WHITEPAPER

Advanced Analytics

von Finn Schröter, Bernd Themann und Alexandros Xiagakis | ISR Information Products AG

1 | ÜBERBLICK

Im Zeitalter der Digitalisierung, in dem Unternehmen mehr und mehr Daten erheben, gewinnt Advanced Analytics zunehmend an Bedeutung. Beispielsweise lässt sich mithilfe von Advanced Analytics ein Scoring-Modell zur automatisierten Kreditentscheidung aufbauen. Auf Ihrem Weg zu Advanced Analytics werden Ihnen einige Begriffe, wie Data Science, Business Intelligence (BI), Machine Learning, Künstliche Intelligenz (KI) und Explainable AI begegnen. In diesem Whitepaper werden wir Sie in das Thema einführen und diese Begriffe sortieren und erläutern. Viele der Konzepte sind bereits seit mehreren Jahren bekannt. Eine Umsetzung in die Praxis gestaltet sich allerdings als schwierig, da die richtigen Daten am richtigen Ort vorhanden sein müssen. Der Grund dafür ist häufig eine veraltete IT-Infrastruktur. Das bedeutet in diesem Fall nicht, dass die Architektur schlecht ist, sondern, dass auf die neuen Bedürfnisse noch nicht eingegangen werden konnte. Im folgenden Whitepaper geben wir Ihnen einen Überblick über verschiedene Umsetzungsmöglichkeiten zur Modernisierung Ihrer IT-Architektur. Dabei orientieren wir uns an dem „Data Pipeline Modell“ und zeigen Ihnen Schritt für Schritt einzelne Handlungs- und Gestaltungsmöglichkeiten auf.

INHALT

2 | BEGRIFFSEINORDNUNG

Dieses Kapitel widmet sich ganz der Theorie. Wir stellen unser Verständnis von Advanced Analytics vor und grenzen diesen Begriff von Data Science und BI ab.

2.1 | DATA SCIENCE/ ADVANCED ANALYTICS

Data Science ist ein interdisziplinäres Themenfeld. Wie der Name bereits impliziert, spielen hier Daten eine sehr große Rolle. Dabei umfasst Data Science einen Mix aus den Themenfeldern **IT-Wissen, mathematisches Wissen** und **Fachwissen** (s. Abb. 1). Dieser Methoden-Mix aus allen Teildisziplinen hat zum Ziel, aus Daten Wissen zu generieren, welches bisher verborgen geblieben ist.

Dabei stellt das **IT-Wissen** das technische Framework bereit und stellt damit sicher, dass in einem Data-Science-Projekt die Ideen technisch umgesetzt werden können.

