft höher ist als der durch Fehlalarme, können kostenbasierte Metriken helfen, das Modell an die wirtschaftlichen Prioritäten des Unternehmens anzupassen.

Fazit: Die Kombination aus Präzision, Recall, F1-Score und AUC-ROC ermöglicht eine umfassende Bewertung des Modells in Bezug auf die Anforderungen der Betrugserkennung. Besonders wichtig ist der Fokus auf einen hohen Recall, um möglichst viele Betrugsfälle zu erkennen, ohne dabei die Präzision zu vernachlässigen.

HO 5.4 Evaluierendes erstellten ML-Modells anhand ausgewählter funktionaler Leistungsmetriken von ML

Aufgabe:

Nutzen Sie ein trainiertes Klassifikationsmodell und berechnen Sie die Metriken Genauigkeit, Präzision, Sensitivität und F1-Wert. Interpretieren Sie die Ergebnisse im Hinblick auf die Modellqualität

Lösung:

Um die Qualität eines trainierten Klassifikationsmodells zu bewerten, werden folgende Metriken berechnet:

- **Genauigkeit (Accuracy):** Sie misst den Anteil der korrekt klassifizierten Instanzen an der Gesamtzahl der Instanzen. Eine hohe Genauigkeit deutet darauf hin, dass das Modell insgesamt gut vorhersagt. Allerdings kann diese Metrik bei unausgeglichene Datensätzen irreführend sein.
- **Präzision (Precision):** Sie gibt an, wie viele der als positiv klassifizierten Instanzen tatsächlich positiv sind. Eine hohe Präzision ist wichtig, wenn falsche Positive vermieden werden sollen.
- **Sensitivität (Recall):** Auch als Trefferquote bekannt, misst sie den Anteil der korrekt identifizierten positiven Instanzen. Hohe Sensitivität ist entscheidend, wenn es wichtig ist, möglichst alle relevanten Fälle zu erfassen.

- **F1-Wert:** Dies ist das harmonische Mittel von Präzision und Sensitivität. Der F1-Wert ist besonders nützlich, wenn ein Gleichgewicht zwischen Präzision und Sensitivität erforderlich ist.

Interpretation:

- Ein Modell mit hoher Genauigkeit, Präzision und Sensitivität zeigt eine gute Leistung.
- Diskrepanzen zwischen Präzision und Sensitivität können auf ein Ungleichgewicht im Modell hinweisen.
- Der F1-Wert hilft, dieses Gleichgewicht zu bewerten, insbesondere bei unausgeglichene Datensätzen.

Die Kombination dieser Metriken ermöglicht eine umfassende Beurteilung der Modellqualität.

5.5 Erläutern Sie die Verwendung von Benchmark-Suiten im Zusammenhang mit ML

Frage:

Welche Rolle spielen Benchmark-Suiten im maschinellen Lernen, und wie können sie zur Bewertung von Modellen und Frameworks beitragen?

Antwort:

Benchmark-Suiten spielen eine entscheidende Rolle im Bereich des maschinellen Lernens, da sie standardisierte Testszenarien bieten, mit denen die Leistung von Modellen und Frameworks objektiv bewertet werden kann. Sie umfassen typischerweise eine Vielzahl von Datensätzen, Aufgaben und Metriken, die unterschiedliche Aspekte wie Genauigkeit, Effizienz und Skalierbarkeit messen.

Durch den Einsatz von Benchmark-Suiten können Entwickler und Forscher die Stärken und Schwächen verschiedener Modelle identifizieren. Dies erleichtert den Vergleich von Algorithmen unter kontrollierten Bedingungen und ermöglicht es, Fortschritte in der Entwicklung von Machine-Learning-Technologien nachzuverfolgen.

Zudem fördern Benchmark-Suiten die Reproduzierbarkeit von Ergebnissen, da sie eine einheitliche Grundlage für Experimente bieten. Dies ist besonders wichtig in der wissenschaftlichen Forschung, wo die Nachvollziehbarkeit von Ergebnissen von hoher Bedeutung ist.

Schließlich dienen Benchmark-Suiten auch als Anreiz für Innovationen, da sie Wettbewerbe und Herausforderungen ermöglichen, bei denen Forscher ihre Modelle optimieren, um die besten Ergebnisse zu erzielen.

6.0 Kapitel: ML – Neuronale Netzwerke und Testen

– K2:

Aufbau neuronaler Netze (DNN, Perceptron), Coverage-Metriken beschreiben;

– H1:

Perceptron implementieren.

6.1 Erläutern der Struktur und Funktion eines neuronalen Netzwerks, einschließlich eines DNN (Deep neural network)

Frage:

Wie ist ein tiefes neuronales Netzwerk (DNN) aufgebaut, und welche Rollen spielen Aktivierungsfunktionen, Gewichte und Bias-Werte in der Verarbeitung von Eingaben?

Antwort:

Ein **tiefes neuronales Netzwerk (Deep Neural Network, DNN)** ist ein künstliches neuronales Netzwerk mit mehreren Schichten zwischen Eingabe- und Ausgabeschicht. Es ist in der Lage, komplexe Muster und Zusammenhänge in Daten zu erkennen. Hier ist ein Überblick über den Aufbau und die Rolle der wichtigsten Komponenten: c

Aufbau eines DNN

Ein typisches DNN besteht aus:

1. **Eingabeschicht (Input Layer)**
 - Nimmt die Rohdaten entgegen (z. B. Pixelwerte eines Bildes, numerische Merkmale).
2. **Verborgene Schichten (Hidden Layers)**
 - Eine oder mehrere Schichten, in denen die eigentliche Verarbeitung stattfindet.
 - Jede Schicht besteht aus Neuronen, die mit den Neuronen der vorherigen Schicht verbunden sind.
3. **Ausgabeschicht (Output Layer)**
 - Gibt das Ergebnis der Verarbeitung aus (z. B. Klassenzugehörigkeit, Regressionswert).

Wichtige Komponenten und ihre Rollen

1. Gewichte (Weights)

- Jede Verbindung zwischen zwei Neuronen hat ein Gewicht.
- Sie bestimmen, wie stark ein Eingangssignal das Ausgangssignal beeinflusst.
- Während des Trainings werden die Gewichte angepasst, um die Fehler zu minimieren.

2. Bias-Werte (Biases)

- Jedes Neuron hat zusätzlich einen Bias-Wert.
- Dieser verschiebt die Aktivierung und hilft dem Modell, komplexere Muster zu lernen.
- Mathematisch:

$$z = \sum (w_i \cdot x_i) + b$$

wobei w_i die Gewichte, x_i die Eingaben und b der Bias ist.

3. Aktivierungsfunktionen

- Sie führen Nichtlinearität ein, damit das Netzwerk komplexe Funktionen lernen kann.
- Ohne sie wäre das Netzwerk nur eine lineare Funktion.
- Beliebte Funktionen:
 - **ReLU (Rectified Linear Unit):** $f(x) = \max(0, x)$
 - **Sigmoid:** $f(x) = \frac{1}{1 + e^{-x}}$
 - **Tanh:** $f(x) = \tanh(x)$
 - **Softmax:** Für Klassifikation, wandelt Ausgaben in Wahrscheinlichkeiten um.

Verarbeitung von Eingaben

1. Vorwärtsthroughlauf (Forward Pass):

- Eingaben werden durch die Schichten propagiert.
- Jede Schicht berechnet:
 $a(l) = \sigma(W(l) \cdot a(l-1) + b(l))$ wobei σ die Aktivierungsfunktion ist.

2. Rückwärtsthroughlauf (Backpropagation):

- Fehler wird berechnet (z. B. Differenz zur Zielausgabe).
- Gewichte und Biases werden angepasst, um den Fehler zu minimieren (z. B. mit Gradientenabstieg).

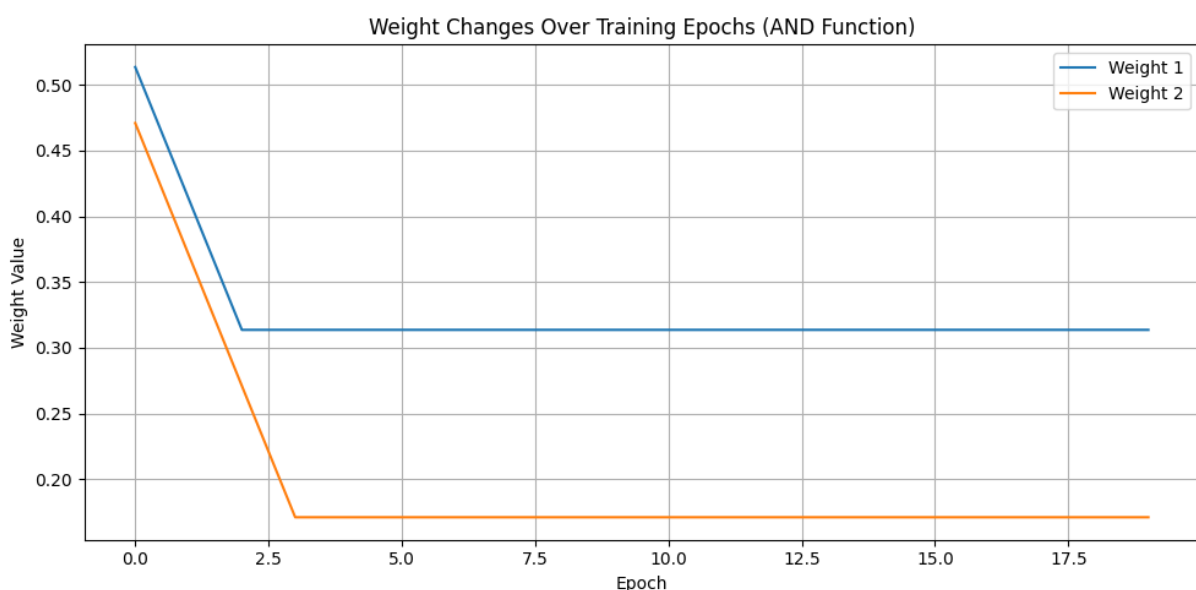
HO 6.1 Beobachten der Implementierung eines Perzeptrons

Aufgabe:

Beobachten Sie, wie ein einfaches Perzeptron eine logische Funktion (z.B. UND-Funktion) lernt. Welche Veränderungen der Gewichte lassen sich über mehrere Trainings-Epochen hinweg beobachten?

Lösung:

Hier ist die Visualisierung der **Gewichtsveränderungen eines einfachen Perzeptrons**, dass die **logische UND-Funktion** lernt:



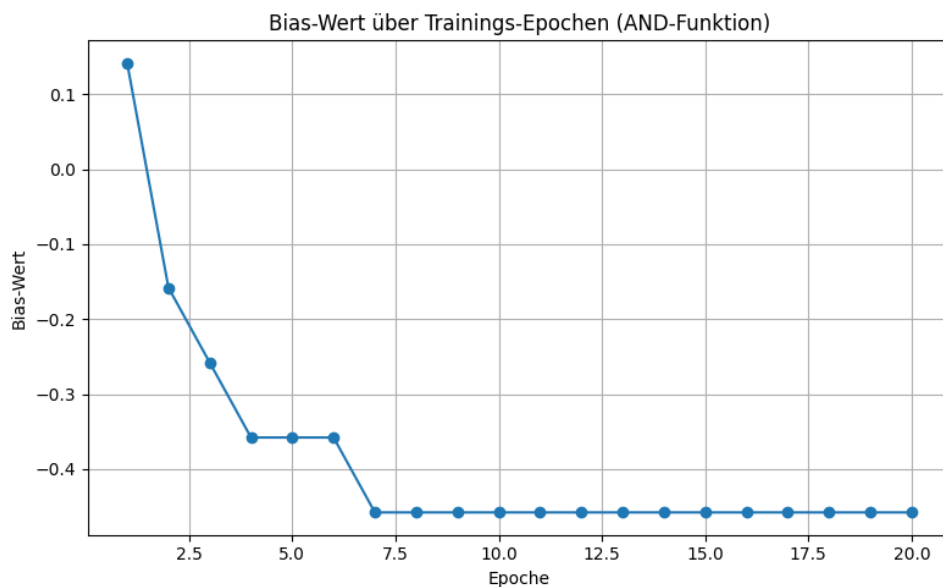
Interpretation:

- **Anfangszustand:** Die Gewichte starten mit zufälligen Werten.
- **Lernprozess:** In den ersten Epochen ändern sich die Gewichte deutlich, da das Perzeptron versucht, die Fehler zu korrigieren.
- **Konvergenz:** Nach einigen Epochen stabilisieren sich die Gewichte, da das Perzeptron gelernt hat, die UND-Funktion korrekt zu klassifizieren.

Warum passiert das?

- Bei der UND-Funktion ist nur der Fall $[1,1] \rightarrow 1$, alle anderen Kombinationen führen zu 0.
- Das Perzeptron passt seine Gewichte so an, dass nur bei beiden Eingaben = 1 die lineare Kombination den Schwellenwert überschreitet.

Hier siehst du, wie sich der **Bias-Wert** während des Trainings eines Perzeptrons zur **logischen UND-Funktion** verändert:



Interpretation:

- **Anfang:** Der Bias startet mit einem zufälligen Wert.
- **Veränderung:** In den ersten Epochen wird der Bias angepasst, um die Klassifikation zu verbessern.
- **Stabilisierung:** Sobald das Perzeptron gelernt hat, die UND-Funktion korrekt zu klassifizieren, bleibt der Bias konstant.

Der Bias hilft dabei, die Entscheidungsgrenze zu verschieben, sodass das Perzeptron auch dann korrekt klassifizieren kann, wenn die Eingaben allein nicht ausreichen.

6.2 Beschreiben der verschiedenen Überdeckungsmaße für neuronale Netzwerke

Frage:

Welche Überdeckungsmaße existieren für neuronale Netzwerke, und wie unterscheiden sich Neuronenüberdeckung, Schwellenwertüberdeckung und Vorzeichenwechselüberdeckung hinsichtlich ihrer Aussagekraft?

Antwort:

In der Analyse und Testabdeckung neuronaler Netzwerke – insbesondere im Kontext von **robustem maschinellern Lernen** und **Testgenerierung** – spielen sogenannte **Überdeckungsmaße (Coverage Metrics)** eine zentrale Rolle. Sie helfen dabei, zu beurteilen, wie gründlich ein neuronales Netzwerk durch Tests oder Trainingsdaten „erforscht“ wurde.

