

Bachelorarbeit

KI-Integration in Cloud Native Platform Engineering: Eine systematische Analyse aktueller Lösungsansätze und deren praktische Anwendung

von

Nils Arnold

zur Erlangung des akademischen Grades

Bachelor of Science

im Studiengang Wirtschaftsinformatik

an der Hochschule Konstanz Technik, Wirtschaft und Gestaltung und der Robert Bosch GmbH

Matrikelnummer: 307179

Abgabedatum: 31.01.2026

Erstbetreuer: Prof. Dr. Johannes Schneider

Zweitbetreuer: Lukas Grodmeier.(M.Sc.)

Abstract

Abstract. . .

Ehrenwörtliche Erklärung

Hiermit erkläre ich, Nils Arnold, geboren am 18. Januar 2003 in Balingen,

(1) dass ich meine Bachelorarbeit mit dem Titel:

„KI-Integration in Cloud Native Platform Engineering: Eine systematische Analyse aktueller Lösungsansätze und deren praktische Anwendung“

bei der Robert Bosch GmbH unter Anleitung von Prof. Dr. Johannes Schneider und Lukas Grodmeier (M.Sc.) selbstständig und ohne fremde Hilfe angefertigt habe und keine anderen als die angeführten Hilfen benutzt habe;

(2) dass ich die Übernahme wörtlicher Zitate, von Tabellen, Zeichnungen, Bildern und Programmen aus der Literatur oder anderen Quellen (Internet) sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Ort, Datum

Unterschrift

Inhaltsverzeichnis

Abstract	i
Ehrenwörtliche Erklärung	ii
Inhaltsverzeichnis	iii
1 Einleitung	1
1.1 Problemstellung	1
1.2 Zielsetzung der Arbeit	2
1.3 Aufbau der Arbeit	2
2 Grundlagen und verwandte Arbeiten	3
2.1 Grundlagen	3
2.1.1 Cloud Native Technologien und Platform Engineering	3
2.1.2 DevOps und CI/CD	4
2.1.3 Lernparadigmen des maschinellen Lernens	4
2.1.4 AIOps und verwandte Konzepte	5
2.2 Verwandte Arbeiten	5
3 Methodisches Vorgehen	6
3.1 Vorgehen der Mapping Study	6
3.2 Forschungsfragen	7
3.3 Literatur Analyse Prozess	8
3.3.1 Suchstrategie	8
3.3.2 Auswahlkriterien	8
3.3.3 Schneeballmethode	9
3.3.4 Datenerhebung aus den Studien	9
3.3.5 Datenanalyse und Kategorisierung	10
4 Ergebnisse der Literaturanalyse	11
4.1 Quantitative Analyse	11
4.1.1 Anwendungsbereiche der KI im Platform Engineering	12
4.1.2 Herausforderungen der KI-Integration im Platform Engineering	13
4.1.3 Formen des maschinellen Lernens	14
4.1.4 Verwendete Algorithmen	15

4.2	Ergebnisse der Mapping Study	17
4.2.1	Zusammenspiel der Anwendungsfelder und Herausforderungen . .	17
4.2.2	Zusammenspiel der Lernparadigmen und Algorithmen	19
4.2.3	Zusammenspiel von Herausforderungen und KI-Technologien (Tools und Frameworks)	21
4.2.4	Mapping der zentralen Dimensionen.	22
4.3	Matching-Framework	25
4.3.1	Muster 1: Prädiktives Auto-Scaling und Ressourcenoptimierung . .	25
4.3.2	Muster 2: Intelligente CI/CD (Fehlervorhersage und adaptive Roll- backs).	27
4.3.3	Muster 3: KI-gestützte Observability und Ursachenanalyse (AI- Ops).	28
4.3.4	Muster 4: Intelligente Sicherheit und Bedrohungserkennung	30
5	Anwendung der Literaturrecherche	32
5.1	Bewertungskonzept/ Framework	32
5.2	Analyse der Bosch Digital Manufacturing Plattform	32
6	Diskussion	33
6.1	Beantwortung der Forschungsfragen	33
6.2	Praxisrückschluss	33
6.3	Limitationen.	33
6.4	Handlungsempfehlungen	34
7	Zusammenfassung und Ausblick	35
7.1	Zusammenfassung	35
7.2	Ausblick	35
	Tabellenverzeichnis	36
	Abbildungsverzeichnis	37
	Literaturverzeichnis	38

1

Einleitung

Mit der zunehmenden Verbreitung von Cloud-Native-Technologien und Plattform-Engineering Ansätzen, stehen Unternehmen vor der Herausforderung, ihre Software-Entwicklung und den Plattformbetrieb effizient, skalierbar und resilient zu gestalten. Laut der CNF-Studie geben 89 Prozent der befragten Organisation an, Cloud Native-Technologien zu nutzen [1]. Die Komplexität von Infrastrukturen, Plattformen und Entwicklungsumgebung wächst in diesem Umfeld stark. Diese Entwicklung eröffnet zwar enorme Potentiale für Agilität und Innovation, stellt Plattform-Teams aber gleichermaßen vor neue Herausforderungen. Der Wunsch nach Automatisierung durch den Einsatz von Künstlicher Intelligenz wird immer größer.

1.1. Problemstellung

Trotz der flächendeckenden Verbreitung von Cloud-Native Architekturen und dem klaren Fokus vieler Unternehmen auf Automatisierung und Effizienzsteigerung, bleibt die Frage offen, wie Plattform-Teams konkret von fortgeschrittenen Automatisierungs- und KI-gestützten Ansätzen profitieren. Während zahlreiche Unternehmen bereits erste KI-gestützte Tools in ihren Cloud-Native Umgebungen einsetzen, existiert bislang keine systematische und evidenzbasierte Analyse, welche Lösungen tatsächlich Mehrwert für Plattform-Teams schaffen. Die Forschung zu AIOps und KI im Software-Engineering ist zwar umfangreich, doch deren Bezug zu spezifischen Domänen des Plattform Engineerings wie CI/CD-Automatisierung, Infrastruktur-Management und Monitoring bleibt häufig unscharf. Zudem fehlen praxisnahe Untersuchungen, die auf realen Plattform-

Stacks aufbauen und konkrete Pain Points der beteiligten Teams berücksichtigen. Daraus ergibt sich die Notwendigkeit, den aktuellen Stand der Forschung systematisch zu erfassen, mit industriellen Anforderungen abzugleichen und daraus Handlungsempfehlungen für die Integration von KI in bestehenden Plattformlandschaften abzuleiten.

1.2. Zielsetzung der Arbeit

Ziel dieser Arbeit ist es, eine wissenschaftlich fundierte Grundlage für die Integration von Künstlicher Intelligenz in Cloud-Native-Plattform-Engineering-Umgebung zu schaffen. Hierzu wird zunächst im Rahmen einer systematischen Mapping Study untersucht, welche aktuellen KI-Ansätze im Plattform Engineering existieren und wo genau sie Anwendung finden. Darauf aufbauend erfolgt eine Analyse der Bosch Digital Manufacturing Plattform, um bestehende Herausforderungen zu identifizieren und potenzielle Einsatzfelder von KI-Technologien zu evaluieren. Das Ende der Arbeit setzt sich aus der Entwicklung eines praxistauglichen Frameworks zur Bewertung von KI-Potenzialen in Plattformumgebung sowie mit einer prototypischen Implementierung zusammen. Damit leistet die Arbeit sowohl einen wissenschaftlichen als auch einen praktischen Beitrag zur Weiterentwicklung moderner Plattform-Engineering Praktiken.

1.3. Aufbau der Arbeit

Dieser Abschnitt gibt einen Überblick über den strukturellen Aufbau der Arbeit.

2

Grundlagen und verwandte Arbeiten

In diesem Kapitel werden die theoretischen Grundlagen dargestellt, die für das Verständnis der Arbeit notwendig sind, sowie verwandte Arbeiten eingeordnet.

2.1. Grundlagen

2.1.1. Cloud Native Technologien und Platform Engineering

Cloud Native (CN) gewann ab 2013 insbesondere durch die Etablierung von Containertechnologien bis hin zu Kubernetes an Bedeutung. Im Kern beschreibt CN heute weniger eine einzelne Technologie, sondern ein Zielbild für modular aufgebaute Systeme (z. B. Microservices), die sich gut bereitstellen, skalieren und resilient betreiben lassen. Die CNCF fasst Cloud Native als Ansatz für skalierbare Anwendungen in dynamischen Cloud-Umgebungen zusammen [2].

Kubernetes hat sich dabei als zentrale Plattform etabliert. Es dient als portable, erweiterbare Open-Source-Lösung zur Verwaltung containerisierter Workloads und unterstützt insbesondere deklarative Konfiguration und Automatisierung [3].

Im Platform Engineering werden diese Grundlagen genutzt, um Entwicklerteams über eine interne Plattform (Internal Developer Platform) standardisierte, sichere Self-Service-Capabilities bereitzustellen und damit Lieferfähigkeit, Compliance und Betrieb zu verbessern [4].

2.1.2. DevOps und CI/CD

Development und Operations (DevOps) ist ein kultureller und organisatorischer Ansatz, bei dem Entwicklung und Betrieb gemeinsam Verantwortung für den gesamten Software-Lebenszyklus tragen. Ziel ist es, Zusammenarbeit und Kommunikation zwischen den beteiligten Rollen zu stärken und Abläufe so zu unterstützen, dass Änderungen zuverlässig und kontinuierlich bereitgestellt werden können [5, 6].

Continuous Integration (CI) und Continuous Delivery/Deployment (CD) sind zentrale Praktiken innerhalb von DevOps. CI beschreibt die regelmäßige und automatisierte Integration von Codeänderungen in ein gemeinsames Repository inklusive Build und Tests. CD baut darauf auf und automatisiert die Auslieferung. Continuous Delivery endet vor dem automatischen Produktiv-Deployment, während Continuous Deployment diesen Schritt ebenfalls automatisiert [7, 8].

2.1.3. Lernparadigmen des maschinellen Lernens

Die grundlegenden Lernparadigmen des maschinellen Lernens lassen sich in Supervised Learning (SL), Unsupervised Learning (UL) und Reinforcement Learning (RL) einteilen.

SL trainiert ein Modell anhand gelabelter Daten, sodass es eine Abbildung von Eingaben auf eine Ziel- bzw. Antwortvariable lernt [9, 10].

UL arbeitet mit ungelabelten Daten und zielt darauf ab, darin Strukturen, Cluster oder Auffälligkeiten zu erkennen, ohne dass eine explizite Zielvariable vorgegeben ist [9, 11].

RL beschreibt Verfahren, bei denen ein Agent durch Interaktion mit einer Umgebung Handlungen lernt, um eine langfristige (kumulative) Belohnung zu maximieren [9, 12].

Die drei Lernparadigmen bilden die Grundlage, während die konkrete Umsetzung in der Praxis typischerweise über spezifische Algorithmen und Methoden erfolgt. Tabelle 2.1 fasst die in dieser Arbeit betrachteten Algorithmusklassen zusammen und schafft damit eine einheitliche begriffliche Grundlage für die spätere Auswertung.

