HANSER


# Leseprobe

zu

# „Handbuch Data Science“

von Stefan Papp, Wolfgang Weidinger,
Mario Meir-Huber et al.

Print-ISBN: 978-3-446-45710-2
E-Book-ISBN: 978-3-446-45975-5

Weitere Informationen und Bestellungen unter
http://www.hanser-fachbuch.de/978-3-446-45710-2
sowie im Buchhandel

© Carl Hanser Verlag, München

# Inhalt

# Vorwort

Künstliche Intelligenz, Machine Learning, Predictive Analytics, Big Data, Smart Data – das sind nur einige der Hype- und Buzzwords, die in den letzten Jahren durch die Medien gegeistert sind und einerseits große Fragezeichen, andererseits aber auch große Erwartungen ausgelöst haben.

Eng damit verbunden ist einer der wichtigsten Trends der letzten und wohl auch der kommenden Jahre: die Digitalisierung. Diese wird nicht nur in IT-Abteilungen, sondern auch in Vorstandssitzungen diskutiert. Die Vorstände sämtlicher großer Unternehmen haben die Digitalisierung als Kernthema in ihre Geschäftsstrategie aufgenommen. Doch ohne die Verfügbarkeit von Daten kommen Digitalisierungsinitiativen sehr schnell zum Erliegen. Und selbst wenn man Daten sammelt, ist dies kein Garant für Erfolg, denn diese müssen auch in hoher Qualität vorliegen.

Mit Daten die eigene Organisation neu erfinden? Klingt gewagt, aber um nichts anderes geht es bei Data Science. Wissenschaft (Science) zielt darauf ab, neues Wissen zu generieren. Und neues Wissen kann neue Geschäftsmodelle ermöglichen.

Dieses Buch ist angetreten, sowohl verständliche Antworten auf die vorhandenen Fragen zu geben als auch die überschießenden Erwartungen einzufangen. Dies passiert, indem wir es Ihnen ermöglichen möchten den Motor der Digitalisierung – die Daten – nicht nur zu verstehen, sondern auch mit diesen zu arbeiten. Zum Beispiel durch die Darstellung ausgewählter Analysetechniken, aber auch durch tiefes Eintauchen in technologische Detailfragen, um zu zeigen, wie aus Daten Erkenntnisse geschaffen werden, die dem an sie gestellten Anspruch gerecht werden können.

Das Ziel dabei ist es, Ihnen ein realistisches Bild der Möglichkeiten von Data Science jenseits des allgegenwärtigen Hypes zu geben.

Denn Data Science ist die Wissenschaft, die den eingangs genannten Begriffen und Buzzwords übergeordnet ist. Sie setzt sich aus drei großen Strömungen zusammen:

- Computer Science/IT
- Mathematik/Statistik
- Domainexpertise in jenen Branchen, in denen Data Science Anwendung findet

Das macht Data Science zu einer interdisziplinären Wissenschaft, in der sich eine sehr heterogene Schar von Spezialisten bewegt, die alle ein Ziel verfolgen:

**Verständliche Antworten auf Fragen anhand von Daten geben.**

Wie später noch im Detail ausgeführt wird, kann diese Aufgabe sehr häufig nur von einem Team aus Data Scientists mit verschiedenen Spezialisierungen bewältigt werden.

Das war einer der Gründe für die Entstehung der Vienna Data Science Group (VDSG): wir wollten einen neutralen Ort schaffen, an dem interdisziplinärer Austausch von Wissen zwischen allen involvierten Experten international stattfinden kann. Ein Produkt des Austausches in unserer Community ist dieses Buch, das die Vielfältigkeit und Diversität der Mitglieder sowohl in deren Ausbildung als auch in ihrer Branchenerfahrung widerspiegelt. In Kapitel 9 ist das am besten illustriert, da dort die Erfahrung weiterer VDSG-Mitglieder neben den Buchautoren eingeflossen ist.

Diese gemeinnützige Gesellschaft ist eine NPO und NGO, die sich die Entwicklung des gesamten Data-Science-Ökosystems (Ausbildungen, Zertifizierungen, Standardisierungen, Untersuchung der gesellschaftlichen Auswirkungen …) europaweit auf die Fahnen geschrieben hat. Dabei wird ganz besonderes Augenmerk auf die Bildung und Erhaltung einer diversen Data-Science-Community gelegt, da diese das Fundament für substanziellen Fortschritt darstellt. Ein vergleichbares Ziel verfolgt zum Beispiel die IEEE im Ingenieursbereich.

Die Analyse und Nutzung von Daten in den unterschiedlichsten Branchen hat das Potenzial

1. bestehende Geschäftsprozesse im Unternehmen zu verbessern und
2. völlig neue Geschäftsmodelle zu ermöglichen.

Unter Punkt 1 werden Investitionen und laufende Kosten optimiert. Durch Datenanalysen können Unternehmen erkennen, wie sich Investitionen langfristig auf den Geschäftserfolg auswirken. Telekommunikationsunternehmen könnten beispielsweise Hunderte Millionen Euro an Kosten innerhalb des 5G-Netzausbaus sparen, worauf wir in Kapitel 9 näher eingehen werden. Laufende Kosten können besser analysiert werden und oft wesentlich reduziert werden. Hierbei gibt es eine ganze Reihe an Möglichkeiten: die Vereinfachung von Prozessen, das Erkennen von Fehlern im Produktionsbetrieb, bevor diese entstehen, oder die zielgerichtete Ansprache von Kunden. Ebenso können durch zielgerichtetes Marketing auch neue Kunden und damit neue Umsätze generiert werden. Eine ausführliche Erläuterung dieser Themen innerhalb verschiedenster Branchen erfolgt ebenfalls in Kapitel 9.

Bei Punkt 2 geht es um völlig neue Ansätze, wie sie etwa große IT-Unternehmen wie Facebook oder Google nutzen, die ihr gesamtes Geschäftsmodell auf die Nutzung von Daten aufgebaut haben. Denn die fortschreitende Digitalisierung von betrieblichen Organisationen bis hin zu NGOs betrifft eben nicht nur die Auswirkungen auf Kosten und Einnahmen innerhalb des Kerngeschäftes. Es können auch neue Einnahmequellen mithilfe von Daten gewonnen werden. Zu diesem Thema wollen wir im Buch aufzeigen, was die rechtlichen Rahmenbedingungen sind (Kapitel 7), aber auch auf das Mindset, das diese Entwicklungen oft begleitet (Kapitel 10), eingehen.

*Wien, Frühjahr 2019*
*Wolfgang Weidinger*

# 1 Einleitung

*Bernhard Ortner, Stefan Papp, Mario Meir-Huber*

*„Data really powers everything that we do." – Jeff Weiner*

## 1.1 Strategie

Am Anfang der Digitalisierung steht immer eine Strategie. Hierbei muss der Chief Data Officer die Konzernstrategie in konkrete Strategiebausteine für Daten runterbrechen. Eine Konzernstrategie ist meist sehr umfangreich und setzt sich aus vielen verschiedenen Bereichen zusammen. Für einen global tätigen Autobauer kann ein Teilbereich dieser Strategie z. B. folgendermaßen formuliert werden:

> *„Unser Unternehmen will bis 2022 die Kostenführerschaft in der globalen Lieferkette einnehmen. Dies soll es uns ermöglichen, Elektromobilität in den Massenmarkt zum absolut besten Preis zu bringen. Damit wir das erreichen können, müssen Einsparungen in der Lieferkette von 20 % erreicht werden."*

In vielen Unternehmen ist der Grad an Optimierung jedoch bereits sehr hoch und damit sind solche Ziele meist nur sehr schwer erreichbar. Datenanalysen können hier wesentlich zur Zielerreichung beitragen.

Der Chief Data Officer wird sich nun an die Arbeit machen und den Prozess analysieren. Hierbei wird zusammen mit verschiedenen Bereichsleitern aus der Produktion eine Strategie erarbeitet. Dieses Vorgehen verdeutlicht bereits einen der wesentlichen Punkte im Umgang mit Daten beziehungsweise der Digitalisierung im Allgemeinen: Es ist kein reines IT-Thema, sondern es geht um die Kooperation zwischen der IT und den operativen Abteilungen.

In den folgenden Abschnitten wird als einführendes Beispiel kurz dargestellt, wie eine Strategie für die Optimierung der Wertschöpfungskette innerhalb eines Unternehmens der produzierenden Industrie aussehen könnte.

### 1.1.1 Wertschöpfungskettendigitalisierung (Value Chain Digitalization)

Analytische Verfahren bieten einige spannende Anwendungsmöglichkeiten in der Produktion. Wir stellen im Folgenden kurz Supply Chain und Predictive Maintenance vor.

### Supply Chain

Heutzutage ist die Supply Chain in der Regel globalisiert, d. h. verschiedene Waren und Vorprodukte werden in unterschiedlichen Ländern hergestellt, gefertigt oder montiert. Eine vielfältige Supply Chain besitzt ein erhöhtes Risiko, dass ein Zulieferer oder eine Komponente ausfällt. Das wiederum kann dazu führen, dass das Endprodukt nicht in der zu erwartenden Zeit fertiggestellt wird.

Gutes Supply Chain Management bietet einen strategischen Vorteil, der das Wachstum einer Firma beschleunigen kann. Wer besser und schneller als die Mitbewerber produziert, wird am Markt überleben.

Hier kann Big Data helfen, proaktiv Probleme zu erkennen. Aufkommende Streiks oder Wetterprobleme können beispielsweise über Advanced Analytics ermittelt werden. Wie das geht, beschreiben wir in den Kapiteln 6, 8 und 9 dieses Buchs. Durch die Simulation einer Supply Chain kann diversen Risiken entgegengesteuert werden, was in weiterer Folge die Just-in-Time-Produktion befeuern wird.

### Predictive Maintenance

Die vorausschauende Instandhaltung, Predictive Maintenance, versucht durch die Analyse der Maschinendaten die Ausfallzeit von Maschinen proaktiv vorherzusagen und zu minimieren. Als Voraussetzung dafür werden Sensoren benötigt, die entsprechende Daten übertragen können. Die erhobenen Daten kommen primär von Maschinensensoren (z. B. Wärme- oder Drucksensoren), aber auch externe Daten (z. B. die Temperatur oder Luftfeuchtigkeit) sind interessant, da diese in das analytische Modell einfließen können. Beispielsweise könnte eine Erkenntnis sein, dass eine Bauteilgruppe tendenziell ab 100 °C in der Produktion eine höhere Fehlerrate aufweist und man mit einer Regulierung der Temperatur den Ausschuss minimieren kann.

Das Ziel von Predictive Maintenance ist es, den Servicemitarbeiter loszuschicken, um einen Fehler zu beheben, noch bevor dieser eintritt. Das erhöht die Quality of Service und führt zu reduzierten Kosten, da die Verfügbarkeit von Diensten erhöht wird.

## 1.1.2 Marketing Segment Analytics

Hier wird die Performance von verschiedenen Marketingmaßnahmen gemessen und in Bezug zu den Mitbewerbern gesetzt.

Das Ziel ist es, den optimalen Mix aus Marketingmaßnahmen und Anreizen für eine Kunden-Fokusgruppe herauszufinden und Neukunden zu gewinnen.

Daten können in verschiedenen Clustern, z. B. Georegionen oder Mikro- und Makrotrends, zusammengefasst werden. Zusätzlich sollen auch Daten über erfolglose Marketingversuche bzw. über Prozesse gespeichert werden. Die Effizienz pro Kampagne wird maximiert und es wird dokumentiert, in welcher Situation welches Marketinginstrument den größten Erfolg geliefert hat. Über die Zeit lässt sich dann Scenario Planning, Performance- und Social-Media-Optimierung durchführen.

Werden diese Maßnahmen in Bezug zu den Kampagnen von Mitbewerbern gesetzt, so ergeben sich gegebenenfalls neue Einsichten, die in einer weiteren Kampagne wiederverwertet werden können, um neue Alleinstellungsmerkmale zu generieren.