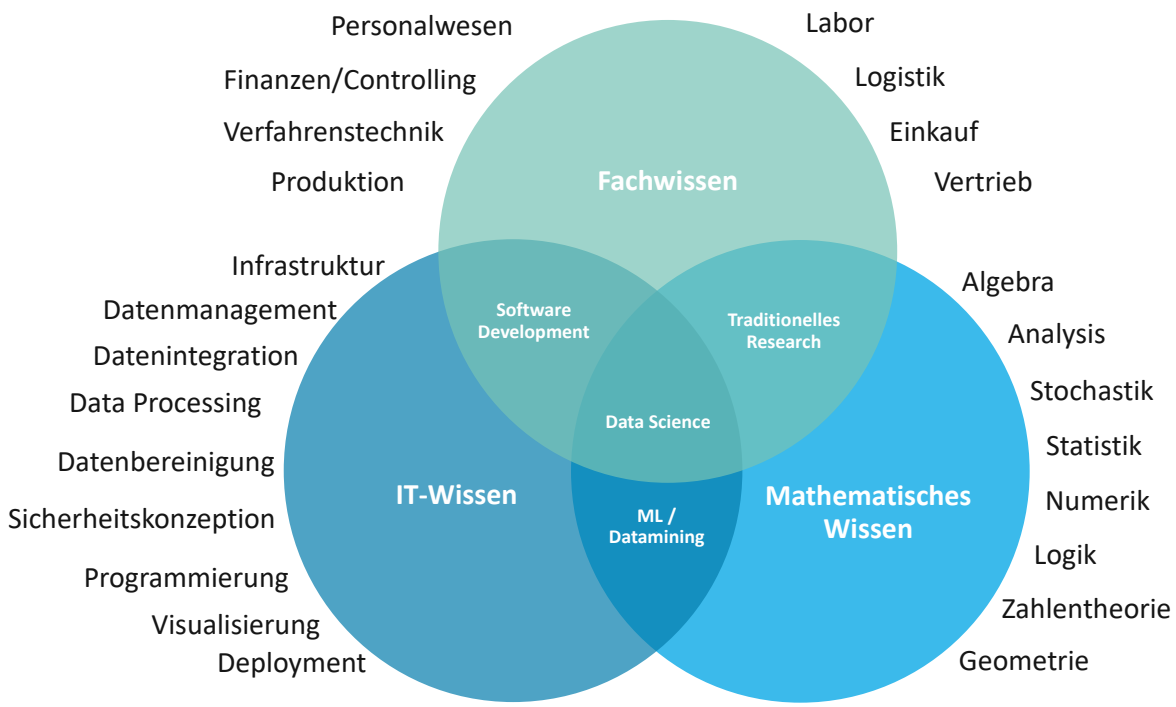

Abb. 1: Die drei Komponenten von Data Science

Ein wichtiger Teilbereich des IT-Wissens (insbesondere in produktiver Umgebung) ist hier das Datenmanagement inklusive der Datenintegration, der Datenbereinigung und des Data Processing. Ein weit verbreitetes Zitat lautet: „Bullshit in, Bullshit out." Dieses bedeutet, dass es keine Rolle spielt, wie gut ein Algorithmus oder eine Prognose auf den Daten berechnet werden kann, solange die Trainingsdaten (sprich die Daten, auf denen der Algorithmus trainiert wird) nicht auf einem ausreichend hohen Qualitätslevel vorliegen.

Die Algorithmen werden durch **mathematisches Wissen** entwickelt. Dabei werden, z.B. Erkenntnisse aus Stochastik, Algebra und Geometrie miteinander verbunden. Die Verknüpfung aus mathematischem Wissen und IT-Wissen führt bereits zu vielen guten Algorithmen des Machine Learnings (ML) und Data Minings. Allerdings werden diese Algorithmen häufig auf künstlich erzeugten Daten erstellt, sodass die Datenqualität keine Probleme bereitet.

Auch die künstliche Intelligenz ist in diesem Bereich einzuordnen, da durch Kombination verschiedener ML-Algorithmen versucht wird, menschliche Intelligenz nachzubilden. Ohne einen Business Case lassen sich Ergebnisse dieser Algorithmen allerdings aus Sicht eines Unternehmens kaum bewerten. Aus Manager-Sicht lohnt sich der Aufwand nicht, wenn es ein signifikantes Verhalten in den Daten gibt, das jedoch nur zu einem marginalem Umsatzwachstum führt.

Mithilfe von **Fachwissen** müssen sinnvolle Business Cases definiert werden, um ein Unternehmen voran zu bringen. Zusätzlich ist es notwendig, um die Daten aus dem operativen Betrieb zu verstehen.

Der Begriff Data Science beschreibt damit eine wissenschaftliche Sicht, wo hingegen der Begriff Advanced Analytics sich aus der Erweiterung der Business Intelligence und damit aus der Praxis heraus entwickelt. Grundsätzlich beschreiben beide Begriffe jedoch die gleiche Thematik und werden deswegen häufig als Synonyme verwendet.

2.2 | BI VERSUS ADVANCED ANALYTICS

Nachdem es sich bei Data Science und Advanced Analytics nun um zwei Bezeichnungen für dasselbe Thema handelt, stellt sich die Frage, ob es sich mit **Business Intelligence** ähnlich verhält. Diese Frage kann jedoch verneint werden, auch wenn es Ähnlichkeiten gibt. So können in beiden Themengebieten Big Data (aber auch „small" Data) durch Datenanalysen untersucht und anschließend visualisiert werden. Bei der Zielsetzung unterscheiden sich **BI** und **Advanced Analytics** signifikant. Während es bei **BI** darum geht, die **Vergangenheit zu analysieren** und Berichte aus vergangenen Daten zu erstellen, richten sich die Fragestellungen des **Advanced Analytics** auf die **Zukunft**. Dabei können sogar die gleichen Daten verwendet werden. Damit bringen sich Unternehmen in eine proaktive Position und können auf Veränderungen reagieren, bevor diese eintreten. Weitere Unterschiede, die die Fortschrittlichkeit von Advanced Analytics beschreiben, sind in Tabelle 1 aufgeführt.

	BUSINESS INTELLIGENCE	ADVANCED ANALYTICS
ORIENTIERUNG	Der Blick zurück	Zukunftsgerichtet
FRAGETYPEN	• Was ist passiert? • Wann /Wer /Wie viel?	• Was wird passieren? • Was wird passieren, wenn wir diesen einen Parameter ändern? • Was passiert als nächstes? • Welche Handlungsempfehlungen gibt es?
METHODEN	• Bericht erstatten (KPIs, Messgrößen) • Automatisierte Überwachung/ Alarmierung (Schwellenwerte) • Dashboards • Scorecards • OLAP (Würfel, Slice & Dice, Drilling) • Ad-hoc Abfrage	• Predictive Modeling • Data Mining • Text Mining • Big Data Analytics • Descriptive Modeling • Statistische/ Quantitative Analyse • Simulation & Optimierung • Machine Learning • Künstliche neuronale Netze
BIG DATA	Ja	Ja
DATENTYPEN	Strukturiert, manche unstrukturiert	Strukturiert und unstrukturiert
WISSENSGENERIERUNG	Manuell	Automatisch
BENUTZER	Gewerbliche Nutzer	Data Scientists, Wirtschaftsanalytiker, IT, Gewerbliche Nutzer
UNTERNEHMERISCHE INITIATIVE	Rückwirkend	Vorausschauend und präskriptiv

Tabelle 1: Vergleich Business Intelligence und Advanced Analytics (in Anlehnung an (Tiedemann 2018))

2.3 | ENTWICKLUNG

Neben den klassischen Themen wie Datenqualität, Datensicherheit und Operationalisierung liegen aktuell einige moderne Ansätze im Bereich des Data Science im Trend:

AUTO-MACHINE-LEARNING (AUTOML)

AutoML hat zum Ziel, verschiedene Prozesse des maschinellen Lernens zu **automatisieren** und ermöglicht Anwendern ohne Data-Science-Wissen, die Vorteile des maschinellen Lernens zu nutzen.