Tabelle 2.1: Beschreibung Algorithmen und Methoden

Algorithmen und Methoden	Beschreibung
Deep Learning (DL)/ Neuronale Netze (NN)	Erfassen komplexe Muster in Daten und eignen sich besonders für unstrukturierte Eingaben wie Log- oder Monitoring-Daten.
Ensemble und Baum-basiert	Kombinieren mehrere Modelle zur Steigerung der Vorhersagegenauigkeit und verarbeiten große sowie semi-strukturierte Datensätze effizient.
Klassische Klassifikation/ Regression	Traditionelle ML-Modelle zur Vorhersage von Mustern oder Ereignissen auf Basis strukturierter Daten.
Clustering	Gruppieren Datenpunkte ohne Labels in inhaltlich ähnliche Cluster und unterstützen dadurch Mustererkennung und Anomaliedetektion.

In der Analyse wurden Lernparadigmen zudem auch dann zugeordnet, wenn sie in den Publikationen nicht explizit benannt waren, sondern nur über typische Aufgabenstellungen erkennbar wurden. Solche Aufgabenstellungen werden in den in diesem Abschnitt zitierten Quellen ausführlicher beschrieben.

2.1.4. AIOps und verwandte Konzepte

Definition und Abgrenzung von AIOps, Monitoring, Observability und autonomen Betriebsansätzen.

2.2. Verwandte Arbeiten

Relevante wissenschaftliche Arbeiten, Industriereports und State-of-the-Art-Übersichten werden beschrieben und kritisch eingeordnet.

3

Methodisches Vorgehen

Dieses Kapitel beschreibt das methodische Vorgehen dieser Arbeit. Es erläutert die zugrunde liegende Forschungslogik, die Auswahl des methodischen Ansatzes sowie die Verfahren zur Datenerhebung und -auswertung. Ziel ist es, ein wissenschaftlich fundiertes und zugleich praxisorientiertes Vorgehen aufzuzeigen, das eine systematische Untersuchung der Forschungsfragen ermöglicht. In Abschnitt 3.1 wird eine systematische Mapping Study (SMS) nach Petersen et al [13] angewendet. Abschnitt 3.2 definiert die spezifischen Ziele und konkreten Forschungsfragen dieser Arbeit, welche als Basis für die anschließende Analyse dienen. Darauf aufbauend beschreibt Abschnitt 3.3 den Prozess der Literaturanalyse. Dieser umfasst die Entwicklung einer Suchstrategie, die Definition von Ein- und Ausschlusskriterien, die Schneeballmethode sowie die Schritte der Datenextraktion und -synthese.

3.1. Vorgehen der Mapping Study

Für diese Arbeit wird eine Systematic Mapping Study (SMS) nach den Richtlinien von Petersen, Vakkalanka und Kuzniarz [13] durchgeführt. Diese Methodik dient dazu, den aktuellen Forschungsstand zu einem Themengebiet systematisch zu erfassen, zu kategorisieren und bestehende Forschungslücken zu identifizieren.

Das Vorgehen umfasst die Phasen Planung, Durchführung und Auswertung. In der Planungsphase werden die Forschungsfragen definiert und die Suchstrategie entwickelt, einschließlich der Auswahl relevanter wissenschaftlicher Datenbanken. Dabei wird gezielt nach bestimmten Keywords gesucht, um Publikationen zu finden, die KI-Anwendungen

im Kontext von Platform Engineering adressieren.

Um eine fundierte Datenerhebung und Auswertung sicherzustellen, werden einzelne Prinzipien einer Systematic Literature Review (SLR) nach Kitchenham und Charters [14] berücksichtigt. In der Durchführungsphase werden identifizierte Studien anhand festgelegter Ein- und Ausschlusskriterien geprüft. Zusätzlich wird das Schneeballverfahren nach Wohlin [15] eingesetzt, um die Literatursammlung zu erweitern. Alle Schritte werden dokumentiert, um die Nachvollziehbarkeit und Reproduzierbarkeit sicherzustellen. Die Auswertung erfolgt durch eine systematische Kategorisierung der Studien entlang zentraler Themenfelder des Platform Engineerings. Darauf aufbauend werden Muster, Trends und Forschungslücken identifiziert.

Die Ergebnisse der Mapping Study bilden die Grundlage für die anschließende Analyse der Bosch Digital Manufacturing Platform sowie für die Entwicklung eines Frameworks zur Bewertung von KI-Potenzialen in Cloud-Native Plattformumgebungen.

3.2. Forschungsfragen

Die Forschungsfragen werden durch ein strukturiertes methodisches Vorgehen beantwortet. Jede Forschungsphase ist darauf ausgelegt, das Verständnis über den Einsatz von Künstlicher Intelligenz im Platform Engineering schrittweise zu vertiefen. Die Mapping Study dient der Beantwortung der ersten beiden Forschungsfragen, indem sie eine systematische Übersicht über bestehende KI-Ansätze und deren Anwendungsfelder liefert. Auf Grundlage der Ergebnisse der Mapping Study zielt die dritte Forschungsfrage darauf ab, ein praxisorientiertes Framework zur Beantwortung und Übertragbarkeit von KI-Lösungen zu entwickeln. Die Arbeit ist entlang der folgenden Forschungsfragen strukturiert:

- RQ1:** Welche typischen Anwendungsfelder (Use Cases) und Herausforderungen bestehen im Platform Engineering, in denen KI-Technologien potenziell Mehrwert bieten können?
Ziel: Systematische Erfassung und Kategorisierung relevanter Use Cases.
- RQ2:** Welche KI-Technologien (einschließlich Frameworks und Tools) finden derzeit im Platform Engineering Anwendung, und welche Formen des maschinellen Lernens und Algorithmen kommen dabei zum Einsatz?
Ziel: Erstellung einer Übersicht über vorhandene KI-Ansätze und deren typische Einsatzkontexte.

RQ3: Wie lassen sich die identifizierten KI-Lösungen auf typische Anwendungsfälle in Cloud-Native-Platform-Umgebungen übertragen und hinsichtlich ihres Mehrwertes und ihrer Umsetzbarkeit bewerten?

Ziel: Entwicklung eines Bewertungsschemas, das die Passung zwischen KI-Lösungen und spezifischen Platform-Use-Cases beschreibt und die praktische Umsetzbarkeit aufzeigt.

3.3. Literatur Analyse Prozess

Der folgende Abschnitt beschreibt den Ablauf der Literaturanalyse im Rahmen der durchgeführten Mapping Study. Ziel ist es, den methodischen Prozess transparent darzustellen. Dazu werden zunächst die Suchstrategie und die Auswahlkriterien erläutert, gefolgt von der Anwendung der Schneeballmethode. Danach wird die Datenextraktion sowie die Datensynthese beschrieben.

3.3.1. Suchstrategie

Zur Durchführung der Mapping Study wurde eine systematische Suchstrategie angewendet, um relevante wissenschaftliche Publikationen zu identifizieren. Die Literaturrecherche erfolgte in den Datenbanken Google Scholar, SpringerLink, ScienceDirect und IEEE Xplore, da diese eine breite Abdeckung im Bereich Software Engineering, Cloud Native Technologien und KI bieten. Ziel der Suche war, eine möglichst vollständige Übersicht aktueller Forschungsarbeiten zu KI-Anwendungen im Platform Engineering zu erhalten. Dafür wurden gezielt Suchbegriffe und Kombinationen von Suchstrings verwendet, die zentrale Themen der Arbeit abbilden. Die Suchbegriffe waren hierbei: "Platform Engineering", "Cloud-Native", "ÄIOps", "Ärtificial Intelligence", "Machine Learning", "MLOps", "DevOps" und "Kubernetes Cluster".

Zur Transparenz und Vollständigkeit sind die exakten Suchstrings sowie ihre logischen Verknüpfungen im Anhang dokumentiert.

3.3.2. Auswahlkriterien

Um relevante Studien und Publikationen zu identifizieren, wurde ein systematischer Auswahlprozess durchgeführt, der auf klar definierten Ein- und Ausschlusskriterien basiert. Eine Studie wurde in die Analyse aufgenommen, wenn sie alle Einschlusskriterien

erfüllt und zugleich keinem der Ausschlusskriterien unterlag. Die vollständige Übersicht der Kriterien ist in Tabelle 3.1 dargestellt.

Tabelle 3.1: Auswahlkriterien

Kriterium	Beschreibung
EK1	Die Publikation befasst sich mit dem Einsatz von KI oder Machine Learning im Kontext von Platform Engineering, Cloud-Native-Technologien, DevOps oder AIOps.
EK2	Die Studie beschreibt konkrete KI-Methoden, Anwendungen, Architekturen oder Use Cases, die sich auf Plattformumgebungen beziehen.
EK3	Die Arbeit ist wissenschaftlich fundiert, z.B. als Konferenz-, Journal- oder White Paper, auch wenn kein Peer-Review-Verfahren vorliegt.
EK4	Veröffentlichungen sind in englischer oder deutscher Sprache verfasst und nach 2020 erschienen.
AK1	Arbeiten, die keinen direkten Bezug zu KI im Platform Engineering oder verwandten Domänen aufweisen.
AK2	Review-Paper oder systematische Übersichtsarbeiten, die keine eigenen empirischen oder technischen Beiträge enthalten.
AK3	Studien ohne nachvollziehbare methodische Grundlage oder ohne Beschreibung der verwendeten KI-Techniken.

3.3.3. Schneeballmethode

Zur Ergänzung der systematischen Suche wurde eine Schneeballmethode nach den Leitlinien von Wohlin [15] angewendet. Dabei erfolgte sowohl eine Rückwärtssuche als auch eine Vorwärtssuche. Als Ausgangspunkt dienten vier relevante Paper, auf deren Basis zwei Iterationen der Vorwärts- und Rückwärtssuche durchgeführt wurden. Die neu gefundenen Publikationen wurden nach denselben Ein- und Ausschlusskriterien geprüft. Der Prozess wurde beendet, sobald keine weiteren relevanten Studien identifiziert werden konnten.

3.3.4. Datenerhebung aus den Studien

Zur Sicherstellung von Konsistenz und Nachvollziehbarkeit wurde ein strukturierter Prozess zur Datenerhebung aus den Studien umgesetzt. Hierzu wurde eine eigene Extraktionsvorlage entwickelt, die die wesentlichen Merkmale der identifizierten Studien erfasst. Diese Merkmale wurden anschließend entlang von fünf zentralen Dimensionen kategorisiert, wie in Tabelle 3.2 dargestellt.

Tabelle 3.2: Datenextraktion

Dimension	Beschreibung
Forschungskontext	Beschreibt Ziel, Umfang und Art der Studie.
KI-Ansatz und Methode	Erfasst die verwendeten KI- oder ML-Verfahren.
Plattform-Domänen	Ordnet den Beitrag einem Bereich des Platform Engineerings zu.
Ergebnisse und Use-Cases	Fasst die zentralen Erkenntnisse, Anwendungsfälle oder Evaluationsergebnisse zusammen.
Forschungslücke	Dokumentiert identifizierte Limitationen und Ansätze für zukünftige Arbeiten.

