Duale Hochschule Baden-Württemberg Mannheim

Bachelorarbeit

Kontextbasierte Vorschläge von verkaufsfördernden Inhalten in der Enterprise-Softwareindustrie

Studiengang Wirtschaftsinformatik

Studienrichtung Software Engineering

Verfasserin:	Anna-Lena Blinken
Matrikelnummer:	1078470
Firma:	SAP SE
Abteilung:	Strategic Value Acceleration
Kurs:	WWI18SEB
Studiengangsleiter:	Prof. Dr. Thomas Holey
Wissenschaftlicher Betreuer:	Prof. Dr. Sebastian Ritterbusch
	sebastian.ritterbusch@dhbw-mannheim.de
Firmenbetreuer:	Dominic Pastoors
	dominic.pastoors@sap.com
Bearbeitungszeitraum:	15.02.2021 – 10.05.2021

Geschlechtsneutrale Formulierung

In der vorliegenden Projektarbeit wird auf die parallele Verwendung männlicher sowie weiblicher Sprachformen verzichtet, um eine bessere Lesbarkeit zu gewährleisten. Es wird die maskuline Form verwendet, wobei beide Geschlechter gleichermaßen zu verstehen sind.

DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

Wirtschaftsinformatik

Kontextbasierte Vorschläge von verkaufsfördernden Inhalten in der Enterprise-Softwareindustrie

Bachelorarbeit
Anna-Lena Blinken
SAP SE

Kurzfassung

Empfehlungssysteme erkennen die Präferenzen und Interessen eines Benutzers, mit dem Ziel, auf Basis dieser Informationen dem Benutzer zu ihm passende Inhalte vorzuschlagen. Durch die personalisierten Empfehlungen soll der Benutzer zu einer Aktion, wie beispielsweise dem Kauf eines Produktes, motiviert werden. Der Entscheidungsprozess eines Benutzers, eine gewisse Aktion durchzuführen, basiert nicht nur auf seinen Interessen sowie Präferenzen, sondern wird maßgeblich durch sein Umfeld beeinflusst. Dieses Umfeld, der sogenannte Kontext, kann zum Beispiel die Zeit, die Stimmung oder gewisse Eigenschaften eines Benutzers umfassen.

Die vorliegende Arbeit analysiert, ob ein Empfehlungssystem unter Hinzunahme von Kontextdaten präzisere Empfehlungen liefert. Dabei liegt der Fokus auf Empfehlungssystemen der Enterprise-Softwareindustrie. Zur Evaluierung dieser Fragestellung wird ein kontextbasiertes Empfehlungssystem beispielhaft für das Unternehmen SAP SE entwickelt, welches einem Kunden die zu ihm passenden Produkte vorschlägt. Um die Ergebnisse des kontextbasierten Empfehlungssystems einordnen zu können, wird ein zweites Empfehlungssystem erstellt, welches ohne die Hinzunahme von Kontextdaten Produktempfehlungen liefert. Anhand ausgewählter Metriken werden die beiden Empfehlungssysteme miteinander verglichen.

Als Resultat der Evaluation der Ergebnisse stellt sich heraus, dass das kontextbasierte Empfehlungssystem präzisere Empfehlungen liefert. Folglich kann durch die Hinzunahme von weiterführenden Daten das Kundenprofil konkreter abgebildet werden. Durch das präzisere Abbilden der Interessen eines Kunden soll der Kunde zu der Aktion verleitet werden, ein Produkt zu kaufen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AUROC	Area Under the Receiver Operating Characteristics
B2C	Business-to-Consumer
CRISP-DM	Cross Industry Standard Process for Data Mining
CRM	Customer-Relationship-Management
DSR	Design Science Research
EAS	Enterprise Application Software
ERM	Entity-Relationship-Modell
ERP	Enterprise-Resource-Planning
F	falsch
FN	falsch negativ
FP	falsch positiv
FPR	Falsch-positiv-Rate
IT	Informationstechnik
KI	künstliche Intelligenz
ML	maschinelles Lernen
NIST	National Institute of Standards and Technology
R	richtig
RN	richtig negativ
RNR	Richtig-negativ-Rate
ROC	Receiver Operating Characteristic
RP	richtig positiv
RPR	Richtig-positiv-Rate
SCM	Supply-Chain-Management
TV	Television

1 Einleitung

1.1 Motivation

Empfehlungssysteme generieren durch die Analyse von Daten Informationen über die Präferenzen sowie Interessen eines Benutzers. Auf Basis dieser Informationen werden den Benutzern personalisierte Inhalte vorgeschlagen.[1] Durch das konkrete Abbilden der Interessen in Form der personalisierten Inhalte soll der Benutzer angesprochen und zu einer Aktion verleitet werden. Diese Aktion kann unter anderem den Kauf eines Produktes, das Anschauen eines Filmes oder das Anklicken eines Artikels darstellen. Die für die Empfehlungen notwendigen Interessen sowie Präferenzen der Benutzer werden in vielen Fällen von Kaufhistorien oder Benutzerbewertungen abgeleitet. Neben diesen direkten Informationen über einen Benutzer wird der Entscheidungsprozess, eine Aktion durchzuführen, durch die Umgebung des Benutzers beeinflusst. Das Umfeld eines Benutzers kann durch weiterführende Daten, dem sogenannten Kontext, erfasst werden.[2] Es stellt sich die Frage, ob durch Hinzunahme von weiterführenden Informationen die Benutzerinteressen besser abgebildet werden können, wodurch die Empfehlungssysteme präzisere Vorschläge generieren würden. In dem Endkonsumenten-Bereich finden Kontextdaten bei dem Erstellen von Empfehlungen bereits Anwendung. Beispielsweise in der Filmindustrie werden Kontextdaten verwendet, um dem Benutzer abhängig von seinen Interessen, der Zeit, seiner Stimmung und seinen sozialen Beziehungen zu anderen Nutzern Filme vorzuschlagen.[3]

Während Kontextdaten in Verbindung mit Empfehlungssystemen im Endkonsumenten-Bereich miteinbezogen werden, werden diese in der Enterprise-Softwareindustrie weniger eingesetzt. Ein beispielhaftes Empfehlungssystem der Enterprise-Softwareindustrie schlägt dem Kunden Softwareprodukte vor, die er in Zukunft kaufen könnte. Anwendung würde ein Empfehlungssystem dieser Art zum Beispiel auf der Internetseite des Softwareanbieters finden. Es gilt zu erforschen, ob im Bereich der Enterprise-Software-

[1] Vgl. Dey 2001, S. 5.
[2] Vgl. Adomavicius, Sankaranarayanan et al. 2005, S. 2.
[3] Vgl. Alan, Shlomo und Ernesto W. De 2013, S. 1 f.

industrie Kontextdaten helfen können, Empfehlungen präziser zu gestalten und dadurch verkäufsfördernde Aktionen beim Kunden hervorzurufen.

1.2 Problemstellung und Abgrenzung

Die Problemstellung bei der zu erforschenden Frage, ob Kontextdaten in der Enterprise-Softwareindustrie zu präziseren Vorschlägen verhelfen können, stellt die Definition der Kontextdaten dar. Während in dem Endkonsumenten-Bereich Kontextdaten über soziale Medien und das Auftreten des Benutzers im Internet gesammelt werden, steht die Enterprise-Softwareindustrie vor einer Herausforderung. Über die in der Enterprise-Softwareindustrie zur Verfügung stehenden Daten, wie zum Beispiel die Kaufhistorie, können Präferenzen sowie der Kontext eines Kunden approximiert werden.[4] Das Problem der Definition von Kontext in der Enterprise-Softwareindustrie gilt es, in der vorliegenden Arbeit zu betrachten. Zur Evaluation dieser Problemstellung erfolgt eine konkrete Umsetzung eines kontextbasierten Empfehlungssystems eingebettet in die Enterprise-Softwareindustrie, welches unter Hinzunahme des zuvor definierten Kontextes Produktvorschläge liefert.

1.3 Vorgehensweise

Zur Evaluation des potenziellen Einflusses kontextbasierter Empfehlungssysteme auf die Enterprise-Softwareindustrie wird die Vorgehensweise der vorliegenden Arbeit in der Darstellung 1.1 definiert. Auf Basis des Design Science Research (DSR) Ansatzes (Abschnitt 3.1) wird ein kontextbasiertes Empfehlungssystem, ein sogenanntes Artefakt, welches in der Enterprise-Softwareindustrie eingebettet ist, entwickelt. Bevor das Artefakt erstellt werden kann, werden die grundlegenden Begriffe, die für das Verständnis der Arbeit und für das Erstellen des kontextbasierten Empfehlungssystems von Relevanz sind, in dem Kapitel 2 eingeführt. Zu diesen Begriffen zählt unter anderem die Einordnung sowie Definition des überwachten maschinellen Lernens und des untergeordneten Random Forest Algorithmus. Das zu erstellende kontextbasierte

[4] Vgl. Hu, Koren und Volinsky 2008, S. 264.

Empfehlungssystem wird den Random Forest verwenden, um Vorschläge zu generieren. Darüber hinaus werden der Begriff Kontext sowie die Charakteristika der Enterprise-Softwareindustrie aufgeführt, um im Praxisteil der Arbeit einen Kontext für ein Unternehmen der Enterprise-Softwareindustrie definieren zu können. Abschließend wird sich im Theorieteil mit der konkreten Betrachtung eines kontextbasierten Empfehlungssystems befasst, mit der Intention, dieses im weiteren Verlauf praktisch umzusetzen.

Anhand des Cross Industry Standard Process for Data Mining (CRISP-DM), welches in Abschnitt 3.2 erläutert wird, wird im Kapitel 4 ein kontextbasiertes Empfehlungssystem beispielhaft in dem Unternehmen SAP SE eingeführt. Das Softwareunternehmen SAP SE ist unter anderem in den Bereichen Datenbanken, Analysen, intelligente Technologien und Experience Management vertreten.[5] Auf Basis von Kundendaten, einer Kundenkaufhistorie und eines zu definierenden Kontextes soll das Empfehlungssystem einem Kunden Produkte vorschlagen. Zur Evaluation des erstellten, kontextbasierten Empfehlungssystems wird ein zweites Empfehlungssystem aufgebaut, welches ohne die Hinzunahme von Kontext Produktempfehlungen ausspricht. Anhand ausgewählter Metriken zur Auswertung von Klassifikationsmodellen werden die beiden Empfehlungssysteme verglichen, mit dem Ziel, den potenziellen Einfluss kontextbasierter Empfehlungssysteme auf die Enterprise-Softwareindustrie quantifizieren zu können.

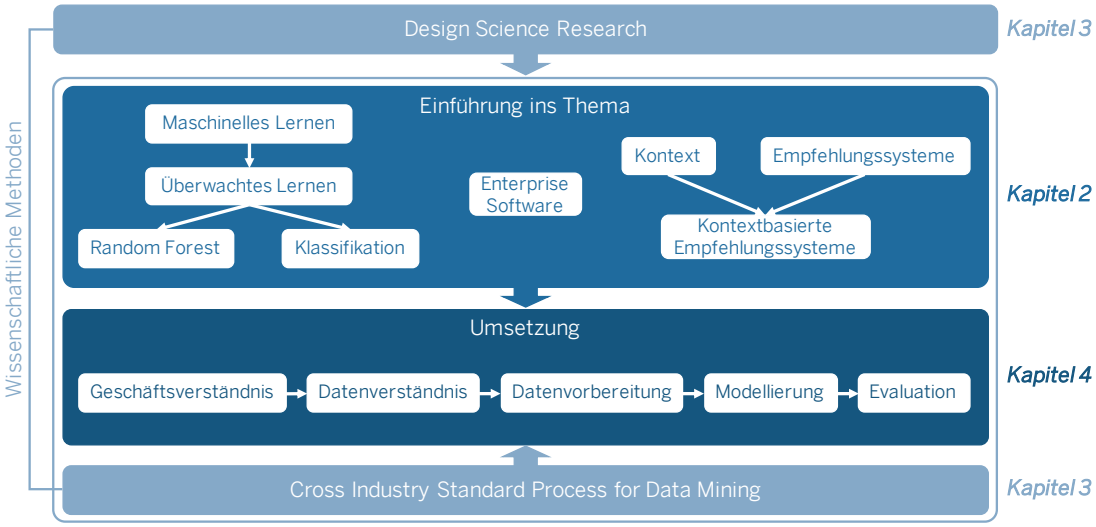


Abbildung 1.1: Aufbau der Arbeit (eigene Darstellung)

[5] Vgl. SAP SE 2021.

2 Einführung ins Thema

2.1 Überwachtes maschinelles Lernen

Bevor sich der Definition des Begriffs **überwachtes maschinelles Lernen** gewidmet wird, bedarf es einer Definition des übergeordneten Begriffes **maschinelles Lernen (ML)** (engl. machine learning).

Definition des maschinellen Lernens

Das ML stellt einen Bereich der künstlichen Intelligenz (KI) dar. Im Jahre 1955 wurde der Begriff der KI erstmals im Rahmen der Dartmouth Konferenz von Minsky, McCarthy, Newell und Simon als Maschinen, die Verhalten aufwiesen, welches eine Form der menschlichen Intelligenz widerspiegele, definiert.[6] Nachdem in den 1950er Jahren der Begriff der KI aufkam, etablierte sich in den 1980er das ML. Das ML ist der Prozess des praktischen Lösens eines Problems durch das Sammeln von Daten und durch das algorithmische Bilden eines statistischen Modells auf Basis der gesammelten Datensätze. Das erstellte statistische Modell wird verwendet, um das vorliegende praktische Problem zu lösen.[7] Die konkrete Fragestellung, die beim ML betrachtet wird, ist: Wie kann ein Computerprogramm mithilfe von Erfahrungen lernen und somit seine Aufgaben optimiert durchführen?[8] Im Vergleich zu klassischen Programmen, bei denen ein statischer Code durch die Übergabe von Daten ein Ergebnis liefert, wird beim ML ein Algorithmus angewandt, der stetig angepasst wird (Abbildung 2.1). Der Algorithmus nimmt Daten entgegen, überprüft verschiedene Entscheidungsregeln und liefert demnach ein Ergebnis. Der Algorithmus sowie die Entscheidungsregeln ändern sich abhängig von dem Erlernten bzw. dem Ergebnis. Somit passt sich das ML an neue Umgebungen an und entdeckt sowie extrapoliert Muster.[9]

[6] Vgl. Welsch, Eitle und Buxmann 2018, S. 370; Vgl. McCarthy et al. 1955, S. 2 f.
[7] Vgl. Burkov 2019, Kapitel 1, S. 3.
[8] Vgl. Wittpahl 2019, S. 25.
[9] Vgl. Russell et al. 2016, S. 2.

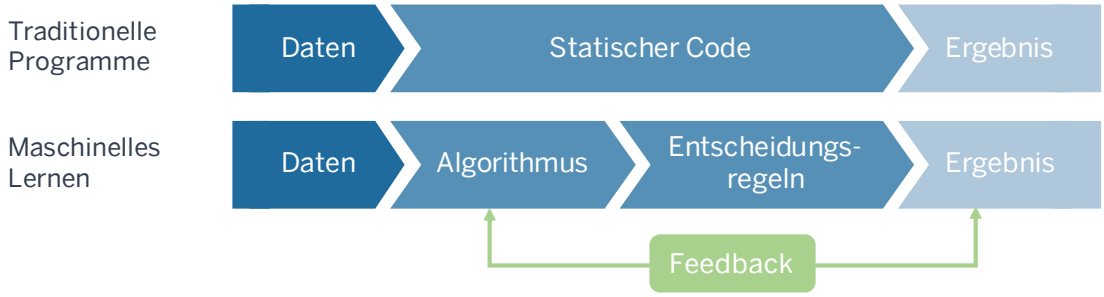


Abbildung 2.1: Traditionelle Programme im Vergleich zum ML angelehnt an Wittpahl 2019, S. 25

Das ML unterteilt sich in drei Kategorien von Lernalgorithmen: *Überwachtes maschinelles Lernen* (engl. supervised machine learning), *unüberwachtes maschinelles Lernen* (engl. unsupervised machine learning) und *verstärktes maschinelles Lernen* (engl. reinforcement learning).[10] Alle drei Lernalgorithmen konstruieren die mathematische Funktion $f : X \to Y$. Unterschiede zwischen den Lernalgorithmen finden sich bei den Mengen X sowie Y und der Beschaffenheit der Daten, um die mathematische Funktion zu erstellen.[11] Aufgrund der Tatsache, dass der Fokus der vorliegenden Arbeit auf dem überwachten maschinellen Lernen liegt, werden an dieser Stelle das unüberwachte sowie verstärkte maschinelle Lernen kurz zur Differenzierung definiert. Das unüberwachte maschinelle Lernen trifft aufgrund von Eingabewerten, denen keine präzisen Ausgabewerte gegenüberstehen, Vorhersagen. Es wird von einem sogenannten unmarkierten Datensatz gesprochen.[12] Auf Basis der Eingabedaten des unmarkierten Datensatzes erkennt das Programm Muster und Strukturen in den Eingabedaten, die in weiterverwendbare Informationen verwandelt werden. Das verstärkte maschinelle Lernen arbeitet mit Formen der Belohnung und Bestrafung. Für falsche Ergebnisse erhält das Programm eine Bestrafung, währenddessen es bei richtigen Ergebnissen belohnt wird. Das Programm speichert sein Verhalten sowie die verursachten Reaktionen und versucht, die Anzahl der Belohnungen zu maximieren.[13]

[10] Vgl. Wittpahl 2019, S. 24.
[11] Vgl. Frochte 2019, S. 20.
[12] Vgl. Burkov 2019, Kapitel 1, S. 4.
[13] Vgl. Welsch, Eitle und Buxmann 2018, S. 371; Vgl. Wittpahl 2019, S. 26 ff.

Definition des überwachten maschinellen Lernens

Das überwachte maschinelle Lernen ist, wie in Abbildung 2.3 dargestellt, ein Bestandteil des ML und verfolgt das Ziel, anhand von gegebenen Eingabewerten den unbekannten Ausgabewert vorherzusagen.[14] Um die Ausgabewerte prognostizieren zu können, wird ein Datensatz von Ein- und den zugehörigen Ausgabewerten, ein sogenannter „gelabelte[r] bzw. markierte[r] "[15] Datensatz der Form $\{(x_i, y_i)\}_{i=1}^{N}$, benötigt. Jedes Element x_i von N wird als Merkmalsvektor (engl. feature vector) bezeichnet. Die Dimensionen des Vektors enthalten Werte bzw. Merkmale, die den Datensatz beschreiben. Das überwachte maschinelle Lernen intendiert, unter Verwendung eines markierten Datensatzes, ein Modell zu erstellen, das einen Merkmalsvektor als Eingabe verwendet und als Ausgabe eine Vorhersage liefert.[16] Der Algorithmus erkennt und merkt sich die Zusammenhänge zwischen den Ein- sowie Ausgabewerten, den markierten Datensätzen, und erlernt dadurch eine Funktion $f : X \to Y$.

Das Modell wird mit einem markierten Datensatz, dem sogenannten Trainingsdatensatz, trainiert. Wird der Algorithmus im Anschluss an das Training angewandt, sagt dieser auf Grundlage von unmarkierten Eingabewerten und seinen erlernten Erfahrungen Ausgabewerte vorher. Um das Modell zu validieren, wird ein markierter Testdatensatz eingesetzt. Dem Modell wird als Eingabe der Merkmalsvektor übergeben, anhand dessen der Output prognostiziert wird. Werden die vorhergesagten Werte des Algorithmus mit den tatsächlichen Ausgabewerten verglichen, kann die Güte des Modells anhand verschiedener Metriken bestimmt werden.[17] Im letzten Schritt wird mit einem markierten Validierungsdatensatz überprüft, ob das überwachte Modell einer Überanpassung (engl. over-fitting) unterliegt. Ein überangepasstes Modell ist optimal an den verwendeten Trainings- und Testdatensatz angepasst, führt aber bei der Anwendung eines unbekannten Datensatzes zu mangelhaften Ergebnissen. Zur Unterbindung dieses Phänomens wird ein Validierungsdatensatz hinzugezogen, mithilfe dessen saisonale Besonderheiten aufgedeckt werden können. Dies hat zur Folge, dass die Genauigkeit des Modells gesteigert werden kann. Anhand des Merkmalsvektors des markierten Validierungsdatensatzes generiert das überwachte Modell Empfehlungen, die mit den

[14] Vgl. Welsch, Eitle und Buxmann 2018, S. 371.
[15] Frochte 2019, S. 20.
[16] Vgl. Burkov 2019, Kapitel 1, S. 3.
[17] Vgl. Welsch, Eitle und Buxmann 2018, S. 371.

tatsächlichen Zielwerten des markierten Validierungsdatensatzes zur Evaluation der Performanz des Modells abgeglichen werden.[18] In Abbildung 2.2 wird der Aufbau eines überwachten Modells anhand eines Trainings-, Test- sowie Validierungsdatensatzes visuell dargestellt.

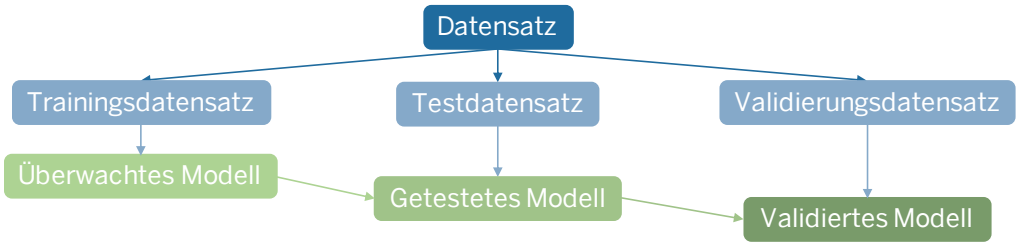


Abbildung 2.2: Ablaufdiagramm zum Aufbauen eines überwachten maschinellen Modells mithilfe eines Trainings-, Test- und Validierungsdatensatzes angelehnt an Vgl. Xu und Goodacre 2018, S. 250

Das überwachte maschinelle Lernen behandelt zwei Problemstellungen: die Klassifikation und die Regression. Im Folgenden wird sich auf die Klassifikation fokussiert. Lediglich diese ist für den weiteren Verlauf von Relevanz. Abbildung 2.3 ordnet das überwachte maschnielle Lernen in die übergeordneten Begriffe ein.

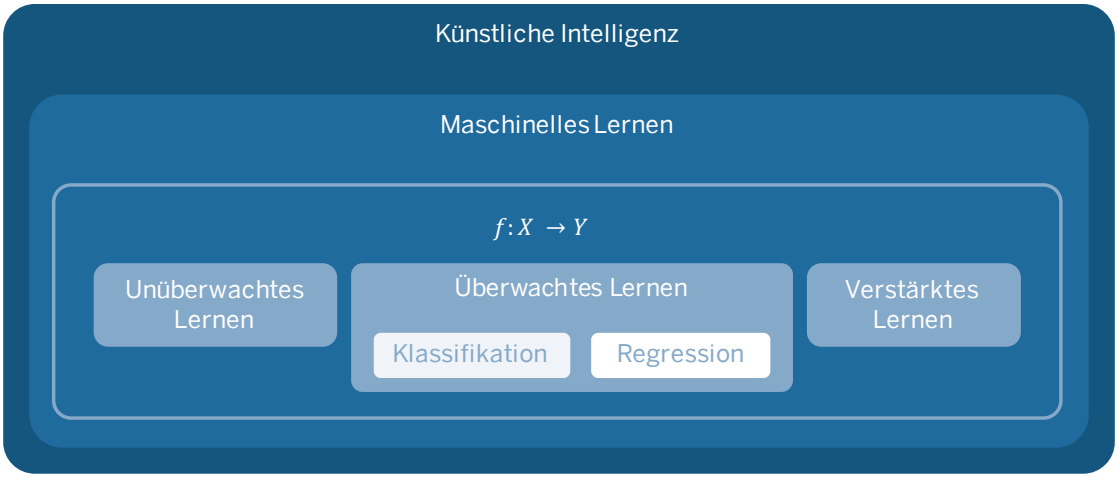


Abbildung 2.3: Einordnung des überwachten maschinellen Lernens angelehnt an Frochte 2019, S. 20 ff.