Hier sind drei wichtige Überdeckungsmaße und ihre Unterschiede:

1. Neuronenüberdeckung (Neuron Coverage)

- **Definition:** Misst, wie viele Neuronen in einem Netzwerk während der Eingabeaktivierung mindestens einmal über einem bestimmten Schwellenwert aktiviert wurden.
- **Formel:**
$$\text{Neuron Coverage} = \frac{\text{Anzahl aktivierter Neuronen}}{\text{Gesamtanzahl der Neuronen}}$$
- **Aussagekraft:**
 - Ähnlich wie Code Coverage in der Softwareentwicklung.
 - Gibt einen groben Überblick darüber, ob alle Teile des Netzwerks „benutzt“ wurden.
 - **Limitation:** Kann leicht hohe Werte erreichen, ohne dass das Netzwerk wirklich gründlich getestet wurde.

2. Schwellenwertüberdeckung (Threshold Coverage)

- **Definition:** Misst, wie oft Neuronenaktivierungen bestimmte Schwellenbereiche überschreiten oder unterschreiten.
- **Beispiel:** Ein Neuron wird als „hoch aktiviert“ gezählt, wenn seine Aktivierung über 0.75 liegt.
- **Aussagekraft:**
 - Feiner als reine Neuronenüberdeckung.
 - Zeigt, ob Neuronen in verschiedenen Intensitätsbereichen aktiviert wurden.
 - **Nutzen:** Hilfreich, um zu erkennen, ob ein Netzwerk auch extreme Aktivierungen erlebt hat.

3. Vorzeichenwechsel-Überdeckung (Sign Change Coverage / Sign-Sign Coverage)

- **Definition:** Misst, wie oft sich das Vorzeichen der Aktivierung eines Neurons ändert, wenn sich die Eingabe leicht verändert.
- **Ziel:** Erfasst die **Sensitivität** des Netzwerks gegenüber kleinen Änderungen.
- **Aussagekraft:**

- Besonders relevant für Robustheitstests (z. B. gegenüber Adversarial Examples).
- Zeigt, ob das Netzwerk stabil oder empfindlich auf kleine Eingabeänderungen reagiert.
- **Stärker interpretierbar** im Kontext von Sicherheit und Verlässlichkeit.

Vergleich der Aussagekraft

Maß	Fokus	Aussagekraft	Typischer Einsatzbereich
Neuronenüberdeckung	Aktivierung überhaupt	Grob, schnell zu berechnen	Erste Analyse, einfache Testabdeckung
Schwellenwert-überdeckung	Aktivierungsintensität	Mittel, differenzierter als Neuronen-abdeckung	Tiefergehende Testgenerierung
Vorzeichenwechsel	Reaktion auf Eingabeänderung	Hoch, robustheitsrelevant	Sicherheit, Adversarial Robustness

7.0 Kapitel: Testen KI-basierter Systeme im Überblick

– K2:

Konzept-Drift erkennen, Test-Notwendigkeit erläutern;

– K4:

Testansatz-Auswahl für Szenarien treffen.

7.1 Systemspezifikationen für KI-basierte Systeme

Frage:

Warum stellen unvollständige oder explorativ entwickelte Spezifikationen bei KI-basierten Systemen eine besondere Herausforderung für das Testen dar?

Antwort:

Unvollständige oder explorativ entwickelte Spezifikationen stellen beim Testen von **KI-basierten Systemen** eine besondere Herausforderung dar, weil sie die **klassischen Testmethoden** untergraben, die auf klaren Anforderungen und erwarteten Ergebnissen basieren.

Hier sind die wichtigsten Gründe im Detail:

1. Fehlende klare Soll-Vorgaben

- **Problem:** Ohne vollständige Spezifikation ist unklar, was das System *tun soll*.
- **Folge:** Es ist schwer zu entscheiden, ob ein Verhalten korrekt oder fehlerhaft ist.
- **Beispiel:** Ein KI-Modell zur Bilderkennung erkennt ein Tier – aber ist das Ergebnis „Hund“ korrekt, wenn keine Referenz vorhanden ist?

2. Erschwerte Testfallableitung

- **Problem:** Klassische Testfälle basieren auf Anforderungen. Fehlen diese, müssen Tests explorativ oder datengetrieben entwickelt werden.
- **Folge:** Höherer Aufwand, weniger systematische Abdeckung, Gefahr von Lücken.

3. Explorative Entwicklung ≠ deterministisches Verhalten

- **Problem:** KI-Systeme verhalten sich oft probabilistisch oder adaptiv.
- **Folge:** Selbst bei gleichen Eingaben kann das Verhalten variieren – was das Testen erschwert, wenn keine klaren Toleranzen oder Erwartungsbereiche definiert sind.

4. Schwierige Fehleridentifikation

- **Problem:** Ohne klare Spezifikation ist unklar, ob ein „ungewöhnliches“ Verhalten ein Fehler oder eine zulässige Variation ist.
- **Folge:** Tester müssen stärker auf Heuristiken, Metriken oder Vergleichsmodelle zurückgreifen.

5. Vertrauen und Nachvollziehbarkeit

- **Problem:** Ohne dokumentierte Anforderungen ist es schwer, Vertrauen in die Ergebnisse zu gewinnen.
- **Folge:** Stakeholder (z. B. Auditoren, Kunden) können die Qualität des Systems nicht objektiv bewerten.

Fazit: Warum das für das Testen kritisch ist

Herausforderung	Auswirkung auf das Testen
Keine klaren Anforderungen	Keine objektiven Testziele
Explorative Entwicklung	Hoher Testaufwand, viele Unbekannte
Adaptives Verhalten	Tests müssen dynamisch und tolerant sein
Fehlende Referenzwerte	Keine einfache Fehlererkennung möglich

7.2 Wie werden KI-basierte Systeme auf jeder Teststufe getestet?

Frage:

Welche zusätzlichen Teststufen sind bei KI-basierten Systemen erforderlich, und wie unterscheiden sich diese von klassischen Teststufen?

Antwort:

KI-basierte Systeme erfordern **zusätzliche Teststufen**, die über die klassischen Phasen wie **Unit-, Integrations-, System- und Abnahmetests** hinausgehen. Das liegt an ihrer **nicht-deterministischen Natur**, der **Datenabhängigkeit** und der **Komplexität des Lernverhaltens**.

Hier ist eine Übersicht über die **zusätzlichen Teststufen** und wie sie sich von klassischen Teststufen unterscheiden:

Zusätzliche Teststufen für KI-Systeme

1. Datenqualitäts- und Datenintegritätstests

- **Ziel:** Sicherstellen, dass Trainings-, Validierungs- und Testdaten korrekt, vollständig und repräsentativ sind.
- **Unterschied:** Klassische Systeme testen Code – KI-Systeme müssen auch die Daten als „Co-Programm“ testen.

2. Modellverhaltenstests

- **Ziel:** Überprüfung, ob das Modell unter verschiedenen Bedingungen erwartungsgemäß reagiert.
- **Beispiele:** Robustheit gegen Rauschen, Generalisierung, Verhalten bei Grenzfällen.
- **Unterschied:** Tests sind oft probabilistisch, nicht deterministisch.

3. Bias- und Fairness-Tests

- **Ziel:** Aufdecken von diskriminierendem Verhalten oder systematischen Verzerrungen.
- **Unterschied:** Klassische Systeme haben keine „Meinung“ – KI kann unbeabsichtigt diskriminieren.

4. Erklärbarkeitstests (Explainability Testing)

- **Ziel:** Prüfen, ob Entscheidungen nachvollziehbar und erklärbar sind.
- **Unterschied:** Klassische Systeme sind transparent – KI-Modelle oft „Black Boxes“.

5. Robustheits- und Adversarial-Tests

- **Ziel:** Testen der Widerstandsfähigkeit gegen gezielte oder zufällige Störungen.
- **Unterschied:** Klassische Systeme sind meist deterministisch – KI kann durch kleine Änderungen stark beeinflusst werden.

6. Kontinuierliche Validierung (Continuous Validation)

- **Ziel:** Sicherstellen, dass das Modell auch nach Updates oder neuen Daten weiterhin korrekt funktioniert.
- **Unterschied:** KI-Systeme verändern sich durch Retraining – klassische Systeme bleiben nach Deployment meist stabil.

7. Ethik- und Compliance-Tests

- **Ziel:** Überprüfung auf Einhaltung gesetzlicher und ethischer Standards (z. B. DSGVO, Transparenzpflichten).
- **Unterschied:** Klassische Systeme unterliegen meist klaren Regeln – KI kann unvorhersehbare Auswirkungen haben.

Vergleich: Klassische vs. KI-Teststufen

Teststufe	Klassisch	KI-basiert
Unit-Test	Logik einzelner Funktionen	Vorverarbeitung, Modellkomponenten
Integrationstest	Zusammenspiel von Modulen	Datenpipeline + Modellintegration
Systemtest	Gesamtsystemverhalten	Modellverhalten + Datenabhängigkeit
Abnahmetest	Anforderungen erfüllt?	Zielverhalten + ethische Aspekte
+ Datenqualitätstest	–	Daten als kritischer Bestandteil
+ Bias/Fairness-Test	–	Gesellschaftlich relevant
+ Robustheitstest	–	Sicherheit gegen Manipulation
+ Erklärbarkeitstest	–	Vertrauen und Nachvollziehbarkeit

7.3 Testdaten die das testen der KI-basierten Systeme erschweren können

Frage:

Welche der folgenden Faktoren erschweren den Umgang mit Testdaten bei KI-basierten Systemen?

- a. Große Datenmengen
- b. Statische Benutzeroberfläche
- c. Datenschutzanforderungen
- d. Sich verändernde Eingabedaten

Antwort:

Die korrekten Faktoren, die den Umgang mit Testdaten bei KI-basierten Systemen **erschweren**, sind:

✓ **a. Große Datenmengen**

- KI-Systeme benötigen oft sehr viele Daten zum Trainieren und Testen. Das Speichern, Verarbeiten und Auswerten dieser Daten ist ressourcenintensiv und kann die Testdurchführung verlangsamen oder verkomplizieren.

✓ **c. Datenschutzanforderungen**

- Testdaten enthalten häufig personenbezogene Informationen. Die Einhaltung von Datenschutzgesetzen (z. B. DSGVO) erfordert Anonymisierung, Einwilligungen und sichere Datenverarbeitung – was den Testprozess deutlich aufwendiger macht.

✓ **d. Sich verändernde Eingabedaten**

- In dynamischen Umgebungen (z. B. Echtzeitdaten, Sensoren, Nutzerverhalten) ändern sich die Eingaben ständig. Das erschwert die Reproduzierbarkeit von Tests und erfordert kontinuierliche Anpassung der Testdaten.

✗ **b. Statische Benutzeroberfläche**

- Eine statische UI ist **kein direkter Hinderungsgrund** für den Umgang mit Testdaten. Sie kann die Interaktion mit dem System beeinflussen, aber nicht die Datenverarbeitung oder -analyse selbst.

7.4 Wie sich Automatisierungsverzerrung auf das Testen auswirkt

Was versteht man unter Automatisierungsverzerrung und wie sich diese auf das Testen auswirkt?

Antwort:

Automatisierungsverzerrung (Automation Bias) bezeichnet die Tendenz von Menschen, automatisierten Systemen (z. B. KI, Algorithmen, Entscheidungsunterstützungssystemen) **zu viel Vertrauen zu schenken** – selbst dann, wenn diese Fehler machen oder unvollständig sind.

Was genau passiert bei Automatisierungsverzerrung?

Menschen neigen dazu:

- **automatisierte Vorschläge unkritisch zu übernehmen**, auch wenn sie falsch sind.
- **eigene Entscheidungen zu unterdrücken**, wenn sie im Widerspruch zur Maschine stehen.
- **Fehler der Automatisierung zu übersehen** oder nicht zu hinterfragen.

Auswirkungen auf das Testen neuronaler Netzwerke

Automatisierungsverzerrung kann das Testen und die Qualitätssicherung von KI-Systemen auf verschiedene Weise beeinträchtigen:

Bereich	Auswirkung
Testdesign	Tester verlassen sich zu stark auf automatische Testtools und übersehen manuelle Tests oder explorative Ansätze.
Fehlererkennung	Falsche Entscheidungen des Netzwerks werden nicht erkannt, weil sie „von der KI kommen“.
Validierung	Tester akzeptieren Ergebnisse, ohne sie kritisch zu hinterfragen oder mit alternativen Methoden zu vergleichen.
Bias-Verstärkung	Wenn das Modell bereits Verzerrungen enthält, werden diese durch unkritische Akzeptanz weitergetragen.
Sicherheitskritische Systeme	In Bereichen wie Medizin, Luftfahrt oder autonomem Fahren kann Automatisierungsverzerrung zu gefährlichen Fehlentscheidungen führen.

Wie kann man Automatisierungsverzerrung vermeiden?

- **Bewusstsein schaffen:** Schulungen für Entwickler und Tester über kognitive Verzerrungen.
- **Transparenz fördern:** Erklärbare KI (Explainable AI) einsetzen.
- **Redundanz einbauen:** Menschliche Überprüfung oder alternative Systeme zur Validierung.
- **Grenzen der Automatisierung kennen:** Automatisierte Tests als Ergänzung, nicht als Ersatz für menschliches Urteilsvermögen.

7.5 Beschreiben von Dokumentation einer KI-Komponente und verstehen, wie die Dokumentation das Testen eines KI-basierten System unterstützt.

Frage:

Welche Informationen in der Dokumentation einer KI-Komponente sind besonders wichtig für das Testen und warum?

Antwort:

Für das **Testen einer KI-Komponente** ist eine **gut strukturierte und vollständige Dokumentation** entscheidend. Sie liefert nicht nur technische Details, sondern auch Kontext, der für die Planung, Durchführung und Bewertung von Tests unerlässlich ist. Hier sind die **wichtigsten Informationen**, die in der Dokumentation enthalten sein sollten – und **warum** sie für das Testen relevant sind:

1. Modellarchitektur

- **Was:** Typ des Modells (z. B. CNN, Transformer), Anzahl der Schichten, Aktivierungsfunktionen, Parameteranzahl.
- **Warum wichtig:** Ermöglicht gezielte Tests auf bestimmte Komponenten (z. B. Schichten, Neuronen) und hilft bei der Auswahl geeigneter Testmetriken (z. B. Neuronenüberdeckung).

2. Trainingsdaten und -prozesse

- **Was:** Herkunft, Umfang, Vorverarbeitung, Labelqualität, Datenverteilung.
- **Warum wichtig:** Tester können prüfen, ob das Modell auf bestimmten Datenbereichen unterrepräsentiert ist (Bias, Fairness) oder ob Overfitting vorliegt.

3. Test- und Validierungsmetriken

- **Was:** Verwendete Metriken (z. B. Accuracy, F1-Score, ROC-AUC), Validierungsstrategien (z. B. Cross-Validation).
- **Warum wichtig:** Dient als Referenz für eigene Testergebnisse und zur Bewertung der Modellqualität.