EXPLAINABLE AI

Explainable AI beschreibt einen Ansatz zur Entwicklung von Algorithmen, bei dem im Nachhinein **nachvollziehbar** bleibt, wie und warum ein Modell **Entscheidungen getroffen** hat. Viele Algorithmen gleichen heutzutage einer Blackbox, in die Daten hineingegeben werden und ein Ergebnis ausgegeben wird, ohne zu wissen, was genau dazwischen passiert.

NATURAL LANGUAGE PROCESSING (NLP)

NLP beschreibt die Möglichkeit einem Computer **menschliche Sprache**, bzw. **Texte**, beibringen zu können (z.B. Alexa, Siri). Dabei geht es nicht nur um das reine Niederschreiben, sondern vor allem um die Analyse und Weiterverarbeitung der gesprochenen Worte.

CLOUD BASED BIG DATA SCIENCE

Cloud based Big Data Science beschreibt die **Speicherung** der Daten und **Implementierung** der Algorithmen für Data Science Projekte in der Cloud. Dabei geht es nicht nur darum die Daten in der Cloud zu speichern, sondern auch zu verarbeiten. Mit wachsender Datenmenge werden immer mehr Ressourcen benötigt, um mithilfe von Data Science Algorithmen die Modelle zu trainieren. Eine flexible und skalierbare Möglichkeit bieten Cloud-Anbieter dafür bereits.

DATA HUB

Im Gegensatz zu einem **Data Lake**, in dem jegliche Form von Daten unbearbeitet abgelegt wird, verfolgt der **Data Hub** mit den gleichen Daten einen anderen Ansatz. Diese verbleiben (wenn möglich) im Ausgangssystem und werden über den Data Hub als **zentrale Plattform** zur Verfügung gestellt. Im Data Hub werden die unterschiedlichen Datenquellen integriert und können anschließend beliebig zusammengefügt werden. Außerdem werden die Daten neu indiziert und suchbar gemacht. Dadurch sind die gesuchten Daten leichter zu finden und können einer breiteren Anwenderschicht zur Verfügung gestellt werden.

Abb. 2: Das Data-Pipeline-Modell

3 | DATA-PIPELINE-MODELL

Das **Data-Pipeline-Modell** eignet sich gut, um die einzelnen Schritte, die für den Aufbau eines Advanced Analytics Cases durchlaufen werden müssen, zu strukturieren. Im Gegensatz zum **CRISP-DM-Modell** ist kein Feedback-Loop oder ein Zurückspringen zwischen den Schritten vorgesehen. Dies wird jedoch in der Realität sicherlich vorkommen bzw. ist sicherlich notwendig. Stattdessen lassen sich mit dem Data-Pipeline-Modell schnell die technischen Komponenten bzw. Funktionen bestimmen. Aus diesem Grund wird das Modell im nachfolgenden Abschnitt genutzt, um mögliche Architektur-Designentscheidungen zu strukturieren und zu diskutieren.

3.1 | COLLECT

Abhängig von der Anzahl und Beständigkeit der Quellsysteme, der Netzwerkanbindung, der Datenmenge und der Formate, kommen unterschiedliche Topologien und Abrufmechanismen für die Datensammlung in Frage.

Grundsätzlich können Quellsysteme die Eingangsdaten aktiv senden (**Push**) oder diese werden bereitgestellt und müssen abgeholt (**Pull**) werden.

Beim **Push-Mechanismus** müssen nur die Quellsysteme den Endpunkt kennen, an welchen die Daten geschickt werden müssen. Sie können eigenständig Änderungen an den Quelldaten weiterleiten oder bei Wiederverfügbarkeit einer instabilen Netzwerkverbindung die Daten zeitverzögert senden. Auf diese Weise lassen sich robuste Daten-Pipelines aufbauen, die (Near-)Realtime-Daten an nachfolgende Systeme weiterleiten.

Bei einem **Pull-Mechanismus** hingegen muss das abholende System Kenntnis von allen Quellsystemen haben und aktiv die Daten von diesen anfragen. Der Vorteil dabei ist, dass dieses System das Scheduling hierfür eigenständig übernehmen kann und so ggfs. Lastspitzen vermieden werden können. Nachteilig dabei ist, dass Änderungen an den Quelldaten nur über kurze Abfrageintervalle, welche wiederum unnötige Last erzeugen könnten, zeitnah weitergeleitet werden können.