### 1.1.3 360° View of the Customer

Hier wird das Verhalten des Kunden analysiert mit dem Ziel, Kundenbedürfnisse genau nachzuvollziehen. Die Analyse basiert auf Daten aus der Vergangenheit und wird „Customer Journey“ genannt. Ein weiteres Ziel ist, schnell Daten über einen entsprechenden Kunden zu finden, z. B. welche Produkte er konsumiert hat, welche Services er bezogen hat und wo es Beschwerden gab. Als Resultat kann dem entsprechenden Kunden punktgenau zum richtigen Zeitpunkt das richtige Angebot zu einer Dienstleistung oder einem Produkt gemacht werden.

Der nächste Schritt ist die Einbindung aktueller Daten, die laufend generiert werden, wie z. B. die aktuelle Interaktion im „Buy & Sell“-Zyklus und die aktuelle Einstellung des Kunden zum Unternehmen. Konkret soll hier ersichtlich werden, in welchem Kontext der Kunde das Unternehmen aktuell kontaktiert hat.

Der letzte Schritt ist das Vorhersagen von möglichen künftigen Interaktionen sowie Upsell- und Cross-Sell-Möglichkeiten.

Als oberste Maxime gilt dabei das Ziel, den vorhandenen Kundenstamm zu halten, d. h. die Customer Retention zu maximieren, und zugleich in neue Segmente vorzustoßen. Die Grundlage hierfür können interne (CRM, ERP …) sowie externe Daten (Social Media) sein, die von unterschiedlichen Datensilos auf einer zentralen Plattform gespeichert und mit Analysen ausgewertet werden.

Dabei kann Predictive Analytics helfen, entsprechende Muster zu erkennen und diese auszuwerten. Beispielsweise kann man feststellen, wann ein Kunde den Verkaufsprozess verlässt, und eventuell lässt sich der Grund hierfür ermitteln. Diese Erkenntnis ist die Grundlage, um entsprechende Marketingmaßnahmen zu optimieren und auszubauen.

### 1.1.4 Zusammenfassung

Der Erfolg dieser Strategien lässt sich an einem erhöhten Return on Invest und einer reduzierten Time to Market ablesen. Da die Umsetzung von Visionen immer einen erhöhten Bedarf an Ressourcen hat, ist ein Top-down-Ansatz, bei dem die Strategie von einem Mitglied des Vorstandes angestoßen und in den Abteilungen umgesetzt wird, besser geeignet als ein Bottom-up-Ansatz, bei dem die Vision in einer Abteilung entsteht und dann auf weitere Abteilungen ausgedehnt wird.

Beim Top-down-Ansatz werden etwaige Organisationsprobleme in den einzelnen Abteilungen minimiert, da die Abteilungen von Anfang an involviert sind. Somit steht von Anfang an die Gesamtlösung im Fokus. Wir gehen in Kapitel 8 genauer darauf ein.

Erstes Ziel der Gesamtlösung kann z. B. sein, die Anzahl der Datensilos zu verringern, und so einen konzernweiten Data Lake aufzubauen. Der nächste Schritt wäre dann, innerhalb des Data Lake die Daten intelligent zu verknüpfen, um einen Mehrwert zu generieren, da die Daten nun harmonisiert, d. h. in einem zentralen Punkt gesammelt, aufbereitet werden.

Parallel dazu können einzelne Fachabteilungen beginnen aus den Daten einen Mehrwert zu generieren, indem sie ihre Probleme anhand dieser Daten lösen. Das führt zu diversen Prototypen oder Initiativen, die in ein „Digitales Investment“ umgewandelt werden können. Um

den maximalen Nutzen aus den Daten zu generieren, ist auch eine entsprechende Governance-Strategie notwendig, um die Datenverarbeitungsschritte und somit die Wertschöpfungskette zu dokumentieren und die angewandten Schritte nachvollziehbar zu machen. Zusätzlich erhöht die Governance den Grad der Vertrauenswürdigkeit der Plattform, da nun ersichtlich ist, woher die Daten kommen und wohin sie gespeichert werden.

Im Kern der Entwicklung des digitalen Geschäftsmodells steht also eine Transformation vom verarbeitenden Gewerbe hin zu einem Dienstleistungsgewerbe, für die ein kontinuierliches Abonnement nötig ist oder die pro Volumen bezahlt wird. Das heißt, „dass physische Produkte nur noch Hilfsmittel sind, um ein Ziel zu erreichen oder ein Kundenproblem zu adressieren“[1] und so einen kontinuierlichen Cashflow zu generieren, anstatt einen einmaligen.

Gegenüber der traditionellen Software-Industrie ergeben sich in der Daten-Industrie folgende Unterschiede:

- Es gibt eine zentralisierte Datenplattform.
- Kunden sind Miterzeuger der Lösungen oder Produkte.
- Neue Lösungen und Produkte können laufend optimiert und skaliert werden.
- Es bestehen erhöhte Anforderungen an die Automatisierung sowie die Qualitätssicherung aufgrund der Menge an Daten, die verarbeitet werden.
- Die Organisationsform der Personen, die Big Data anwenden, ist agil und passt sich der Problemstellung kontinuierlich an. Sie erlaubt auch entsprechende Fehler zu machen, folgt aber den vorher definierten Anwendungsfällen bzw. der Vision.

Im Folgenden werden nun die Bereiche vorgestellt, die eine generelle Digitalisierungsstrategie für die Entwicklung eines Geschäftsfeldes in der Regel umfasst.

## 1.2 Umsetzung einer Digitalisierungsstrategie

Die Digitalisierungsstrategie leitet sich von der Strategie des Unternehmens ab und unterstützt diese mit Big-Data-Werkzeugen. In der Digitalisierungsstrategie wird der operative Fokus von Big-Data-Initiativen festgelegt und es wird definiert, wie Daten und die darauf aufbauende Modellierung geschäftsseitig interpretiert werden. Sie legt auch fest, wie aus Daten Informationen gewonnen werden und welche Fragen des Unternehmens beantwortet werden.

[1] [Karch 2017]

### 1.2.1 Daten

Daten liegen in Unternehmen aktuell horizontal, d. h. in verschiedenen Abteilungen, oder vertikal, d. h. gekoppelt an verschiedene Funktionen, fragmentiert und in Silos vor. Zusätzlich wächst der Anteil der kritischen Informationen, die außerhalb der üblichen Prozesse generiert werden. Das Ziel der Datenstrategie ist also einen Prozess zu erstellen, der verschiedenste Datenformate verarbeiten und in ein strukturiertes und orchestrierbares Format überführen kann. Dabei lassen sich die Daten in vier verschiedenen Eigenschaften beschreiben.

- **Volumen:** Beschreibt die Menge der Daten, die durch tägliche Geschäftsprozesse erhoben werden. Hierbei handelt es sich um eine Größenordnung – wie etwa Gigabyte, Terabyte oder Petabyte.
- **Velocity:** Beschreibt die Geschwindigkeit der Daten, die während einer Session oder Transaktion generiert werden. Sensordaten sind typischerweise Daten mit einer sehr hohen Geschwindigkeit, da sie oftmals unmittelbar verarbeitet werden müssen. Erkennt man beispielsweise mit Sensoren Probleme in der Produktion, so will man innerhalb weniger Minuten reagieren können.
- **Veracity:** Dieser Wert beschreibt die Vertrauenswürdigkeit bzw. die Genauigkeit der Daten. Als vertrauenserhöhende Maßnahmen können beispielsweise Data Lineage, also das Nachzeichnen der einzelnen Datenverarbeitungsschritte und Datenströme, sowie entsprechende Signaturmechanismen verwendet werden. Zusätzlich kann ein Wasserzeichen mitgeführt werden, das den letzten Zeitpunkt der Verwendung dokumentiert.
- **Variety:** Beschreibt die Varietät der Daten. So muss eine gesamte Plattform unterschiedliche Daten, wie Sprachdaten oder Textdaten, konsumieren können, um effektiv und effizient betrieben werden zu können. Zusätzlich muss sie auch die gängigen Konnektoren zu den einzelnen im Unternehmen verwendeten Schnittstellen haben, um so die benötigten Daten effizient zur Verfügung zu stellen.

### 1.2.2 Modellierung und Analyse

Daten selbst erzeugen noch keinen Wert für das Unternehmen. Um aus den Daten einen Wert zu generieren, werden analytische Modelle oder Algorithmen benötigt. Typische Anwendungsfälle sind Optimierungsaufgaben, Voraussagen für die nächste Rechnungsperiode oder Risikoklassifizierungen. Es muss also ein Plan erstellt werden, wie die analytischen Ressourcen, wie Personal oder Hardware, am effizientesten und effektivsten eingesetzt werden können und somit das meiste Potenzial haben, einen Wert für das Unternehmen zu generieren. Dabei müssen auch das automatisierte Training und die Anpassung an neue, dem System noch unbekannte Werte berücksichtigt werden. Eine mögliche Lösung wäre beispielsweise etwaige Probleme bereits während des System-Designs zu berücksichtigen oder während des Betriebs darauf angemessen zu reagieren.

Hierbei ist es wichtig, den gesamten Ende-zu-Ende-Prozess im Auge zu behalten, einschließlich der Anwender, der Komplexität der Lösung sowie des Managements der Modelle und der Daten selbst.

### Ende-zu-Ende-Prozess der Datenverarbeitung

Die Resultate der technischen Überlegungen, die während der Analyse-Phase getroffen worden sind, beispielsweise welche Daten wie analysiert werden, müssen für die Entscheidungsträger und Stakeholder aufbereitet werden. Der Mehrwert der Arbeit ergibt sich erst, wenn die Endbenutzer, beispielsweise Manager oder Angestellte, die Analyse-Ergebnisse direkt verwenden können.

Ob sie verwendet werden, hängt meist von einigen nichttechnischen Parametern ab, wie Nachvollziehbarkeit und Bedienbarkeit der Benutzeroberfläche. Zusätzlich werden organisatorische Ressourcen benötigt, beispielsweise jemand, der die Strategie aktiv weiter voranbringt sowie Leute mit den entsprechenden analytischen und technischen Fähigkeiten, um den Ende-zu-Ende-Prozess umzusetzen.

Wird nun ein Produkt, das physisch vorliegt, in eine Dienstleistung transformiert, dann muss das neue Produkt ein Bedürfnis oder ein Problem besser lösen, als dies bisher der Fall war, und so die Customer Experience weiter steigern, um sich durchzusetzen.

Daher ist es unerlässlich, die strategischen Ziele mit den längerfristigen operativen Zielen des Business in Einklang zu bringen. Eine solche Strategie kann unterschiedlich angewandt werden und benötigt mitunter einige Zeit, insbesondere wenn Investitionen in Hardware-Ressourcen notwendig werden. Alternativ zu den eigenen Investitionen kann auch ein Cloud-Ansatz gewählt werden, in dem die ganze Infrastruktur in der Cloud, etwa AWS oder Azure, angeboten wird. Das hat den Vorteil, dass schnell begonnen werden kann, jedoch steigen die Kosten mit der Zeit. Zusätzlich entsteht ein Vendor-Lock-in, da nach der Wahl eines Cloud-Providers ein Wechsel nur mit erheblichem Mehraufwand möglich ist.

Im Gegensatz dazu gibt es den On-Premise-Ansatz, der die physische Hardware in der Firma aufbaut und somit erhebliche Investitionen zu Beginn des Projekts erfordert. Im On-Premise-Ansatz entstehen wesentlich höhere Kosten, da eine dynamische Ressourcenzuteilung nicht möglich ist. Man muss die gesamte Kapazität vorhalten und kann auf geänderte Anforderungen nur mit erheblichem Einsatze neuer Ressourcen reagieren.

Die Umsetzung der Digitalisierungsstrategie umfasst folgende Überlegungen:

- Die erhobenen Daten des angebotenen Service bilden die Grundlage für weitere Aktionen. Diese Daten können beispielsweise verwendet werden, um den Service selbst zu optimieren und etwaige Schwachstellen zu beheben.
- Über einen möglichst vernetzten digitalen Kanal kann mit dem Endbenutzer, dem Service oder anderen Partnern kommuniziert werden. Zusätzlich lässt ein digitaler Kanal die Anwendung von analytischen Modellen zu, da diese meist erheblich mehr Ressourcen benötigen, als bei der Datenerhebung verfügbar wären.
- Durch eine zentrale Plattform werden alle Prozesse und Services automatisiert verwaltet und stehen so schnell und übersichtlich dem entsprechenden Endbenutzer oder der Business Unit zur Verfügung.

