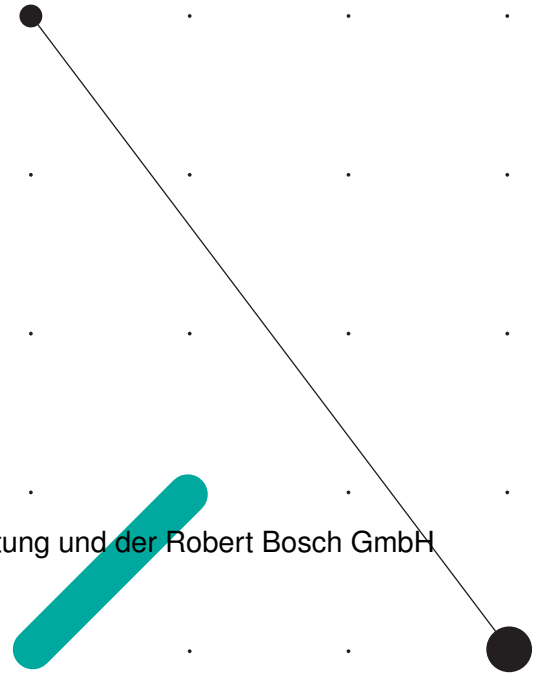
Hochschule Konstanz
Department of Computer Science

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

1

Einleitung

Mit der zunehmenden Verbreitung von Cloud-Native-Technologien und Plattform-Engineering Ansätzen, stehen Unternehmen vor der Herausforderung, ihre Software-Entwicklung und den Plattformbetriebt 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. Hierzu werden zentrale Begriffe aus dem Bereich Cloud Native Platform Engineering, DevOps, CI/CD, Lernparadigmen des maschinellen Lernens und AIOps erläutert. Anschließend werden relevante verwandte Arbeiten beschrieben und kritisch eingeordnet.

2.1. Grundlagen

Ziel dieses Abschnitts ist es, die für die Arbeit relevanten Konzepte aus Cloud-Native-Technologien, DevOps sowie Künstlicher Intelligenz (KI) im Plattformbetrieb strukturiert darzustellen. Die dargestellten Konzepte bilden die Basis für die anschließende Literaturanalyse und die Auswertung der KI-Anwendungen im Platform Engineering.

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, er-

weiterbare Open-Source-Lösung zur Verwaltung containerisierter Workloads und unterstützt insbesondere deklarative Konfiguration und Automatisierung. Aufbauend auf diesem deklarativen Ansatz wird Kubernetes häufig in GitOps-basierten Betriebsmodellen eingesetzt, bei denen der gewünschte Systemzustand versionskontrolliert abgelegt und automatisiert mit dem laufenden Cluster abgeglichen wird [3, 4].
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 [5].

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. DevOps entstand vor allem als Antwort auf längere Release-Zyklen und Silos zwischen Entwicklung und Betrieb, die schnelle Änderungen in produktiven Systemen erschwerten. Ziel ist es, Zusammenarbeit und Kommunikation zu stärken und Abläufe so zu gestalten, dass Änderungen häufiger, zuverlässiger und mit klaren Verantwortlichkeiten bereitgestellt werden können. Der DevOps-Ansatz entstand im Kontext verteilter Plattformumgebungen, in denen klassische Trennungen zwischen Entwicklung und Betrieb an ihre Grenzen stießen [6, 7].
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 [8, 9].

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 [10, 11].
UL arbeitet mit ungelabelten Daten und zielt darauf ab, darin Strukturen, Cluster oder Auffälligkeiten zu erkennen, ohne dass eine explizite Zielvariable vorgegeben ist [10,

12].
RL beschreibt Verfahren, bei denen ein Agent durch Interaktion mit einer Umgebung Handlungen lernt, um eine langfristige (kumulative) Belohnung zu maximieren [10, 13]. 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

AIOps (Artificial Intelligence for IT Operations) bezeichnet den Einsatz von KI-Technologien zur Automatisierung und Optimierung von IT-Betriebsprozessen. Im Kontext dieser Arbeit liegt der Fokus auf ÄI for Ops", also der Anwendung von KI zur Überwachung, Analyse und Optimierung von Cloud-native Plattformen und deren Betrieb. Im Gegensatz zu klassischen Monitoring-Ansätzen, die überwiegend auf statischen Schwellenwerten basieren, ermöglichen KI-gestützte Verfahren eine proaktive Erkennung von Anomalien und betrieblichen Mustern auf Basis großer, heterogener Betriebsdaten. AIOps ist dabei klar von MLOps abzugrenzen, das primär die Entwicklung, das Training und den Lebenszyklus von KI-Modellen adressiert, während AIOps den stabilen und effizienten Plattformbetrieb unterstützt [14, 15].

2.2. Verwandte Arbeiten

Dieser Abschnitt ordnet zentrale Arbeiten zur KI-Integration im Cloud-Native Platform Engineering ein und grenzt den Fokus der Arbeit ab. Im Mittelpunkt steht KI-gestützter Plattformbetrieb (AI for Ops) als operative Unterstützung für Platform Engineers, nicht MLOps als Infrastruktur für datenwissenschaftliche Arbeitsabläufe.
Ein Teil der Literatur untersucht KI zur Automatisierung von Kubernetes- und Cloud-Plattformen. Dabei werden manuelle Betriebsaufgaben reduziert und Ressourcen effizienter zugeteilt [16, 14]. Weitere Arbeiten adressieren prädiktive Skalierung, um Lastverläufe vorherzusagen und Ressourcen vorausschauend anzupassen [17]. Auch Reinforcement Learning wird zur dynamischen Steuerung der Lastverteilung und zur Erhöhung der Fehlertoleranz eingesetzt [18].
Ein weiterer Schwerpunkt liegt auf KI in DevOps- und CI/CD-Prozessen. Hierzu zählen die Vorhersage von Fehlern in Build- und Bereitstellungsprozessen sowie automatisierte Gegenmaßnahmen wie Rollbacks oder Anomaliealarme [10, 19, 20]. Ergänzend wird AIOps als Ansatz beschrieben, um komplexe Cloud-Infrastrukturen durch automatisierte Analyse und Monitoring besser beherrschbar zu machen [21, 15]. Sicherheitsbezogene Aspekte werden von Uddoh u. a. [22] ebenfalls aufgegriffen, etwa durch KI-basierte Erkennung von Bedrohungen und automatisierte Reaktionen im Plattformbetrieb. Das CNCF-Whitepaper verortet diese Richtung als „AI for Cloud Native Operations“ und beschreibt Assistenzwerkzeuge im operativen Cloud-Native Betrieb [2].
Kritisch ist festzuhalten, dass viele Arbeiten einzelne Anwendungsfälle isoliert betrachten oder stark werkzeug- bzw. monitoringgetrieben sind. Zudem existieren thematisch nahe Beiträge mit MLOps- oder Datenpipeline-Fokus, die den Plattformbetrieb aus Sicht von Platform Engineers nur indirekt adressieren und daher bewusst ausgeklammert wurden. Eine systematische, übergreifende Einordnung entlang typischer Aufgaben im Platform Engineering sowie eine Bewertung von Übertragbarkeit und praktischem Nutzen bleibt häufig offen. An dieser Stelle setzt die vorliegende Arbeit an.

Auf der Grundlage dieser Einordnung wird in Kapitel 3 das methodische Vorgehen beschrieben. Dort wird dargestellt, wie die Literaturanalyse durchgeführt wurde, um die späteren Ergebnisse der Mapping Study nachvollziehbar und reproduzierbar abzuleiten.