Der Push-Mechanismus wird gerade in einem **IoT-Umfeld**, bei dem sehr viele Endgeräte Daten senden (können), genutzt. Zum einen muss dort die Netzwerkverbindung nicht immer stabil sein und zum anderen können so leicht neue Endgeräte hinzugefügt werden. Ein weiterer Vorteil, der sich daraus ergibt, ist, dass die IoT-Endgeräte nicht ständig online sein müssen, um Daten bereitstellen zu können, sondern durch den Push-Mechanismus die Daten nur versenden, wenn die Endgeräte genutzt werden bzw. eingeschaltet sind und somit Strom sparen können.

Eine vergleichsweise einfache Lösung, um ein solches Szenario umzusetzen, ist der Einsatz eines **Brokers**, der nach dem **Pub-Sub-Prinzip** arbeitet. Daten können von den Quellsystemen an den Broker publiziert werden und die Zielsysteme können ihr Interesse an den Daten durch eine **Sub**scription ausdrücken. Werden neue Daten publiziert, leitet der Broker diese automatisch an die Subscriber weiter.

Wird ein Push-Mechanismus eingesetzt, um in festgelegten Zeitintervallen (anstatt ausschließlich bei Änderungen) Daten zu schicken, können sich für die Zielsysteme Schwierigkeiten bei der Verknüpfung bzw. Synchronisation ergeben. Beispielsweise, wenn zwei Quellsysteme, deren Daten verknüpft werden sollen, zu unterschiedlichen Zeitpunkten die Daten senden. Dann wäre zu keinem Zeitpunkt sichergestellt, dass die verknüpften Daten aktuell sind, da sich bei beiden Quellsystemen währenddessen wieder Änderungen ergeben haben können. Würde das Zielsystem die Daten gleichzeitig von beiden Systemen abfragen (Pull), wäre immerhin zu diesem Zeitpunkt sichergestellt, dass die Daten aktuell sind.

3.2 | STORE, COMBINE

Grundsätzlich gibt es unterschiedlichste Speichertypen, die den unterschiedlichen Anforderungen gerecht werden. In einer Analytics-Architektur ist es häufig der Fall, dass zumindest für das Speichern und Kombinieren von Daten eine Technologie oder mehrere gemeinsame Technologien genutzt werden.

Also die Daten aus unterschiedlichen Quellen in das Zielsystem geladen werden, wo diese dann miteinander verknüpft werden.

Die Daten werden häufig entweder im **Enterprise Data Warehouse** (EDW) oder in dessen Gegenspieler, dem **Data Lake** gespeichert (s. Abb. 3).

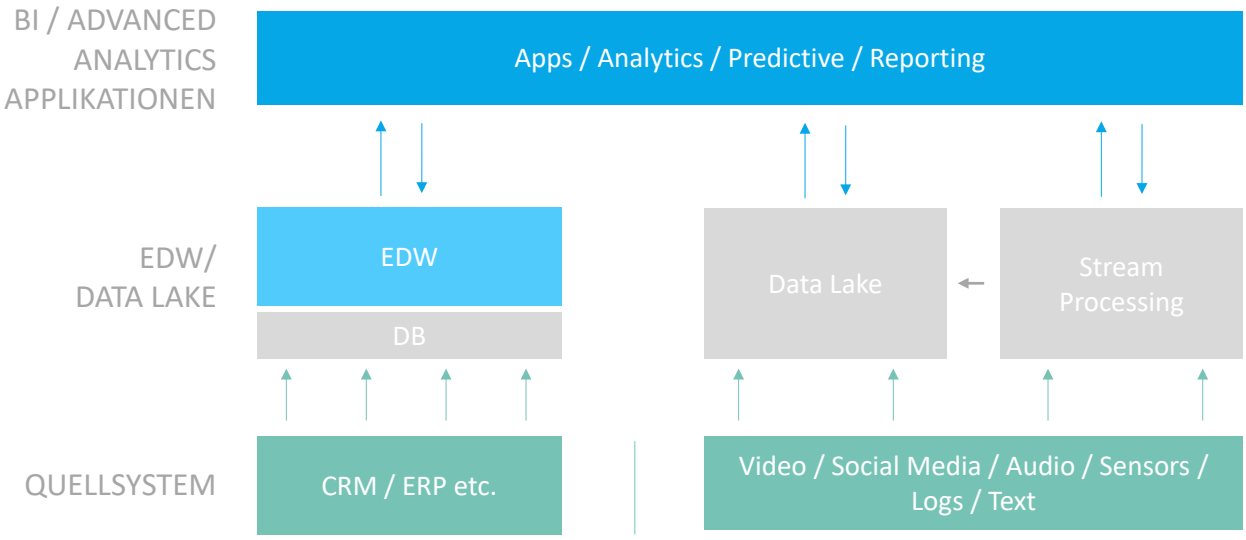

Abb. 3: High-Level Analytics-Architektur

EDW und Data Lake sind zwei unterschiedliche Konzepte, die beide ihre Berechtigung haben und insbesondere im Zusammenspiel einen Großteil der Anforderungen an eine **Analytics-Architektur** erfüllen können. Zur Übersichtlichkeit sind die Unterschiede in Tabelle 2 dargestellt.

