Die Operationalisierung von Data Science Pipelines in Unternehmen

Carl Schünemann (00107827)

Technische Hochschule Ingolstadt
Fakultät Informatik
Cloud Applications and Security Engineering
cas0597@thi.de

Zusammenfassung. Der *Data-Science-Prozess* fördert als iterative, wissenschaftlich fundierte und teamübergreifende Methode die kontinuierliche Entwicklung neuer Anwendungen maschinellen Lernens mithilfe von *Data Science Pipelines*. Diese beschreiben eine Reihe von Verarbeitungsschritten, um aus einem gegebenen Datensatz ein Machine-Learning-Modell abzuleiten. In dieser Arbeit werden insbesondere die Unterschiede der Anwendung des zyklischen Data-Science-Prozesses zwischen der praktischen Forschung und Unternehmen herausgestellt. Die zentrale These lautet, dass *Continuous Deployment* als etabliertes Werkzeug der Software-Entwicklung dazu genutzt werden kann, um die Entwicklung von Machine-Learning-Modellen mit Data Science Pipelines erfolgreich in Unternehmen zu integrieren. Diese Annahme wird durch eine praktische Implementierung des Frameworks *MLOps* bestätigt sowie auf *DataOps* zur Steuerung von Informationsflüssen abgebildet.
Keywords: Wissenschaftliche Methode, Data Science, Data Science Pipeline, Operationalisierung, MLOps, DataOps

Seminar Data Science
WS 23/24, Technische Hochschule Ingolstadt

Chapter 1

Einleitung

Im 17. Jahrhundert revolutionierte Sir Isaac Newton mit seinem physikalischen Modell der Gravitation die Astronomie. Über 300 Jahre lang konnten mit den Newtonschen Gleichungen die Bewegungen der Himmelskörper erklärt und berechnet werden. Doch im Laufe der Zeit traten mit neuen Beobachtungen Unstimmigkeiten auf, insbesondere bei der präzisen Vorhersage der Umlaufbahn des Planeten Merkur um die Sonne. Albert Einstein entwickelte deshalb im frühen 20. Jahrhundert auf Basis aktueller Beobachtungen seiner Zeit ein neues Modell. Anhand seiner Theorie konnte erstmals die Umlaufbahn Merkurs exakt berechnet werden. Jedoch genügte es nicht nur, ein neues Gravitationsmodell zu präsentieren, mit dem Umlaufbahnen korrekt berechnet werden konnten – es musste auch in seinen weitergehenden Aussagen validiert werden. Um dies zu erreichen, wurden im Jahr 1919 zwei Expeditionen initiiert, um während einer Sonnenfinsternis in Südamerika und Afrika die Beugung von Sternenlicht im Gravitationsfeld der Sonne zu messen. Die erzielten Messwerte stimmten mit den theoretischen Vorhersagen des Einsteinschen Modells überein und die "allgemeine Relativitätstheorie" etablierte sich als neuer Status Quo in der Physik. [8]

Diese Anekdote illustriert das Konzept der sogenannten *wissenschaftlichen Methode*: Durch fortlaufende Messungen und Beobachtungen wird der in der wissenschaftlichen Gemeinde vorherrschende Konsens zur Erklärung eines Wissensstands, das *Modell*, überprüft. Identifizierte Ungenauigkeiten des akzeptierten Modells führen zur Entwicklung eines neuen Modells, basierend auf dem jeweils aktuellen Wissensstand. Nach Validierung ersetzt dieses neue Modell den Status Quo und der Zyklus der wissenschaftlichen Methode setzt sich fort. [28]

Das Konzept der wissenschaftlichen Methode lässt sich auch auf Modelle maschinellen Lernens übertragen. So definiert *Data Science* einen ganzheitlichen und zyklischen Prozess auf Basis der oben dargestellten wissenschaftlichen Methode zum Lösen von Datenproblemen mit *Machine-Learning-Modellen*. [28] Im Rahmen dieser Seminararbeit wird dieser *Data-Science-Prozess* vorgestellt. Der Fokus liegt insbesondere auf der Prozessphase zur Erstellung von Machine-Learning-Modellen, der sogenannten *Data Science Pipeline*. In dieser Arbeit wird die Divergenz bei der praktischen Umsetzung von Data Science in Forschung und Unternehmen aufgezeigt. Mit *MLOps* wird ein Framework zur *Operationalisierung von Data Science Pipelines* vorgestellt. Es dient dem erfolgreichen Einsatz von Data Science Pipelines in Unternehmen zur Entwicklung von Software-Produkten mit Machine-Learning-Modellen.

Chapter 2

Data Science

In der Fachliteratur sind Begriffe im Kontext von Data Science häufig unklar und unterschiedlich definiert. Wo nicht anders angegeben, entschied sich der Autor dieser Seminararbeit für die Übernahme der nachfolgenden Begriffsdefinitionen von Raina und Krishnamurthy. Sie zeichnen sich durch eine besonders klare Abgrenzung der relevanten Fachbegriffe aus. [28]

2.1 Wissenschaftliche Methode

Die *wissenschaftliche Methode* beschreibt die zyklische Verbesserung der akzeptierten Erklärung eines Wissensstands, des *Modells*. Das aktuelle akzeptierte Modell wird als *Status Quo* bezeichnet. Wie das Eingangsbeispiel illustriert, wird die wissenschaftliche Methode durch die drei orange in Abbildung 1 dargestellten Merkmale charakterisiert:

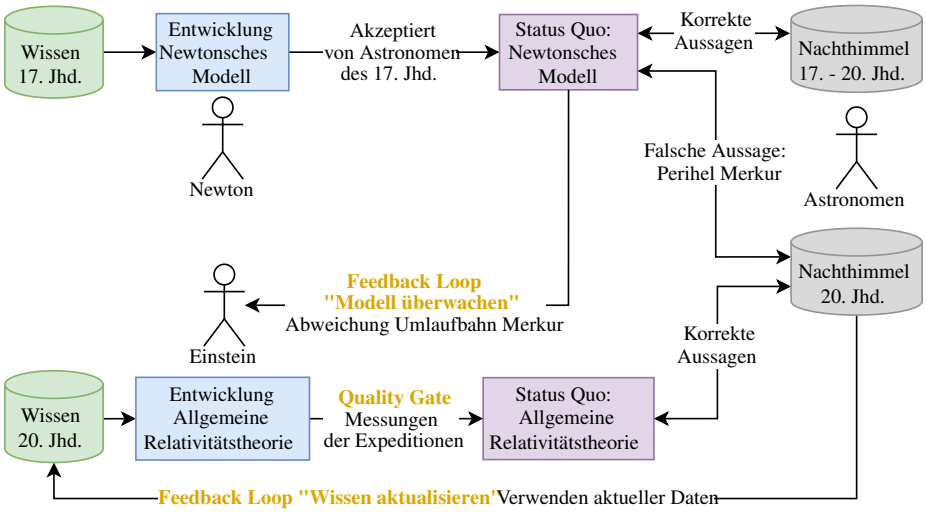

Figure 1. Wissenschaftliche Methode am Beispiel von Einsteins Relativitätstheorie

Feedback Loop "Modell überwachen" [3]: Durch Rückmeldungen der Anwender des Status Quo werden Abweichungen und Ungenauigkeiten identifiziert. So stellten die Astronomen des frühen 20. Jahrhunderts Ungenauigkeiten bei der Anwendung des Newtonschen Gravitationsmodells zur Berechnung der Umlaufbahn Merkurs fest.

Feedback Loop "Wissen aktualisieren" [3]: Vor der Entwicklung eines neuen Modells wird der Datensatz aktualisiert. Einstein verwendete für sein Modell nicht Newtons Daten des 17. Jahrhunderts, sondern aktuelle Messungen seiner Zeit.

Quality Gate [19]: Bevor ein neues Modell zum Status Quo wird, muss es umfassend validiert werden. So wurde die Korrektheit von Einsteins Modell durch die Messung der Lichtbeugung empirisch belegt. Auch als *Gatekeeper* bezeichnet [1, 2].

2.2 Data-Science-Prozess

Als *Data Science* wird die Anwendung der wissenschaftlichen Methode auf *Machine-Learning-Modelle* bezeichnet. Der daraus resultierende *Data-Science-Prozess*, kurz *DS-Prozess*, ist in Abbildung 2 dargestellt.

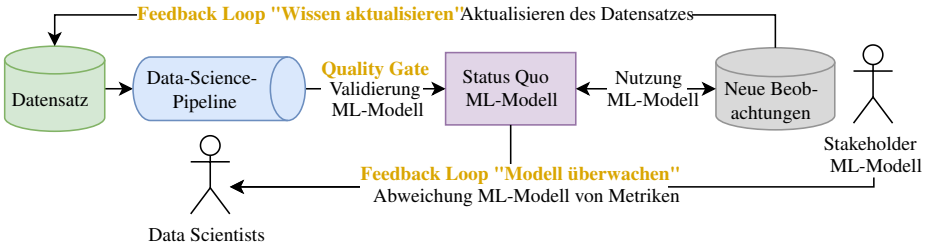

Figure 2. Data-Science-Prozess: Anwendung der wissenschaftlichen Methode auf Modelle maschinellen Lernens

Machine-Learning-Modell

Machine-Learning-Modelle, kurz *ML-Modelle*, sind Anwendungen des *maschinellen Lernens*, kurz *ML*. Dazu zählen gemäß des aktuellen wissenschaftlichen Konsenses alle Software-Anwendungen statistischer Methoden und Algorithmen sowie deren Erweiterungen um Technologien wie Deep Learning. Ein ML-Modell verarbeitet stets Input-Daten, um eine vorgegebene Problemstellung der Stakeholder des ML-Modells zu lösen. Dazu liefert das ML-Modell Erklärungen von Zuständen (*Deskription*), Vorhersagen über künftige Zustände (*Prädiktion*) oder gibt Handlungsempfehlungen zum Erreichen bestimmter Zustände (*Präskription*). Synonym zu ML-Modell wird in der Literatur der Begriff *AI Engine* und *KI-Maschine* verwendet [3].