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 [23] 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 [23] 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 [24] berücksichtigt. In der Durchführungsphase werden identifizierte Studien anhand festgelegter Ein- und Ausschlusskriterien geprüft. Zusätzlich wird das Schneeballverfahren nach Wohlin [25] 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 [25] 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.
Platform-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

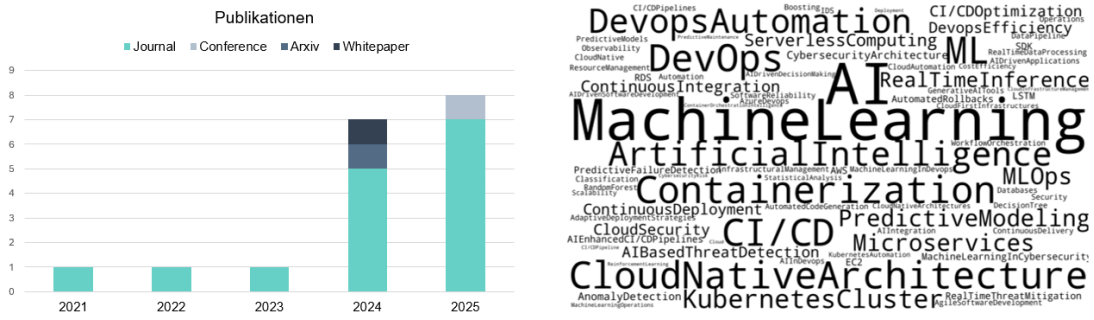
Ergebnisse der Literaturanalyse

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 Platform-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 Platform-Engineering-Kontexten hin. Das lässt sich auch unter anderem auf Grund des ChatGPT/GenAI Hype in den letzten beiden Jahren erklären [21].

Ergänzend dazu zeigt eine Word Cloud (Abbildung 4.1 b) die am häufigsten vorkommenden Keywords aus allen Publikationen. Die Visualisierung bietet einen schnellen Überblick über zentrale thematische Schwerpunkte der Literatur und unterstützt die anschließende inhaltliche Analyse.



a) Verteilung ausgewählter Publikationen nach Jahr und Typ

b) Ergebnisse der Themenmodellierung aus den Keywords der 21 ausgewählten Artikel

Abbildung 4.1: Jährliche und thematische Verteilung der DevOps AI-Forschung

Diese Betrachtung bildet die Grundlage für die folgenden Unterkapitel, in denen die Studien hinsichtlich ihrer inhaltlichen Merkmale detaillierter ausgewertet werden.

4.1.1. Anwendungsbereiche der KI im Platform Engineering

Im ersten Schritt der quantitativen Analyse wurden die insgesamt 18 identifizierten Studien hinsichtlich ihrer Hauptanwendungsbereiche untersucht. Dazu wurden die ausgewählten Bereiche in vier Kategorien unterteilt: Optimierung von CI/CD-Pipelines, Ressourcen- und Workload-Optimierung, Sicherheits- und Bedrohungserkennung sowie Betrieb und Orchestrierung der Plattform (GenOps).
Die Auswertung zeigt, dass Ressourcen- und Workload-Optimierung mit sieben Publikationen am häufigsten als dominanter Use Case auftritt. Ein plausibler Grund ist, dass Maßnahmen in diesem Bereich häufig einen direkten, messbaren Hebel auf Kosten und Performance haben (z.B. bessere Ressourcen-Auslastung oder stabilere Laufzeiten) [26]. An zweiter Stelle folgt der Bereich Betrieb und Orchestrierung mit fünf Publikationen. Dies deutet darauf hin, dass KI-Ansätze besonders im laufenden Betrieb relevant sind, etwa zur Unterstützung von Monitoring oder der automatisierten Steuerung von Plattformkomponenten. [20] Die Optimierung von CI/CD-Pipelines und Sicherheits- und Bedrohungserkennung sind mit jeweils drei Publikationen vertreten. Das lässt sich daraus erklären, dass diese beiden Anwendungsfelder zwar in vielen Publikationen thematisiert werden, jedoch seltener als primärer Fokus der Studie dienen. Die Optimierung der CI/CD-Pipeline wird oft als Teilaspekt in ein größeres Gesamtbild der Infrastruktur-Transformation behandelt [15].

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.

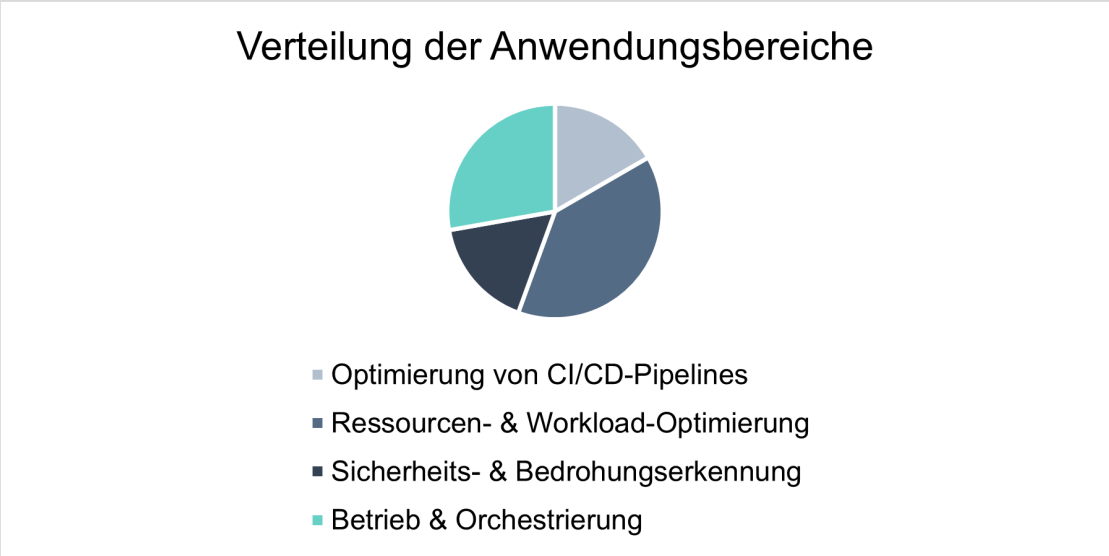


Abbildung 4.2: Verteilung der Anwendungsbereiche

4.1.2. Herausforderungen der KI-Integration im Platform Engineering

Im nächsten Schritt wurden die in den Publikationen beschriebenen Herausforderungen analysiert, die beim Einsatz von KI im Platform 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 [19].

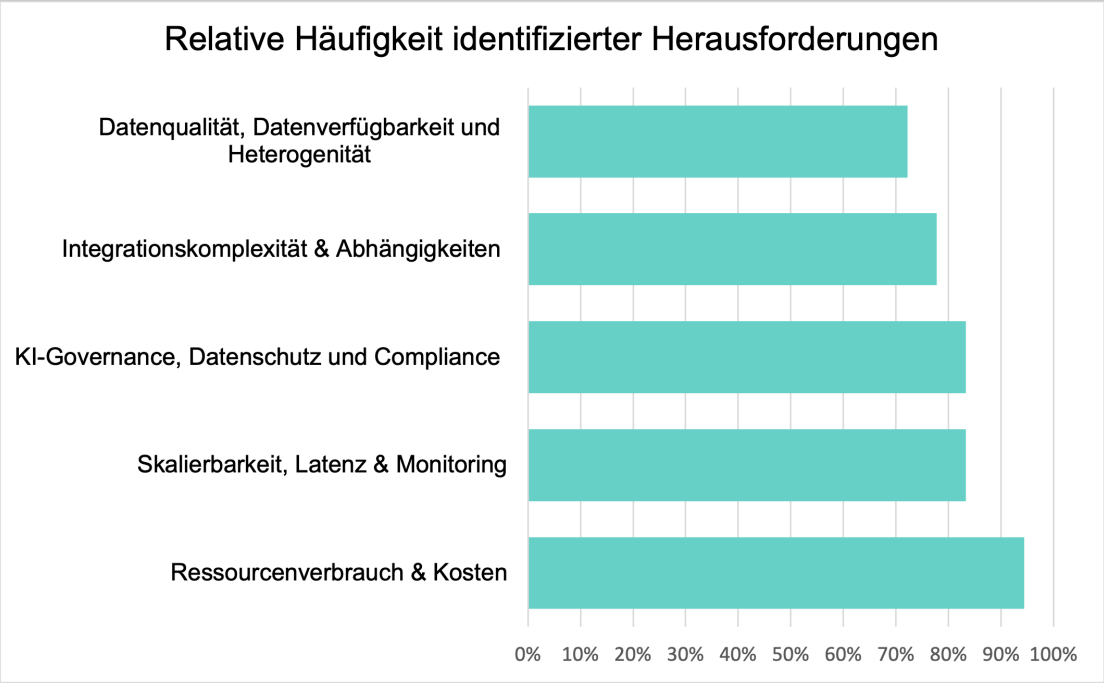


Abbildung 4.3: Relative Häufigkeit der Herausforderungen

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.

