

Big Data: DBA64
Graded Assignment Part 2: Theoretical Part

Assignment

eingereicht bei

Dr. Martin Prause
AKAD

von

Justin Stange-Heiduk
Hengstrücken 132
37520 Osterode am Harz
Telefon: 015233817587
Studiengang: Data Science
3. Fachsemester
Matrikelnummer: 8149363
Datum: 29.10.2024

1.	Einleitung.....	1
2.	Einführung in MLDevOps.....	2
2.1	Unterschied zwischen MLDevOps und traditionellem DevOps	2
2.2	Bedeutung von MLDevOps für moderne Unternehmen.....	4
3.	Überblick über den MLDevOps-Prozess	5
3.1	Schritte im MLDevOps	5
3.2	Gesamtprozess: Wie die Schritte nahtlos ineinandergreifen.....	7
4.	MLDevOps mit Azure Komponenten.....	9
4.1	Data (ML)	9
4.1.1	Herausforderungen und Risiken.....	9
4.2	Model (ML)	10
4.2.1	Herausforderungen und Risiken.....	10
4.3	Verify (Übergang ML → Dev).....	11
4.3.1	Herausforderungen und Risiken.....	11
4.4	Package (Dev)	12
4.5	Release (Übergang Dev → Ops)	12
4.5.1	Herausforderungen und Risiken.....	13
4.6	Configure (Ops).....	13
4.6.1	Herausforderungen und Risiken.....	13
4.7	Monitor (Ops).....	14
4.7.1	Herausforderungen und Risiken.....	14
4.8	Plan (Übergang Ops → Dev)	14
4.8.1	Herausforderungen und Risiken.....	15
4.9	Create (Übergang Dev → ML)	15
4.9.1	Herausforderungen und Risiken.....	15
5.	Zusammenfassung.....	17
	Literaturverzeichnis	18

1. Einleitung

Der Einsatz von Machine Learning in Unternehmen wird zunehmend unverzichtbar, da datengetriebene Ansätze Wettbewerbsvorteile schaffen und die Effizienz steigern können. Um jedoch Machine-Learning-Modelle erfolgreich in die Produktionsumgebung zu integrieren und kontinuierlich zu verbessern, bedarf es spezifischer Entwicklungs- und Betriebsprozesse. Hier setzt MLDevOps an, ein Ansatz, der klassische DevOps-Praktiken erweitert und speziell auf die Bedürfnisse der ML-Entwicklung zugeschnitten ist. (Wodecki, 2023, S. 1471f)

Während DevOps-Prozesse auf Softwareentwicklung ausgelegt sind, ist MLDevOps auf die besonderen Herausforderungen des maschinellen Lernens fokussiert. Hierzu zählen Aspekte wie die kontinuierliche Anpassung und Retrainings der Modelle, die Einbindung großer und oft unstrukturierter Datenmengen sowie die Notwendigkeit, Modelle in Echtzeit zu überwachen und anzupassen. Trotz dieser Anforderungen existieren noch keine weit verbreiteten Standards oder Best Practices zur strukturierten Integration von ML-Modellen in den Produktionsprozess, was viele Unternehmen vor große Herausforderungen stellt.

Ziel der vorliegenden Arbeit ist es, einen vollständigen Überblick über den MLDevOps-Prozess zu geben und für jede Phase spezifische Azure-Tools zu identifizieren, die zur Umsetzung und Optimierung beitragen. Dabei werden für jede Phase sowohl die relevanten Komponenten als auch die Herausforderungen und Risiken beschrieben.

Zunächst wird eine Einführung in MLDevOps gegeben und die Unterschiede zum traditionellen DevOps herausgearbeitet. Danach folgt eine detaillierte Beschreibung des MLDevOps-Prozesses, in der die einzelnen Schritte vorgestellt werden. Anschließend wird die Umsetzung des MLDevOps-Prozesses mithilfe spezifischer Azure-Komponenten erläutert, wobei jede Phase – von der Datenerfassung über die Modellbereitstellung bis zur Überwachung – systematisch analysiert wird. Abschließend werden die Ergebnisse zusammengefasst, und es erfolgt ein Ausblick auf zukünftige Entwicklungen im Bereich MLDevOps.

2. Einführung in MLDevOps

MLDevOps stellt eine Weiterentwicklung traditioneller DevOps-Praktiken dar, die speziell auf die Bedürfnisse des Machine Learning (ML) zugeschnitten ist. Während DevOps primär die Automatisierung und Effizienz in der Softwareentwicklung fokussiert, erweitert MLDevOps diesen Ansatz um spezifische Anforderungen des ML-Lebenszyklus, wie die Verarbeitung großer Datenmengen, das Training komplexer Modelle und die Überwachung der Modellleistung im Einsatz. In diesem Kapitel werden die wesentlichen Unterschiede zwischen MLDevOps und herkömmlichem DevOps erläutert. (Islam, 2022)

2.1 Unterschied zwischen MLDevOps und traditionellem DevOps

MLDevOps und traditionelles DevOps haben beide das Ziel, Entwicklungsprozesse effizienter und skalierbarer zu gestalten, jedoch unterscheiden sie sich grundlegend in ihrer Anwendung und den spezifischen Herausforderungen, die sie adressieren. Während DevOps ursprünglich entwickelt wurde, um die Zusammenarbeit zwischen Entwicklungs- und Betriebsteams zu verbessern und Software schneller und zuverlässiger bereitzustellen, erweitert MLDevOps diesen Ansatz um die spezifischen Anforderungen des Machine Learning. (Islam, 2022)

Um die Unterschiede zwischen MLDevOps und traditionellem DevOps besser zu verstehen, werden im Folgenden die zentralen Aspekte beider Ansätze verglichen und gegenübergestellt (Islam, 2022):

- Pipeline-Komplexität

DevOps: Im traditionellen DevOps-Prozess umfasst die Pipeline die Schritte von der Code-Entwicklung, über das Testen bis zur Bereitstellung und dem Monitoring von Softwareanwendungen. Diese Pipelines sind oft linear und können gut automatisiert werden, um eine kontinuierliche Integration (CI) und kontinuierliche Bereitstellung (CD) zu ermöglichen.

MLDevOps: Im Gegensatz dazu erfordert MLDevOps eine wesentlich komplexere Pipeline, die neben der Softwareentwicklung auch die Phasen der Datenbeschaffung, Datenvorbereitung, Modellentwicklung, Training, Validierung und fortlaufenden Modellüberwachung umfasst. Diese zusätzlichen Schritte führen zu einer verzweigten und iterativen Pipeline, da Modelle kontinuierlich mit neuen Daten aktualisiert und angepasst werden müssen.

- Datenzentrierte Prozesse

DevOps: Der traditionelle DevOps-Ansatz konzentriert sich primär auf den Code und seine Integration in die Produktionsumgebung. Daten spielen eine untergeordnete Rolle und werden meist nur als Konfigurations- oder Testdaten betrachtet.

MLDevOps: Im Gegensatz dazu sind Daten der zentrale Bestandteil des gesamten Prozesses. Die Qualität und Verfügbarkeit der Daten bestimmen maßgeblich den Erfolg eines ML-Projekts. Der Prozess erfordert nicht nur eine konstante Datenverfügbarkeit, sondern auch Techniken zur Datenbereinigung, Transformation und Skalierung, um die Modelle zu trainieren und zu evaluieren.