[18] Vgl. Xu und Goodacre 2018, S. 249 f.

Klassifikation

Die Zielvariable bzw. die Vorhersage Y ist bei der Klassifikation diskret.[19] Eine diskrete Variable nimmt im Vergleich zu einem stetigen Merkmal nur bestimmte Werte an. Diskret und stetig können der übergeordneten *metrischen* Skalierung zugeordnet werden. Zwischen metrischen Merkmalen kann der Abstand gemessen und eine Reihenfolge gebildet werden.[20] Ein Klassifikationsproblem wird von einem Algorithmus gelöst, der eine Reihe von markierten Datensätzen als Eingabe verwendet und ein Modell erstellt, welches einen unmarkierten Datensatz als Eingabe entgegennimmt und diesen Datensatz einer Klasse zuordnet.[21] Es existiert eine Zuordnung $c : X \to C$, die einen Merkmalsvektor X eindeutig einer Klasse C zuordnet.[22] Ein klassisches Beispiel für eine Klassifikation ist das Einstufen einer Mail als „*Spam-Mail*“ oder als „*nicht Spam-Mail*“. Bei der Klassifikation gibt es eine endliche Menge von Klassen. Liegen wie in dem erläuterten Beispiel genau zwei Klassen vor, wird von einer *binären Klassifikation* gesprochen. Hingegen ist die *multidimensionale Klassifikation* ein Klassifikationsproblem mit mindestens drei Klassen. Die Ausgabe eines Klassifikationsmodells kann diskret oder kontinuierlich sein. Bei einer diskreten Ausgabe wird eine Klasse vorhergesagt, die dem Datensatz zugeordnet wird. Eine kontinuierliche Ausgabe demgegenüber gibt Wahrscheinlichkeiten für die Zugehörigkeit des Datensatzes zu den jeweiligen Klassen aus.[23]

Metriken zur Evaluation eines Klassifikationsmodells

Ein Klassifikationsmodell liefert in der Testphase Rohdaten über korrekte sowie inkorrekte Vorhersagen für die jeweiligen Klassen. Diese Informationen werden in einer sogenannten *Konfusionsmatrix* (engl. confusion matrix), welche die Unterschiede zwischen den vorhergesagten und tatsächlichen Werten widerspiegelt, dargestellt. Auf Grundlage dieser Konfusionsmatrix können Metriken berechnet werden, die die Güte und die Genauigkeit eines Klassifikationsmodells evaluieren.[24] Die beispielhafte 2×2 Konfusi-

[19] Vgl. Frochte 2019, S. 21.
[20] Vgl. Bleymüller, Weißbach und Dörre 2020, S. 5 f.
[21] Vgl. Burkov 2019, Kapitel 1, S. 12.
[22] Vgl. Frochte 2019, S. 22.
[23] Vgl. Tharwat 2020, S. 170.
[24] Vgl. Bradley 1997, S. 1145.

onsmatrix für binäre Klassifikationsprobleme, welche in Abbildung 2.4 dargestellt wird und von den zwei Klassen *richtig (R)* und *falsch (F)* ausgeht, hat vier mögliche Ergebnisse. Es wird zwischen richtigen R und falschen Vorhersagen F unterschieden. Wenn

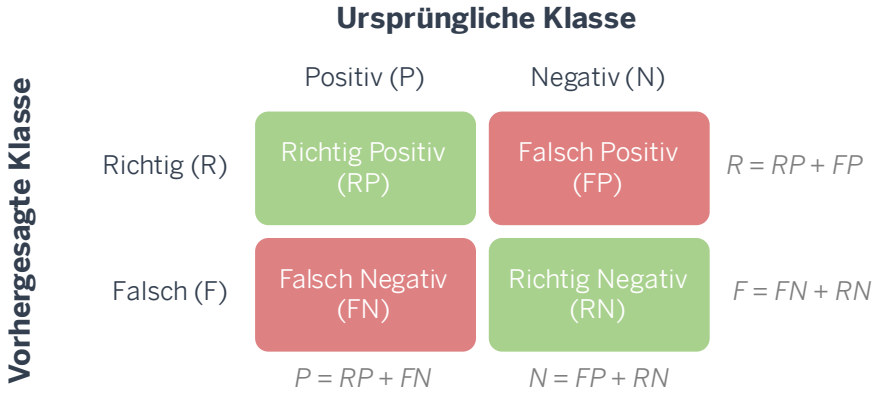


Abbildung 2.4: 2×2 Konfusionsmatrix angelehnt an Tharwat 2020, S. 170

ein als richtig klassifizierter Datensatz als richtig eingestuft wird, wird die Vorhersage mit *richtig positiv (RP)* umschrieben. Wird ein negativer Datensatz als negativ vorhergesagt, wird von einer *richtig negativen (RN)* Klassifizierung gesprochen. Ein negativ eingestufter Datensatz, der aber positiv ist, wird *falsch negativ (FN)* genannt. Der umgekehrte Fall, ein negativer Datensatz wird positiv eingeschätzt, heißt *falsch positiv (FP)*.[25] Ausgehend von der Konfusionsmatrix lassen sich Metriken, wie die *Treffergenauigkeit*, die *Spezifität*, die *Falsch-positiv-Rate (FPR)*, die *Trefferquote*, die *Genauigkeit*, der *F_1-Score*, *Receiver Operating Characteristic (ROC)* Kurve oder die *Area Under the Receiver Operating Characteristics (AUROC)*, berechnen. Die folgenden Formeln sowie Erläuterungen sind Tharwat 2020, S. 170 ff. zu entnehmen.

Die Treffergenauigkeit (engl. accuracy) berechnet das Verhältnis zwischen den richtig eingestuften Datensätzen und der Anzahl aller Datensätze. Es kann ein Wert zwischen 0 und 1 angenommen werden, wobei 1 den optimalen Wert darstellt, bei welchem keine falschen Vorhersagen vorliegen.

$$Treffergenauigkeit = \frac{RP + RN}{RP + RN + FP + FN}$$

[25] Vgl. Tharwat 2020, S. 170.

Zur Berechnung der Trefferquote (engl. recall), oder auch Richtig-positiv-Rate (RPR) genannt, wird das Verhältnis zwischen den richtig vorhergesagten RP zu der Anzahl aller richtigen Datensätzen $RP + FN$ herangezogen. Demnach umschreibt die Trefferquote den Anteil der positiven Datensätze, die richtig klassifiziert wurden. Die Trefferquote akzeptiert einen Wert zwischen 0 und 1. Dabei sagt eine Trefferquote von 1 aus, dass alle richtigen Datensätze als richtig eingestuft werden und es somit keine FN gibt.

$$Trefferquote = \frac{RP}{RP + FN}$$

Die Spezifität (engl. specificity), oder auch Richtig-negativ-Rate (RNR) genannt, bezeichnet das Verhältnis zwischen den richtigen negativen Vorhersagen RN und allen falschen Datensätzen $RN + FP$. Somit stellt die Spezifität die invertierte Trefferquote dar und beschreibt den Anteil der als richtig eingestuften negativen Datensätze. Genauso wie die Trefferquote akzeptiert die Spezifität einen Wert zwischen 0 und 1. Bei einer Spezifität von 1 wurden alle negativen Datensätze als solche eingestuft.

$$Spezifität = \frac{RN}{RN + FP}$$

Die FPR repräsentiert den Anteil der negativen Datensätze, die falsch eingestuft wurden. Bei einem anzunehmenden Intervall der FPR zwischen 0 und 1 sagt der Wert 1 aus, dass alle negativen Datensätze falsch eingestuft wurden und somit keine RN vorliegen.

$$FPR = 1 - RNR = \frac{FP}{FP + RN} = \frac{FP}{N}$$

Die Genauigkeit (engl. precision) wird definiert als das Verhältnis zwischen den tatsächlich richtigen Datensätzen RP und der Anzahl der als richtig vorhergesagten Datensätze $RP + FP$. Es kann ein Wert in dem Intervall von 0 bis 1 angenommen werden. Die Genauigkeit 1 besagt, dass keine FP vorhanden sind.

$$Genauigkeit = \frac{RP}{RP + FP}$$

Für die Berechnung des balancierten F_1-Scores wird das harmonische Mittel der Ge-

nauigkeit und der Trefferquote gebildet.[26] Dabei ist zu beachten, dass der F_1-Score unabhängig von der Anzahl der RN ist. Demnach bezieht der F_1-Score nur drei der vier Elemente der Konfusionsmatrix mit ein.[27] Vergleichbar mit der Trefferquote und der Genauigkeit akzeptiert der F_1-Score einen Wert zwischen 0 und 1. Je höher die Genauigkeit sowie die Trefferquote, desto größer wird der F_1-Score. Ein F_1-Score von 1 drückt aus, dass keine FP sowie FN und somit keine falschen Vorhersagen vorliegen. Der balancierte F_1-Score versucht, sowohl die FN als auch die FP zu verringern. Um den FN oder FP eine höhere Gewichtung zu geben, kann der F_β-Score angewandt werden.

$$F_1 - Score = \frac{2}{\frac{1}{Genauigkeit} \cdot \frac{1}{Trefferquote}} = \frac{2RP}{2RP + FP + FN}$$

Die ROC-Kurve ist ein Graph, welcher die Performanz eines Klassifikationsmodells für jeden möglichen Grenzwert aufzeigt. Der Grenzwert definiert den Wert, ab dem ein Datensatz einer spezifischen Klasse zugeordnet wird. Auf der x-Achse der ROC-Kurve wird die FPR dargestellt und die y-Achse repräsentiert die RPR bzw. die Trefferquote. Je geringer der Grenzwert, desto mehr Datensätze werden als positiv eingestuft, wodurch die FPR sowie RPR erhöht werden. Zur Evaluierung der ROC-Kurve kann die Metrik AUROC verwendet werden. Die AUROC berechnet die gesamte zweidimensionale Fläche unter der ROC-Kurve, das Integral, im Intervall zwischen 0 und 1. Demnach kann die AUROC einen Wert zwischen 0 und 1 annehmen. Ein Modell, wessen Vorhersagen zu 100 % falsch sind, hat ein Integral von 0. Hingegen ein Modell, das jeden Datensatz der zugehörigen Klasse zuordnet, weist ein AUROC von 1 vor. Wenn ein AUROC von 0,5 vorzufinden ist, kann das Modell keine Unterschiede zwischen den Klassen ausfindig machen.[28]

Neben der Konfusionsmatrix und der daraus entstehenden Metriken kann zur Evaluierung eines Modells die *Wahrscheinlichkeitsdichtefunktion*, oder auch *Dichtefunktion* genannt, betrachtet werden. Die Wahrscheinlichkeitsdichtefunktion $p(x)$ wird für kontinuierliche Variablen genutzt und zeigt das Vorkommen der verschiedenen Wahrschein-

[26] Vgl. Tharwat 2020, S. 174; Vgl. Rößler und Ungerer 2019, S. 40.
[27] Vgl. Chicco und Jurman 2020, S. 2.
[28] Vgl. Google Developers 2020b; Google Developers 2020a.

lichkeiten auf. Da es sich bei der Wahrscheinlichkeitsdichtefunktion um kontinuierliche Werte handelt, die jeden Wert in einem Intervall annehmen können, wird die Wahrscheinlichkeitsdichtefunktion als ein Graph dargestellt. Folglich gibt die Wahrscheinlichkeitsdichtefunktion keine Wahrscheinlichkeit für einen gewissen Wert, sondern die Wahrscheinlichkeit, dass ein Wert in einem bestimmten infinitesimalen Bereich landet. Die Wahrscheinlichkeitsdichtefunktion $p(x)$ muss die folgenden Charakteristika vorweisen:[29]

- Die Funktion p muss alle möglichen x-Werte beinhalten.
- $\forall x \in x, p(x) \geq 0$. $p(x)$ darf dabei den Wert 1 überschreiten.
- Das Integral der Funktion p muss 1 sein: $\int p(x)\, dx = 1$.

Random Forest

Es gibt diverse überwachte Lernalgorithmen, die genutzt werden, um ein Modell zu erstellen. Darunter fällt beispielsweise der *Nearest Neighbor Algorithmus*, der Datenpunkte aufgrund von ihrem Abstand einer Klasse zuordnet, oder der *Support Vector Machine Algorithmus*, der Klassen mit dem größten Abstand aufspaltet.[30] Aufgrund der Tatsache, dass in der vorliegenden Arbeit der *Random Forest Algorithmus* eingesetzt wird, wird dieser im Detail betrachtet.

Der Random Forest Algorithmus gilt als Erweiterung der *Entscheidungsbäume*.[31] Ein Entscheidungsbaum ist ein azyklischer binärer Graph, der mit einem Wurzelknoten startet und nach N Verzweigungen, die jeweils eine Entscheidung darstellen, in einem Blatt und somit einer Vorhersage endet.[32] Bei jeder Entscheidung wird ein Merkmal des Merkmalsvektors betrachtet. Eine Vorhersage stellt einen Pfad durch einen Entscheidungsbaum dar. Der Entscheidungsbaum kann genauso wie der Random Forest für Klassifikations- sowie Regressionsprobleme eingesetzt werden.[33] Der Random Forest, welcher von dem Statistiker Leo Breiman entwickelt wurde,[34] ist der Definition

[29] Vgl. Goodfellow, Bengio und Courville 2016, S. 56.
[30] Vgl. Welsch, Eitle und Buxmann 2018, S. 373.
[31] Vgl. Welsch, Eitle und Buxmann 2018, S. 373.
[32] Vgl. Burkov 2019, Kapitel 2, S. 9.
[33] Vgl. Donges 2019.
[34] Vgl. Frochte 2019, S. 155.

nach eine Kombination mehrerer Entscheidungsbäume: „A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(x, \Theta_k), k = 1, ...\}$ where the $\{\Theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x.“[35] Es entsteht ein Wald (engl. forest) an Bäumen. Jeder Baum k wird auf Basis eines Trainingsdatensatzes x als Eingabe erstellt.[36] Somit werden auf Basis eines Datensatzes mehrere Entscheidungsbäume aufgebaut. Bei jedem Baum wird bei jeder Verzweigung eine zufällige Teilmenge von Merkmalen betrachtet. Durch die zufällige Auswahl von Merkmalen eines jeden Baumes wird sichergestellt, dass der Einfluss von stark korrelierten Merkmalen reduziert wird. Die endgültige Vorhersage wird aus den Ergebnissen aller Entscheidungsbäume berechnet.[37] Bei einem binären Klassifikationsmodell gewinnt bei einem Datensatz die Klasse, für welche die meisten Entscheidungsbäume gestimmt haben. Demnach liegt der Grenzwert bei einem binären Klassifikationsmodell bei 0,5. Wenn bei einem Modell mit den Klassen A und B mehr als 50 % der Entscheidungsbäume für die Klasse A stimmen, sagt der Random Forest die Klasse A vorher.[38]

Der Random Forest stellt diverse Anforderungen an den Datensatz, auf welchem dieser angewandt wird. Die Zielvariable Y sowie die Merkmale des Modells können numerische oder binäre Werte annehmen. Enthalten die Merkmale oder die Zielvariable kategorische Werte, müssen diese zum Beispiel mittels des *One-Hot Encodings* in binäre Vektoren konvertiert werden.[39] Aufgrund der Tatsache, dass der Random Forest versucht, die gesamte Fehlerrate des Modells zu verringern, fokussiert sich der Algorithmus auf die Genauigkeit der überlegenen Klasse und vernachlässigt dadurch die weniger vertretene Klasse. Folglich sollte der auf den Random Forest angewendete Datensatz hinsichtlich der Zielvariablen ausbalanciert sein. Eine Möglichkeit, das Problem von Imbalancen bei einer binären Zielvariable anzugehen, stellt das Heruntertakten (engl. downsampling) der vermehrt vertretenen Klasse dar.[40] Eine weitere Anforderung an den Datensatz ist, dass dieser keine Null-Werte enthalten darf. Diese müssen entweder

[35] Breiman 2001, S. 6.
[36] Vgl. Liu, Wang und J. Zhang 2012, S. 247.
[37] Vgl. Donges 2019.
[38] Vgl. Baumann et al. 2014, S. 95.
[39] Vgl. Cerda und Kégl 2018, S. 1478; Vgl. Apache Spark 2018b.
[40] Vgl. Chen, Breiman und Liaw 2004, S. 2.

entnommen oder mittels einer Datenimputation vervollständigt werden.[41] Zudem müssen die Spalten des Datensatzes mindestens zwei verschiedene Inhaltswerte aufweisen, damit diese einen Einfluss auf die Zielvariable vorweisen. Andernfalls handelt es sich um eine redundante Spalte, welche die Zielvariable nicht beeinflusst.[42]

Zusammenfassende Vorgehensweise beim überwachten maschinellen Lernen

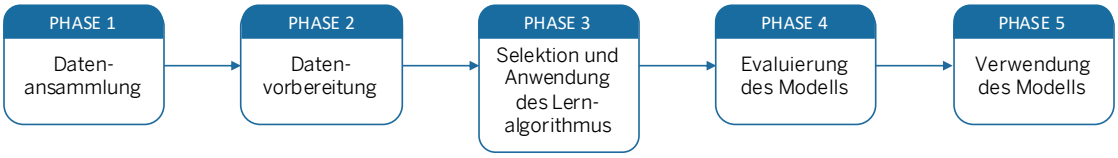


Abbildung 2.5: Prozess des überwachten maschinellen Lernens (eigene Darstellung) Vgl. Frochte 2019, S. 20 ff.

Der Prozess des überwachten maschinellen Lernens, welcher in Abbildung 2.5 dargestellt wird, startet mit der Sammlung von Datenpaaren der Form $(Eingabe, Ausgabe)$. Als Eingabe können verschiedene Formen von Daten übergeben werden, wie zum Beispiel Texte, Zahlen, Bilder oder E-Mails. Ausgaben sind üblicherweise Zahlen oder Klassen, wie „Spam“ oder „kein Spam“.[43] In dem nächsten Schritt werden die Daten aufbereitet. Die Datenaufbereitung setzt sich zum einen aus der Datenbereinigung und zum andern aus der Datenvorbereitung zusammen. Ausschlaggebend für die Vorhersage ist eine angemessene Qualität der Daten. Darunter fällt unter anderem das Entfernen redundanter sowie fehlerhaften Daten oder der Umgang mit Ausreißern. Darüber hinaus müssen irrelevante Merkmale entfernt werden, um zu verhindern, dass diese die Ergebnisse missverständlich beeinflussen.[44] Bei der Datenaufbereitung werden die Eingabewerte zu Merkmalsvektoren konvertiert, um maschinenlesbare Eingabedaten zu erzeugen. Die Ausgabewerte müssen abhängig von ihrer Beschaffenheit ebenfalls angepasst werden. Beispielsweise akzeptieren manche Lernalgorithmen nur binäre Werte. Folglich müssten die Ausgabewerte in binäre Zahlen umgerechnet werden.[45] Im dritten

[41] Vgl. Ströhl und Rockel 2020, S. 5.
[42] Vgl. Arora 2019.
[43] Vgl. Burkov 2019, Kapitel 1 S. 5.
[44] Vgl. Welsch, Eitle und Buxmann 2018, S. 371 f.
[45] Vgl. Burkov 2019, Kapitel 1 S. 5.

Schritt wird ein Lernalgorithmus passend zu dem vorliegenden Anwendungsfall selektiert. Die Wahl des Lernalgorithmus ist abhängig von der Art des zugrundeliegenden Problems. Es muss differenziert werden, ob es sich um ein Klassifikations- oder Regressionsproblem handelt.[46] Nach der Wahl des Lernalgorithmus wird dieser auf den vorbereiteten markierten Datensatz angewandt.[47] Der Datensatz wird in einen Trainings-, Test- sowie Validierungsdatensatz aufgeteilt. Mithilfe des Trainingsdatensatzes stellt der Lernalgorithmus ein Modell auf. Zur Validierung des aufgebauten Modells wird der Testdatensatz herangezogen. Dem Modell werden die Eingabewerte des Testdatensatzes übergeben und das Modell gibt Empfehlungen aus. Die Genauigkeit des Modells und somit der Empfehlungen kann mit verschiedenen Metriken, wie dem F_1-Score oder der Treffergenauigkeit, evaluiert werden.[48] Um sicherzustellen, dass das Modell nicht überangepasst ist, wird der Validierungsdatensatz eingesetzt.[49]

2.2 Definition der Enterprise-Software

Enterprise-Software, oder auch Enterprise Application Software (EAS) genannt, inkludiert gemäß Gartner Inhalts-, Kommunikations- und Kollaborationssoftware.[50] Dazu zählen zum Beispiel Enterprise-Resource-Planning (ERP), Customer-Relationship-Management (CRM) oder Supply-Chain-Management (SCM) Systeme.[51] Aufgrund des EAS-Ansatzes müssen Unternehmen Informationssysteme nicht mehr selbst entwickeln, sondern erwerben diese über einen Drittanbieter.[52] Bekannte Anbieter von EAS sind SAP, Oracle oder Microsoft. EAS bieten Komponenten und Module an, die bestimmte Geschäftsprozesse abbilden, welche zumeist an die Geschäftsprozesse des Kunden angepasst werden müssen.

EAS werden als On-Premise- oder Cloud Computing-Lösung angeboten. Bei der traditionellen On-Premise-Lösung kauft der Kunde eine Softwarelizenz und installiert diese

[46] Vgl. Welsch, Eitle und Buxmann 2018, S. 373.
[47] Vgl. Burkov 2019, Kapitel 1 S. 5.
[48] Vgl. Welsch, Eitle und Buxmann 2018, S. 371.
[49] Vgl. Xu und Goodacre 2018, S. 249 f.
[50] Vgl. Gartner 2021.
[51] Vgl. Shin 2006, S. 244; Vgl. Li et al. 2017, S. 178.
[52] Vgl. Shin 2006, S. 242.

auf seinen Rechnern.[53] Das National Institute of Standards and Technology (NIST) definiert Cloud Computing wie folgt: „Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."[54] Beim Cloud Computing werden dem Kunden Ressourcen, wie Server, Datenbanken, Speicher oder Analysefunktionen, über das Internet bereitgestellt.[55] Ein Vorteil des Cloud Computings gegenüber der On-Premise-Lösung sind die Kosten. Beim Cloud Computing fallen keine Kosten für den Erwerb von Software, Hardware oder die Rechenzentren für den Betrieb der Software an. Der Kunde bezahlt lediglich den Preis für die tatsächliche Nutzung der Informationstechnik (IT)-Ressourcen.[56]

2.3 Kontextbasierte Empfehlungssysteme

Zur Lieferung einer Definition von **kontextbasierten Empfehlungssystemen** bedarf es einer Definition der Begriffe **Kontext** und **Empfehlungssysteme**.

Definition des Kontextes

In der Literatur sind verschiedene Definitionen des Begriffes Kontext abhängig von der Anwendung und den vorhandenen Daten vorzufinden.[57] Durch das Vorliegen diverser Begriffsdefinitionen werden drei ausgewählte Definitionen von Kontext betrachtet, mit dem Ziel, eine eigene Begriffsdefinition zu erschließen.