4. Bekannte Einschränkungen und Risiken

- **Was:** Schwächen des Modells, bekannte Fehlerfälle, Unsicherheiten.
- **Warum wichtig:** Tester können gezielt auf diese Schwachstellen testen (z. B. adversarial robustness, edge cases).

5. Sicherheits- und Robustheitsaspekte

- **Was:** Maßnahmen gegen adversariale Angriffe, Interpretierbarkeit, Schutz vor Datenlecks.
- **Warum wichtig:** Kritisch für Tests in sicherheitsrelevanten Anwendungen (z. B. Medizin, autonomes Fahren).

6. Einsatzkontext und Zielgruppen

- **Was:** Wofür ist das Modell gedacht? Wer nutzt es? In welchem Umfeld?
- **Warum wichtig:** Tests können auf realistische Nutzungsszenarien abgestimmt werden (z. B. Sprache, Kultur, Gerätetyp).

7. Schnittstellen und APIs

- **Was:** Eingabe-/Ausgabeformate, unterstützte Datenformate, Fehlerbehandlung.
- **Warum wichtig:** Ermöglicht automatisierte Tests und Integrationstests mit anderen Systemkomponenten.

8. Versionierung und Änderungsverlauf

- **Was:** Modellversionen, Änderungen an Architektur oder Daten.
- **Warum wichtig:** Tests müssen ggf. angepasst oder wiederholt werden, wenn sich das Modell ändert.

7.6 Notwendigkeit, das trainierte Modell häufig zu testen um einen Konzeptdrift zu vermeiden

Frage:

Was ist ein Konzeptdrift, und wie kann regelmäßiges Testen helfen, deren Auswirkungen zu erkennen und zu minimieren?

Antwort:

Konzeptdrift (Concept Drift) bezeichnet eine Veränderung in den zugrunde liegenden Datenmustern, auf denen ein KI-Modell basiert. Das bedeutet: Das, was das Modell ursprünglich gelernt hat, **gilt irgendwann nicht mehr** – weil sich die Realität verändert hat.

Was ist Konzeptdrift?

Einfach gesagt:

„Das Modell trifft Entscheidungen auf Basis von gestern – aber heute gelten andere Regeln.“

Typische Ursachen:

- **Verändertes Nutzerverhalten** (z. B. neue Trends, Saisonalität)
- **Technologische Veränderungen** (z. B. neue Geräte, Formate)
- **Externe Einflüsse** (z. B. Pandemien, Gesetzesänderungen)
- **Datenquellen ändern sich** (z. B. neue Sensoren, andere Formate)

Wie hilft regelmäßiges Testen?

Regelmäßiges Testen kann helfen, **Konzeptdrift frühzeitig zu erkennen und zu minimieren**, durch:

1. Monitoring von Modellmetriken

- Vergleich von aktuellen Vorhersageleistungen mit früheren.
- Plötzliche Einbrüche in Accuracy, Precision, etc. können auf Drift hinweisen.

2. Vergleich von Datenverteilungen

- Statistische Tests (z. B. KL-Divergenz, Chi²-Test) zeigen, ob sich Eingabedaten verändert haben.
- Auch Label-Verteilungen können driften.

3. Drift-spezifische Tests

- Spezielle Tests prüfen, ob sich die Beziehung zwischen Eingabe und Zielwert verändert hat.
- Beispiel: Ein Modell für Kreditwürdigkeit muss neu bewertet werden, wenn sich wirtschaftliche Rahmenbedingungen ändern.

4. Retraining-Trigger

- Tests können als Auslöser für automatisiertes Retraining dienen.
 - So bleibt das Modell aktuell und relevant.
-

Arten von Konzeptdrift

Art des Drifts	Beschreibung	Beispiel
Sudden Drift	Plötzliche Änderung	Neue Gesetzgebung verändert Nutzerverhalten
Gradual Drift	Langsame Verschiebung	Kundenpräferenzen ändern sich über Monate
Recurring Drift	Wiederkehrende Muster	Saisonalität im Kaufverhalten
Incremental Drift	Schrittweise Änderung	Technologische Entwicklung

7.7 Analysieren eines Szenarios, bestimmen der Test Vorgehensweise, die bei der Entwicklung des ML-Systems befolgt werden muss.

Frage:

Ein Unternehmen entwickelt ein ML -System zur Kreditwürdigkeitsprüfung. Welchen Risiken sollten bei der Auswahl der Testvorgehensweise berücksichtigt werden, und welche Testmethoden wären geeignet?

Antwort:

Beim Testen eines **ML-Systems zur Kreditwürdigkeitsprüfung** müssen besonders hohe Anforderungen an **Fairness, Transparenz, Sicherheit und Datenschutz** erfüllt werden. Hier sind die wichtigsten **Risiken** und dazu passende **Testmethoden**, die bei der Auswahl der Testvorgehensweise berücksichtigt werden sollten:

Risiken bei der Testvorgehensweise

1. Diskriminierung und Bias

- **Risiko:** Das Modell könnte bestimmte Gruppen (z. B. nach Alter, Geschlecht, Herkunft) systematisch benachteiligen.
- **Folge:** Verstöße gegen Antidiskriminierungsgesetze, Reputationsschäden.
- **Testansatz:**
 - Fairness-Tests (z. B. disparate impact, equal opportunity)
 - Subgruppenanalysen

2. Intransparente Entscheidungen

- **Risiko:** Entscheidungen sind nicht nachvollziehbar („Black Box“).
- **Folge:** Kunden können Entscheidungen nicht anfechten, rechtliche Probleme.
- **Testansatz:**
 - Erklärbarkeitstests (z. B. LIME, SHAP)
 - Vergleich mit regelbasierten Baselines

3. Konzeptdrift

- **Risiko:** Das Modell wird mit der Zeit ungenau, weil sich wirtschaftliche Bedingungen oder Kundendaten ändern.
- **Folge:** Fehlentscheidungen, sinkende Modellleistung.
- **Testansatz:**
 - Drift-Erkennung (z. B. statistische Tests auf Datenverteilungen)
 - Kontinuierliche Validierung

4. Fehlklassifikationen mit hoher Auswirkung

- **Risiko:** Falsch negative oder falsch positive Entscheidungen (z. B. Kredit verweigert trotz guter Bonität).
- **Folge:** Finanzielle Schäden, rechtliche Konsequenzen.
- **Testansatz:**
 - Sensitivitätsanalysen
 - Edge-Case-Tests
 - Konfidenzbasierte Bewertung

5. Datenschutzverletzungen

- **Risiko:** Verwendung oder Leckage sensibler personenbezogener Daten.
- **Folge:** DSGVO-Verstöße, Bußgelder.
- **Testansatz:**
 - Datenschutz-Tests (z. B. auf Datenminimierung, Anonymisierung)
 - Zugriffskontrollen und Logging

Geeignete Testmethoden

Testmethode	Ziel
Black-Box-Tests	Verhalten des Modells ohne Kenntnis der internen Struktur prüfen
White-Box-Tests	Interne Logik und Entscheidungswege analysieren
Fairness-Tests	Diskriminierungsfreiheit sicherstellen
Adversarial Testing	Robustheit gegen gezielte Manipulation prüfen
Explainability-Tests	Nachvollziehbarkeit der Entscheidungen bewerten
Regressionstests	Sicherstellen, dass Änderungen keine unerwünschten Effekte haben,
Drift Detection & Monitoring	Veränderungen in Daten oder Modellleistung erkennen
Compliance-Tests	Einhaltung gesetzlicher Vorgaben (z. B. DSGVO, Kreditwesengesetz)

8.0 Kapitel: Testen KI-spezifischer Qualitätsmerkmale

Fokus auf Bias-Testing, Nicht-Determinismus, Transparenz/Erklärbarkeit, Side-Effects, Safety –
Aufbauend auf Kap. 2+7.

8.1 Herausforderungen beim Testen die durch das Selbstlernen KI-basierter Systeme entstehen

Frage:

Welche Herausforderungen ergeben sich beim Testen selbstlernender Systeme, insbesondere im Hinblick auf unerwartete Verhaltensänderungen und Testorakel?

Antwort:

Das Testen selbstlernender Systeme – insbesondere solcher, die auf maschinellem Lernen (ML) oder künstlicher Intelligenz (KI) basieren – stellt eine erhebliche Herausforderung dar. Zwei zentrale Problembereiche sind dabei:

1. Unerwartete Verhaltensänderungen

Selbstlernende Systeme passen ihr Verhalten auf Basis neuer Daten an. Das führt zu folgenden Herausforderungen:

- **Nicht-deterministisches Verhalten:** Das System kann bei gleichem Input unterschiedliche Outputs liefern, je nachdem, wie es sich weiterentwickelt hat.
- **Verhaltensdrift (Concept Drift):** Die zugrunde liegenden Datenverteilungen ändern sich über die Zeit, was zu einer schleichenden Veränderung des Systemverhaltens führt.
- **Exploration vs. Exploitation:** Lernende Systeme (z. B. in Reinforcement Learning) probieren neue Strategien aus, was zu unvorhersehbarem Verhalten führen kann.
- **Verdeckte Fehler:** Ein Modell kann scheinbar korrekt funktionieren, aber in bestimmten Randfällen oder unter neuen Bedingungen versagen.

2. Testorakel-Problem

Ein **Testorakel** ist ein Mechanismus, der entscheidet, ob das Verhalten eines Systems korrekt ist. Bei ML-Systemen ist das besonders schwierig:

- **Keine klare Spezifikation:** Oft gibt es keine formale Definition, was „korrektes Verhalten“ ist – insbesondere bei komplexen Aufgaben wie Bilderkennung oder Sprachverarbeitung.
 - **Subjektivität:** Die Bewertung hängt oft von menschlichem Urteil ab (z. B. bei Chatbots oder Empfehlungssystemen).
 - **Unvollständige Ground Truth:** Die verfügbaren Testdaten decken nicht alle möglichen Szenarien ab.
 - **Verallgemeinerung:** Ein Modell kann auf Testdaten gut abschneiden, aber in der realen Welt versagen – das Testorakel erkennt das nicht.
-

Mögliche Lösungsansätze

- **Metamorphes Testen:** Statt zu prüfen, ob ein Output „richtig“ ist, prüft man, ob sich der Output bei bestimmten Input-Transformationen erwartungsgemäß verändert.
- **Robustheitstests:** Systematisches Testen mit verrauschten, manipulierten oder adversarialen Inputs.
- **Monitoring im Betrieb:** Kontinuierliche Überwachung des Verhaltens nach dem Deployment.
- **Erklärbarkeit (Explainable AI):** Hilft, das Verhalten besser zu verstehen und zu bewerten.
- **Simulation und Szenarien-Tests:** Besonders bei autonomen Systemen (z. B. Fahrzeuge) wichtig.

8.2 Wie werden autonome KI-basierende-Systeme getestet

Frage:

Wie kann getestet werden, ob ein autonomes System in kritischen Situationen korrekt entscheidet, ob es menschliches Eingreifen benötigt?

Antwort:

Das Testen, ob ein **autonomes System in kritischen Situationen korrekt entscheidet, ob menschliches Eingreifen notwendig ist**, ist besonders anspruchsvoll, da es um **Sicherheitskritik, Kontextverständnis** und **dynamische Entscheidungsfindung** geht. Hier sind zentrale Aspekte und Methoden, wie man das testen kann:

1. Definition kritischer Situationen

Zunächst muss klar definiert werden, **was eine kritische Situation ist**, z. B.:

- Sensorfehler oder widersprüchliche Daten
- Unbekannte Objekte oder Umgebungen
- Unsichere Vorhersagen (z. B. niedrige Konfidenz)
- Gesetzlich oder ethisch ambivalente Entscheidungen

Diese Situationen müssen **systematisch identifiziert und simuliert** werden.

2. Testmethoden

a) Szenarienbasiertes Testen

- Verwendung von **Simulationsumgebungen** (z. B. CARLA, LGSVL, AirSim), um kontrollierte kritische Situationen zu erzeugen.
- Beispiel: Ein Fußgänger tritt plötzlich auf die Straße – entscheidet das System, ob es selbst handeln kann oder den Fahrer alarmieren muss?

b) Unsicherheitsbasierte Tests

- Testen, ob das System bei **hoher Unsicherheit** (z. B. durch verrauschte Sensoren oder widersprüchliche Inputs) korrekt auf „Handover“ an den Menschen schaltet.
- Einsatz von **Bayesian Networks** oder **Monte-Carlo Dropout**, um Unsicherheiten zu quantifizieren.

c) Metamorphes Testen

- Wenn das System bei leicht veränderten, aber semantisch ähnlichen Inputs **unterschiedlich reagiert**, kann das auf inkonsistentes Entscheidungsverhalten hinweisen.

d) Human-in-the-Loop-Tests

- Integrierte Tests mit echten Fahrern oder Operatoren, um zu prüfen, ob das System in kritischen Momenten **verständlich und rechtzeitig** zur Übergabe auffordert.

3. Bewertungskriterien

- **Reaktionszeit:** Wie schnell erkennt das System die Notwendigkeit menschlichen Eingreifens?
- **Konfidenzschwelle:** Wird die Schwelle zur Übergabe korrekt gesetzt?
- **Kommunikation:** Wird der Mensch klar und verständlich informiert?
- **Fehlalarme vs. verpasste Übergaben:** Balance zwischen unnötigen Eingriffen und gefährlichen Unterlassungen.

4. KI-spezifische Herausforderungen

- **Black-Box-Charakter:** Warum das System eine Übergabe fordert, ist oft nicht transparent.
- **Vertrauenswürdigkeit:** Zu häufige oder zu seltene Übergaben können das Vertrauen in das System untergraben.
- **Domänenverschiebung:** Das System kann in realen Situationen auf Daten stoßen, die im Training nicht vorkamen.

5. Ergänzende Maßnahmen

- **Explainable (XAI):** Erklärungen für Übergabeentscheidungen helfen beim Testen und bei der Akzeptanz.
- **Fail-Safe-Designs:** Wenn das System unsicher ist, sollte es **immer** eine sichere Entscheidung treffen (z. B. anhalten oder Übergabe fordern).
- **Regulatorische Tests:** In vielen Ländern (z. B. durch UNECE oder ISO 21448 – SOTIF) sind Tests für solche Systeme vorgeschrieben.

8.3 KI-basiertes System auf Verzerrungen prüfen

Frage:

Welche Methoden können eingesetzt werden, um algorithmische, stichprobenartige und angemessene Verzerrungen in einem KI-System zu erkennen?

Antwort:

Das Erkennen von **algorithmischen, stichprobenartigen und angemessenen Verzerrungen** (Bias) in KI-Systemen ist entscheidend für Fairness, Transparenz und Verlässlichkeit. Hier ist ein Überblick über die drei Bias-Typen und geeignete **Methoden zu ihrer Erkennung**:

1. Algorithmische Verzerrung (Model Bias)

Diese entsteht durch die Art und Weise, wie das Modell trainiert oder strukturiert ist – unabhängig von den Daten.