Datensatz

Die Güte eines ML-Modells hängt vollständig von den Daten ab, aus denen es abgeleitet wird, dem *Datensatz*. Ein prädiktives Machine-Learning-Modell könnte beispielsweise basierend auf historischen Sensordaten Vorhersagen über das zukünftige Wetter an einem Ort treffen.

Data Science Pipeline

Die Ableitung eines ML-Modells aus dem Datensatz findet in der *Data Science Pipeline*, kurz *DS Pipeline*, statt. Dieser bildlich als Pipeline darstellbare Prozess stützt sich auf den Datensatz als Eingabe und liefert ein ML-Modell für die gegebene Problemstellung als Ausgabe. Die DS Pipeline wird von *Data Scientists* erstellt und gewartet, einem Team von Experten mit ML- und Domänenwissen.

Anwendung der wissenschaftlichen Methode

Laut Raina und Krishnamurthy definiert sich Data Science erst durch die Erfüllung der Merkmale der wissenschaftlichen Methode im DS-Prozess als eigene Wissenschaft.

Feedback Loop "Modell überwachen": Durch definierte Metriken werden Ungenauigkeiten und Abweichungen des ML-Modells in der Produktivumgebung identifiziert. Quantitative Metriken können beispielsweise anhand der Nutzung des ML-Modells Änderungen in der Grundgesamtheit der Daten identifizieren. Ebenso können neue Wünsche oder Rückmeldungen der Stakeholder als qualitative Metriken berücksichtigt werden [3].

Feedback Loop "Wissen aktualisieren": Nachdem Abweichungen des ML-Modells durch Metriken identifiziert wurden, wird vor Entwicklung des neuen Modells der Datensatz aktualisiert. Somit wird die aktuelle Grundgesamtheit der für die Problemstellung des ML-Modells relevanten Daten erfasst. Dies umfasst auch die initiale Erstellung des Datensatzes in der ersten Iteration des DS-Prozesses.

Quality Gate: Bevor das neue ML-Modell als akzeptierter Status Quo gilt, muss es von allen beteiligten Stakeholdern validiert werden. Beispielsweise überprüfen die Stakeholder des ML-Modells die Erfüllung der Problemstellung oder andere Data Science Teams validieren mit Peer-Reviews dessen technische Umsetzung.

Zusammenfassend beschreibt der Data-Science-Prozess eine datengetriebene Anwendung der wissenschaftlichen Methode mit dem Ziel bei jeder Iteration den Status Quo durch ein verbessertes und validiertes ML-Modell zu ersetzen [6].

Chapter 3

Data Science Pipelines

Der Begriff *Pipeline* wurde durch das von Shaw und Garlan definierte Software-Entwurfsmuster *Pipes-and-Filter* geprägt: Verarbeitungseinheiten (Filter) sowie Verbindungen zwischen den Einheiten (Pipes) regeln das Verhalten aufeinanderfolgender Teilsysteme [11]. Eine Data Science Pipeline, kurz DS Pipeline, beschreibt eine Reihe von Verarbeitungsschritten, die *DS-Pipeline-Stufen*, im folgenden auch nur *Stufen* genannt, in denen aus einem gegebenen Datensatz ein ML-Modell abgeleitet wird. Eine DS-Pipeline-Stufe erhält optional die Eingaben vorheriger Stufen, verarbeitet Daten deterministisch und liefert optional für nachfolgende Stufen Ausgaben. [3, 28]

Im Rahmen einer umfassenden Literaturrecherche identifizierten Biswas et al. Gemeinsamkeiten unterschiedlicher DS-Pipeline-Implementierungen und leiteten daraus eine generische DS Pipeline ab. Die DS-Pipeline-Stufen wurden in drei sogenannten *Ebenen* gruppiert:
Vorverarbeitung, **Modellbildung** und **Nachbereitung** [3]. Je nach Domäne und Problemstellung des ML-Modells werden die Stufen dieser Ebenen unterschiedlich gewichtet.

Dieser Aufbau der generischen DS Pipeline nach Biswas et al. wird nachfolgend anhand eines Beispiels vorgestellt. Ein prädiktives Modell soll den Kraftstoffverbrauch eines Autos in Meilen pro Gallone (mpg) durch Regression vorhersagen. Als gegebener Datensatz wird das "Auto-mpg dataset" der University of California Irvine als CSV-Datei verwendet: Die Daten stammen von einer Erhebung der American Statistical Association aus dem Jahr 1983 für 406 Fahrzeugtypen unterschiedlicher Hersteller. Der Datensatz umfasst acht Variablen, deren Bedeutung in Tabelle 1 erläutert ist. [27]

Tabelle 1. Variablen des Auto-mpg-Datensatzes

Variable	Bedeutung
mpg	Kraftstoffverbrauch in Meilen pro Gallone
cylinders (cyl)	Zylinderzahl
displacement (dpl)	Hubraum in Kubikzoll
horsepower (hp)	Leistung in PS
weight (wght)	Gewicht in Pound (lb)
acceleration (acc)	Beschleunigung in Sekunden von 0 auf 60 Meilen pro Stunde
year	Baujahr Modulo 100
origin (org)	Kodiertes Ursprungsland: 1 = amerikanisch, 2 = europäisch, 3 = japanisch
name	Modellbezeichnung

Für die Implementierung wird ein Jupyter Notebook verwendet. Jede Zelle enthält ein Pythonskript, das die Verarbeitungslogik der jeweiligen DS-Pipeline-Stufe ausführt.

Ebene “Vorverarbeitung“

Ziel der Vorverarbeitungsebene ist es, den gegebenen Datensatz für den später verwendeten Machine-Learning-Algorithmus aufzubereiten. Die Datenerhebung selbst ist nicht Teil der DS Pipeline. Gemäß des übergeordneten Data-Science-Prozesses, entspricht dies dem Feedback Loop “Wissen aktualisieren“.

Stufe “Datenspeicherung“: Der gegebene Datensatz muss in ein für alle DS-Pipeline-Stufen handhabbares Datenformat umgewandelt werden. Abbildung 3 zeigt, wie die gegebene CSV-Datei als Pandas Dataframe eingelesen wird.

```
# Einlesen Datensatz "auto_mpg" aus aktuellem Ordner
df = pd.read_csv('data.csv')
```
✓ 0.0s

Figure 3. Einlesen der CSV-Datei als Pandas Dataframe

Stufe “Datenaufbereitung“: Im Datensatz müssen für die Problemstellung unerwünschte Artefakte identifiziert (*Datenexploration*) und entfernt (*Data Cleaning*) werden. Für das Regressionsbeispiel werden in Abbildung 4 alle Variablen mit nicht-numerischen Daten identifiziert.

```
# Ausgabe aller Variablen mit nicht-numerischen Daten
display(df.select_dtypes(exclude=['number']).columns)
```
✓ 0.0s

```
Index(['hp', 'name'], dtype='object')
```

Figure 4. Datenexploration: Identifikation von Variablen mit nicht-numerischen Daten

Das nominale Merkmal “name“ wird in Abbildung 5 entfernt, während inkompatible Dateneinträge des Merkmals “horsepower“ in numerische Datentypen umgewandelt werden. Zusätzlich werden alle *Ausreißerdaten* entfernt, welche um ein Vielfaches der Standardabweichung vom Mittelwert abweichen.

```
# Entfernung der nominaler Merkmale für die Regression
df = df.drop(columns=['name'])

# Umwandlung in numerische Datentypen, löschen inkompatibler Daten
df = df.apply(pd.to_numeric, errors='coerce').dropna()

# Entfernen von Ausreißerdaten außerhalb von Mittelwert +/- 3 Standardabweichung
for col in df.columns:
    df = df.drop(df[(df[col] < (df[col].mean() - 3*df[col].std()))
                    | (df[col] > (df[col].mean() + 3*df[col].std()))].index)
```

Figure 5. Data Cleaning: Umwandlung in einen ausschließlich numerischen Datensatz ohne Ausreißerdaten

Ebene “Modellbildung“

Basierend auf dem aufbereiteten Datensatz wird ein ML-Modell für die gegebene Problemstellung abgeleitet.

Stufe “Feature Engineering“: Die Variablen des Datensatzes tragen möglicherweise nicht gleichermaßen zur Problemlösung des ML-Modells bei. Für den Kraftstoffverbrauch wird aufgrund vorhandenen Domänenwissens ein linearer Zusammenhang vermutet. Deshalb wird mithilfe einer *Heat Map* in Abbildung 6 die bivariate Korrelation untersucht: je heller die Verfärbung, desto größer ist die paarweise lineare Korrelation der Variablen.

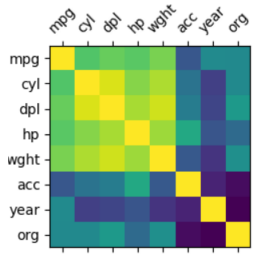

Figure 6. Heat Map zur Visualisierung der bivariaten Korrelation

Die Variablen Gewicht und Baujahr werden aufgrund der hohen linearen Korrelation mit dem Verbrauch sowie der geringen linearen Korrelation untereinander ausgewählt. In Abbildung 7 wird dies mit *Scatterplots* überprüft.