### Datenprodukte

Daten und Informationen können beispielsweise über Sensoren am physischen Produkt (IoT) oder in Social-Media-Anwendungen sehr nahe am Entstehungsort gesammelt und ausgewertet werden. Dabei wird letztendlich Wissen, wie ein Service oder Produkt benutzt wird, generiert. Das Produkt wird im ersten Schritt in ein intelligentes Produkt verwandelt,

das heißt das Produkt erhält die Fähigkeit seinen physikalischen Zustand zu beschreiben, indem es mit Sensoren ausgestattet wird.

Ein Wesensmerkmal von intelligenten Produkten aus Kundensicht ist, dass sie einen Zusatznutzen zum herkömmlichen Produktnutzen liefern, der durch Datensammlung zustande kommt. Ein Beispiel hierfür wäre ein Gabelstapler, in dem Sensoren verbaut sind. Neben der eigentlichen Funktion als Gabelstapler, kann nun der Hersteller z. B. ein Logistikprogramm anbieten, sofern er entsprechende Sensoren verarbeitet hat, um den Logistikfluss zu optimieren. Der Gabelstapler könnte auch bestimmte Orte autonom ansteuern und über Sensordaten auf seine Umwelt reagieren.

Der nächste Schritt ist nun das intelligente Produkt zu vernetzen und so einen Mehrwert zu generieren, beispielsweise indem verschiedene Individuen interagieren können. Mit den so erfassten Daten und Verknüpfungen können die Produkte, anhand der Daten die während der Benutzung des Produkts gesammelt worden sind, optimiert werden. In der Regel wird dieser Schritt mit einer Cloud gelöst, da diese die Möglichkeit bietet Daten zentral und sicher zu verwalten. Wichtig ist hierbei, dass entsprechende rechtliche Rahmenbedingungen berücksichtigt werden.

Meist erfolgt in diesem oder im nächsten Schritt die Umstellung vom analogen zum digitalen Produkt, sofern entsprechende Vorarbeit geleistet worden ist.

Bisher wurde nur eine spezifische Produktgruppe oder ein Produkt bearbeitet. Der finale Schritt ist die laterale Diversifikation, d. h. das Angebot mit einem definierten Rahmen wird nun so erweitert, dass verschiedene neue Produktgruppen auf einer Plattform eingebaut werden können. Der Vorteil einer Plattform ist, dass das einzelne Produkt, das digitalisiert worden ist, nur mehr eins von vielen ist. Falls sich die Marktverhältnisse ändern und ein Produkt nicht mehr rentabel ist, kann es einfach gegen ein neues ausgetauscht werden.

### Anwendungsfälle

Ein weiterer Vorteil einer Datenplattform ist, dass mehrere unterschiedliche Anwendungsfälle eines Produkts neu entwickelt und skaliert werden können bzw. sich die Entwicklungszeit von Anwendungsfällen erheblich verringert.

Um das Generieren von Anwendungsfällen und in weiterer Folge den Umsatz effizient voranzutreiben, können zwei Ansätze kombiniert werden.

Beim **konzeptuellen Ansatz** steht die Ideenfindung im Vordergrund. Es sollen möglichst viele neue Anwendungsfälle gefunden werden, die anschließend verwertet werden können. Dabei können verschiedene Werkzeuge wie Open Innovation oder Design Thinking zum Einsatz kommen, um den Prozess möglichst offen zu halten und die Kreativität zu fördern. Design Thinking beschreibt den Ansatz der Lösungsfindung aus Endanwendersicht, während Open Innovation den Lösungsraum erheblich vergrößert.

Wichtig in dieser Phase ist es, in einen Zyklus der Validierung und Verifizierung einzutreten, um früh mögliche Sackgassen und nicht verfolgbare Ziele ausschließen zu können.

Beim **datengetriebenen Ansatz** werden vorhandene Datenquellen schrittweise erweitert, um so neuen Wert zu generieren. Dabei können Workshops wie „Bring your own Data“ oder Hackathons veranstaltet werden, um das Datenverständnis auszubauen bzw. zu erweitern. Dieser Ansatz empfiehlt sich insbesondere für die initiale Analyse der Datenquellen, ist jedoch mit steigender Kenntnis der Daten weniger geeignet.

Zusätzlich und unabhängig von dem gewählten Ansatz muss der Anwendungsfall auch eine Prüfung der Wirtschaftlichkeit durchlaufen, um festzustellen, ob er einen Business Value beisteuert oder nicht. Meistens werden zu Beginn eines Projekts bestehende Prozesse digitalisiert, d. h. der Value ergibt sich dann aus den eingesparten Kosten.

Ein praktikabler Weg, den Prozess zu starten, ist ein Workshop mit Daten- und Domainexperten aus unterschiedlichen Abteilungen und Business Units, um herauszufinden, wo es noch Optimierungspotenzial gibt. Das Ergebnis des Workshops sollte eine priorisierte Liste von Anwendungsfällen sein, die in einer explorativen Phase weiter ausgewertet werden kann. Diese Liste wird anhand von Daten quantifiziert, um zu ermitteln, wie umsetzbar ein Anwendungsfall ist und ob es sich lohnt diesen weiter zu verfolgen. Hierbei sind agile Methoden wie Scrum von Vorteil, da diese eine iterative Annäherung an das Ergebnis erlauben oder es ermöglichen, das Ziel innerhalb der Entwicklung zu ändern, um es z. B. auf neue Business Goals auszurichten.

Die unterschiedlichen Daten, die aktuell erzeugt werden, können mit den herkömmlichen Werkzeugen nicht effizient verarbeitet werden, da die zugrunde liegende Technologie für die Informationsverwaltung seit ca. 1980 Datenbanken sind und diese auf strukturierte Daten ausgelegt sind. Datenbanken erlauben es, die Daten, die eine Form haben, als Tabelle zu organisieren, zu gruppieren und abzufragen sowie verschiedene Extraktions-, Transformations- und Ladeschritte (ETL) auszuführen, um so eine harmonisierte Sicht auf die darunterliegenden Daten zu bekommen.

Die Analysten konnten sich in der Vergangenheit also auf den Schritt der Interpretation und Selektion der Daten fokussieren. Das führte dazu, dass die Fehler, die bei der Extraktion gemacht worden sind, entsprechend multipliziert wurden, indem einige Informationen überbewertet, andere unterbewertet und andere verfälscht wurden. Eine derartige Selektion wird im Allgemeinen kognitive Verzerrung oder Bias genannt.

Seit den 90er-Jahren hat sich die verfügbare Menge an generierten Daten alle zwei Jahre verdoppelt. Dieses Wachstum wird sich mit dem Trend zur Vernetzung der Dinge noch weiter beschleunigen. Ein offenes Problem, das sich aus der steigenden Datenmenge ergibt, ist die effiziente Verwaltung dieser Daten bzw. die Informationsextraktion aus den Rohdaten.

Durch den Einsatz von Big-Data-Technologien wie Hadoop erhoffte man sich, diese Probleme zu lösen, indem mehrere Computer zu einem Verbund zusammengeschlossen werden. Somit wird ein Problem auf mehrere Teilprobleme heruntergebrochen und entsprechend berechnet, um dann das Ergebnis am Ende wieder zusammenzuführen. Der Nachteil einer solchen Lösung ist, dass sie technisch komplexer ist als ein einzelnes Programm, denn Teilprogramme werden auf unterschiedlichen Rechnern unterschiedlich ausgeführt.

In den vergangenen Entwicklungsjahren wurde der Datenmenge eine steigende Priorität zugewiesen, während die wissenschaftliche Aufbereitung, wie das Formulieren der Arbeitshypothese, immer weiter in den Hintergrund rückte. Die Verheißung von Big Data ist, dass jedes Problem mit einer hinreichend großen Menge an Computer-Power und Daten gelöst werden kann. Das lässt sich auch anhand des Produktivitätsparadoxons ablesen: Mehr Daten und bessere Algorithmen machen uns (derzeit) nicht produktiver, aktuell trifft eher das Gegenteil zu, da es immer schwieriger wird, das Signal vom Rauschen zu unterscheiden. Das Signal sind jene Informationen, die für eine Fragestellung relevant sind und somit zu deren Beantwortung beitragen, während das Rauschen die irrelevanten Informationen sind.

Im wissenschaftlichen Bereich wird versucht diese Signale messbar zu machen, indem man die Genauigkeit der Signalerkennung misst und betrachtet, wie oft das Signal gefunden worden ist. Der Quotient beider Messungen drückt die Gesamtgenauigkeit der verwendeten Lösung aus und dient somit der Bewertung von einzelnen Algorithmen. Er wird üblicherweise in Prozent angeben. Ein hoher F1 Score bedeutet dabei, dass es eine präzise Lösung ist, während Werte um 50 % ein Zufallsergebnis repräsentieren. Wenn ein Algorithmus also eine Genauigkeit von z. B. 90 % aufweist, dann heißt das, dass 90 % aller Informationen richtig verarbeitet werden. Das mag nach viel klingen, jedoch sind Daten mit einem großen Volumen in Big Data der Regelfall. Beispielsweise wurden im Jahr 2018 pro Sekunde auf Facebook 510 000 Kommentare gepostet. Legt man die 10 % aus dem Beispiel zugrunde, bedeutet das 51 000 falsch verarbeitete Posts. „Verarbeiten" heißt hier, diverse Algorithmen klassifizieren z. B. einen Kommentar falsch. Das heißt, dass die falsch verarbeiteten Kommentare nochmals oder im schlimmsten Fall manuell nachverarbeitet werden müssen, und so ein entsprechender Mehraufwand entsteht.

Um dieses Problem zu vermeiden, wird derzeit künstliche Intelligenz oder Deep Learning, eine Gruppe von Machine-Learning-Algorithmen, die auf neuronalen Netzen basiert, als abstrakte Lösung für die verschiedensten Probleme angewendet. Der Vorteil von Deep Learning gegenüber dem klassischen Machine Learning ist, dass Deep Learning in der Regel besser mit der Menge an Daten skaliert und so präzisere Resultate liefert und auf verschiedene Probleme angewandt werden kann.

Der Nachteil von künstlicher Intelligenz ist, dass es mitunter schwierig sein kann, eine Vorhersage zu interpretieren, da der Lösungsweg auf Anhieb nicht nachvollziehbar ist. Des Weiteren kann eine statistisch generierte Vorhersage zutreffen oder auch nicht, da sie in der Regel eine Genauigkeit von weniger als 100 % aufweist. Zusätzlich können statistische Vorhersagen nicht oder nur eingeschränkt verwendet werden, um neue, noch unzureichend analysierte Daten vorherzusagen. Diese Aussage mag trivial erscheinen, ist jedoch essenziell, da eine statistische Analyse primär von den Eingangsdaten, und somit vom Modellierungsgeschick des Data Scientists, abhängt. Es ist daher unumgänglich das Ergebnis richtig zu interpretieren und nicht als Wahrheit zu übernehmen.

Ein gutes Beispiel hierfür sind numerische Wettervorhersagen wie der Wetterbericht. Hier sind die physikalischen Grundgesetze in Form von Differentialgleichungen bekannt, es kommt aber immer wieder zu falschen Vorhersagen aufgrund nicht vorhandener oder falscher Daten. Ein Ergebnis der gelösten Differentialgleichungen kann sein: „Morgen beträgt die Regenwahrscheinlichkeit 10 %". Statistisch betrachtet heißt das, dass es in 10 % aller Szenarien, die berechnet worden sind, geregnet hat. So können 10 % sehr viel oder sehr wenig sein, wichtig ist es, eine entsprechende Bezugsgröße zu haben und diese mit der erhaltenen Größe in Bezug zu setzen. In diesem Fall heißt das, dass es durchaus möglich ist, wenn auch nicht wahrscheinlich, dass es morgen regnet.

## 1.3 Teams

Um Digitalisierung mit Big-Data-Technologien umzusetzen, wird Expertenwissen benötigt. Es gibt zwei Expertengruppen in der Datenwelt, die sich im Lauf der Zeit entwickelt haben.

Die erste Gruppe, Personen mit statistischem Hintergrund, hat üblicherweise einen akademischen Hintergrund und erstellt Modelle, um die Fragen der Fachabteilungen zu beantworten.