	ENTERPRISE DATA WAREHOUSE	DATA LAKE
DATEN	Relationale Daten von operativen Datenbanken, Unternehmensanwendungen und transaktionalen Systemen	Nicht relationale und relationale Daten aus Social Media, Mobile Apps, IoT Endgeräte oder anderen Unternehmensanwendungen
SCHEMA	Vor der Implementierung festgelegt. Daten werden in vordefiniertes Schema geschrieben (schema-on-write)	Daten werden abgelegt und Schema wird später bei der Analyse festgelegt (schema-on-read)
PREIS/ LEISTUNG	Mehr Leistung durch hochwertigeren und teureren Speicher (bspw. In-Memory)	Weniger Leistung, jedoch mehr kostengünstigeren Speicher
DATENQUALITÄT	Single-Point-of-Truth – mit hohem Aufwand gepflegte Daten	Alle (Roh-)Daten
INTEGRATION	Hohe Integration mit anderen Systemen	Fokus auf Agilität, Daten werden ggfs. sporadisch aktualisiert
ZIELGRUPPE	Business Analyst	Data Scientist, Data Developers (und Business Analyst)
ANALYSEN	BI, Visualisierungen, Batch Reporting	Machine Learning, Data Discovery, Advanced Analytics

Tabelle 2: Vergleich EDW und Data Lake

Es gibt Alternativen bzw. Ergänzungen zum **EDW/Data Lake**. Beispielsweise kann eine Integrationsschicht bzw. Technologie genutzt werden, bei der die Daten im Quellsystem verbleiben und dann bei Bedarf Ad-Hoc geladen und verknüpft werden. Dies entspräche den Schritten Store und Combine. Bei größeren Datenmengen kann es hier schnell zu Performance-Engpässen kommen, insbesondere wenn diese miteinander in Verbindung gebracht werden müssen. Des Weiteren existieren Lösungen, die mehrere Aspekte gleichzeitig abdecken können. Hier wäre **Apache Kafka** ein Beispiel. Kafka ist auf der einen Seite ein **Broker**, der nach dem **Pub-Sub-Prinzip** arbeitetet. Gleichzeitig kann mit Kafka auch eine **Persistenz** hergestellt werden, mit der Subscriber im Daten-Stream zurückspringen können.

In einigen Fällen ist es bei Analysen gewünscht, gleichzeitig konstant neu eintreffende Daten als auch historisch gesammelte Daten zu berücksichtigen. Lange Zeit war hierfür die **Lambda-Architektur** das Mittel der Wahl. Bei dieser werden eingehende Daten aufgesplittet und sowohl in einem **„Stream Layer"** als auch einem **„Batch Layer"** persistiert und verarbeitet. Beide Schichten werden dann in einem **Serving Layer** kombiniert (s. Abb. 4).