- Automatisierung und Iteration

DevOps: Automatisierung in DevOps bezieht sich vor allem auf die kontinuierliche Integration und Bereitstellung von Code. Dies ermöglicht schnellere Releases und kontinuierliche Updates. Sobald die Anwendung produktionsreif ist, gibt es in der Regel weniger Notwendigkeit für Änderungen am Code.

MLDevOps: Automatisierung ist auch in MLDevOps entscheidend, jedoch betrifft sie hier zusätzlich die kontinuierliche Datenaufbereitung, das Modelltraining und die Anpassung. Da ML-Modelle auf neuen Daten oder aktualisierten Datensätzen basieren, müssen sie regelmäßig neu trainiert werden, was eine wiederholte, automatische Verarbeitung erforderlich macht. Die Iteration ist somit ein ständiger Prozess, der sicherstellt, dass Modelle aktuell und genau bleiben.

- Versionierung von Artefakten

DevOps: In einem DevOps-Prozess werden hauptsächlich Code-Versionen und Konfigurationen verwaltet. Änderungen am Code können leicht nachverfolgt und bei Bedarf zurückgesetzt werden.

MLDevOps: In MLDevOps müssen neben dem Code auch Daten und Modelle versioniert werden. Dies bedeutet, dass nicht nur unterschiedliche Versionen eines Modells, sondern auch die Daten, auf denen es trainiert wurde, dokumentiert und gespeichert werden müssen. Dies ermöglicht es, die Modellleistung zu reproduzieren und sicherzustellen, dass bei Änderungen oder Updates Rückschlüsse auf die Datenbasis gezogen werden können.

- Monitoring und Wartung

DevOps: Nach der Bereitstellung konzentriert sich das Monitoring auf die Verfügbarkeit, Leistung und Fehler der Anwendung. Änderungen werden nur dann vorgenommen, wenn es Probleme gibt oder neue Funktionen erforderlich sind.

MLDevOps: Im ML-Bereich ist das Monitoring komplexer. Es umfasst nicht nur die Überwachung der Systemleistung, sondern auch die kontinuierliche Kontrolle der Modellleistung, insbesondere die Analyse der Modellgenauigkeit und der Datenverschiebung (Data Drift). Modelle können im Laufe der Zeit an Genauigkeit verlieren, wenn sich die zugrunde liegenden Daten ändern, weshalb eine regelmäßige Neubewertung und Anpassung erforderlich ist.

- Technologische Integration

DevOps: Die Werkzeuge, die in traditionellen DevOps-Prozessen verwendet werden, sind auf die kontinuierliche Integration und Bereitstellung von Software ausgelegt, wie Jenkins, Docker und Kubernetes.

MLDevOps: Für MLDevOps werden zusätzlich spezialisierte Tools benötigt, die den gesamten ML-Lebenszyklus unterstützen. Azure ML Studio bietet beispielsweise integrierte Lösungen für Datenaufbereitung, Modellentwicklung, Training, Deployment und Monitoring, wodurch es Teams ermöglicht wird, komplexe ML-Workflows effizient zu managen. Die nahtlose Integration dieser Werkzeuge ist entscheidend für den Erfolg des MLDevOps-Prozesses.

2.2 Bedeutung von MLDevOps für moderne Unternehmen

Die zunehmende Bedeutung von MLDevOps für moderne Unternehmen spiegelt sich in der wachsenden Notwendigkeit wider, Machine Learning-Modelle effizient und zuverlässig zu entwickeln, zu implementieren und zu betreiben. MLDevOps vereinfacht diesen Prozess, indem es die spezifischen Herausforderungen adressiert, die bei der Integration und Verwaltung von Machine Learning in Produktionsumgebungen auftreten. (Islam, 2022)

Mit der wachsenden Integration von Machine Learning in reale Anwendungen bietet MLDevOps einen strukturierten Ansatz, um die Herausforderungen der Skalierbarkeit, Zuverlässigkeit und Wartbarkeit von ML-Modellen zu bewältigen. Es ermöglicht eine effiziente Verwaltung des gesamten Lebenszyklus und optimiert die Entwicklung und Bereitstellung von KI-Anwendungen, unabhängig davon, ob sie im Cloud- oder Edge-Computing ausgeführt werden. (Alberti, Alvarez-Napagao, Anaya et al., 2024)

Zusätzlich zeigt sich, dass die Komplexität der Entwicklung und Wartung produktionsreifer intelligenter Dienste durch MLDevOps-Methoden erheblich vereinfacht werden kann. Lösungen zur Automatisierung der Modellwartung, wie sie von Google Vertex AI, Amazon SageMaker und Microsoft Azure angeboten werden, ermöglichen eine kontinuierliche Überwachung und Diagnose von Modellen. Dies hilft Unternehmen, Modellverschlechterungen frühzeitig zu erkennen und zu beheben. Darüber hinaus spielen fortschrittliche Techniken wie CausalML und Reinforcement Learning eine wesentliche Rolle, da sie eine tiefere Analyse der Modellleistung erlauben und selbstoptimierende Systeme unterstützen, die experimentelle Optimierungen und kausale Zusammenhänge berücksichtigen. (Wodecki, 2023, S.1471f)

3. Überblick über den MLDevOps-Prozess

Der MLDevOps-Prozess umfasst eine Reihe von ineinandergreifenden Phasen, die sicherstellen, dass Machine Learning-Modelle effizient entwickelt, implementiert und kontinuierlich überwacht werden können. Im Gegensatz zu herkömmlichen Software-Entwicklungsprozessen erfordert MLDevOps zusätzliche Schritte, um Daten zu verarbeiten und Modelle anzupassen, wodurch ein ganzheitlicher Ansatz entsteht, der von der Datenerfassung bis zur laufenden Modellüberwachung reicht. In diesem Kapitel wird der MLDevOps-Prozess anhand eines Diagramms erläutert, das die wesentlichen Schritte beschreibt und zeigt, wie sie miteinander verbunden sind.

3.1 Schritte im MLDevOps

Der folgende Überblick zeigt die einzelnen Schritte im MLDevOps-Prozess in Form eines Diagramms, das den Fluss und die Abhängigkeiten zwischen den Phasen verdeutlicht. Dieses Bild dient als Referenz, um die Struktur und den Ablauf des gesamten Prozesses besser zu verstehen.