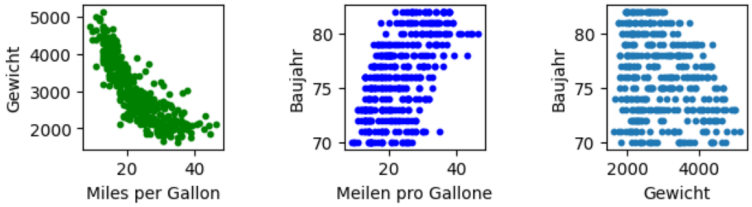

Figure 7. Scatterplots visualisieren die vermuteten Zusammenhänge der Variablen.

Stufe “Modellerzeugung“: Die Erkenntnisse des Feature Engineerings werden verwendet, um aus dem vorverarbeiteten Datensatz ein ML-Modell abzuleiten. Aufgrund der vermuteten linearen Abhängigkeit wird in Abbildung 8 ein lineares Regressionsmodell abgeleitet. Zur Vereinfachung des Beispiels erfolgt keine Trennung in Trainings- und Testdaten.

```
X = df[["wght", "year"]] # Ergebnis Feature Engineering: unabhängige Variablen
y_true = df_selected.mpg # abhängige Variable "Verbrauch"

lm = LinearRegression()
lm.fit(X, y_true) # Durchführen der Regression
```

Figure 8. Ableitung eines linearen Regressionsmodells aus den Variablen des vorverarbeiteten Datensatzes

Stufe “Evaluierung“: Zur Bewertung der Güte des ML-Modells werden Metriken, abhängig vom verwendeten ML-Algorithmus, bewertet. In Abbildung 9 zeigt das hohe [17] Bestimmtheitsmaß $R^2 \approx 0,81$, dass circa 81% der Streuung des Kraftstoffverbrauchs durch das Gewicht und Baujahr erklärt werden kann. Die Güte der Regression kann grafisch durch den Vergleich der vorhergesagten Werte mit den tatsächlichen Werten dargestellt werden. Im Optimalfall bilden die Punkte eine Ursprungsgerade. Weiterhin kann die Streuung der Differenzen zwischen Vorhersage und tatsächlichen Werten, den *Residuen*, grafisch evaluiert werden.

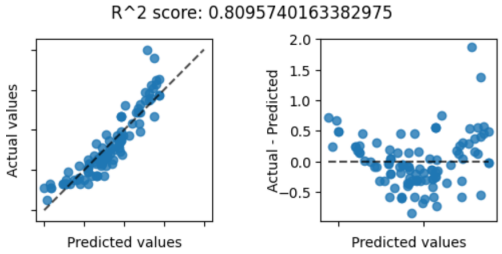

Figure 9. Bewertung der Regressionsgüte durch den Vergleich der Vorhersagen des ML-Modells mit den tatsächlichen Werten des Datensatzes

Ebene “Nachbereitung“

Nachdem ein Modell mit hinreichender Güte aus dem vorverarbeiteten Datensatz abgeleitet wurde, wird dieses in ein auslieferbares Format umgewandelt.

Stufe “Bereitstellung“ In der einzigen Stufe dieser Ebene wird das Modell in Abbildung 10 als “Pickle“-Datei gespeichert. Somit kann es als Python-Objekt wiederverwendet werden.

```
with open("model.pkl",'wb') as f:
    pickle.dump(lm,f)
```

Figure 10. Im “Pickle“-Format ist das Modell als Python-Objekt wiederverwendbar.

Iterationen zwischen den Stufen

Die Ausführung der DS-Pipeline-Stufen erfolgt teilautomatisiert und iterativ durch das Data Science Team. Biwal et al. stellten in ihrer Literaturrecherche fest, dass insbesondere die Ebene “Modellbildung“ wiederholt wird [3].

Im angeführten Python-Beispiel zur Vorhersage des Kraftstoffverbrauchs wird ein Data Scientist nach der Stufe “Evaluierung“ versuchen, die Güte des ML-Modells zu verbessern. So kann beispielsweise in den Scatterplots der Stufe “Feature Engineering“ in Abbildung 7 eine leicht polynomiale statt lineare Abhängigkeit vermutet werden. Somit wird in Abbildung 11 zurück zur Stufe “Modellerzeugung“ gesprungen und statt einem linearen, ein polynomiales Regressionsmodell vom Grad zwei abgeleitet.

```
X = df[["wght", "year"]] # Ergebnis Feature Engineering: unabhängige Variablen
y_true = df["mpg"] # abhängige Variable "Verbrauch"

X_poly = PolynomialFeatures(degree=2).fit_transform(X)
poly = LinearRegression()
poly.fit(X_poly, y_true)
```

Figure 11. Rücksprung zur Stufe “Modellerzeugung“ und Ableiten eines polynomialen Regressionsmodells

Abbildung 12 zeigt eine Verbesserung des neu berechneten Bestimmtheitsmaßes $R^2 \approx 0,86$) in der Stufe “Evaluierung“.

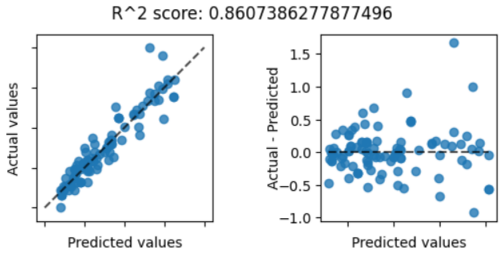

Figure 12. Das neue polynomiale ML-Modell verbessert das Bestimmtheitsmaß.

Um die Güte des ML-Modells weiter zu verbessern, wird die Stufe “Feature Engineering“ wiederholt und die Variable “Baujahr“ durch das kodierte “Ursprungsland“ ersetzt. Der mittlere Scatterplot in Abbildung 13 lässt eine Abhängigkeit zwischen “Ursprungsland“ und dem Kraftstoffverbrauch vermuten.

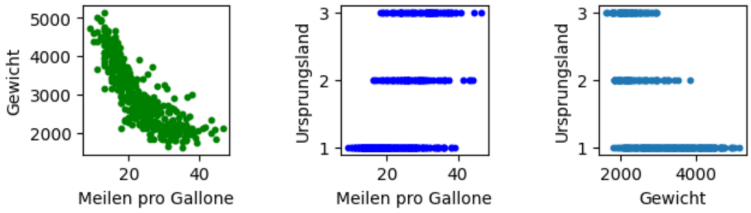

Figure 13. Der mittlere Scatterplot visualisiert den Zusammenhang des Kraftstoffverbrauchs mit dem kodierten “Ursprungsland“.

Nach der Ableitung des neuen polynomialen Modells zeigt Abbildung 14 der Stufe “Evaluation“ eine deutliche Verschlechterung des Bestimmtheitsmaßes $(0, 72)$.

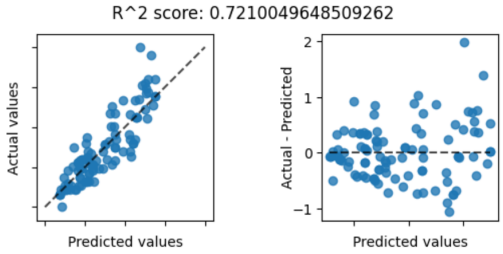

Figure 14. Die neue Variable “Ursprungsland“ bringt eine Verschlechterung des Bestimmtheitsmaßes.

Das lässt sich auf die Abhängigkeit der neuen Variable “Ursprungsland“ von der zweiten unabhängigen Variablen “Gewicht“ zurückführen – zu sehen im rechten Scatterplot der Abbildung 13: In den USA gibt es schwerere Fahrzeuge als in Japan oder Europa. Die Änderungen dieser Iteration werden somit rückgängig gemacht.

Gemäß Biswas et al. handelt es sich bei der DS Pipeline um einen iterativen und dynamischen Ablauf: ein zyklischer Prozess innerhalb des zyklischen DS-Prozesses. Sie argumentieren deshalb, dass die DS Pipeline einem DS-Prozess im kleinen Maßstab entspricht. [3]

Chapter 4

Data Science in Unternehmen

Unternehmen betrachten maschinelles Lernen als Schlüsselinstrument, um Wettbewerbsvorteile zu erlangen [20]. So betont Netflix, dass es jährlich Milliardeneinsparungen durch selbst entwickelte Empfehlungsalgorithmen erzielt [15]. Dies führt zu erheblichen Investitionen in neue Projekte und übt gleichzeitig Wettbewerbsdruck auf andere Unternehmen aus [3]. Weiterhin gibt es Erfolge bei der Erforschung des maschinellen Lernens, wie beispielsweise die Durchbrüche von OpenAI bei generativen ML-Modellen zeigen [26]. Doch trotz Investitionen und technologischer Erfolge scheitern laut einer Studie von Gartner aus dem Jahr 2018 85% aller Unternehmensprojekte zur Entwicklung von Software-Produkten mit ML-Modellen, sogenannte *ML-Produkte* [12, 18].