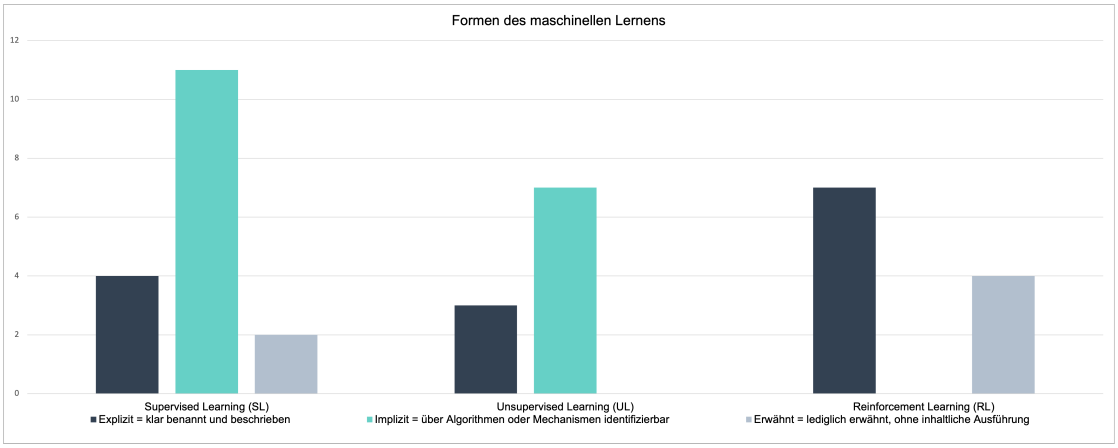


Abbildung 4.4: Formen des maschinellen Lernens

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 [19].

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 [10], 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 [10, 19].
Diese Verteilung ist in der Abbildung 4.5 dargestellt.

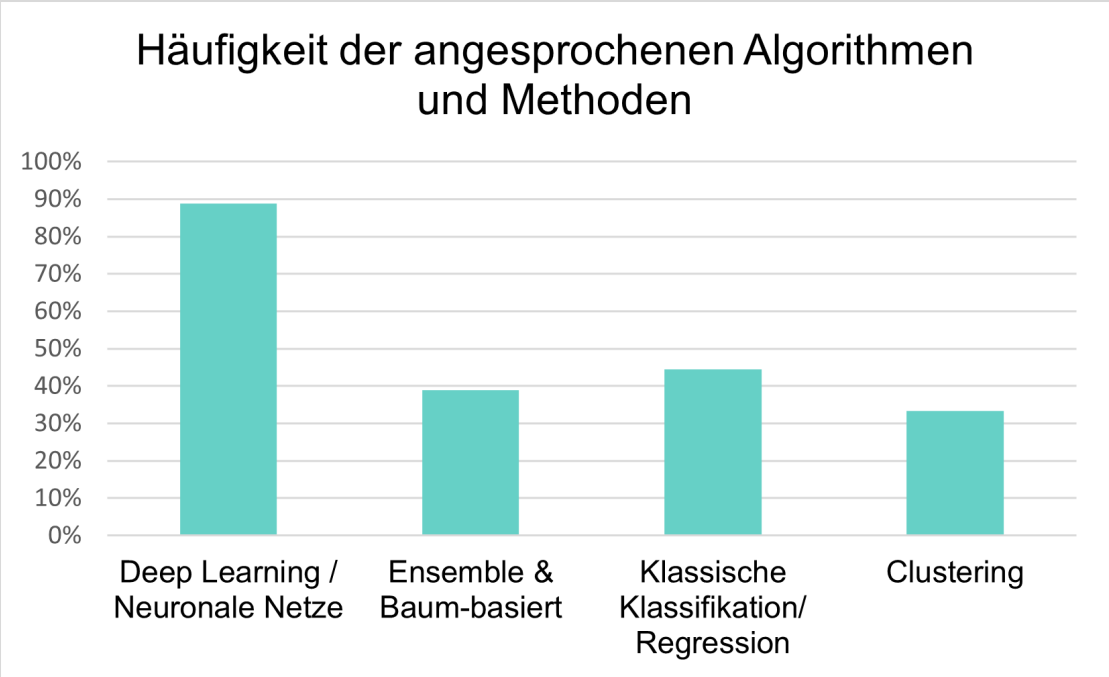


Abbildung 4.5: Verteilung der Algorithmen und angesporchenen Methoden

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) sowie Anwendungsfelder und Datenquellen (Abschnitt 4.2.3). 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 Anwendungsbereiche berücksichtigt und den jeweils adressierten Herausforderungen zugeordnet. Die Häufigkeiten geben an, wie oft eine bestimmte Kombination in der Literatur thematisiert wurde.

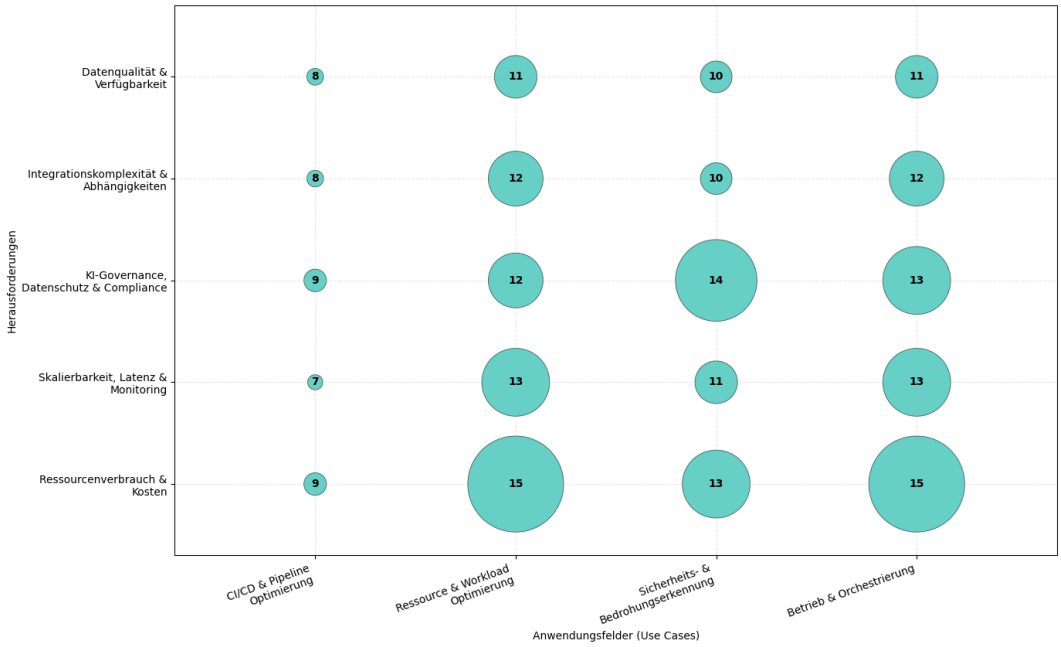


Abbildung 4.6: Korrelation zwischen Anwendungsfelder (Use Cases) und Herausforderungen

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 [17]. 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 [27].

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 [19].
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.