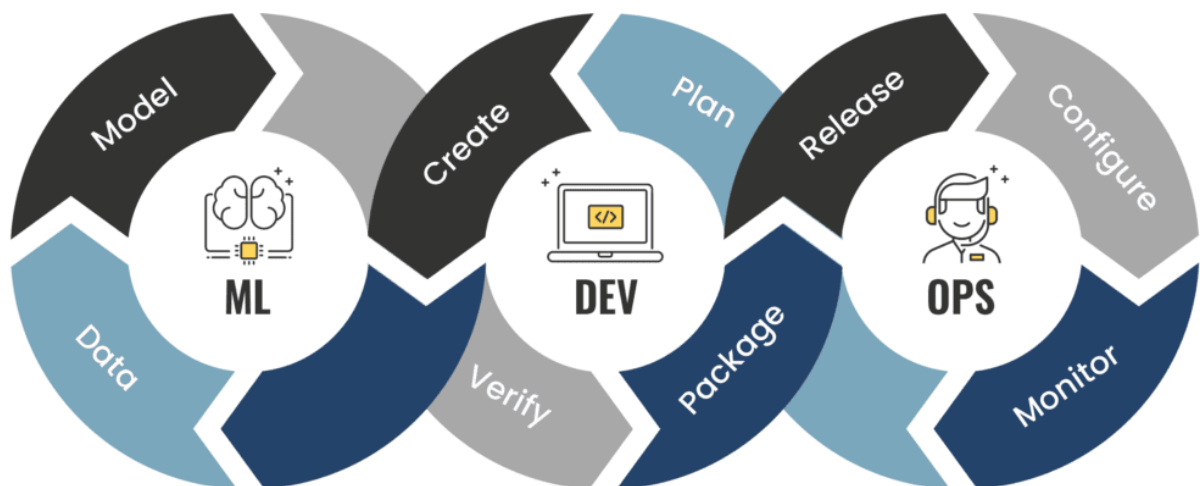


Abbildung 1: MLDevOps Zyklus Quelle: (Islam, 2022)

Der folgende Überblick zeigt die einzelnen Schritte im MLDevOps-Prozess in Form eines Diagramms, das den Fluss und die Abhängigkeiten zwischen den Phasen verdeutlicht. Außerdem lässt sich der MLDevOps-Prozess in verschiedene Phasen unterteilen, die den Bereichen ML (Machine Learning), Dev (Development) und Ops (Operations) zugeordnet sind. Diese Abbildung dient als Referenz, um die Struktur und den Ablauf des gesamten Prozesses besser zu verstehen (Islam, 2022) (Alberti, Alvarez-Napagao, Anaya et al., 2024):

- Data (ML)

Der Prozess beginnt mit der Erfassung, Bereinigung und Vorbereitung der Daten, die für das Training von Machine Learning-Modellen verwendet werden. Dies umfasst die Datenakquisition, -integration und -transformation, um sicherzustellen, dass die Daten für die Analyse geeignet sind. Eine sorgfältige Datenvorbereitung ist entscheidend, um die Grundlage für die Modellentwicklung zu schaffen.

Ziel: Sicherstellen, dass die Daten vollständig, konsistent und repräsentativ sind, um eine solide Grundlage für die Modellentwicklung zu schaffen.

- Model (ML)

In dieser Phase werden die Modelle entwickelt, trainiert und optimiert. Der Fokus liegt auf der Auswahl geeigneter Algorithmen, dem Splitten der Daten in Trainings- und Validierungssätze sowie dem iterativen Training und Feintuning der Modelle.

Ziel: Erstellen eines präzisen und robusten Modells, das die spezifische Aufgabe effektiv erfüllt, sei es Klassifikation, Regression oder eine andere Art von Vorhersage.

- Verify (Übergang ML → Dev)

Nach der Modellentwicklung folgt die Phase der Validierung und Evaluierung. Hier wird sichergestellt, dass die Modelle und Anwendungen korrekt und wie erwartet funktionieren. Hierbei werden sowohl die Modellleistung als auch die Stabilität des Codes geprüft. Diese Phase markiert den Übergang, da die entwickelten ML-Modelle nun in die nächste Stufe der Softwareentwicklung übergehen, wo sie nahtlos in Anwendungen integriert und weiter verfeinert werden.

Ziel: Sicherstellen, dass Modelle und Anwendungen fehlerfrei sind und zuverlässig in der Produktionsumgebung laufen.

- Package (Übergang Dev → Ops)

In dieser Phase werden die validierten Modelle und ihre zugehörigen Anwendungen für die Produktion verpackt. Dies umfasst das Erstellen von Containern oder Paketen, die die benötigten Abhängigkeiten beinhalten und so eine einfache Bereitstellung ermöglichen. Diese Phase stellt einen Übergang dar, da die Entwicklungsarbeit abgeschlossen ist und die Modelle nun für den Einsatz in der operativen Produktionsumgebung vorbereitet werden, wo sie effektiv eingesetzt und skaliert werden können.

Ziel: Schaffung eines transportfähigen und wiederverwendbaren Pakets, das leicht in verschiedenen Umgebungen bereitgestellt werden kann.

- Release (Übergang Dev → Ops)

Die freigegebenen Pakete werden in die Produktionsumgebung überführt. Dieser Schritt markiert den Übergang von der Entwicklungs- zur Betriebsphase. Dabei werden die Modelle auf geeigneter Infrastruktur implementiert, wobei Skalierbarkeit, Zuverlässigkeit und Sicherheit berücksichtigt werden. Diese Phase stellt einen Übergang dar, da die in der Entwicklung erstellten und verpackten Modelle nun in den operativen Betrieb überführt werden, wo sie unter realen Bedingungen laufen und genutzt werden können.

Ziel: Sicherstellen, dass die bereitgestellten Modelle und Anwendungen zuverlässig und skalierbar in der Produktionsumgebung laufen.

- Configure (Ops)

In dieser Phase wird die Produktionsumgebung so konfiguriert, dass die Modelle unter optimalen Bedingungen laufen. Dies kann die Einrichtung von Hardware, Software und

Netzwerkeinstellungen umfassen, um die Leistung zu maximieren und den reibungslosen Betrieb zu gewährleisten.

Ziel: Bereitstellung einer Umgebung, die die bestmögliche Leistung und Zuverlässigkeit der Modelle sicherstellt.

- Monitor (Ops)

Die kontinuierliche Überwachung der Modellleistung ist entscheidend, um sicherzustellen, dass sie wie erwartet funktioniert. Hierbei werden wichtige Leistungskennzahlen verfolgt, und es wird auf potenzielle Probleme wie Modelldrift reagiert, um die Genauigkeit und Effizienz des Modells zu erhalten. Zudem kann das Modell bei Bedarf retrainiert oder angepasst werden.

Ziel: Frühzeitige Erkennung von Problemen und kontinuierliche Optimierung der Modelle, um sicherzustellen, dass sie im Laufe der Zeit genau und effizient bleiben.

- Plan (Übergang Ops → Dev)

Nach der Überwachung und Bewertung werden Strategien entwickelt, um Modelle zu aktualisieren oder neue Modelle zu entwickeln. Dies umfasst auch die Planung von Ressourcen und die Definition von Zielen, um den gesamten MLDevOps-Prozess zu verbessern. Diese Phase verbindet die operativen Erkenntnisse mit der Entwicklungsplanung. Dies stellt einen Übergang dar, da die in der Betriebsphase gewonnenen Erkenntnisse und Daten genutzt werden, um Entwicklungsentscheidungen zu treffen und zukünftige Verbesserungen und Iterationen zu planen.

Ziel: Strukturierte Organisation und Planung der Projektabläufe, um sicherzustellen, dass zukünftige Entwicklungszyklen effizient und im Rahmen des Budgets umgesetzt werden.