Der Begriff *Kontext* wurde das erste Mal von Schilit und Theimer im Jahre 1994 definiert.[58] In diesem Zusammenhang bezogen sich Schilit und Theimer ausschließlich auf einen ortsabhängigen Kontext: „In general, location information enables software to adapt according to its location of use, the collection of nearby people and objects, as

[53] Vgl. Li et al. 2017, S. 178.
[54] Mell und Grance 2011, S. 2.
[55] Vgl. Microsoft 2021.
[56] Vgl. Amazon Web Services 2021; Vgl. Microsoft 2021.
[57] Vgl. Domingues, Jorge und Soares 2009, S. 1 f.
[58] Vgl. Dey 2001, S. 4.

well as the changes to those objects over time."[59] Im Jahre 1999 erkannten Schmidt et al., dass sich Kontext nicht nur auf die Lokation bezieht. Sie erweiterten den Begriff Kontext in ihrem Schreiben „There is more to context than location": „A primary concern of context-awareness in mobile computing is awareness of the physical environment surrounding a user and their ultra-mobile device."[60] Der Begriff Kontext wurde von Schmidt et al. in zwei Dimensionen aufgeteilt, die menschlichen Faktoren und die physikalische Umgebung. Zu den menschlichen Faktoren zählt der Benutzer, die soziale Umgebung und die zu erfüllende Aufgabe. Die physikalische Umgebung spaltet sich in der ersten Stufe in die Infrastruktur und die Lokation auf.[61] Ausgehend von den Definitionen von Schilit und Theimer stellte Dey eine eigene Definition für den Begriff Kontext auf. Er kritisierte, dass die Definition von Schilit und Theimer zu spezifisch sei. Aufgrund dessen, dass der Kontext abhängig von der jeweiligen individuellen Situation sei, können nicht alle relevanten Aspekte im Vorhinein aufgelistet werden. Diesen Kritikpunkt miteinbezogen definierte Dey den Kontext als jede Information, die verwendet werden könne, um die Situation einer Entität zu charakterisieren. Eine Entität könne eine Person, ein Ort oder ein Objekt sein, welches als relevant für die Interaktion zwischen einem Benutzer und einer Anwendung angesehen werde.[62] Demnach kann eine Information als Kontext eingestuft werden, wenn diese den Zustand eines Beteiligten einer Interaktion beschreibt.[63]

Folglich zeigen die aufgelisteten Definitionen, dass es verschiedene Ansätze gibt, den Begriff Kontext zu definieren. Zu erkennen ist, dass sich der Begriff über die Zeit von einem ortsbezogenen Kontext zu einem facettenreichen Kontext, der abhängig von der jeweiligen Situation ist, erweitert und konkretisiert hat. In der vorliegenden Arbeit wird mit der folgenden Definition von Kontext gearbeitet, die sich aus den zuvor genannten Definitionen erschließt: Kontext beinhaltet jegliche Informationen, die den situationsabhängigen Zustand einer Entität in einer spezifischen Interaktion charakterisieren.

[59] Schilit und Theimer 1994, S. 22.
[60] Schmidt, Beigl und Gellersen 1999, S. 893.
[61] Vgl. Schmidt, Beigl und Gellersen 1999, S. 895.
[62] Vgl. Dey 2001, S. 5.
[63] Vgl. Ntanos et al. 2014, S. 138.

Definition der Empfehlungssysteme

Empfehlungssysteme sammeln Informationen über die Präferenzen und die zuvor konsumierten Inhalte eines Benutzers für eine Menge von Elementen und sagen auf Basis der gesammelten Informationen das optimale Element dieser Menge für den Benutzer vorher.[64]: „[A recommendation system describes] any system that produces individualised recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options.“[65] Üblicherweise gibt es Benutzer, die Inhalte konsumieren, und es existiert eine Historie über die von den Benutzern konsumierten Inhalte. Im Bereich der Enterprise-Softwareindustrie wird unter Konsumieren der Kauf sowie der Gebrauch von Software verstanden. Alleine das Interesse des Kunden an einer bestimmten Software wird in der Enterprise-Softwareindustrie von dem zugehörigen Vertriebsmitarbeiter entsprechend dokumentiert und ist somit Teil der Kaufhistorie. Auf Basis der Historie der konsumierten Inhalte sollen dem Benutzer neue Inhalte vorgeschlagen werden, die dem Benutzer gefallen. Im Business-to-Consumer (B2C) Bereich könnten Filme auf Netflix oder Bücher auf Amazon vorgeschlagen werden.[66]

Zum Vorschlagen eines Elements kann eine Nützlichkeitsfunktion (engl. utility function) aufgestellt werden, welche die Nützlichkeit n eines Elements e für einen Benutzer b angibt. Für den Benutzer b soll eine Empfehlung ausgesprochen werden. Im Anwendungsbeispiel der Filmindustrie kann der Benutzer b der Betrachter eines oder mehrerer Filme sein. Dabei sei B die Menge der Benutzer und E die Menge der Objekte, die empfohlen werden können, wie zum Beispiel Bücher oder Filme. Die Dimensionen B und E setzen sich aus verschiedenen Attributen zusammen. Der Benutzer kann über die Attribute Name, Adresse, Geschlecht und Alter definiert werden und die Elemente über einen Namen, eine Kategorie und einen Preis. Ausgehend von dem Inhalt der Attribute kann ein Profil der Dimensionen, beispielsweise das Profil eines Benutzers, erstellt werden. Folglich fasst das Profil die Eigenschaften einer Dimension zusammen.[67] Gesucht wird eine Zuordnung f, die für alle Benutzer B eine Nützlichkeit N für die jeweiligen Elemente E angibt: $f : B \times E \rightarrow N$. Die Nützlichkeitswerte N repräsentieren in den

[64] Vgl. Seyednezhad et al. 2018, S. 5.
[65] Burke 2002, S. 331.
[66] Vgl. Burkov 2019, Kapitel 10, S. 7.
[67] Vgl. Adomavicius, Sankaranarayanan et al. 2005, S. 14.

meisten Fällen ein Rating, das die Meinung eines Benutzers bezüglich eines bestimmten Elementes indiziert. Übertragen in das Beispiel der Filmindustrie bedeutet dies, dass eine Funktion aufgestellt werden soll, die einen Wert ausgibt, welcher erklärt, wie stark ein Film den Interessen eines spezifischen Betrachters entspricht. Für jeden Benutzer $b \in B$ werden die ausgewählten Elemente $e \in E$ gesucht, die die Nützlichkeitswerte des Benutzers b maximieren.[68] Folglich wird der Film gesucht, der die Interessen des Benutzers am besten widerspiegelt.

Zur Vorhersage von Empfehlungen bedarf es an Daten über die Präferenzen der Benutzer für eine Menge von Elementen. Diese Informationen können entweder **explizit** durch Bewertungen der Benutzer oder **implizit** durch das Beobachten der Benutzer sowie die Analyse der Benutzeraktivitäten generiert werden.[69] Dabei bilden Bewertungen die Einschätzungen eines Benutzers bezüglich eines betrachteten Objektes ab.[70] Ein wesentlicher Vorteil der expliziten Informationen ist der Verzicht auf aufwendige Analysen zum Erlangen der Daten. Im Umkehrschluss ist die Aktivität der Benutzer gefragt, eine Bewertung abzugeben. Implizite Informationen werden durch eine Analyse des Benutzerverhaltens generiert, wodurch keine Aktivität der Benutzer erforderlich ist.[71] Durch das passive Beobachten des Benutzers können die Präferenzen und Motive eines Benutzers jedoch nur vermutet werden, wodurch ein *Rauschen* in der Datenbasis verursacht werden kann. Beispielsweise bei dem Vorliegen einer Kaufhistorie eines Produktes weiß das Unternehmen zwar, dass ein Kunde ein Produkt gekauft hat. Dem Unternehmen ist jedoch nicht bekannt, ob der Kunde mit dem Produkt zufrieden ist oder das Produkt verschenkt hat.[72] In der Literatur können zwei Arten von Rauschen in Empfehlungssystemen identifiziert werden: *natürliches* und *böswilliges* Rauschen. Das hier betrachtete Rauschen kann als natürliches Rauschen klassifiziert werden. Natürliches Rauschen kann durch zwei Ursachen ausgelöst werden. Zum einen können sich die Präferenzen sowie Prioritäten des Benutzers langfristig ändern. Zum anderen gibt es verschiedene Faktoren, die einen Benutzer beeinflussen, wie zum Beispiel die persönlichen Umstände, das soziale Umfeld, der emotionale Zustand oder der Kon-

[68] Vgl. Adomavicius und Tuzhilin 2005, S. 734 f.
[69] Vgl. Seyednezhad et al. 2018, S. 5 ff.
[70] Vgl. Brockhaus Enzyklopädie 2021a.
[71] Vgl. Seyednezhad et al. 2018, S. 8 f.
[72] Vgl. Hu, Koren und Volinsky 2008, S. 264.

text des Benutzers. Verrauschte Daten können unter anderem korrigiert werden, indem zusätzliche Informationen des Benutzers hinzugezogen werden.[73]

Zur Erstellung eines Empfehlungssystems müssen diverse Anforderungen erfüllt werden. Es müssen entsprechende Datensätze mit einer angemessenen Datenqualität zur Verfügung stehen und es muss ein Filteralgorithmus gewählt werden.[74] Im Allgemeinen weisen Empfehlungssysteme die folgenden drei Charakteristika auf:[75]

- Es existiert ein **Hintergrunddatensatz**, welcher dem System bekannt ist, bevor der Empfehlungsprozess beginnt. Beispielsweise bei den Vorschlägen von Filmen handelt es sich bei Hintergrunddaten um die historischen Daten aller Benutzer, ihre geschauten Filme und etwaige Filmbewertungen. Die Daten liegen vor, bevor das Empfehlungssystem eine Vorhersage trifft.
- Dem Empfehlungssystem werden **Inputdaten** übergeben, auf deren Basis eine Empfehlung ausgesprochen wird. Bei den Inputdaten werden im Anwendungsbeispiel eines Empfehlungssystems für Filme die Daten eines spezifischen Benutzers verstanden, für welchen eine Empfehlung ausgesprochen werden soll.
- Ein **Algorithmus** verbindet den Hintergrunddatensatz mit den Inputdaten, mit dem Ziel, eine Vorhersage treffen zu können. Der Algorithmus sagt auf Basis der vorhandenen Hintergrunddaten und den übergebenen Inputdaten des Benutzers spezifische Filme angepasst an seine Interessen vorher. Der Algorithmus erstellt die Zuordnung $f : B \times E \rightarrow N$, welche für einen spezifischen Benutzer B eine Nützlichkeit N für die jeweiligen Elemente E, in dem Anwendungsszenario die Filme, angibt und sucht das Element bzw. den Film, für den der Nützlichkeitswert maximal wird. Somit sagt der Nützlichkeitswert aus, wie stark ein Film den Interessen des Benutzers entspricht. Der Algorithmus versucht den Film zu finden, der am besten zu einem Benutzer und seinen Interessen passt.

[73] Vgl. Choudhary, Kant und Dwivedi 2017, S. 855.
[74] Vgl. Seyednezhad et al. 2018, S. 5.
[75] Vgl. Burke 2002, S. 332 f.

Definition der kontextbasierten Empfehlungssysteme

In dem Abschnitt 2.3 werden Empfehlungssysteme betrachtet, die Empfehlungen basierend auf den Hintergrund- und Inputdaten, bestehend aus den Benutzern B und den Elementen E der Form $f : B \times E \to N$, vorhersagen. Jedoch ist es nicht ausreichend, ausschließlich die Benutzer und die zu empfehlenden Elemente zu betrachten. Darüber hinaus ist es relevant, kontextuelle Informationen in den Empfehlungsprozess miteinzubeziehen und das zweidimensionale Ursprungsmodell hin zu einem multidimensionalen Modell zu erweitern. Unter Hinzunahme von kontextuellen Informationen kann das traditionelle Empfehlungsmodell angepasst werden: $f : B \times E \times K \to N$.[76] Die kontextuellen Informationen können unter anderem Daten über die Zeit, den Ort, den sozialen Kontext oder die Umgebung des Benutzers umfassen.[77] Bei der Dimension Zeit können einem Benutzer zum Beispiel unter der Woche Nachrichten angezeigt werden, währenddessen ihm am Wochenende Filmreviews vorgeschlagen werden. Aufgrund des Kontextes kann derselbe Benutzer verschiedene Entscheidungsprozesse durchlaufen und somit diverse Produkte, Marken oder Dienstleistungen präferieren.[78] Wird das zuvor aufgestellte kontextbasierte Empfehlungsmodell $f : B \times E \times K \to N$ betrachtet, gibt es formal gesehen n Dimensionen, die eine Empfehlung beeinflussen, den Benutzer B, die Elemente E und den Kontext K, der sich aus verschiedenen Dimensionen zusammensetzen kann. Ausgehend von den n Dimensionen wird eine n-dimensionale Nutzenfunktion aufgestellt, welche die Dimensionen Benutzer und Elemente um den Kontext erweitert: $f : B \times E \times K_1 \times K_2 \times ... \times K_m \to N$.[79] Beispielsweise können die drei Dimensionen Benutzer, Elemente und Zeit die Basis eines Empfehlungssystems bilden, für welche die Nutzenfunktion wie folgt aussehen würde: $f : Benutzer \times Elemente \times Zeit \to N$.[80]

[76] Vgl. Adomavicius, Sankaranarayanan et al. 2005, S. 1 f.
[77] Vgl. Abbas, L. Zhang und Khan 2015, S. 671.
[78] Vgl. Adomavicius, Sankaranarayanan et al. 2005, S. 2.
[79] Vgl. Kim und Yoon 2016, S. 15303.
[80] Vgl. Domingues, Jorge und Soares 2013, S. 699; Vgl. Adomavicius, Sankaranarayanan et al. 2005, S. 11 f.

State of the Art in der Enterprise-Softwareindustrie

Die Relevanz von Kontextinformationen für Empfehlungssysteme wurde erstmals im Jahre 2001 von Adomavicius und Tuzhilin erkannt.[81] In den darauffolgenden Jahren wurden diverse wissenschaftliche Artikel veröffentlicht, die sich mit Empfehlungssystemen und dem Einfluss kontextbasierter Daten beschäftigen.

Park et al. analysierten 210 Artikel über Empfehlungssysteme, die zwischen 2001 und 2010 erschienen sind. Die Empfehlungssysteme der jeweiligen Artikel wurden in die Anwendungsfelder Bücher, Dokumente, Bilder, Filme, Musik, Einkaufen, Television (TV) Programme und Sonstige unterteilt.[82] Zu erkennen ist, dass sich keine der aufgelisteten Gruppierungen explizit auf die Enterprise-Softwareindustrie bezieht, sondern der Fokus auf dem Endkonsumentenbereich liegt.

Während sich die Analysen von Park et al. auf zweidimensionale Empfehlungssysteme konzentrieren, befassen sich die Reviews von Haruna, Akmar Ismail et al. mit kontextbasierten Empfehlungssystemen. Haruna, Akmar Ismail et al. betrachteten 122 Artikel, die zwischen 2010 und 2017 veröffentlicht wurden. Von 2010 bis 2017 wird ein Aufwärtstrend für die Veröffentlichung von Artikeln über kontextbasierten Empfehlungssystemen deutlich. Die Anzahl von veröffentlichten Artikeln im Bereich kontextbasierter Empfehlungssysteme hat sich im Jahre 2016 im Vergleich zum Jahre 2010 nahezu verdreifacht. Dies unterstreicht die steigende Relevanz von kontextbasierten Empfehlungssystemen. Die kontextbasierten Empfehlungssysteme der 122 betrachteten Artikel wurden den sechs Kategorien elektronischer Handel (engl. e-commerce), elektronische Dokumente (engl. e-document), Multimedia, Lokationen, Tourismus und Sonstige zugeordnet. Wie bereits bei Park et al. wird ersichtlich, dass sich die Klassifikationen auf dem Bereich der Endkonsumenten beschränken.[83]

Darüber hinaus unterstreichen Jannach und Jugovac im Jahre 2019, dass sich die akademischen Forschungen bezüglich Empfehlungssysteme primär auf den Bereich der Endkonsumenten fokussieren, um unter anderem den Informationsüberhang zu reduzieren: „Academic research mostly focuses on the consumer perspective, with the implicit assumption that improved customer value is indirectly also beneficial for the recommen-

[81] Vgl. Adomavicius und Tuzhilin 2001, S. 1.
[82] Vgl. Park et al. 2012, S. 10060 ff.
[83] Vgl. Haruna et al. 2017, S. 5 f.

dation provider."[84] Inwieweit Empfehlungssysteme einen Mehrwert für Unternehmen stiften, wird in der Literatur bislang weniger thematisiert. In diesem Zusammenhang könnten wirtschaftliche Metriken wie der Umsatz, die Klickrate oder die Kundenbindungsrate inkludiert werden.[85] Jannach und Jugovac zeigten durch eine Untersuchung von Anwendungen von Empfehlungssystemen, dass Empfehlungssysteme helfen, den wirtschaftlichen Mehrwert für Unternehmen zu steigern: „These systems either help to increase revenue or profit directly, or they lead to indirect positive effects such as higher user engagement, loyalty, and customer retention."[86]

Folglich ist die Problemstellung zu erfassen, dass sich vermehrt mit dem Thema kontextbasierter Empfehlungen sowie Empfehlungssystemen im Bereich der Endkonsumenten beschäftigt wird und dies in der Enterprise-Softwareindustrie bisher vernachlässigt wurde. Die Möglichkeiten von kontextbasierten Empfehlungen in der Enterprise-Softwareindustrie sind demnach nicht umfassend erforscht. Analysen von Jannach und Jugovac unterstreichen den wirtschaftlichen Mehrwert von Empfehlungssystemen für Unternehmen, welcher durch die Anwendung von kontextbasierten Empfehlungssystemen gesteigert werden könnte. Die vorliegende Arbeit wird sich mit dem Einfluss von kontextbasierten Empfehlungen in der Enterprise-Softwareindustrie befassen, mit der Intention, die Empfehlungen aufgrund von Hinzunahme von Kontext konkreter im Vergleich zu herkömmlichen Empfehlungen erfassen zu können. Durch die Hinzunahme von kontextbasierten Daten könnten die wirtschaftlichen Ausmaße von Empfehlungssystemen gesteigert werden.

[84] Jannach und Jugovac 2019, S. 1.
[85] Vgl. Jannach und Jugovac 2019, S. 1 f.
[86] Jannach und Jugovac 2019, S. 16.

3 Wissenschaftliche Vorgehensweisen und Methoden

In der vorliegenden Arbeit wird die wissenschaftliche Methode des **DSR** angewandt. Als untergliederte Prozesse des DSR wird im Zuge der Umsetzung das **CRISP-DM** Vorgehensmodell verwendet.

3.1 Design Science Research

Der Pionier der wissenschaftlichen Methode des DSR ist Herbert Simon. Er publizierte im Jahre 1969 sein Buch „The Science of the Artificial“, in dem er die Methode „Science of Design“ einführte, welche später zu „Design Science“ umbenannt wurde. Aus Kritik an den traditionellen wissenschaftlichen Methoden entstand das DSR. Ein Kritikpunkt war der Fakt, dass die traditionellen wissenschaftlichen Methoden nicht für das Erstellen und das Erforschen von Systemen, die bislang nicht existieren, dediziert sind.[87] Das DSR strebt nach der Verbesserung von Technologien und wissenschaftlichen Wissensbasen durch das Kreieren eines innovativen Artefakts, welches ein vorliegendes Problem löst und die Umgebung, in welche das Artefakt eingesetzt wird, verbessert.[88] Ein Artefakt kann in Textform vorliegen, eine Software oder ein mathematisches Modell sein.[89] Es wird die Intention verfolgt, Wissen zu generieren, wie Objekte designed werden müssen, um ein spezifisches Ziel zu erfüllen.[90] Unter Design wird in diesem Sachzusammenhang, die „bewusste Organisation von Ressourcen zur Zielerreichung“[91] verstanden: „Design, on the other hand, is concerned with how things ought to be, with devising artifacts to attain goals“.[92] Als lösungsorientiertes Modell ist das

[87] Vgl. Dresch, Lacerda und Antunes Jr. 2015, S. 50.
[88] Vgl. Brocke, Hevner und Maedche 2020, S. 1.
[89] Vgl. Bichler 2006, S. 133.
[90] Vgl. Brocke, Hevner und Maedche 2020, S. 2.
[91] Bichler 2006, S. 133.
[92] Simon 1996, S. 198.

DSR darauf ausgerichtet, lediglich eine zufriedenstellende Lösung zu finden und keine optimale Lösung. Einsatzgebiete des DSR stellen unter anderem das Ingenieurswesen, die Medizin, das Recht, die Architektur und die Informatik dar.[93]

Abbildung 3.1: DSR-Ansatz angelehnt an Brocke, Hevner und Maedche 2020, S. 3

Abbildung 3.1 zeigt einen konzeptionellen Ansatz, um das DSR zu verstehen, auszuführen und zu evaluieren. Der Ansatz besteht aus drei übergeordneten Bereichen: dem Umfeld, dem Artefakt und der Wissensbasis. Das Umfeld, welches aus den Menschen, der Organisation und den Technologien besteht, definiert das wissenschaftliche Problem, welches anhand des DSR für die Stakeholder gelöst werden soll. Die Geschäftsanforderungen der Stakeholder an das Artefakt unterstreichen die Relevanz des zugrundeliegenden Problems. Die Wissensbasis stellt die Grundlagen und die Methoden bereit, mit denen das DSR durchgeführt wird. Mithilfe von Grundlagen wie Theorien, Modelle und Instrumente wird das Artefakt erstellt. Die Methodiken, wie beispielsweise Datenanalysetechniken oder Validationskriterien, werden in der Evaluierungsphase mit der Intention angewandt, das Artefakt anzupassen. Durch die ordnungsgemäße Anwendung der Grundlagen und Methoden, erlangt das Artefakt an Exaktheit.[94]

Im Kontext der vorliegenden Arbeit wird die *wissenschaftliche Forschung* im Theorieteil betrieben, mit dem Ziel, die Grundlagen des überwachten maschinellen Lernens und die Begriffsdefinitionen des Kontextes sowie der Empfehlungssysteme zu skizzieren. Diese Grundlagen werden als Basis des Artefakts angesehen. Das Artefakt stellt im Kontext der vorliegenden Arbeit das Erstellen eines kontextbasierten Empfehlungssystems dar.

[93] Vgl. Dresch, Lacerda und Antunes Jr. 2015, S. 55 f.
[94] Vgl. Brocke, Hevner und Maedche 2020, S. 2.

Das Empfehlungssystem wird durch die Anwendung des CRISP-DM Vorgehensmodells dem *Geschäftskontext* entsprechend *entwickelt und erstellt*. Durch die iterative Vorgehensweise des CRISP-DM wird das Empfehlungssystem *verfeinert* und erlangt an *Exaktheit*. In der Abbildung 1.1 sind die einzelnen Aspekte des DSR Ansatzes ebenfalls eingebettet.