Dieser Widerspruch lässt sich laut Atwal durch Unterschiede in der Praxis von Forschung und Unternehmen erklären [1]. Biswas et al. kamen im Zug ihrer Literaturrecherche zum Ergebnis, dass sich Forschungsprojekte unterschiedlicher Größen fast ausschließlich auf die Entwicklung von ML-Modellen mit DS Pipelines konzentrieren, ohne den DS-Prozess vollständig umzusetzen [3]:

Beispielsweise schreiben gemeinnützige Organisationen oder Unternehmen auf der Plattform *Kaggle* Wettbewerbe für ML-Modelle aus. Als Stakeholder formulieren sie eine Problemstellung, geben einen Datensatz vor und definieren eine numerische Metrik zur Bewertung der eingereichten ML-Modelle. In diesen Wettbewerben treten Forschungsgruppen und Privatpersonen gegeneinander an. [3, 21]

Fehlendes Qualiy Gate: Das gewinnende ML-Modell erfüllt die vorgegebene Metrik besser als die konkurrierenden ML-Modelle. So wird der Status Quo nur relativ zu anderen ML-Modellen validiert, eine umfängliche Validierung bezüglich der Stakeholder-Anforderungen findet nicht statt.

Fehlende Feedback Loops: Nach Ende eines Kaggle-Wettbewerbs und Auszahlung des Preisgeldes werden alle teilnehmenden ML-Modelle zum Download veröffentlicht. Die Rückkopplung mit den Stakeholdern sowie die Aktualisierung des Datensatzes ist nicht vorgesehen.

Das DS Pipeline-zentrische Arbeiten von Forschungsgruppen ohne Quality Gate und Feedback Loops sieht Atwal als Ursache des Scheiterns von ML-Produkten: Betrachtet man das in Kapitel 3 entwickelte ML-Modell zur Vorhersage des Kraftstoffverbrauchs als ML-Produkt, würde für Data Scientists die Arbeit mit der Übergabe des ML-Modells im Pickle-Format an die Stakeholder enden. [1]

Die Herausforderung für Unternehmen ist deshalb die Anwendung der wissenschaftlichen Methode auf DS Pipelines zur Entwicklung von ML-Produkten. Dieser notwendige Schritt wird als sogenannte *Operationalisierung von DS Pipelines* bezeichnet. [1, 14, 20]

4.1 Operationalisierung der Entwicklung von Cloud Software durch Continuous Deployment (CDP)

Operationalisierung beschreibt die teamübergreifende Einbettung eines Software-Entwicklungsprozesses in ein Unternehmen [22]. In der Vergangenheit wurden Softwareanwendungen manuell auf der lokalen Kundeninfrastruktur installiert. Die Verantwortung für die sogenannte *Produktivumgebung* der Software lag somit vollständig beim Kunden. Die zeitnahe Reaktion des Software-Anbieters auf auftretende Softwareprobleme wurde durch ausgeklügelte Wartungsverträge ausgeschlossen. [31] Mit der zunehmenden Popularität von Cloud-Produkten sowie deren propagierten Vorteile wie Skalierbarkeit und Verfügbarkeit, migrierten Unternehmen ihre Softwareprojekte auf externe oder selbst vom Unternehmen verwaltete Cloud-Infrastrukturen. Teamübergreifende Prozesse zwischen der Softwareentwicklung und deren Bereitstellung sowie Überwachung in der Cloud wurden unausweichlich. [34]

Der Wendepunkt war die Einführung von *DevOps* als neuer Industriestandard zur Entwicklung von Cloud Software. DevOps fördert durch Leitlinien eine Kultur der funktionsübergreifenden Zusammenarbeit zwischen Entwicklern von Software (***Dev****elopers*) sowie deren Bereitstellung in der Cloud durch IT-Teams (***Op****erators*). *Continuous Deployment*, kurz *CDP* implementiert die DevOps Leitlinien mit Fokus auf Automatisierung. CDP operationalisiert damit die Entwicklung von Cloud Software. Wie Abbildung 15 zeigt, transformiert die sogenannte *Continuous Deployment Pipeline*, kurz *CDP Pipeline*, Codeänderungen in ein auslieferbares Software-Produkt für die Produktivumgebung. Nach teamübergreifender Validierung erfolgt die automatische Bereitstellung der Software auf die Cloud-Infrastruktur, wo sie kontinuierlich überwacht wird. Rückkopplungen zwischen den Teams sowie die Rückführung neuer Stakeholder-Anforderungen ermöglichen schnelle Reaktionen der Entwickler. [31, 34]

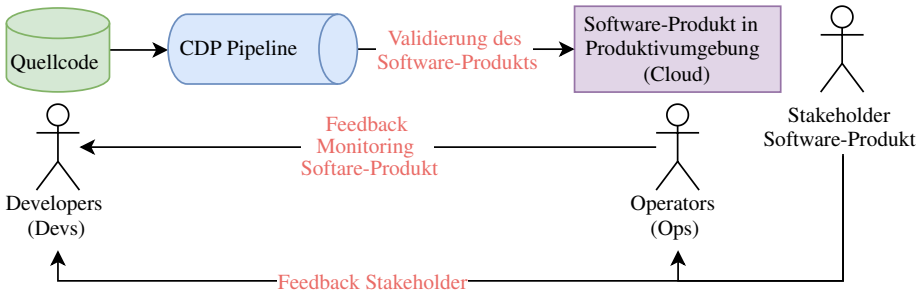

Figure 15. Continuous Deployment als wissenschaftliche Methode

Biswas et al. argumentieren, dass sich Continuous Deployment wegen seines Erfolgs in der Praxis zur Operationalisierung von Cloud-Softwareentwicklung gleichermaßen als Werkzeug zur Operationalisierung von DS Pipelines nutzen lässt [3]. Damit lässt sich folgende These formulieren: **CDP angewendet auf DS Pipelines unter Erfüllung der wissenschaftlichen Methode entspricht einem DS-Prozess.**

$$\begin{aligned} &\textbf{These:} && \textit{CDP} + \textit{DS Pipeline} + W \mathrel{\widehat{=}} \textit{DS-Prozess} \\ &\textbf{Wobei gilt:} && W = \textit{Merkmale der wiss. Methode} \end{aligned} \tag{1}$$

4.2 Operationalisierung von DS Pipelines mit MLOps

Wie einleitend beschrieben, scheitern die meisten Projekte zur Herstellung von ML-Produkten an der fehlenden Operationalisierung von DS Pipelines. So schaffen es 90% der ML-Modelle nicht bis zur Produktivumgebung und liefern keinen finanziellen Wert [13, 33]. Jedoch sehen Hummer et al. bei der Operationalisierung von DS Pipelines zur Entwicklung von ML-Produkten folgendes Dilemma: Einerseits muss betriebsintern eine Arbeitsweise mit teamübergreifender Kommunikation und Validierung sichergestellt werden, andererseits benötigen Data-Scientists zur Entwicklung des ML-Modells einen isolierten, experimentellen Freiraum [19]. Wie anhand des Beispiels zur Vorhersage des Kraftstoffverbrauchs in Kapitel 3 gezeigt, handelt es sich bei einer DS Pipeline um einen hochdynamischen und iterativen Prozess innerhalb des Data Science Teams.

Um dieses Dilemma der Operationalisierung von DS Pipelines zu lösen, entstand das in Abbildung 16 dargestellte Framework *MLOps* [16, 32]. MLOps nutzt CDP, um durch zwei DS Pipelines mit gleichem Datensatz sowie deren Verknüpfung durch ein Quality Gate den DS-Prozess als wissenschaftliche Methode umzusetzen [14, 23].

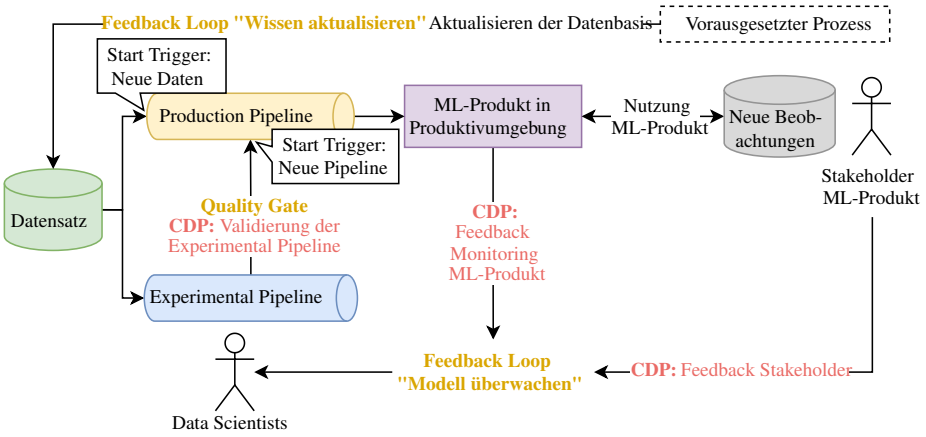

Figure 16. Operationalisierung von DS Pipelines mit MLOps

Experimental Pipeline: Diese DS Pipeline ermöglicht den Data Scientists eine dynamische Entwicklung des ML-Modells. Getrennt von der Produktivumgebung können dort neue DS-Pipeline-Stufen und ML-Algorithmen getestet werden. [23, 32]

Production Pipeline: Um ein ML-Produkt für die Produktivumgebung zu erzeugen, durchläuft die Production Pipeline die Schritte einer replizierten Experimental Pipeline. Das entstehende ML-Produkt wird automatisch in der Produktivumgebung eingesetzt. Ausgelöst wird diese Pipeline, wenn eine neue Experimental Pipeline übertragen wurde und der Datensatz mit neuen Daten aktualisiert wurde. [32]