Die andere Gruppe sind Personen mit einem Ingenieurshintergrund. Sie sind dafür zuständig, die Daten vollautomatisiert auf die Plattform zu laden und die Daten der entwickelten Modelle kontinuierlich in der Produktivumgebung laufen zu lassen.

Innerhalb dieser beiden Gruppen gibt es weitere Untergruppen. Diese werden in Bild 1.1 gezeigt. Eine Person, die einen Ingenieurshintergrund hat, kann beispielsweise im Bereich DevOps oder als Data Engineer arbeiten.

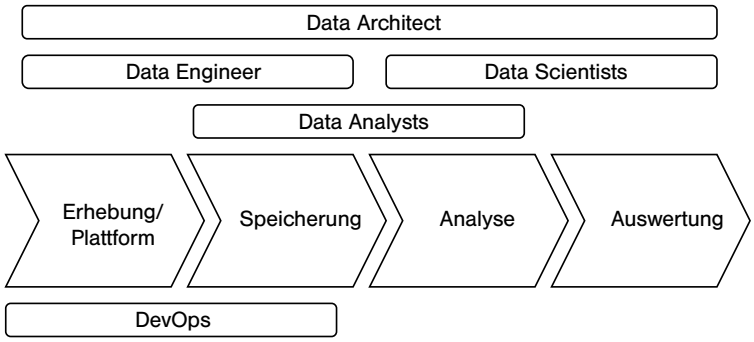


**Bild 1.1** Rollenverteilung in Datenprogrammen

### 1.3.1 Data Scientist

Ein Data Scientist ist der Experte mit einem statistischen Hintergrund. Er kann Modelle bauen und bereitstellen. Je nach Anwendungsfall kann ein Modell entweder eine Klassifikation von Daten, eine Voraussage, ein Optimierungsproblem oder eine abstrakte Datentransformation sein. Ein Data Scientist versucht, die Fragen aus einer Fachabteilung wie etwa Marketing oder der Produktion zu beantworten.

Dabei wird zwischen zwei großen Gruppen von Modellen unterschieden. Die erste Kategorie, Supervised bzw. überwachte Machine-Learning-Algorithmen benötigen aufbereitete Daten, die mit den entsprechenden Ergebnissen bereits bekannt sind. Nachdem der Algorithmus von den Daten gelernt hat, z. B. wie man sie richtig klassifiziert, können auch unbekannte Daten klassifiziert werden. Im Gegensatz dazu gibt es die unsupervised bzw. unüberwachten Machine-Learning-Algorithmen, die durch eine entsprechende Zielfunktion festlegen, wie die Daten z. B. klassifiziert werden. Sie benötigen dabei keine vorbereiteten Daten und können direkt verwendet werden.

Die Hauptaufgabe von Data Scientists ist die Modellierung, Erstellung und Optimierung von Daten mit den oben beschriebenen Algorithmen. Diese Tätigkeit inkludiert das Aufbereiten von Daten sowie das Harmonisieren von verschiedenen Datenquellen bis hin zur Interpretation des gelieferten Ergebnisses sowie dessen Präsentation.

Die Aufbereitung der Datenquellen erfolgt mittels explorativer statistischer Analyse, d. h. es werden Daten in einer Testumgebung geladen und „erforscht".

Basierend auf diesen Erkenntnissen wird ein Datenset erzeugt, um darauf die entsprechenden Algorithmen zu testen und zu evaluieren. Nachdem das Modell fertig entwickelt worden ist, kann es produktiv verwendet werden.

Weitere Standardaufgaben von Data Scientists sind unter anderem:

- kontinuierliche Evaluierung und Anpassung von statistischen Modellen, um den Daten zugrunde liegende Änderungen auszubessern
- Kommunikation mit diversen Abteilungen, um die entsprechenden Business-Fragen zu beantworten
- Visualisieren der Ergebnisse und deren Präsentation

### 1.3.2 Business Analyst

Business Analysten sammeln Daten. Sie manipulieren und analysieren diese mit dem Ziel einen Prototyp für die weiteren Arbeitsschritte zu finden bzw. erstellen.

Business Analysten bereiten Berichte vor und bilden die Schnittstelle zwischen einzelnen Abteilungen und der datenverarbeitenden Abteilung. Sie unterstützen das Business, indem sie die von den Abteilungen erhaltenen Informationen aufbereiten und diese in die jeweilige „Sprache" der anderen übersetzen.

Analysten sind oft Business-Anwender mit einem klaren operativen Ziel. Anders als Data Scientists generieren sie kein neues Wissen und arbeiten im Regelfall nicht strategisch. Beispielsweise ermittelt ein Business Analyst in der Risk-Abteilung eines Finanzdienstleisters anhand der vorhandenen Daten das Kreditrisiko eines Kunden. Die Vision eines Data Scientists ist mit allen vorhandenen Daten ein Modell zu generieren, das dem Business Analysten hilft, die Analysen weiter zu verbessern.

Die eigentlichen Aufgaben des Business Analysten hängen daher von dem Einsatzgebiet ab, umfassen aber im Allgemeinen folgende Tätigkeiten:

- Entwicklung einer Harmonisierungsstrategie der Daten, sodass ein Datensatz entsteht, mit dem gearbeitet werden kann
- mit deskriptiver Statistik ein Big-Picture der Datenanalyse erstellen
- Analyse der Daten, um entsprechende Trends zu finden
- visuelle Aufbereitung, um diese Trends zu interpretieren
- Präsentation der erzielten Ergebnisse sowie etwaige Kommunikation mit verschiedenen Abteilungen
- Entscheidungsempfehlungen anhand der gesammelten Daten geben

### 1.3.3 Data Architect

Ein Data Architect spielt eine zentrale Rolle bei der Umsetzung einer Datenstrategie im Unternehmen. Wesentliche Aufgaben des Data Architects sind, die Daten-Plattformen der einzelnen Abteilungen im Unternehmen zugänglich zu machen sowie das Aufstellen des Gesamtkonzeptes. Hierbei arbeitet der Data Architect eng mit den Data Engineers, den Data Scientists und den Business Analysten zusammen. Die Aufgaben für den Data Architect sind hierbei vielfältig: der Data Architect stellt sicher, dass Firmen und Organisationen einen einheitlichen Datenstandard haben, dass es eine definierte Architektur einer Datenplattform gibt, die die Umsetzung von Businesszielen ermöglicht.

Dabei muss er durchschnittlich zwei operative Ziele sicherstellen. Das sind die Skalierbarkeit der Architektur und das Erzeugen von kritischen Erkenntnissen in Echtzeit. Ein Data Architect hat ebenso die Aufgabe, eine kostengünstige Architektur zu bauen. Daher benötigt der Architect eine Gesamtsicht auf das System sowie eine permanente Kommunikation mit den einzelnen Abteilungen, um entsprechende Wünsche umsetzen zu können.

Typische Aufgaben eines Data Architects sind:

- Konzepte zu Metadaten- und Schemapflege definieren
- Datenmanagement inklusive Sicherstellung der Integrität von Daten
- Optimierung von Datenbanken und Gewährleistung des Quelldatenzugriffs
- Umsetzung von Sicherungskonzepten und Garantie von Zugriffkonzepten
- Festlegen der Komponenten der Plattform, um Businessziele zu erfüllen
- Festlegen der Architektur der Plattform (Echtzeitplattform vs. stapelbasierende Plattform vs. Plattform für analytische Anwendungen)

Im Unterschied zu Data Engineers kümmern sich Data Architects um die Beladung der Plattform, während Data Engineers sich auf der Datenplattform um die Bereitstellung von Daten und Informationen kümmern.

### 1.3.4 Data Engineer

Data Engineers bauen und optimieren Datenplattformen, sodass Data Scientists und Analysten Zugriff auf die entsprechenden Daten haben. Sie sorgen dafür, dass die Daten im richtigen Format mit entsprechender Geschwindigkeit in die Datenplattform geladen werden und die Daten gemäß der vom Architekten festgelegten Policy gespeichert werden.

Data Engineers setzen diese Tätigkeit mittels Datenpipelines um, laden durch sie Daten von Drittsystemen, transformieren die Daten und speichern sie dann auf der Plattform ab. Kernaufgabe hierbei ist, dass die Datenpipeline mit steigendem Datenaufkommen skalieren und entsprechend robust sein muss, d. h. die Pipeline muss eine entsprechende Fehlertoleranz aufweisen. Sie bildet somit die Basis, die von Data Scientists und Analysten verwendet werden kann, um Wissen zu generieren.

Im Unterschied zu anderen Team-Mitgliedern ist es unverzichtbar, dass Data Engineers solide Programmierfähigkeiten aufweisen, da diese Rolle entsprechende Programmiertätigkeiten mit sich bringt. Der Data Engineer hat eine fundamentale Rolle in jedem Data-Science-Team und entlastet die anderen Mitglieder.

Die Kerntätigkeiten umfassen:

- Bauen von diversen Schnittstellen, um lesend und schreibend Daten bereitzustellen
- Integration von internen oder externen Daten in existierenden Pipelines
- Anwenden von Datentransformationen, um analytische Datasets zu erzeugen
- Monitoring und Optimierung, um die kontinuierliche Qualität des Systems sicherzustellen (und ggf. zu verbessern)
- Entwicklung eines Ladeframeworks, um Daten effizient zu laden

### 1.3.5 DevOps

DevOps sind eine Mischung aus Entwickler (Developer) und Personen, die die operativen Systeme betreuen. Ihre Aufgabe ist der Betrieb der Datenplattform, auf Basis welcher die Data Engineers und Data Scientists arbeiten. Hierbei handelt es sich um eine reine Infrastrukturaufgabe bzw. unterstützende Funktion.

DevOps setzen den Architekturentwurf der Data Architects um und gehen auf die Änderungswünsche der Data Engineers ein.

Zu ihren Tätigkeiten zählen:

- Skalierung von Datenplattformen
- Identifikation von Performance-Problemen der Software
- Automatisieren von erneuten Deployments
- Monitoring und Logging der Applikationen
- Identifikation von Ressourcenengpässen und -problemen
- Behebung von Problemen, die durch den Betrieb auftreten

### 1.3.6 Weitere Rollen

Es gibt weitere Rollen, auf die hier nicht im Detail eingegangen werden kann. Data Stewards sind beispielsweise Personen, die auf die Einhaltung von Governance-Regeln achten. Man kann sie auch als Datenbuchhalter sehen, die daran interessiert sind, dass Qualitätsmetriken eingehalten werden und dass gespeicherte Daten Besitzer haben, die für die Daten verantwortlich sind. Auch gibt es Security-Experten, die die Sicherheit einer Datenplattform gewährleisten sollen. Diese reicht von der reinen Informationssicherheit (Sind meine Daten so abgelegt, dass nur befugte Personen darauf Zugriff haben?) bis zur Plattformsicherheit (Übersteht die Datenplattform einen Eindringungsversuch via Penetration Testing?).

### 1.3.7 Team Building

Aus den oben beschriebenen Rollen wird typischerweise ein Daten-Team geformt. Je nach „digitalem Reifegrad“ wird das Team erweitert und der Fokus des Teams ändert sich.

Beim Etablieren der digitalen Plattform empfiehlt es sich, mehrere Data Engineers zu haben, da die Daten erst aus ihren Abteilungssilos und operativen Systemen extrahiert und auf die Plattform geladen werden müssen. Im weiteren Verlauf eines Projekts sinkt jedoch der Aufwand der Datenerhebung und die Hauptkonzentration verlagert sich vom Aufwand der notwendig ist, um den Lake initial zu beladen, auf den analytischen und Auswertungsaufwand. Im späteren Verlauf empfiehlt es sich Mitarbeiter mit DevOps- oder Data-Analyst-Kenntnissen zu haben, um die Arbeit der Data Scientists und Data Engineers zu unterstützen.

Nichtsdestotrotz empfiehlt es sich, einmal aufgebaute Kapazitäten nicht wieder abzubauen, sondern maximal auszubauen, da immer wieder neue und unterschiedliche Daten gespeichert werden müssen und die Plattform weiterentwickelt werden muss.

# 4 Data Pipelines

*Bernhard Ortner*

*„Thank you, Mario! But our princess is in another castle“ – Super Mario Bros*

Eine Data Pipeline ist ein mehrstufiger Prozess, der Daten konsumiert, verarbeitet und diese in einer meist aggregierten Form, die zur Entscheidungsunterstützung verwendet wird, abspeichert. Eine Pipeline stellt somit einen standardisierten Workflow dar, der auf den betrachteten Anwendungsfall zugeschnitten ist.