3.2 Cross Industry Standard Process for Data Mining

Das CRISP-DM ist ein de-facto Standard und ein industrieunabhängiges Prozessmodell zur Anwendung von Data-Mining Projekten.[95] Gemäß der Brockhaus Enzyklopädie wird *Data-Mining* als die „softwaregestützte Ermittlung bisher unbekannter Zusammenhänge, Muster und Trends in großen Datenbanken“[96] definiert. Entwickelt wurde das CRISP-DM von Daimler-Benz im Jahre 1996.[97] Das CRISP-DM kann als das Wasserfallmodell der Data Science angesehen werden.[98] In sechs iterativen Phasen zeigt das CRISP-DM Prozessmodell den Lebenszyklus von Data-Mining Projekten auf (Abbildung 3.2). Die Beschreibung des CRISP-DM Vorgehensmodells erfolgt durch Chapman et al..[99]

In der ersten Phase, dem *Geschäftsverständnis*, werden die Projektziele sowie die Anforderungen aus der Geschäftsperspektive definiert. Die initiale Datensammlung startet in der zweiten Phase, dem *Datenverständnis*. Abhängig von den Ergebnissen des Datenverständnisses kann es notwendig sein, dass die Ziele und somit die Anforderungen des Geschäftsverständnisses angepasst werden müssen. Diese gesammelten Daten werden exploriert, beschrieben und hinsichtlich ihrer Datenqualität analysiert.[100] Die *Datenvorbereitung*, die dritte Phase, umfasst alle Aktivitäten, die aus den Rohdaten einen finalen Datensatz konstruieren. Zu den Aktivitäten zählen die Tabellen- sowie Attributsselektion und die Transformation sowie Bereinigung der Daten. Nach der Da-

[95] Vgl. Schröer, Kruse und Gómez 2021, S. 527.
[96] Brockhaus Enzyklopädie 2021b.
[97] Vgl. K. C. Laudon, J. P. Laudon und Schoder 2015, S. 305.
[98] Vgl. Saltz et al. 2018, S. 4.
[99] Vgl. Chapman et al. 2000, S. 13 f.
[100] Vgl. Schröer, Kruse und Gómez 2021, S. 527.

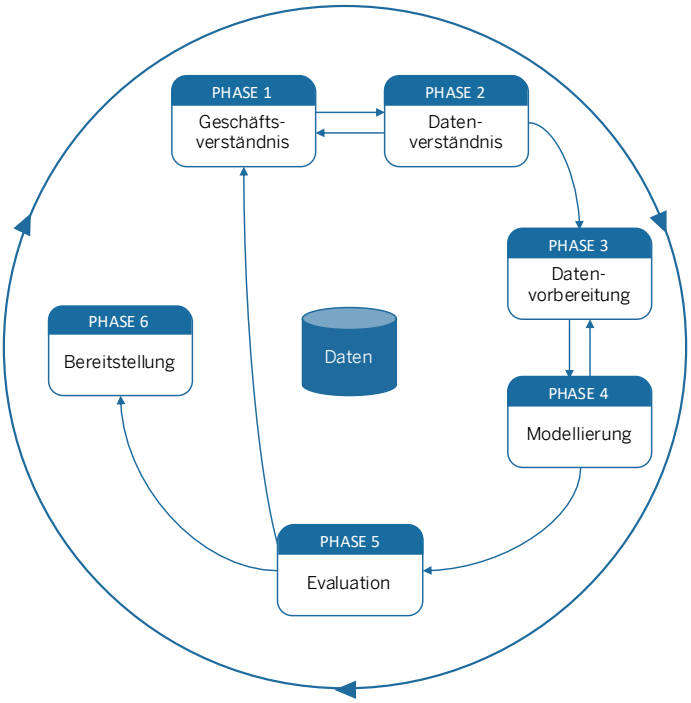


Abbildung 3.2: Phasen des CRISP-DM Vorgehensmodells angelehnt an Chapman et al. 2000, S. 13

tenvorbereitung folgt die *Modellierung*, bei der die Modelltechnik selektiert wird und dessen Parameter optimal kalibriert werden. Manche Techniken setzen bestimmte Anforderungen an die Beschaffenheit und Aufbereitung der Daten voraus. Demnach mag ein Zurückspringen in die dritte Phase erforderlich sein, um die Daten entsprechend anzupassen. Vor der Bereitstellung des entwickelten Modells ist es von Relevanz, das Modell zu evaluieren und die verschiedenen durchgeführten Schritte zu überprüfen. Unter diesem Gesichtspunkt wird kontrolliert, ob die Geschäftsziele erreicht wurden und die Richtigkeit des Modells nachzuweisen ist. Als Resultat der *Evaluation* folgt die Entscheidung, ob das Modell bereitgestellt, optimiert oder abgeschlossen wird. In der letzten Phase, der *Bereitstellung*, wird das erlangte Wissen dokumentiert und präsentiert, sodass der Kunde die Erkenntnisse versteht und nutzen kann. Die Dokumentation erfolgt durch einen finalen Bericht oder durch eine Implementierung eines wiederholbaren Data-Mining Prozesses. Dabei ist zu beachten, dass der Kunde die Schritte der Bereitstellung definiert.

4 Umsetzung anhand des Cross Industry Standard Process for Data Mining

Im Endkonsumenten-Bereich ist es, wie in dem Abschnitt 2.3 beschrieben, üblich, dass kontextbasierte Empfehlungsmethoden angewandt werden, um die Empfehlungen spezifischer auf einen Kunden und seinen Kontext anzupassen.[101] Bekannte Beispiele des Endkonsumenten-Bereichs sind das Videoportal YouTube oder der Streaminganbieter Netflix.[102] Hingegen in der Enterprise-Softwareindustrie, in dem vorliegenden Anwendungsbeispiel das Unternehmen SAP SE, werden kontextbasierte Empfehlungen seltener eingesetzt. Das deutsche Softwareunternehmen SAP SE, das im Jahre 1972 gegründet wurde und ursprünglich für seine ERP-Systeme bekannt war, ist unter anderem in den Bereichen Datenbanken, Analysen, End-to-End-Unternehmenssoftware, intelligente Technologien und Experience Management vertreten.[103] Wird die Internetseite der SAP SE aufgerufen, erhalten die Kunden eine Übersicht über eine Vielzahl von Inhalten. Dieses Vorgehen verfolgt die Intention, die Kunden in verschiedenen Stadien ihrer Customer Journey von sich zu überzeugen oder weiterhin zu binden. Die Customer Journey umschreibt den Prozess, den ein Kunde durchläuft, um ein spezifisches Ziel, wie beispielsweise das Erwerben eines Produktes oder einer Dienstleistung, zu erreichen.[104] Während dieses Prozesses findet eine wiederholte Interaktion zwischen dem Dienstleister und dem Kunden statt.[105] Angelehnt an die kontextbasierten Empfehlungen des Endkonsumenten-Bereichs ist anzunehmen, dass durch eine wohldefinierte kleinere Menge von Inhalten ein verkaufsförderndes Verhalten beim Kunden

[101] Vgl. Misztal und Indurkhya 2015, S. 2.
[102] Vgl. Netflix 2021; Vgl. Covington, Adams und Sargin 2016, S. 1 ff.
[103] Vgl. SAP SE 2021.
[104] Vgl. Følstad, Kvale und Halvorsrud 2014, S. 413.
[105] Vgl. Følstad und Kvale 2018, S. 197.

wahrzunehmen ist.[106] Hierbei stellt sich die Frage, ob der Inhalt durch einen Kontext eingegrenzt werden kann, sodass dieser passend auf den Kunden zugeschnitten ist und seine Interessen widerspiegelt. Anhand des CRISP-DM Vorgehensmodells und des DSR soll diese Fragestellung am Beispiel der SAP SE betrachtet werden.

4.1 Geschäftsverständnis

Mithilfe des CRISP-DM Vorgehensmodells soll analysiert werden, ob kontextbasierte Empfehlungen in der Enterprise-Softwareindustrie an den Kunden angepasstere Produktempfehlungen liefern als die eines zweidimensionalen Empfehlungssystems, mit dem Ziel, ein für das Unternehmen verkaufsförderndes Kundenverhalten hervorzurufen. Konkret werden Empfehlungssysteme betrachtet, die einem spezifischen Kunden die zu ihm passenden Produkte vorschlagen. Um sich dieser Fragestellung zu widmen, muss in der ersten Phase des CRISP-DM Vorgehensmodells die konkrete betriebswirtschaftliche Vorgehensweise mit den verbundenen Zielen spezifiziert und der Kontext im Rahmen von Kunden der Enterprise-Softwareindustrie definiert werden. Im weiteren Verlauf des CRISP-DM Vorgehensmodells wird dieser Kontext in ein Empfehlungssystem eingebunden. Anschließend werden die Vorhersagen dieses multidimensionalen kontextbasierten Empfehlungssystems anhand ausgewählter Metriken mit denen eines originären zweidimensionalen Empfehlungssystems verglichen. Das Geschäftsverständnis stellt dabei den Bereich des *Umfelds* des DSR dar, welches das wissenschaftliche Problem definiert und mithilfe des DSR gelöst werden soll.

Ziel- und Hypothesendefinition

Im Allgemeinen wird das Ziel verfolgt, zu evaluieren, ob kontextbasierte Empfehlungen in der Enterprise-Softwareindustrie an den Kunden angepasstere Produktempfehlungen liefern als klassische zweidimensionale Empfehlungssysteme. In dem vorliegenden Anwendungsfall wird das Unternehmen SAP SE, welches der Enterprise-Softwareindustrie zuzuordnen ist, betrachtet. Das hier definierte Ziel unterstreicht die *Relevanz* des

[106] Vgl. Adomavicius und Tuzhilin 2005, S. 744; Vgl. Adomavicius, Sankaranarayanan et al. 2005, S. 2 f.

zu lösenden Problems der *Organisation* der SAP SE im Rahmen des DSR-Ansatzes. Es wird die Hypothese aufgestellt, dass mithilfe von Kontext im Zusammenhang mit Empfehlungen die Präferenzen sowie Interessen der Kunden der SAP besser abgebildet werden können. Diese Hypothese gilt es, im weiteren Verlauf zu verifizieren oder negieren. Das klassische Empfehlungssystem der SAP soll durch ein multidimensionales Modell erweitert werden und einen zuvor zu definierenden Kontext miteinbeziehen. Aufgrund der Tatsache, dass diese kontextbasierten Empfehlungen einem konkreteren Abbild des Kunden entsprechen und die Empfehlungen den Kunden optimalerweise ansprechen, soll ein verkaufsförderndes Verhalten bei den Kunden hervorgerufen werden. Durch benutzerspezifische Produktempfehlungen erhält der Kunde Informationen über die aktuellen zu ihm passenden Produkte. Es wird die Intention verfolgt, den Kunden einen Schritt voraus zu sein und ihre Bedürfnisse neuer Software zu erkennen, bevor der Kunde sich dessen bewusst ist. Die zuvor aufgestellte Hypothese wird im weiteren Verlauf anhand des CRISP-DM Vorgehensmodells evaluiert.

Vorgehensweise

Zur Validierung oder Negierung der zuvor aufgestellten Annahme wird ein Artefakt und in diesem Fall ein Klassifikationsmodell erstellt, welches unter Hinzunahme von Kontext vorhersagt, ob ein Kunde ein Produkt in Zukunft kaufen könnte und demnach dem Kunden empfohlen werden sollte. Das Klassifikationsmodell verfügt über die beiden Klassen „kauft“ sowie „kauft nicht“ und wird somit als ein *binäres* Klassifikationsmodell definiert. Zur Evaluierung dieses Klassifikationsmodells wird ein zweites Modell aufgebaut, dass anhand des um den Kontext reduzierten Datensatzes das Kundenverhalten vorhersagt. Mithilfe der in dem Abschnitt 2.1 definierten Metriken kann analysiert werden, welches Klassifikationsmodell für den Anwendungsfall und den vorliegenden Datensatz bessere Empfehlungen liefert.

Für die Vorhersagen bzw. Empfehlungen verwenden beide Klassifikationsmodelle als *Hintergrunddaten* historische Vertriebspipelinedaten des Kunden (Tabelle A.1), die Aufschluss über die konsumierten Inhalte des Kunden liefern, sowie Kundendaten (Tabelle A.2). An dieser Stelle bedarf es keiner *Datensammlung*, da die Daten bereits erhoben wurden und vorliegen. Beide Datensätze wurden *implizit* über das Beobachten der Kun-

denaktivitäten generiert. Die Attribute der Tabellen der Vertriebspipelinedaten sowie der Kundendaten werden im Anhang A detailliert beschrieben. Zu den Kundendaten zählen generelle Daten über den Kunden, wie zum Beispiel der Beziehungsstatus oder die Industrie des Kunden. Unter dem Beziehungsstatus wird verstanden, ob ein Kunde beispielsweise von einer Partnerorganisation betreut wird oder ob dieser auf inaktiv gesetzt ist und somit seit über fünf Jahre keine Lösung erworben hat. Die Vertriebspipelinedaten beinhalten rund 1,14 Millionen historische Datensätze über die potenziellen Käufe eines Kunden seit dem 16. Februar 2004. Dabei werden bei den Vertriebspipelinedaten nicht nur Produkte miteinbezogen, die ein Kunde erworben hat, sondern auch Produkte, an denen ein Kunde Interesse gezeigt hat.[107] Aufgrund dessen, dass das vorliegende binäre Klassifikationsmodell nur zwischen gekauften und nicht gekauften Produkten unterscheidet, wird ein Interesse eines Kunden an einem Produkt der Klasse „kauft nicht“ zugeordnet. In der Abbildung 4.1 werden die *Hintergrunddaten* der Klassifikationsmodelle mit ihren Attributen in einem Entity-Relationship-Modell (ERM) dargestellt.

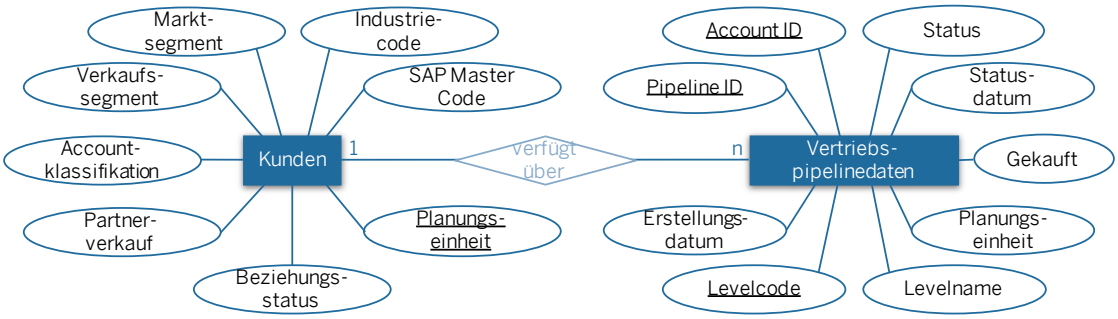


Abbildung 4.1: ERM für die Vertriebspipeline- und Kundendaten (eigene Darstellung)

Die beiden Tabellen werden über das Attribut `Planungseinheit` miteinander verbunden. Eine Planungseinheit wird bei der SAP SE als das übergeordnete Unternehmen definiert, welches die Verträge mit der SAP SE unterschreibt. Dieses Unternehmen vertritt als Planungseinheit alle ihm zugeordneten Tochtergesellschaften. Somit kann eine Planungseinheit mehrere Lösungen erwerben. Die Hintergrunddatensätze, die Kunden- und Vertriebspipelinedaten, bilden die zweidimensionale Nützlichkeitsfunktion der Form $f : Kunde \times Vertriebspipelinedaten \rightarrow N$ ab. Auf Basis der Hintergrund-

[107] Vgl. Haase 2020.

daten berechnet das Klassifikationsmodell die sogenannten *Nützlichkeitswerte*. Ein Nützlichkeitswert bezeichnet in diesem Sachzusammenhang die eindeutige Zuordnung $c : X \rightarrow C$ zu der Klasse „kauft“ oder „kauft nicht“. Folglich stellt die Zuordnung zu einer Klasse eine *diskrete* Ausgabe dar, da nur zwei Werte angenommen werden können.

Das originäre Klassifikationsmodell verwendet zur Erstellung von Vorhersagen die zuvor definierten Hintergrunddatensätze Vertriebspipeline- und Kundendaten. Unter Hinzunahme von Kontext wird dieses originäre Klassifikationsmodell zu einem multidimensionalen Modell wie folgt erweitert: $f : Unternehmen \times Vertriebspipelinedaten \times Kontext \rightarrow N$. Es werden weitere Datensätze, die im Zusammenhang mit dem Kunden und dem Kauf einer Software in der Enterprise-Softwareindustrie stehen, bereitgestellt. Anhand dieser Datensätze soll im Folgenden der Kontext definiert werden.

Definition des Begriffs „Kontext“ im Rahmen von Kunden in der Enterprise-Softwareindustrie

Nach der Definition des Begriffes *Kontext* in dem Abschnitt 4.1 wird dieser in die Enterprise-Softwareindustrie eingebettet und entsprechend spezifiziert. Durch den Kontext sollen Empfehlungen an einen Kunden optimaler angepasst werden können. Die in der Definition genannte *Entität*, die in dem vorliegenden Anwendungsfall betrachtet wird, stellt einen Kunden bzw. ein Unternehmen dar. Anhand von bereitgestellten Datensätzen, die über das notwendige Wissen der Kunden und deren Vertriebspipelinedaten hinausgehen, wird der situationsabhängige Zustand eines Unternehmens in einer spezifischen Interaktion, in diesem Fall dem Produktkauf, charakterisiert. Es wird der hypothetische Kontext für die Enterprise-Softwareindustrie am Unternehmensbeispiel der SAP SE definiert, der das Umfeld eines Kunden miteinbezieht. Die bereitgestellten Datensätze, die in einem ERM in Abbildung 4.2 aufgezeigt sind, umfassen Produkt- (Tabelle A.3), Empfehlungs- (Tabelle A.4), Funktionsumfangs- (Tabelle A.5) und Lizenzmodelldaten (Tabelle A.6). Die Attribute der aufgelisteten Tabellen werden im Anhang A detailliert erklärt. Eine Empfehlung kann ein oder mehrere Produkte umfassen. Diese Art von Empfehlungen werden manuell auf Grundlage von Expertenwissen erstellt und sind spezifisch auf eine Unternehmensfunktion angepasst. Dies hat zur

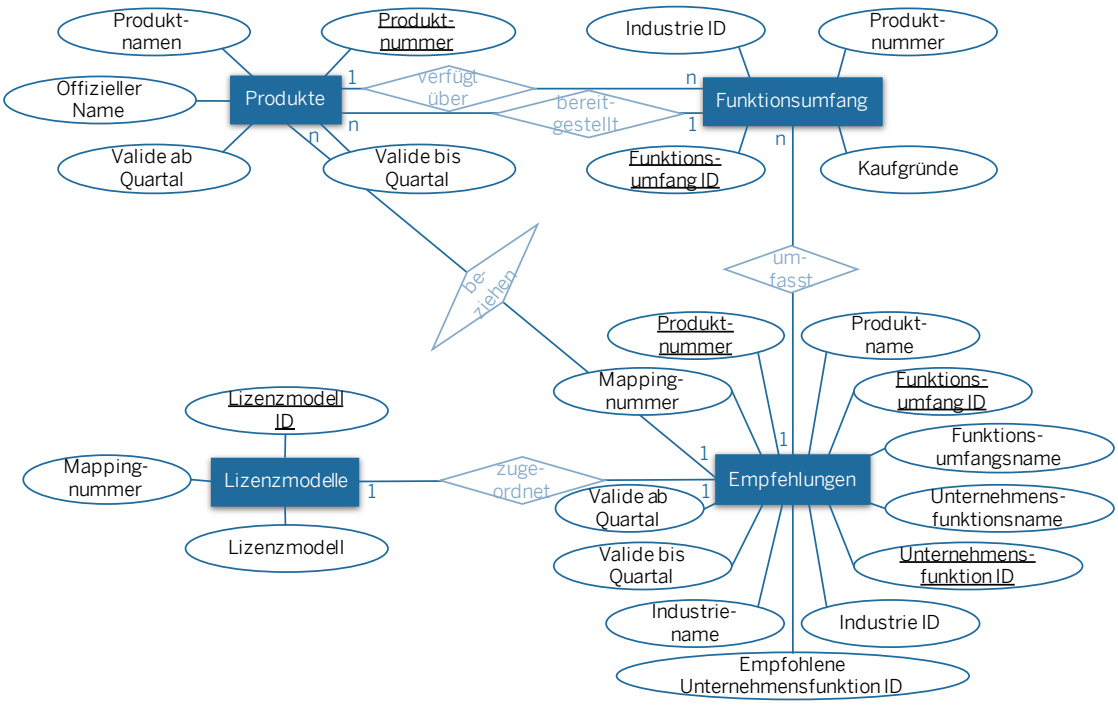


Abbildung 4.2: ERM für die Kontextdatensätze (eigene Darstellung)

Folge, dass die Empfehlungen nicht individuell auf einen Kunden zugeschnitten sind. Beinhaltet eine Empfehlung mehrere Produkte, liegen Produktalternativen vor. Ein Produkt bzw. eine Produktalternative einer Empfehlung ist einem Lizenzmodell Cloud oder On-Premise zugeordnet. Jeder Eintrag der Empfehlungstabelle wird durch einen zusammengesetzten Schlüssel der Attribute `Produktnummer`, `Funktionsumfang` und `Unternehmensfunktion` identifiziert. Eine Unternehmensfunktion stellt die von einem Kunden geforderten abzubildenden Geschäftsprozesse dar, wie zum Beispiel das Erstellen einer Gehaltsabrechnung. Werden diese Geschäftsprozesse durch SAP-Lösungen abgebildet, existieren zu der Unternehmensfunktion Funktionsumfänge, welche auf verschiedene mögliche Produkte der SAP verweisen. Ein Produkt, welches durch die `Produktnummer` identifiziert wird, kann somit über beliebig viele Funktionen verfügen. Eine Unternehmensfunktion kann durch mehrere Funktionsumfänge dargestellt werden, da die Produkte verschiedene Lizenzmodelle wie On-Premise oder Cloud vorweisen können.

Ausgehend von den bereitgestellten Datensätzen wird der hypothetische Kontext de-

finiert, welcher das originäre Klassifikationsmodell erweitert und den es im weiteren Verlauf der Modellierung zu verifizieren oder zu negieren gilt:

- **Lizenzmodell:** In der Enterprise-Softwareindustrie werden, wie in dem Abschnitt 2.2 beschrieben, Produkte als Cloud- und On-Premise-Lösungen zum Verkauf angeboten. Ein Produkt der SAP kann somit entweder als On-Premise- oder Cloud-Lösung verkauft werden. Dabei ist zu beachten, dass der Kunde sowohl Interesse an der Cloud- sowie On-Premise-Lösung eines Produktes haben kann. Seit der Einführung der Cloud-Produkte bei der SAP SE im Jahre 2010, ist ein stetiges Wachstum der Cloud-Lösungen ersichtlich. Der Umsatz der Cloud-Subskriptionen und -support wuchs von 18 Millionen Euro im Jahre 2011 auf 8.080 Millionen Euro im Jahre 2020. Im Vergleich zu den Cloud-Subskriptionen stieg der Umsatz der Softwarelizenzen und -support von 11.012 Millionen Euro im Jahre 2011 auf 15.148 Millionen Euro im Jahre 2021 an, wobei zu erkennen ist, dass der Umsatz zwischen 2015 und 2021 bei um die 15.000 Millionen Euro stagniert.[108] Demnach ist anzunehmen, dass die Anzahl an Cloud-Subskriptionen in Zukunft weiter steigen wird.
- **Lizenzmodelltendenz:** Das Verhältnis zwischen den Interessen an Cloud- und On-Premise-Lösungen eines Kunden gibt Auskunft über die Tendenz eines Kunden, ein spezifisches Lizenzmodell in der Zukunft zu kaufen. Bei diesem Kontext handelt es sich um eine abgeleitete (engl. derived) Metrik.
- **Industrie:** Abhängig von der Industrie eines Unternehmens sollen Präferenzen beim Kauf der Software erkannt werden. Ein Unternehmen einer spezifischen Industrie wird wahrscheinlich eine Lösung kaufen, die auf die Industrie des Unternehmens angepasst sowie geeignet ist und die notwendigen Geschäftsprozesse abbildet. Demnach wird die Industrie der Empfehlung als Kontext hinzugezogen. Zu beachten ist, dass einer Empfehlung mehrere Industrien zugeordnet sein können.
- **Anzahl der empfohlenen Unternehmensfunktionen:** Die Anzahl der empfohlenen Unternehmensfunktionen für einen Kunden kann die Entscheidungsfindung beeinflussen. Werden einem Kunden zur Abbildung seiner gewünsch-

[108] Vgl. SAP 2021.

ten Geschäftsprozesse mehrere Unternehmensfunktionen empfohlen, stehen dem Kunden somit mehrere mögliche Produkte zur Auswahl.