Erkennungsmethoden:

- **Fairness-Metriken:**
 - *Demographic Parity*: Prüft, ob alle Gruppen gleichbehandelt werden.
 - *Equal Opportunity*: Prüft, ob alle Gruppen gleiche Chancen auf ein positives Ergebnis haben.
 - *Disparate Impact Ratio*: Verhältnis der positiven Vorhersagen zwischen Gruppen.
 - **Feature Importance Analyse:**
 - Zeigt, ob sensible Merkmale (z. B. Geschlecht, Ethnie) übermäßig Einfluss haben.
 - **Counterfactual Fairness Testing:**
 - Prüft, ob das Ergebnis gleich bleibt, wenn nur ein sensibles Attribut geändert wird.
-

2. Stichprobenverzerrung (Sampling Bias)

Diese entsteht, wenn die Trainingsdaten nicht repräsentativ für die reale Welt sind.

Erkennungsmethoden:

- **Datenvisualisierung:**
 - Histogramme, Heatmaps oder PCA zur Analyse der Verteilung von Klassen und Attributen.
 - **Vergleich mit Populationsdaten:**
 - Abgleich der Datenverteilung mit bekannten demografischen oder statistischen Referenzwerten.
 - **Subgruppenanalyse:**
 - Performance-Messung (Accuracy, F1, AUC) getrennt nach Gruppen (z. B. Altersklassen, Regionen).
 - **Coverage-Analyse:**
 - Prüft, ob alle relevanten Szenarien oder Gruppen ausreichend vertreten sind.
-

3. Angemessene Verzerrung (Justifiable Bias)

Diese ist **nicht notwendigerweise problematisch**, sondern kann durch reale Unterschiede gerechtfertigt sein (z. B. medizinische Unterschiede zwischen Altersgruppen).

Erkennungsmethoden:

- **Domänenwissen einbeziehen:**
 - Experten prüfen, ob Unterschiede in den Ergebnissen durch reale Unterschiede erklärbar sind.
 - **Causal Inference / Kausalanalyse:**
 - Unterscheidung zwischen Korrelation und Kausalität.
 - **Sensitivity Analysis:**
 - Wie stark beeinflussen bestimmte Merkmale das Ergebnis? Ist das gerechtfertigt?
-

Tools & Frameworks

- **IBM Fairness 360 (AIF360)**
- **Google What-If Tool**
- **Fairlearn (Microsoft)**
- **SHAP / LIME** für erklärbare Modelle
- **Datasheets for Datasets** (Gebru et al.) zur strukturierten Dokumentation von Datensätzen

8.4 Herausforderungen bei probabilistische und nichtdeterministische KI-basierter Systeme

Frage:

Warum ist das Testen probabilistischer oder nichtdeterministischer Systeme besonders herausfordernd, und wie kann man mit Mehrfachausführungen umgehen?

Antwort:

Das Testen **probabilistischer** oder **nichtdeterministischer Systeme** (z. B. viele KI-Modelle, simulationsbasierte Systeme, verteilte Systeme) ist besonders herausfordernd, weil das Verhalten bei gleicher Eingabe **nicht immer gleich** ist. Hier sind die Hauptgründe und wie man damit umgehen kann:

Warum ist das Testen so schwierig?

1. Nicht-reproduzierbarkeit

- Ein und derselbe Testlauf kann unterschiedliche Ergebnisse liefern.
- Fehler treten möglicherweise **nur sporadisch** auf („Heisenbugs“).

2. Keine klare Erwartung (Testorakel-Problem)

- Es ist oft unklar, was das „richtige“ Ergebnis ist – besonders bei stochastischen Modellen.

3. Statistische Schwankungen

- Ergebnisse können durch Zufall beeinflusst sein, was die Interpretation erschwert.

4. Komplexe Zustandsräume

- Die Zahl möglicher Zustände oder Ausgänge ist oft riesig oder unendlich.

Wie kann man mit Mehrfachausführungen umgehen?

1. Wiederholte Ausführung (Multiple Runs)

- Führe denselben Test **mehrfach** aus (z. B. 30–100 Mal).
- Sammle statistische Kennzahlen: **Mittelwert, Varianz, Konfidenzintervalle**.
- Beispiel: „In 95 % der Fälle liefert das System eine akzeptable Antwort.“

2. Statistische Hypothesentests

- Verwende Tests wie **t-Test**, **Chi²-Test** oder **ANOVA**, um Unterschiede zwischen Modellen oder Konfigurationen zu bewerten.
- Hilfreich, um zu prüfen, ob beobachtete Unterschiede **signifikant** sind oder nur zufällig.

3. Robustheitsmetriken

- **Worst-case-Analyse:** Wie schlecht kann das System im ungünstigsten Fall abschneiden?
- **Stabilitätsmetriken:** Wie stark schwankt das Verhalten bei kleinen Änderungen?

4. Seed-Kontrolle

- Verwende **kontrollierte Zufallszahlen** (z. B. durch Setzen eines Seeds), um Tests reproduzierbarer zu machen – zumindest in der Entwicklung.

5. Vertrauensgrenzen und Toleranzbereiche

- Definiere **akzeptable Schwankungsbereiche** für Outputs.
- Beispiel: „Die Vorhersage darf um $\pm 5\%$ vom erwarteten Mittelwert abweichen.“

Tools und Frameworks

- **Hypothesis (Python):** Property-based Testing mit zufälligen Inputs.
- **Monte Carlo Testing:** Systematische Zufallstests mit statistischer Auswertung.
- **Statistical Model Checking:** Kombination aus Simulation und formaler Verifikation.

8.5 Herausforderungen beim Testen, die durch Komplexität KI-basierter Systeme entstehen?

Frage:

Welche Probleme entstehen beim Testen komplexer KI-Systeme, insbesondere wenn mehrere interagierende Modelle beteiligt sind?

Antwort:

Beim Testen **komplexer KI-Systeme mit mehreren interagierenden Modellen** (z. B. in autonomen Fahrzeugen, intelligenten Agentensystemen oder multimodalen KI-Anwendungen) entstehen eine Reihe spezifischer Herausforderungen. Diese betreffen sowohl die **technische Komplexität** als auch die **Teststrategie**. Hier sind die wichtigsten Probleme:

1. Intermodulare Abhängigkeiten

Problem:

- Modelle beeinflussen sich gegenseitig, z. B. ein Objekterkennungsmodell liefert Input für ein Entscheidungsmodell.
- Fehler in einem Modell können sich **kaskadierend** auf andere auswirken.

Folge:

- **Fehlersuche wird erschwert**, da unklar ist, wo der Fehler ursprünglich entstanden ist.
- **Testisolierung** ist schwierig: Ein Modell kann isoliert korrekt funktionieren, aber im Zusammenspiel versagen.

2. Nichtdeterminismus und emergentes Verhalten

Problem:

- Das Gesamtsystem zeigt Verhalten, das **nicht direkt aus den Einzelkomponenten ableitbar** ist.
- Besonders bei Reinforcement Learning oder Multi-Agenten-Systemen.

Folge:

- **Vorhersagbarkeit und Reproduzierbarkeit** sind eingeschränkt.
- Klassische Testmethoden (z. B. Unit Tests) greifen zu kurz.

3. Testorakel-Problem auf Systemebene

Problem:

- Es ist oft **unklar, was das „richtige“ Verhalten** des Gesamtsystems ist.
- Besonders bei offenen Aufgaben (z. B. Sprachverarbeitung, Navigation, Empfehlungssysteme).

Folge:

- Automatisierte Tests sind schwer zu formulieren.
- Es braucht **heuristische oder statistische Bewertungsmethoden**.

4. Heterogene Modelle und Datenformate

Problem:

- Unterschiedliche Modelle nutzen verschiedene Architekturen, Frameworks oder Datenrepräsentationen (z. B. NLP + Vision + Zeitreihen).
- Unterschiedliche Anforderungen an Latenz, Genauigkeit, Robustheit.

Folge:

- **Inkompatibilitäten** und **Integrationsfehler** sind häufig.
- Testumgebungen müssen **multimodal und flexibel** sein.

5. Verdeckte Fehler und Bias-Propagation

Problem:

- Ein Modell kann **systematisch verzerrte Outputs** liefern, die von nachfolgenden Modellen **nicht erkannt oder sogar verstärkt** werden.

Folge:

- Bias oder Fehler werden **nicht lokalisiert**, sondern im System „weitergetragen“.
- Besonders kritisch bei sicherheitsrelevanten oder ethisch sensiblen Anwendungen.

Strategien zur Bewältigung

- **Modularisiertes Testen mit Mocking/Stubbing:** Einzelne Modelle durch kontrollierte Platzhalter ersetzen.
- **End-to-End-Tests mit Szenarien:** Realistische, komplexe Abläufe simulieren.
- **Traceability & Logging:** Detaillierte Protokollierung der Modellinteraktionen.
- **Sensitivity & Ablation Tests:** Systematisch einzelne Modelle deaktivieren oder variieren.
- **Explainability-Tools:** SHAP, LIME oder Attention-Visualisierungen zur Analyse von Modellentscheidungen.
- **Simulation & Digital Twins:** Virtuelle Testumgebungen für komplexe Interaktionen.

8.6 Wie können Transparenz, Interpretierbarkeit und Erklärbarkeit KI-basierter Systeme getestet werden

Frage:

Wie können Transparenz, Interpretierbarkeit und Erklärbarkeit KI-basierter Systeme getestet werden, und welche Rolle spielt z.B. die LIME-Methode dabei?

Antwort:

Das Testen von **Transparenz**, **Interpretierbarkeit** und **Erklärbarkeit** (engl. *transparency, interpretability, explainability*) in KI-Systemen ist essenziell, um Vertrauen, Nachvollziehbarkeit und regulatorische Konformität sicherzustellen. Diese Eigenschaften sind besonders wichtig in sicherheitskritischen oder ethisch sensiblen Bereichen wie Medizin, Justiz oder autonomem Fahren.

Was bedeuten die Begriffe?

- **Transparenz:** Wie offen ist das System hinsichtlich seiner Struktur, Daten und Entscheidungslogik?
- **Interpretierbarkeit:** Wie leicht kann ein Mensch das Verhalten des Modells verstehen?
- **Erklärbarkeit:** Wie gut kann das Modell oder ein Hilfssystem *erklären*, warum eine bestimmte Entscheidung getroffen wurde?

Wie kann man diese Eigenschaften testen?

1. Modellbasierte Tests

- **White-Box-Analyse:** Bei interpretierten Modellen (z. B. Entscheidungsbäume, lineare Modelle) kann direkt geprüft werden, ob die Entscheidungslogik nachvollziehbar ist.
- **Komplexitätsmetriken:** Anzahl der Parameter, Tiefe von Entscheidungsbäumen, Anzahl der Regeln etc.

2. Erklärbarkeitstests mit XAI-Methoden

- **LIME (Local Interpretable Model-agnostic Explanations):**
 - Erzeugt lokale Erklärungen für einzelne Vorhersagen durch Approximation mit einem einfachen Modell.
 - **Teststrategie:** Prüfen, ob die erzeugten Erklärungen konsistent, stabil und für Menschen verständlich sind.
 - Beispiel: Ändert sich die Erklärung stark bei minimal veränderten Inputs? → Hinweis auf mangelnde Robustheit.
- **SHAP (SHapley Additive exPlanations):**
 - Liefert globale und lokale Erklärungen basierend auf Spieltheorie.
 - **Teststrategie:** Vergleich der SHAP-Werte mit Domänenwissen – stimmen sie mit menschlicher Intuition überein?

3. Benutzerzentrierte Tests

- **Usability-Tests:** Menschen bewerten, wie verständlich die Erklärungen sind.
- **Fragebögen:** Z. B. System Causability Scale (SCS) zur Bewertung der Erklärqualität.
- **A/B-Tests:** Vergleich von Systemvarianten mit und ohne Erklärungen.

4. Robustheit und Konsistenz

- **Stabilitätstests:** Kleine Änderungen im Input sollten nicht zu völlig anderen Erklärungen führen.
- **Kontrastive Erklärungen:** Warum wurde *dieses* Ergebnis gewählt und nicht ein anderes?

Rolle von LIME im Detail

LIME ist besonders nützlich, weil es:

- **modellagnostisch** ist (funktioniert mit beliebigen Modellen),
- **lokale Erklärungen** liefert (für einzelne Vorhersagen),
- **visuell interpretierbar** ist (z. B. Balkendiagramme mit Feature-Wichtigkeit).

Beispiel-Test mit LIME:

1. Wähle eine Reihe von Testinstanzen.
2. Generiere LIME-Erklärungen.
3. Prüfe:
 - **Kohärenz:** Sind die wichtigsten Features plausibel?
 - **Stabilität:** Ändern sich die Erklärungen bei ähnlichen Inputs stark?
 - **Domänenvalidität:** Stimmen die Erklärungen mit Expertenwissen überein?

Fazit

Das Testen von Erklärbarkeit ist **interdisziplinär**: Es braucht technische, statistische und menschenzentrierte Methoden. Tools wie **LIME**, **SHAP**, **Anchors** oder **Counterfactual Explanations** sind zentrale Bausteine, aber sie müssen **systematisch evaluiert** werden – nicht nur technisch, sondern auch hinsichtlich ihrer Verständlichkeit für Nutzer.

HO 8.6 Werkzeug verwenden, um zu zeigen, wie die Erklärbarkeit genutzt werden kann

Aufgabe:

Nutzen Sie das Tool LIME, um zu analysieren, welche Eingabemerkmale die Entscheidung eines ML-Modells beeinflusst haben

Lösung:

8.7 Herausforderungen bei der Erstellung von Testorakeln, die sich aus den spezifischen Merkmalen KI-basierter Systeme ergeben

Frage:

Was ist das Testorakelproblem bei KI-Systemen, und welche Methoden können helfen, dieses zu umgehen?

Antwort:

Das **Testorakelproblem** ist eines der zentralen Probleme beim Testen von KI-Systemen – insbesondere bei solchen, die auf maschinellem Lernen basieren. Es beschreibt die Schwierigkeit, **automatisch zu entscheiden, ob das Verhalten eines Systems korrekt ist**, wenn keine klare oder vollständige Spezifikation des erwarteten Outputs existiert.

Was ist das Testorakelproblem?

Ein **Testorakel** ist normalerweise ein Mechanismus, der für einen gegebenen Input den „richtigen“ Output kennt und so die Korrektheit eines Systems überprüfen kann.