Je nach Anwendungsfall des zu liefernden unternehmerischen Werts werden unterschiedliche Anforderungen an die zugrunde liegenden Algorithmen gestellt. Diese haben dementsprechend Auswirkungen auf die Data Pipeline.

In diesem Kapitel werden zuerst die Anforderungen, „Best Practices“ sowie die häufigsten Schwierigkeiten bei der Entwicklung und Inbetriebnahme von Data Pipelines beschrieben. Danach werden unterschiedliche Typen von Data Pipelines vorgestellt, je nachdem wie die Modelle erstellt worden sind. Am Ende wird eine autonome Pipeline beschrieben, die Modelle automatisiert aktualisieren kann.

## 4.1 Data Pipelines im Big-Data-Zeitalter

Das Konzept der Data Pipeline hat sich über Jahre entwickelt. Es kann in vier Kategorien unterteilt werden. Die einzelnen Kategorien hängen davon ab, wie die Daten angeliefert werden und in welchem Zustand (Schema oder Schema-frei) die Daten sind. Das Bereinigen der Daten bedeutet je nach Art der Daten unterschiedlich hohen Aufwand, da eine Data Pipeline ähnlich aufbereitete Daten benötigt.

### Datei-Ära

Die Daten werden als Datei *(flat file)* angeliefert. Beispielsweise sind das Log-Dateien von Servern. Diese Dateien können nicht direkt verarbeitet werden, da sie keine Abfragen wie „Gib mir meine Top-10-Einträge“ zulassen. Zusätzlich skaliert der Ansatz schlecht, da die Dateien von den einzelnen Maschinen abgeholt werden müssen und lokale Dateisysteme begrenzte Kapazitäten haben.

Die dateibasierte Speicherung ist im Gegensatz zu anderen Ansätzen leicht umzusetzen, da sie von keinen weiteren Komponenten wie Orchestrierungssoftware abhängig ist und die

Speicherung weniger Verarbeitungsschritte benötigt als eine Data Pipeline. Die gespeicherten Daten in den Dateien können bei Bedarf adaptiert werden, da sie keinem fixen Schema folgen.

### Datenbank-Ära

Die Daten werden in einer Datenbank, wie z. B. MySQL oder RedShift, gespeichert. Sie können daher leicht extrahiert, transformiert und geladen werden (Extract, Transform, Load, ETL). Die Datenbank selbst wird mit den entsprechenden ETL-Werkzeugen des jeweiligen Vendors beladen. Das setzt voraus, dass die Daten ein vorher bekanntes Schema aufweisen oder in ein entsprechendes gebracht werden.

Der Vorteil dieses Ansatzes ist, dass er ausgereifte Werkzeuge mitbringt, da es ihn im Vergleich zum Big-Data-Ansatz schon lange gibt. Zusätzlich ist die am häufigsten verwendete Schnittstelle (SQL) zur Datenbank weitverbreitet und standardisiert.

Auch ist es möglich, über Datenbanken zu skalieren und beispielsweise Daten über mehrere Knoten zu verteilen. Dabei kann entweder Verfügbarkeit einer Datenbank oder Konsistenz in einem verteilten System erreicht werden, aber niemals beides zusammen.

### Data-Lake-Ära

Wenn es zu aufwendig ist, die Daten in datenbanktaugliche Strukturen zu konvertieren und lokale Filesysteme zu klein sind, bietet sich der Data-Lake-Ansatz an. Dieser baut auf verteilten Persistenzlayern auf. Neben verteilten Dateisystemen können dafür auch Object-Stores verwendet werden. Wir sind auf Persistenzlayer schon in Kapitel 2 eingegangen.

Wesentlich für das Thema Pipelining ist, dass in einem Data Lake Informationen in Iterationen in spezifische Schemata überführt und abgelegt werden können. Dadurch wird eine Skalierung und Flexibilität hinsichtlich der Daten in der Data Pipeline erreicht.

Nachteile des Data-Lake-Ansatzes sind die architektonische Komplexität sowie der erhöhte Wartungsaufwand der Infrastruktur und der Pipeline. Eine mögliche Lösung hierfür sind Managed Services, d. h. Dienste, die beispielsweise von Cloud-Providern gehostet werden.

Zusätzlich haben verschiedene Data-Lake-Werkzeuge unterschiedliche Schnittstellen und oft muss noch eine gemeinsame Schnittstelle für die operationale Software der Firmenlandschaft eingebaut werden.

### Serverless-Ära

Beim Serverless-Ansatz werden Funktionen von Programmen, wie z. B. Teile der Data Pipeline, in einzelne Container gekapselt. Das stellt eine Abstraktion zur aktuellen Betriebssystem-Schicht des Servers dar. Die einzelnen Container werden dabei ausgeführt und von einem Orchestrator verwaltet. Das hat den Vorteil, dass Komponenten ad hoc hinzugefügt oder entfernt werden können. Somit können die einzelnen benötigten Ressourcen dynamisch hoch- oder herunterskaliert werden, je nachdem, wo sie gebraucht werden.

Ein Nachteil ist, dass Serverless-Ansätze im Vergleich zu Data Lake derzeit noch teuer sind.

Im Folgenden beziehen sich die beschriebenen Data-Pipeline-Konzepte auf den Data-Lake-Ansatz, da dieser derzeit State of the Art ist. Er kann jedoch mit wenig Aufwand in den Serverless-Ansatz überführt werden.

Ein weiterer Vorteil von Serverless-Architekturen ist ihre Flexibilität hinsichtlich Deployment und Orchestrierung. Sie haben einen Container, der Funktionalität enthält und den man überall laufen lassen kann. Man benötigt niemanden, der erst einmal die grundlegende Systemsoftware installieren muss. Somit kann Software nahezu beliebig auf einer Clusterinfrastruktur hochgefahren werden, ohne die einstige Notwendigkeit, erst mal ein Deployment auszuführen.

## 4.2 Anforderungen an eine Data Pipeline

Um eine Data Pipeline erfolgreich umzusetzen, müssen diverse Annahmen darüber getroffen werden, wie die an den Data Lake angelieferten Daten verarbeitet werden sollen.

- **Typ der verarbeitenden Daten**
  Soll die Verarbeitung der Daten in Echtzeit passieren oder reicht es aus, wenn sie stapelweise verarbeitet werden? Sind die Daten voraggregiert oder liegen sie unverarbeitet vor?
- **Zugrunde liegende Modellierungssprache**
  In welcher Programmiersprache wurde das Ursprungsmodell entwickelt? Ist die ursprüngliche Sprache dieselbe, die dann in der Produktivumgebung verwendet wird? Wie funktioniert das Skalieren in dieser Sprache?
- **Repräsentationsform der Ergebnisse**
  Reicht es, die Daten aggregiert zu präsentieren, oder muss auf die Rohdaten zugegriffen werden? Welche Daten werden am Ende der Verarbeitungskette für Entscheidungen benötigt?
- **Updatefrequenz und Anzahl der Modelle**
  Wie oft muss ein Modell angepasst und neu trainiert werden? Gibt es viele einfache Modelle oder einige wenige komplexe? Wie werden Data-Science-Prozesse in die bereits vorhandenen Prozesse eingebaut?

Sind diese grundsätzlichen Fragen geklärt, können die Anforderungen an eine Pipeline festgelegt werden. Es ist später zwar möglich, etwaige Fehlentscheidungen zu korrigieren, jedoch erfordert das meistens einen hohen Ressourcenaufwand, um bereits vorhandene Prozesse dahingehend zu adaptieren.

Konzeptionell sind alle Pipelines ähnlich aufgebaut:

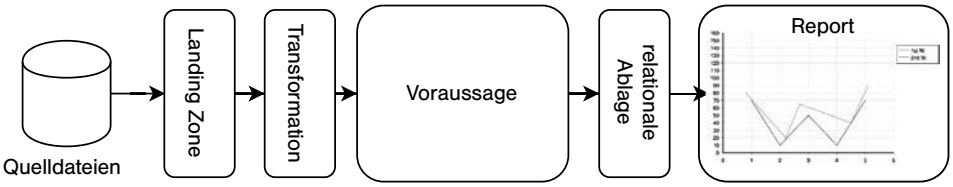


**Bild 4.1** Konzept einer Data Pipeline

Es werden Daten geladen und entsprechend des vorher festgelegten Anwendungsfalls transformiert und vorbereitet. Danach werden diverse Modelle auf den Daten angewandt und die Ergebnisse in einer Datenbank (äquivalent zu einem DataMart in der relationalen SQL-Welt) abgespeichert. Die Datenbank enthält dabei nur die aggregierten Daten, während die Rohdaten im Data Lake bleiben. Die aggregierten Daten sind mit den Rohdaten über einen Governance-Prozess verknüpft, sodass sie bei Bedarf nachgeladen werden können.

Ein Kernmerkmal von Data Pipelines ist, dass sie die folgenden drei Eigenschaften besitzen.

### Daten können reproduzierbar verarbeitet werden

Um die Datenverarbeitung nachvollziehbar zu machen, muss ein entsprechender Audittrail dokumentiert werden. Das heißt, dass jeder Verarbeitungsschritt dokumentiert wird, und klar ist, wie die Daten verarbeitet werden. Hierbei hilft es, sich den technischen Teil als „Blackbox", also als nicht sichtbare Box vorzustellen, auf deren Inhalt und Funktionsweise mittels Audittrails geschlossen werden muss.

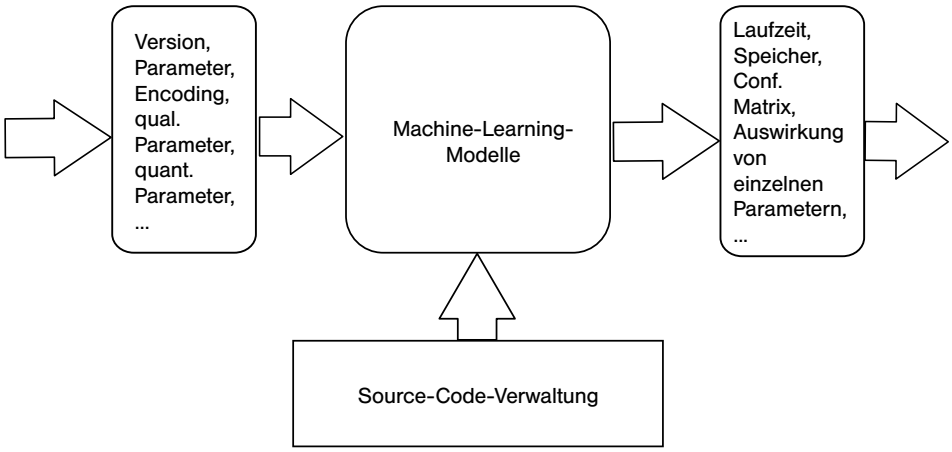


**Bild 4.2** Blackbox eines Modells

Eine weitere Anforderung an Pipelines ist, dass entsprechende Konfigurationen, Eingangsparameter von Modellen usw. sowie deren Code getrennt verwaltet werden, um die Fehlerfindung zu erleichtern. Speziell für Big Data gilt, dass Modelle, z. B. aus dem Bereich von Deep Learning, ohne die richtigen Anfangsparameter nicht zu denselben Ergebnissen kommen. Die Ergebnisse von Deep Learning sind daher eingeschränkt reproduzierbar.

Dieses Problem kann mit einem SourceCode-Verwaltungstool wie Git gelöst werden. Die Modellparameter sowie die Modelle selbst können z. B. in einer Datenbank oder in einer Binärdatei verwaltet werden. Wichtig ist, dass das gewählte Werkzeug einen Versionierungsmechanismus enthält, es also möglich ist verschiedene Versionen einer Datei abzulegen.

### Daten werden konsistent verarbeitet

Daten konsistent zu verarbeiten ist aktuell schwer, da die gängigen Auditmechanismen zwar auf produzierten Code oder Modelle abzielen, meistens jedoch vergessen wird, die Daten der Modelle selbst zu auditieren. Ein daraus resultierendes Problem ist, dass sich unerwartet Verteilungen ändern, neue oder ungültige Grenzwerte ergeben oder sich das Encoding der Daten verändert. Es ist daher umso wichtiger, dass regelmäßig Datenaudits stattfinden, um auf diese Änderung entsprechend zu reagieren und dies dann in weiterer Folge in den Modellen einzuarbeiten.