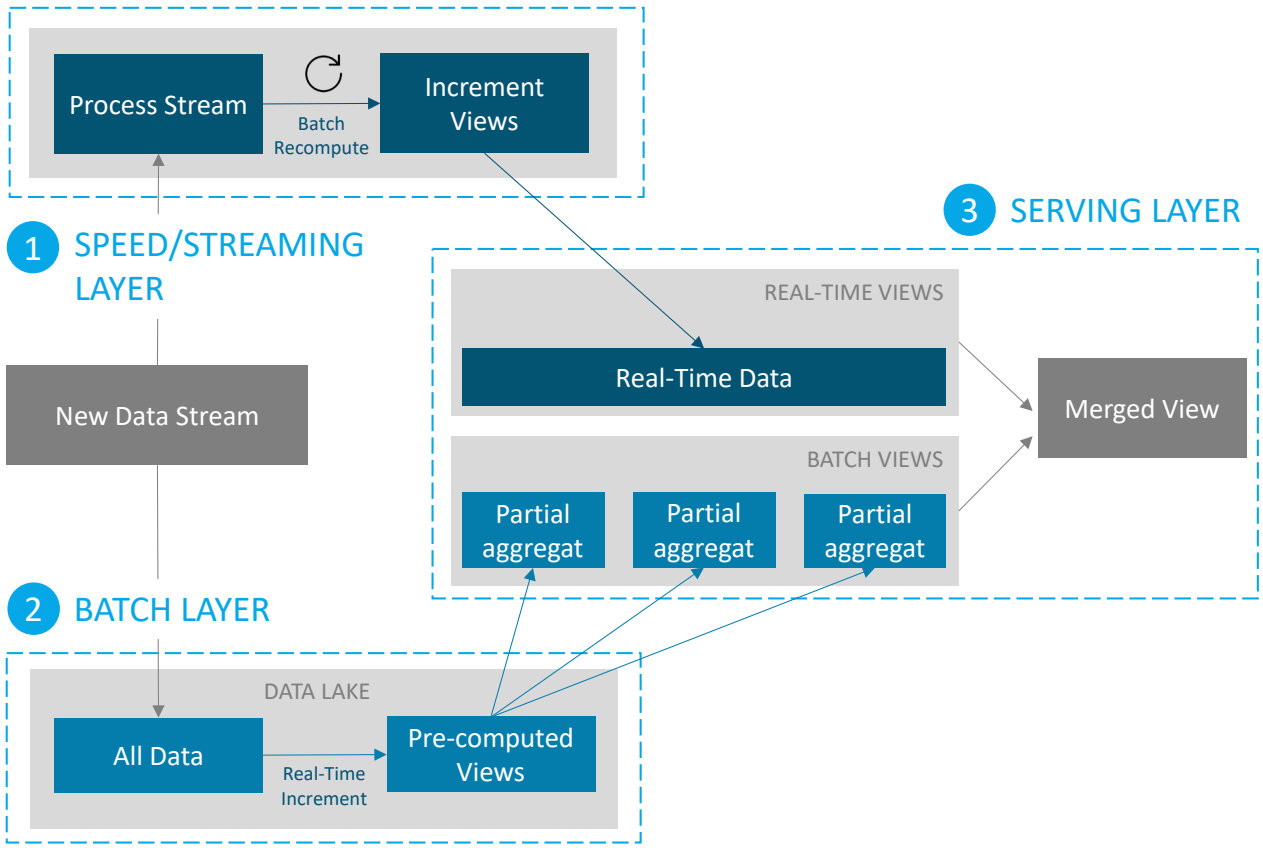

Abb. 4: Lambda-Architektur

Ein großer Nachteil dieses Vorgehens ist, dass die gleiche oder ähnliche Verarbeitung der Daten mit zwei unterschiedlichen Technologien umgesetzt werden muss. Mit Apache Kafka als **„Streaming Platform"** kann dies vereinfacht werden (s. Abb. 5).

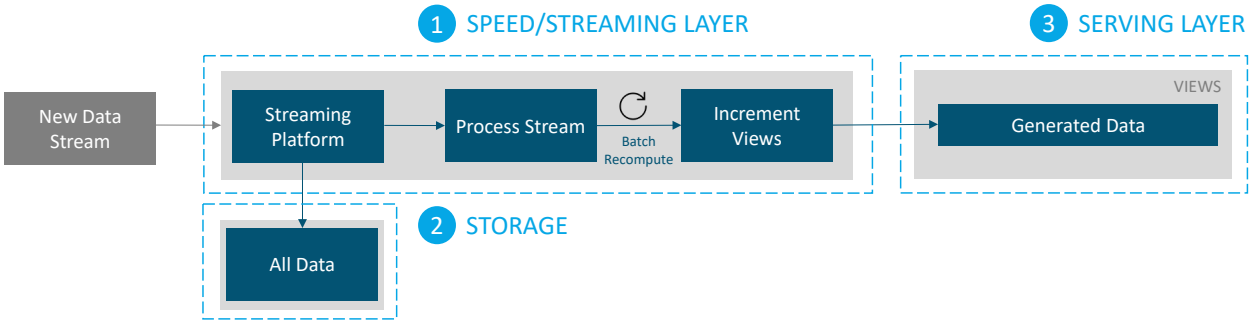

Abb. 5: Kappa Architektur

Bei der sogenannten **„Kappa Architektur"** wird der „Batch Layer" eingespart. Die Verarbeitung der Daten kann dementsprechend mit nur einer Technologie umgesetzt werden. Um auch historische Daten zu berücksichtigen, wird die Persistenz von Kafka genutzt. Bei Bedarf wird der Daten-Stream von Anfang an neu abgespielt, um auch diese zu berücksichtigen.

Der **Serving Layer** kann in beiden Fällen das bestehende EDW sein, um bestehende Pfade für das Reporting und Dashboarding nutzen zu können.

3.3 | ANALYZE

Generell gibt es zwei Möglichkeiten, um aus Daten Statistiken, KPIs oder Modelle herzuleiten – entweder toolgestützt mit einer **grafischen Oberfläche** in Kombination mit unterschiedlichen Operatoren oder durch eigenhändige Programmierung der Operationen und Berechnungen. Auch wenn Daten mit so gut wie jeder **Programmiersprache** verarbeitet und analysiert werden können, haben sich in den letzten Jahren besonders zwei Programmiersprachen etabliert. Diese sind die Statistiksprachen **R** und **Python**. Ein Hauptgrund hierfür ist die große wachsende Anzahl an Bibliotheken, die für beide Sprachen existieren. Durch die Verwendung dieser Programmiersprachen kann der Data Scientist oder Data Developer sich stärker auf die Analyse der Daten anstatt auf die Implementierung von Algorithmen konzentrieren. Zudem ist die Korrektheit der Algorithmen hier wahrscheinlicher als bei einer Eigenimplementierung.

R ist vornehmlich eine Programmiersprache für **statistische Berechnungen** und **Grafiken** und spielt in diesem Gebiet seine Stärken aus. **Python** hingegen wurde nicht mit Fokus auf die Datenanalyse entwickelt. Durch das Aufkommen mächtiger Bibliotheken für **Python** hat sich diese Programmiersprache mehr in Richtung Data Science entwickelt. Insbesondere für **Bilderkennung, Natural-Language-Processing** und **Deep Learning** existieren bekannte Python-Bibliotheken/Frameworks.