Bei klassischen Softwaresystemen:

- Gibt es oft **formale Spezifikationen** oder **deterministische Regeln**.
- Das Orakel ist einfach: „Ist der Output gleich dem erwarteten Ergebnis?“

Bei KI-Systemen (z. B. Bilderkennung, Sprachverarbeitung, Empfehlungssysteme):

- Gibt es **keine eindeutige Spezifikation**.
- Der Output kann **mehrdeutig, probabilistisch oder subjektiv** sein.
- Beispiel: Ist ein Bild eine „Katze“ oder ein „Luchs“? → Unterschiedliche Menschen könnten unterschiedlich entscheiden.

Warum ist das problematisch?

- **Automatisiertes Testen wird erschwert.**
- **Fehler können unentdeckt bleiben**, weil es keine klare Referenz gibt.
- **Vergleichbarkeit von Modellen** wird schwierig.
- **Vertrauen und Nachvollziehbarkeit** leiden, besonders in sicherheitskritischen Anwendungen.

Methoden zur Umgehung des Testorakelproblems

1. Metamorphes Testen

- Nutzt bekannte Beziehungen zwischen Inputs und Outputs.
- Beispiel: Wenn ein Bild heller gemacht wird, sollte die Klassifikation gleichbleiben.
- Vorteil: Kein „richtiges“ Ergebnis nötig, nur **relationale Erwartungen**.

2. Vergleich mit Referenzsystemen

- Vergleich der Vorhersagen mit einem etablierten Modell oder menschlichen Experten.
- Beispiel: „Stimmt das Modell in 90 % der Fälle mit einem Radiologen überein?“

3. Crowdsourcing / Expertenbewertungen

- Menschen bewerten die Qualität oder Korrektheit von Vorhersagen.
- Besonders bei NLP, Empfehlungssystemen oder generativen Modellen.

4. Statistische Orakel

- Statt einzelner Outputs werden **Verteilungen** oder **Konfidenzintervalle** geprüft.
- Beispiel: „Liegt die Vorhersage innerhalb eines akzeptablen Bereichs?“

5. Explainability-Tools (z. B. LIME, SHAP)

- Helfen zu verstehen, *warum* ein Modell eine Entscheidung getroffen hat.
- Erhöht die **Transparenz** und erlaubt **indirekte Validierung**.

6. Testdaten mit Ground Truth

- Verwendung von sorgfältig annotierten Datensätzen mit „Goldstandard“-Labels.
- Beispiel: Medizinische Diagnosedaten, die von mehreren Experten validiert wurden.

Fazit

Das Testorakelproblem ist **nicht vollständig lösbar**, aber durch clevere Teststrategien und Hilfsmittel **beherrschbar**. Besonders wichtig ist ein **Mix aus automatisierten, statistischen und menschenzentrierten Methoden**, um die Qualität und Vertrauenswürdigkeit von KI-Systemen sicherzustellen.

8.8 Geeignete Testziele und Akzeptanzkriterien für die KI-spezifischen Qualitätsmerkmale eines KI-basierten System

Frage:

Ein Unternehmen entwickelt ein autonomes System zur medizinischen Diagnose. Welche Testziele und Akzeptanzkriterien sollten für Akzente wie Erklärbarkeit, Verzerrung und funktionale Sicherheit definiert werden?

Antwort:

Für ein Unternehmen, das ein **autonomes medizinisches Diagnosesystem** entwickelt, sind **Testziele** und **Akzeptanzkriterien** entscheidend, um Vertrauen, Sicherheit und regulatorische Konformität sicherzustellen. Hier ist eine strukturierte Übersicht zu den drei genannten Aspekten:

1. Erklärbarkeit (Explainability)

Testziele

- Sicherstellen, dass das System **nachvollziehbare Begründungen** für Diagnosen liefert.
- Überprüfen, ob medizinisches Fachpersonal die Erklärungen **versteht und validieren** kann.
- Evaluieren, ob Erklärungen **konsistent und stabil** sind.

Akzeptanzkriterien

- Die Erklärungen müssen für $\geq 90\%$ der Testfälle von Fachärzt:innen als **plausibel** bewertet werden.
- Verwendung von XAI-Methoden wie **LIME**, **SHAP** oder **Konfidenzvisualisierungen**.
- Erklärungen dürfen sich bei minimalen Input-Änderungen **nicht signifikant ändern** (Stabilitätsgrenze definieren).
- Dokumentation der Erklärlogik für **Audits und regulatorische Prüfungen**.

2. Verzerrung (Bias & Fairness)

Testziele

- Identifikation und Quantifizierung von **systematischen Benachteiligungen** (z. B. nach Geschlecht, Alter, Ethnie).
- Sicherstellen, dass das System **gleichwertige Leistung** für alle relevanten Patientengruppen erbringt.

Akzeptanzkriterien

- Unterschied in der Diagnosegenauigkeit zwischen Gruppen darf **nicht $> 5\%$** betragen (z. B. Sensitivität bei Männern vs. Frauen).
- Einsatz von Fairness-Metriken wie:
 - *Equal Opportunity*
 - *Demographic Parity*
 - *Disparate Impact Ratio*
- Durchführung von **Bias-Audits** mit realen und synthetischen Testdaten.
- Dokumentation der Maßnahmen zur **Bias-Reduktion** (z. B. Rebalancing, Fairness Constraints).

3. Funktionale Sicherheit (Functional Safety)

Testziele

- Sicherstellen, dass das System in **kritischen Situationen sicher reagiert** (z. B. bei Unsicherheit, Datenfehlern).
- Verifikation, dass das System **Fehler erkennt und korrekt eskaliert** (z. B. Übergabe an menschliche Ärzt:innen).

Akzeptanzkriterien

- Bei Unsicherheiten > definierter Schwelle (z. B. Konfidenz < 70 %) muss das System eine **menschliche Überprüfung anfordern**.
- Nachweis der Einhaltung von Normen wie **ISO 14971 (Risikomanagement)** oder **IEC 62304 (Software für Medizinprodukte)**.
- Durchführung von **Failure Mode and Effects Analysis (FMEA)**.
- Nachweis, dass das System bei simulierten Fehlern **sicher in einen definierten Zustand übergeht** (z. B. Diagnose verweigern statt falscher Diagnose).

Ergänzend: Weitere sinnvolle Testziele

- **Robustheit** gegenüber verrauschten oder unvollständigen Daten.
- **Transparenz** der Datenquellen und Trainingsprozesse.
- **Auditierbarkeit:** Jede Entscheidung muss **rückverfolgbar** sein.

9.0 Kapitel: Methoden u. Verfahren für das Testen KI-basierter Systeme

– K2:

Techniken wie

Adversarial Attacks, Data Poisoning, Pairwise/AB-/Back-to-Back/Metamorphic/
Experience-Based Testing.

– K4:

Techniken auswählen;

– H2:

Metamorphic & Exploratives Testing anwenden

9.1 ML-Systemen dazu beitragen gegnerische Angriffe und Datenverunreinigungen zu verhindern

Frage:

Wie unterscheiden sich gegnerische Angriffe von Datenverunreinigungen, und wie kann Testen helfen, diese Risiken zu minimieren?

Antwort:

Das ist eine sehr gute und wichtige Frage, besonders im Kontext von KI-Systemen und maschinellem Lernen. Lass uns die Begriffe **gegnerische Angriffe** und **Datenverunreinigungen** klar voneinander abgrenzen und dann erklären, wie **Testen** helfen kann, diese Risiken zu minimieren.

1. Unterschied zwischen gegnerischen Angriffen und Datenverunreinigungen

Merkmal	Gegnerische Angriffe (Adversarial Attacks)	Datenverunreinigungen (Data Poisoning)
Ziel	Modell während der Nutzung täuschen	Modell während des Trainings manipulieren
Zeitpunkt	Beim Einsatz (z. B. in der Produktion)	Beim Training oder bei der Datenvorbereitung
Beispiel	Ein Bild wird leicht verändert, sodass ein Panda als Gibbon erkannt wird	Ein Angreifer fügt absichtlich falsche Labels in das Trainingsset ein
Art der Manipulation	Kleine, gezielte Änderungen an Eingabedaten	Falsche oder manipulierte Trainingsdaten
Risiko	Fehlentscheidungen im Betrieb	Langfristige Modellverzerrung oder Schwächung

Testen ist ein zentraler Bestandteil der **robusten KI-Entwicklung**. Hier sind einige Strategien:

Für **gegnerische Angriffe**:

- **Adversarial Testing**: Modelle gezielt mit manipulierten Eingaben testen, um ihre Robustheit zu prüfen.
- **Gradientenbasierte Angriffe simulieren** (z. B. FGSM, PGD), um Schwachstellen zu identifizieren.
- **Verteidigungsmechanismen testen**, z. B. Rauschen, Input-Normalisierung oder adversarial training.

Für **Datenverunreinigungen**:

- **Datenvalidierung und -bereinigung**: Automatisierte Checks auf Ausreißer, Label-Inkonsistenzen oder ungewöhnliche Muster.
- **Trainingsdaten-Monitoring**: Herkunft und Qualität der Daten regelmäßig prüfen.
- **Robustheits-Tests**: Modelle mit verschiedenen Subsets der Daten trainieren und vergleichen (z. B. Cross-Validation mit gezielten Ausschlüssen).

Allgemeine Testmethoden:

- **Unit-Tests für Datenpipelines**: Sicherstellen, dass Daten korrekt verarbeitet werden.
- **Explainability-Tests**: Prüfen, ob das Modell nachvollziehbare Entscheidungen trifft.
- **Bias- und Fairness-Tests**: Sicherstellen, dass keine systematischen Verzerrungen vorliegen.

Fazit

Gegnerische Angriffe und **Datenverunreinigungen** sind zwei unterschiedliche, aber ernstzunehmende Bedrohungen für KI-Systeme. Durch gezieltes **Testen** – sowohl während der Entwicklung als auch im Betrieb – lassen sich viele dieser Risiken frühzeitig erkennen und abmildern.

9.2 Paarweises Testen KI-basierter Systeme

Frage:

Warum ist paarweises Testen besonders nützlich für KI-Systeme mit vielen Eingabeparameter, und wie funktioniert es?

Antwort:

Paarweises Testen ist besonders nützlich für KI-Systeme mit vielen Eingabeparametern, weil es eine **effiziente Methode ist, um Fehler zu entdecken**, die durch **Wechselwirkungen zwischen zwei Parametern** entstehen – ohne alle möglichen Kombinationen testen zu müssen.

Warum ist paarweises Testen bei vielen Parametern so hilfreich?

KI-Systeme (z. B. ML-Modelle, neuronale Netze, Entscheidungsbäume) haben oft:

- viele **Hyperparameter** (z. B. Lernrate, Batch-Größe, Aktivierungsfunktion),
- viele **Eingabemerkmale** (Features),
- viele **Konfigurationsoptionen** (z. B. Vorverarbeitung, Normalisierung, Regularisierung).

Wenn man alle möglichen Kombinationen testen wollte, würde die Anzahl der Tests **exponentiell wachsen** – das ist in der Praxis nicht machbar.

Paarweises Testen reduziert die Testanzahl drastisch, indem es sicherstellt, dass **jede mögliche Kombination von zwei Parametern mindestens einmal getestet wird**.

Wie funktioniert paarweises Testen?

1. Parameter und Werte definieren

Beispiel:

- Optimierer: {SGD, Adam}
- Lernrate: {0.01, 0.1}
- Batch-Größe: {32, 64}
- Aktivierungsfunktion: {ReLU, Tanh}

2. Alle möglichen 2er-Kombinationen erzeugen

z. B. (SGD, 0.01), (Adam, 0.1), (32, ReLU), usw.

3. Minimale Menge an Testfällen generieren, die alle 2er-Kombinationen abdecken.

Das geschieht meist mit einem Tool oder Algorithmus (z. B. IPOG, PICT, ACTS).

4. Testfälle ausführen und auf Fehler, Instabilitäten oder unerwartetes Verhalten prüfen.

Vorteile für KI-Systeme

- **Effizienz:** Deckt viele Fehlerquellen mit wenigen Tests ab.
- **Skalierbarkeit:** Auch bei vielen Parametern praktikabel.
- **Fehlererkennung:** Besonders gut bei Interaktionsfehlern zwischen Parametern.
- **Automatisierbar:** Lässt sich gut in Testpipelines integrieren.

Fazit

Paarweises Testen ist eine **strategisch kluge Methode**, um bei komplexen KI-Systemen mit vielen Parametern **systematisch und effizient zu testen**, ohne in eine Testexplosion zu geraten. Es ist besonders nützlich in der **Modellvalidierung**, **Hyperparameteroptimierung** und **Robustheitsprüfung**.

HO 9.2 Anwenden von paarweisem Testen für Entwurf und Ausführung von Testfällen für ein KI-basiertes System

Aufgabe:

Nutzen Sie ein Tool zu den paarweisen Kombinationen, um Testfälle für ein KI-System mit mindestens fünf Parameter zu erstellen und führen Sie diese aus

Lösung:

9.3 Vergleichendes Testen von KI basierten Systeme

Frage:

Was ist vergleichendes Testen (Back-to-Back-Testing), und wie kann es bei fehlenden Testorakeln helfen?

Antwort:

Vergleichendes Testen – auch bekannt als **Back-to-Back-Testing** – ist eine Testmethode, die besonders dann hilfreich ist, wenn **kein zuverlässiges Testorakel** vorhanden ist. Ein *Testorakel* ist normalerweise ein Mechanismus (z. B. eine Spezifikation oder ein Referenzwert), der angibt, ob das Ergebnis eines Tests korrekt ist.

Was ist vergleichendes Testen (Back-to-Back-Testing)?

Beim vergleichenden Testen werden **zwei (oder mehr) Systeme** mit denselben Eingaben getestet, und ihre **Ausgaben werden miteinander verglichen**. Die Idee ist: Wenn zwei Systeme bei denselben Eingaben **unterschiedliche Ergebnisse** liefern, liegt **mindestens bei einem ein Fehler** vor.

Wie funktioniert das konkret?

1. **Zwei Systeme bereitstellen:**
 - z. B. ein altes und ein neues KI-Modell,
 - oder ein Modell in zwei verschiedenen Frameworks (z. B. TensorFlow vs. PyTorch),
 - oder ein Modell und eine Referenzimplementierung.
 2. **Gleiche Eingaben verwenden:**
 - z. B. dieselben Bild-, Text- oder Tabellendaten.
 3. **Ausgaben vergleichen:**
 - numerisch (z. B. Klassifikationswahrscheinlichkeiten),
 - semantisch (z. B. gleiche Vorhersageklassen),
 - oder mit Toleranz (z. B. Abweichung < 1%).
 4. **Abweichungen analysieren:**
 - Wenn Unterschiede auftreten, wird untersucht, ob sie durch Fehler, Rundungsdifferenzen oder Modellunterschiede erklärbar sind.
-

Warum ist das bei fehlenden Testorakeln hilfreich?