Quality Gate: Bevor die Änderungen der Experimental Pipeline in die Production Pipeline übertragen werden, muss ein Antrag auf Validierung gestellt werden. Das stellt die ausschließliche Bereitstellung validierter ML-Produkte im Sinne von

CDP sicher. Neben Human-in-the-Loop-Validierungen wie Peer-Reviews, können automatische Tests durchgeführt werden, beispielsweise um den Sprachfilter eines Sprachmodells zu überprüfen. [32]

Feedback Loop "Modell überwachen": Der Betrieb des ML-Produkts der Produktivumgebung wird im Sinne von CDP kontinuierlich von den Data Scientists überwacht, beispielsweise, um die Überanspruchung von Hardware-Ressourcen festzustellen und zu beheben. Ebenso müssen die Data Scientists auf das Feedback der Stakeholder des ML-Produkts reagieren. [32, 35]

Feedback Loop "Wissen aktualisieren": Die Aktualisierung der Daten setzt MLOps als einen von den Anwendern umzusetzenden Prozess voraus. Es wird sowohl bei der Experimental als auch bei der Production Pipeline stets von einem aktuellen Datensatz ausgegangen. [16, 32]

Aus einer Industriestudie von Shankar et al. geht hervor, dass MLOps in Unternehmen erfolgreich angewendet wird [32]. So bezeichnet Ericsson MLOps als DS-Prozess, der durch die Experimental und Production Pipeline das Dilemma der Operationalisierung von DS Pipelines löst [10]:

$$\textbf{In der Praxis (Ericsson) gilt:} \quad \textit{MLOps} \mathrel{\widehat{=}} \textit{DS-Prozess} \tag{2}$$

Anhand dieser Feststellung aus der Praxis lässt sich die im vorherigen Abschnitt 4.1 aufgestellte These (1) prüfen:

Definition MLOps:	*MLOps*
	$:=$ *CDP*
	$+$ *Experimental Pipeline*
	$+$ *Production Pipeline*
	$+$ *W*
(2)	$\widehat{=}$ *DS-Prozess*
Wobei gilt:	*W = Feedback Loop "Wissen aktualisieren"*

Somit gilt die aufgestellte These (1) unter folgenden Bedingungen:

MLOps entspricht CDP mit Experimental sowie Production Pipeline und ermöglicht unter Voraussetzung des Feedback Loops "Wissen aktualisieren" die Operationalisierung von DS Pipelines als DS-Prozess.

4.3 Praktische Umsetzung von MLOps

Beispiele für ML-Produkte sind deskriptive ML-Modelle zur Preisoptimierung im Vertrieb, welche Kundensegmente unterschiedlicher geografischer Regionen identifizieren [4]. Auch cyber-physikalische Systeme können ML-Produkte sein, beispielsweise wenn ein Fahrzeug-Steuergerät mit einem geeigneten Machine-Learning-Modell automatisch Fußgänger erkennt [5]. Nachfolgend wird mit MLOps die DS Pipeline aus Kapitel 3 zur Vorhersage des Kraftstoffverbrauchs als sogenanntes "*Inference-as-a-Service*"-ML-Produkt operationalisiert [29]: Der Kunde kann über eine HTTP-Schnittstelle die Fahrzeugdaten in Gewicht und Baujahr angeben. Das prädiktive ML-Modell liefert in der Cloud eine Vorhersage zum Verbrauch in Meilen pro Gallone. Der Quellcode wurde vom Autor dieser Seminararbeit auf GitHub veröffentlicht [30]. Die Architektur der Implementierung ist in Abbildung 17 dargestellt und wird nachfolgend vollständig beschrieben:

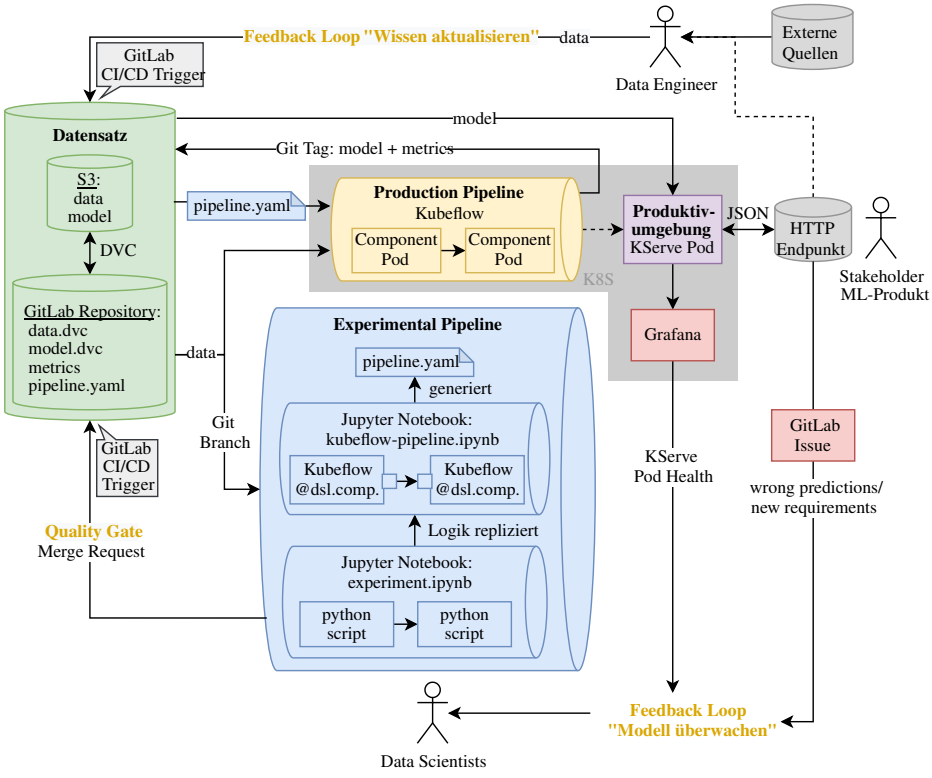

Figure 17. Implementierung von MLOps: Operationalisierung der DS Pipeline zur Vorhersage des Kraftstoffverbrauchs aus Kapitel 3

Datensatz (grün): Als initialer Datensatz wird die als `data` referenzierte CSV-Datei mit Fahrzeugdaten der statistischen Erhebung von 1983 verwendet. Um ein realitätsnahes

System abzubilden, wird diese Datei nicht direkt in einem Git Repository gespeichert, sondern in einem leistungsfähigen S3-Cloud-Speicher. Für diese Implementierung wurde AWS als Anbieter gewählt.

Aufgrund des Feedback Loops “Wissen aktualisieren“ wird sich der Datensatz ändern. Um die Versionierung der Daten sicherzustellen, wird *Data Version Control*, kurz *DVC* genutzt. Diese Erweiterung für Git Repositories erzeugt für entfernt gespeicherte Daten leichtgewichtige `.dvc`-Dateien mit Metainformationen, wie die aktuelle Prüfsumme. Diese Datei wird in einem GitLab Git Repository versioniert. Bei Änderung der entfernten Daten im S3-Speicher, ändert sich ebenfalls die DVC-Datei. Zur Vereinfachung dieser Implementierung werden im Datensatz sowohl Daten als auch entwickelte ML-Modelle gespeichert. In der Architekturskizze sind die Bestandteile des Datensatzes grün dargestellt.

Production Pipeline (gelb): Die Anwendung *Kubeflow* ermöglicht die Ausführung von DS Pipelines, den *Kubeflow Pipelines*, auf Kubernetes Clustern. Wie exemplarisch in Abbildung 18 gezeigt, werden die DS-Pipeline-Stufen als *Kubeflow-Komponenten* mit dem Kubeflow Python Framework definiert. Die Kubeflow Pipeline wird aus diesen Komponenten als YAML-Datei exportiert. Kubeflow läuft als Anwendung auf einem Kubernetes Cluster. Über die Kubeflow API werden die Komponenten der YAML-Datei als Pods auf dem Cluster gestartet. Eine grafische Benutzeroberfläche stellt die Abhängigkeiten der *Komponenten-Pods* dar.

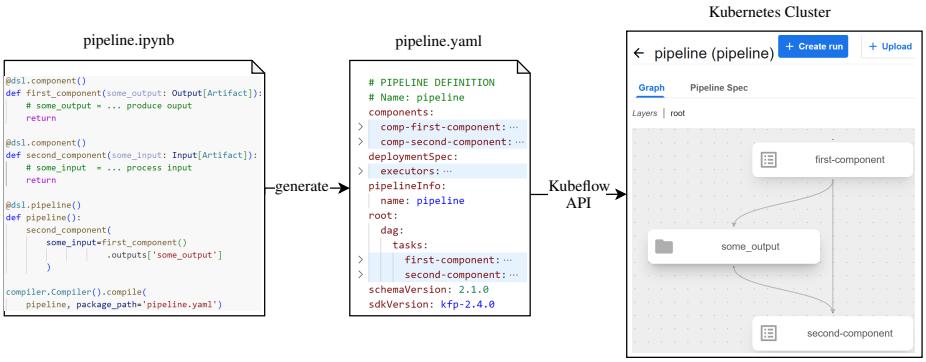

Figure 18. Eine DS Pipeline bestehend aus zwei Komponenten wird mit dem Kubeflow Framework in Python abgebildet (links), wodurch eine YAML-Datei generiert wird (Mitte), welche auf einem Kubernetes Cluster über die Kubeflow API ausgeführt wird (rechts)

Die Kubeflow API erwartet im Rahmen dieser Implementierung die Datei `pipeline.yaml`. Die daraus generierten Komponenten-Pods auf einem Microsoft Azure Kubernetes Cluster beziehen den Datensatz aus dem S3-Speicher. Nachdem alle Komponenten-Pods ausgeführt wurden, wird das Modell `model` ebenfalls über DVC in den S3-Speicher geladen. Dazu wird ein neuer Git Tag im Repository erstellt, welcher neben dem neuen Modell die dazugehörigen Metriken `metrics` der Stufe “Evaluation“ umfasst. In der Architekturskizze sind die Bestandteile der Production Pipeline gelb dargestellt.