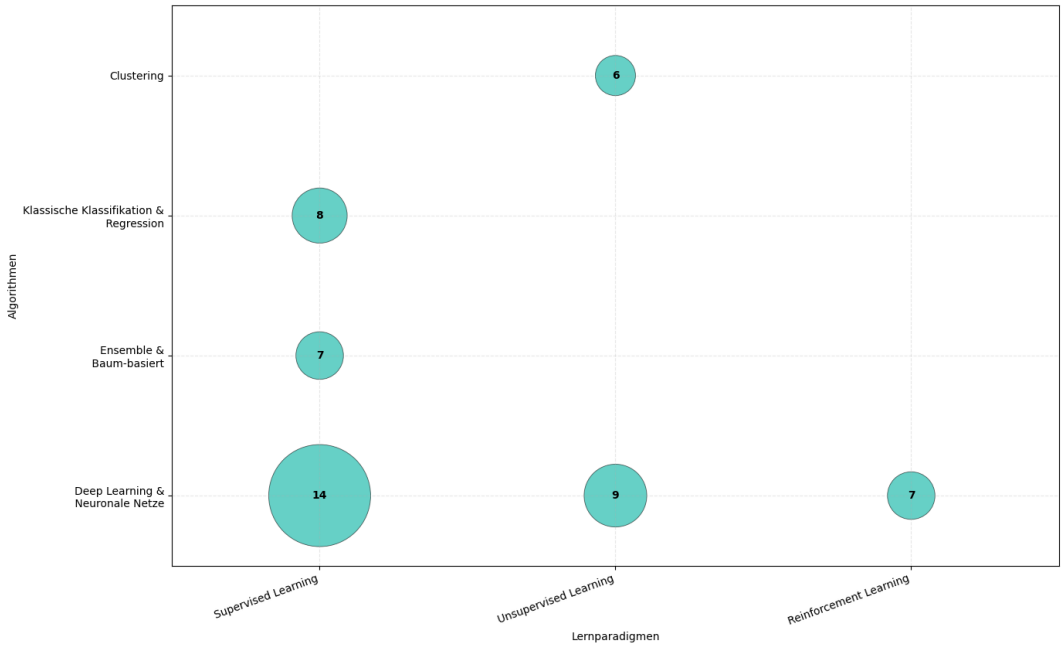


Abbildung 4.7: Korrelation zwischen Lernparadigmen und Algorithmen

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 einsetbar ist. Solange gelabelte Daten vorliegen, können verschiedene Algorithmen angewendet werden. Ein wesentlicher Grund für die Dominanz neuronaler Netze ist ihre Fähigkeit, komplexe und nicht-lineare Beziheungen in großen Datenmengen zu erfassen, die für einfachere Modelle nicht erkennbar sind [19].
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 [10] beschrieben. Der praktische Einsatz ist jedoch erschwert, da sich das als „normal“ erlernte Systemverhalten in dynamischen CI/CD-Umgebungen durch neue Funktionen oder Architekturänderungen häufig verschiebt und dadurch vermehrt Fehlalarme ausgelöst werden können, was den operativen Nutzen solcher Ansätze einschränkt [19].
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 [18], insbesondere wenn komplexe Zustandsräume berücksichtigt werden müssen. Ein zentrales Problem von RL ist jedoch die hohe Komplexität bei der Modellierung interagierender Systemkomponenten sowie der damit verbundene Rechenaufwand, wodurch eine Überführung in den produktiven Betrieb häufig erschwert wird [18]. Dies verdeutlicht, warum Reinforcement-Learning-Verfahren in der Praxis bislang nur begrenzt eingesetzt werden. Insgesamt bleibt die Zuordnung zu spezifischen Algorithmen jedoch begrenzt, weshalb viele Kombinationen in der Tabelle nicht belegt sind.

4.2.3. Zusammenspiel der Anwendungsfelder und Datenquellen

In diesem Unterkapitel wird analysiert, welche Datenquellen in den identifizierten Anwendungsfeldern (Use Cases) genutzt werden. Dafür wurden alle angesprochenen Anwendungsfelder pro Studie mit den jeweils verwendeten Datenquellen verknüpft. Abbildung 4.8 zeigt die resultierenden Häufigkeiten (n) je Kombination.

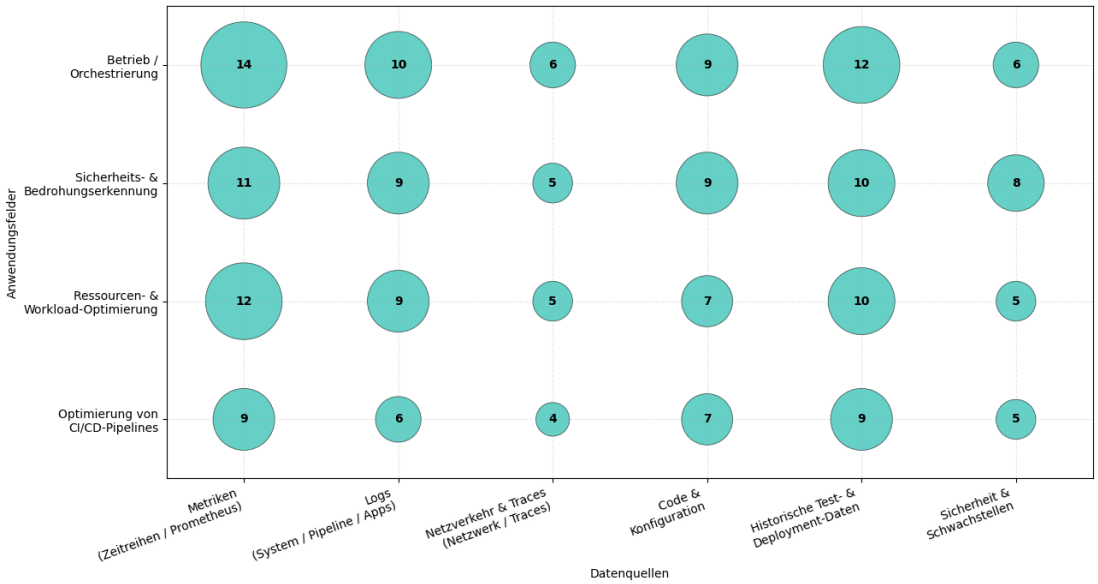


Abbildung 4.8: Korrelation zwischen Anwendungsfeldern und Datenquellen

Metriken (z. B. Zeitreihen aus Prometheus oder CloudWatch) beschreiben den quantitativen Systemzustand wie CPU, Speicher oder Latenzen. Entsprechend treten sie in nahezu allen Use Cases stark auf, besonders in Betrieb/Orchestrierung (n=14) sowie Ressourcen- und Workload-Optimierung (n=12), da diese Aufgaben auf kontinuierli-

chem Monitoring und wiederkehrenden Lastmustern basieren [14, 17].
Logs (System-, Pipeline- und Applikationslogs) sind ereignisbasierte, häufig unstrukturierte Daten und werden vor allem für Diagnose, Fehlerklassifikation und Ursachenanalyse genutzt. Im Mapping zeigen sie über nahezu alle Use Cases hinweg eine zentrale Rolle (typisch n=6-10), mit hohen Werten in Betrieb/Orchestrierung (n=10) und sicherheitsnahen Szenarien (n=9), da Betriebsstörungen, Fehlkonfigurationen oder Angriffsindikatoren meist zuerst in Logdaten sichtbar werden [28].
Netzwerkdaten und Traces ergänzen diese Sicht um Kommunikationsbeziehungen und Laufzeitpfade verteilter Systeme (z. B. Flow-Daten oder OpenTelemetry-Traces). Die Ausprägung ist insgesamt niedriger als bei Metriken und Logs, aber konsistent in Betrieb/Orchestrierung (n=6) sowie Sicherheits- und Bedrohungserkennung (n=5) vorhanden. Das deutet darauf hin, dass diese Daten vor allem dann relevant werden, wenn Ursachen nicht lokal erklärbar sind (z. B. in Microservice-Architekturen) oder wenn Anomalien über Kommunikationsmuster erkannt werden sollen [2].
Code und Konfiguration (Quellcode, Container-Artefakte, IaC-Definitionen) werden insbesondere dann wichtig, wenn KI nicht nur Symptome, sondern Änderungen und Konfigurationsursachen bewerten soll. Im Mapping zeigt sich eine breite Nutzung über alle Use Cases (n=7-9). Auffällig ist die hohe Ausprägung bei Betrieb/Orchestrierung und Security (je n=9), was zu typischen Plattformproblemen wie riskanten Änderungen, unsicheren Defaults oder Fehlkonfigurationen passt [15].
Historische Test- und Deployment-Daten (Build-Historien, Testergebnisse, Deployment-Verläufe) sind erwartungsgemäß stark vertreten, da viele Ansätze aus wiederkehrenden Mustern der Vergangenheit lernen. Besonders Betrieb/Orchestrierung (n=12) sowie CI/CD-Optimierung (n=9) und Ressourcenoptimierung (n=10) zeigen hohe Werte. Das ist plausibel, weil Stabilitäts- und Effizienzentscheidungen in der Praxis häufig auf Release-Historien, Fehlerraten und Zeitverläufen aufbauen [19].
Sicherheits- und Schwachstellendaten (z. B. Vulnerability-Scans, Threat-Feeds, Policy-Checks, Verhaltensmuster) konzentrieren sich erwartungsgemäß auf Sicherheits- und Bedrohungserkennung (n=8). Gleichzeitig sind sie auch im Betrieb/Orchestrierung sichtbar (n=6), was darauf hinweist, dass Security in Plattformumgebungen zunehmend als Betriebsaufgabe mit Observability- und Lebenszyklusdaten zusammenwächst [29].
Deutlich wird, dass der Use case Betrieb und Orchestrierung die stärksten Überschneidungen mit nahezu allen Datenquellen hat, weil hier Monitoring-Daten und Änderungsdaten zusammelaufen. Für die KI heißt das konkret, ohne diese Kombination entstehen eher Fehleralarme (Incidents), und die eigentliche Ursache lässt sich schlechter finden.