- **Anzahl der Produkte:** Ein Unternehmen kann sich innerhalb einer Pipeline für ein oder mehrere Produkte interessieren. Die Anzahl der Produkte einer Vertriebspipeline kann einen Einfluss auf die Wahrscheinlichkeit eines Verkaufs haben.
- **Verweildauer eines Kunden in einer Vertriebspipeline:** Ein Kunde, der sich in der Vergangenheit in kurzer Zeit zu einem Kauf entschieden hat, mag auch in Zukunft dazu tendieren, eine kurze Customer Journey zu durchlaufen. Dementgegen wird sich ein Kunde, dessen Entscheidungsprozess üblicherweise einen größeren Zeitintervall in Anspruch nimmt, wahrscheinlich nicht innerhalb eines definierten Zeitraums zu einem Kauf entscheiden.
- **Durchschnittliche Verweildauer eines Kunden in einer Vertriebspipeline:** Neben der Verweildauer eines Kunden in einer Vertriebspipeline ist in Betracht zu ziehen, wie die durchschnittliche Verweildauer eines Kunden in einer Vertriebspipeline aussieht, um Rückschlüsse zu der Tendenz eines Kunden schließen zu können.
- **Durchschnittliche Verweildauer eines Produktes in einer Vertriebspipeline:** Von der durchschnittlichen Dauer des Kaufens eines bestimmten Produktes kann abgeleitet werden, wie lange sich der Entscheidungsprozess eines Kunden für ein Produkt hinziehen kann. Dies kann einen Einfluss auf den Kauf eines Produktes haben.
- **Summe der gekauften Lösungen im Verhältnis zu den betrachteten Lösungen:** Abhängig davon, wie viele Lösungen ein Unternehmen bereits von der SAP SE gekauft hat, ist es wahrscheinlicher bzw. unwahrscheinlicher, dass der Kunde ein weiteres Produkt erwirbt. Ein Unternehmen, welches bereits viele SAP-Lösungen integriert hat, tendiert dazu, weitere Lösungen zu kaufen oder die vorhandenen Lösungen zu erweitern. Dabei muss beachtet werden, für wie viele Lösungen ein Kunde bereits Interesse geäußert hat. Aus diesem Grund wird das Verhältnis zwischen den gekauften Lösungen und den betrachteten Lösungen berechnet.

Schlussfolgernd setzt sich der hypothetische Kontext aus den aufgelisteten Attributen zusammen: *Kontext* $\subseteq$ *Lizenzmodell* $\times$ *Lizenzmodelltendenz* $\times$ *Industrie* $\times$ *Anzahl der empfohlenen Unternehmensfunktionen* $\times$ *Anzahl der Produkte* $\times$ *Verweildauer eines Kunden in einer Vertriebspipeline* $\times$ *durchschnittliche Verweildauer eines Kunden in einer Vertriebspipeline* $\times$ *durchschnittliche Verweildauer eines Produktes in einer Vertriebspipeline* $\times$ *Summe der gekauften Lösungen im Verhältnis zu den betrachteten Lösungen*.

4.2 Datenverständnis

Zur Erhaltung eines Überblicks über die vorhandenen Datensätze werden im zweiten Schritt des CRISP-DM Vorgehensmodells die zur Verfügung stehenden Daten inspiziert. Das Verständnis sowie die Vorbereitung der Daten, welches in dem Abschnitt 4.3 beschrieben wird, erfolgt durch das *Jupyter Notebook* und *pandas*. Mithilfe der Open Source Webanwendung *Jupyter Notebook* können Dokumente erstellt und geteilt werden, welche Code, Gleichungen, Visualisierungen und narrativen Text enthalten können.[109] *Pandas* ist ein Open Source Datenanalyse- und -manipulationstool, das auf der Programmiersprache Python aufbaut.[110]

Bei der Inspektion der Datenbasis werden im Zuge der Datenexploration die Spaltennamen der einzelnen Tabellen mit der Intention beschrieben, die Bedeutungen der Spalten zu verstehen, Verbindungen zwischen den Tabellen zu erkennen und folglich für den Anwendungsfall irrelevante Spalten auszuschließen. Die Beschreibungen der Spalten können dem Anhang A entnommen werden. Abbildung 4.3 dient der Visualisierung der Zusammenhänge der einzelnen Tabellen. Dabei ist zu beachten, dass die Tabelle „Kunde“ durch den Namen „Unternehmen“ ersetzt wird. Ein Unternehmen kann erst als Kunde betitelt werden, wenn das Unternehmen in der Vergangenheit bereits eine Lösung gekauft hat. Jedoch gibt es Unternehmen, die noch keine Lösung erworben haben, aber an einer Lösung interessiert sind. Diese Unternehmen gelten somit als potenzielle Kunden. Zudem ist dem ERM in Abbildung 4.3 zu entnehmen, dass ein Produkt in mehreren Vertriebspipelinedatensätzen vorkommen und von verschie-

[109] Vgl. Project Jupyter 2021.
[110] Vgl. pandas 2021.

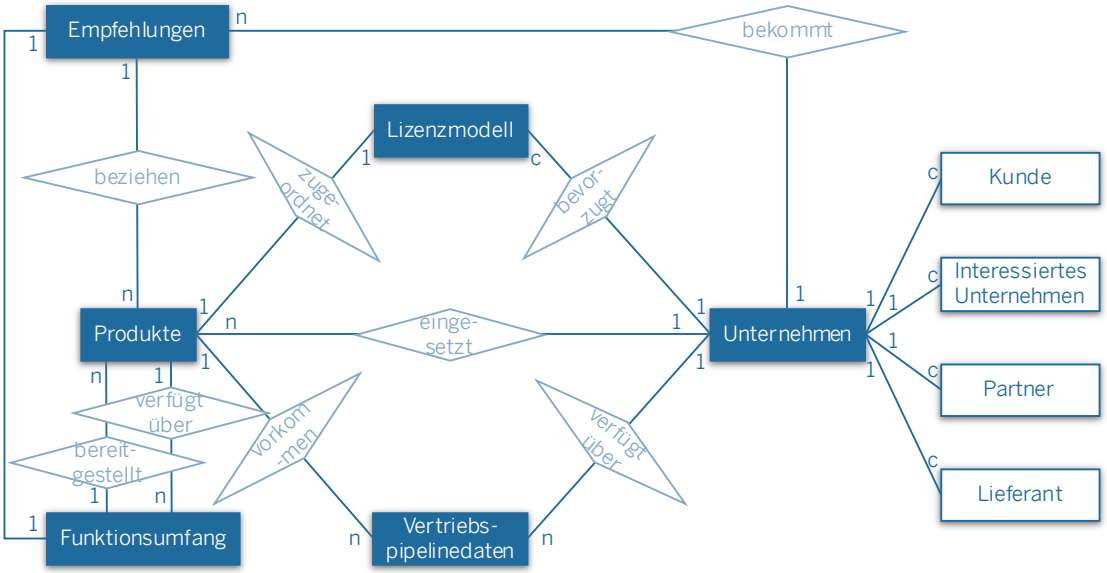


Abbildung 4.3: ERM für die Verbindungen zwischen dem Kontext- und Hintergrunddatensatz (eigene Darstellung)

denen Unternehmen eingesetzt werden kann. Ein Unternehmen weist eine oder keine Präferenz bezüglich eines Lizenzmodells vor und bekommt keine, eine oder mehrere Empfehlungen. Die weiteren Verbindungen zwischen den Tabellen, die in Abbildung 4.3 dargestellt sind, werden bereits beim Betrachten der Kontext- bzw. Hintergrunddaten im Abschnitt 4.1 beschrieben. Auf Grundlage des ERM in Abbildung 4.3 sollen die Tabellen in dem Abschnitt 4.3 miteinander verbunden werden, mit der Intention, die Daten als Grundlage des überwachten Modells zu verwenden. Als Zielvariable Y des zu erstellenden überwachten Modells wird das boolesche Attribut `Gekauft` angesehen. Der Merkmalsvektor bildet sich aus ausgewählten Spalten, die in den ERMs bezüglich den Kontext- und den Hintergrunddaten aufgezeigt sind. Zur Erstellung des markierten Datensatzes als Basis des kontextbasierten Modells, bestehend aus dem Merkmalsvektor und der zugehörigen Zielvariable, müssen die vorliegenden Datensätze im Zuge der Datenvorbereitung vereint werden.

4.3 Datenvorbereitung

Im Ausgangszustand liegen insgesamt sechs alleinstehende Tabellen vor, die zusammengeführt werden müssen, um das kontextbasierte überwachte Modell zur Vorhersage des Kundenkaufverhaltens trainieren sowie aufstellen zu können. Folglich wird sich in der dritten Phase des CRISP-DM Vorgehensmodells mit der Datenzusammenführung und der Aufbereitung der zusammengeführten Daten beschäftigt, mit dem Ziel, eine Datenbasis als Grundlage für das Artefakt vorliegen zu haben. Zur Generierung dieser Datenbasis werden im ersten Schritt die Kontext- sowie Hintergrunddaten untereinander zusammengeführt, um diese im Anschluss zu einer Datenbasis zu verbinden (Abbildung 4.4). Die Datenzusammenführung sowie die Aufbereitung der zusammengeführten Datensätze erfolgt mithilfe von *pandas* sowie der „unified Analytics Engine“[111] *Apache Spark*, welche für die Verarbeitung von Big Data und unter anderem im Bereich des ML Verwendung findet.[112] Der Programmcode für die Vorbereitung und Zusammenführung der Hintergrund- sowie Kontextdatensätze ist im Anhang B.1 vorzufinden.

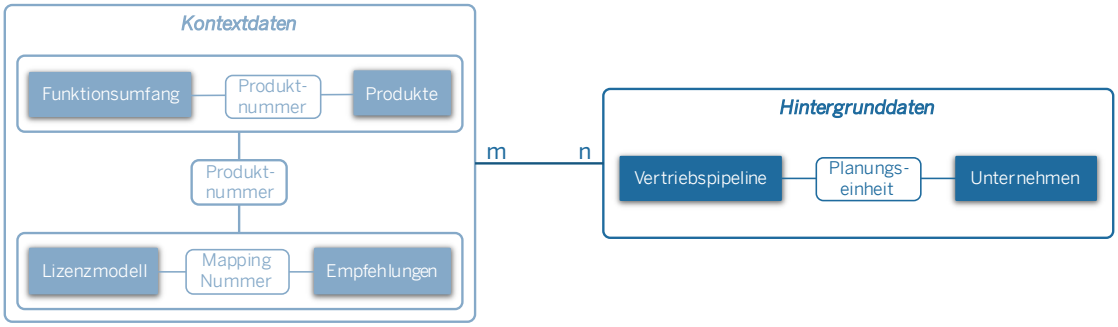


Abbildung 4.4: Konzept der Datenzusammenführung (eigene Darstellung)

Zusammenführung und Aufbereitung der Kontextdaten

Der Kontextdatensatz besteht, wie in Abbildung 4.2 dargestellt, aus vier verschiedenen Tabellen, die in einer Datenbasis vereint werden sollen. Bei der Zusammenführung werden zuerst die Produktdaten mit dem Funktionsumfangsdatensatz über die Spalte

[111] Apache Spark 2018a.
[112] Vgl. Apache Spark 2018a.

`Produktnummer` verbunden und die Lizenzmodell- sowie die Empfehlungsdaten über die `Mapping Nummer`. Die Zusammenführung der zwei resultierenden Datensätze erfolgt über die drei identifizierenden Attribute `Produktnummer`, `Industrie ID` sowie `Funktionsumfang ID`, wodurch der Kontextdatensatz entsteht. Zur Sicherstellung, dass keine Datensätze aufgrund von Duplikaten eine verstärkte Gewichtung in dem zu erstellenden Klassifikationsmodell erhalten, werden alle Duplikate über eine von *pandas* bzw. *Spark* bereitgestellte Funktion gelöscht. Damit nur für die Modellierung und den Anwendungsfall relevante Attribute inkludiert werden, wird im Zuge der Datenaufbereitung eine Attributsselektion bzw. Spaltenreduktion durchgeführt. Infolgedessen werden Spalten, die freien Text enthalten und zu denen ein entsprechender Indikator existiert, gelöscht. Beispielsweise wird die Spalte `Produktname`, welche eindeutig durch die Spalte `Produkt ID` vertreten wird, entnommen. Das Löschen der Spalte `Kaufgrund` ist durch eine mangelnde Aussagekraft für die Aussprache einer Produktempfehlung begründet. Als Resultat der Zusammenführung und Aufbereitung entsteht ein Datensatz mit 2.185 Einträgen und 9 Attributen, was ebenfalls in der Tabelle 4.1 veranschaulicht wird. Der Abbildung 4.5 ist zu entnehmen, dass keine Null-Werte vorliegen und somit im Zuge der Datenbereinigung keine Datenimputation gefordert ist, um fehlende Datensätze zu vervollständigen.[113]

```
2185 Einträge, 0 bis 2184, insgesamt 10 Spalten:
#   Spalte                            Anzahl Nicht-Null  Datentyp
--- ------                            -----------------  --------
0   Produktnummer                     2185 Nicht-Null    Objekt
1   Valide ab Quartal (Produkt)       2185 Nicht-Null    Integer
2   Valide bis Quartal (Produkt)      2185 Nicht-Null    Integer
3   Industrie ID                      2185 Nicht-Null    Objekt
4   Funktionsumfang ID                2185 Nicht-Null    Objekt
5   Unternehmensfunktion ID           2185 Nicht-Null    Objekt
6   Empfohlene Unternehmensfunktion ID 2185 Nicht-Null   Objekt
7   Valide ab Quartal (Empfehlung)    2185 Nicht-Null    Integer
8   Valide bis Quartal (Empfehlung)   2185 Nicht-Null    Integer
9   Lizenzmodell ID                   2185 Nicht-Null    Objekt
```

Abbildung 4.5: Übersicht über die aufbereiteten Kontextdaten (eigene Darstellung)

[113] Vgl. Ströhl und Rockel 2020, S. 5.

Zusammenführung und Aufbereitung der Hintergrunddaten

Die Hintergrunddaten setzen sich aus den Unternehmens- und Vertriebspipelinedaten zusammen, die sich über das Attribut `Planungseinheit` vereinen lassen. Genauso wie bei den Kontextdaten werden bei den Hintergrunddaten alle Duplikate gelöscht, um sicherzustellen, dass alle Datensätze die gleiche Gewichtung zugerechnet bekommen. Durch die Entfernung von Attributen, die für die Modellierung nicht von Relevanz sind, entsteht ein Datensatz mit rund 11 Millionen Einträgen und 15 Spalten. Zur Reduzierung des umfangreichen Datensatzes werden Annahmen definiert, mit dem Ziel, lange Berechnungszeiten zu verhindern und die Komplexität zu minimieren.

Im Folgenden liegt der Fokus auf Unternehmen des Marktsegments „M“. Das Marktsegment klassifiziert ein Unternehmen anhand seiner Größe in „S“, „M“ oder „L“. Aufgrund des persönlichen Kontakts und den engen Kundenbeziehungen zu den Unternehmen des Marktsegments „L“ ist dem Unternehmen SAP SE der Bedarf an Software in diesem Bereich bekannt. Dies hat zur Folge, dass sich die vorliegende Arbeit auf die unbekannten Bereiche fokussiert. Da das Marktsegment „S“ in der Datenbasis nur schwach vertreten ist, wird sich auf das Marktsegment „M“ konzentriert.

Darüber hinaus werden die Produkte auf das Produktlevel 3 beschränkt. Dabei stellen die Produktlevel Kategorien dar, welche hierarchisch aufgebaut sind. Je größer das Produktlevel ist, desto granularer und expliziter wird das Produkt erfasst. Das Produktlevel beschreibt somit die Spezifität eines Produktes. Übertragen in ein Beispiel aus dem Alltag würde die Kategorie Haushaltsware das erste Level umschreiben. Auf dem zweiten Level würde unter anderem von Putzmitteln die Rede sein und auf dem untersten Level versammeln sich die einzelnen Produkte der jeweiligen Marken. Bei der SAP werden Produkte des Levels 2 oder 3 an den Kunden vermarktet. Wenn sich der Kunde für ein Produkt entscheidet, kann dieser zwischen verschiedenen Spezialisierungen wählen, welche ein höheres Produktlevel repräsentieren.

Damit der `Industriecode` sowie der `SAP Master Code` der Hintergrunddaten mit der `Industrie ID` des Kontextdatensatzes in Verbindung gesetzt werden können, wird im Rahmen einer Datenanreicherung eine weitere Tabelle hinzugezogen, welche den `SAP Master Code` auf eine `Industrie ID` abbildet. Enthält eine hinzugefügte `Industrie ID` keinen Wert, wird dem Eintrag im Rahmen der Imputation der Inhalt „I_NO“, also

„keine Industrie“, übergeben. Als Resultat liegt, wie in der Abbildung 4.6 veranschaulicht, ein aufbereiteter sowie reduzierter Hintergrunddatensatz mit 724.433 Einträgen und 12 Spalten vor. Dieser Datensatz wird bei der Modellierung des klassischen zweidimensionalen Klassifikationsmodells eingesetzt.

```
724433 Einträge, 0 bis 724432, insgesamt 12 Spalten:
#   Spalte                  Anzahl Nicht-Null  Datentyp
--- ------                  -----------------  --------
0   Verkaufssegment         724433 Nicht-Null  Objekt
1   Accountklassifikation   724433 Nicht-Null  Objekt
2   Beziehungsstatus        724433 Nicht-Null  Objekt
3   Planungseinheit         724433 Nicht-Null  Objekt
4   Account ID              724433 Nicht-Null  Objekt
5   Pipeline ID             724433 Nicht-Null  Objekt
6   Status                  724433 Nicht-Null  Objekt
7   Statusdatum             724433 Nicht-Null  Objekt
8   Levelcode               724433 Nicht-Null  Objekt
9   Erstellungsdatum        724433 Nicht-Null  Float
10  Gekauft                 724433 Nicht-Null  Boolean
11  Industrie ID            472858 Nicht-Null  Objekt
```

Abbildung 4.6: Übersicht über die aufbereiteten Hintergrunddaten (eigene Darstellung)

Zusammenführung und Aufbereitung des konsolidierten Datensatzes

Während der Hintergrunddatensatz bei dem originären Klassifikationsmodell angewandt wird, wird dem multidimensionalen Empfehlungsmodell der konsolidierte Datensatz der Hintergrund- sowie Kontextdaten übergeben. Aufgrund der Herausforderung, dass zwischen den Hintergrund- und Kontextdaten eine `m:n`-Beziehung besteht, wird individuell für diesen Anwendungsfall eine Mappingtabelle kreiert, welche die Zusammenführung der beiden Tabellen ermöglicht. Die Mappingtabelle verbindet die `Produktnummer` der Kontextdaten mit dem `Levelcode` der Hintergrunddaten. Die Attribute der Mappingtabelle und ihre Beschreibungen sind in der Tabelle A.7 im Anhang vorzufinden. Das Resultat der Zusammenführung ist der Tabelle 4.1 zu entnehmen.

Durch die Zusammenführung der Hintergrund- und Kontextdaten, werden einer `Pipeline ID` mehrere Datensätze zugewiesen. Ein Grund für das mehrfache Vorkommen einer `Pipeline ID` ist, dass einer Empfehlung mehrere Industrien zugeordnet sein können. Die Industrien werden nicht in einem Array gespeichert, sondern für jede Industrie einer Empfehlung wird eine eigene Zeile angelegt. Darüber hinaus kann sich ein Kunde sowohl für die On-Premise- als auch die Cloud-Variante eines Produkts interessieren. Für beide Lizenzmodelle wird wiederum ein Datensatz erstellt. Eine weitere Ursache ist, dass für jede empfohlene Unternehmensfunktion eines Kunden genauso wie bei der Industrie eine Zeile angelegt wird. Das Gleiche gilt an dieser Stelle für die Funktionen sowie die Unternehmensfunktionen. Zudem können einer Pipeline mehrere Produkte zugeordnet sein, wodurch für jedes Produkt eine Zeile eingefügt wird. Diese Ursachen haben zur Folge, dass sich eine Kundenpipeline aus mehreren Datensätzen zusammensetzen kann. Damit genau eine Zeile eine Kundenpipeline repräsentiert, werden die zusammengehörenden Zeilen in einer Zeile zusammengefasst. Um dies zu erreichen, werden die empfohlenen Unternehmensfunktionen, die Unternehmensfunktionen, die Funktionsumfänge sowie die Produkte einer Vertriebspipeline, für die sich ein Kunde interessiert, in jeweils einer Spalte des Typs Array gespeichert. Die verschiedenen Industrien sowie die Lizenzmodelle erhalten jeweils eine eigene Spalte. Ist eine Empfehlung einer gewissen Industrie bzw. ein Kunde einem gewissen Lizenzmodell zugeordnet, erhält der Datensatz in der entsprechenden Spalte eine „1“, andernfalls wird eine „0“ eingetragen. Zur anschließenden Reduzierung des umfangreichen Datensatzes wird dieser mittels einer bereitgestellten Sampling-Methode von *Spark* und durch die Entfernung von Duplikaten reduziert. Der Programmcode für die Aufbereitung des konsolidierten Datensatzes ist dem Anhang B.2 zu entnehmen.

Dem konsolidierten Datensatz werden im Rahmen der Datenanreicherung weitere Kontextdaten hinzugefügt. Bei den in dem Abschnitt 4.1 definierten Kontextdaten handelt es sich zum Teil um abgeleitete Metriken, die aus den vorhandenen Kontext- sowie Hintergrunddaten berechnet werden. Dazu zählen die Lizenzmodelltendenz, die Anzahl der empfohlenen Unternehmensfunktionen, die Anzahl der Produkte, die Verweildauer eines Kunden in einer Vertriebspipeline, die durchschnittliche Verweildauer eines Kunden in einer Vertriebspipeline, die durchschnittliche Verweildauer eines Produktes in einer Vertriebspipeline und die Summe der gekauften Lösungen im Verhältnis zu den

betrachteten Lösungen. Zum Beispiel die Lizenzmodelltendenz vergleicht die Summe der gekauften On-Premise- sowie Cloud-Produkte eines Kunden. Das Lizenzmodell mit der größeren Summe stellt die präferierte Lösung dar. Der Code für die Berechnung der abgeleiteten Kontextdaten kann dem Anhang B.3 entnommen werden. Die Tabelle 4.1 veranschaulicht die jeweiligen zusammengeführten Datensätze und ihre Eigenschaften.