Bereitstellung des ML-Produkts (violett): Um das ML-Modell als "Inference-as-a-Service"-ML-Produkt zur Verfügung zu stellen, erzeugt der letzte Komponenten-Pod einen sogenannten *KServe Pod*. Mithilfe des Frameworks *KServe* wird das Modell aus dem S3-Speicher geladen sowie über eine HTTP-Schnittstelle **(grau)** öffentlich zur Verfügung gestellt. Kunden können nun HTTP-Anfragen an die öffentliche IP des KServe Pods stellen. Dazu müssen die Werte der im "Feature Engineering" ausgewählten Variablen (siehe Kapitel 3) in Form eines JSON Request Bodys enthalten sein. In der Architekturskizze sind die Bestandteile der Bereitstellung des ML-Produkts violett dargestellt.

Wie Abbildung 19 zeigt, liefert der KServe Pod für eine HTTP-Anfrage mit den Fahrzeugdaten eines 1976 gebauten Audi 100 C2 mit einem Gewicht von 2448 lbs (1110 kg) eine Vorhersage für den Kraftstoffverbrauch von ca. $25,7$ Meilen pro Gallone ($9,15$ l/100km). Die Herstellerangabe liegt bei $25,02$ mpg ($9,4$ l/100 km)

```python
import requests
import json
from sklearn.preprocessing import PolynomialFeatures

audi_100_c2 = [
    [2448, # Gewicht in Pfund (1110 kg)
     76]   # Baujahr Modulo 10 (Produktionsbeginn 1976)
    ]

transformed_input = PolynomialFeatures(degree=2).fit_transform(audi_100_c2).tolist()
inference_input = {
  'instances': transformed_input
}

response = requests.post(
  "http://20.22.254.98/kserve/v1/models/sklearn-mpg:predict",
  json=inference_input,
  headers={"Host": "sklearn-mpg.kserve-deploy-test.example.com"}
  )
print(response.text)
```

✓ 0.2s

```
{"predictions":[25.757758948674763]}
```

Figure 19. "Inference-as-a-Service": Eine HTTP-Anfrage an den KServe Pod mit dem Gewicht und Baujahr eines Audi 100 C2 liefert eine Vorhersage von 25,7 Meilen pro Gallone (9,15 l/100km) für den Kraftstoffverbrauch.

Experimental Pipeline (blau): Das Kubeflow Framework ermöglicht keine lokale Ausführung der Komponenten, sondern nur die Erzeugung der YAML-Datei. Deshalb müssen alle Experimente der Data Scientists in einem lokalen Jupyter Notebook `experiment.ipynb` durchgeführt werden, so wie es beispielhaft in Kapitel 3 vorgestellt wurde. Der Datensatz wird dabei aus dem S3-Speicher bezogen. Um die Schritte der Experimental Pipeline in der Production Pipeline mit Kubeflow ausführen zu können, muss der Autor des Experiments die DS Pipeline-Schritte manuell auf das Kubeflow Python Framework abbilden. Das daraus entstandene Jupyter Notebook `kubeflow-pipeline.ipynb` wird zusammen mit der daraus generierten `pipeline.yaml` in einem neuen Branch des GitLab Repositories versioniert. In der Architekturskizze sind die Bestandteile der Experimental Pipeline blau dargestellt.

Abbildung 20 zeigt exemplarisch anhand der DS-Pipeline-Stufe "Modellerzeugung" den Unterschied zwischen dem experimentellen Jupyter Notebook und der Syntax des Kubeflow Frameworks.

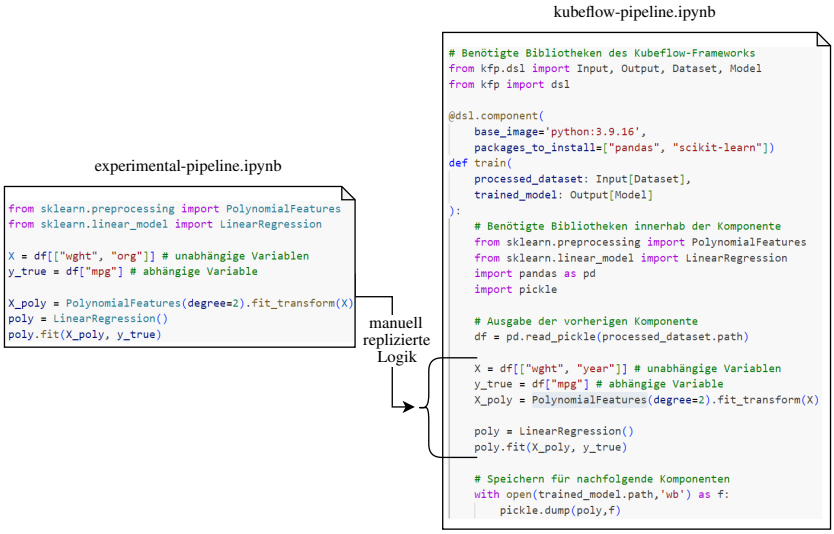

Figure 20. Um die Änderungen in Kubeflow ausführen zu können, muss die Logik des experimentellen Jupyter Notebooks (links) in die Syntax des Kubeflow Frameworks übertragen werden (rechts).

Quality Gate Bevor die generierte `pipeline.yaml` der Experimental Pipeline an die Kubeflow API übergeben wird, müssen die Änderungen gemäß MLOps teamübergreifend validiert werden. Dazu eröffnet der Data Scientist in GitLab einen *Merge Request* für die Änderungen seines Experiments. Nachdem der Merge Request teamübergreifend akzeptiert wurde, übergibt ein GitLab CI/CD Runner automatisch die validierte `pipeline.yaml` der Kubeflow API. Dieser in Abbildung 21 dargestellte Prozess implementiert gemäß MLOps das Quality Gate zur Übertragung der Experimental Pipeline in die Production Pipeline.

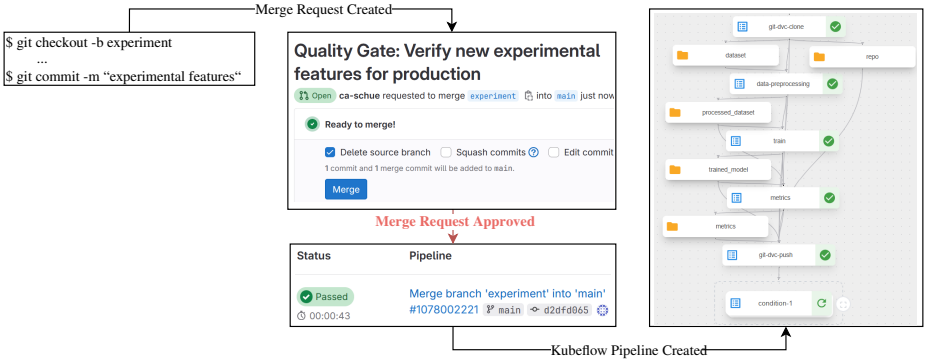

Figure 21. Jedes Experiment muss durch einen Merge Request validiert werden, bevor es als Kubeflow Pipeline bereitgestellt wird.

Feedback Loop "Wissensstand aktualisieren": MLOps setzt einen Prozess zur kontinuierlichen Aktualisierung des Datensatzes voraus. In dieser Implementierung findet die

Aktualisierung manuell durch einen Data Engineer statt. Dieser überträgt beispielsweise Informationen über neue Automodelle aus externen Quellen **(grau)** in die CSV-Datei und lädt sie in den S3-Speicher. Daraufhin bringt ein GitLab CI/CD Runner die aktuelle `pipeline.yaml`-Datei zur Ausführung. Die Komponenten-Pods erzeugen aus dem aktualisierten Datensatz ein neues ML-Modell, welches automatisch in der Produktivumgebung als ML-Produkt mit KServe bereitgestellt wird.

Feedback Loop "Modell überwachen": Die durch CDP geforderte Überwachung des ML-Produkts wird über Health Monitoring des KServe Pods durch *Grafana* **(rot)** umgesetzt. Somit erhalten die Data Scientists Live-Informationen über die Anzahl der Zugriffe auf das Modell oder die in Abbildung 22 gezeigte durchschnittliche Antwortzeit des KServe Pods.

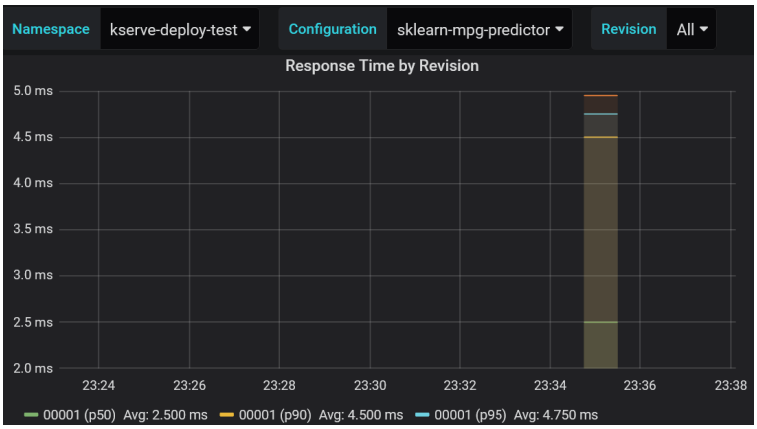

Figure 22. Mit Grafana kann die durschnittliche Antwortzeit des KServe Pods auf HTTP-Anfragen angezeigt werden.