### Die Pipeline kann in der Produktion effizient betrieben werden

Das Ziel jeder Big-Data-Pipeline ist es, diese früher oder später in die Produktivumgebung einzusetzen, um so einen Mehrwert für das Unternehmen zu generieren. Speziell in Big-Data-Umgebungen reicht es nicht, nur gute Modelle oder Code zu produzieren. Eines der Hauptprobleme ist, dass die entwickelte Lösung entsprechend den vorher festgelegten Bedingungen skalieren muss. Das heißt, dass das Modell mit einer größeren Datenmenge zurechtkommen muss, jedoch nur einen eingeschränkten Ressourcenaufwand (Laufzeit, Speicherplatzverbrauch) haben darf.

Das Modell kann in der Entwicklungsumgebung trainiert und verbessert werden, bis es einen gewissen Reifegrad besitzt, während es dann in der Produktivumgebung nur für den ursprünglichen Zweck, z. B. Vorhersagen, verwendet wird.

## 4.3 Sechs Stufen der Data Pipeline

Bevor eine Data Pipeline erstellt wird, stellt sich die Frage, für welchen Zweck sie verwendet werden oder welches Problem sie lösen soll. Dabei rücken die technischen Fragestellungen in den Hintergrund und die Frage nach der Wertgenerierung tritt in den Vordergrund, denn die Pipeline wird für eine spezielle fachliche Anforderung erstellt. Einige typische Problemstellungen sind Optimierungs- oder Risikoanalysen, d. h. der Wert der Pipeline ergibt sich z. B. aus den eingesparten Kosten oder aus dem vermiedenen Risiko. Bild 4.3 zeigt exemplarisch, wie diverse Datenverarbeitungsschritte einer Pipeline zusammenhängen. Besonders zu erwähnen ist die Schleife Exploration – Datenbereinigung – Modellierung, die das erzeugte Modell iterativ verbessert. Dabei kann es vorkommen, dass die einzelnen Schritte mehrmals ausgeführt werden und sich dadurch der Fokus der vorhergehenden Iteration stark ändert.

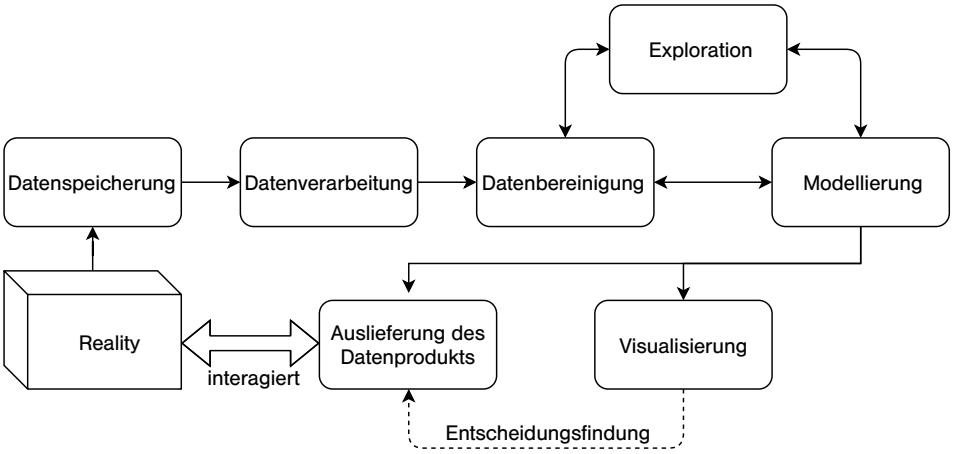


**Bild 4.3** Data-Science-Prozess[1]

Eine Data Pipeline besteht in der Regel aus den folgenden sechs Stufen.

### Stufe 1: Datenerhebung

Die Quelldaten werden in einen dafür vorgesehenen Platz am Data Lake geladen. Dabei können unterschiedliche Datenquellen wie ERP- oder CRM-Systeme mit dem Data Lake verknüpft werden. Wesentlich ist hier, dass es Mechanismen gibt, die verhindern, dass operative Systeme für einen Datenexport zu sehr belastet werden. Eine dieser Methoden ist Change Data Capture (CDC), bei der häufig die Transaction Logs die Basis für die Beladung eines Lakes sind.

Es können aber auch offene Daten (Open Data[2]) aus dem Internet verknüpft und geladen werden, da der Data Lake nicht zwingend ein Schema voraussetzt.

Essenziell hierbei ist, dass die Datenbeladung effizient passiert, keine oder kaum Transformationen ausgeführt werden und alle Daten abgelegt werden.

### Stufe 2: Datenbereinigung

Die Daten werden im nächsten Schritt bereinigt und in ein einheitliches Encoding gebracht. Typischerweise haben unterschiedliche Daten unterschiedliche Encodings, beispielsweise beim Datum ('yyyy-MM-dd' vs. 'dd-MM-yyyy') oder ähnliche Attribute werden einheitlich neu kodiert. Aus Data-Science-Perspektive werden dabei folgende Arten unterschieden:

- *Kategorische oder nominale Variablen*
  Sie haben eine abzählbare Menge an Ausprägungen, wie z. B. Blutgruppen. Ein Merkmal ist, dass es keine Ordnung zwischen den einzelnen Ausprägungen gibt.
- *Ordinale Variablen*
  Ordinale Variablen ähneln den nominalen Variablen, haben aber eine Ordnung zwischen ihren einzelnen Ausprägungen. Ein Beispiel ist die typische Notenskala im Schulsystem.

[1] Data-Science-Prozess basierend auf *https://nl.wikipedia.org/wiki/Datawetenschap*
[2] Daten, die durch eine entsprechende Lizenz die Wiederverwendung ermöglichen

- *Intervallvariablen*
  Eine Intervallvariable ist eine ordinale Variable, die sich in Intervalle unterteilen lässt, beispielsweise Altersgruppen oder Gehaltsklassen.

Je nach Attributart können vorhandene Daten mit 1-aus-n-Kodierung (one-to-Hot Encoding) kodiert werden. Das heißt, die vorhandenen Werte werden durch 0 und 1 ersetzt und die aktuelle Werteausprägung wird als Matrix gespeichert. Das hat den Vorteil, dass es keine Ordnung zwischen den einzelnen Elementen gibt, wie es bei einem klassischen Zahlen-Encoding der Fall wäre.

Das Ziel ist, eine Abbildung auf einen numerischen Wert zu machen, um die Daten für entsprechende Algorithmen vorzubereiten sowie etwaige Datenfehler auszubessern. So erhält man ein analytisches Datenset. Das sind jene Daten, auf denen die einzelnen Machine-Learning-Algorithmen angewandt werden können. Dabei haben analytische Datensets folgende Charakteristika und sehen wie eine große Tabelle aus:

1. Es gibt einen identifizierenden Schlüssel.
2. Sämtliche Attribute sind gleich enkodiert.
3. Die Abfrage und Exploration wird mittels einer Abfragesprache unterstützt.
4. Fehlende oder Standardwerte folgen einem festgelegten Schema.

### Stufe 3: Exploration & Visualisierung

Parallel zur Datenbereinigung werden die Daten visualisiert und explorativ ausgewertet. Häufig werden hierfür verschiedene Darstellungsarten wie BoxPlots, Histogramme, Streudiagramme etc. verwendet. Das Ziel ist es, einen hinreichend genauen Zusammenhang zwischen einer und mehreren Variablen zu finden unter der Berücksichtigung der fachlichen Vorgaben. Um die Daten korrekt interpretieren und „erforschen“ zu können, ist ein gewisses Maß an Domänenwissen notwendig.

### Stufe 4: Modellierung

Basierend auf den vorbereiteten Daten können Modelle gebaut und trainiert werden. Dabei sollen unterschiedliche Modelle ausprobiert und anhand einiger vorher festgelegter Kriterien verglichen werden. Ein gängiges Bewertungskriterium ist Cross-Validation. Hierbei werden die Datenmengen in ähnlich große Mengen aufgeteilt und eine Menge davon wird zum Testen und Validieren verwendet, während die restlichen Mengen zum Trainieren der Modelle dienen. Die Ergebnisse des Testdatensets können dann mittels A/B Testing mit einem anderen Modell verglichen werden. Genaueres zu diesen Verfahren finden Sie in den Kapiteln 5 und 6.

### Stufe 5: Interpretation

Bei der Interpretation der Ergebnisse werden einzelne Elemente in eine ansprechendere, präsentierbare Form gebracht, sodass auch Personen, die nicht täglich mit Big-Data-Technologien arbeiten, Sachverhalte leicht nachvollziehen können.

Eine mögliche Visualisierungsart, die die Vor- und Nachteile einer Pipeline darstellt, ist die Confusion oder Error Matrix. In dieser Visualisierung werden die vom Modell klassifizierten Daten den echten Daten gegenübergestellt und so Stärken und Schwächen des Modells

in einer übersichtlichen Art visualisiert. Ein Merkmal ist dabei, dass die Diagonale der Matrix meist die höchste Anzahl an richtig klassifizierten Daten anzeigt, während von der Diagonale abweichende Nummern falsch klassifizierte Daten darstellen, siehe Bild 4.4.

| | Hund | Katze | Fisch | Hase | Vogel | Sonstiges |
|---|---|---|---|---|---|---|
| Hund | 12 | 2 | 3 | 9 | 0 | 1 |
| Katze | 1 | 12 | 0 | 0 | 5 | 7 |
| Fisch | 2 | 9 | 12 | 0 | 0 | 1 |
| Hase | 0 | 4 | 0 | 12 | 1 | 0 |
| Vogel | 9 | 0 | 7 | 2 | 12 | 1 |
| Sonstiges | 1 | 2 | 4 | 0 | 1 | 12 |

Tatsächliche Werte

Vorhergesagte Werte

**Bild 4.4**
Confusion Matrix

Zu beachten ist, dass hier der Fokus auf fachliche Fragen gelegt wird und dementsprechend der technische Teil in den Hintergrund rückt.

### Stufe 6: Auslieferung in die Produktivumgebung

Ein entscheidender Unterschied zu den vorigen Stufen ist, dass die Produktivumgebung bereits wesentlich mehr Daten enthält oder diese nach einer gewissen Zeit abgearbeitet werden müssen sowie etwaige Service Level Agreements der Firma hinsichtlich Qualität und Zuverlässigkeit eingehalten werden müssen. Zusätzlich muss man sich über den Update-Prozess von einzelnen Modellen oder Parametern Gedanken machen, da je nach Grundannahme die Modelle kontinuierlich verwendet werden und somit schwerer ausgetauscht und gewartet werden können.

Diese sechs Stufen sind meistens in Form einer Big-Data-Pipeline aufgebaut.

Je mehr die Abteilungen lernen, wie Big Data funktioniert bzw. wie es eingesetzt wird, desto mehr können die Schritte automatisiert werden. Idealerweise soll eine Vollautomatisierung, das heißt eine „Ende zu Ende“-Automatisierung angestrebt werden.

## 4.4 Automatisierung der Stufen

Da sich die einzelnen Automatisierungsmethoden voneinander unterscheiden, sind verschiedene Herangehensweisen für die Automatisierung notwendig. Außerdem sind nicht alle Stufen derzeit automatisierbar, weswegen wir uns hier auf die Beschreibung der automatisierbaren Stufen konzentrieren. Die oben beschriebene Schleife Datenbereinigung, Exploration und Modellierung kann aufgrund der unterschiedlichen Eingangsdaten nur manuell effizient durchgeführt werden.