Name	**Beinhaltende Datensätze**	**Spaltenanzahl**	**Zeilenanzahl**
Kontextdaten	Produkte, Funktionsumfang, Empfehlungen, Lizenzmodelle	10	2.185
Hintergrunddaten	Vertriebspipeline, Unternehmen	12	724.433
Unaufbereiteter, konsolidierter Datensatz	Kontextdaten, Hintergrunddaten	21	116.643.664
Aufbereiteter, konsolidierter sowie reduzierter Datensatz	Kontextdaten, Hintergrunddaten	55	20.684

Tabelle 4.1: Zusammenfassende Übersicht der erstellten Datensätze (eigene Darstellung)

4.4 Modellierung

Die im Zuge der Datenvorbereitung zusammengeführten Datensätze werden in der vierten Phase des CRISP-DM Vorgehensmodells für die Modellierung des kontextbasierten Empfehlungssystems, welches mithilfe der *Technologie* des Random Forests erstellt wird, eingesetzt. Für die Modellierung eines überwachten Modells stehen neben dem Random Forest weitere Algorithmen, wie beispielsweise der *Nearest Neighbor Algorithmus* oder der *Support Vector Machine Algorithmus*, zur Verfügung. Aufgrund der Existenz eines zweidimensionalen Empfehlungssystems innerhalb der SAP SE, welches anhand des Random Forests Produktvorschläge für einen Kunden liefert, wird diese

Technologie für den vorliegenden Anwendungsfall gewählt. Dieses Empfehlungssystem soll unter Hinzunahme von Kontextdaten erweitert werden. Es gilt die Annahme, dass die Wahl des passenden Algorithmus für die Modellierung bereits bei der Erstellung des existierenden Modells untersucht wurde. Der Fokus der Arbeit liegt auf der Überprüfung, ob Empfehlungssysteme durch zusätzliche hinzuzunehmende Kontextdaten präzisere Produktvorschläge für einen Kunden liefern. Der Programmcode für die im Folgenden beschriebene Modellierung des Empfehlungssystems ist dem Anhang B.4 zu entnehmen.

Das kontextbasierte Empfehlungssystem, welches im Zuge der Modellierung entwickelt wird, stellt das *Artefakt* dar. Durch die Definition eines kontextbasierten Empfehlungssystems in dem Abschnitt 2.3 wird das Wissen erlangt, wie das zu erstellende Artefakt aufgebaut sein muss. Wie im Abschnitt 4.2 beschrieben wird die Zielvariable des binären Klassifikationsmodells aus den Klassen „kauft“ sowie „kauft nicht“ gebildet. Der Merkmalsvektor des kontextbasierten Empfehlungsmodells wird durch die Attribute Accountklassifikation, Beziehungsstatus, Industrie ID des Kunden, Marktsegment, Planungseinheit, Statusdatum der Vertriebspipeline, Erstellungsdatum der Vertriebspipeline, Lizenzmodell, Industrie IDs der Empfehlung und die in dem Abschnitt 4.3 berechneten abgeleiteten Attribute gebildet. Der Merkmalsvektor des originären Empfehlungssystems fasst sich aus den Attributen Accountklassifikation, Beziehungsstatus, Industrie ID des Kunden, Marktsegment, Planungseinheit, Statusdatum und Erstellungsdatum der Vertriebspipeline zusammen. Dabei ist zu beachten, dass identifizierende Attribute, wie die Account ID oder die Pipeline ID, und der Status der Pipeline, welcher Aufschluss über den Kauf eines Produktes gibt, den Merkmalsvektoren nicht hinzugefügt wird. Aus den Merkmalsvektoren sowie den Zielvariablen entstehen markierte Datensätze der Form $\{(Merkmalsvektor_i, Gekauft_i)\}_{i=1}^{N}$. Bevor die Datensätze in Trainings-, Test- sowie Validierungsdaten aufgeteilt und die Modellierung initiiert werden kann, bedarf es einer für die Modellierung entsprechende Anpassung der vorliegenden Datensätze.

Aufbereitung der Daten für die Modellierung

Damit der konsolidierte Datensatz für die Modellierung des kontextbasierten Empfehlungssystems und der Hintergrunddatensatz für die Modellierung des zweidimensionalen Empfehlungsmodells angewandt werden können, müssen diese entsprechend der Anforderungen des anzuwendenden Random Forest Algorithmus angepasst werden. Die Anpassung der Daten an die Anforderungen der eingesetzten Technik werden gemäß des CRISP-DM in einem iterativen Prozess wahrgenommen und umgesetzt, wodurch das Artefakt an *Exaktheit* erlangt und verfeinert wird.

Zur Anwendung des ausgewählten *Random Forest* Lernalgorithmus müssen die Imbalancen der Zielvariable Y ausgeglichen werden. Abbildung 4.7 veranschaulicht die Imbalancen innerhalb der Datensätze. Die verstärkt vertretene Klasse „kauft nicht“ wird anhand der Methode des *Downsamplings* und mithilfe der von *Spark* bereitgestellten Sampling-Methode reduziert, sodass beide Klassen die gleiche Anzahl an Einträgen vorweisen. Zudem wird die Zielvariable Y, welche in einer booleschen Variable

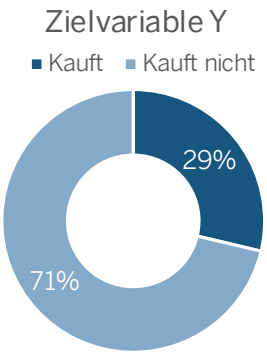


Abbildung 4.7: Imbalancen der Datensätze (eigene Darstellung)

gespeichert ist, in eine binäre Variable des Datentyps Integer umgewandelt. Dabei repräsentiert 1 die Klasse „kauft“ und 0 die Klasse „kauft nicht“. Um die kategorischen Werte des Merkmalsvektors dem Random Forest übergeben zu können, werden diese mittels des *One-Hot Encodings* in binäre Vektoren konvertiert. Dies hat zur Folge, dass die kategorischen Daten in maschinenlesbare Inhalte umgewandelt werden. Zu den kategorischen Merkmalen zählen in den vorliegenden Daten die Spalten der Typen String und Objekt. Darüber hinaus darf der Datensatz keine Null-Werte beinhalten und die Spalten müssen mindestens zwei verschiedene Inhaltswerte vorweisen. Die Her-

ausforderung der Null-Werte wird bereits im Rahmen der Datenvorbereitung in dem Abschnitt 4.3 behoben. Durch die Reduktion des Datensatzes, ist es unter anderem möglich, dass manche Spalten keine unterschiedlichen Inhaltswerte enthalten. Die entsprechende Anwendung einer Gruppierungsfunktion zeigt, dass diese Annahme für die Spalten `Valide ab Quartal` und `Valide bis Quartal` der Produkttabelle zutrifft, woraufhin diese entnommen werden. Folglich erfüllen die Datensätze alle zuvor definierten Anforderungen für die Anwendung des Random Forests. Die finalen Datensätze können somit für die Modellierung angewandt werden.

Training, Testen und Validieren der Modelle

Für die Erstellung der beiden überwachten Modelle werden die Datensätze in einen Trainings-, Test- und Validierungsdatensatz unterteilt. Die Modelle werden auf Basis der Trainingsdaten erstellt. Mithilfe der Test- und Validierungsdaten kann die Richtigkeit der erstellten Modelle überprüft werden. Zur Vergleichbarkeit des originären Empfehlungssystems basierend auf den Hintergrunddaten und des kontextbasierten Empfehlungssystems basierend auf den Hintergrund- sowie Kontextdaten, werden in beiden Empfehlungssystemen die gleichen Trainings-, Test- und Validierungsdatensätze eingesetzt. Der Unterschied an dieser Stelle ist, dass die Spalten der Kontextdaten bei dem originären Empfehlungsmodell nicht enthalten sind, sondern nur die Spalten der Hintergrunddaten.

Die Unterteilung der vorliegenden Datensätze erfolgt angelehnt an die Zeitspanne. Die Trainings- sowie Testdatensätze umfassen das Jahr 2018, während sich die Validierungsdatensätze auf das Jahr 2019 beschränken. Es wird eine weitere Annahme getroffen, dass nur diejenigen Produkte Betrachtung finden, die innerhalb der zuvor genannten Zeitspannen von jeweils einem Jahr gekauft werden. Dadurch erhalten nur Produkte die Klassifizierung „gekauft“, wenn das Unternehmen diese innerhalb des betrachteten Kalenderjahrs erworben hat. Die Unterteilung in Trainings- und Testdatensätze, welche das Jahr 2018 betrachten, erfolgt durch eine von *Spark* bereitgestellte Sampling-Methode. Die Trainingsdaten nehmen 70 % der entsprechenden Datensätze des Jahres 2018 an und die Testdaten enthalten die übrigen 30 % der Daten.

Zum Trainieren der Modelle wird der Random Forest Algorithmus im ersten Durch-

lauf auf den markierten Trainingsdatensätzen durchgeführt. Als Resultat der Trainings entstehen zwei Modelle, eins für die Hintergrunddaten und das andere für die konsolidierten Daten. Die Modelle schreiben jedem Datensatz Wahrscheinlichkeiten für die beiden Klassen „kauft“ und „kauft nicht“ zu. Diejenige Klasse, die eine Wahrscheinlichkeit von über 50 % annimmt, wird als vorhergesagte Klasse angegeben. Der Grenzwert liegt somit bei 50 %. Wird beispielsweise einem fiktiven Kunden A für die Klasse „kauft“ eine Wahrscheinlichkeit von 46,75 % zugeschrieben und der Klasse „kauft nicht“ eine Wahrscheinlichkeit von 53,25 %, ordnet das Modell den Datensatz der stärkeren Klasse „kauft nicht“ zu. Die Tabelle 4.2 zeigt eine beispielhafte Ausgabe des Trainings. Der erste Arrayeintrag der Spalte „Wahrscheinlichkeit“ ist der Klasse

Index	Gekauft	Wahrscheinlichkeit	Vorhersage
0	0	[0.5324786, 0.4675214]	1
1	1	[0.2100892, 0.7899108]	1
2	1	[0.3690015, 0.6309985]	1
3	0	[0.8796327, 0.1203673]	0

Tabelle 4.2: Beispielhafte Ausgabe eines Trainings (eigene Darstellung)

„kauft nicht (0)“ zuzuordnen und der zweite Eintrag der Klasse „kauft (1)“. Dem zweiten, dritten und vierten Datensatz werden die richtigen Klassen zugeordnet. Hingegen bei dem ersten Eintrag wird die Klasse „kauft“ vorhergesagt, obwohl dieser der Klasse „kauft nicht“ angehört. Grundsätzlich handelt es sich bei der vorhergesagten Klasse um einen *diskreten* Ausgabewert, während die Zuschreibung einer Wahrscheinlichkeit zu den Klassen eine *stetige* Ausgabe darstellt. Die trainierten Random Forest Modelle weisen nun jedem Datensatz aufgrund seiner Hintergrunddaten bzw. konsolidierten Daten eine Vorhersage zu.

Zur Überprüfung der erstellten Modelle werden den Modellen die entsprechenden unmarkierten Testdatensätze übergeben. Um einen unmarkierten Testdatensatz zu kreieren, wird die Spalte der Zielvariable `Gekauft` aus den Testdaten entfernt. Das fertige Modell soll bei einer tatsächlichen Anwendung auf Basis von Merkmalen den unbekannten Ausgabewert, ob ein Unternehmen ein Produkt kaufen wird oder nicht, zurückgeben. Durch das Entfernen der Zielvariable wird demnach ein fiktives Echtszenario

inszeniert. Die Modelle weisen in der Testphase den übergebenen Merkmalsvektoren eine Klasse als Ausgabe zu. Eine beispielhafte Ausgabe der Testphase wird in der Tabelle 4.3 veranschaulicht. Der Tabelle ist zu entnehmen, dass die Zielvariable `Gekauft` nicht vorliegt. Damit die Richtigkeit der Vorhersagen der Modelle überprüft werden kann, muss die Zielvariable `Gekauft` wieder hinzugezogen werden.

Index	Wahrscheinlichkeit	Vorhersage
0	[0.6789542, 0.3210458]	0
1	[0.1584632, 0.8415368]	1
2	[0.5666894, 0.4333106]	0
3	[0.9875123, 0.0124877]	0

Tabelle 4.3: Beispielhafte Ausgabe eines Testdurchlaufs

Zur Sicherstellung, dass die erstellten Modelle nicht einer *Überanpassung* unterliegen, werden die Validierungsdatensätze herangezogen. Hierbei gleicht das Vorgehen der Testphase. Den Validierungsdatensätzen wird die Zielvariable `Gekauft` entnommen, mit dem Ziel, unmarkierte Datensätze zu generieren. Diese werden den entsprechenden Modellen übergeben, um auf Basis der Merkmalsvektoren Vorhersagen zu erstellen. Zur Validierung der Vorhersagen wird die Zielvariable den unmarkierten Datensätzen wieder hinzugefügt. Anhand ausgewählter Metriken werden die beiden in der Trainingsphase erstellten Modelle anhand der Vorhersagen der Test- sowie Validierungsphase im folgenden Abschnitt evaluiert und verglichen.

4.5 Evaluation der Modelle

Nach Abschluss des Trainierens, Testens und Validierens des kontextbasierten sowie klassischen Empfehlungssystems stellt sich die Frage, ob die zu Anfang aufgestellte Hypothese, dass Empfehlungssysteme in der Enterprise-Softwareindustrie beim Unternehmensbeispiel SAP SE unter Hinzunahme von Kontext präzisere Vorhersagen liefern, negiert oder verifiziert werden kann. Um die Präzision des kontextbasierten Empfehlungssystems einzustufen, werden das kontextbasierte sowie klassische Empfehlungs-

system in der fünften Phase des CRISP-DM Vorgehensmodells anhand ausgewählter Metriken verglichen. Zu diesen Metriken zählt die Treffergenauigkeit, die Spezifität, die FPR, die Trefferquote, die Genauigkeit, der F_1-Score, die ROC-Kurve, die AUROC und die Wahrscheinlichkeitsdichtefunktion. Der Programmcode für die Evaluation der Modelle ist dem Anhang B.4 zu entnehmen.

Auswertung der Ergebnisse im Vergleich

Bei der Auswertung der Ergebnisse werden sowohl die getesteten als auch validierten Modelle des kontextbasierten sowie klassischen Empfehlungssystems in Vergleich gesetzt. Zur Berechnung der aufgelisteten Metriken werden die jeweiligen Konfusionsmatrizen benötigt, welche in Abbildung 4.8 vorzufinden sind. Auf Basis dieser Matrizen

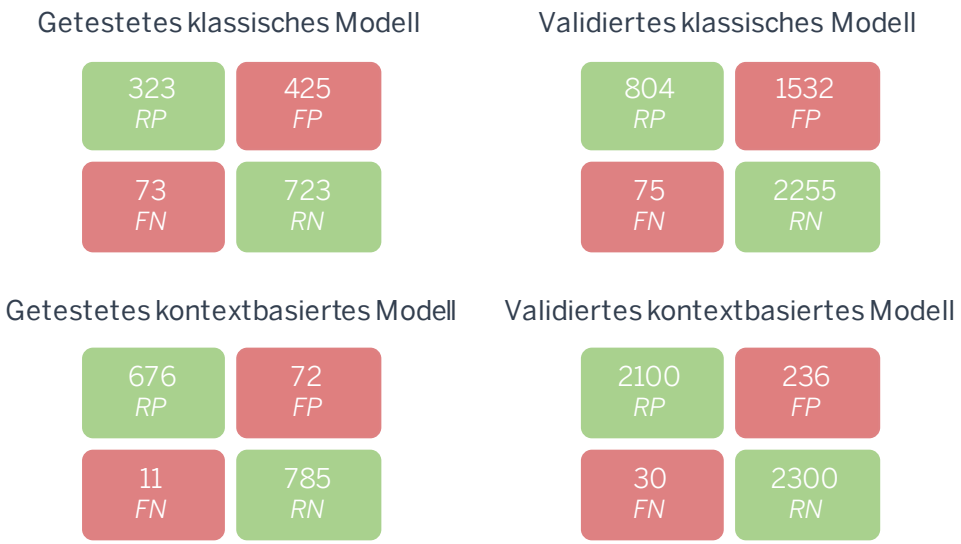


Abbildung 4.8: Konfusionsmatrizen der Test- und Validierungsdatensätze (eigene Darstellung)

werden die Metriken nach den definierten Formeln des Abschnitts 2.1 berechnet. Zur Veranschaulichung werden die Metriken in der Abbildung 4.9 zusammengetragen. Zu beachten ist, dass in der Abbildung nur ebenjene Metriken aufgelistet sind, die ein Optimum von 1 vorweisen. Demnach wird die FPR nicht angezeigt. Bei dem F_1-Score ist zu erwähnen, dass der ausgeglichene F_1-Score gewählt wird. Die Gründe zur Wahl des ausgeglichenen F_1-Scores werden im Folgenden aufgelistet. Einerseits sollen den

Kunden keine falschen Empfehlungen (FP) vorgeschlagen werden, um sicherzustellen, dass den Kunden eine wohldefinierte kleine Menge von Produkten vorgeschlagen wird, die tatsächlich zu dem Profil der Kunden passt. Durch die Verhinderung der Ausgabe von FP, werden den Kunden andererseits passende Produkte vorenthalten, da diese als FN eingestuft werden. Aufgrund der Tatsache, dass den Unternehmen weder passende Produkte vorenthalten noch falsche Produkte vorgeschlagen werden sollten, wird der balancierte F_1-Score angewandt. Demnach wird ein Grenzwert von 0,5 gewählt. Ebenjene Klasse, die eine Wahrscheinlichkeit von über 50 % vorweist, wird als die vorhergesagte Klasse angegeben. Abbildung 4.9 zeigt, dass im Allgemeinen die Ergebnisse

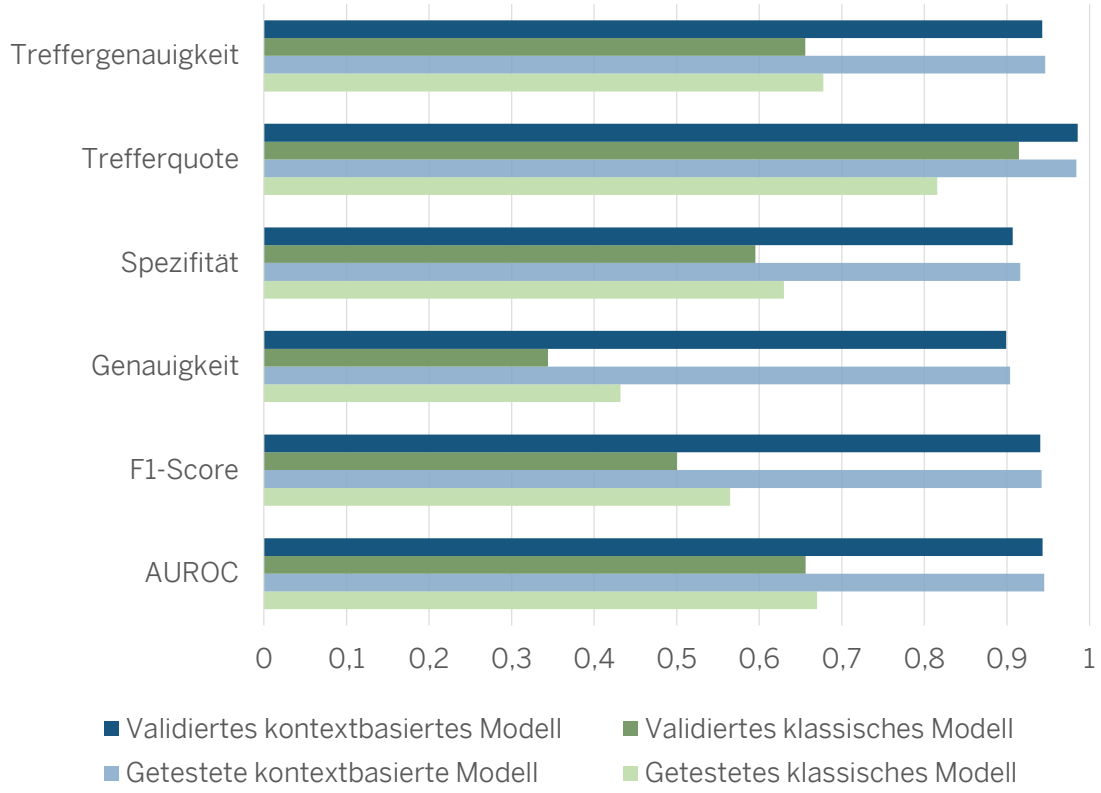


Abbildung 4.9: Evaluation der Klassifikationsmodelle anhand ausgewählter Metriken (eigene Darstellung)

des kontextbasierten Empfehlungssystems sowohl bei dem Testdurchlauf als auch bei der Validierung dem Optimum von 1 näherkommen als die des klassischen. Die Metriken des kontextbasierten Empfehlungssystems halten sich in dem Bereich zwischen 0,89897

und 0,98592 auf. Währenddessen liegen die Metriken des klassischen Empfehlungssystems in dem weitaus breiteren Intervall von 0,34418 bis 0,91468. Dazu kommt, dass der Test- und Validierungsdurchlauf des kontextbasierten Modells im Vergleich zu dem klassischen Modell keine starken Abweichungen bei den Metriken vorweisen. Zudem ist zu erkennen, dass die Testdurchläufe beider Modelle bei allen Metriken, abgesehen von der Trefferquote, bessere Ergebnisse liefern. Folglich können die entwickelten Modelle nicht als überangepasst bezeichnet werden. Aufgrund dessen, dass die Modelle beim Validierungsdurchlauf ähnliche Metriken liefern wie beim Testdurchlauf, ist sichergestellt, dass die Modelle nicht nur auf den Trainings- und Testdatensatz angepasst sind und auch bei einem losgelösten Datensatz Performanz liefern. Werden die Werte detaillierter betrachtet wird deutlich, dass das kontextbasierte Modell folgende Metriken liefert:

- 94,6 % aller Datensätze des Testdurchlaufs und 94,3 % aller Datensätze der Validierung werden in die richtige Klasse gruppiert (Treffergenauigkeit).
- 98,4 % der tatsächlich positiven Datensätze des Testdurchlaufs und 98,6 % der tatsächlich positiven Datensätze der Validierung werden als solche klassifiziert (Trefferquote).
- 91,6 % der tatsächlich negativen Datensätze des Testdurchlaufs und 90,7 % der tatsächlich negativen Datensätze der Validierung werden als solche eingestuft (Spezifität).
- 8,4 % der negativen Datensätze des Testdurchlaufs und 9,3 % der negativen Datensätze der Validierung werden falsch klassifiziert (FPR).
- 90,4 % der als richtig eingestuften Datensätze des Testdurchlaufs und 89,9 % der als richtig eingestuften Datensätze der Validierung gehören tatsächlich dieser Klasse an (Genauigkeit).
- Aus der Trefferquote von 0,984 des Testdurchlaufs bzw. 0,986 der Validierung und der Genauigkeit von 0,904 des Testdurchlaufs bzw. 0,899 der Validierung resultiert der F_1-Score von 0,942 des Testdurchlaufs bzw. 0,94 der Validierung.

Gleichermaßen werden bei den klassischen Modellen die folgenden ausformulierten Metriken berechnet:

- 67,7 % aller Datensätze des Testdurchlaufs und 65,6 % aller Datensätze der Validierung werden der richtigen Klasse zugewiesen (Treffergenauigkeit).
- 81,6 % der tatsächlich positiven Datensätze des Testdurchlaufs und 91,5 % der tatsächlich positiven Datensätze der Validierung werden als solche eingestuft (Trefferquote).
- 63 % der tatsächlich negativen Datensätze des Testdurchlaufs und 59,5 % der tatsächlich negativen Datensätze der Validierung werden als solche klassifiziert (Spezifität).
- 37 % der negativen Datensätze des Testdurchlaufs und 40,5 % der negativen Datensätze der Validierung werden falsch zugeordnet (FPR).
- 43,2 % der als richtig eingestuften Datensätze des Testdurchlaufs und 34,4 % der als richtig eingestuften Datensätze der Validierung gehören tatsächlich dieser Klasse an (Genauigkeit).
- Aus der Trefferquote von 0,816 des Testdurchlaufs bzw. 0,915 der Validierung und der Genauigkeit von 0,432 des Testdurchlaufs bzw. 0,344 der Validierung resultiert der F_1-Score von 0,565 des Testdurchlaufs bzw. 0,5 der Validierung.