Wie bei jeder Individualprogrammierung ergibt sich hier das Problem der **Nachvollziehbarkeit und Überprüfbarkeit des Codes**. Aus diesem Grunde haben sich die Notebooks als eine bevorzugte Entwicklungsumgebung des Data Scientists herauskristallisiert. Für Python ist hier vor allem **Jupyter** zu nennen und für R, **R-Studio**. In einem Notebook werden Programm-Code, (Zwischen-)Ergebnisse, Freitext und Grafiken miteinander in einer „Story“ verknüpft. Bei einem entsprechend gut strukturierten Notebook lässt sich somit schnell nachvollziehen, was gemacht wurde. Jupyter und R-Studio können mit entsprechenden Paketen für die jeweils andere Programmiersprache genutzt werden. Der R- bzw. Python-Code wird auf dem Client oder Server, auf welchem auch die Notebook-Anwendung installiert ist (nachfolgend Workbench genannt), ausgeführt. Größere Datenmengen können bei der Modellgenerierung schnell zu Ressourcenengpässen führen. Es gibt mehrere Strategien, um dies zu umgehen. Auf dem Client/Server können die Daten auf einer kleineren Teilmenge und erst, wenn die gewünschte Qualität erreicht ist, mit höheren Ressourcen auf allen Daten berechnet werden. Alternativ kann beispielsweise über eine Principle-Component-Analyse (PCA) die Anzahl der Dimensionen und somit das Volumen der Daten reduziert werden. Anstatt die Daten auf die Workbench zu laden, können alternativ mit entsprechenden Technologien auch die Berechnungen ausgelagert werden. In den meisten Fällen werden dabei die Berechnungen aufgeteilt und zu den Daten (auf der Datenbank bzw. dem Cluster) gebracht.

Für spezielle Anforderungen existieren auch weniger verbreitete Programmiersprachen. Hier wären für Performance und Typsicherheit **Julia** oder **Scala** im Zusammenhang mit **SPARK** als Analyseplattform für Big Data und Streaming Daten zu nennen.

3.4 | APPLY

Die Ergebnistypen eines Algorithmus können stark variieren und umfassen statistische Kennzahlen, Wahrscheinlichkeiten, vorhergesagte Werte und Klassifikationen. Entsprechend breit ist auch das mögliche Nutzungspotential dieser Ergebnisse. Es kann sein, dass einmalig Statistiken erhoben werden, um entsprechende **Maßnahmen abzuleiten** oder **Erkenntnisse zu gewinnen**. Es ist auch möglich, dass diese Ergebnisse in **Dashboards** dargestellt werden und regelmäßig zur **Unternehmenssteuerung** genutzt werden. Teilweise erhält der Nutzer das Ergebnis nur indirekt. Insbesondere bei Scoring-Modellen kann es sein, dass beispielsweise ein Kredit automatisch abgelehnt wurde, weil der Score-Wert einen festgelegten Schwellenwert unterschritten hat.

Wurden statistische Modelle genutzt, muss neben der eigentlichen **Implementierung** des Modells die **Überwachung** und das **Nachtrainieren** umgesetzt werden.

Bei der **Implementierung** von eigenem Python-/R-Code muss entschieden werden, wie dieser getriggert wird, wo und wie sich das Script die Daten holt und wie die Ergebnisse dann weiterverarbeitet werden. Die gewählte Strategie ist auch abhängig von den Designentscheidungen, die in den vorherigen Schritten gemacht wurden. Wird ein Push-Mechanismus bei der Datenerfassung genutzt, liegt es nahe, dass das Script neue Daten abonniert (per Subscription s. Collect) und die Berechnungen sofort durchführt. Ist in einem Unternehmen eine Microservices-Architektur etabliert, kann es sinnvoll sein, das Python-/R-Script auch als ein Microservice anzubieten. Dieser kann dann beispielsweise über eine REST-Schnittstelle angesprochen werden. Müssen sehr große Datenmengen selten analysiert werden, kommt eine zeitgesteuerte Batch-Verarbeitung in Frage, bei der sich das Script zur Laufzeit die Daten holt (Push).

Auch die Nutzung der Ergebnisse kann stark variieren. Bei unregelmäßigen, einmaligen Analysen werden die Ergebnisse wahrscheinlich mithilfe des Notebooks des Data Scientists direkt mit dem Business User geteilt. Bei einer aufwendigen Integration in die Prozesse stimmt das Kosten-Nutzen Verhältnis in diesem Fall nicht. Werden die Ergebnisse hingegen schnell von den weiteren Geschäftsprozessen benötigt, beispielsweise bei der Prüfung von Kreditvergaben, kann eine tiefgehende Integration den gesamten Kreditvergabeprozess beschleunigen und langfristig Kosten einsparen.