Die Extraktion erfolgte qualitativ, wobei relevante Textpassagen und zentrale Aussagen aus jeder Publikation manuell erfasst und den entsprechenden Dimensionen zugeordnet wurden

3.3.5. Datenanalyse und Kategorisierung

Nach der Datenerhebung wurden die Ergebnisse zusammengeführt und ausgewertet, um zentrale Themen, Muster und Forschungslücken zu erkennen. Die ausgewählten Studien wurden nach ihren Inhalten und Schwerpunkten strukturiert und den Forschungsfragen zugeordnet. Zur besseren Übersicht erfolgte die Kategorisierung der Arbeit entlang wichtiger Bereiche des Platform Engineerings. Innerhalb dieser Kategorien wurden die identifizierten KI-Ansätze, Anwendungsfälle und Herausforderungen miteinander verglichen, um wiederkehrende Trends sichtbar zu machen. Die Ergebnisse der Datenanalyse und Kategorisierung werden anschließend in Form einer Mapping Study dargestellt. Diese Übersicht zeigt, in welchen Themenfelder bereits Forschungsschwerpunkte existieren und wo noch Forschungslücken bestehen.

4

Ergebnisse der Literaturanalyse

Dieses Kapitel präsentiert die Ergebnisse der im methodischen Vorgehen beschriebenen Literaturuntersuchung. Zunächst werden in Abschnitt 4.1 die quantitativ erhobenen Merkmale der ausgewählten Publikationen ausgewertet. Abschnitt 4.2 stellt anschließend die Ergebnisse der eigentlichen Mapping Study vor, indem die Arbeiten systematisch klassifiziert und Muster sichtbar gemacht werden. Darauf aufbauend fasst Abschnitt 4.3 die identifizierten KI-Anwendungen im Plattform-Engineering zusammen und leitet ein erstes strukturiertes Matching in Form eines konzeptionellen Frameworks ab. Die dargestellten Ergebnisse bilden die Grundlage für die Beantwortung der Forschungsfragen in Kapitel 6.

4.1. Quantitative Analyse

Zur Einordnung des untersuchten Forschungsfeldes wurden zunächst grundlegende Merkmale der insgesamt 18 Studien analysiert. Abbildung 4.1 a) zeigt die jährliche Verteilung der Publikationen sowie die Zuordnung zu verschiedenen Publikationstypen. Zwischen 2021 und 2023 erscheinen nur wenige Arbeiten (insgesamt drei), während ab 2024 ein deutlicher Anstieg sichtbar wird. Im Jahr 2024 wurden sieben und 2025 wurden acht Publikationen identifiziert, überwiegend Journalartikel, ergänzt durch jeweils ein Konferenzbeitrag, ArXiv-Paper und Whitepaper. Dies weist auf ein zunehmendes wissenschaftliches Interesse am Einsatz von KI in Cloud-Native- und Plattform-Engineering-Kontexten hin. Das lässt sich auch unter anderem auf Grund des ChatGPT/GenAI Hype in den letzten beiden Jahren erklären [16].

Abbildung 4.2 zeigt diese Verteilung. Zu beachten ist, dass viele Studien mehr als einen Bereich ansprechen. Für die Vergleichbarkeit wurde jedoch jeweils der dominante Use Case pro Publikation ausgewählt. In der Praxis überschneiden sich die Anwendungsbereiche häufig, da KI-Lösungen oft mehrere Aufgaben gleichzeitig unterstützen.

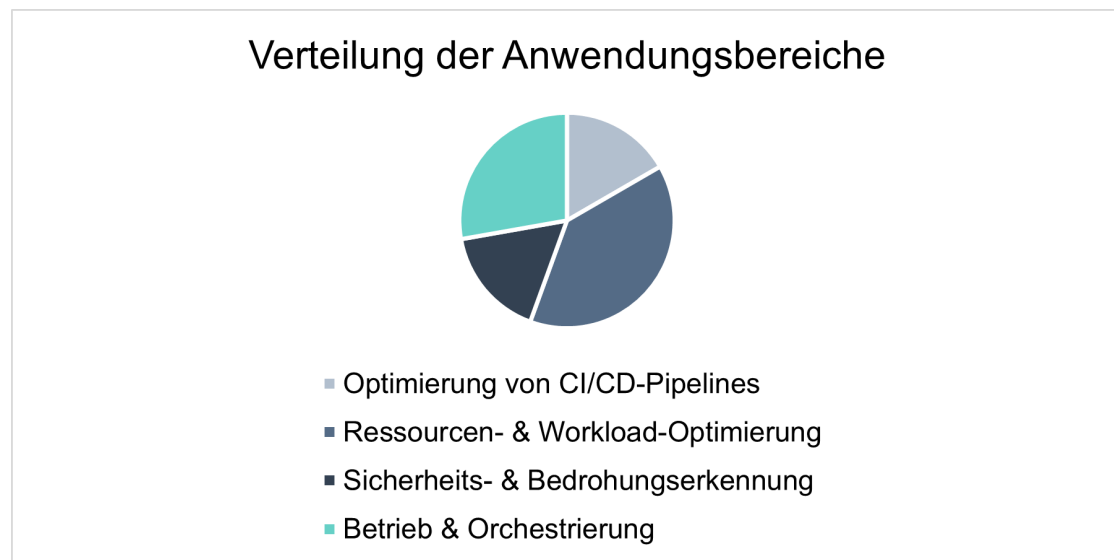


Abbildung 4.2: Verteilung der Anwendungsbereiche

4.1.2. Herausforderungen der KI-Integration im Plattform Engineering

Im nächsten Schritt wurden die in den Publikationen beschriebenen Herausforderungen analysiert, die beim Einsatz von KI im Plattform Engineering auftreten. Die analysierten Paper wurden fünf Kategorien zugeordnet. Die prozentuale ist in Abbildung 4.3 dargestellt.

Die Auswertung zeigt, dass Ressourcenverbrauch und Kosten mit 94 % (17/18) am häufigsten als Herausforderung genannt werden. Ebenfalls häufig werden Skalierbarkeit, Latenz und Monitoring mit 83% (15/18) sowie KI-Governance, Datenschutz und Compliance mit 83% (15/18) adressiert. Integrationskomplexität und Abhängigkeiten werden in 78% (15/18) der Studien thematisiert. Die fünfte Kategorie, Datenqualität, Datenverfügbarkeit und Heterogenität, wird in 72% (13/18) der Arbeiten als Herausforderung genannt.

Diese Ergebnisse deuten auf ein Spannungsfeld hin: KI wird zwar eingesetzt, um Plattformen effizienter und kostengünstiger zu betreiben, erzeugt jedoch durch Training, Inferenz und zusätzliche Observability- und Datenpipelines oft selbst signifikante Ressourcen- und Betriebskosten [2]. Die hohe Nennung von KI-Governance, Datenschutz und Compliance legt zudem nahe, dass der produktive KI-Einsatz in Plattformen häufig weniger

an der reinen technischen Machbarkeit scheitert, sondern stark durch Anforderungen an Datenzugriff, Nachvollziehbarkeit und Regelkonformität begrenzt wird [20].

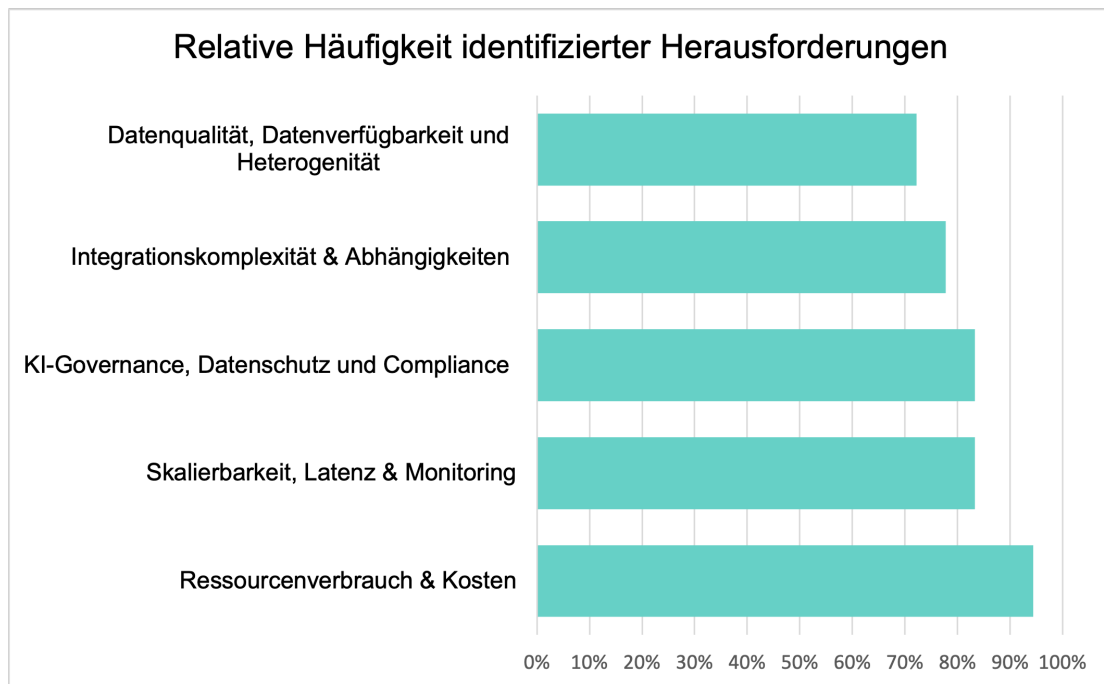


Abbildung 4.3: Relative Häufigkeit der Herausforderungen

4.1.3. Formen des maschinellen Lernens

Ein weiterer Bestandteil der quantitativen Analyse umfasst die in den Publikationen verwendeten Formen bzw. Lernparadigmen des maschinellen Lernens. Dabei wurde unterschieden zwischen (1) explizit benannt, (2) implizit anhand der beschriebenen Methode ableitbar und (3) nur kurz erwähnt ohne weitere methodische Ausführung. Insgesamt wurden die drei grundlegenden Paradigmen identifiziert: Supervised Learning, Unsupervised Learning und Reinforcement Learning.

Die Auswertung zeigt, dass Supervised Learning in den meisten Arbeiten eine zentrale Rolle spielt. Es wird in vier Publikationen ausdrücklich beschrieben, elfmal implizit erkennbar und in zwei kurz erwähnt. Reinforcement Learning wird insgesamt siebenmal explizit genannt und in vier weiteren Arbeiten erwähnt. Unsupervised Learning tritt mit drei expliziten und sieben impliziten Nennungen seltener auf, ist jedoch ebenfalls präsent.

Die Verteilung ist in der folgenden Abbildung 4.4 dargestellt.