In vielen KI-Anwendungen gibt es **keine eindeutige „richtige“ Antwort** – z. B. bei:

- **Sprachmodellen** (mehrere plausible Antworten),
- **Bilderkennung** (mehrdeutige Objekte),
- **Empfehlungssystemen** (subjektive Relevanz).

In solchen Fällen hilft Back-to-Back-Testing, **Inkonsistenzen oder Regressionen** zu erkennen, **ohne zu wissen, was „richtig“ ist** – nur durch den Vergleich mit einem etablierten System.

Vorteile

- Kein Orakel nötig
- Gut für Regressionstests
- Automatisierbar
- Ideal bei Migrationen oder Refactorings

Herausforderungen

- Unterschiedliche Systeme können legitimerweise leicht abweichen
 - Vergleichskriterien müssen gut gewählt sein (z. B. Toleranzgrenzen)
 - Nicht alle Unterschiede sind Fehler
-

Fazit

Back-to-Back-Testing ist eine clevere Strategie, um **Vertrauen in neue oder veränderte KI-Systeme** zu gewinnen – besonders dann, wenn es **keine eindeutige Referenzlösung** gibt. Es ersetzt kein vollständiges Testorakel, aber es ist ein **wertvolles Werkzeug**, um **Fehler, Inkonsistenzen oder unerwartete Änderungen** frühzeitig zu erkennen.

9.4 A/B Testen von KI-basierten Systeme

Frage:

Wie funktioniert A/B Testen bei KI-Systemen, und in welchen Szenarien ist es besonders wichtig?

Antwort:

A/B-Testen ist eine bewährte Methode zur Bewertung von Änderungen in KI-Systemen, insbesondere wenn diese **direkt mit Nutzerinteraktionen** zu tun haben. Es hilft dabei, datenbasierte Entscheidungen zu treffen, indem zwei Varianten eines Systems **unter realen Bedingungen verglichen** werden.

Wie funktioniert A/B-Testen bei KI-Systemen?

1. **Zwei Varianten erstellen:**
 - **A:** das bestehende System (Kontrollgruppe)
 - **B:** das neue System oder eine Variante (Testgruppe)
2. **Zufällige Nutzerzuweisung:**
 - Nutzer oder Anfragen werden zufällig auf A oder B verteilt.
 - Beispiel: 50 % der Nutzer sehen Empfehlungen von Modell A, 50 % von Modell B.
3. **Metriken definieren:**
 - z. B. Klickrate, Conversion Rate, Fehlerrate, Nutzerzufriedenheit, Modelllatenz.
4. **Daten sammeln und vergleichen:**
 - Statistische Auswertung zeigt, ob Variante B **signifikant besser** (oder schlechter) ist als A.
5. **Entscheidung treffen:**
 - Wenn B besser ist → Rollout.
 - Wenn nicht → Beibehalten oder weiter optimieren.

Typische Einsatzszenarien für A/B-Tests in KI-Systemen

Szenario	Beispiel
Empfehlungssysteme	Testen neuer Algorithmen für Produktempfehlungen
Sprachmodelle / Chatbots	Vergleich von Antwortqualität oder Dialogführung
Suchalgorithmen	Bewertung neuer Ranking-Strategien
Personalisierung	Unterschiedliche Modelle für Nutzersegmente
Computer Vision	Vergleich von Objekterkennungsmodellen in einer App
Modelloptimierung	Neue Hyperparameter oder Feature-Sets testen

Vorteile

- **Realitätsnah:** Test unter echten Bedingungen mit echten Nutzern
- **Objektiv:** Statistisch fundierte Entscheidungen
- **Risikoarm:** Neue Varianten werden nur einem Teil der Nutzer gezeigt

Herausforderungen

- **Ethik & Fairness:** Nutzer dürfen nicht benachteiligt werden
- **Datenmenge:** Signifikanz braucht oft viele Nutzer
- **Verzerrung vermeiden:** Z. B. durch saisonale Effekte oder Nutzerverhalten

Fazit

A/B-Testen ist ein **unverzichtbares Werkzeug**, um KI-Systeme **kontinuierlich zu verbessern**, ohne auf bloße Intuition angewiesen zu sein. Es ist besonders wichtig in **produktiven Systemen mit Nutzerinteraktion**, wo jede Änderung messbare Auswirkungen haben kann.

9.5 Metamorphes Testen von KI-basierten Systeme

Frage:

Wie funktioniert metamorphes Testen, und wie kann es fehlenden erwarteten Ergebnissen eingesetzt werden?

Antwort:

Metamorphes Testen (engl. *Metamorphic Testing*) ist eine Testmethode, die besonders dann hilfreich ist, wenn **kein erwartetes Ergebnis (Testorakel)** vorhanden ist – also wenn man **nicht genau weiß, was das „richtige“ Ergebnis sein sollte**. Das ist oft der Fall bei KI-Systemen, z. B. bei neuronalen Netzen, Sprachmodellen oder komplexen Optimierungsalgorithmen.

Wie funktioniert metamorphes Testen?

Statt das Ergebnis direkt zu prüfen, testet man **Beziehungen zwischen Eingaben und Ausgaben**, sogenannte **metamorphe Relationen**. Diese Relationen beschreiben, **wie sich die Ausgabe ändern sollte**, wenn man die Eingabe auf eine bestimmte Weise verändert.

Grundidee:

Wenn Eingabe A zu Ergebnis X führt, und man A zu A' verändert, dann sollte das Ergebnis X' eine bestimmte Beziehung zu X haben.

Beispiele für metamorphe Relationen in KI-Systemen

Anwendung	Eingabeveränderung	Erwartete Ausgabeveränderung
Bildklassifikation	Bild drehen um 90°	Klasse sollte gleich bleiben
Textklassifikation	Synonym ersetzen	Klassifikation sollte gleich bleiben
Regressionsmodell	Eingabe skalieren	Ausgabe sollte entsprechend skaliert sein
Empfehlungssystem	Nutzerprofil leicht ändern	Empfehlungen sollten sich leicht ändern

Wie hilft das bei fehlenden erwarteten Ergebnissen?

Wenn man **nicht weiß, was das richtige Ergebnis ist**, kann man trotzdem testen, ob das System **konsistent und logisch reagiert**:

- **Keine absolute Korrektheit nötig**, nur **relative Konsistenz**
- **Fehler aufdecken**, die klassische Tests übersehen würden
- **Robustheit und Generalisierungsfähigkeit** prüfen

Vorteile

- Ideal bei **Black-Box-Systemen** wie neuronalen Netzen
- Funktioniert **ohne Ground Truth**
- Deckt **systematische Fehler** auf
- Unterstützt **automatisiertes Testen**

Herausforderungen

- Metamorphe Relationen müssen **sinnvoll definiert** werden
- Nicht jede Relation ist **immer gültig** (z. B. bei stark nichtlinearen Modellen)
- Kann **falsch-positive oder -negative** Ergebnisse liefern, wenn Relationen zu streng oder zu locker sind

Fazit

Metamorphes Testen ist eine **leistungsfähige Strategie**, um KI-Systeme zu testen, wenn **kein klares Orakel vorhanden ist**. Es prüft nicht das Ergebnis selbst, sondern **logische Konsistenz** zwischen verwandten Eingaben und Ausgaben – und ist damit besonders wertvoll für **komplexe, datengetriebene Systeme**.

H0 9.5

Aufgabe:

Lösung:

9.6 Erfahrungsbasiertes Testen von KI-basierten Systeme

Frage:

Welche Rolle spielt exploratives Testen bei KI-Systemen, und wie kann es mit EDA kombiniert werden?

Antwort:

Exploratives Testen spielt eine zentrale Rolle bei der Qualitätssicherung von **KI-Systemen**, insbesondere in frühen Entwicklungsphasen oder bei Systemen mit **hoher Komplexität und Unsicherheit**. Es lässt sich hervorragend mit **Explorativer Datenanalyse (EDA)** kombinieren, um sowohl das Verhalten des Modells als auch die zugrunde liegenden Daten besser zu verstehen.

Was ist exploratives Testen bei KI-Systemen?

Exploratives Testen ist ein **nicht vollständig spezifizierter, erfahrungsbasierter Ansatz**, bei dem Tester:innen:

- Hypothesen aufstellen,
- gezielt mit dem System „experimentieren“,
- ungewöhnliche Eingaben ausprobieren,
- und **intuitiv nach Schwächen oder Inkonsistenzen suchen**.

Es ist besonders nützlich, wenn:

- Anforderungen unklar sind,
- das System komplex oder adaptiv ist (z. B. ML-Modelle),
- oder wenn automatisierte Tests nicht alle Fälle abdecken.

Was ist EDA (Explorative Datenanalyse)?

EDA ist ein analytischer Prozess, bei dem man:

- **Daten visuell und statistisch untersucht**,
- **Verteilungen, Ausreißer, Korrelationen** erkennt,
- und **Hypothesen über Datenqualität und -struktur** bildet.

Typische Werkzeuge: Histogramme, Boxplots, Scatterplots, PCA, Korrelationstabellen.

Wie ergänzen sich exploratives Testen und EDA?

Exploratives Testen	Explorative Datenanalyse
Testet das Verhalten des Modells	Untersucht die Eigenschaften der Daten
Nutzt kreative Eingaben und Szenarien	Nutzt Visualisierungen und Statistiken
Erkennt unerwartete Modellreaktionen	Erkennt Datenprobleme oder Verzerrungen
Kann durch EDA inspiriert werden	Liefert Hinweise für gezielte Tests

Beispiel-Kombination:

1. **EDA zeigt**, dass ein Feature stark schief verteilt ist.
2. **Explorativer Test** prüft, wie das Modell auf Extremwerte dieses Features reagiert.
3. **Ergebnis**: Modell ist instabil → Feature Engineering oder Regularisierung nötig.

Vorteile der Kombination

- **Früherkennung** von Daten- und Modellproblemen
- **Besseres Verständnis** für Modellverhalten
- **Gezieltere Testszenarien**
- **Höhere Testabdeckung** bei unbekannten Risiken

Fazit

Die Kombination aus **explorativem Testen** und **EDA** ist besonders wertvoll für KI-Systeme, weil sie **intuitives Testverhalten mit datengetriebener Analyse** verbindet. Sie hilft, **blinde Flecken** zu entdecken, die bei rein automatisierten Tests oft übersehen werden.

H0 9.6 Anwenden von explorativem Test auf ein KI-basiertes System

Aufgabe:

Führen Sie eine Datentour und eine explorative Datenanalyse (EDA) durch, um mögliche Verzerrungen oder Datenprobleme zu identifizieren

Lösung:

9.7 Geeignetes Testverfahren für ein bestimmtes Szenario

Frage:

Ein Unternehmen entwickelt ein KI-System zur Gesichtserkennung. Welche Testverfahren sollten eingesetzt werden, um Risiken wie Verzerrung, Überanpassung und fehlende Erklärbarkeit zu adressieren?

Antwort:

Beim Testen eines **KI-Systems zur Gesichtserkennung** ist es entscheidend, gezielt gegen Risiken wie **Verzerrung (Bias)**, **Überanpassung (Overfitting)** und **fehlende Erklärbarkeit (Lack of Explainability)** vorzugehen. Hier sind die wichtigsten **Testverfahren und Strategien**, die eingesetzt werden sollten:

1. Testverfahren gegen Verzerrung (Bias)

Ziel: Sicherstellen, dass das System **fair** und **nichtdiskriminierend** gegenüber bestimmten Gruppen ist.

Empfohlene Tests:

- **Demografische Fairness-Tests:** Prüfen, ob die Erkennungsrate über Altersgruppen, Geschlechter, Hautfarben oder ethnische Gruppen hinweg konsistent ist.
- **Disparate Impact Analysis:** Statistische Analyse, ob bestimmte Gruppen systematisch benachteiligt werden.
- **Stratifizierte Testsets:** Getrennte Testsets für verschiedene Subgruppen, um gezielt Unterschiede zu erkennen.
- **Counterfactual Fairness Testing:** Testen, ob das Ergebnis gleichbleibt, wenn nur sensitive Merkmale (z. B. Geschlecht) verändert werden.

2. Testverfahren gegen Überanpassung (Overfitting)

Ziel: Sicherstellen, dass das Modell **generalisierbar** ist und nicht nur auf Trainingsdaten gut funktioniert.

Empfohlene Tests:

- **Cross-Validation:** Mehrfache Aufteilung der Daten in Trainings- und Testsets.
 - **Holdout-Testsets:** Verwendung von echten, nie gesehenen Daten zur Validierung.
 - **Adversarial Testing:** Testen mit leicht veränderten Bildern (z. B. Rauschen, Beleuchtung), um Robustheit zu prüfen.
 - **Drift Detection:** Überwachung der Modelleleistung über Zeit, um Datenverschiebungen zu erkennen.
-

3. Testverfahren zur Verbesserung der Erklärbarkeit

Ziel: Das Modellverhalten soll **transparent und nachvollziehbar** sein – besonders wichtig bei sicherheitskritischen Anwendungen wie Gesichtserkennung.

Empfohlene Tests:

- **Explainability-Tools einsetzen:**
 - z. B. **LIME**, **SHAP**, **Grad-CAM** für visuelle Erklärungen.
- **Saliency Maps analysieren:** Zeigen, welche Bildbereiche das Modell zur Entscheidung heranzieht.
- **Plausibilitätsprüfung durch Experten:** Prüfen, ob die Erklärungen mit menschlicher Intuition übereinstimmen.
- **Kontrastive Tests:** Was passiert, wenn ein Gesicht leicht verändert wird (z. B. Brille, Frisur)?

Zusätzliche Teststrategien

Testart	Nutzen
Back-to-Back-Testing	Vergleich mit älteren Modellen oder anderen Frameworks
Metamorphes Testen	Prüfen, ob das Modell bei Transformationen (z. B. Rotation, Skalierung) konsistent bleibt
Explorative Testen	Kreatives Testen mit ungewöhnlichen oder grenzwertigen Eingaben
A/B-Tests	Vergleich von Modellvarianten im realen Einsatz (z. B. Nutzerfeedback, Fehlerraten)

Fazit

Ein robustes Testkonzept für Gesichtserkennungssysteme sollte **technische, ethische und gesellschaftliche Risiken** abdecken. Die Kombination aus **automatisierten Tests, statistischer Analyse und menschlicher Bewertung** ist entscheidend, um ein **verlässliches, faires und erklärbares** System zu entwickeln.

10.0 Kapitel: Testumgebung für KI-basierte Systeme

– K2:

Unterschiede zu klassischen Testumgebungen.

– K2:

Vorteile virtueller Testumgebungen.

10.1 Die wichtigsten Faktoren bei der Unterscheidung von konventionellen Testumgebung und insbesondere bei selbst lernenden oder autonomen Systemen

Frage:

Welche besonderen Anforderungen stellen KI-Systeme an Testumgebungen, insbesondere bei selbstlernenden oder autonomen Systemen?