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 und damit eine nachvollziehbare Grundlage zu schaffen, um KI-Lösungen später hinsichtlich operativem Nutzen und Umsetzbarkeit einzuordnen.

4.3.1. Proaktives Ressourcen-Management

Das Muster Proaktives Ressourcen-Management beschreibt KI-gestützte Verfahren, um Infrastrukturressourcen vorausschauend und automatisiert an die erwartete Last anzupassen. Tabelle 4.1 fasst dieses Muster als konkrete Plattformaufgabe zusammen, bei der Instanztypen (EC2/RDS) oder Kubernetes-Pods (HPA) nicht reaktiv über statische Schwellwerte, sondern auf Basis prognostizierter CPU-/RAM-Spitzen gesteuert werden.

Tabelle 4.1: Übersicht zum Muster Proaktives Ressourcen-Management

Proaktives Ressourcen-Management	**Beschreibung**
Konkreter Plattform-Task	Automatisierte Anpassung von Instanztypen (EC2/RDS) oder Kubernetes-Pods (HPA) basierend auf der Vorhersage von CPU-/RAM-Spitzen.
Empfohlene KI-Methode	Deep Learning (z..B. LSTM) zur Zeitreihenprognose oder Reinforcement Learning zur dynamischen Lastverteilung in Echtzeit.
KPI/Mehrwert	Ressourcenauslastung bis zu +25% (RL vs. Heuristik) [17], Kosteneffizienz durch dynamische Allokation mit bis zu 55% geringeren Cloud-Kosten [26].
Limitationen	Hoher Rechenaufwand für das Modelltraining (z..B. LSTMs), Cold-Start-Latenzen in Serverless-Umgebungen, benötigt ca. 2 Monate historische Daten [1, 17].
Einsatzvoraussetzungen	Mind. 2 Monate historische Zeitreihen-Telemetrie (feingranular, z..B. 1-15 Minuten) aus CloudWatch/Prometheus sowie automatisierter API-Schreibzugriff für Instanz- bzw. Skalierungsanpassungen [17].

Als methodischer Kern werden in der Literatur insbesondere Zeitreihenmodelle (z.B. LSTM) sowie Reinforcement-Learning-Ansätze beschrieben, die Skalierungs- und Al-

lokationsentscheidungen dynamisch ableiten [17]. Der operative Nutzen zeigt sich in einer verbesserten Ressourcenauslastung gegenüber heuristischen Policies sowie in messbaren Kosteneffekten durch bedarfsgerechte Allokation [26]. Gleichzeitig ist die praktische Umsetzbarkeit an Voraussetzungen wie ausreichend historischer Telemetrie und automatisierten Eingriffsmöglichkeiten (API-Schreibzugriffe) gebunden. Zudem erhöhen Trainingsaufwand und Cold-Start-Effekte, insbesondere in serverlosen Umgebungen, die Implementierungs- und Betriebsanforderungen.

4.3.2. Automatisierte Release-Absicherung

Das Muster Automatisierte Release-Absicherung adressiert die Absicherung von Deployments, indem fehlerhafte Releases möglichst früh erkannt und automatisiert begrenzt werden. Tabelle 4.2 konkretisiert dieses Muster als Plattform-Task im Kontext von Canary-Deployments, bei denen neue Versionen schrittweise ausgerollt und anhand von Latenz- sowie Fehlerraten kontinuierlich überwacht werden.

Tabelle 4.2: Übersicht zum Muster Automatisierte Release-Absicherung

Automatisierte Release-Absicherung	**Beschreibung**
Konkreter Plattform-Task	Automatisiertes Rollback bei Canary-Deployments (schrittweise Ausrollung) durch Überwachung von Abweichungen in Latenz und Fehlerraten (Release-Gating).
Empfohlene KI-Methode	Unsupervised Learning (Autoencoder) zur Anomalieerkennung oder Reinforcement Learning (z.B. DQN) zur Optimierung des Rollback-Zeitpunkts [19].
KPI/Mehrwert	MTTR (Mean Time To Recovery) bis zu -40%; Systemverfügbarkeit bis zu +18%; Vorfallerkennung um bis zu 35% beschleunigt [14, 18, 19].
Limitationen	Risiko von False Positives (unnötige Rollbacks); zusätzlicher Validierungs- und Governance-Aufwand bei RL-Entscheidungen (z.B. Nachvollziehbarkeit/Erklärbarkeit).
Einsatzvoraussetzungen	Etablierte Baseline für Normalverhalten in Service-Mesh-gestützten Telemetriedaten (Traffic-Steuerung & Observability; Latenz, Fehlerraten, SLOs) sowie automatisierte Rollback-Mechanismen und Echtzeit-Streaming der Deployment-Metriken.

Methodisch werden dafür insbesondere Anomalieerkennungsverfahren (z.B. Autoencoder) genutzt, um Abweichungen vom etablierten Normalverhalten zu detektieren, während Reinforcement Learning (z.B. DQN) eingesetzt werden kann, um den Rollback-Zeitpunkt unter Unsicherheit zu optimieren [19]. Als operativer Mehrwert wird in den betrachteten Studien eine Reduktion der Mean Time To Recovery (MTTR) sowie ei-

ne höhere Systemverfügbarkeit berichtet, da fehlerhafte Releases schneller identifiziert und automatisiert zurückgenommen werden können [14, 18, 19]. Gleichzeitig zeigen sich in der Praxis Grenzen durch False Positives, die unnötige Rollbacks auslösen können, sowie durch zusätzlichen Validierungs- und Governance-Aufwand, insbesondere wenn RL-basierte Entscheidungen nachvollziehbar begründet werden müssen. Für eine robuste Umsetzung sind daher eine belastbare Baseline für Normalverhalten aus service-mesh-gestützten Telemetriedaten (Traffic-Steuerung und Observability) sowie automatisierte Rollback-Mechanismen und ein Echtzeit-Streaming der relevanten Deployment-Metriken erforderlich.

4.3.3. Intelligente Build-Fehlerdiagnose

Das Muster Intelligente Build-Fehlerdiagnose adressiert die Frage, wie CI-Pipelines Build-Fehlschläge frühzeitig erkennen und die Ursachenanalyse im Entwicklungsprozess beschleunigen können. Tabelle 4.3 fasst das Muster als Plattformaufgabe zusammen, bei der Build- und Testprotokolle aus der CI (z.B. Jenkins und Git) automatisiert ausgewertet werden, um typische Fehlermuster zu identifizieren und wiederkehrende Fehlerbilder zu gruppieren.

Tabelle 4.3: Übersicht zum Muster Intelligente Build-Fehlerdiagnose

Intelligente Build-Fehlerdiagnose	**Beschreibung**
Konkreter Plattform-Task	Früherkennung von Build-Fehlschlägen durch Analyse von CI-Logs (z.B. Jenkins/Git) sowie automatisierte Clusterung/Filterung von Log-Signaturen zur Ursachenanalyse.
Empfohlene KI-Methode	Supervised Learning (z.B. Random Forest, XGBoost) zur Klassifikation von Fehlerrisiken und Fehlertypen auf Basis historischer Build-Metadaten und Log-Features [14, 19].
KPI/Mehrwert	Deployment-Zeit bis zu -30%; Vorhersagegenauigkeit ca. 87%; Reduzierung der Fehlersuchzeit bis zu 40% [14, 19].
Limitationen	Class Imbalance (wenige Fehlersamples); Konzeptdrift bei Änderungen an Build-Pipelines/Dependencies erfordert kontinuierliches Monitoring und Retraining.
Einsatzvoraussetzungen	Zentralisierte und konsistent strukturierte Build- und Test-Logs sowie eine ausreichend große Historie gelabelter Daten (Erfolg vs. Fehlertyp) inklusive stabiler Referenz auf Commit, Pipeline-Stage und Artefaktversion.