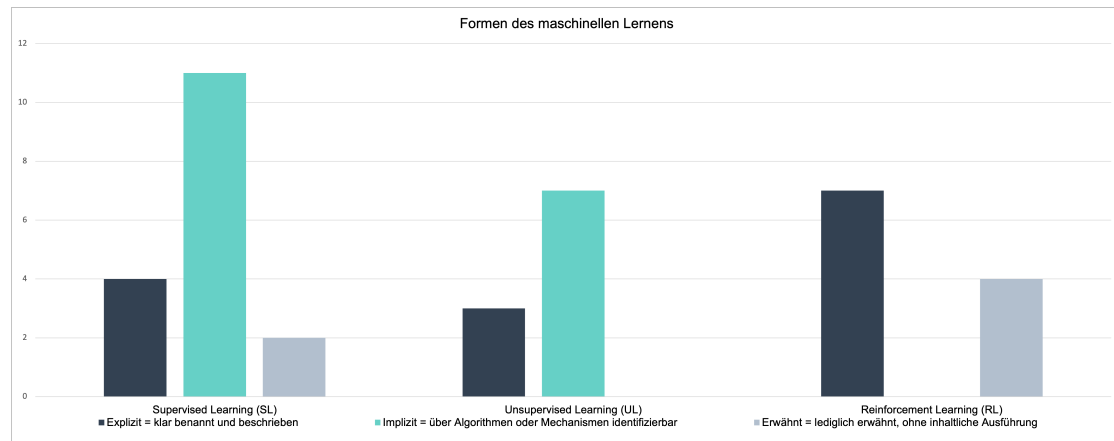


Abbildung 4.4: Formen des maschinellen Lernens

Auffällig ist die deutliche Differenz zwischen explizit und implizit erkennbaren Anwendungen, insbesondere bei Supervised und Unsupervised Learning. Dies zeigt, dass viele Publikationen die entsprechenden Paradigmen beschreiben, ohne die zugrunde liegende Lernform ausdrücklich zu benennen. Ein möglicher Erklärungsansatz für die Dominanz von Supervised Learning ist, dass in Plattformumgebungen häufig gut messbare Zielgrößen und bereits beschriftete Verlaufsdaten vorliegen (z.,B. Störungsmeldungen, Support-Tickets oder Metriken mit bekanntem Ergebnis), wodurch sich SL-Ansätze besonders gut anwenden und bewerten lassen [20].

4.1.4. Verwendete Algorithmen

Die in den Publikationen verwendeten Algorithmen lassen sich den jeweiligen Lernparadigmen zuordnen. Die Analyse zeigt, dass eine Vielfalt an KI- und ML-Verfahren im Platform-Engineering-Kontext eingesetzt wird. Für eine systematische Bewertung wurde eine eigene Kategorisierung entwickelt. Den Ausgangspunkt bildet die Tabelle aus Enemosah [9], in der verschiedene KI-Techniken für Testfallpriorisierung beschrieben werden. Die Einteilung wurde für den breiteren Kontext dieser Arbeit angepasst.

Auf dieser Basis wurden vier Kategorien definiert, die in Kapitel 2.1 kurz beschrieben sind. Anschließend folgt die quantitative Auswertung der identifizierten Methoden und Algorithmen. Dies zeigt, wie häufig die jeweiligen Verfahren in den Publikationen genannt werden.

Die Ergebnisse zeigen ein deutliches Übergewicht von Deep Learning / neuronalen Netzen, die in 89% (16/18) der Publikationen eingesetzt oder thematisiert werden. Klassische Klassifikation/Regression tritt mit 44% (8/18) ebenfalls häufig auf. Ensemble- und baumbasierte Verfahren werden in 39% (7/18) der Arbeiten genannt. Clustering

ist mit 33% (6/18) vertreten. Insgesamt zeigt sich, dass insbesondere neuronale Netze die dominierende Methodenklasse bilden. Die Dominanz von Deep-Learning-Verfahren kann darauf hindeuten, dass in der Forschung häufig komplexere Modelle bevorzugt werden, weil sie mit heterogenen Daten (z.,B. Logs, Metriken, Text) flexibel umgehen können. Gleichzeitig deutet sie auch auf ein Spannungsfeld hin: Für einige Aufgaben könnten einfachere und besser erklärbare Modelle bereits ausreichen, werden in der Literatur aber seltener als Schwerpunkt gesetzt [9, 20].

Diese Verteilung ist in der Abbildung 4.5 dargestellt.

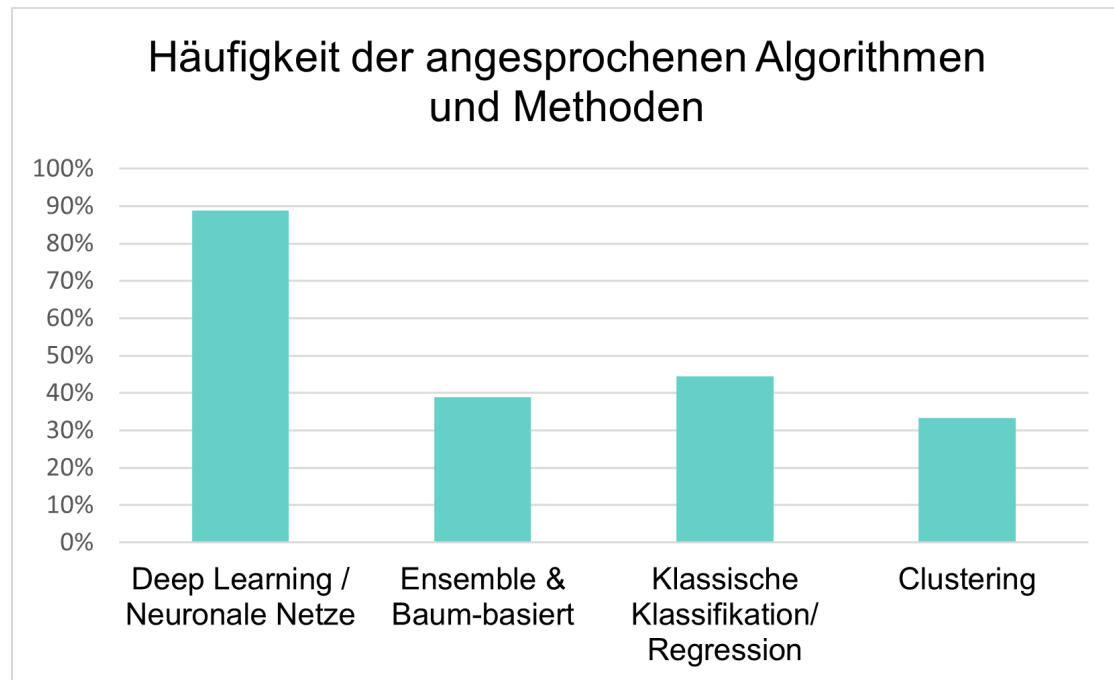


Abbildung 4.5: Verteilung der Algorithmen und angesprochenen Methoden

Die Ergebnisse aus Abschnitt 4.1 zeigen die zentralen Schwerpunkte der betrachteten Studien. Darauf aufbauend wird in Abschnitt 4.2 untersucht, welche Kombinationen zwischen den Kategorien besonders häufig gemeinsam auftreten. Dadurch werden typische Muster innerhalb der Literatur deutlicher sichtbar.

4.2. Ergebnisse der Mapping Study

Um die Beziehungen innerhalb der KI-Forschung im Platform Engineering weiter zu verdeutlichen, baut dieser Abschnitt auf den vorherigen Erkenntnissen auf und präsentiert detaillierte Kreuztabellenanalysen. Diese Zuordnungen untersuchen Beziehungen zwischen Anwendungsfeldern und Herausforderungen (Abschnitt 4.2.1), Lernparadigmen und Algorithmen (Abschnitt 4.2.2), Herausforderungen und KI-Technologien (Abschnitt 4.2.3) sowie Lernparadigmen, Algorithmen und Anwendungsfeldern (Abschnitt 4.2.4). Die Ergebnisse werden als Bubble-Chart-Diagramme visualisiert, in denen die Blasengröße die Häufigkeit der jeweiligen Zuordnung (n) widerspiegelt. Die Häufigkeiten (n) geben an, in wie vielen der betrachteten Studien die jeweilige Zuordnung vorkommt. Pro Studie wird eine Zuordnung je Kombination höchstens einmal gezählt, unabhängig davon, wie häufig sie im Text erwähnt wird. Ziel ist es, durch diese systematischen Mappings tiefere Einblicke in Struktur und Schwerpunkte der aktuellen Forschung zu gewinnen.

4.2.1. Zusammenspiel der Anwendungsfelder und Herausforderungen

Die Abbildung 4.6 zeigt das Zusammenspiel zwischen den identifizierten Use Cases und den zentralen Herausforderungen im Platform Engineering. Im Gegensatz zur Abbildung 4.2 wurden hier alle in den analysierten Studien genannten Anwendungsbe-
reiche berücksichtigt und den jeweils adressierten Herausforderungen zugeordnet. Die Häufigkeiten geben an, wie oft eine bestimmte Kombination in der Literatur thematisiert wurde.

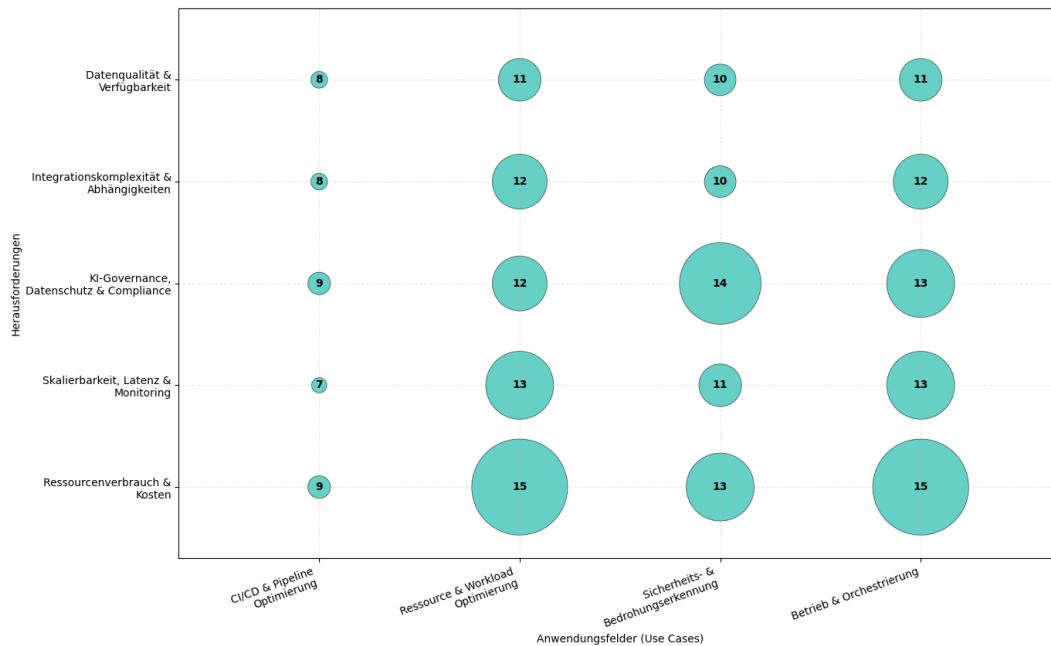


Abbildung 4.6: Korrelation zwischen Anwendungsfelder (Use Cases) und Herausforderungen

Besonders deutlich wird die starke Verknüpfung der Use Cases Ressource- und Workload-Optimierung sowie Betrieb und Orchestrierung mit nahezu allen Herausforderungskategorien. Die Ressourcen- und Workload-Optimierung weist über alle Bereiche hinweg hohe Werte auf ($n=15, 13, 12, 12, 11$). Diese breite Korrelation ist plausibel, da Ressourcen- und Workload-Optimierung einen besonders direkten Bezug zu Kosten hat und sich Effekte häufig über die Cloud-Abrechnung unmittelbar sichtbar machen. KI-gesteuertes Auto-Scaling vermeidet die Überbereitstellung von Ressourcen und senkt dadurch die monatlichen Abrechnungskosten direkt [21]. Gleichzeitig wird deutlich, dass Echtzeit-Inferenz zwar Genauigkeit und Latenz verbessert, jedoch durch höheren CPU-/Memory-Verbrauch die Ressourcen- und Kostenbelastung erhöht und damit eine Abwägung zwischen Modellqualität und Betriebsaufwand erforderlich macht [22].