- Create (Übergang Dev → ML)

Die erstellten Modelle werden in Anwendungen integriert, und es wird der Code entwickelt, der benötigt wird, um das Modell in einer produktiven Umgebung zu betreiben. Dies umfasst die Entwicklung von Schnittstellen und Logiken, die das Modell verwenden. Dieser Schritt schließt den Kreislauf, indem er zur Verbesserung der Modelle führt und neue Iterationen vorbereitet. Diese Phase markiert einen Übergang, da die entwickelten Lösungen in den Machine Learning-Prozess zurückgeführt werden, um die Modelle weiter zu verfeinern und zu trainieren, basierend auf neuen Anforderungen und Erkenntnissen.

Ziel: Vorbereitung der Softwareumgebung, in der das Modell eingesetzt wird, um eine reibungslose Integration in bestehende Systeme sicherzustellen und den Weg für neue Trainingszyklen zu ebnen.

3.2 Gesamtprozess: Wie die Schritte nahtlos ineinandergreifen

Der MLDevOps-Prozess ist darauf ausgelegt, die einzelnen Phasen eines Machine Learning-Projekts zu einem nahtlosen Workflow zu verbinden, der die Entwicklung, Implementierung und kontinuierliche Optimierung von ML-Modellen unterstützt. Es handelt es sich dabei nicht nur um isolierte Schritte, sondern um einen ganzheitlichen Prozess, der es ermöglicht, Modelle effizienter zu entwickeln und schneller auf Veränderungen zu reagieren.

Ein Schlüsselmerkmal des MLDevOps-Prozesses ist die enge Integration der verschiedenen Phasen, die von der Datenerfassung über die Modellentwicklung bis hin zur kontinuierlichen Überwachung reichen. Diese Integration ermöglicht eine schnelle und automatisierte Rückkopplungsschleife, bei der Modelle nicht nur einmalig entwickelt, sondern regelmäßig aktualisiert und optimiert werden. Dadurch wird die gesamte Wertschöpfungskette des Machine Learning automatisiert und skaliert, was insbesondere für große und komplexe Systeme entscheidend ist. (Hegeman et al., 2021, S.57f)

Ein weiterer Vorteil des Gesamtprozesses ist die Verwendung eines einheitlichen Ausführungsmodells, das die unterschiedlichen Datenverarbeitungsframeworks innerhalb eines ML-Workflows zusammenführt. Dies erleichtert die Identifikation von Engpässen und ermöglicht eine reibungslose Orchestrierung der verschiedenen Aufgaben, ohne dass eine spezielle Anpassung für jedes Framework notwendig ist. (Hegeman et al., 2021, S.57f)

Durch diese eng verzahnten Prozesse wird es möglich, Modelle effizient zu überwachen und bei Bedarf schnell Anpassungen vorzunehmen. Dies ist besonders wichtig in Umgebungen, in denen Daten sich schnell ändern oder kontinuierlich neue Informationen verarbeitet werden müssen. Somit bietet der MLDevOps-Ansatz einen klaren Vorteil gegenüber traditionellen, statischen Entwicklungsmodellen, indem er auf Skalierbarkeit und Flexibilität setzt, die notwendig sind, um moderne AI-Anwendungen effektiv zu betreiben.

4. MLDevOps mit Azure Komponenten

Der MLDevOps-Prozess erfordert eine nahtlose Integration verschiedener Phasen, die von der Datenaufbereitung bis zur kontinuierlichen Überwachung reichen. Um diesen Prozess effizient zu gestalten, stellt Azure ML Studio eine umfassende Suite von Tools zur Verfügung, die jede Phase des MLDevOps unterstützt. In diesem Kapitel wird erläutert, wie die einzelnen Schritte des MLDevOps-Prozesses mithilfe der Komponenten von Azure insbesondere Azure ML umgesetzt werden. Dabei wird aufgezeigt, wie Azure ML die Automatisierung, Skalierbarkeit und Effizienz in jeder Phase fördert, und es werden die spezifischen Herausforderungen und Risiken besprochen, die bei der Anwendung dieser Technologien auftreten können.

4.1 Data (ML)

Im MLDevOps-Prozess ist die Datenvorbereitung ein kritischer erster Schritt, der sicherstellt, dass die verwendeten Daten vollständig, konsistent und repräsentativ sind. Azure bietet mehrere Tools zur Unterstützung dieses Prozesses, darunter Azure Data Factory, Azure Data Prep, und Dataflow. Diese Tools ermöglichen es, Daten aus verschiedenen Quellen zu integrieren, zu transformieren und für die Modellentwicklung bereitzustellen.

Azure Data Factory ist ein Dienst, der die Integration, Transformation und das Verschieben von Daten aus unterschiedlichen Quellen automatisiert. Er unterstützt die Erstellung und Verwaltung von Datenpipelines und die nahtlose Integration in Big-Data-Plattformen. Dies ermöglicht es Unternehmen, große Datenmengen effizient zu verarbeiten und die Datenaufbereitung zu skalieren. (Jonburchel, 2024)

Azure Data Prep hilft bei der automatisierten Bereinigung, Transformation und Profilierung von Daten. Das Tool verbessert die Qualität der Eingangsdaten, indem es Fehler und Inkonsistenzen erkennt und behebt. Es wird häufig in Verbindung mit Azure ML Studio genutzt, um sicherzustellen, dass die Daten für das Training von Modellen geeignet. (MayMSFT, 2024)

Dataflows in Azure Data Factory bieten eine grafische Benutzeroberfläche für die Erstellung von ETL-Prozessen (Extrahieren, Transformieren, Laden). Dies ermöglicht eine visuelle Gestaltung von Datenmappings und vereinfacht komplexe Transformationsprozesse, ohne dass umfangreicher Code erforderlich ist. Diese Funktion ist besonders nützlich, um schnell und effizient automatisierte Pipelines zu erstellen. (Kromerm, 2024)

4.1.1 Herausforderungen und Risiken

Datenqualität: Ein häufiger Stolperstein in der Datenvorbereitung ist die Sicherstellung, dass die Daten vollständig und fehlerfrei sind. Inkonsistenzen oder fehlende Daten können die Modellleistung erheblich beeinträchtigen (Kromer, 2022, S.54).

Datenverfügbarkeit: Der Zugriff auf aktuelle und relevante Datenquellen ist entscheidend für die Erstellung verlässlicher Modelle. Schwierigkeiten beim Datenzugriff können den gesamten Prozess verlangsamen. (Kulkarni et al., 2024, S.275ff)

Datenschutz: Die Einhaltung von Datenschutzrichtlinien ist eine Herausforderung, insbesondere wenn sensible Daten verarbeitet werden. Automatisierte Tools wie Azure Data Factory helfen, Compliance zu gewährleisten, erfordern jedoch sorgfältige Konfiguration und Überwachung. (Coté et al., 2018, S.248)

4.2 Model (ML)

Die Modellentwicklung ist ein zentraler Bestandteil des MLDevOps-Prozesses, bei dem Machine Learning-Modelle entwickelt, trainiert und optimiert werden, um spezifische Aufgaben zu erfüllen. Azure ML Studio bietet mehrere leistungsstarke Tools, um diesen Prozess zu unterstützen, darunter Automated ML und Azure ML Designer. Diese Tools erleichtern die Auswahl von Algorithmen, die Feinabstimmung von Hyperparametern und die Automatisierung von Experimenten, was die Entwicklung effizienter und skalierbarer ML-Modelle ermöglicht.