Als methodische Grundlage dominieren SL Verfahren wie Random Forest und XGBoost, die aus historischen Build-Metadaten und Protokollmerkmalen sowohl das Ri-

siko eines Fehlschlags als auch häufig den zu erwartenden Fehlertyp ableiten [14, 29, 19]. Der operative Mehrwert liegt in verkürzten Rückkopplungszyklen, da problematische Änderungen früher auffallen und die Fehlersuche durch eine automatisierte Vorselektion relevanter Protokollauszüge reduziert wird, was sich in den Studien u.a. in kürzeren Bereitstellungszeiten und einer deutlichen Reduktion der Fehlersuchzeit widerspiegelt [14, 19]. Gleichzeitig ist die Umsetzung anfällig für unausgewogene Klassenverteilungen, da Fehlschläge in der Praxis typischerweise deutlich seltener als erfolgreiche Builds auftreten und Modelle dadurch verzerrt lernen können. Zudem führen Änderungen an Pipeline-Schritten, Abhängigkeiten oder Build-Umgebungen häufig zu veränderten Datenmustern, wodurch ein kontinuierliches Überwachen und regelmäßiges Nachtrainieren erforderlich wird. Voraussetzung für eine robuste Anwendung sind daher zentralisierte, konsistent strukturierte Protokolle sowie gelabelte Historien, die Builds zuverlässig mit Commit, Pipeline-Phase und Artefaktversion verknüpfen, sodass Fehlertypen eindeutig zugeordnet werden können.

4.3.4. DevSecOps (Security Ops)

Das Muster DevSecOps (Security Ops) beschreibt KI-gestützte Mechanismen, um Sicherheitsereignisse im Plattformbetrieb frühzeitig zu erkennen und Reaktionsmaßnahmen zu priorisieren. Tabelle 4.4 fasst das Muster als Plattformaufgabe zusammen, bei der sowohl die Priorisierung von Sicherheits-Scans als auch die Echtzeit-Erkennung von Angriffen auf Basis von Netzwerk- und Audit-Telemetrie adressiert wird.

Tabelle 4.4: Übersicht zum Muster DevSecOps (Security Ops)

DevSecOps (Security Ops)	**Beschreibung**
Konkreter Plattform-Task	Priorisierung von Sicherheits-Scans sowie Echtzeit-Erkennung von Angriffen (z.B. DDoS, Malware) anhand von Netzwerk- und Audit-Telemetrie.
Empfohlene KI-Methode	Deep Learning (z.B. CNNs auf flow-basierten Merkmalsrepräsentationen) oder Gradient Boosting (z.B. XGBoost) für unausgewogene Sicherheitsdaten [22].
KPI/Mehrwert	Erkennungsgenauigkeit bis ca. 95%; Reduktion von Fehlalarmen um ca. 20%; hohe Detektionsraten auch für neuartige Angriffsmuster (studienabhängig) [22].
Limitationen	Hoher Rechenbedarf (insbesondere bei Deep-Learning-Inferenz); Datenschutz- und Compliance-Risiken (DSGVO) bei der Analyse paketbasierter Protokolle und potenziell personenbezogener Daten.
Einsatzvoraussetzungen	Echtzeit-Erfassung von Netzwerk-Flows (z.B. NetFlow/sFlow) und Logdaten (z.B. Cloud-Audit-Logs) sowie Integration von Threat-Intelligence-Informationen (z.B. MITRE ATT&CK) zur Kontextualisierung und Priorisierung.

Als methodische Grundlage werden in der Literatur sowohl Deep-Learning-Ansätze (z.B. CNNs) als auch Gradient-Boosting-Verfahren (z.B. XGBoost) diskutiert, wobei letzteres insbesondere bei unausgewogenen Sicherheitsdaten robuste Ergebnisse liefern kann [29, 22]. Der operative Mehrwert besteht in einer verbesserten Detektionsleistung bei gleichzeitiger Reduktion von Fehlalarmen, wodurch Security-Teams weniger Zeit in manuelle Triage investieren müssen und kritische Ereignisse schneller eskaliert werden können [22]. Gleichzeitig ist die praktische Umsetzung an Grenzen gebunden, da Deep-Learning-Verfahren in der Inferenz einen erhöhten Rechenbedarf verursachen und damit zusätzliche Plattformkosten auslösen können. Darüber hinaus entstehen Datenschutz- und Compliance-Anforderungen, sobald paketbasierte Protokolle oder Logdaten potenziell personenbezogene Informationen enthalten und entsprechend nach DSGVO verarbeitet werden müssen. Für eine belastbare Anwendung sind daher eine Echtzeit-Erfassung von Netzwerk-Flows und relevanten Logdaten sowie eine Kontextualisierung über Threat-Intelligence-Informationen (z.B. MITRE ATT&CK) erforderlich, um Funde priorisieren und automatisierte Reaktionen kontrolliert auslösen zu können.

5

Anwendung der Literaturrecherche

In diesem Kapitel wird das in Kapitel 4 entwickelte theoretische Konzept zur Bewertung und Priorisierung von KI-Use-Cases im Cloud-Native Platform Engineering angewendet. Zunächst wird in Kapitel 5.1 ein Bewertungskonzept vorgstellt, das die Grundlage für die Einordnung und Analyse der identifizierten Use Cases bildet. Anschließend wird in Kapitel 5.2 die Bosch Digital Manufacturing Plattform (BMLP) analysiert, um einen praxisnahen Kontext für die Anwendung des Bewertungskonzepts zu schaffen. Dafür werden bekannte Pain-Points innerhalb der Plattform identifiziert. Abschließend wird in Kapitel 5.3 das zuvor entwickelte Bewertungskonzept konkret auf die identifizierten Pain-Points angwendet und es wird eine Handlungsempfehlung ausgesprochen.

5.1. Bewertungskonzept

Aufbauend auf den zuvor identifizierten KI-Use-Cases wird im Folgenden ein Bewertungskonzept vorgestellt, das eine strukturierte Einordnung und Vergleichbarkeit dieser Anwendungsfälle ermöglicht. Ziel des Frameworks ist es, KI-Anwendungsfälle im Cloud-Native Platform Engineering systematisch hinsichtlich ihres Implementierungsaufwands und ihres operativen Mehrwerts zu bewerten. Damit soll eine fundierte Entscheidungsgrundlage für die Priorisierung und Implementierung von KI-Lösungen geschaffen werden.
Implementierungsaufwand beschreibt dabei den Aufwand, eine KI-Funktion robust, sicher und wartbar produktiv zu integrieren und zu betreiben. Operativer Mehrwert beschreibt den erwarteten Beitrag zur messbaren Verbesserung zentraler Betriebsziele,

insbesondere der Betriebszuverlässigkeit (z. B. MTTR), der Effizienz sowie der Systemstabilität. Als weitere betriebliche Zielgrößen können Service Level Objectives (SLOs) herangezogen werden, die messbare Zielwerte für Servicequalität (z. B. Verfügbarkeit oder Latenz) definieren. Aspekte wie Datenzugang, Governance und organisatorische Voraussetzungen werden bewusst nicht in der X/Y-Achse abgebildet, sondern in zusätzlichen Dimensionen vertieft betrachtet.
Die erste Bewertungsdimension ist der Implementierungsaufwand, der auf der X-Achse der Bewertungsmatrix abgebildet wird. Tabelle 5.1 konkretisiert diese Dimension anhand der qualitativen Ausprägung von niedrig bis hoch. Berücksichtigt werden dabei insbesondere der Grad der erforderlichen Modellanpassungen, der Trainings- und Anpassungsaufwand sowie die Komplexität des operativen Betriebs. Darüber hinaus fließen Integrations- und Betriebsaspekte in die Einordnung ein, z. B. Anforderungen an Datenpipelines, Security-Mechanismen, Observability, Evaluation und laufende Qualitätssicherung. Ein niedriger Implementierungsaufwand beschreibt demnach den Einsatz bestehender KI-Funktionen oder Standardwerkzeuge mit geringem Integrations- und Betriebsaufwand. Ein hoher Aufwand hingegen ist typischerweise mit umfangreicher Integration und eigener Anpassung (z. B. RAG mit Datenpipeline und Evaluation oder Modelltraining) sowie kontinuierlicher Überwachung, Wartung und Qualitätssicherung verbunden.