Die in der Abbildung 4.9 vernachlässigte FPR wird im Rahmen der ROC-Kurve in der Abbildung 4.10 miteinbezogen. Es wird ersichtlich, dass das kontextbasierte Modell sowohl bei dem Test- als auch bei dem Validierungsdurchlauf eine bessere ROC-Kurve vorweist als das klassische Modell. Dies wird ebenfalls durch die um circa 42,4 % höhere AUROC des kontextbasierten Empfehlungssystems unterstrichen. Die AUROC sagt in diesem Fall aus, wie präzise ein Modell zwischen den zwei Klassen „gekauft“ und „nicht gekauft“ unterscheidet. Folglich weist das kontextbasierte Modell die Datensätze eindeutiger den zwei Klassen zu als das klassische Modell. Darüber hinaus zeigt Abbildung 4.10, dass die ROC-Kurven des Test- sowie Validierungsdurchlaufs des kontextbasierten Modells nahezu identisch sind. Die Ebene unter der ROC-Kurve unterscheidet sich lediglich um 0,02. Dementgegen weist das klassische Modell einen Unterschied von 0,05 auf. Durch die im Allgemeinen geringen Unterschiede zwischen den AUROCs der Test- sowie Validierungsdurchläufe wird deutlich, dass weder das kontextbasierte noch das klassische Modell einer Überanpassung unterliegen.

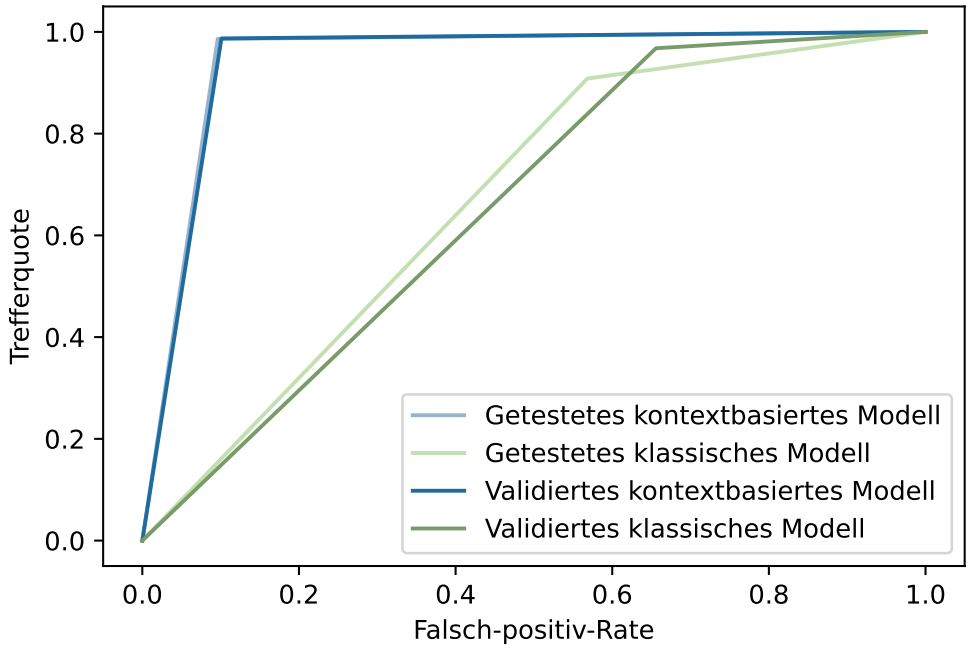


Abbildung 4.10: Vergleich der ROC-Kurven des klassischen und kontextbasierten Modells (eigene Darstellung)

Nach der Betrachtung der ausgewählten Metriken werden in der Abbildung 4.11 die Wahrscheinlichkeitsdichtefunktionen der jeweiligen Modelle beleuchtet. Hierbei repräsentiert die x-Achse die Wahrscheinlichkeit der Klassenzuordnung und die y-Achse die relative Häufigkeit des Vorkommens der Wahrscheinlichkeit der Klassenzuordnung. Die Datensätze mit einer Wahrscheinlichkeit unterhalb des Grenzwertes von $x = 0,5$ erhalten die Klassifizierung „gekauft“ und die Datensätze mit einer Wahrscheinlichkeit von über 0,5 werden der Klasse „nicht gekauft“ zugeordnet. Die angezeigten Funktionen erfüllen die Charakteristika einer Dichtefunktion, welche in dem Abschnitt 2.1 aufgelistet sind. Mithilfe einer von *seaborn* bereitgestellten Funktion zum Aufbau einer Dichtefunktion werden die in Abbildung 4.11 gezeichneten Funktionen auf Basis aller verfügbaren Datensätze erstellt. *Seaborn* ist eine Bibliothek der Programmiersprache Python zur Visualisierung von Daten.[114] Die von *seaborn* bereitgestellte Funktion stellt durch eine Normalisierung sicher, dass das Integral zwischen allen möglichen Werten

[114] Vgl. Waskom 2020a.

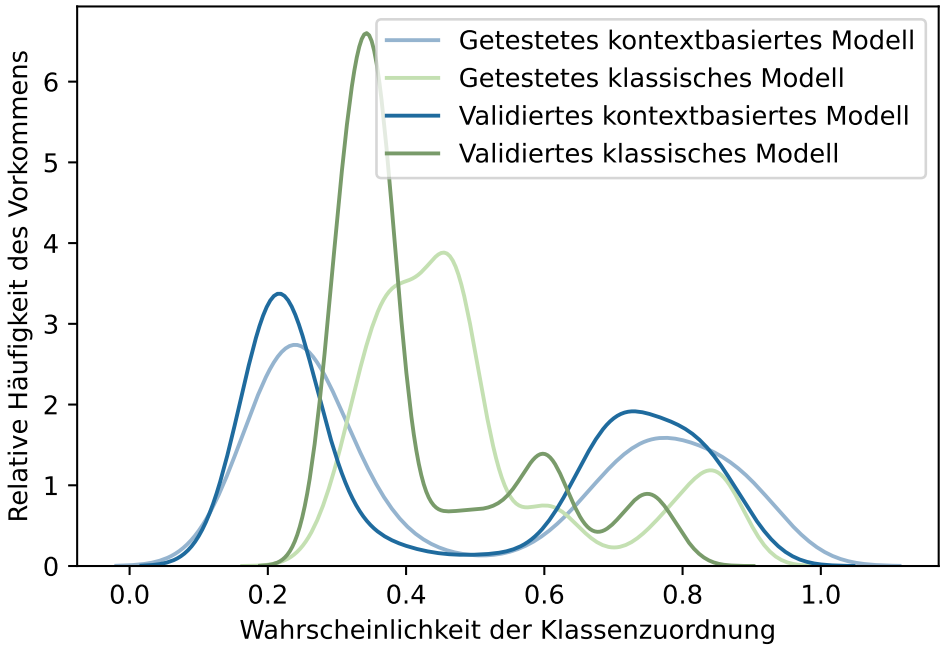


Abbildung 4.11: Vergleich der Wahrscheinlichkeitsdichtefunktionen des klassischen und kontextbasierten Modells (eigene Darstellung)

1 ist.[115] Dadurch, dass die Dichtefunktionen keine negativen Funktionswerte annehmen, erfüllen die in Abbildung 4.11 aufgezeichneten Funktionen alle Eigenschaften einer Wahrscheinlichkeitsdichtefunktion. Aus den Graphen geht hervor, dass beide Modelle eine leichte umgekehrte Normalform annehmen. Übertragen in den vorliegenden Kontext bedeutet dies, dass je weiter die beiden Hochpunkte der umgekehrten Normalform voneinander entfernt sind, desto eindeutiger können die Datensätze einer Klasse zugeordnet werden. Ein Hochpunkt signalisiert, dass die zugehörige Wahrscheinlichkeit häufiger vorkommt. Beispielsweise bei einer Wahrscheinlichkeit von 0,464 wird der Datensatz zwar der Klasse „kauft“ zugewiesen, jedoch könnte es sich bei diesem auch um ein FP handeln. Die Wahrscheinlichkeit von 0,464 liegt in der Nähe des Grenzwerts, wodurch der Datensatz der falschen Klasse zugeordnet sein könnte. Eine Wahrscheinlichkeit von 0,108 hingegen zeigt deutlich, dass der Datensatz mit großer Wahrscheinlichkeit als „gekauft“ klassifiziert werden kann. Beim Betrachten der Brei-

[115] Vgl. Waskom 2020b.

te der Funktionen, also der Entfernung zwischen den beiden Maxima, wird ersichtlich, dass die kontextbasierten Modelle eine breitere Funktion vorweisen. Die Funktionen der kontextbasierten Modelle liegen in dem Intervall zwischen 0 und 1, während die Funktionen des klassischen Modells sich in dem Bereich von 0,3 bis 0,9 bzw. 1,0 aufhalten. Folglich kann das kontextbasierte Modell die Datensätze eindeutiger einer Klasse zuordnen als das klassische Modell. Des Weiteren ist zu erwähnen, dass das klassische Modell verhältnismäßig viele Datensätze der Klasse „kauft“ zuordnet. Dies wird durch den Hochpunkt bei dem x-Wert von circa 0,39 und y-Wert von ungefähr 6,7 deutlich, welcher nicht im Bereich von über 0,5 zu entdecken ist. Diese Imbalance wird durch die Asymmetrie der klassischen Funktionen verdeutlicht. In minimierter Form ist diese Asymmetrie ebenfalls bei den kontextbasierten Modellen zu erkennen.

Verifikation oder Negation des definierten Kontextes

Die Auswertung der Ergebnisse des kontextbasierten sowie des klassischen Modells zeigt eindeutig, dass das kontextbasierte Modell präzisere Vorhersagen und in diesem Fall präzisere Klassenzuordnungen liefert. Anhand der ausgewählten Metriken der Treffergenauigkeit, der Spezifität, der FPR, der Trefferquote, der Genauigkeit, des F_1-Scores und der AUROC wird bestätigt, dass das kontextbasierte Modell ausnahmslos bessere Ergebnisse liefert als das klassische Empfehlungssystem. Die Ergebnisse des kontextbasierten Modells halten sich in der Nähe des Optimums im Intervall zwischen 0,89897 und 0,98592 auf. Demgegenüber deckt das klassische Modell einen weiteren Bereich von 0,5705 ab. Die ROC-Kurve sowie die AUROC heben hervor, dass das kontextbasierte Empfehlungssystem präziser zwischen den Klassen „kauft“ und „kauft nicht“ unterscheiden kann. Darüber hinaus stechen beim Betrachten der Wahrscheinlichkeitsdichtefunktion die breiteren Funktionen des kontextbasierten Empfehlungssystems im Vergleich zu dem klassischen Modell hervor, die die Eindeutigkeit der Klassenzuordnung des kontextbasierten Modells unterstreichen. Aufgrund der Tatsache, dass die Hochpunkte der Wahrscheinlichkeitsdichtefunktionen des kontextbasierten Modells näher an der 0 bzw. 1 angegliedert sind, kann das Empfehlungssystem die Datensätze eindeutiger einer Klasse zuordnen. Darüber hinaus unterliegen weder das klassische noch das kontextbasierte Empfehlungssystem einer Überanpassung. Dies wird durch die ähnlichen Metriken und den vergleichbaren Verlauf der ROC-Kurven des Test- sowie Validierungs-

durchlaufs verdeutlicht.

Zur Validierung der Kontextdaten werden die Einflüsse der in dem Abschnitt 4.1 beschriebenen Attribute auf das Modell analysiert. Es ist zu erkennen, dass das Lizenzmodell, die Lizenzmodelltendenz, die Anzahl der empfohlenen Unternehmensfunktionen, die Anzahl der Produkte, die Verweildauer eines Kunden in einer Vertriebspipeline, die durchschnittliche Verweildauer eines Kunden in einer Vertriebspipeline, die durchschnittliche Verweildauer eines Kunden in einer Vertriebspipeline und die Summe der gekauften Lösungen im Verhältnis zu den betrachteten Lösungen das Modell nicht beeinflusst. Lediglich die Industrie der Empfehlung hat eine Auswirkung auf die Produktvorschläge. Infolgedessen kann der Kontext wie folgt reduziert werden: $Kontext \subseteq Industrie$. Das kontextbasierte Empfehlungssystem setzt sich somit aus den drei Dimensionen Vertriebspipelinedaten, Unternehmen und Industrie zusammen: $f : Unternehmen \times Vertriebspipelinedaten \times Industrie \rightarrow N$.

Folglich kann der in dem Abschnitt 4.1 definierte Kontext teilweise sowie die Hypothese, dass mithilfe von Kontext im Zusammenhang mit Empfehlungen die Präferenzen sowie Interessen der Kunden der SAP besser abgebildet werden, verifiziert werden. Als beispielhaftes Unternehmen der Enterprise-Softwareindustrie beweist das Empfehlungssystem zur Vorhersage von Produktempfehlungen für Kunden der SAP SE, dass Kontextdaten dazu verhelfen, das Profil eines Kunden konkreter abzubilden und daraus resultierend präzisere Produktempfehlungen zu liefern. Das Empfehlungssystem schlägt dem Kunden die zu ihm passenden Produkte vor, wodurch dem Kunden eine wohldefinierte kleinere Menge an Produkten angezeigt werden kann. Trotz der impliziten Erhebung der Kundendaten durch das passive Beobachten des Kunden, ist kein *natürliches Rauschen* in der Datenbasis vorzufinden, welches die Vorhersagen negativ beeinflusst hätte. Dies beweisen die im Vergleich zu dem klassischen Empfehlungssystem präziseren Produktempfehlungen des kontextbasierten Empfehlungssystems. Durch die Verifizierung der Hypothese wird das kontextbasierte Empfehlungssystem als Resultat der fünften Phase des CRISP-DM in der Zukunft weiterentwickelt und präzisiert werden, mit dem Ziel, eine konkrete Umsetzung des kontextbasierten Empfehlungssystems innerhalb der SAP SE zu erzielen.

5 Fazit

5.1 Zusammenfassung

Die aufgestellte Hypothese, dass unter Hinzunahme von Kontextinformationen Empfehlungssysteme in der Enterprise-Softwareindustrie präzisere Vorschläge liefern, wurde im Rahmen der Arbeit verifiziert.

Das in diesem Zuge betrachtete kontextbasierte Empfehlungssystem der Enterprise-Softwareindustrie schlägt Unternehmen zu ihm passende Produkte vor. Aufgrund der präzise auf den Kunden abgestimmten Produktempfehlungen soll der Kunde zu der Aktion motiviert werden, das empfohlene Produkt zu kaufen. Zur Evaluation, ob die Hinzunahme von Kontext dazu verhelfen kann, das Kundenprofil optimaler abzubilden, wurde beispielhaft für das Unternehmen SAP SE ein kontextbasiertes Empfehlungssystem entwickelt. Das Empfehlungssystem wurde mithilfe des überwachten maschinellen Lernens und konkret des Random Forest Algorithmus erstellt. Auf Basis von Kaufhistorien, Kunden- sowie Kontextdaten liefert das kontextbasierte Empfehlungssystem Produktvorschläge für einen Kunden. Das hinter dem kontextbasierten Empfehlungssystem liegende binäre Klassifikationsmodell ordnet den zuvor aufgelisteten Datensätzen die Klassen „kauft“ und „kauft nicht“ zu.

Die Herausforderung bei der Entwicklung des kontextbasierten Empfehlungssystems stellt die Definition des Kontextes dar. In der Enterprise-Softwareindustrie kann das Umfeld des Kunden nicht, wie in dem Endkonsumenten-Bereich, über soziale Medien erlangt werden, sondern muss durch das passive Beobachten des Kunden erhoben oder von vorhandenen Kundendaten erschlossen werden. Infolgedessen wurden unter anderem abgeleitete Attribute berechnet, die das Umfeld des Kunden und somit den Kontext repräsentieren. Zu diesen Attributen zählt beispielsweise die Lizenzmodelltendenz eines Kunden oder die Kaufrate, also das Verhältnis zwischen den gekauften Produkten und der Anzahl der betrachteten Produkte eines Kunden. Um die Vorhersagen des kontextbasierten Empfehlungssystems einordnen zu können, wurde ein weiteres Empfehlungssystem entwickelt, das ohne die Hinzunahme von Kontextdaten Produktemp-

fehlungen ausspricht. Vergleichen ließen sich beide entwickelten Empfehlungssysteme anhand ausgewählter Metriken. Hierbei lag ein besonderes Augenmerk unter anderem auf dem F_1-Score und der AUROC. Aus der Evaluation ging hervor, dass das kontextbasierte Empfehlungssystem ausnahmslos bei allen Metriken präzisere Ergebnisse liefert als das klassische Modell. So ist der F_1-Score des kontextbasierten Empfehlungssystems zum Beispiel um circa 76,8 % größer als der des klassischen Modells. Darüber hinaus ist das kontextbasierte Empfehlungssystem durch eine um 42,4 % höhere AUROC ausgezeichnet. Durch das Betrachten der Wahrscheinlichkeitsdichtefunktionen sowie der ROC-Kurven wird ersichtlich, dass das kontextbasierte Empfehlungssystem eindeutiger zwischen den Klassen „kauft“ und „kauft nicht“ unterscheiden kann als das klassische Modell. Des Weiteren zeigt die Auswertung der Metriken sowie die ROC-Kurven, dass das Modell keiner Überanpassung unterliegt. Dies wird durch die ähnlichen Ergebnisse der Metriken sowie die vergleichbaren Verläufe der ROC-Kurven bestätigt. Demnach ist das Modell nicht nur auf den Trainingsdatensatz angepasst, sondern beweist seine Präzision von Vorhersagen ebenfalls bei einem unbekannten Datensatz.

Als Resultat der Evaluation der Ergebnisse kann der potenzielle Einfluss des kontextbasierten Empfehlungssystems auf die Enterprise-Softwareindustrie quantifiziert werden. Das kontextbasierte Empfehlungssystem liefert präzisere Produktempfehlungen als das klassische Empfehlungssystem. Folglich führt die Hinzunahme von Kontext dazu, das Umfeld eines Kunden eindeutiger abzubilden und das Kundenprofil zu konkretisieren. Durch die auf das Kundenprofil angepassten Produktempfehlungen soll der Kunde dazu verleitet werden, das Produkt zu kaufen, wodurch ein verkaufsförderndes Verhalten evoziert wird.

5.2 Ausblick

Die im Rahmen der vorliegenden Arbeit erlangten Ergebnisse müssen im nächsten Schritt mithilfe des DSR praktisch in der Realität umgesetzt werden, sodass der Geschäftskontext der SAP SE abgebildet wird. Das definierte kontextbasierte Empfehlungssystem trifft Annahmen über die betrachtete Menge der Vertriebspipeline- sowie Kundendaten, wodurch im Folgenden evaluiert werden muss, ob das kontextbasierte Empfehlungssystem auf die gesamte Grundmenge übertragbar ist. So werden in dieser

Arbeit beispielsweise nur Unternehmen von mittlerer Größe betrachtet.

Im Rahmen der Arbeit wurde das Beispiel genannt, dass auf der Internetseite der SAP SE den Kunden eine Vielzahl von Produkten vorgeschlagen werden, mit dem Ziel, den Kunden in jedem Stadium seiner Customer Journey von sich zu überzeugen. Durch die Verifizierung der Hypothese, dass in der Enterprise-Softwareindustrie unter Hinzunahme von Kontext die Empfehlungen präziser auf den Kunden abgestimmt werden können, ist anzunehmen, dass eine wohldefinierte kleinere Menge von Produktvorschlägen auf der Internetseite der SAP SE verkaufsfördernde Ausmaße hervorrufe. Zur Umsetzung dieses Ansatzes auf der Internetseite der SAP SE bedarf es an Vertriebspipeline- und generellen Kundendaten. Die Herausforderung an dieser Stelle ist, dass diese Daten für Neukunden nicht im vollen Umfang vorliegen. Demnach müssen fiktive Vertriebspipelinedaten zu den Besuchern der Internetseite der SAP SE entwickelt werden, die auf Grundlage des Webseitenbesuches der Benutzer generiert werden. Dieser Herausforderung wird sich die SAP SE in der Zukunft im Rahmen der Phase der Bereitstellung des CRISP-DM Vorgehensmodells stellen müssen, mit dem Ziel, ein kontextbasiertes Empfehlungssystem bei der SAP SE zu verwirklichen.

Darüber hinaus gilt es, zu erforschen, ob über den in der vorliegenden Arbeit definierten Kontext hinaus, weitere Daten helfen können, das Umfeld eines Kunden zu konkretisieren. Zur Generierung weiterführender Daten können zum einen weitere Attribute aus den vorhandenen Daten abgeleitet werden. Zum anderen können neue, implizite Inhalte durch das Beobachten der Kunden erhoben werden. Des Weiteren liegt der Fokus der vorliegenden Arbeit auf der Umsetzung eines kontextbasierten Empfehlungssystems. In diesem Zuge wurde die Optimierung der Anwendung des Random Forest Algorithmus vernachlässigt. Demnach gilt es, zu explorieren, ob die Präzision des Empfehlungssystems durch eine optimierte Verwendung des Random Forest Algorithmus gesteigert werden kann.

Literaturverzeichnis

Abbas, Assad, Limin Zhang und Samee U. Khan (Juli 2015). „A survey on context-aware recommender systems based on computational intelligence techniques“. In: *Computing* 97.7, S. 667–690.

Adomavicius, Gediminas, Ramesh Sankaranarayanan et al. (Jan. 2005). „Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach“. In: *ACM Transactions on Information Systems* 23.1, S. 103–145.

Adomavicius, Gediminas und Alexander Tuzhilin (2001). „Extending Recommender Systems: A Multidimensional Approach“. In: *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-01), Workshop on Intelligent Techniques for Web Personalization (ITWP2001)*. Seattle, Washington.

– (Juni 2005). „Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions“. In: *IEEE Transactions on Knowledge and Data Engineering* 17.6, S. 734–749.

Alan, Said, Berkovsky Shlomo und Luca Ernesto W. De (Jan. 2013). „Movie recommendation in context“. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 4.1, S. 1–9.

Amazon Web Services (2021). *Was ist Cloud Computing?* URL: https://aws.amazon.com/de/what-is-cloud-computing/ (besucht am 03. 10. 2021).

Apache Spark (2018a). *Apache Spark. Lightning fast unified analytics engine*. URL: https://spark.apache.org/ (besucht am 19. 03. 2021).

– (2018b). *Extracting, transforming and selecting features. OneHotEncoder*. URL: https://spark.apache.org/docs/latest/ml-features#onehotencoder (besucht am 12. 04. 2021).

Arora, Lakshay (Nov. 2019). *Want to Build Machine Learning Pipelines? A Quick Introduction using PySpark*. URL: https://www.analyticsvidhya.com/blog/2019/11/build-machine-learning-pipelines-pyspark/ (besucht am 12. 04. 2021).

Baumann, Florian et al. (2014). „Thresholding a Random Forest Classifier“. In: *Advances in Visual Computing*. Hrsg. von George Bebis et al. Cham: Springer International Publishing, S. 95–106. URL: https://link.springer.com/content/pdf/10.1007%2F978-3-319-14364-4.pdf (besucht am 13. 04. 2024).

Bichler, Martin (2006). „Design science in information systems research“. In: *Wirtschaftsinformatik* 48.2, S. 133–135.