Automated ML automatisiert den Prozess der Modellentwicklung, indem es verschiedene Algorithmen und Hyperparameter-Konfigurationen testet, um die besten Kombinationen zu finden. Dies beschleunigt den Entwicklungsprozess erheblich und reduziert den manuellen Aufwand, der normalerweise mit der Modelloptimierung verbunden ist. (Ssalgadodev, 2024)

Azure ML Designer ist ein drag-and-drop Tool, das es ermöglicht, Machine Learning-Modelle visuell zu erstellen und zu trainieren, ohne Code zu schreiben. Es bietet eine einfache Möglichkeit, verschiedene Schritte der Modellentwicklung zu verknüpfen und ermöglicht die schnelle Entwicklung von Modellen, indem wiederverwendbare Komponenten genutzt werden. Dies erleichtert die Erstellung und Anpassung von Modellen erheblich. (Lgayhardt, 2024a)

4.2.1 Herausforderungen und Risiken

Overfitting: Eines der häufigsten Probleme bei der Modellentwicklung ist das Overfitting, bei dem das Modell zu spezifisch auf die Trainingsdaten abgestimmt ist und in der Praxis schlecht abschneidet. Automated ML hilft, dieses Risiko zu minimieren, indem es verschiedene Modelle und Hyperparameter testet, aber es bleibt eine ständige Herausforderung, die durch regelmäßige Validierung adressiert werden muss.

Hoher Rechenaufwand: Das Training komplexer Modelle kann erhebliche Rechenressourcen erfordern, was die Kosten und die Zeit für die Entwicklung erhöht. Training Pipelines in Azure ML ermöglichen es, diese Anforderungen durch den Zugriff auf skalierbare Rechenressourcen zu bewältigen, aber die effiziente Nutzung bleibt eine Herausforderung.

Reproduzierbarkeit: Die Nachvollziehbarkeit von Experimenten ist entscheidend für die Weiterentwicklung von Modellen. Training Pipelines tragen zur Reproduzierbarkeit bei, indem sie sicherstellen, dass jeder Schritt des Prozesses dokumentiert und wiederholbar ist. Dennoch kann es herausfordernd sein, komplexe Workflows konsistent zu halten, insbesondere bei kontinuierlicher Modellverbesserung.

4.3 Verify (Übergang ML → Dev)

Die Validierung von Machine Learning-Modellen ist ein entscheidender Schritt im MLDevOps-Prozess, um sicherzustellen, dass die entwickelten Modelle zuverlässig und leistungsfähig sind, bevor sie in der Produktionsumgebung eingesetzt werden. Azure ML Studio bietet spezialisierte Tools und Funktionen, um diesen Prozess zu unterstützen, darunter Methoden zur Bewertung der Modellleistung: Mithilfe der Azure ML Evaluation-Komponente lassen sich wichtige Leistungsmetriken wie Genauigkeit, Präzision, Recall und F1-Score berechnen, um die Eignung des Modells für die Anforderungen zu bestätigen. Diese Schritte können direkt im Azure ML Designer hinzugefügt werden.

Ergänzend zur Modellentwicklung bietet der Azure ML Designer nicht nur die Möglichkeit, Modelle visuell zu erstellen und zu trainieren, sondern unterstützt auch die Validierung und Evaluierung in derselben Pipeline. Dadurch lassen sich die Schritte zur Modellentwicklung und -validierung nahtlos verbinden (Lgayhardt, 2024a):

- **Modell trainieren und bewerten:** Nach der Modellentwicklung mit der Train Model-Komponente im Designer können Sie die Evaluate Model-Komponente hinzufügen, um die Modellleistung umfassend zu bewerten. Diese Komponente berechnet automatisch wichtige Metriken und zeigt die Ergebnisse visuell an, um die Bewertungsschritte zu erleichtern.
- **Cross-Validation durchführen:** Die Cross-Validate Model-Komponente erweitert die Möglichkeiten des Azure ML Designers, indem sie das Modell auf mehreren Datenpartitionen trainiert und validiert. Dies bietet eine umfassendere Analyse der Modellgüte und reduziert das Risiko von Overfitting.
- **Metrikauswahl und Analyse:** Die Evaluate Model-Komponente erlaubt die Konfiguration relevanter Metriken und bietet nach dem Lauf eine visuelle Darstellung der Ergebnisse. So können die erzielten Ergebnisse leicht interpretiert und zur weiteren Optimierung des Modells genutzt werden.
- **Automatisierte Workflows und Experiment-Tracking:** Azure ML Designer und das Experiment-Tracking in Azure ML ermöglichen eine kontinuierliche Versionierung und Dokumentation aller Modelltests und Metriken. So wird sichergestellt, dass die Validierungsschritte jederzeit reproduzierbar und nachverfolgbar sind, was für das MLDevOps-Modellmanagement von zentraler Bedeutung ist.

4.3.1 Herausforderungen und Risiken

Herausforderungen bei der Datenverteilung: Unterschiedliche Datenquellen und das Vorhandensein von Feedback-Schleifen können dazu führen, dass sich Datenverteilungen verschieben, was zu sogenannten "Training-Serving Skews" führt. Diese können das Modell negativ beeinflussen und eine kontinuierliche Überwachung notwendig machen. (Polyzotis & Whang, 2019)

Fehlende Datenvalidierung: Ohne geeignete Validierungsmechanismen können verdeckte Annahmen über die Daten unentdeckt bleiben, was zu Performance-Problemen oder fehlerhaften Vorhersagen führt, wenn unvorhergesehene Datenformate auftreten. (Polyzotis & Whang, 2019)

4.4 Package (Dev)

Die Paketierung von Modellen ist ein wesentlicher Schritt im MLDevOps-Prozess, bei dem verifizierte Modelle und alle benötigten Abhängigkeiten für die Produktionsumgebung vorbereitet werden, um eine reibungslose Bereitstellung zu ermöglichen. Azure ML bietet hierfür mehrere leistungsstarke Tools, darunter die Azure ML Model Registry zur Verwaltung und Versionierung von Modellen sowie die Azure ML Docker Environment die eine konsistente Laufzeitumgebung sicherstellen. Diese Komponenten erleichtern eine effiziente und skalierbare Bereitstellung der Modelle in verschiedenen Produktionsumgebungen.