Wie Abbildung 23 zeigt, wird Stakeholder Feedback über *Issues* **(rot)** in GitLab zurückgeführt. Die Data Scientists koordinieren selbstständig die Bearbeitung der Issues, beispielsweise im Rahmen des Scrum-Vorgehensmodells.

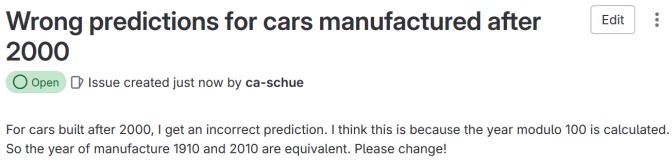

Figure 23. Stakeholder erstellen bei erkannten Ungenauigkeiten ein Issue in GitLab. Hier beispielsweise für Modellungenauigkeiten bei Autos ab dem Baujahr 2000.

4.4 MLOps mit DataOps

MLOps setzt den Feedback Loop “Wissen aktualisieren“ als organisatorischen Prozess voraus. In der gezeigten Implementierung wurde dieser Prozess vereinfacht durch einen einzelnen Data Engineer umgesetzt. In Unternehmen erfolgt das Sammeln und Analysieren von Daten sowie deren Aufbereitung für verschiedene Zielgruppen meist in einer eignen Einheit, der sogenannten *Business Intelligence*, kurz *BI*. [2,9] *BI Use Cases* umfassen beispielsweise die Überwachung interner Prozesse sowie die Analyse von Kundenverhalten. Fragmentierte, heterogene Datenlandschaften sowie exponentiell zunehmende Datentypen und Datenquellen erfordern dabei eine flexible Verarbeitung. [1,2]

Sogenannte *ETL Workflows* ermöglichen die **E**xtraktion von Daten aus verschiedenen Quellen, die **T**ransformation in geeignete Datenformate sowie das **L**aden in zentrale Speicherorte [25, 36]. Das entspricht gemäß Kapitel 3 einer azyklischen DS Pipeline ohne Modellbildungsebene – nicht ein ML-Modell ist das ML-Produkt, sondern die DS Pipeline selbst.

DS Pipeline als ML-Produkt: $\textit{ML-Produkt} = \textit{ETL Workflow} = \textit{DS Pipeline}$ (3)

ETL Workflows dienen stets einem übergeordnetem BI Use Case, beispielsweise dem Feedback Loop “Wissensstand aktualisieren“ von MLOps. Wie in Abbildung 24 dargestellt, ergänzen ETL Workflows an dieser Stelle MLOps.

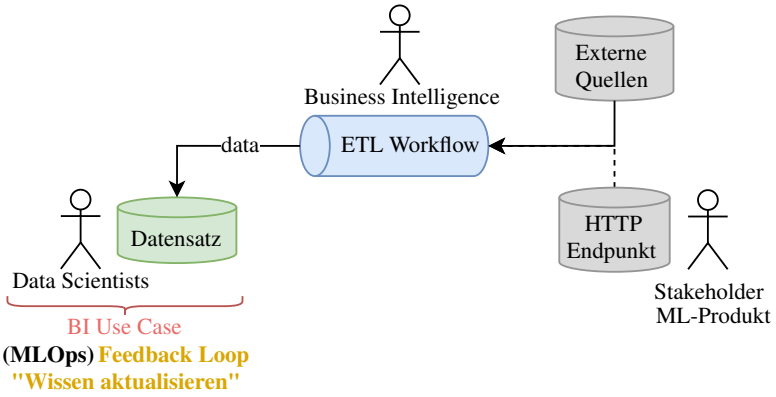

Figure 24. Mögliche Implementierung des Feedback Loop “Wissen aktualisieren“ als ETL Workflow

Jedoch scheitern laut einer Industriestudie von Munappy et al. Projekte zur Entwicklung von ETL Workflows aufgrund von mangelndem Feedback und fehlender Qualitätssicherung. Das zugrundeliegende Problem liegt in der azyklischen Natur der ETL Workflows als DS Pipelines ohne Einbettung in einen DS-Prozess. Der Lösungsvorschlag von Munappy et al. sieht deshalb analog zur Operationalisierung von ML-Produkten, wie in Kapitel 4 dargestellt, die Operationalisierung von ETL Workflows vor. [25]

Abbildung 25 zeigt *DataOps*, eine Anwendung von MLOps auf ETL Workflows [22]:

- Der "**Experimental Workflow**" dient der Entwicklung neuer Funktionen für die ETL-Schritte.
- Der "**Production Workflow**" dient der Bereitstellung von Daten für den BI Use Case in der Produktivumgebung.

DataOps ermöglicht die Operationalisierung von ETL Workflows zur Datenbereitstellung für einen BI Use Case, beispielsweise der Feedback Loop "Wissen aktualisieren" in MLOps. Die Datenbereitstellung in der Produktivumgebung wird über den Feedback Loop "Modell überwachen" vom Data Engineer überwacht. Sowohl Experimental als auch Production Workflow verwenden bereits aktuelle Daten des Unternehmens. Somit entfällt bei DataOps der Feedback Loop "Wissen aktualisieren".

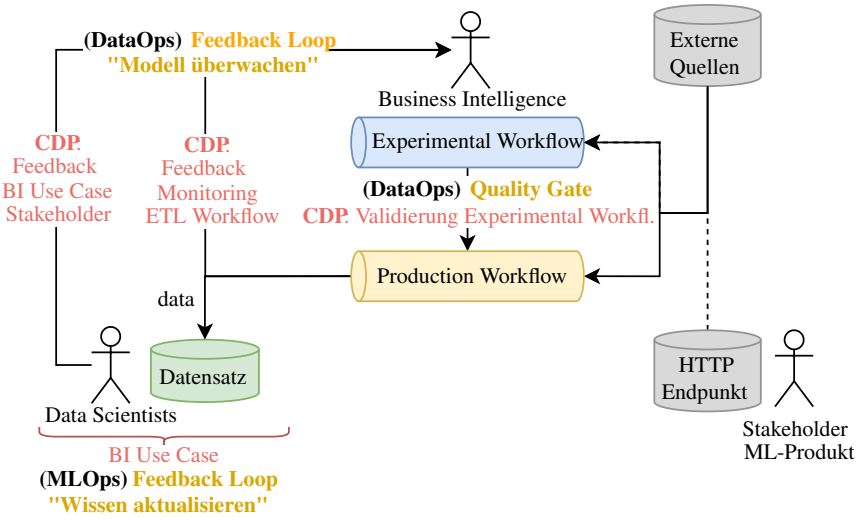

Figure 25. DataOps zur Operationalisierung von ETL Workflows

Ericsson betont, dass sich DataOps als vollständiger, zyklischer DS-Prozess für die Entwicklung von ETL Workflows erwiesen hat [10]. Ein beliebtes Tool zur Implementierung ist dabei Apache Airflow [24, 25]:

Anhand dieser Feststellung aus der Praxis lässt sich die im vorherigen Abschnitt 4.1 aufgestellte These (1) prüfen:

$$\textbf{In der Praxis (Ericsson) gilt:} \quad \textit{DataOps} \mathrel{\widehat{=}} \textit{DS-Prozess} \tag{4}$$

Anhand dieser Aussage aus der Praxis zu DataOps lässt sich analog zu MLOps die im Kapitel 4 formulierte These (1) überprüfen

Definition DataOps: *DataOps*

$:=$ *MLOps für ETL Workflows*

$-$ *Feedback Loop "Wissen aktualisieren"*

$:=$ *CDP*

$+$ *Experimental Workflow*

$+$ *Production Workflow*

(4) $\widehat{=}$ *DS-Prozess*

Somit gilt die aufgestellte These (1) unter folgenden Bedingungen:

DataOps entspricht CDP mit Experimental sowie Production Workflow und ermöglicht die Operationalisierung von ETL Workflows als DS-Prozess.

4.5 Zusammenfassung

Abschließend sind die Zusammenhänge zwischen der wissenschaftlichen Methode, dem Data-Science-Prozess sowie dessen Umsetzung durch MLOps und DataOps in Tabelle 2 aufgelistet: DataOps wird oft als *Enabler* für MLOps genannt: Durch DataOps als DS-Prozess zur Aktualisierung des Datensatzes ist die Operationalsierung von DS Pipelines für ML-Produkte mit MLOps möglich [22, 25].

Tabelle 2. Abbildung der wissenschaftlichen Methode durch den Data-Science-Prozess sowie dessen Umsetzung durch MLOps und die MLOps Variante DataOps.