Ein sehr ähnliches Muster zeigt sich im Bereich Betrieb & Orchestrierung, der ebenfalls in allen Herausforderungskategorien hohe Häufigkeiten erreicht ($n=15, 13, 13, 12, 11$). Das deutet darauf hin, dass KI im Plattformbetrieb häufig nicht als Einzellösung betrachtet wird. Stattdessen ist sie meist in mehrere Betriebsbausteine eingebettet, z.B. Monitoring, Deployment, Orchestrierung und Governance. Damit werden in diesem Bereich oft mehrere Herausforderungen gleichzeitig berührt, etwa Kosten, Skalierung, Integration und datenbezogene Voraussetzungen.

Der Use Case Sicherheits- und Bedrohungserkennung zeigt seine höchste Ausprägung im Bereich KI-Governance, Datenschutz und Compliance ($n=14$) sowie bei Ressour-

cenverbrauch und Kosten (n=13). Auch die übrigen Herausforderungen weisen weiterhin hohe Werte auf (n=11, 10, 10). Dies deutet darauf hin, dass sicherheitsbezogene KI-Ansätze neben Governance-Themen häufig auch Monitoring, Integrationsprozesse und die zugrunde liegende Datenlage betreffen.

Die CI/CD- & Pipeline-Optimierung zeigt insgesamt die niedrigsten Werte, bleibt aber über alle Herausforderungen hinweg relativ konstant (n=9, 7, 9, 8, 8). KI wird hier vor allem eingesetzt, um einzelne Schritte wie Builds, Tests oder Deployments zu verbessern. Im Vergleich zu ressourcen- oder betriebsnahen Use Cases werden jedoch weniger Herausforderungen gleichzeitig berührt. Kosten, Tool-Abhängigkeiten und Governance-Fragen bleiben trotzdem relevant, etwa durch zusätzlichen Rechenaufwand und die notwendige Nachvollziehbarkeit automatisierter Entscheidungen [20].

Insgesamt verdeutlicht die Verteilung der Häufigkeiten, dass KI-Anwendungen im Platform Engineering besonders dort adressiert werden, wo betriebliche Effizienz und Automatisierung direkt mit Kosten, Skalierung, Governance und Integrationsfragen zusammenwirken. Die Häufungen zeigen somit, welche Themen in der Forschung besonders im Fokus stehen und in welchen Bereichen KI-Lösungen aktuell besonders häufig diskutiert werden.

4.2.2. Zusammenspiel der Lernparadigmen und Algorithmen

Die Abbildung 4.7 visualisiert die Korrelation zwischen den in den analysierten Studien verwendeten Lernparadigmen des maschinellen Lernens und den eingesetzten Algorithmen. Einige Kombinationen werden in der Darstellung nicht ausgewiesen (n=0 bzw. ohne Blase), da sie in den betrachteten Studien nicht eindeutig als eigenständige Zuordnung berichtet wurden.

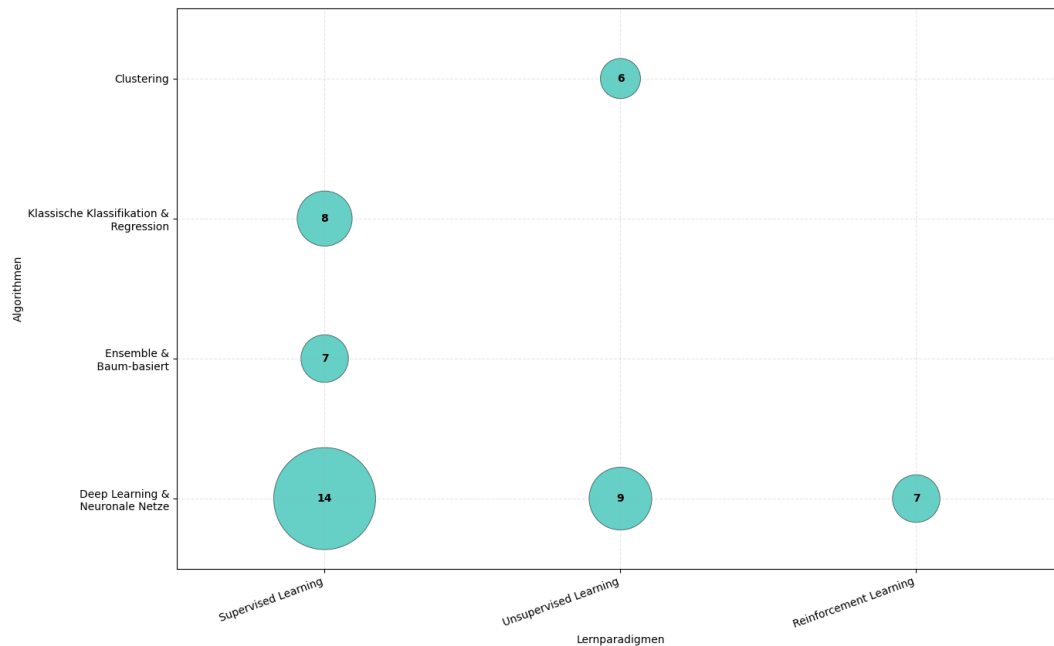


Abbildung 4.7: Korrelation zwischen Lernparadigmen und Algorithmen

Supervised Learning weist die größte Bandbreite an möglichen Algorithmen auf. Dies liegt daran, dass überwachte Lernverfahren sowohl tiefen neuronalen Netzen als auch klassischen Klassifikations- und Regressionsmodellen sowie ensemblebasierten Ansätzen zugrunde liegen. Der hohe Anteil an SL-Kombinationen ($n=14$ DL/NN, $n=8$ Klassifikation/Regression, $n=7$ Ensemble/baum-basiert) zeigt, dass SL sehr flexibel einsetzbar ist. Solange gelabelte Daten vorliegen, können verschiedene Algorithmen angewendet werden.

Unsupervised Learning zeigt hingegen ein engeres Spektrum. Die Ergebnisse verdeutlichen, dass UL hauptsächlich in Kombination mit Deep Learning eingesetzt wird ($n=9$) oder mit Clustering-Methoden ($n=6$). Dies ist erwartungsgemäß, da UL in den untersuchten Studien vor allem zur Mustererkennung und Anomalieanalyse eingesetzt wird und Clustering hierbei eine zentrale Rolle einnimmt, wie in [9] beschrieben.

Reinforcement Learning tritt zwar in mehreren Studien auf, wird jedoch selten algorithmisch konkretisiert. RL wird vor allem für dynamische Optimierungsaufgaben eingesetzt. Durch das Lernen auf Basis beobachteter Systemzustände und Feedback können RL-Modelle geeignete Aktionen auswählen. Dadurch können sie zur Stabilisierung des Betriebs und zur Verbesserung der Systemleistung beitragen. Einzelne Arbeiten nutzen hierfür Deep-RL-Ansätze wie Deep Q-Networks [23], insbesondere wenn komplexe Zustandsräume berücksichtigt werden müssen. Insgesamt bleibt die Zuordnung zu spezifischen Algorithmen jedoch begrenzt, weshalb viele Kombinationen in der Tabelle nicht belegt sind.

4.2.3. Zusammenspiel von Herausforderungen und KI-Technologien (Tools und Frameworks)

Im Folgenden wird das Zusammenspiel von Herausforderungen und KI-Technologien (Tools & Frameworks) anhand der Abbildung 4.8 analysiert. Die Werte geben an, in wie vielen Publikationen (n) eine bestimmte Herausforderung gemeinsam mit einer Tool- bzw. Technologiekategorie adressiert wird.

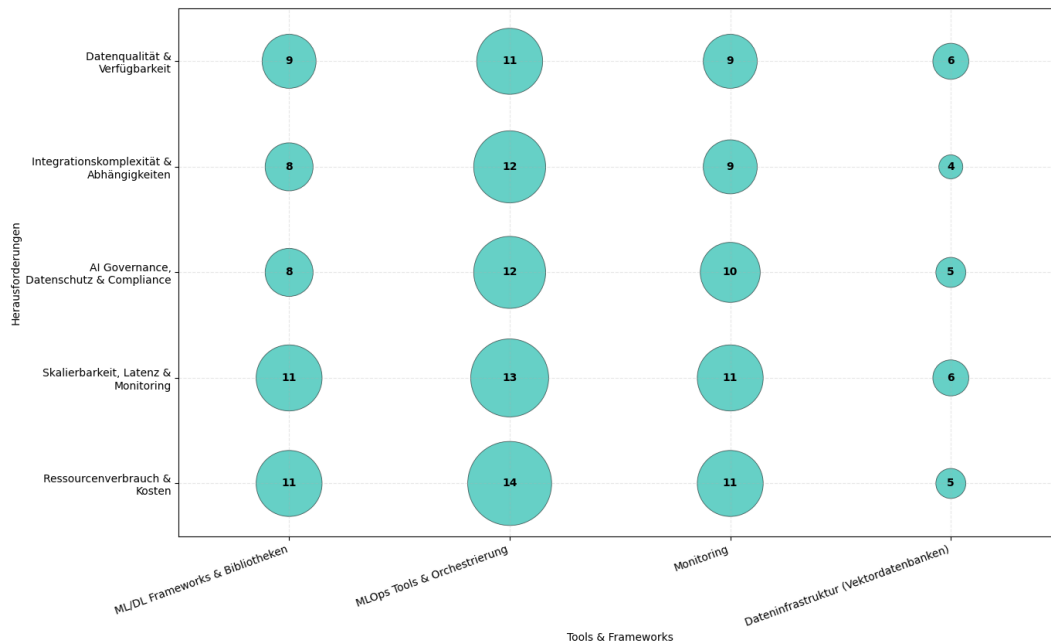


Abbildung 4.8: Korrelation zwischen Herausforderungen und KI-Technologie

Deutlich wird, dass MLOps-Tools & Orchestrierung über alle Herausforderungen hinweg am stärksten vertreten ist (zwischen $n=11$ und $n=14$). Damit zeigt sich, dass MLOps-nahe Technologien in den betrachteten Publikationen besonders häufig im Zusammenhang mit den Herausforderungen adressiert werden.

Für Ressourcenverbrauch und Kosten sowie Skalierbarkeit, Latenz und Monitoring ergeben sich die höchsten Gesamtwerte über alle Tool-Kategorien hinweg. In beiden Fällen treten ML/DL Frameworks & Bibliotheken sowie Monitoring vergleichbar häufig auf (jeweils $n=11$). Das spricht dafür, dass diese Herausforderungen häufig zusammen mit Entwicklungs- und Betriebsaspekten diskutiert werden.

Bei KI-Governance, Datenschutz und Compliance zeigt sich ein ähnlicher Schwerpunkt. MLOps-Tools & Orchestrierung werden hier am häufigsten zusammen mit dieser Herausforderung genannt ($n=12$), gefolgt von Monitoring ($n=10$). ML/DL Frameworks & Bibliotheken treten etwas seltener auf ($n=8$). Insgesamt werden Governance- und

Compliance-Themen damit eher auf Betriebs- als nur auf Modell- oder Trainingsebene verortet.