Antwort:

KI-Systeme, insbesondere selbstlernende oder autonome Systeme, stellen besondere Anforderungen an Testumgebungen, die weit über klassische Softwaretests hinausgehen. Hier sind die wichtigsten Aspekte:

Dynamik und Lernfähigkeit

Selbstlernende Systeme verändern ihr Verhalten über die Zeit. Das bedeutet:

- **Testfälle müssen wiederholt werden**, da sich das Systemverhalten durch neue Trainingsdaten ändern kann.
- **Regressionstests** sind komplexer, weil es keine festen „richtigen“ Ergebnisse gibt – das System kann auf neue Weise korrekt reagieren.

Realitätsnahe und kontrollierte Testumgebungen

Autonome Systeme (z. B. autonome Fahrzeuge oder Roboter) benötigen:

- **Simulationsumgebungen**, die realistische Szenarien abbilden (z. B. Verkehr, Wetter, menschliches Verhalten).
- **Edge-Case-Tests**, um seltene, aber kritische Situationen zu prüfen (z. B. Kind läuft auf die Straße).

Erklärbarkeit und Nachvollziehbarkeit

Black-Box-Modelle wie Deep Learning erschweren die Fehlersuche:

- **Testumgebungen müssen Logging und Monitoring** auf Modellentscheidungen ermöglichen.
- **Explainable (XAI)**-Methoden sollten integriert sein, um Entscheidungen zu analysieren.

Datenqualität und -vielfalt

Die Qualität der Trainings- und Testdaten ist entscheidend:

- **Bias-Tests:** Prüfen, ob das System diskriminierende Entscheidungen trifft.
- **Robustheitstests:** Wie reagiert das System auf verrauschte, manipulierte oder unvollständige Daten?

Kontinuierliches Testen und Validieren

Da sich Modelle durch neue Daten verändern:

- **MLOps-Ansätze** (Machine Learning Operations) sind nötig, um kontinuierliches Testen, Deployment und Monitoring zu ermöglichen.
- **A/B-Tests** und **Shadow Deployments** helfen, neue Modelle risikofrei zu evaluieren.

Sicherheit und ethische Aspekte

- **Adversarial Testing:** Wie reagiert das System auf gezielte Manipulationen?
- **Sicherheitsüberprüfungen:** Besonders bei autonomen Systemen mit physischer Wirkung (z. B. Drohnen, Fahrzeuge).

Fazit: Anforderungen an Testumgebungen

Eine geeignete Testumgebung für KI-Systeme sollte:

- **Skalierbar und automatisierbar** sein
- **Realistische Szenarien simulieren**
- **Erklärbarkeit und Monitoring** unterstützen
- **Datenvielfalt und -qualität** sicherstellen
- **Sicherheits- und Ethiktests** integrieren

10.2 Vorteile von einer virtuellen Testumgebung für KI-basierte Systeme

Frage:

Welche Vorteile bieten virtuelle Testumgebungen beim Testen von KI-Systemen und welche Szenarien sind sie besonders nützlich?

Antwort:

Virtuelle Testumgebungen bieten beim Testen von KI-Systemen eine Vielzahl von Vorteilen, insbesondere bei komplexen, dynamischen oder sicherheitskritischen Anwendungen. Hier sind die wichtigsten Vorteile und typische Einsatzszenarien:

Vorteile virtueller Testumgebungen

Kostenersparnis

- Keine teure Hardware oder reale Infrastruktur nötig.
- Tests können in großem Maßstab durchgeführt werden, ohne physische Ressourcen zu verbrauchen.

Sicherheit

- Kritische oder gefährliche Szenarien (z. B. Notbremsung bei autonomen Fahrzeugen) können risikofrei simuliert werden.
- Kein Risiko für Menschen, Maschinen oder Umwelt.

Skalierbarkeit

- Tausende Szenarien können parallel getestet werden.
- Ermöglicht umfangreiche Tests in kurzer Zeit.

Wiederholbarkeit

- Szenarien können exakt reproduziert werden, was für Debugging und Vergleich von Modellversionen essenziell ist.

Flexibilität

- Schnelles Anpassen von Parametern (z. B. Wetter, Lichtverhältnisse, Verkehrsaufkommen).
- Testen unter extremen oder seltenen Bedingungen möglich.

Datengenerierung

- Virtuelle Umgebungen können synthetische Daten erzeugen, die für das Training und Testen von KI-Modellen genutzt werden können.
 - Besonders hilfreich bei Datenknappheit oder Datenschutzanforderungen.
-

Besonders nützliche Szenarien

Autonome Fahrzeuge

- Simulation von Verkehrssituationen, Fußgängern, Wetterbedingungen.
- Testen von Fahrverhalten, Notfallreaktionen, Sensorfusion.

Robotik

- Navigation in komplexen Umgebungen (z. B. Lagerhäuser, Krankenhäuser).
- Interaktion mit Menschen oder anderen Robotern.

KI-gestützte Entscheidungsfindung

- Testen von Empfehlungssystemen, Chatbots oder Diagnosesystemen in kontrollierten Gesprächs- oder Entscheidungsszenarien.

Sicherheitskritische Systeme

- KI in der Luftfahrt, Medizin oder Industrieautomatisierung.
- Simulation von Systemausfällen, Angriffen oder Fehlverhalten.

Virtuelle Agenten und Spiele-KI

- Training und Testen von KI-Spielern in simulierten Spielwelten.
 - Analyse von Strategien und Verhalten.
-

11.0 Kapitel: Einsatz von KI für Tests

– K2:

KI-Nutzung in

Defektanalyse, Testspannungserstellung, Regressionstest-Optimierung, UI-Testing, Defektvorhersage, Interface-Testing – inkl. optional einfache Defektvorhersage (H2)

11.1 Kategorisieren der bei Softwaretest verwendeten KI-Techniken

Frage:

Welche drei Hauptkategorien von KI-Techniken werden im Softwaretest eingesetzt, und welche Aufgaben unterstützen sie?

Antwort:

Im Softwaretest werden **drei Hauptkategorien von KI-Techniken** eingesetzt, die jeweils unterschiedliche Aufgaben unterstützen. Hier ist eine strukturierte Übersicht:

Maschinelles Lernen (ML)

Unterstützte Aufgaben:

- **Testfallpriorisierung:** ML-Modelle analysieren historische Testdaten, um die wichtigsten Testfälle zuerst auszuführen.
- **Fehlerprognose:** Vorhersage, welche Module wahrscheinlich fehleranfällig sind.
- **Anomalieerkennung:** Erkennung ungewöhnlicher Systemverhalten oder Testergebnisse.
- **Testdaten-Generierung:** Automatisches Erzeugen realistischer Eingabedaten.

Typische Techniken:

- Klassifikation (z. B. fehlerhaft/nicht fehlerhaft)
- Clustering (z. B. Gruppierung ähnlicher Fehler)
- Regressionsmodelle (z. B. Vorhersage von Fehlerhäufigkeit)

Natural Language Processing (NLP)

Unterstützte Aufgaben:

- **Automatisierte Testfallgenerierung** aus Anforderungen in natürlicher Sprache.
- **Analyse von Bug-Reports:** Extraktion relevanter Informationen aus Fehlerbeschreibungen.
- **Anforderungsanalyse:** Erkennung von Inkonsistenzen oder Lücken in Spezifikationen.
- **Chatbots für Testautomatisierung:** Unterstützung von Testern durch dialogbasierte Systeme.

Typische Techniken:

- Textklassifikation
- Named Entity Recognition (NER)
- Semantische Ähnlichkeitsanalyse

Symbolische KI / Wissensbasierte Systeme

Unterstützte Aufgaben:

- **Regelbasierte Testfallgenerierung:** Ableitung von Tests aus formalen Regeln oder Modellen.
- **Constraint-Solving:** Generierung von Eingaben, die bestimmte Bedingungen erfüllen.
- **Fehlerdiagnose:** Logikbasierte Analyse von Ursache-Wirkungs-Beziehungen bei Fehlern.

Typische Techniken:

- Logikbasierte Inferenzsysteme
- Entscheidungsbäume
- Ontologien und Wissensgraphen

Fazit:

Diese drei Kategorien ergänzen sich oft:

- **ML** bringt datengetriebene Intelligenz,
- **NLP** erschließt unstrukturierte Informationen,
- **Symbolische KI** sorgt für formale Präzision.

H0 11.1 Erläutern von Tätigkeiten im Testbereich, bei denen der Einsatz von KI-Techniken weniger wahrscheinlich ist.

Aufgabe:

Diskutieren Sie, welche Testaktivitäten sich derzeit nicht sinnvoll durch KI automatisieren lassen und warum?

Lösung:

Das Testen mit KI bietet viele Automatisierungsmöglichkeiten, aber **nicht alle Testaktivitäten lassen sich derzeit sinnvoll durch KI automatisieren**. Hier sind einige zentrale Bereiche, in denen KI an ihre Grenzen stößt – und warum:

Exploratives Testen

Warum nicht automatisierbar?

- Exploratives Testen erfordert **kreatives Denken**, Intuition und spontane Entscheidungen.
- Tester*innen entdecken Fehler durch **unvorhersehbare Interaktionen**, die nicht leicht modellierbar sind.
- KI kann Muster erkennen, aber **nicht kreativ** „um die Ecke denken“ wie ein Mensch.

Usability- und UX-Tests

Warum nicht automatisierbar?

- Bewertung von **Benutzerfreundlichkeit, Ästhetik und emotionaler Wirkung** ist subjektiv.
- KI kann zwar Metriken analysieren (z. B. Klickpfade), aber **nicht menschliche Wahrnehmung oder Frustration** nachvollziehen.
- Tests erfordern oft **direktes Feedback von echtem Nutzer*innen**.

3. Ethik- und Vertrauenswürdigkeitstests

Warum nicht automatisierbar?

- Beurteilung, ob ein System **ethisch vertretbar** oder **gesellschaftlich akzeptabel** ist, hängt von **Kontext, Kultur und Werten** ab.
- KI kann keine moralischen Entscheidungen treffen oder **gesellschaftliche Normen** vollständig verstehen.

4.

Warum nicht automatisierbar?

- Anforderungen sind oft **mehrdeutig, widersprüchlich oder unvollständig**.
- KI kann helfen, Anforderungen zu analysieren, aber **nicht sicher beurteilen**, ob sie korrekt, vollständig oder sinnvoll sind.
- Menschliche Interpretation bleibt entscheidend.

5. Stakeholder-Kommunikation und Teststrategie

Warum nicht automatisierbar?

- Kommunikation mit Projektbeteiligten erfordert **Empathie, Verhandlungsgeschick und Kontextverständnis**.
- Entscheidungen über Testziele, Prioritäten und Risiken sind **strategisch und oft politisch motiviert**.

Fazit:

KI ist stark in **datengetriebenen, repetitiven und regelbasierten Aufgaben**, aber schwach in Bereichen, die:

- **Kreativität**
 - **Subjektivität**
 - **menschliches Urteilsvermögen**
 - **soziale Interaktion** erfordern.
-

11.2 Erläutern wie KI bei der Analyse neuer Fehlerzustände unterstützen kann

Frage:

Wie kann KI bei der Kategorisierung, Priorisierung und Zuweisung von Fehlerberichten helfen?

Antwort:

KI kann bei der **Kategorisierung, Priorisierung und Zuweisung** von Fehlerberichten (Bug Reports) erheblich unterstützen, insbesondere in großen Softwareprojekten mit vielen eingehenden Meldungen. Hier ist eine detaillierte Übersicht, wie das funktioniert:

Kategorisierung von Fehlerberichten

Wie hilft KI?

- **Natural Language Processing (NLP)** analysiert die Beschreibung von Fehlern.
- **Textklassifikationsmodelle** (z. B. auf Basis von BERT oder GPT) ordnen Fehler automatisch Kategorien zu wie:
 - Funktionaler Fehler
 - UI/UX-Problem
 - Performance-Problem
 - Sicherheitslücke

Vorteile:

- Schnellere Sortierung großer Mengen an Bug Reports
 - Einheitliche Klassifikation trotz unterschiedlicher Schreibweisen
-

Wie hilft KI?

- **Maschinelles Lernen** nutzt historische Daten (z. B. Fehlerauswirkungen, Häufigkeit, betroffene Nutzergruppen), um die **Wichtigkeit** eines Fehlers vorherzusagen.
- **Regression oder Klassifikation** zur Einteilung in Prioritätsstufen (z. B. „kritisch“, „hoch“, „niedrig“).

Eingehende Faktoren:

- Anzahl betroffener Nutzer
- Systemkomponenten
- Wiederholbarkeit
- Zusammenhang mit früheren kritischen Fehlern

Vorteile:

- Schnellere Reaktion auf kritische Probleme
- Bessere Ressourcenzuteilung im Entwicklungsteam

Automatische Zuweisung an Entwicklerteams

Wie hilft KI?

- **Empfehlungssysteme** analysieren:
 - Wer hat ähnliche Fehler in der Vergangenheit behoben?
 - Wer ist Experte für das betroffene Modul?
- **Graph basierte Modelle** oder **Clustering** helfen, passende Teams oder Personen zu identifizieren.

Vorteile:

- Reduziert manuelle Koordination
- Schnellere Bearbeitung durch passende Zuständigkeit

Beispiel-Workflow mit KI-Unterstützung:

1. **Bug wird eingereicht** (z. B. über ein Ticketsystem).
 2. **NLP-Modell analysiert den Text** → ordnet Kategorie zu.
 3. **ML-Modell bewertet die Dringlichkeit** → setzt Priorität.
 4. **Zuweisungsmodell schlägt Entwicklerteam vor** → Ticket wird automatisch weitergeleitet.
-

11.3 Erläutern wie KI bei der Erstellung von Testfällen helfen kann

Frage:

Welche Vorteile bietet der Einsatz von KI bei der automatisierte Testfallgenerierung und welche Einschränkungen bestehen?

Antwort:

Der Einsatz von **Künstlicher Intelligenz (KI)** bei der **automatisierten Testfallgenerierung** bietet viele Vorteile, aber auch einige Einschränkungen. Hier ist eine strukturierte Übersicht:

Vorteile der KI-gestützten Testfallgenerierung

Zeit- und Kostenersparnis

- KI kann automatisch Testfälle aus Anforderungen, Code oder Benutzerverhalten ableiten.
- Reduziert manuellen Aufwand bei der Testfallerstellung erheblich.

Höhere Testabdeckung

- KI erkennt Muster und Abhängigkeiten, die manuell leicht übersehen werden.
- Generiert Testfälle auch für seltene oder komplexe Pfade im Code.