Tabelle 5.1: X-Achse: Implementierungsaufwand

Ausprägung	Beschreibung
Niedrig	Einsatz vorhandener KI-Funktionen oder Standardtools ohne eigenes Modelltraining
Mittel	Anpassung bestehender Modelle, Feature Engineering, begrenztes Retraining
Hoch	Eigenes Modelltraining, kontinuierlicher Betrieb und Überwachung notwendig

Die zweite Dimension bildet der operative Mehrwert, der auf der Y-Achse dargestellt wird und in Tabelle 5.2 näher beschrieben ist. Hier wird der tatsächliche Beitrag eines KI-Use-Cases zur Verbesserung des Plattformbetriebs erfasst. Ein niedriger operativer Mehrwert umfasst vor allem unterstützende Funktionen mit begrenztem Einfluss auf zentrale Betriebskennzahlen. Ein hoher Mehrwert hingegen zeigt sich in deutlichen Verbesserungen wesentlicher KPIs und Betriebsziele, z. B. MTTR, SLO-Erfüllung, Error-Budget-Verbrauch, Infrastrukturkosten oder Systemstabilität.

Tabelle 5.2: Y-Achse: Operativer Mehrwert

Ausprägung	**Beschreibung**
Niedrig	Geringer Einfluss auf den Plattformbetrieb, primär unterstützende Funktionen wie Log-Analyse oder einfache Anomalieerkennung.
Mittel	Messbare Effizienzsteigerung oder Zeitersparnis im Plattformbetrieb durch KI-gestützte Automatisierung oder Optimierung.
Hoch	Deutliche Verbesserung zentraler KPIs (z.B. MTTR, Kostenreduktion, Stabilität)

Abbildung 5.1 führt beide Dimensionen in einer zweidimensionalen Bewertungsmatrix zusammen. Hierfür wurde die Matrix in vier Quadranten unterteilt, wodurch eine qualitative Einordnung der KI-Use-Cases in Bezug auf das Verhältnis von Implementierungsaufwand und operativem Mehrwert ermöglicht wird. Anwendungsfälle mit hohem Mehrwert und niedrigem Aufwand weisen ein günstiges Aufwand-Nutzen-Verhältnis auf und eignen sich als priorisierte Kandidaten für eine erste Umsetzung. Use Cases mit hohem Mehrwert und hohem Aufwand sind strategische Kandidaten, die typischerweise eine Roadmap, Vorarbeiten (z. B. Daten- und Integrationsgrundlagen) sowie ein risikobewusstes Vorgehen erfordern. Hingegen sollten Anwendungsfälle mit hohem Aufwand und geringem Mehrwert nur weiterverfolgt werden, wenn sie notwendige Vorstufen für höherwertige Use Cases darstellen oder klare Abhängigkeiten bestehen.

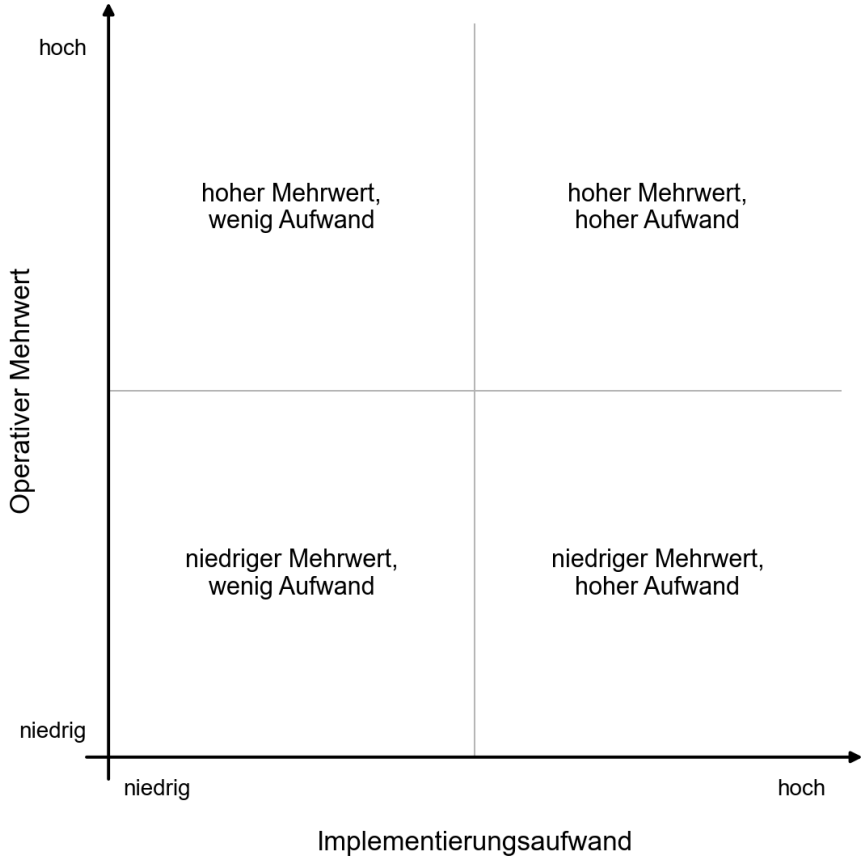


Abbildung 5.1: Bewertungsmatrix zur Einordnung von KI-Use-Cases im Cloud-Native Platform Engineering entlang von Implementierungsaufwand und operativem Mehrwert

Die zweidimensionale Bewertungsmatrix ermöglicht eine erste Priorisierung von KI-Use-Cases anhand des Verhältnisses von Implementierungsaufwand und operativem Mehrwertpotenzial. Sie dient als Ausgangspunkt für eine vertiefende Bewertung entlang zusätzlicher Faktoren, die in einer rein zweidimensionalen Darstellung nur unzureichend abbildbar sind. Die zusätzlichen Dimensionen erweitern die Matrix dabei nicht um weitere Achsen, sondern dienen der kontextbezogenen Bewertung der Realisierbarkeit und der betrieblichen Einbettung einzelner Use Cases.

Dimension 1: Umsetzbarkeit

Die erste zusätzliche Dimension beschreibt die technische und organisatorische Umsetzbarkeit eines KI-Use-Cases. Sie umfasst (i) die Verfügbarkeit und Qualität geeigneter Datenquellen, (ii) den technischen Fit mit der bestehenden Plattformarchitektur sowie die Integrationsfähigkeit und (iii) das erforderliche Know-how für Entwicklung, Betrieb und Wartung. Für die Bewertung wird eine einheitliche dreistufige Skala (niedrig/mittel/hoch) verwendet. Niedrig liegt vor, wenn Datenzugang und Integrationspfade vorhanden sind und das erforderliche Know-how im Team verfügbar ist. Hoch liegt vor,

wenn Datenlücken, fehlende Schnittstellen oder signifikanter Kompetenzaufbau erforderlich sind.

Dimension 2: Betriebswirksamkeit & Skalierbarkeit
Die zweite Dimension bewertet, ob sich der erwartete operative Mehrwert eines KI-Use-Cases im täglichen Plattformbetrieb zuverlässig realisieren lässt. Zusätzlich wird betrachtet, ob die Lösung über Services und Teams hinweg skalierbar ist. Bewertet werden (i) die Betriebswirksamkeit, z. B. Robustheit der Ergebnisse, Fehlalarme, Drift-Risiken und laufender operativer Aufwand, sowie (ii) die Skalierbarkeit in Bezug auf Wiederverwendbarkeit und Rollout-Fähigkeit. Für die Bewertung wird eine einheitliche dreistufige Skala (niedrig/mittel/hoch) verwendet. Niedrig beschreibt einen lokalen, stark kontextabhängigen oder schwer reproduzierbaren Effekt mit begrenzter Skalierbarkeit. Hoch beschreibt eine im Betrieb verlässlich wirksame Lösung mit breiter Übertragbarkeit über Services und Teams hinweg.

Dimension 3: Governance & Reifegrad
Die dritte Dimension berücksichtigt Governance-Anforderungen im Sinne von Compliance- und Security-Vorgaben. Bewertet werden (i) Anforderungen an Datenzugriff, Datenschutz und Sicherheitsmaßnahmen sowie (ii) der Reifegrad der jeweiligen Lösung (Konzeptnachweis, Pilotierung oder produktiv erprobt). Zur Einordnung dienen regulatorische Vorgaben, Sicherheitsanforderungen und der Nachweis operativer Stabilität. Niedrig liegt vor, wenn geringe Compliance-Hürden bestehen und die Lösung produktiv erprobt ist. Hoch liegt vor, wenn sensible Daten, strenge Vorgaben oder ein niedriger Reifegrad (z. B. Konzeptnachweis) vorliegen.

In Abschnitt 5.2 werden die identifizierten KI-Use-Cases zunächst in die Bewertungsmatrix eingeordnet (X/Y). Anschließend werden sie entlang der drei Zusatzdimensionen qualitativ bewertet, um Umsetzbarkeit, Betriebswirksamkeit und Governance-Aspekte kontextbezogen zu prüfen. Dadurch entsteht eine priorisierte und zugleich praxisnahe Grundlage, um identifizierte Pain Points der Plattform gezielt mit geeigneten KI-Use-Cases zu adressieren und daraus Handlungsempfehlungen abzuleiten.