Integrationskomplexität & Abhängigkeiten wird zwar häufig als Herausforderung genannt (vgl. 4.1.2), jedoch seltener explizit mit konkreten Toolkategorien in Verbindung gebracht. Auch hier liegt der Schwerpunkt bei MLOps-Tools & Orchestrierung (n=12) und Monitoring (n=9). ML/DL Frameworks & Bibliotheken (n=8) und Vektordatenbanken (n=4) treten seltener auf. Das verweist darauf, dass Integrations- und Abhängigkeitsfragen vor allem auf System- und Pipeline-Ebene diskutiert werden.

Für Datenqualität, Datenverfügbarkeit und Heterogenität liegen die Werte im Mittelfeld. MLOps wird am häufigsten gemeinsam adressiert (n=11), während ML/DL Frameworks und Monitoring jeweils bei n=9 liegen. Vektordatenbanken sind hier vergleichsweise stärker vertreten (n=6).

Zusammenfassend ist MLOps die am häufigsten vertretene Tool-Kategorie über alle Herausforderungen hinweg. Vektordatenbanken treten insgesamt seltener auf, sind jedoch insbesondere in datenbezogenen Kontexten sichtbar.

4.2.4. Mapping der zentralen Dimensionen

In diesem Abschnitt wird das Zusammenspiel von Lernparadigmen, Algorithmen und den identifizierten Anwendungsfeldern betrachtet. In der Abbildung 4.9 wurden alle Kategorien miteinander dargestellt. Kombinationen ohne Nachweis in den betrachteten Studien werden als n=0 dargestellt (ohne Blase).

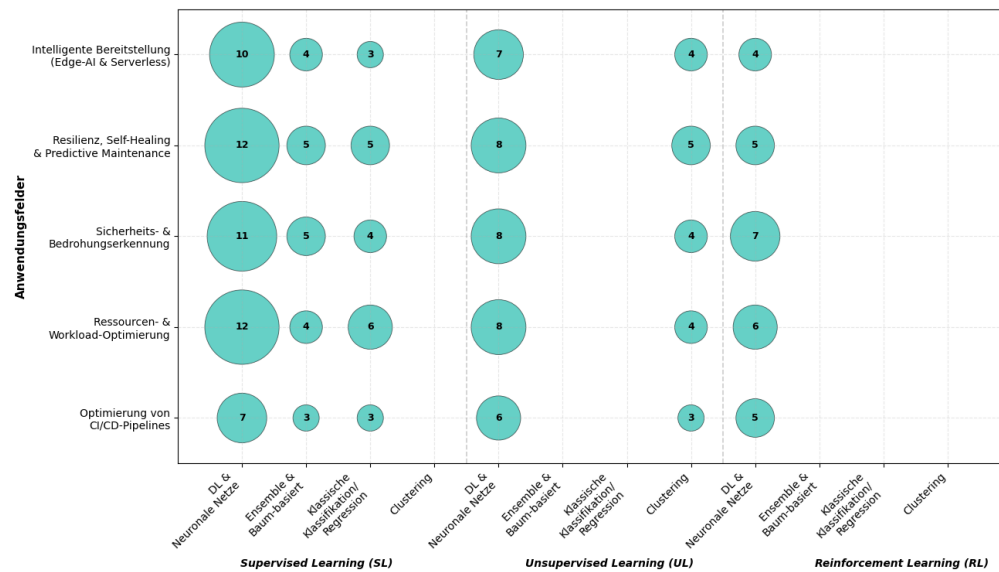


Abbildung 4.9: Mapping von Lernparadigmen, Algorithmen und Anwendungsfeldern

Im Supervised Learning dominiert die Kombination aus DL & neuronalen Netzen in allen Anwendungsfeldern. Besonders häufig wird sie bei Ressourcen- und Workload-Optimierung sowie Self-Healing/Predictive Maintenance genannt (jeweils $n=12$), gefolgt von Sicherheits- und Bedrohungserkennung ($n=11$) und Intelligenter Bereitstellung ($n=10$). Auch die Optimierung von CI/CD-Pipelines wird häufig mit DL adressiert ($n=7$). Insgesamt zeigt sich damit, dass Supervised Learning in der betrachteten Literatur vor allem in Verbindung mit DL und neuronalen Netzen eingesetzt wird.

Neben DL werden im Supervised Learning auch Ensemble- und baum-basierte Methoden sowie klassische Klassifikations- und Regressionsverfahren erwähnt, jedoch mit deutlich geringeren Häufigkeiten. Ensemble- und baum-basierte Ansätze liegen je nach Anwendungsfeld zwischen $n=3$ und $n=5$, während klassische Klassifikation/Regression insbesondere bei Ressourcen- und Workload-Optimierung ($n=6$) etwas häufiger vorkommt. Ansonsten sind die Werte ebenfalls geringer und liegen zwischen $n=2$ und $n=5$. Clustering wird im Kontext von Supervised Learning nicht beobachtet, sodass diese Zuordnungen nicht belegt sind ($n=0$).

Für Unsupervised Learning zeigt sich ein anderes Muster. Auch hier wird die Kombination aus DL & neuronalen Netzen über alle Anwendungsfelder hinweg am häufigsten genannt ($n=6-8$). Clustering-Verfahren treten als zweite zentrale Kategorie auf, werden jedoch deutlich seltener thematisiert ($n=3-5$). Insgesamt deutet dies darauf hin, dass Unsupervised Learning hauptsächlich dort eingesetzt wird, wo Abweichungen und Muster ohne gelabelte Daten erkannt werden sollen (z. B. bei Anomalien oder Wartungsvorher-

sagen). Andere Algorithmusklassen (z. B. Ensemble- bzw. klassische Klassifikations-/Regressionsverfahren) sind im Unsupervised-Learning-Kontext nicht belegt ($n=0$), wodurch sich eine klare Fokussierung auf DL/NN und Clustering ergibt.

Im Reinforcement Learning konzentrieren sich die Zuordnungen ausschließlich auf Deep Learning und neuronale Netze. Die Häufigkeiten sind insgesamt geringer als im Supervised- und Unsupervised Learning. Die höchsten Werte zeigen sich bei der Sicherheits- und Bedrohungserkennung ($n=7$) sowie bei der Ressourcen- und Workload-Optimierung ($n=6$). Für die Optimierung von CI/CD-Pipelines und Self-Healing/Predictive Maintenance liegen die Werte jeweils bei $n=5$, während Intelligente Bereitstellung am seltensten genannt wird ($n=4$). Andere Algorithmusklassen sind im RL-Kontext nicht belegt ($n=0$), was die enge Kopplung von RL an DL-basierte (Deep-RL) Ansätze in den betrachteten Publikationen unterstreicht.

Die Kreuztabellenanalysen machen sichtbar, welche Zusammenhänge in der Forschung wiederkehrend sind. Darauf aufbauend wird in Abschnitt 4.3 ein Matching-Framework abgeleitet, das die Ergebnisse strukturiert zusammenfasst. Ziel ist es, daraus eine nachvollziehbare Grundlage für die Bewertung bzw. Einordnung von KI-Ansätzen im Platform Engineering zu schaffen.

4.3. Matching-Framework

Aufbauend auf den quantitativen Ergebnissen aus Abschnitt 4.1 sowie den Mustern der Mapping Study in Abschnitt 4.2 wird in diesem Abschnitt ein konzeptionelles Matching-Framework abgeleitet. Ziel ist es, die identifizierten KI-Ansätze in übertragbare Anwendungsmuster für generische Use Cases im Platform Engineering zu strukturieren. Damit entsteht eine nachvollziehbare Grundlage, um KI-Lösungen später hinsichtlich operativem Nutzen und Umsetzbarkeit einzuordnen.

4.3.1. Muster 1: Prädiktives Auto-Scaling und Ressourcenoptimierung

Das Muster Prädiktives Auto-Scaling und Ressourcenoptimierung beschreibt KI-gestützte Verfahren, um Ressourcen vorausschauend zu skalieren und damit Kosten sowie Performance-Risiken zu optimieren. Im Unterschied zu klassischem Auto-Scaling auf Basis statischer Schwellenwerte werden historische Betriebsdaten genutzt, um Lastspitzen frühzeitig zu antizipieren und Skalierungsentscheidungen rechtzeitig auszuführen. Die Tabelle 4.1 fasst die charakteristischen Merkmale dieses Musters zusammen.

Tabelle 4.1: Prädiktives Auto-Scaling & Ressourcenoptimierung

Kategorie	Beschreibung	Beispielquellen
Plattformaufgabe	Vorausschauende Skalierung von Instanzen (EC2, RDS) und Pod-Anzahlen basierend auf Lastprognosen.	[21], [24]
Datenquellen	Metriken: CPU-Auslastung, Arbeitsspeicher, Disk-I/O, Netzwerkverkehr (CloudWatch/Prometheus).	[21], [24]
Datennutzung	Analyse historischer Zeitreihen zur Identifikation saisonaler Muster und Lastspitzen.	[21], [24]
KI-/ML-Methoden	LSTM (Long Short-Term Memory) für Zeitreihen; Reinforcement Learning (Q-Learning, PPO) zur Optimierung der Ressourcenzuteilung.	[21], [23]
Werkzeuge (Tools)	AWS Boto3 SDK, KEDA (Kubernetes Event-driven Autoscaling), TensorFlow/PyTorch für das Modelltraining.	[21], [24], [2]
Ziele/Nutzen	Reduzierung der Cloud-Kosten um bis zu 25–50%; Vermeidung von Performance-Engpässen durch Skalierung vor dem Erreichen von Schwellenwerten.	[21], [17]

Die Daten stammen typischerweise aus Cloud- und Cluster-Monitoring, z. B. AWS CloudWatch oder Prometheus. Relevante Metriken sind CPU-Auslastung, Arbeitsspeicher, Disk-I/O und Netzwerkverkehr [21, 2].

Die Metriken werden als Zeitreihen ausgewertet. Ziel ist es, wiederkehrende Muster (z. B. tages-/wochenbasierte Zyklen) und Lastspitzen zu erkennen, um daraus Prognosen für den zukünftigen Ressourcenbedarf abzuleiten [21, 2].

Für die Prognose kommen vor allem überwachte Lernverfahren zum Einsatz, insbesondere LSTM für Zeitreihen-Vorhersagen. Ergänzend wird Reinforcement Learning genutzt (z. B. Q-Learning, PPO), um eine Strategie zu lernen, welche Skalierungsaktion unter bestimmten Zuständen langfristig am besten ist [21, 25].

Für die technische Ausführung werden Cloud- und Plattform-Tools eingebunden, z. B. AWS Boto3 zur Anpassung von Instanzen und KEDA zur ereignis- bzw. metrikbasierten Pod-Skalierung in Kubernetes. Für Modelltraining und -betrieb werden typische ML-

Frameworks wie TensorFlow/PyTorch genutzt [21, 2, 24].