Dynamische Anpassung

- Testfälle können automatisch aktualisiert werden, wenn sich Anforderungen oder Code ändern.
- Besonders nützlich in agilen oder DevOps-Umgebungen.

Nutzung natürlicher Sprache

- NLP-Techniken ermöglichen die Ableitung von Testfällen direkt aus User Stories oder Spezifikationen in natürlicher Sprache.

Fehlermustererkennung

- KI kann historische Fehlerdaten analysieren und gezielt Testfälle für ähnliche Schwachstellen erzeugen.

Einschränkungen und Herausforderungen

Qualität der Eingabedaten

- KI ist nur so gut wie die Daten, mit denen sie trainiert wird.
- Ungenaue, unvollständige oder widersprüchliche Anforderungen führen zu schlechten Testfällen.

Mangel an Domänenwissen

- KI versteht nicht automatisch den fachlichen Kontext.
- Testfälle können technisch korrekt, aber fachlich irrelevant sein.

Erklärbarkeit und Nachvollziehbarkeit

- Automatisch generierte Testfälle sind oft schwer nachvollziehbar.
- Tester müssen verstehen, **warum** ein Testfall generiert wurde, um ihn zu validieren.

Wartung und Kontrolle

- KI-generierte Testfälle müssen regelmäßig überprüft und ggf. angepasst werden.
- Gefahr von „Testfall-Spam“: zu viele, aber wenig nützliche Testfälle.

Begrenzte Kreativität

- KI kann keine kreativen, explorativen Tests erzeugen.
- Menschliche Intuition bleibt bei unvorhersehbaren Szenarien unersetzlich.

Fazit:

KI ist ein **mächtiges Werkzeug zur Unterstützung**, aber kein vollständiger Ersatz für menschliche Tester. Sie eignet sich besonders gut für:

- **Regressionstests**
- **Pfadabdeckung**
- **Testfallgenerierung aus Anforderungen**

Aber sie stößt an Grenzen bei:

- **Explorativem Testen**
- **Fachlich komplexen oder ethisch sensiblen Szenarien**

11.4 Erklären wie KI bei der Optimierung von Regressionsfällen helfen kann

Frage:

Wie kann KI helfen, Regressionsfälle effizienter zu gestalten, ohne die Fehlererkennung zu beeinträchtigen?

Antwort:

KI kann die **Effizienz von Regressionsfällen** deutlich verbessern, ohne die **Fehlererkennung zu beeinträchtigen**, indem sie gezielt hilft, **relevante Tests auszuwählen, zu priorisieren und zu optimieren**. Hier sind die wichtigsten Ansätze:

1. Testfall-Priorisierung mit Machine Learning

Wie es funktioniert:

- ML-Modelle analysieren historische Testdaten, Codeänderungen und Fehlerhäufigkeit.
- Sie bewerten, **welche Testfälle mit hoher Wahrscheinlichkeit Fehler aufdecken**.

Vorteile:

- Kritische Tests werden zuerst ausgeführt.
- Spart Zeit bei großen Test-Suiten.

2. Impact-Analyse durch KI

Wie es funktioniert:

- KI analysiert, **welche Teile des Codes sich geändert haben** (z. B. durch Git-Diffs).
- Nur die Testfälle, die diese Bereiche betreffen, werden ausgeführt.

Vorteile:

- Vermeidet unnötige Tests.
- Reduziert Testlaufzeiten drastisch.

3. Anomalie Erkennung bei Testergebnissen

Wie es funktioniert:

- KI erkennt **ungewöhnliche Muster** in Testergebnissen (z. B. plötzliche Laufzeitverlängerung, instabile Tests).
- Frühzeitige Erkennung von **nicht offensichtlichen Fehlern**.

Vorteile:

- Höhere Testqualität.
 - Schnellere Reaktion auf neue Probleme.
-

4. Automatische Testfall-Generierung bei Änderungen

Wie es funktioniert:

- KI kann bei Codeänderungen automatisch neue Testfälle vorschlagen oder generieren.
- Besonders nützlich bei Refactorings oder neuen Features.

Vorteile:

- Höhere Testabdeckung.
- Weniger manuelle Arbeit.

5. Testreduzierung ohne Qualitätsverlust

Wie es funktioniert:

- KI identifiziert **redundante oder überlappende Testfälle**.
- Nur die **repräsentativsten Tests** werden ausgeführt.

Vorteile:

- Schnellere Testzyklen.
- Geringerer Ressourcenverbrauch.

Wichtig: Qualitätssicherung bleibt entscheidend

Obwohl KI hilft, Tests effizienter zu gestalten, ist **menschliche Kontrolle weiterhin notwendig**, um sicherzustellen, dass:

- **kritische Tests nicht versehentlich ausgelassen** werden,
- **neue Fehlerarten erkannt** werden,
- und die **Teststrategie mit den Projektzielen übereinstimmt**.

11.5 Erklären wie KI bei den Fehlervorhersagen helfen kann

Frage:

Welche Datenquellen nutzt die KI zur Fehlervorhersagen und wie kann dies zur Testpriorisierung helfen?

Antwort:

Für **Fehlervorhersagen** in Softwareprojekten nutzt KI eine Vielzahl von **Datenquellen**, die aus dem Entwicklungs- und Testprozess stammen. Diese Vorhersagen können dann gezielt zur **Testpriorisierung** eingesetzt werden, um effizienter und risikoorientierter zu testen.

Wichtige Datenquellen für Fehlervorhersage durch KI

1. Historische Fehlerdaten

- Frühere Bug-Reports, Fehlerklassen, Schweregrade
- Zeit bis zur Behebung, betroffene Komponenten

2. Code-Metriken

- Komplexität (z. B. zyklomatische Komplexität)
- Anzahl der Codezeilen (LOC)
- Anzahl der Änderungen (Churn)
- Kopplung und Kohäsion

3. Versionskontrollsysteme (z. B. Git)

- Commit-Historie
- Häufigkeit und Umfang von Änderungen
- Beteiligte Entwickler (z. B. Erfahrung, Teamwechsel)

4. Testdaten und Testergebnisse

- Bestehende Testabdeckung
- Häufigkeit von Testfehlschlägen
- Flaky Tests (instabile Tests)

5. Anforderungs- und Spezifikationsdaten

- Änderungen in Anforderungen
- Unklare oder widersprüchliche Anforderungen (z. B. durch NLP erkannt)

6. Build- und CI/CD-Daten

- Build-Erfolgsraten
- Dauer und Häufigkeit von Builds
- Fehler während der Integration

Wie hilft das bei der Testpriorisierung?

1. Risikobasierte Testauswahl

- KI bewertet, **welche Codebereiche wahrscheinlich fehleranfällig** sind.
- Testfälle, die diese Bereiche abdecken, werden **höher priorisiert**.

2. Effizientere Ressourcennutzung

- Tests mit **niedriger Fehlerwahrscheinlichkeit** können seltener oder später ausgeführt werden.
- Spart Zeit und Rechenressourcen.

3. Dynamische Anpassung

- Testprioritäten passen sich **automatisch an neue Codeänderungen** oder Fehlertrends an.

4. Frühzeitige Fehlererkennung

- Kritische Fehler werden **früher im Testprozess** entdeckt, was die Behebungskosten senkt.

Fazit:

Die Kombination aus **datengetriebener Fehlervorhersage** und **intelligenter Testpriorisierung** ermöglicht:

- **schnellere Feedbackzyklen**
 - **höhere Testeffizienz**
 - **bessere Fehlerabdeckung bei weniger Aufwand**
-

H0 11.5 Implementieren eines einfachen KI-basierten Fehlerprognosesystems

Aufgabe:

Nutzen Sie einen Datensatz mit Quellcode -Metriken, um ein einfaches ML-Modell zur Fehlervorhersage zu erstellen und zu evaluieren.

Lösung:

Ich habe ein einfaches Machine-Learning-Modell zur **Fehlervorhersage** basierend auf einem synthetischen Datensatz mit typischen **Quellcode-Metriken** erstellt und evaluiert. Hier sind die wichtigsten Ergebnisse:

Evaluations Metriken

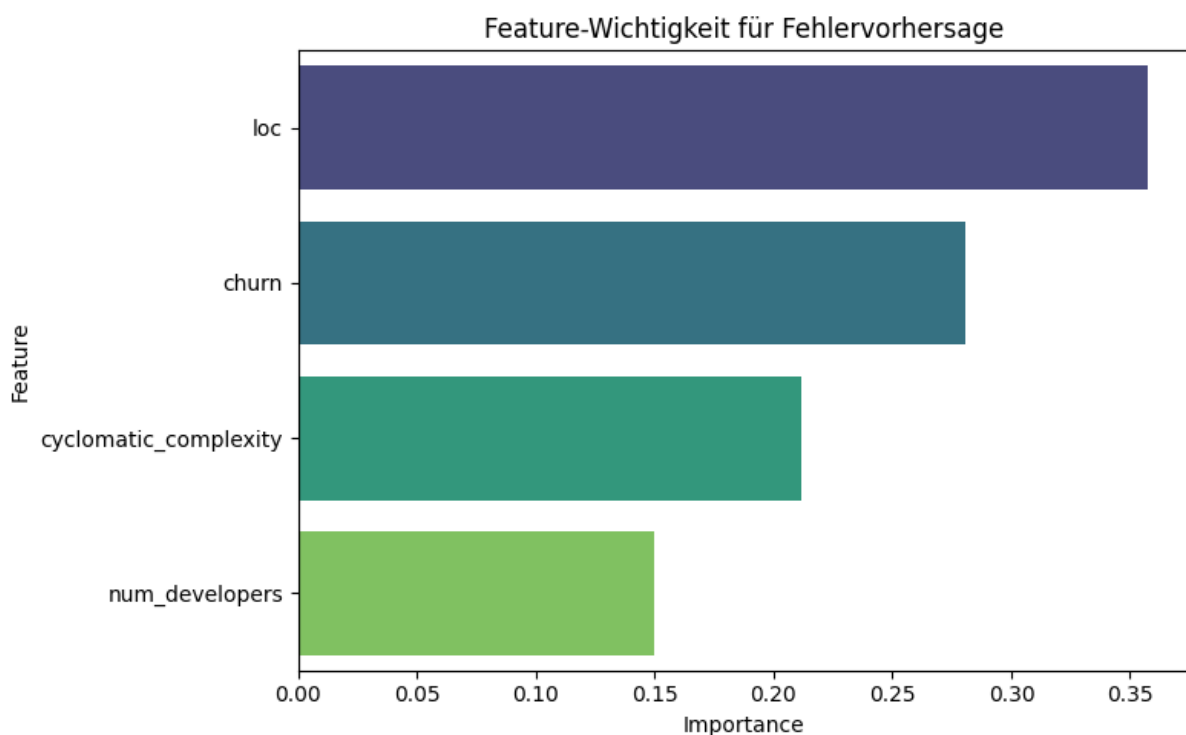
Metrik	Wert
Accuracy	0.67
Precision	0.14
Recall	0.04
F1-Score	0.06

Interpretation:

- Das Modell erkennt fehlerhafte Module **nur sehr eingeschränkt** (Recall = 0.04).
- Die hohe Accuracy ist irreführend, da die meisten Module **keine Fehler enthalten** (Klassenungleichgewicht).
- **Precision und F1-Score sind niedrig**, was auf eine geringe Vorhersagequalität für Fehler hinweist.

Feature-Wichtigkeit

Die folgende Grafik zeigt, welche Metriken für die Fehlervorhersage am wichtigsten war:



Top-Merkmale:

- **Lines of Code (LOC)** und **Cyclomatic Complexity** sind die wichtigsten Prädiktoren.
- **Churn** (Änderungshäufigkeit) und Anzahl der Entwickler haben geringere Bedeutung.

Fazit & Ausblick

- Das Modell zeigt, dass **Quellcode-Metriken prinzipiell zur Fehlervorhersage genutzt werden können**, aber:
 - Es leidet unter **Klassenungleichgewicht** (nur 30 % fehlerhafte Module).
 - Eine bessere Performance könnte durch **Feature Engineering, Balancing-Techniken** (z. B. SMOTE) oder **komplexere Modelle** erreicht werden.

11.6 Erklären des Einsatzes von KI beim Testen von Benutzerschnittstellen

Frage:

Wie kann KI beim Testen grafischer Benutzungsschnittstellen GUI eingesetzt werden und welche Vorteile bietet dies gegenüber klassischen Ansätzen?

Antwort:

KI kann beim Testen grafischer Benutzungsschnittstellen (GUIs) auf vielfältige Weise eingesetzt werden und bietet gegenüber klassischen Ansätzen deutliche Vorteile. Hier ist eine strukturierte Übersicht:

Wie KI beim GUI-Testen eingesetzt wird

1. Visuelle Erkennung und Validierung

- KI-Modelle (z. B. Computer Vision) erkennen GUI-Elemente wie Buttons, Menüs, Eingabefelder.
- Sie vergleichen **Soll- und Ist-Zustände** visuell, z. B. Layout, Farben, Positionen.

2. Automatische Interaktion

- KI kann Benutzeraktionen simulieren: Klicks, Eingaben, Navigation.
- Reinforcement Learning kann genutzt werden, um **optimale Interaktionspfade** zu lernen.

3. Fehlererkennung durch Bildvergleich

- KI erkennt **visuelle Fehler**, z. B. überlappende Elemente, fehlende Icons, falsche Schriftgrößen.
- Deep Learning kann auch **semantische Unterschiede** erkennen, nicht nur Pixelabweichungen.

4. Testfallgenerierung aus GUI-Strukturen

- KI analysiert die GUI-Hierarchie und generiert automatisch Testfälle für verschiedene Zustände und Übergänge.
- NLP kann helfen, aus GUI-Beschreibungen oder User Stories passende Tests abzuleiten.

5. Selbstheilende Tests

- Wenn sich GUI-Elemente ändern (z. B. ID, Position), kann KI die Tests **automatisch anpassen**, statt sie fehlschlagen zu lassen.
 - Erhöht die Robustheit gegenüber UI-Änderungen.
-

Vorteile gegenüber klassischen GUI-Testansätzen

Klassisch	KI-gestützt
Manuelle Identifikation von Elementen	Automatische visuelle Erkennung
Statische Testskripte	Dynamische, adaptive Teststrategien
Hoher Wartungsaufwand	Selbstheilende Tests
Begrenzte Testabdeckung	Breitere und intelligentere Abdeckung
Keine semantische Bewertung	KI erkennt auch inhaltliche Fehler

Typische Einsatzszenarien

- **Web- und Mobile-Apps** mit häufigen UI-Änderungen
 - **Cross-Browser-Tests** (z. B. visuelle Unterschiede zwischen Chrome und Firefox)
 - **Barrierefreiheitstests** (z. B. Kontrast, Lesbarkeit)
 - **Regressionstests** nach UI-Updates
-