5.2. Analyse der Bosch Digital Manufacturing Plattform

Die Bosch Digital Manufacturing Plattform (BMLP) ist eine modular aufgebaute, Cloud-Native-Plattform zur Unterstützung von Fertigungs- und Logistikprozessen. Anwendungen werden als lose gekoppelte Services bereitgestellt und in verteilten Umgebungen betrieben. Dazu zählen Edge-Standorte in den Werken, zentrale Rechenzentren sowie cloudbasierte Instanzen. Die Plattform ist auf einen hybriden Betrieb ausgelegt und adressiert Anforderungen an Latenz, Verfügbarkeit und Compliance gleichermaßen.
Cloud-Native ist die Plattform, da sie konsequent auf Container-Orchestrierung mit Kubernetes, deklarative Schnittstellen, automatisierte CI/CD-Prozesse sowie durchgängige Beobachtbarkeit setzt. Diese Eigenschaften bilden die technische Grundlage für eine hohe Änderungsfrequenz und einen standardisierten Plattformbetrieb. Gleichzeitig entstehen umfangreiche Betriebsdaten, die für KI-gestützte Verfahren im Plattformbetrieb nutzbar sind. Alle Informationen basierend auf interenen Plattformdokumentationen und Expertengesprächen (nicht öffentlich zugänglich).

5.2.1. Architektur und Betriebsmodell

Fachliche Funktionen in der BMLP werden als eigenständige deploybare Module umgesetzt, auf Basis der verteilten und modularen Grundstruktur. Jedes Modul bringt typischerweise eigene Logik und eigene Datenhaltung mit, wodurch direkte Abhängigkeiten zwischen Modulen vermieden werden. Die Zusammenarbeit zwischen Modulen erfolgt überwiegend ereignisbasiert, während synchrone Schnittstellen nur für spezielle Anwendungsfälle genutzt werden.
Für den werks- und organisationsübergreifenden Einsatz unterstützt die Plattform eine klare Trennung von Zugriffen und Verantwortlichkeiten. Diese wird durch zentrale Plattformdienste und ein gemeinsames Rollen- und Berechtigungsmodell umgesetzt. Dadurch können mehrere Organisationseinheiten dieselbe Plattform nutzen, ohne die Zugriffskontrolle zu verlieren.
Im Betrieb werden zentrale Mechanismen möglichst vereinheitlicht. Dazu zählen gemeinsame Einstiegspunkte für Anwendungen sowie einheitliche Informationen zum Betriebszustand der Module. Monitoring und Supportprozesse bauen auf diesen Grundlagen auf und sind über die Plattform hinweg vergleichbar organisiert.

5.2.2. Entwicklungsmodell und Plattformstandards

Die Entwicklung der Module erfolgt intern und ist über mehrere Repositories organisiert. Zentrale Entwicklungsschritte wie Build, Tests, Container-Erstellung und Sicherheitsprüfungen sind in standardisierten CI-Pipelines automatisiert. Die daraus entstehenden Artefakte werden versioniert abgelegt und für den weiteren Betrieb bereitgestellt.
Auch die Auslieferung folgt einem weitgehend automatisierten Ansatz. Ein wiederverwendbares Pipeline-Modell verbindet Artefaktverwaltung, Konfigurations- und Geheimnisverwaltung sowie deklarative Deployments zu einem durchgängigen Prozess. Ziel ist eine reproduzierbare und nachvollziehbare Bereitstellung, die zugleich regulatorische Anforderungen berücksichtigt.
Ergänzend definieren Plattformstandards zentrale Leitplanken für Entwicklung und Betrieb. Dazu zählen einheitliche Vorgaben für Container-Images und deklarative Infrastruktur. Schnittstellen werden nach gemeinsamen Gestaltungsregeln umgesetzt und konsistent dokumentiert, während die Kommunikation zwischen Modulen überwiegend ereignisbasiert erfolgt. Für den Betrieb werden einheitliche Logformate und durchgängige Korrelationskennungen genutzt, sodass Abläufe über mehrere Komponenten hinweg nachvollziehbar bleiben.

5.2.3. Operativer Stack und Datenbasis für AIOps

Für AIOps ist weniger die Tool-Landschaft entscheidend als die Daten, die im Plattformbetrieb entstehen. In der BMLP werden Betriebsdaten über Logs, Metriken, Traces sowie technische und fachliche Ereignisse erfasst und zentral nutzbar gemacht. Logs liegen in einem einheitlichen Format vor und enthalten neben Zeitstempel und Systemkontext auch Versions- und Instanzinformationen sowie Korrelationskennungen. Damit lassen sich Abläufe über mehrere Komponenten hinweg zusammenführen und automatisiert auswerten.
Metriken und Traces ergänzen diese Sicht um messbare Größen wie Last, Latenz und Fehlerraten sowie um Ablaufspuren über Servicegrenzen hinweg. Ereignisdaten bilden Zustandswechsel und Prozessketten ab und unterstützen dadurch die Einordnung von Störungen. Zusätzlich entstehen Daten aus Build- und Rollout-Prozessen, etwa Testergebnisse, Sicherheitsprüfungen und Rollout-Verläufe, die Rückschlüsse auf Änderungsrisiken erlauben. Auch Änderungen an Konfigurationen können Hinweise auf Abweichungen zwischen geplantem und aktuellem Zustand liefern.
Auf dieser Datenbasis lassen sich insbesondere Auffälligkeiten frühzeitig erkennen und Störungen schneller eingrenzen. Ebenso können Rollouts datenbasiert überwacht und Kapazitätsbedarfe besser abgeschätzt werden.

5.3. Anwendung des Bewertungskonzepts und Handlungsempfehlungen

An Bosch gerichtet

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.

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

Abbildungsverzeichnis

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. 11. 2025).

[4] *GitOps in 2025: From Old-School Updates to the Modern Way*. https://www.cncf.io/blog/2025/06/09/g in-2025-from-old-school-updates-to-the-modern-way/. Juni 2025. (Besucht am 30. 11. 2025).

[5] juliakm. *What Is Platform Engineering?* https://learn.microsoft.com/en-us/platform-engineering/what-is-platform-engineering. (Besucht am 15. 12. 2025).

[6] *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).

[7] Atlassian. *Was ist DevOps?* https://www.atlassian.com/de/devops. (Besucht am 16. 12. 2025).

[8] *Was ist CI/CD? — Automatisierung in der Softwareentwicklung*. https://www.redhat.com/de/topics/de is-ci-cd. (Besucht am 16. 12. 2025).

[9] 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).

[10] 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. (Besucht am 03. 11. 2025).

[11] 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. (Besucht am 27. 11. 2025).

[12] 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. (Besucht am 27. 11. 2025).

[13] Majid Ghasemi u. a. *An Introduction to Reinforcement Learning: Fundamental Concepts and Practical Applications*. Aug. 2024. DOI: 10.48550/arXiv.2408.07712.

[14] Giridhar Kankanala und Sudheer Amgothu. “AI/ML – DevOps Automation”. In: 13 (Okt. 2024), S. 111–117.

[15] 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/ijsra.2024.13.1.1838. (Besucht am 03. 11. 2025).

[16] 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. (Besucht am 03. 11. 2025).

[17] 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.

[18] 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. (Besucht am 02. 12. 2025).

[19] 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 ().

[20] 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. (Besucht am 03. 11. 2025).

[21] 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. arXiv: 2401.12230 [cs]. (Besucht am 12. 11. 2025).

[22] 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. (Besucht am 04. 11. 2025).

[23] 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. (Besucht am 06. 11. 2025).

[24] 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).

[25] 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. (Besucht am 06. 11. 2025).

[26] 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. (Besucht am 05. 11. 2025).

[27] 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.

[28] 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. (Besucht am 03. 11. 2025).

[29] 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.

[30] 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).

[31] Yu Jeffrey Hu, Jeroen Rombouts und Ines Wilms. *MLOps Monitoring at Scale for Digital Platforms*. Apr. 2025. DOI: 10.48550/arXiv.2504.16789. arXiv: 2504.16789 [econ]. (Besucht am 18. 11. 2025).

[32] 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.

[33] 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).

[34] 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.