4.3.2. Muster 2: Intelligente CI/CD (Fehlervorhersage und adaptive Rollbacks)

Das zweite Architekturpattern konzentriert sich auf die intelligente CI/CD-Pipeline (Intelligent CI/CD). Dabei werden die Prozesse der kontinuierlichen Integration und Bereitstellung durch prädiktive Fehlererkennung und adaptive Rollback-Mechanismen optimiert. Während klassische Pipelines auf statischen Regeln basieren, ermöglichen KI-Modelle eine dynamische Reaktion auf Anomalien während des Release-Prozesses [9, 20, 19]. Die Tabelle 4.2 gibt einen detaillierten Überblick über die Komponenten dieses Patterns:

Tabelle 4.2: Intelligente CI/CD (Fehlervorhersage & adaptive Rollbacks)

Kategorie	Beschreibung	Beispielquellen
Plattformaufgabe	Vorhersage von Build-Fehlern; automatisierte Rollbacks bei Anomalien nach dem Deployment.	[20], [9]
Datenquellen	Logs: Build-Logs, Testergebnisse, Code-Metadaten (Commit-Historie), Fehlerraten während des Rollouts.	[20], [19]
Datennutzung	Training von Modellen auf historischen Fehlermustern, um risikoreiche Änderungen frühzeitig zu stoppen.	[20]
KI-/ML-Methoden	Supervised Learning (Random Forest, Entscheidungsbäume); Reinforcement Learning (DQN) zur Optimierung von Rollback-Entscheidungen.	[20]
Werkzeuge (Tools)	Jenkins X (mit ML-Plugins), Spinnaker, StackStorm (für ereignisbasierte Automatisierung).	[24], [19]
Ziele/Nutzen	Reduzierung der Deployment-Zeit um 30%; Steigerung der Systemstabilität durch proaktive Fehlervermeidung.	[24], [20]

Die prädiktive Fehlererkennung handelt konkret davon, KI-Modelle darauf zu trainieren, Muster in historischen Daten zu erkennen, die in der Vergangenheit, häufig zu Build- oder Deployment Fehler geführt haben. Hierbei werden Code-Änderungen und Testresultate analysiert, wodurch die KI ein hohes Risiko vorhersagen kann bevor der Code in die Produktionsumgebung gelangt. Dies ermöglicht es Plattform-Engineerings, risikoreiche Änderungen frühzeitig zu stoppen und die Erfolgsrate von Build signifikant zu

steigern.

Anomalie-basierte Rollbacks ergänzen diese Funktionalität, indem die KI Echtzeit-Metriken unmittelbar nach einem Release überwacht. Algorithmen wie Isolation FOrests oder Autoencoder identifizieren Abweichungen vom normalen Systemverhalten, wie etwas ungewöhnliche Latensptizen oder erhöhte Fehlerraten. Wird eine Anomalie erkannt, triggert das System autonom einen Rollback auf einen stabilen Zustand, um die Auswirkungen auf Endnutzer zu minimieren [20, 19, 18, 9].

KI wird zudem eingesetzt um Testsequenz innerhalb der Pipeline intelligent zu priorisieren. Mittels Reinforcement Learning lernt das System, welche Tests die höchste Wahrscheinlichkeit haben, kritische Fehler in einer spezifischen Code-Änderung aufzudecken. Dies reduziert die Gesamtlauzeit der Pipeline, ohne die Qualitätssicherung zu beeinträchtigen [26, 18].

Der Einsatz dieses Ptttern führt zu einer Reduktion des manuellen Intervemtionsaufwands bei fehlgeschlagenen Deployments. Die Deployment-Zyklen werden durch automatisierte Entscheidungsfindung um etwas 30% verkürzt [24]. Zudem verbessert sich die Systemstabilität, da die KI als SSicherheitsnetz"fungiert und fehlerhafte Releases abfängt, bevor sie kritische Systemzustände verursachen [20, 18].

4.3.3. Muster 3: KI-gestützte Observability und Ursachenanalyse (AIOps)

Das dritte Muster im Matching-Framework fokussiert sich auf KI-gestützte Observability und Ursachenanalyse, auch bekannt als AIOps (Artificial Intelligence for IT Operations). In modernen Cloud-native Umgebungen übersteigt die Menge der genreirten Telemetriedaten oft die Kapazitäten manueller Analyse. Klassische Monitoring-Ansätze sind häufig reaktiv und führen zu einer hohen Anzahl von Fehlalarmen. Durch den Einsatz von KI können Plattform-Engineers unstrukturierte Log-Daten ud komplexe Systemzustände in Echtzeit interpetieren. Die Tabelle 4.3 fasst die wesentlichen Merkmale dieses Musters zusammen:

Tabelle 4.3: KI-gestützte Observability & Ursachenanalyse (AIOps)

Kategorie	Beschreibung	Beispielquellen
Plattformaufgabe	Log-basierte Anomalieerkennung; automatisierte Root-Cause-Analyse (RCA); natürlichsprachliche Schnittstelle für Cluster-Abfragen.	[2], [25]
Datenquellen	Telemetrie: System-Logs, Kubernetes-Events, Traces (OpenTelemetry), Warnsignale von Sensoren/Nodes.	[2], [22]
Datennutzung	Mustererkennung in unstrukturierten Log-Daten zur Korrelation von Fehlern über Microservices hinweg.	[2]
KI-/ML-Methoden	LLMs (Natural Language Processing) zur Interpretation von Fehlermeldungen; Clustering (K-Means) zur Bündelung ähnlicher Alerts.	[2], [9]
Werkzeuge (Tools)	K8sGPT, OpenLLMetry, Prometheus/Grafana mit KI-Erweiterungen.	[2], [25]
Ziele/Nutzen	Reduktion der mittleren Zeit bis zur Fehlerbehebung (MTTR) um bis zu 39%; Reduktion von Fehlalarmen (False Positives) in der Alarmierung.	[25]

Im Kern adressiert das Pattern operative Plattformaufgaben im AIOps-Kontext: Es erkennt Auffälligkeiten in Logs, unterstützt eine automatisierte Ursachenanalyse und ermöglicht zusätzlich natürlichsprachliche Abfragen an den Cluster, um Fehler schneller einzugrenzen [2, 25].

Als Datengrundlage werden verschiedene Telemetriequellen zusammengeführt, insbesondere System-Logs, Kubernetes-Events und verteilte Traces (z. B. OpenTelemetry) sowie Warnsignale aus der Infrastruktur. Dadurch entsteht ein konsistenteres Bild des Systemzustands über mehrere Komponenten hinweg [2, 22].

Die Datennutzung fokussiert unstrukturierte Log-Daten: Muster werden erkannt und so korreliert, dass Zusammenhänge von Fehlern über Microservices hinweg sichtbar werden [2].

Methodisch werden große Sprachmodelle genutzt, um Fehlermeldungen verständlich zu interpretieren, während Clustering (z. B. K-Means) ähnliche Alarmer bündelt und Alarmfluten reduziert [2, 9].

Auf Werkzeugebene wird das Pattern typischerweise durch K8sGPT (natürlichsprachliche Unterstützung), OpenLLMetry (herstellerneutrale Integration) sowie Prometheus/Grafana mit KI-Erweiterungen umgesetzt [2, 25].

Der operative Nutzen liegt vor allem in schnellerer Fehlerbehebung und besserer Alarmqualität; die Quellen berichten u. a. eine MTTR-Reduktion von bis zu 39 % sowie weniger Fehlalarme in der Alarmierung [25].

4.3.4. Muster 4: Intelligente Sicherheit und Bedrohungserkennung

Intelligente Sicherheit und Bedrohungserkennung beschreibt KI-gestützte Verfahren, um Sicherheitsereignisse in Cloud-Native-Plattformen frühzeitig zu erkennen und schneller einzuordnen. Im Unterschied zu rein regelbasierten Ansätzen werden dabei Muster aus Betriebs- und Sicherheitsdaten gelernt, um Auffälligkeiten automatisch zu identifizieren. Die Tabelle 4.4 fasst die Kernelemente dieses Patterns zusammen.

Tabelle 4.4: Intelligente Sicherheit & Bedrohungserkennung

Kategorie	Beschreibung	Beispielquellen
Plattformaufgabe	Erkennung von Intrusionen (DDoS, Malware), Anomalieerkennung im Netzwerkverkehr sowie im Benutzerverhalten.	[27], [28]
Datenquellen	Netzwerk: IP-Traffic, System-Logs, Cloud-Audit-Logs, Benutzer-Zugriffsmuster.	[27], [28]
Datennutzung	Abgleich von Echtzeit-Datenströmen mit Baselines für „normales“ Verhalten zur Identifikation von Ausreißern.	[27], [28]
KI-/ML-Methoden	Deep Learning (CNN für visuelle Musteranalyse); XGBoost/Random Forest zur Klassifikation auf unausgewogenen Datensätzen (Klassenungleichgewicht).	[27], [28]
Werkzeuge (Tools)	IBM Watson AI, Microsoft Sentinel, Trivy/Clair (für Scans in CI/CD).	[24]
Ziele/Nutzen	Detektionsrate von bis zu 95% bei fortgeschrittenen Bedrohungen; Reduktion von Sicherheitsvorfällen.	[27], [25]

Im Kern adressiert das Pattern die Erkennung von Angriffen (z. B. DDoS, Malware) sowie Anomalien im Netzwerkverkehr und im Benutzerverhalten. Dazu werden IP- und Netzwerk-Traffic, System- und Cloud-Audit-Logs sowie Zugriffsmuster als Datenbasis zusammengeführt [27, 28].

Die Datennutzung erfolgt häufig über den Abgleich von Echtzeit-Datenströmen mit Baselines für „normales“ Verhalten, sodass Ausreißer priorisiert und schneller untersucht werden können. Als Methoden werden Deep-Learning-Verfahren (z. B. CNNs zur Mustererkennung) sowie Modelle wie XGBoost oder Random Forest eingesetzt, insbeson-

dere um trotz Klassenungleichgewicht robuste Klassifikationen zu ermöglichen [27, 28]. Auf Werkzeugebene werden Security-Plattformen wie IBM Watson AI oder Microsoft Sentinel genutzt und durch Scanner wie Trivy/Clair ergänzt, die Sicherheitsprüfungen in CI/CD-Pipelines integrieren. Der operative Nutzen liegt in höherer Erkennungsleistung und weniger Sicherheitsvorfällen; die Literatur berichtet Detektionsraten von bis zu 95 % bei fortgeschrittenen Bedrohungen [24, 27, 25].

5

Anwendung der Literaturrecherche

Darstellung des theoretischen Konzepts, seiner Bestandteile und Begründung.

5.1. Bewertungskonzept/ Framework

Hier kommt meine Antwort zu RQ3 hin.

5.2. Analyse der Bosch Digital Manufacturing Plattform

6

Diskussion

Die Ergebnisse werden eingeordnet, Limitationen diskutiert und Implikationen abgeleitet.

6.1. Beantwortung der Forschungsfragen

Zusammenführung der Ergebnisse zur direkten Beantwortung der Forschungsfragen.

6.2. Praxisrückschluss

Übertragbarkeit und Nutzen der Ergebnisse für die Praxis.

6.3. Limitationen

Grenzen der Studie und Ansatzpunkte für Verbesserungen.

6.4. Handlungsempfehlungen

An Bosch gerichtet

7

Zusammenfassung und Ausblick

Abschließende Zusammenfassung der Arbeit sowie ein Ausblick auf zukünftige Forschung.

7.1. Zusammenfassung

Die wichtigsten Erkenntnisse und Ergebnisse der Arbeit werden hier zusammengefasst.

7.2. Ausblick

Mögliche zukünftige Forschungsrichtungen und offene Fragen werden hier diskutiert.