Azure ML Model Registry dient zur zentralen Verwaltung und Versionierung von Modellen. In der Model Registry können verschiedene Versionen eines Modells gespeichert werden, um sicherzustellen, dass stets die richtige Version für das Deployment bereitsteht. (Blackmist, 2024)

Azure ML Docker Environment wird benötigt um die Abhängigkeiten und Bibliotheken für das Modell zu bündeln, dafür kann eine Docker-Umgebung in Azure ML erstellt werden. Diese Umgebung stellt sicher, dass das Modell unabhängig von der Zielplattform mit denselben Konfigurationen läuft. (Dbradish-microsoft, 2024a)

4.4.1 Herausforderungen und Risiken

Komplexität der Containerisierung: Der Prozess des Verpackens eines Modells in einem Docker-Container oder einer anderen Umgebung für die Produktion kann technische Komplexitäten aufwerfen. Die Anforderungen an Abhängigkeiten, Bibliotheken und spezifische Umgebungen machen die Erstellung einer einheitlichen und reproduzierbaren Laufzeitumgebung schwierig. (Naumova, 2024, S.8f)

Kompatibilitätsprobleme: Wenn Modelle auf verschiedenen Plattformen bereitgestellt werden müssen, können Kompatibilitätsprobleme auftreten. Die Abhängigkeiten und Umgebungen sind oft nicht einheitlich, was eine nahtlose Integration erschwert. (Naumova, 2024, S.13)

4.5 Release (Übergang Dev → Ops)

In der Release-Phase des MLDevOps-Prozesses werden die verpackten Modelle in die Produktionsumgebung überführt, wo sie skalierbar und zuverlässig laufen sollen. Azure ML bietet hierfür verschiedene Komponenten, darunter Azure Kubernetes Service (AKS) für die Bereitstellung und Skalierung in produktionsreifen Umgebungen, Azure Container Instances (ACI) für schnelle und flexible Bereitstellungen in weniger kritischen Umgebungen sowie Azure ML Endpoints, die eine REST-API-Schnittstelle für den Zugang zu Modellvorhersagen bieten. Diese Komponenten erleichtern eine kontrollierte und skalierbare Einführung der Modelle in die Produktionslandschaft.

Azure Container Instances (ACI) und Azure Kubernetes Service (AKS) unterstützen die Bereitstellung der Modelle in Containern. ACI ist besonders für die schnelle Bereitstellung und kleinere Testumgebungen

geeignet, während AKS eine skalierbare Option für größere Produktionsumgebungen darstellt. Container bieten eine flexible und standardisierte Möglichkeit, Modelle inklusive ihrer Abhängigkeiten zu verpacken. (Tomvcassidy, 2024) (Nickomang, 2024)

Mit Azure ML Endpoints können Benutzer Endpunkte für ihre Modelle erstellen, die eine REST-API für den Zugriff auf Modellvorhersagen bereitstellen. Diese Endpunkte lassen sich sowohl in AKS als auch in ACI erstellen und bieten eine flexible Möglichkeit, Modelle in der Produktion zugänglich zu machen. (Dbradish-microsoft, 2024b)

4.5.1 Herausforderungen und Risiken

Sicherheitsanforderungen: Sicherheitsrisiken sind besonders hoch, da das verpackte Modell Zugang zu sensiblen Daten haben könnte. Um Sicherheitslücken zu vermeiden, müssen strenge Sicherheitsmaßnahmen ergriffen und kontinuierliche Audits durchgeführt werden. (Naumova, 2024, S.45)

Skalierbarkeit und Lastmanagement: Um eine reibungslose Skalierung zu ermöglichen, sind umfassende Optimierungen und Planung erforderlich. In hochskalierbaren Umgebungen, wie sie durch ML-Modelle in Produktionssettings nötig sind, kann unzureichende Planung zu Engpässen oder Leistungsproblemen führen. (Naumova, 2024, S.10)

Latenz und Reaktionszeit: Produktionsumgebungen stellen oft strenge Anforderungen an die Antwortzeiten. Fehlerhafte Konfigurationen oder unzureichende Ressourcen können die Latenzzeiten erhöhen, was die Benutzererfahrung negativ beeinflusst und somit das Betriebsziel gefährden kann. (Naumova, 2024)

4.6 Configure (Ops)

In der Configure-Phase des MLDevOps-Prozesses wird die Produktionsumgebung so eingerichtet, dass die Modelle unter optimalen Bedingungen laufen. Azure ML bietet hierfür Komponenten wie **Azure Machine Learning Compute**, die die Bereitstellung und Verwaltung skalierbarer Rechenressourcen ermöglicht, sowie Azure Virtual Network (VNet), um eine sichere und kontrollierte Netzwerkstruktur zu schaffen. Diese Komponenten helfen, die Umgebung optimal zu konfigurieren und den reibungslosen Betrieb der Modelle sicherzustellen.

Azure Machine Learning Compute ermöglicht die Einrichtung und Verwaltung von Compute-Ressourcen, die für das Training und die Bereitstellung von Modellen erforderlich sind. Durch die flexible Skalierbarkeit kann die Leistung der Produktionsumgebung an die jeweiligen Anforderungen angepasst werden. (Sdgilley, 2024)

Durch die Nutzung von Azure Virtual Network (VNet) können Ressourcen in Azure ML sicher miteinander verbunden werden. VNets sorgen für eine bessere Netzwerksicherheit und ermöglichen den kontrollierten Zugriff auf Modelle und Datenquellen, indem sie die Umgebung isolieren und den Datenverkehr steuern. (Mbender-ms, 2024)

4.6.1 Herausforderungen und Risiken

Komplexitätsverwaltung: Die Konfiguration verschiedener Infrastruktur-Komponenten birgt Herausforderungen, da Änderungen in einem Netzwerkbereich andere Teile beeinflussen können (He et al., 2019)

Ressourcenmanagement: da eine falsche Konfiguration zu Über- oder Unterauslastung und somit zu erhöhten Kosten oder Leistungsabfällen führen kann (He et al., 2019).

4.7 Monitor (Ops)

In der Monitor-Phase des MLDevOps-Prozesses steht die kontinuierliche Überwachung der Modellleistung und -genauigkeit im Fokus, um eine stabile und langfristig zuverlässige Nutzung sicherzustellen. Azure bietet dafür verschiedene Komponenten wie Azure Monitor und Application Insights, die eine Echtzeitüberwachung der Modelle ermöglichen und frühzeitig auf potenzielle Leistungsprobleme hinweisen. Ergänzend dazu hilft die Model Drift Detection, Abweichungen in den Datenverteilungen zu erkennen und somit Anzeichen einer Modellverschlechterung frühzeitig zu identifizieren. Diese Tools tragen maßgeblich dazu bei, Probleme rechtzeitig zu beheben und die Modellgüte im Produktionsbetrieb aufrechtzuerhalten.