### 4.4.1 Datenerhebung

Im Big-Data-Umfeld ergeben sich meist folgende typische Datenladeszenarien:

- Flache Dateien wie JSON, CSV, HTML …
- Datenbankschnittstellen zum Datenwarehouse wie Oracle, Teradata …
- Streaming-Daten wie zKafka oder RabbitMQ …

Die klassischen Verarbeitungstypen können generisch programmiert werden, und dann mit wenig Aufwand auf die Daten des entsprechenden Anwendungsfalls der Datenquelle adaptiert werden. Bei der Datenerhebung werden die Daten nicht korrigiert, sondern nur von einer Quelle gelesen und auf den Data Lake in der Ursprungsform gespeichert. Die Datenerhebung muss der Startpunkt des Datenaudits sein, da nur hier auf die Originaldaten zugegriffen werden kann, bevor die Daten verändert werden. Dabei muss der Aufwand des Audits gegenüber den anderen Randbedingungen, wie Verarbeitungszeit der Daten, abgeschätzt werden, da gerade bei Streaming-Daten eine gewisse Latenzzeit eingehalten werden muss.

Durch die generische Programmierung lassen sich in weiterer Folge auch zukünftige Pipelines entsprechend einfach automatisieren, da sie bis auf wenige Details ähnlich aufgebaut sind. Die Automatisierung unterscheidet zwischen zwei Szenarien:

1. Kontinuierliche Beladung oder
2. Beladung zu einem gewissen Zeitpunkt.

Die kontinuierliche Beladung lädt permanent kleinere Datenmengen in den Data Lake und ist somit über einen längeren Zeitraum aktiv. Dabei werden das Überwachen der Beladung und die Orchestrierung meist von dem gewählten Framework erledigt und es muss nur mehr sichergestellt werden, dass Start und Stopp der Automatisierung reibungslos funktionieren.

Bei der Beladung zu einem gewissen Zeitpunkt werden viele Daten zu einem bestimmten Zeitpunkt, meist täglich außerhalb von Bürozeiten, auf den Data Lake geladen. Diese Beladung funktioniert ähnlich wie die kontinuierliche Beladung, hat aber den Vorteil, dass entsprechende Zeitbedingungen entfallen. Durch die erhöhte Datenmenge ist bei punktueller Verarbeitung die Wahrscheinlichkeiten höher, Ressourcenprobleme oder Skalierungsprobleme zu bekommen.

### 4.4.2 Datenbereinigung

Die Datenbereinigung ist Schätzungen zufolge die zeitaufwendigste Verarbeitungsstufe der manuellen Data-Science-Modellierung und sie ist dementsprechend schwer zu automatisieren. Bei diesem Schritt werden einzelne, mitunter sehr unterschiedliche Datenquellen konsolidiert und zu einem analytischen Datensatz verschmolzen. Wie oben schon erwähnt, können gleiche Variablen unterschiedlich in den Quelldaten modelliert sein. Diese Mehrdeutigkeit der Variablen wird schrittweise im analytischen Datensatz behoben, sodass alle Variablen eindeutig sind. Die unten gezeigte Tabelle veranschaulicht das Problem an der Modellierung des Geschlechts. Zwecks Vereinfachung wurde angenommen, dass es sich um eine binäre Variable, d. h. eine Variable mit zwei Ausprägungen (Männlich und Weiblich) handelt.

| | | |
|---|---|---|
| M/W | Männlich/Weiblich | 1/0 |
| M/F | Male/Female | |

Um die Problematik der Eindeutigkeit von Variablen in weiterer Folge zu verringern, empfiehlt es sich einen Datenkatalog aufzubauen, in dem festgehalten wird, wie in unserem Fall das Geschlecht auszusehen hat, und jedes Attribut in Zukunft gleich zu modellieren. So kann z. B. per Policy festgelegt werden, dass das Geschlecht mit M/W gespeichert wird. Dadurch kann auf lange Sicht erreicht werden, dass die Daten wie in einem Katalog auswählbar sind, und so leichter kombiniert werden können.

Ebenso soll in dem Katalog festgehalten werden, wie die Daten ursprünglich modelliert worden sind, sodass bei Bedarf nachgeschlagen werden kann, woher die Daten gekommen sind, wie sie ausgesehen haben und welche Werte ungefähr erwartet werden können. Basierend auf dem Katalog können dann weitere Prozesse aufgebaut werden, wie z. B. rechtliche Prozesse, Data Lineage, oder es können qualitätssichernde Maßnahmen getroffen werden, um die Datenbereinigung zu unterstützen und so ein Set von Best Practices zu etablieren.

Wie in den Anforderungen in Kapitel 2 beschrieben, kann der Katalog auch für Verifikationsmaßnahmen herangezogen werden, indem die selektierten Variablen sowie das analytische Modell mitdokumentiert werden.

Die Stufen der Visualisierung und Modellierung sind kaum automatisierbar, da diese von den Daten und den einzelnen Anwendungsfällen abhängen. Es gibt aber durchaus Applikationen, die anhand der Art der Daten Visualisierungsvarianten vorselektieren.

## 4.5 AnalyticsOps und DataOps

Soll eine Daten-Pipeline in eine Software-Umgebung der Produktivumgebung eingesetzt werden, dann wird das anhand eines vorher festgelegten Prozesses mit definierten Parametern geschehen. Dieser Prozess lehnt sich in der Regel an Continuous Delivery (CD)[3], einen mehrphasigen Software-Auslieferungsprozess, an. Continuous Delivery sieht dabei vor, dass jederzeit die aktuelle Software in die Produktion ausgeliefert (delivered) werden kann. CD kann als Sammlung von Best Practices, Werkzeugen und Prozessen verstanden werden, die dieses Ziel möglichst automatisiert unterstützen sollen.

Analytics Operation oder kurz AnalyticsOps oder DataOps ist eine Weiterentwicklung von DevOps, die speziell für Big-Data-Prozesse angepasst worden ist und auf die in Abschnitt 4.3 beschriebenen Punkte abzielt. DevOps ist ein Ansatz zur Prozessverbesserung, der es erlaubt, eine möglichst effiziente und effektive Auslieferung von Software in die Produktion durchzuführen. Um dies zu erreichen, arbeiten Development (Entwicklung) und Operation (IT-Betrieb) stärker zusammen als bisher und sollten idealerweise als Einheit betrachtet werden.

[3] Weiterführende Literatur: *https://continuousdelivery.com/*

Die Autoren möchten die Qualitätssicherungen besonders hervorheben, da diese essenziell für die Funktionsweise einer CD-Umgebung sind. Zusätzlich wird die Auslieferung in Abschnitt 4.6 separat behandelt, da erst die komplette Auslieferung einer Data Pipeline einen Mehrwert für das Unternehmen schafft.

### Qualitätssicherung und Tests

Qualität wird bei Software mit Tests geprüft, die sicherstellen, dass die Software gemäß den vorher spezifizierten Anforderungen funktioniert. Diese Anforderungen können in Komponenten-, Modul- und Integrationstests für einzelne Teile der Software unterteilt werden. Diese Tests werden jedes Mal, wenn der Code geändert wird, ausgeführt und stellen sicher, dass die Kernfunktionalität der Software erhalten bleibt, selbst wenn entsprechende Teile ausgetauscht werden. Integrationstests sind meistens in einer Big-Data-Umgebung schwerer zu realisieren, weswegen diese häufig erst in der Testumgebung stattfinden. Um Integrationstests durchführen zu können, müssen Teile der Umgebung im Speicher nachgebaut werden und ein kontrollierter Input erzeugt werden, der danach mit dem Output verglichen wird.

Eine Metrik, um die Qualität von Code quantifizierbar zu machen, ist die Messung des Verhältnisses von abgedecktem Code zum gesamten Code. Dieses Verhältnis sollte hoch sein, um möglichst viele Fehlerquellen während der Erstellung auszuschließen.

### Qualitätssichernde Instanzen

Es gibt im Unternehmen Ressourcen, die die Qualitätssicherung durchführen. Basierend auf dem Ergebnis wird die produzierte Software ein weiteres Mal überarbeitet oder der Auslieferungsprozess wird gestartet. Angepasst auf Big Data ist es auch unerlässlich, dass die Vorhersage der erstellten Modelle getestet wird, beispielsweise indem die Accuracy, Confusion Matrix, ROC oder Cross Validation verglichen werden. Genauso wichtig ist es, die Eingangsdaten zu testen, indem diese beispielsweise ausgetauscht oder variiert werden. So wird das Modell auf komplett neue Daten getestet, damit sich kein Fehler einschleicht. Bedauerlicherweise lässt sich dieser Schritt derzeit nur unzureichend automatisieren.

Generell unterscheidet sich die Entwicklungsumgebung teilweise signifikant von der Produktionsumgebung. Die Umgebungen unterscheiden sich hinsichtlich der Anzahl der Hardware-Ressourcen und der Datenmenge sowie dadurch, dass einzelne Fachabteilungen von der Produktionsumgebung eine höhere Qualität erwarten als von einer Entwicklungsumgebung.

Dabei ändern sich die Anforderungen an das entwickelte Modell: Es muss autonom und ohne menschliche Interaktion arbeiten, bis es ausgetauscht oder abgelöst wird. Es soll dabei kontrollierte Ergebnisse liefern und somit überwachbar sein.

Derzeit gibt es zwei unterschiedliche Arten, wie Data-Science-Modelle verwendet werden können. Entweder wird das Modell mit einem zusätzlichen Programm betrieben und die Entwicklung erfolgt losgelöst von dem Treiber-Programm, oder das Modell wird in einem Container, also containerized betrieben und dann auf die Daten angewandt. Containerization beschreibt dabei einen Ansatz, in dem das Modell in einem Container gekapselt wird. Bei beiden Verwendungsarten ist von manuellen Auslieferungen in die Produktivumgebung abzuraten, da sich bei manuellen Prozessen meist weitere Fehler einschleichen.

Als Konsequenz heißt das, dass die Modelle aus einer Versionskontrolle wie Git kommen und über einen Buildserver (Jenkins) in der entsprechenden Umgebung (semi-)automatisch bereitgestellt werden sollten.

Üblicherweise wird in drei unterschiedlichen Umgebungen gearbeitet, abhängig vom Reifegrad der Software.

#### Development

Die Development-Umgebung enthält den aktuellen Stand der Software und wird häufig aktualisiert. Das Ziel ist, den Entwicklungszyklus zu unterstützen und neue Ideen auszuprobieren oder Probleme zu beheben.

#### Test

In der Testumgebung wird die Software getestet, bevor sie ausgerollt wird. Hier werden System- und Integrationstests gemacht, um das Zusammenspiel der Software mit den restlichen Komponenten sowie den Prozessen zu erproben. Es soll aber auch bereits ausprobiert werden, wie die Software unter möglichst realitätsnahen Bedingungen funktioniert.

#### Produktivumgebung

Ist die Testphase erfolgreich, dann wird während der monatlichen Release-Zyklen die Software in der Produktivumgebung installiert. Erst dann erzielt sie einen Mehrwert für das Unternehmen.

## 4.6 Auslieferung

### 4.6.1 Containerization und Kubernetes

Eine Technologie, die dieses Verhalten unterstützt, ist Containerization. Ein prominentes Beispiel für Containerization ist Docker oder die Open Container-Initiative.[4] Ein Container ist eine Abstraktion von einzelnen Betriebssystemkomponenten und fügt eine Zwischenschicht zwischen Betriebssystem und dem entwickelten Code ein. Ein Container enthält alle Komponenten wie Konfigurationen, Bibliotheken, Programmabhängigkeiten, die notwendig sind, um den entsprechenden Code auszuführen, mit dem Ziel diesen vom Betriebssystem zu abstrahieren und zu isolieren. Der Container kann also als leichtgewichtige Virtualisierung betrachtet werden, ohne den zusätzlichen Aufwand, ein eigenes Betriebssystem bereitzustellen. Dadurch kann „containerized“ Software sehr leicht skaliert werden, da mit wenig Aufwand ein neuer Container erzeugt werden kann. Ebenso kann der enthaltene Code leicht von einer Umgebung in eine andere portiert werden, da er alles enthält, um die entwickelte Software auszuführen, und auf Standardschnittstellen aufsetzt.

[4] [Open Container 2018]

Mit einer steigenden Anzahl an Containern wird ein effizienter Mechanismus notwendig, um die einzelnen Instanzen zu verwalten. Ein Produkt, das diese Funktionalität bereitstellt, ist Kubernetes. Kubernetes folgt dabei dem Master-Slave-Prinzip, d. h. dass eine zentrale Stelle die einzelnen Container (in Kubernetes Pods genannt) verwaltet, und kommuniziert mit den einzelnen Treiberprogrammen (Kubelets) auf einem Node, einer Maschine, die die Software ausführt. Das Kubelet orchestriert einzelne Container-Instanzen und kümmert sich ggf. um den Neustart, falls ein Container in einem nicht zulässigen Zustand ist.