Tabellenverzeichnis

2.1	Beschreibung Algorithmen und Methoden	5
3.1	Auswahlkriterien	9
3.2	Datenextraktion	10
4.1	Prädiktives Auto-Scaling & Ressourcenoptimierung	26
4.2	Intelligente CI/CD (Fehlervorhersage & adaptive Rollbacks)	27
4.3	KI-gestützte Observability & Ursachenanalyse (AIOps)	29
4.4	Intelligente Sicherheit & Bedrohungserkennung	30

Abbildungsverzeichnis

4.1	Jährliche und thematische Verteilung der DevOps AI-Forschung	12
4.2	Verteilung der Anwendungsbereiche	13
4.3	Relative Häufigkeit der Herausforderungen	14
4.4	Formen des maschinellen Lernens	15
4.5	Verteilung der Algorithmen und angesprochenen Methoden	16
4.6	Korrelation zwischen Anwendungsfelder (Use Cases) und Herausforderungen	18
4.7	Korrelation zwischen Lernparadigmen und Algorithmen	20
4.8	Korrelation zwischen Herausforderungen und KI-Technologie	21
4.9	Mapping von Lernparadigmen, Algorithmen und Anwendungsfeldern . . .	23

Literaturverzeichnis

- [1] Rahul Amte. "Cloud-Native AI: Challenges and Innovations in Deploying Large-Scale Machine Learning Models". In: *ISCSITR - INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING (ISCSITR-IJSRAIML) ISSN (Online): 3067-753X* 6.2 (März 2025), S. 9–18. (Besucht am 05. 11. 2025).
- [2] Adel Zaalouk u. a. "CLOUD NATIVE ARTIFICIAL INTELLIGENCE". In: ().
- [3] Overview. <https://kubernetes.io/docs/concepts/overview/>. (Besucht am 15. 12. 2025).
- [4] juliakm. *What Is Platform Engineering?* <https://learn.microsoft.com/en-us/platform-engineering/what-is-platform-engineering>. (Besucht am 15. 12. 2025).
- [5] *Was ist DevOps? Erläuterung zu DevOps — Microsoft Azure*. <https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-devops>. (Besucht am 16. 12. 2025).
- [6] Atlassian. *Was ist DevOps?* <https://www.atlassian.com/de/devops>. (Besucht am 16. 12. 2025).
- [7] *Was ist CI/CD? — Automatisierung in der Softwareentwicklung*. <https://www.redhat.com/de/topics/deis-ci-cd>. (Besucht am 16. 12. 2025).
- [8] Dhia Elhaq Rzig u. a. *Empirical Analysis on CI/CD Pipeline Evolution in Machine Learning Projects*. <https://arxiv.org/abs/2403.12199v4>. März 2024. (Besucht am 16. 12. 2025).
- [9] Aliyu Enemosah. "Enhancing DevOps Efficiency through AI-Driven Predictive Models for Continuous Integration and Deployment Pipelines". In: *International Journal of Research Publication and Reviews* 6.1 (Jan. 2025), S. 871–887. ISSN: 25827421. DOI: [10.55248/gengpi.6.0125.0229](https://doi.org/10.55248/gengpi.6.0125.0229). (Besucht am 03. 11. 2025).
- [10] Dirk Valkenborg u. a. "Supervised Learning". In: *American Journal of Orthodontics and Dentofacial Orthopedics* 164.1 (Juli 2023), S. 146–149. ISSN: 08895406. DOI: [10.1016/j.ajodo.2023.04.010](https://doi.org/10.1016/j.ajodo.2023.04.010). (Besucht am 27. 11. 2025).
- [11] Dirk Valkenborg u. a. "Unsupervised Learning". In: *American Journal of Orthodontics and Dentofacial Orthopedics* 163.6 (Juni 2023), S. 877–882. ISSN: 08895406. DOI: [10.1016/j.ajodo.2023.04.001](https://doi.org/10.1016/j.ajodo.2023.04.001). (Besucht am 27. 11. 2025).
- [12] Majid Ghasemi u. a. *An Introduction to Reinforcement Learning: Fundamental Concepts and Practical Applications*. Aug. 2024. DOI: [10.48550/arXiv.2408.07712](https://doi.org/10.48550/arXiv.2408.07712).

- [13] Kai Petersen, Sairam Vakkalanka und Ludwik Kuzniarz. "Guidelines for Conducting Systematic Mapping Studies in Software Engineering: An Update". In: *Information and Software Technology* 64 (Aug. 2015), S. 1–18. ISSN: 0950-5849. DOI: [10.1016/j.infsof.2015.03.007](https://doi.org/10.1016/j.infsof.2015.03.007). (Besucht am 06. 11. 2025).
- [14] Barbara Kitchenham und Stuart M. Charters. (PDF) *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. <https://www.researchgate.net/publication/30> (Besucht am 06. 11. 2025).
- [15] Claes Wohlin. "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering". In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. EASE '14. New York, NY, USA: Association for Computing Machinery, Mai 2014, S. 1–10. ISBN: 978-1-4503-2476-2. DOI: [10.1145/2601248.2601268](https://doi.org/10.1145/2601248.2601268). (Besucht am 06. 11. 2025).
- [16] Yao Lu u. a. *Computing in the Era of Large Generative Models: From Cloud-Native to AI-Native*. Jan. 2024. DOI: [10.48550/arXiv.2401.12230](https://doi.org/10.48550/arXiv.2401.12230). arXiv: [2401.12230 \[cs\]](https://arxiv.org/abs/2401.12230). (Besucht am 12. 11. 2025).
- [17] Karthik Puthraya, Rachit Gupta und Beverly DSouza. "The Role of Cloud-Native Architectures in Accelerating Machine Learning Workflows through Data Engineering Innovations". In: (Feb. 2025). ISSN: 2945-3437. DOI: [10.5281/ZENODO.15106432](https://doi.org/10.5281/ZENODO.15106432). (Besucht am 05. 11. 2025).
- [18] Satheesh Reddy Gopireddy. "Integrating AI into DevOps: Leveraging Machine Learning for Intelligent Automation in Azure". In: *International Journal of Science and Research (IJSR)* 11.6 (Juni 2022), S. 2035–2039. ISSN: 23197064. DOI: [10.21275/SR22619111757](https://doi.org/10.21275/SR22619111757). (Besucht am 03. 11. 2025).
- [19] Suprit Pattanayak, Pranav Murthy und Aditya Mehra. "Integrating AI into DevOps Pipelines: Continuous Integration, Continuous Delivery, and Automation in Infrastructural Management: Projections for Future". In: *International Journal of Science and Research Archive* 13.1 (Okt. 2024), S. 2244–2256. ISSN: 25828185. DOI: [10.30574/ijjsra.2024.13.1.1838](https://doi.org/10.30574/ijjsra.2024.13.1.1838). (Besucht am 03. 11. 2025).
- [20] Venkata Mohit Tamanampudi. "AI-Enhanced Continuous Integration and Continuous Deployment Pipelines: Leveraging Machine Learning Models for Predictive Failure Detection, Automated Rollbacks, and Adaptive Deployment Strategies in Agile Software Development". In: 10 ().
- [21] Sudip Poudel u. a. "AI-Driven Intelligent Auto-Scaling for Cloud Resource Optimization 1". In: *Journal of Advanced College of Engineering and Management* Vol. 11 (Okt. 2025), S. 27–36. DOI: [10.3126/jacem.v11i1.84521](https://doi.org/10.3126/jacem.v11i1.84521).

- [22] Abhishek Gupta und Yashovardhan Chaturvedi. "Cloud-Native ML: Architecting AI Solutions for Cloud-First Infrastructures". In: *Nanotechnology Perceptions* 20 (Dez. 2024), S. 930–939. DOI: [10.62441/nano-ntp.v20i7.4004](https://doi.org/10.62441/nano-ntp.v20i7.4004).
- [23] Josson Paul Kalapparambath, Yugandhar Suthari und Gaurav Mishra. "Advancing Distributed Systems with Reinforcement Learning: A New Frontier in AI-Integrated Software Engineering". In: (März 2025). ISSN: 2945-3585. DOI: [10.5281/ZENODO.15305618](https://doi.org/10.5281/ZENODO.15305618). (Besucht am 02. 12. 2025).
- [24] Giridhar Kankanala und Sudheer Amgothu. "AI/ML – DevOps Automation". In: 13 (Okt. 2024), S. 111–117.
- [25] Varun Tamminedi. "Automating Kubernetes Operations with AI and Machine Learning". In: *IJFMR - International Journal For Multidisciplinary Research* 6.6 (Dez. 2024). ISSN: 2582-2160. DOI: [10.36948/ijfmr.2024.v06i06.33430](https://doi.org/10.36948/ijfmr.2024.v06i06.33430). (Besucht am 03. 11. 2025).
- [26] Gopinath Kathiresan. "Cybersecurity Risk Modeling in CI/CD Pipelines Using Reinforcement Learning for Test Optimization". In: *International Journal of Innovative Science and Research Technology* (Mai 2025), S. 15–25. DOI: [10.38124/ijisrt/25may339](https://doi.org/10.38124/ijisrt/25may339). (Besucht am 03. 11. 2025).
- [27] Jeanette Uddoh u. a. "AI-Based Threat Detection Systems for Cloud Infrastructure: Architecture, Challenges, and Opportunities". In: *Journal of Frontiers in Multidisciplinary Research* 2.2 (2021), S. 61–67. ISSN: 30509718, 30509726. DOI: [10.54660/.IJFMR.2021.2.2.61-67](https://doi.org/10.54660/.IJFMR.2021.2.2.61-67). (Besucht am 04. 11. 2025).
- [28] Vijay Govindarajan. *Machine Learning Based Approach for Handling Imbalanced Data for Intrusion Detection in the Cloud Environment*. März 2025, S. 815. DOI: [10.1109/ICDT63985.2025.10986614](https://doi.org/10.1109/ICDT63985.2025.10986614).
- [29] Muhammad Talha Tahir Bajwa u. a. "CLOUD-NATIVE ARCHITECTURES FOR LARGE-SCALE AI-BASED PREDICTIVE MODELING". In: *Journal of Emerging Technology and Digital Transformation* 4.2 (Aug. 2025), S. 207–221. ISSN: 3006-9726. (Besucht am 25. 10. 2025).
- [30] Yu Jeffrey Hu, Jeroen Rombouts und Ines Wilms. *MLOps Monitoring at Scale for Digital Platforms*. Apr. 2025. DOI: [10.48550/arXiv.2504.16789](https://doi.org/10.48550/arXiv.2504.16789). arXiv: [2504.16789 \[econ\]](https://arxiv.org/abs/2504.16789). (Besucht am 18. 11. 2025).
- [31] Swarup Panda. *Scalable Artificial Intelligence Systems: Cloud-Native, Edge-AI, MLOps, and Governance for Real-World Deployment*. Deep Science Publishing, Juli 2025. ISBN: 978-93-7185-753-6.

- [32] Vijay Kartik Sikha. "Cloud-Native Application Development for AI- Conducive Architectures." In: *International Journal on Recent and Innovation Trends in Computing and Communication* 11.11 (2023).
- [33] Sumanth Tatineni. *Integrating Artificial Intelligence with DevOps: Advanced Techniques, Predictive Analytics, and Automation for Real-Time Optimization and Security in Modern Software Development*. Libertatem Media Private Limited, März 2024. ISBN: 978-81-971382-1-8.