Insbesondere bei einer tiefgreifenden Integration ist die **Überwachung** der Analysen und deren Ergebnisqualität umso wichtiger. Veränderungen in den Eingangsdaten oder bislang unbekannte Fälle können schnell dazu führen, dass ein Modell nicht im ursprünglich geplanten Sinne funktioniert. Zudem werden Analysen/Modelle nie zum Selbstzweck eingesetzt, sondern dienen in der Regel einem höheren unternehmerischen Ziel. Um bei dem Beispiel der Kreditvergabe zu bleiben, kann ein Scoring-Modell, welches immer mehr Kreditanfragen ablehnt, zu wirtschaftlichen Verlusten führen. Aus diesem Grund ist es wichtig, die Modellqualität an sich als auch den Zweck des Modells regelmäßig zu prüfen. Für die Modellqualität gibt es einige mathematische Kennzahlen, die erhoben werden können. Einen Ausschnitt dieser Kennzahlen bilden z.B. P-Wert, R^2, F1-Score, Silhouettenwert und die Standardabweichung.

Damit der zuvor beschriebene Effekt, die Verschlechterung von Modellen, nicht eintritt, muss neben der Implementierung des Modells auch das **Nachtrainieren** dieses eingeplant werden. Das (Nach-)Lernen von Modellen verbraucht deutlich mehr Ressourcen als das reine Anwenden. Daher wird beides in der Praxis oft technisch getrennt. In der Lambda-Architektur kann es beispielsweise sein, dass das Modell im Speed Layer auf einen Daten-Stream angewandt wird und im Batch Layer regelmäßig auf dem Data Lake (nach-)trainiert wird.

4 | ZUSAMMENFASSUNG

Zusammenfassend lässt sich sagen, dass nicht eine perfekte Architektur für jedes Unternehmen existiert. In fast jedem Bereich bringen Designentscheidungen Vor- und Nachteile mit sich. Diese lassen sich nur mit den konkreten Anforderungen an die Use-Cases bzw. im Allgemeineren an das Unternehmen für den Kontext bewerten. Sollten Anforderungen noch nicht klar sein, kann ein Prototyp oder ein schneller technischer Durchstich helfen. Dieser erlaubt es, insbesondere weniger technisch affinen Stakeholdern, sich einen Überblick zu verschaffen. Solch eine „schnelle Lösung“ ist kein Ersatz, sondern eher eine Hilfestellung, um fundiert und vorrausschauend die Architektur für zukünftige Advanced Analytics Use-Cases zu formen.

Die häufige Frage in dem Kontext ist: “How to start?”. Unser erprobtes Verfahren aus moderierten 1-Tages-Workshops zur Bestandsaufnahme und zum Ausbau von Advanced Analytics-Themen (z.B. Aufbau einer Big Data Architektur) hat sich in der Praxis bewährt. Bestandteile des Workshops können unter anderem die Reifegradbestimmung eines Unternehmens bzgl. Advanced Analytics, die Vorstellung der Theorie von Advanced Analytics, die Bestimmung von relevanten Use-Cases für ein Unternehmen oder die Vorstellung von Live-Demos sein. Im Nachgang kann auf Basis dieses Überblicks die verschiedenen Stoßrichtungen der weiteren Maßnahmen geplant werden bzw. die Bausteine der „Daten-Pipeline“ durch die sukzessive Umsetzung von weiteren Business Cases optimiert bzw. erstellt werden. Weitere Informationen zum “Advanced Analytics Orientierungsworkshop” finden Sie hier.

5 | LITERATURVERZEICHNIS

Tiedemann. Advanced Analytics in Theorie und Praxis. 31. Januar 2018. https://www.alexanderthamm.com/de/blog/advanced-analytics-theorie-und-praxis/ (Zugriff am 20. Juli 2020).

ÜBER ISR

Im Jahr 1993 gegründet, avancierte die ISR Information Products AG als Beratungsunternehmen mit Standorten in Braunschweig, Münster, Hamburg, Köln, München und Frankfurt zum Experten für Analytics, Prozess-Digitalisierung und Application Management.

Mit einem umfassenden Blick auf die Bedürfnisse namhafter Kunden konzipieren, modernisieren, implementieren und betreuen wir IT-Architekturen, Software-Lösungen und IT-Infrastrukturen.

Unsere ca. 200 Mitarbeiter schaffen dabei neue Lösungen im Umfeld SAP HANA, Big Data, Data Management, Cloud Computing und Künstliche Intelligenz (KI) für die Herausforderungen der digitalen Welt.

Unser Ziel:

Ihnen die wirtschaftliche Nutzung von Daten zu ermöglichen. Gezielte Analysen, strategisches Vorgehen, saubere Projektsteuerung und ganzheitliche IT-Lösungen sind für uns dabei die Basis, um Sie mithilfe verschiedener Digitalisierungsmaßnahmen fit zu machen für morgen.

ISR
for your digital smile

KONTAKTIEREN SIE UNS!

Ihr Ansprechpartner

Kay Rohweder

Senior Manager | SAP Information Management
kay.rohweder@isr.de

ISR INFORMATION PRODUCTS AG

Hintern Brüdern 23
38100 Braunschweig
Tel. +49 (0) 531 12 08-0
hello@isr.de

Vorstand: Bernd Rosemeyer (Sprecher) | Manfred Merßmann
Aufsichtsratsvorsitzender: Peter Kallien
Sitz der Gesellschaft: Braunschweig