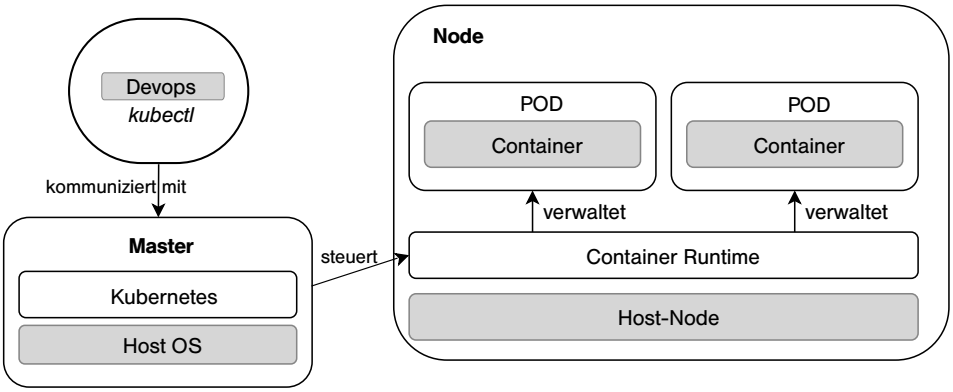


**Bild 4.5** Kubernetes-Architektur[5]

Kubernetes unterstützt auch Rolling Updates, d. h. Updates, die während des Betriebs ausgeliefert werden können. Generell können zwei Update-Arten von Containern unterschieden werden.

## 4.6.2 Modell-Update

Bevor auf die beiden Update-Arten eingegangen wird, soll kurz auf den Unterschied zwischen R- und Python-Modellen eingegangen werden, da dies derzeit die häufigsten Data-Science-Sprachen sind.

### Modelle in R

R hat seinen Ursprung in der akademischen Welt und wird von verschiedenen Forschungsinstitutionen als Open-Source-Projekt vorangetrieben. Das hat zur Folge, dass neue Modelle häufig zuerst in R implementiert werden, bevor sie in anderen Sprachen adaptiert werden. Die Sprache R ist übersichtlich gehalten und vereint verschiedene Konzepte mit dem Ziel eine sehr flexible und erweiterbare Sprache zu haben.

Um den Konflikt zwischen verschiedenen Konzepten und Versionen der Bibliotheken zu vermeiden, werden R-Data-Pipelines üblicherweise in einem Container ausgeliefert. Ein Nachteil ist jedoch, dass immer eine komplette Umgebung im Container aufgebaut werden muss, und diese N-mal die Ressourcen benötigt. Das Ursprungsproblem bei R-Modellen ist,

[5] *https://www.redhat.com/de/topics/containers/what-is-kubernetes*

dass es keine effiziente Bibliotheksverwaltung gibt und die Konflikte mit dem Container-Ansatz umgangen werden können. Darüber hinaus kann R-Code auch mit Data Processing Engines wie Spark ausgeführt werden.

#### Modelle in Python

Ähnlich wie R ist Python ebenfalls übersichtlich gehalten und wurde entwickelt, um gut lesbaren Programmierstil zu fördern. Durch die Fokussierung von Big-Data-Firmen auf Python sind aktuelle Machine-Learning-Bibliotheken in Python verfügbar bzw. werden bevorzugt in Python implementiert. Sie können direkt verwendet werden, ohne dass man auf den Komfort von modernen Entwicklungsumgebungen verzichten muss.

Diese Entwicklungsbibliotheken haben den Vorteil, dass Python gängige Entwicklungswerkzeuge wie eine Bibliotheksverwaltung enthält und dementsprechend Best Practices von Programmiersprachen etabliert sind.

Auch Python-Code kann mit Data Processing Engines wie Spark ausgeführt werden.

Ein containerized Modell kann mit dem Rolling Update von Kubernetes angepasst oder ersetzt werden. Dabei werden sequenziell die vorhandenen Pods durch neue Pods ersetzt, die die letzten Änderungen der Modelle enthalten. Dadurch sind kurzzeitig zwei potenziell unterschiedliche Versionen aktiv, und es ist empfehlenswert, sich ein entsprechendes Migrationsszenario zu überlegen, um etwaige Dateninkonsistenzen zu vermeiden.

Als Alternative kann der Weg über ein Treiberprogramm gewählt werden. Hier ruft das Programm, welches die Daten verarbeitet, eine Schnittstelle im Container auf oder portiert das Modell nativ in den datenverarbeitenden Code.

Diese Schnittstelle zwischen den Containern und dem Code ist meistens eine Web-Schnittstelle, die die Antwort in einer bestimmten Zeit in einem bestimmten Format zur Verfügung stellen muss, da sonst Probleme hinsichtlich der Latenzzeiten in der Weiterverarbeitungskette auftreten. Das Treiberprogramm kann ein Big-Data-Framework sein, welches die entsprechenden Endpunkte aufruft und die Web-Antwort in den Daten anreichert, bevor die Daten in der Kette weiterverarbeitet werden. Es wird also das Data-Science-Modell strikt von dem Code getrennt, der die Hauptlast der Daten trägt und verarbeitet.

Alternativ kann das Modell, das in einer Entwicklungsumgebung erstellt worden ist, mit einem Werkzeug in eine Umgebung portiert werden, das die entsprechenden Schnittstellen zur jeweiligen Modellsprache besitzt. Ein Vorteil dieser Lösung ist, dass der Ressourcenverbrauch effizienter ist als beim vorigen Ansatz, jedoch können Inkompatibilitäten zwischen den Versionen auftreten, beispielsweise wenn ein bestimmter Algorithmus nicht unterstützt wird. Zudem kann es Skalierungsprobleme bei einigen Bibliotheken geben, da sie nicht auf einen parallelisierten Betrieb ausgelegt sind.

### 4.6.3 Modell- oder Parameter-Update

Falls ein Modell oder Parameter ausgetauscht werden soll, der eigentliche Code des Modells aber gleich bleibt, so kann dies durch ein Update der entsprechenden Datei oder des entsprechenden Datenbank-Eintrags geschehen. Dieser Parameter wird sozusagen im laufenden Betrieb ausgetauscht. Aus diesem Grund empfiehlt es sich eine Zwischenschicht zwi-

schen dem Modell-Code und dem eigentlichen Modell einzubauen, etwa mit PMML oder Onnx. Diese Schicht stellt sicher, dass die einzelnen Modelle portierbar bleiben. Wie in Abschnitt 4.6.2 beschrieben, müssen auch die Parameter der einzelnen Modelle portierbar sein, da sichergestellt werden muss, dass diese mit dem Update optimal zusammenarbeiten. Sie können z. B. auch in JSON oder XML abgelegt werden.

Unabhängig davon, wie die Parameter geändert oder zur Verfügung gestellt werden, muss ein Weg gefunden werden, die Parameter entsprechend der Anzahl der Aufrufe zu skalieren.

### 4.6.4 Modell-Skalierung

Werden Modelle in der Produktion skaliert, so wird das Modell zunächst den entsprechenden Datenmengen ausgesetzt. Dabei können verschiedene Probleme auftreten, wie z. B. Ressourcenprobleme oder im schlimmsten Fall, dass die Modelle auf falschen Trainingsdaten berechnet worden oder anders enkodiert sind. Das Ziel der Qualitätssicherung ist, diese Fehler möglichst früh zu vermeiden, indem verschiedene Aspekte von Big Data während der Erstellung und Modellierung betrachtet worden sind. Das Ziel bei der Skalierung ist, das vorher entwickelte Modell effizient, das heißt mit möglichst wenig Ressourcen, zu betreiben.

Wird ein Modell horizontal skaliert, also über mehrere Computer verteilt berechnet, so ergibt sich das Problem, dass das Modell unter Umständen nur eine Teillösung berechnet und keine global eindeutige Lösung. Im nächsten Schritt müssen die einzelnen Teillösungen dann zu einer Gesamtlösung verarbeitet werden. Das heißt, dass die Datenmenge auf die verfügbaren Rechner gleich verteilt wird und dann pro Computer das Modell die Verarbeitung der Daten durchführt. Sobald diese fertig ist, wird die Gesamtlösung berechnet.

Im Gegensatz dazu steht die vertikale Skalierung, die hardwareseitiges Aufrüsten vorsieht. Das heißt, dass mehr physikalische Ressourcen dem Computer zur Verfügung gestellt werden, um so die Berechnung effizienter zu machen. Es wird dann die Gesamtlösung auf einem Computer berechnet. Der Nachteil ist, das diese Skalierungsart nur durchführbar ist, wenn der Computer über entsprechende ungenutzte Hardware-Ressourcen verfügt.

## 4.7 Feedback in die operationalen Prozesse

Die Aktualisierung oder Ersetzung von Modellen muss sich auch in der Data Governance niederschlagen, da diese Änderungsinformation der Modelle einen essenziellen Input für weitere Verbesserungen liefert. Als Betrachtungsweise für ausgelieferte Modelle empfiehlt sich wieder die Blackbox-Sicht, da diese Modelle nicht verändert werden können und trotzdem ein Indikator für den Status sein müssen.

Folglich sollten, zusätzlich zu den vorhergehenden Parametern, die operationalen Metadaten wie benötigte Zeit, verfügbarer Speicher etc. persistiert werden, um eine Analyse zu ermöglichen.

Der Einbau von Modellen in die operationalen Prozesse sollte dabei einen Eindruck vermitteln, welche Parameter, Variablen oder Features großen Einfluss auf die Qualität bzw. auf das Verhalten des Modells haben. Der Einfluss der Parameter auf das Modell kann ggf. simuliert und iterativ verbessert werden, um weniger wichtige Features und Parameter basierend auf deren vorher festgelegte Kriterien zu entfernen. Das Ziel dieser Vorgehensweise ist, einzelne Modelle performanter, kleiner und somit effizienter zu machen.

## 4.8 Fazit

Data Pipelines sind ein essenzieller Teil jeder Big-Data-Strategie, da sie den Mehrwert für Unternehmen erzielen, indem sie Modelle (semi)-automatisch auf Daten anwenden. Je größer dabei der Grad der Automatisierung ist, desto höher wird der erzielte Mehrwert sein. Die Art, wie Modelle angewendet werden, unterscheidet sich, je nachdem, welcher Anwendungsfall betrachtet wird und welche Modelle verwendet werden.

Ebenso wichtig wie ein hoher Automatisierungsgrad ist, dass man sich auch überlegt, wie das erstellte Modell ausgetauscht und aktualisiert wird. Im Update-Prozess ist es essenziell, dass das Modell portierbar bleibt, da es sonst im schlimmsten Fall zu Unterbrechungen der Pipeline kommen kann, besonders wenn sich die Entwicklungsumgebung und Programmiersprache stark vom Produktivsystem unterscheiden.

Um Updates ohne Probleme durchzuführen, ist es auch wichtig, dass dokumentiert wird, welche Parameter welchen Einfluss auf das Modell haben. Dieses Wissen kann, falls es Probleme gibt, zur Fehlerbehebung herangezogen werden und andere Prozesse, wie z. B. die DSGVO, unterstützen.

## 4.9 Weiterführende Literatur

Folgende Quellen können für den Leser interessant sein:

| Thema | Weiterführender Link |
|---|---|
| Data Pipelines | *https://www.oreilly.com/ideas/three-best-practices-for-building-successful-data-pipelines* |
| Beginners Guide to DataScience | *https://towardsdatascience.com/a-beginners-guide-to-the-data-science-pipeline-a4904b2d8ad3* |
| Serverless | *https://martinfowler.com/articles/serverless.html* |
| Pipeline-Beispiel | *https://spandan-madan.github.io/DeepLearningProject/docs/Deep_Learning_Project-Pytorch.html* |
| DataOps Manifesto | *http://dataopsmanifesto.org/* |
| Kubernetes | *https://www.redhat.com/de/topics/containers/what-is-kubernetes* |
| Continuous Delivery | *https://continuousdelivery.com/* |

# Index

## D



## E



## F

## N



## O



## P



## Q



## R

## S



## T



## U



## V



## W



## X

## Y



## Z