Azure Monitor erfasst und überwacht Modellleistungsmetriken in Echtzeit. Azure Monitor kann genutzt werden, um benutzerdefinierte Warnungen einzurichten, die bei Leistungsproblemen oder Anomalien im Modell automatisch ausgelöst werden. (Msangapu-msft, 2024)

Application Insights ist ideal, um detaillierte Einblicke in die Nutzung und Leistung der Modelle zu erhalten. Es erfasst Metriken wie Antwortzeiten, Fehlerquoten und Nutzungsverhalten, was die frühzeitige Problembehebung unterstützt. (Rboucher, 2024)

Model Drift Detection erkennt Änderungen in den Datenverteilungen, die auf eine Verschlechterung der Modellleistung hinweisen könnten. Das frühzeitige Erkennen von Modelldrift hilft, das Modell rechtzeitig zu aktualisieren und so die Genauigkeit langfristig zu gewährleisten. (SturgeonMi, 2024)

4.7.1 Herausforderungen und Risiken

Data Drift: Veränderungen in den zugrunde liegenden Daten können die Modellleistung verschlechtern. (Bayram et al., 2022)

Verzögerte Erkennung: Verzögerungen bei der Erkennung von Problemen können zu ungenauen Vorhersagen führen. (Bayram et al., 2022)

4.8 Plan (Übergang Ops → Dev)

In der Plan-Phase des MLDevOps-Prozesses werden auf Grundlage der operativen Erkenntnisse Strategien für zukünftige Modellverbesserungen und -entwicklungen erstellt. Azure DevOps Integration ist hierbei eine zentrale Komponente, die eine nahtlose Zusammenarbeit und Planung ermöglicht. Mit Funktionen wie Work Item Tracking und Versionskontrolle unterstützt Azure DevOps die Organisation von Entwicklungs-Roadmaps, die Planung von Ressourcenanforderungen und die Verfolgung von Zielen

für zukünftige Iterationen. Dadurch können die Erkenntnisse aus der Produktionsphase gezielt genutzt werden, um den gesamten MLDevOps-Prozess kontinuierlich zu verbessern und weiterzuentwickeln.

Durch die Integration mit Azure DevOps können Planungs- und Kollaborations-Tools genutzt werden, um Entwicklungs-Roadmaps und Ressourcenanforderungen für künftige Iterationen festzulegen. Work Item Tracking und Versionskontrolle helfen dabei, Entwicklungsziele effizient zu organisieren und zu verfolgen. (Chcomley, 2024)

4.8.1 Herausforderungen und Risiken

Reaktionszeit und Flexibilität: Schnelle Anpassungen und Planung für zukünftige Modellentwicklungen sind nötig, um Anforderungen rechtzeitig zu erfüllen und Verzögerungen zu minimieren (Alberti, Alvarez-Napagao & Anaya, 2024).

Sicherstellung der Compliance: Die Einhaltung von Richtlinien bei Modellerweiterungen ist essenziell, insbesondere für Industriesysteme, die hohen Sicherheitsanforderungen unterliegen (Alberti, Alvarez-Napagao & Anaya, 2024), S.8.

4.9 Create (Übergang Dev → ML)

In der Create-Phase, dem letzten Schritt des MLDevOps-Zyklus, werden die erstellten Modelle produktionsreif gemacht, indem sie in Anwendungen integriert und alle notwendigen Schnittstellen und Logiken entwickelt werden. Hierbei wird sichergestellt, dass das Modell nahtlos in der produktiven Umgebung läuft und seine Funktionen wie gewünscht bereitstellt. Diese Phase schließt den Zyklus ab und bereitet die Grundlage für neue Iterationen und Verbesserungen.

Modelle werden so verfeinert, dass sie optimal in bestehende Systeme passen. Mithilfe von Azure ML SDK und Azure ML Designer kann der Code angepasst werden, um Schnittstellen und Logiken zur Anbindung an die Produktionsumgebung zu entwickeln. Dies erleichtert die Wiederverwendbarkeit und Kompatibilität mit anderen Systemen. (Lgayhardt, 2024c)

Automatisierung und kontinuierliche Anpassung mit Azure ML Pipelines ermöglichen die Automatisierung des gesamten Workflows und erlauben eine kontinuierliche Rückführung der Modelle in den ML-Zyklus. Dadurch wird der Weg für wiederholtes Training und Testen basierend auf neuen Anforderungen geebnet. (Lgayhardt, 2024b)

4.9.1 Herausforderungen und Risiken

Anpassungsfähigkeit: Die kontinuierliche Weiterentwicklung der Modelle erfordert flexible Anpassungen an wechselnde Daten und Anforderungen, was in industriellen Umgebungen schwer umzusetzen ist. (Lwakatare et al., 2020)

Versionskontrolle: Bei der Integration neuer Iterationen muss sichergestellt werden, dass alle Modellversionen nachvollziehbar dokumentiert und gespeichert werden. Falls ein Fehler in der neuen Version auftritt, ermöglicht eine konsistente Versionskontrolle, problemlos auf eine frühere, funktionierende Version zurückzugreifen. Ohne klare Versionsverwaltung besteht das Risiko, dass

frühere, stabile Modelle verloren gehen, was die Modellzuverlässigkeit und -stabilität beeinträchtigen kann.

5. Zusammenfassung

In der Arbeit wird ein umfassender Überblick über den MLDevOps-Prozess gegeben. Zunächst erfolgt eine Einführung in MLDevOps mit einer klaren Abgrenzung zu traditionellem DevOps, um das spezifische Verständnis für maschinelles Lernen in Entwicklungs- und Produktionsumgebungen zu fördern. Die Bedeutung dieses Prozesses für moderne Unternehmen wird hervorgehoben.

Im Hauptteil werden alle Schritte im MLDevOps-Prozess detailliert beschrieben und anhand von Azure-Komponenten erklärt. Jede Phase – von der Datenerfassung bis hin zur Integration neuer Iterationen – wird beleuchtet und durch spezifische Azure-Tools unterstützt. So werden beispielsweise Azure Monitor für die kontinuierliche Überwachung und Azure ML Pipelines zur Automatisierung des Workflows genutzt. Für jede Phase werden die Herausforderungen und Risiken detailliert beschrieben, um die potenziellen Schwierigkeiten zu veranschaulichen.

Die Arbeit zeigt auf, dass MLDevOps ein kontinuierlicher Prozess ist, der fortlaufende Modellanpassungen erfordert. Zukünftige Entwicklungen sollten auf die kontinuierliche Verbesserung und die Gewährleistung ethischer Standards ausgerichtet sein, um ML-Modelle in modernen Unternehmen langfristig erfolgreich zu implementieren.