Wissenschaftliche Methode	**Data-Science-Prozess**	**MLOps**	**DataOps (Variante von MLOps)**
Feedback Loop "Wissen aktualisieren"	Datensatz aktualisieren	Wird vorausgesetzt (Enabler: DataOps)	Entfällt
Feedback Loop "Modell überwachen"	Abweichung ML-Modell von Metriken	**CDP:** Feedback Monitoring ML-Produkt + Feedback Stakeholder ML-Produkt	**CDP:** Feedback Monitoring ETL Workflow + Feedback Stakeholder BI Use Case
Quality Gate	Validierung ML-Modell	**CDP:** Validierung Experimental Pipeline	**CDP:** Validierung Experimental Workflow

Chapter 5

Fazit

In dieser Arbeit wurde Data Science als wissenschaftliche Methode definiert. Anschließend wurden die Phasen des Data-Science-Prozesses vorgestellt, mit Fokus auf Data Science Pipelines zur Erzeugung eines Modells maschinellen Lernens. Anhand eines Beispiels zur Vorhersage des Kraftstoffverbrauchs wurden die Stufen einer DS Pipeline demonstriert.

Im Anschluss wurden auftretende Probleme aufgezeigt, wenn Unternehmen lediglich Data Science Pipelines zur Entwicklung von Machine-Learning-Produkten verwenden, ohne Einbettung in einen DS-Prozess. Mit Continuous Deployment (CDP) wurden Merkmale vorgestellt, wie Unternehmen die Entwicklung von Cloud Software in einem teamübergreifenden Prozess koordinieren können. Es wurde die These formuliert, dass diese sogenannte *Operationalisierung* der Cloud-Softwareentwicklung mithilfe von CDP zur teamübergreifenden Entwicklung von ML-Produkten anwendbar ist.

Anhand des Frameworks MLOps zur Operationalisierung von DS Pipelines wurde diese These bestätigt. Anschließend wurde durch eine umfassende Implementierung die praktische Umsetzbarkeit von MLOps demonstriert. Schließlich wurde diskutiert, wie DataOps als Variante von MLOps die Entwicklung von DS Pipelines zur Steuerung von Informationsflüssen operationalisiert. Die Synergie von MLOps und DataOps ermöglicht Unternehmen – letztendlich unter Umsetzung aller Charakteristika der wissenschaftlichen Methode – den Sprung von isolierten DS Pipelines zu DS-Prozessen.

Dieser Prozess kann selbst als Iteration der wissenschaftlichen Methode in einem übergeordneten Kontext betrachtet werden: Das Modell entspricht hierbei der Vorgehensweise von Unternehmen zur Umsetzung von Data Science. Der ursprüngliche Status Quo – die Entwicklung von ML-Produkten in Unternehmen durch isolierte DS Pipelines – wurde als Quelle des Scheiterns identifiziert. Aus dieser Erkenntnis heraus wurde MLOps als neues Modell entwickelt. Als Quality Gate vor der Anwendung in Projekten wird überprüft, ob MLOps als neues Modell die Merkmale des DS-Prozesses erfüllt.

Die wissenschaftliche Methode endet jedoch nicht mit einem neuen Status Quo, sondern fördert aufgrund ihres zyklischen Kerns mit “Feedback Loops“ die kontinuierliche Entwicklung neuer Modelle. So wurde Einsteins deterministisches Modell in der zweiten Hälfte des 20. Jahrhunderts durch neue Erkenntnisse der Quantenphysik ergänzt – validiert durch sogenannte Bell-Tests mit verschränkten Photonen [7]. Analog wurde bei MLOps die Datenrückführung durch DataOps ergänzt – validiert durch die Erfüllung der Eigenschaften des DS-Prozesses.

Literatur

1. Atwal, H.: Practical DataOps: Delivering agile data science at scale. Springer (2019)
2. Bergh, C., Benghiat, G., Strod, E.: The dataops cookbook. DataKitchen Hqrs (2019)
3. Biswas et al.: The art and practice of data science pipelines: A comprehensive study of data science pipelines in theory, in-the-small, and in-the-large. In: 44th International Conference on Software Engineering. p. 2091–2103 (2022)
4. Blog, N.T.: Reinforcement learning for budget constrained recommendations (2022), https://netflixtechblog.com/reinforcement-learning-for-budget-constrained-recommendations-6cbc5263a32a, letzter Zugriff am 07.12.2023
5. Bosch Global: Künstliche Intelligenz: Automatisiertes Fahren: Kamera läuft!, https://www.bosch.com/de/stories/mpc3-kamera-autonomes-fahren/, letzter Zugriff am 07.12.2023
6. Capizzi, A., Distefano, S., Mazzara, M.: From devops to devdataops: Data management in devops processes. In: Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment: Second International Workshop. pp. 52–62 (2020)
7. Christine Sutton: Fifty years of bell's theorem (2014), https://home.cern/news/news/physics/fifty-years-bells-theorem, letzter Zugriff am 07.12.2023
8. Deutschlandfunk Kultur: 100 Jahre Allgemeine Relativitätstheorie - Einsteins größter Geniestreich (2015), https://www.deutschlandfunkkultur.de/100-jahre-allgemeine-relativitaetstheorie-einsteins-100.html, letzter Zugriff am 07.12.2023
9. Ereth, J.: Dataops-towards a definition. LWDA **2191**, 104–112 (2018)
10. Ericsson: Cognitive networks: towards a 6g architecture (2022), https://www.ericsson.com/en/blog/2022/1/cognitive-networks-6g-architecture, letzter Zugriff am 07.12.2023
11. Garlan, D., Shaw, M.: Architectures for software systems: A curriculum development proposal in undergraduate software engineering. CMU School of Computer Science (1993)
12. Gartner: Organizations are slow to advance in d & a | gartner (2018), https://www.gartner.com/en/newsroom/press-releases/2018-02-05-gartner-survey-shows-organizations-are-slow-to-advance-in-data-and-analytics, letzter Zugriff am 07.12.2023
13. Gartner: Gartner survey reveals less than half of data and analytics teams effectively provide value to the organization (2023), https://www.gartner.com/en/newsroom/press-releases/03-21-2023-gartner-survey-reveals-less-than-half-of-data-and-analytics-teams-effectively-provide-value-to-the-organization, letzter Zugriff am 07.12.2023
14. Giray, G.: A software engineering perspective on engineering machine learning systems: State of the art and challenges. Journal of Systems and Software **180**, 111031 (2021)
15. Gomez-Uribe et al.: The netflix recommender system: Algorithms, business value, and innovation. ACM Trans. Manage. Inf. Syst. **6**(4) (2016)
16. Google Cloud Architecture Center: Mlops: Continuous delivery and automation pipelines in machine learning. Google Cloud AI and machine learning resources (2023), https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning, letzter Zugriff am 07.12.2023
17. Handl, A., Kuhlenkasper, T.: Einführung in die Statistik: Theorie und Praxis mit R. Springer-Verlag (2018)

18. Heudecker, N.: We were too conservative. the failure rate is closer to 85%. and the problem isn't technology. (2017), https://web.archive.org/web/20180923065705/https://twitter.com/nheudecker/status/928720268662530048, letzter Zugriff über web.archive.org am 07.12.2023
19. Hummer et al.: Modelops: Cloud-based lifecycle management for reliable and trusted ai. In: 2019 IEEE International Conference on Cloud Engineering (IC2E). pp. 113–120 (2019)
20. Jöhnk, J., Weißert, M., Wyrtki, K.: Ready or not, ai comes—an interview study of organizational ai readiness factors. Business & Information Systems Engineering **63**, 5–20 (2021)
21. Kaggle: How to use kaggle: Competitions: Find challenges for every interest level (2023), https://www.kaggle.com/docs/competitions, letzter Zugriff am 07.12.2023
22. Mainali et al.: Discovering dataops: a comprehensive review of definitions, use cases, and tools. In: DATA ANALYTICS 2021 The Tenth International Conference on Data Analytics (2021)
23. Mäkinen et al.: Who needs mlops: What data scientists seek to accomplish and how can mlops help? In: 2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN). pp. 109–112 (2021)
24. Moreschini, S., Hästbacka, D., Taibi, D.: Mlops pipeline development: The ossara use case. In: 2023 International Conference on Research in Adaptive and Convergent Systems. pp. 1–8 (2023)
25. Munappy et al.: From ad-hoc data analytics to dataops. In: 2020 International Conference on Software and System Processes (ICSSP). p. 165–174. Association for Computing Machinery (2020)
26. OpenAI: Research: Generative models (2016), https://openai.com/research/generative-models, letzter Zugriff am 07.12.2023
27. Quinlan, R.: Auto MPG. UCI Machine Learning Repository (1993), DOI: 10.24432/C5859H
28. Raina, V., Krishnamurthy, S.: Building an effective data science practice. Apress BerNeley (2021)
29. Romero et al.: Infaas: Automated model-less inference serving. In: 2021 USENIX Annual Technical Conference (USENIX ATC). pp. 397–411 (2021)
30. Schünemann, C.: MLOps implementation for DS seminar, https://github.com/ca-schue/mlops-demo.git, letzter Zugriff am 07.12.2023
31. Serban et al.: Adoption and effects of software engineering best practices in machine learning. In: 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). pp. 1–12 (2020)
32. Shankar et al.: Operationalizing machine learning: An interview study. arXiv:2209.09125 (2022)
33. Staff, V.: Why do 87% of data science projects never make it into production? (2019), https://venturebeat.com/ai/why-do-87-of-data-science-projects-never-make-it-into-production/, letzter Zugriff am 07.12.2023
34. Ståhl, D., Martensson, T., Bosch, J.: Continuous practices and devops: beyond the buzz, what does it all mean? In: 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA). pp. 440–448 (2017)
35. Subramanya, R., Sierla, S., Vyatkin, V.: From devops to mlops: Overview and application to electricity market forecasting. Applied Sciences **12**(19), 9851 (2022)
36. Zhang, Z.: Devops for data science system. KTH School of Electrical Engineering and Computer Science (2020)