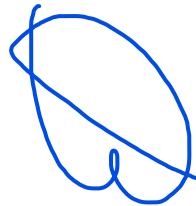
Literaturverzeichnis

- Alberti, E., Alvarez-Napagao, S. & Anaya, V. (2024). AI Lifecycle Zero-Touch Orchestration within the Edge-to-Cloud Continuum for Industry 5.0. *Systems*, 12(2), 48.
<https://doi.org/10.3390/systems12020048>
- Alberti, E., Alvarez-Napagao, S., Anaya, V., Barroso, M., Barrué, C. & Beecks, C. (2024). AI Lifecycle Zero-Touch Orchestration within the Edge-to-Cloud Continuum for Industry 5.0. *Systems*, 12(2), 48.
<https://doi.org/10.3390/systems12020048>
- Bayram, F., Ahmed, B. S. & Kassler, A. (2022). From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems*, 245, 108632.
<https://doi.org/10.1016/j.knosys.2022.108632>
- Blackmist. (2024, 28. August). *Register and work with models - Azure Machine Learning*.
- Chcomley. (2024, 19. Juli). *What is Azure DevOps? - Azure DevOps*. <https://learn.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops>
- Coté, C., Ciaburro, G. & Gutzait, M. (2018). *Hands-on data warehousing with Azure Data Factory: ETL techniques to load and transform data from various sources, both on-premises and on cloud*. Packt Publishing.
<https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1823664>
- Dbradish-microsoft. (2024a, 26. Juni). *az ml environment*. <https://learn.microsoft.com/en-us/cli/azure/ml/environment?view=azure-cli-latest>
- Dbradish-microsoft. (2024b, 26. Juni). *az ml online-endpoint*. <https://learn.microsoft.com/en-us/cli/azure/ml/online-endpoint?view=azure-cli-latest>
- He, M., Alba, A., Basta, A., Blenk, A. & Kellerer, W. (2019). Flexibility in Softwarized Networks: Classifications and Research Challenges. *IEEE Communications Surveys & Tutorials*, 21(3), 2600–2636. <https://doi.org/10.1109/COMST.2019.2892806>
- Hegeman, T., Jansen, M., Iosup, A. & Trivedi, A. (2021). GradeML: Towards Holistic Performance Analysis for Machine Learning Workflows. In J. Bourcier (Hrsg.), *ACM Digital Library, Companion of the ACM/SPEC International Conference on Performance Engineering* (S. 57–63). Association for Computing Machinery. <https://doi.org/10.1145/3447545.3451185>
- Islam, S. (2022). *MLDevOps stellt eine Weiterentwicklung traditioneller DevOps-Praktiken dar, die speziell auf die Bedürfnisse des Machine Learning (ML) zugeschnitten ist. Während DevOps primär die Automatisierung und Effizienz in der Softwareentwicklung fokussiert, erweitert MLDevOps diesen Ansatz um spezifische Anforderungen des ML-Lebenszyklus, wie die Verarbeitung großer Datenmengen, das Training komplexer Modelle und die Überwachung der Modellleistung im Einsatz. In diesem Kapitel werden die wesentlichen Unterschiede zwischen MLDevOps und herkömmlichem DevOps erläutert*. <https://www.phdata.io/blog/mlops-vs-devops-whats-the-difference/>
- Jonburchel. (2024, 6. September). *Azure Data Factory-Dokumentation - Azure Data Factory*. <https://learn.microsoft.com/de-de/azure/data-factory/>
- Korner, C. (2020). *Mastering Azure Machine Learning: Perform large-scale end-to-end advanced machine learning in the cloud with Microsoft Azure Machine Learning* (1. Aufl.). Packt Publishing Limited.
https://www.wiso-net.de/document/PKEB__9781789801521436

- Kromer, M. (2022). *Mapping Data Flows in Azure Data Factory: Building Scalable ETL Projects in the Microsoft Cloud* (1st ed.). Apress L. P.
<https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=7077817>
- Kromerm. (2024, 30. September). *Zuordnen von Datenflüssen - Azure Data Factory*.
<https://learn.microsoft.com/de-de/azure/data-factory/concepts-data-flow-overview>
- Kulkarni, R. V., Jagtap, V., Naik, T., Shaha, S. & Nikumbh, K. (2024). Leveraging Azure Data Factory for COVID-19 Data Ingestion, Transformation, and Reporting. In T. Senjyu, C. So-In & A. Joshi (Hrsg.), *Lecture Notes in Networks and Systems: Bd. 947, Smart Trends in Computing and Communications: Proceedings of SmartCom 2024, Volume 3* (1st ed. 2024, S. 275–285). Springer Nature Singapore; Imprint Springer. https://doi.org/10.1007/978-981-97-1326-4_23
- Lgayhardt. (2024a, 28. August). *What is Designer (v2)? - Azure Machine Learning*.
<https://learn.microsoft.com/en-us/azure/machine-learning/concept-designer?view=azureml-api-2>
- Lgayhardt. (2024b, 16. September). *What are machine learning pipelines? - Azure Machine Learning*.
<https://learn.microsoft.com/en-us/azure/machine-learning/concept-ml-pipelines?view=azureml-api-2>
- Lgayhardt. (2024c, 7. Oktober). *Tutorial: ML pipelines with Python SDK v2 - Azure Machine Learning*.
<https://learn.microsoft.com/en-us/azure/machine-learning/tutorial-pipeline-python-sdk?view=azureml-api-2>
- Lwakatare, L., Raj, I. & Olsson, H. (2020). Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. *Information and Software Technology*, 127, 106368. <https://doi.org/10.1016/j.infsof.2020.106368>
- MayMSFT. (2024, 15. Januar). *Prepare data for ML modeling using Microsoft Data Prep SDK for .NET - Microsoft DataPrep.NET SDK*. <https://learn.microsoft.com/en-us/dotnet/api/overview/data-prep/overview-data-prep>
- Mbender-ms. (2024, 25. Oktober). *Azure networking services overview*. <https://learn.microsoft.com/en-us/azure/networking/fundamentals/networking-overview>
- Msangapu-msft. (2024, 24. September). *Monitor Azure App Service - Azure App Service*.
<https://learn.microsoft.com/en-us/azure/app-service/monitor-app-service>
- Naumova, J. (1. Oktober 2024). Automation and Autonomous system Architecture Framework – Phase 2. *NGMN e. V.* <https://www.ngmn.org/publications/automation-autonomous-framework-phase-2.html>
- Nickomang. (2024, 18. September). *Azure Kubernetes Service (AKS) documentation*.
<https://learn.microsoft.com/en-us/azure/aks/>
- Polyzotis, N. & Whang, S. (2019). Data Validation for Machine Learning. *Proceedings of Machine Learning and Systems*, 1, 334–347.
- Rboucher. (2024, 11. September). *Azure Monitor Insights Overview - Azure Monitor*.
<https://learn.microsoft.com/en-us/azure/azure-monitor/insights/insights-overview>
- Sdgilley. (2024, 18. Oktober). *Understand compute targets - Azure Machine Learning*.
<https://learn.microsoft.com/en-us/azure/machine-learning/concept-compute-target?view=azureml-api-2>
- Ssalgadodev. (2024, 28. August). *What is automated ML? AutoML - Azure Machine Learning*.
<https://learn.microsoft.com/en-us/azure/machine-learning/concept-automated-ml?view=azureml-api-2>
- SturgeonMi. (2024, 24. September). *Detect data drift on datasets (preview) - Azure Machine Learning*.

Tomvcassidy. (2024, 30. August). *Serverless containers in Azure - Azure Container Instances*.
<https://learn.microsoft.com/en-us/azure/container-instances/container-instances-overview>
Wodecki, A. (2023). A Cross-Disciplinary Knowledge Management Framework for Artificial Intelligence in
Business Projects. [https://www.semanticscholar.org/paper/A-Cross-Disciplinary-Knowledge-
Management-Framework-Wodecki/e1f27c868394115add9d7c414036b86cf895eae9](https://www.semanticscholar.org/paper/A-Cross-Disciplinary-Knowledge-Management-Framework-Wodecki/e1f27c868394115add9d7c414036b86cf895eae9)

Ich versichere, dass ich das beiliegende Assignment selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlich oder sinngemäß übernommenen Stellen in der Arbeit gekennzeichnet habe.

A handwritten signature in blue ink, consisting of a large, loopy 'O' shape with a small 'u' or 'e' at the bottom, and a diagonal line crossing through the middle.