Bleymüller, Josef, Rafael Weißbach und Achim Dörre (2020). *Statistik für Wirtschaftswissenschaftler*. 18., überarbeitete und erweiterte Auflage. Vahlen eLibrary Mathematik für Wirtschaftswissenschaftler. München: Verlag Franz Vahlen.

Bradley, Andrew P. (1997). „The use of the area under the ROC curve in the evaluation of machine learning algorithms“. In: *Pattern Recognition* 30.7, S. 1145–1159.

Breiman, Leo (Okt. 2001). „Random Forests“. In: *Machine Learning* 45.1, S. 5–32. URL: https://doi.org/10.1023/A:1010933404324 (besucht am 24. 02. 2021).

Brocke, Jan vom, Alan Hevner und Alexander Maedche (2020). *Design Science Research. Cases*. Progress in IS. Springer International Publishing.

Brockhaus Enzyklopädie (2021a). „Bewertung (allgemein)“. In: *Brockhaus Enzyklopädie Online*. NE GmbH | Brockhaus. URL: https://brockhaus-de.ezproxy-dhma-1.redi-bw.de/ecs/permalink/EF4DD992C1AF2FCEF02A50B80CA70770.pdf (besucht am 24. 03. 2021).

– (2021b). „Data-Mining“. In: *Brockhaus Enzyklopädie Online*. NE GmbH | Brockhaus. URL: https://brockhaus-de.ezproxy-dhma-2.redi-

bw.de/ecs/permalink/0B012052BD7C7245F541203F6F606BAA.pdf (besucht am 26. 02. 2021).

Burke, Robin (Nov. 2002). „Hybrid Recommender Systems: Survey and Experiments“. In: *User Modeling and User-Adapted Interaction* 12.4, S. 331–370. URL: https://doi.org/10.1023/A:1021240730564 (besucht am 02. 03. 2021).

Burkov, Andriy (2019). *The Hundred-Page Machine Learning Book*. Quebec City, Canada.

Cerda Patricioand Varoquaux, Gaël und Balázs Kégl (Sep. 2018). „Similarity encoding for learning with dirty categorical variables“. In: *Machine Learning* 107.8, S. 1477–1494. URL: https://doi.org/10.1007/s10994-018-5724-2 (besucht am 12. 04. 2021).

Chapman, Pete et al. (2000). „CRISP-DM 1.0. Step-by-step data mining guide“. In: SPSS. URL: https://www.kde.cs.uni-kassel.de/wp-content/uploads/lehre/ws2012-13/kdd/files/CRISPWP-0800.pdf (besucht am 26. 02. 2021).

Chen, Chao, Leo Breiman und Andy Liaw (Juli 2004). „Using Random Forest to Learn Imbalanced Data“. In: *University of California, Berkeley*. URL: https://statistics.berkeley.edu/tech-reports/666 (besucht am 12. 04. 2021).

Chicco, Davide und Giuseppe Jurman (Jan. 2020). „The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation“. In: *BMC Genomics* 21.1, S. 6. URL: https://doi.org/10.1186/s12864-019-6413-7 (besucht am 09. 04. 2021).

Choudhary, Priyankar, Vibhor Kant und Pragya Dwivedi (Dez. 2017). „Handling Natural Noise in Multi Criteria Recommender System utilizing effective similarity measure and Particle Swarm Optimization“. In: *Procedia Computer Science* 115, S. 853–862.

Covington, Paul, Jay Adams und Emre Sargin (2016). „Deep Neural Networks for YouTube Recommendations“. In: *Proceedings of the 10th ACM Conference on Recommender Systems*. New York, USA.

Dey, Anind K. (Feb. 2001). „Understanding and Using Context“. In: *Personal and Ubiquitous Computing* 5.1, S. 4–7. URL: https://doi.org/10.1007/s007790170019 (besucht am 01. 03. 2021).

Domingues, Marcos Aurélio, Alípio Mário Jorge und Carlos Soares (Okt. 2009). „Using Contextual Information as Virtual Items on Top-N Recommender Systems“. In: New York, NY, USA.

– (2013). „Dimensions as Virtual Items: Improving the predictive ability of top-N recommender systems“. In: *Information Processing & Management* 49.3. Personalization and Recommendation in Information Access, S. 698–720. URL: https://www.sciencedirect.com/science/article/pii/S0306457312001069 (besucht am 04. 03. 2021).

Donges, Niklas (Juni 2019). *The Random Forest Algorithm: A Complete Guide*. URL: https://builtin.com/data-science/random-forest-algorithm (besucht am 24. 02. 2021).

Dresch, Aline, Daniel Pacheco Lacerda und José Antônio Valle Antunes Jr. (2015). *Design Science Research. A Method for Science and Technology Advancement*. SpringerLink Bücher. Springer.

Følstad, Asbjørn und Knut Kvale (Jan. 2018). „Customer journeys: a systematic literature review“. In: *Journal of Service Theory and Practice* 28.2, S. 196–227. URL: https://doi.org/10.1108/JSTP-11-2014-0261 (besucht am 24. 03. 2021).

Følstad, Asbjørn, Knut Kvale und Ragnhild Halvorsrud (Jan. 2014). „Customer Journeys: Involving Customers and Internal Resources in the Design and Management of Services“. In: S. 412–417.

Frochte, Jörg (2019). *Maschinelles Lernen. Grundlagen und Algorithmen in Python*. 2., aktualisierte Auflage. München: Carl Hanser Verlag.

Gartner (2021). *Enterprise Application Software. Gartner Glossary*. URL: https://www.gartner.com/en/information-technology/glossary/enterprise-application-software (besucht am 30. 10. 2021).

Goodfellow, Ian, Yoshua Bengio und Aaron Courville (2016). *Deep Learning*. MIT Press. URL: http://www.deeplearningbook.org (besucht am 22. 04. 2021).

Google Developers (Feb. 2020a). *Classification: ROC Curve and AUC*. URL: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.

– (Feb. 2020b). *Classification: Thresholding*. URL: https://developers.google.com/machine-learning/crash-course/classification/thresholding (besucht am 09. 04. 2021).

Haase, Alex (Apr. 2020). *Sales-Pipeline: Vertriebsprozesse transparent nachverfolgen*. URL: https://blog.hubspot.de/sales/sales-pipeline (besucht am 05. 03. 2021).

Haruna, Khalid et al. (2017). „Context-Aware Recommender System: A Review of Recent Developmental Process and Future Research Direction“. In: *Applied Sciences* 7.12. URL: https://www.mdpi.com/2076-3417/7/12/1211 (besucht am 28. 04. 2021).

Hu, Yifan, Yehuda Koren und Chris Volinsky (2008). „Collaborative Filtering for Implicit Feedback Datasets“. In: *2008 Eighth IEEE International Conference on Data Mining*, S. 263–272. URL: http://yifanhu.net/PUB/cf.pdf (besucht am 13. 04. 2021).

Jannach, Dietmar und Michael Jugovac (Dez. 2019). „Measuring the Business Value of Recommender Systems“. In: *ACM Transactions on Management Information Systems* 10.4. URL: https://doi.org/10.1145/3370082 (besucht am 29. 04. 2021).

Kim, Svetlana und Yonglk Yoon (2016). „Recommendation system for sharing economy based on multidimensional trust model“. In: *Multimedia Tools and Applications: An International Journal* 75.23, S. 15297–15310.

Laudon, Kenneth C., Jane Price Laudon und Detlef Schoder (2015). *Wirtschaftsinformatik. Eine Einführung*. 3., vollständig überarbeitete Auflage. Studium Economic BWL. Pearson Studium.

Li, Shengli et al. (2017). „A Study of Enterprise Software Licensing Models“. In: *Journal of Management Information Systems* 34.1, S. 177–205.

Liu, Yanli, Yourong Wang und Jian Zhang (2012). „New Machine Learning Algorithm: Random Forest“. In: *Information Computing and Applications*. Hrsg. von Baoxiang Liu, Maode Ma und Jincai Chang. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 246–252.

McCarthy, John et al. (Aug. 1955). *A proposal for the Darthmout Summer Research Project on Artificial Intelligence*. URL: http://jmc.stanford.edu/articles/dartmouth/dartmouth.pdf (besucht am 19. 02. 2021).

Mell, Peter und Timothy Grance (Sep. 2011). „The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology“. In: *National Institute of Science and Technology, Special Publication* 800, S. 145.

Microsoft (2021). *Was ist Cloud Computing? Leitfaden für Einsteiger*. URL: https://azure.microsoft.com/de-de/overview/what-is-cloud-computing/ (besucht am 10. 03. 2021).

Misztal, Joanna und Bipin Indurkhya (2015). „Explaining Contextual Recommendations: Interaction design study and prototype implementation“. In: *IntRS@RecSys*.

Netflix (2021). *Netflix Research*. URL: https://research.netflix.com/articles?q=Recommendations (besucht am 05. 03. 2021).

Ntanos, C. et al. (2014). „A context awareness framework for cross-platform distributed applications“. In: *Journal of Systems and Software* 88, S. 138–146. URL: https://www.sciencedirect.com/science/article/pii/S0164121213002562 (besucht am 01. 03. 2021).

pandas (2021). *pandas - Python Data Analysis Library*. URL: https://pandas.pydata.org/ (besucht am 08. 03. 2021).

Park, Deuk Hee et al. (Sep. 2012). „A literature review and classification of recommender systems research“. In: *Expert Systems with Applications* 39.11, S. 10059–10072.

Project Jupyter (Feb. 2021). *The Jupyter Notebook*. URL: https://jupyter.org/ (besucht am 06. 03. 2021).

Rößler, Irene und Albrecht Ungerer (2019). *Statistik für Wirtschaftswissenschaftler. Eine anwendungsorientierte Darstellung*. 6. Aufl. BA KOMPAKT. Mannheim: Springer Gabler.

Russell, Stuart J. et al. (2016). *Artificial Intelligence. A Modern Approach*. 3. Aufl. Prentice Hall series in artificial intelligence. Pearson.

Saltz, Jeff et al. (Aug. 2018). „Exploring Project Management Methodologies Used Within Data Science Teams“. In: *AMCIS 2018 Proceedings*. Twenty-fourth Americas Conference on Information Systems.

SAP (Jan. 2021). *Umsatz des Unternehmens SAP nach Segmenten in den Jahren 2011 bis 2020 (in Millionen Euro)*. URL: https://de.statista.com/statistik/daten/studie/28272/umfrage/umsatzerloese-von-sap-nach-taetigkeitsbereichen/ (besucht am 22. 03. 2021).

SAP SE (2021). *Über SAP SE*. URL: https://www.sap.com/corporate/de.html (besucht am 06. 03. 2021).

Schilit, Bill N. und Marvin M. Theimer (1994). „Disseminating Active Map Information to Mobile Hosts“. In: *IEEE Network* 8.5, S. 22–32.

Schmidt, Albrecht, Michael Beigl und Hans-W. Gellersen (1999). „There is more to context than location“. In: *Computers & Graphics* 23.6, S. 893–901. URL: https://www.sciencedirect.com/science/article/pii/S009784939900120X (besucht am 01. 03. 2021).

Schröer, Christoph, Felix Kruse und Jorge Marx Gómez (2021). „A Systematic Literature Review on Applying CRISP-DM Process Model“. In: *Procedia Computer Science* 181. CENTERIS - International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist -

International Conference on Health and Social Care Information Systems and Technologies 2020, S. 526–534.

Seyednezhad, S.M. Mahdi et al. (Nov. 2018). „A Review on Recommendation Systems: Context-aware to Social-based“. In: *ArXiv*.

Shin, Ilsoon (Apr. 2006). „Adoption of Enterprise Application Software and Firm Performance“. In: *Small Business Economics* 26.3, S. 241–256. URL: https://doi.org/10.1007/s11187-005-0215-9 (besucht am 03. 10. 2021).

Simon, Herbert Alexander (1996). *The Sciences of the Artificial*. 3. Aufl. MIT Press.

Ströhl, Florian und Tobias Rockel (Sep. 2020). „Auswirkungen fehlender Daten in der multiplen Regression. Eine Simulationsstudie“. In: *Biostatistics* 15.4, S. 719–730.

Tharwat, Alaa (Jan. 2020). „Classification assessment methods“. In: *Applied Computing and Informatics* 17.1, S. 168–192. URL: https://doi.org/10.1016/j.aci.2018.08.003 (besucht am 07. 04. 2021).

Waskom, Michael (2020a). *seaborn: statistical data visualization*. URL: https://seaborn.pydata.org/index.html (besucht am 22. 04. 2021).

– (2020b). *seaborn.kdeplot. seaborn 0.11.1 documentation*. URL: https://seaborn.pydata.org/generated/seaborn.kdeplot.html (besucht am 22. 04. 2021).

Welsch, Andreas, Verena Eitle und Peter Buxmann (Apr. 2018). „Maschinelles Lernen“. In: *HMD Praxis der Wirtschaftsinformatik* 55.2, S. 366–382. URL: https://doi.org/10.1365/s40702-018-0404-z (besucht am 19. 02. 2021).

Wittpahl, Volker (2019). *Künstliche Intelligenz. Technologie, Anwendung, Gesellschaft*. SpringerLink Bücher. Berlin: Springer Vieweg.

Xu, Yun und Royston Goodacre (2018). „On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning“. In: *Journal of Analysis and Testing* 2.3, S. 249–262. URL: https://link.springer.com/content/pdf/10.1007/s41664-018-0068-2.pdf (besucht am 12. 04. 2021).

A Anhang

Beschreibung der Tabellenattribute

Attributsname	**Technischer Name**	**Beschreibung**
Pipeline ID	PIPELINE_ID	ID der Vertriebspipeline (Schlüssel)
Account ID	ACCOUNT_ID	Interner ID des Kundenaccounts (Schlüssel)
Levelcode	LEVEL_CODE	Code des Produktlevels (Schlüssel). Die Levels stellen hierarchisch aufgebaute Produktkategorien dar. Je größer das Level ist, desto granularer und expliziter wird das Produkt.
Levelname	LEVEL_NAME	Name des Produktlevels
Planungseinheit	PLANNING_ENTITY	Eine Planungseinheit bietet die Möglichkeit, zusammengehörige Accounts zu gruppieren.
Gekauft	BOUGHT	Boolesche Variable, ob die Pipeline mit einem Kauf abgeschlossen wurde.
Status	STATUS_PIPELINE	Status der Vertriebspipeline
Statusdatum	STATUS_SINCE_PIPE-LINE	Änderungsdatum des Status (YYYYMMDD)
Erstellungsdatum	CREATED_AT_PIPELI-NE	Datum der Erstellung des Eintrags (YYYYMMDD)

Tabelle A.1: Beschreibung der Vertriebspipelinedaten (df_pipeline)

Attributsname	**Technischer Name**	**Beschreibung**
Planungseinheit	PLANNING_ENTITY	Eine Planungseinheit bietet die Möglichkeit, zusammengehörige Accounts zu gruppieren (Schlüssel).
SAP Master Code	SAP_MASTER_CODE	Kernindustrie, in welcher der Account aktiv ist.
Industriecode	INDUSTRY_CODE	Code der Industrie (nicht gleichzusetzen mit der Industrie ID), welcher einer SAP internen Klassifizierung entspricht.
Beziehungsstatus	RELATIONSHIP_STATUS	Beziehungsstatus zu dem Kunden
Partnerverkauf	BUSINESS_PARTNER_OWNED	Verkauf über eine Partnerorganisation
Accountklassifikation	ACCOUNT_CLASSIFICATION	Die Accountklassifikation wird anhand des Umsatzes bestimmt.
Marktsegment	MARKET_SEGMENT	Unternehmensgröße, Unterteilung in S, M und L
Verkaufssegment	SALES_SEGMENT	Interne kategorische Einordnung des Kunden nach diversen Dimensionen, wie beispielsweise Umsatz, Zukunftspotenzial, Marktführerschaft

Tabelle A.2: Beschreibung der Unternehmensdaten (df_accounts)

Attributsname	**Technischer Name**	**Beschreibung**
Produktnummer	PRODUCT_NO	Identifizierende Nummer des Produkts (Schlüssel)
Produktnamen	PRODUCT_NAMES	Interne Namen des Produkts
Offizieller Name	COMMERCIAL_PRODUCT_NAME	Offizieller Name des Produkts, welcher mit dem Kunden kommuniziert wird.
Valide ab Quartal	VALID_FROM_QUARTER	Quartal, ab dem das Produkt valide ist.
Valide bis Quartal	VALID_TO_QUARTER	Quartal, bis zu welchem das Produkt valide ist.

Tabelle A.3: Beschreibung der Produkttabelle (df_products)

Attributsname	**Technischer Name**	**Beschreibung**
Unternehmens-funktion ID	BUSINESS_CAPABILI-TY_ID	ID der Unternehmensfunktion (Schlüssel). Unternehmensfunktionen beschreiben, die vom Kunden geforderten abzubildenden Geschäftsprozesse.
Funktionsumfang ID	SOLUTION_CAPABI-LITY_ID_RECOMMEN DED	ID des Funktionsumfangs (Schlüssel). Funktionsumfänge verweisen auf zu einer Unternehmensfunktion passende Produkte.
Produktnummer	PRODUCT_NO	Identifizierende Nummer des Produkts (Schlüssel)
Funktionsum-fangsname	SOLUTION_CAPABILI-TY_NAME	Name des Funktionsumfangs
Unternehmens-funktionsname	BUSINESS_CAPABILI-TY_NAME	Name der Unternehmensfunktion
Produktname	PRODUCT_NAME	Interner Name des Produkts
Empfohlene Unternehmens-funktion ID	BUSINESS_CAPABILI-TY_ID_RECOMMEN-DED	ID der empfohlenen Unternehmensfunktion
Industrie ID	INDUSTRY_ID	ID der Industrie, welche einer SAP internen Klassifizierung entspricht.
Industriename	INDUSTRY_NAME	Name der Industrie
Mapping Nummer	MAPPING_NO	Mapping einer Nummer zu dem Lizenzmodell (8 = Cloud, 2 = On-Premise)
Valide ab Quartal	VALID_FROM_QUAR-TER	Quartal, ab dem das Produkt valide ist.
Valide bis Quartal	VALID_TO_QUARTER	Quartal, bis zu welchem das Produkt valide ist.

Tabelle A.4: Beschreibung der Empfehlungstabelle (df_recommendations)

Attributsname	**Technischer Name**	**Beschreibung**
Funktionsumfang ID	SOLUTION_CAPABILI-TY_ID	ID des Funktionsumfangs (Schlüssel)
Produktnummer	PRODUCT_NO	ID des Produkts
Industrie ID	INDUSTRY_ID	ID der Industrie, welche einer SAP internen Klassifizierung entspricht.
Kaufgrund	RATIONALE	Gründe, warum der Kunde das Produkt kaufen sollten.

Tabelle A.5: Beschreibung der Funktionsumfangstabelle (df_solution_capability)

Attributsname	**Technischer Name**	**Beschreibung**
Lizenzmodell ID	DEPLOYMEN_ID	ID der Präferenz eines Unternehmen bezüglich des Lizenzmodells für ein Produkt zum Beispiel Cloud, On-Premise (Schlüssel)
Lizenzmodell	DEPLOYMENT	Präferenz eines Unternehmen bezüglich des Lizenzmodells für ein Produkt zum Beispiel Cloud, On-Premise
Mapping Nummer	MAPPING_NO	Mapping einer Nummer zu dem Lizenzmodell (8 = Cloud, 2 = On-Premise)

Tabelle A.6: Beschreibung der Lizenzmodelltabelle (df_deployment)

Attributsname	**Technischer Name**	**Beschreibung**
Produktnummer	PRODUCT_NO	ID des Produkts (Schlüssel)
Levelcode	LEVEL_CODE	Code des Produktlevels (Schlüssel)

Tabelle A.7: Beschreibung der Mappingtabelle (df_map)

B Anhang

B.1 Datenvorbereitung: Datenaufbereitung und Datenzusammenführung

Im ersten Schritt werden bei der Datenzusammenführung die Kontextdaten verbunden. Die Kontextdaten beinhalten die Empfehlungen, die Produkte, den Funktionsumfang und die Lizenzmodelle. Im nächsten Schritt werden die Hintergrunddaten, die Kunden- bzw. Unternehmensdaten und die Vertriebspipelinedaten, zusammengeführt. Anschließend werden die Hintergrund- und Kontextdaten mithilfe von weiteren Mappingtabellen miteinander verbunden.

Bei jeder Zusammenführung werden vorweg die einzelnen Tabellen bearbeitet. Die Spalten der jeweiligen Tabellen werden in einem Spaltenverzeichnis beschrieben und in sprechende Namen umbenannt. Zudem werden nicht benötigte Spalten entfernt. Duplikate in den Tabellen werden mithilfe der `drop_duplicates()` Funktion gelöscht. Nach der Aufbereitung der einzelnen Tabellen wird die Datenzusammenführung durchgeführt.

[1]:
```
# Importieren von Spark und pandas und Starten einer Spark Session.
import pandas as pd
import numpy as np
from pyspark.sql import SparkSession
import findspark
findspark.init()
spark = SparkSession.builder.appName('DataMergings')\
.config('spark.eventLog.enabled', 'false')\
.config('spark.executor.memory', '8g')\
.config('spark.executor.cores', '4')\
.config('spark.driver.memory','8g')\
.getOrCreate()
```

Zusammenführen der Kontextdaten Empfehlungen, Produkte, Funktionsumfang und Lizenzmodelle

Vorbereitung der Zusammenführung

[2]:
```
# Laden der Kontext-Excel-Sheets, darunter die „Empfehlungs-",␣
↪„Produkt-", „Funktionsumfangs-" und „Lizenzmodelletabellen" und␣
↪speichern dieser in der Variable „context_data".
context_data = pd.ExcelFile('./data_recommendations.xlsx')

# Laden der einzelnen Excel-Sheets in DataFrames.
df_recommendations = pd.read_excel(context_data, 'Recommendations')
df_products = pd.read_excel(context_data, 'Product')
df_solution_capability = pd.read_excel(context_data, 'SC to P')
df_deployment = pd.read_excel(context_data, 'MMP Bits')

# Entfernen der Spalten, die für den Anwendungsfall nicht benötigt␣
↪werden.
df_recommendations = df_recommendations.drop(['MANDT',␣
↪'MATCHMAKING_PATTERN'], axis='columns')
df_products = df_products.drop(['VALID_FROM', 'VALID_TO',␣
↪'STN_RELEVANCE', 'IS_RELEVANT_4_COMP_SCOPE',␣
↪'IS_RELEVANT_4_SIMPLIFICATION'], axis='columns')

# Umbenennung der Spalten in sprechende Namen.
df_recommendations = df_recommendations.rename(columns={'BCR_ID':␣
↪'BUSINESS_CAPABILITY_ID_RECOMMENDED','PNR_CURRENT': 'PRODUCT_NO',␣
↪'P_NAME': 'PRODUCT_NAME', 'BC_ID': 'BUSINESS_CAPABILITY_ID',␣
↪'BC_NAME':'BUSINESS_CAPABILITY_NAME', 'MMP_BITS': 'MAPPING_NO',␣
↪'SC_ID_RECOMMENDED': 'SOLUTION_CAPABILITY_ID', 'SC_NAME':␣
↪'SOLUTION_CAPABILITY_NAME'})
df_products = df_products.rename(columns={'PNR': 'PRODUCT_NO',␣
↪'PNAMES': 'PRODUCT_NAMES', 'OFFICIAL_NAME':␣
↪'COMMERCIAL_PRODUCT_NAME'})
df_solution_capability = df_solution_capability.
↪rename(columns={'SC_ID': 'SOLUTION_CAPABILITY_ID', 'PNR':␣
↪'PRODUCT_NO'})
df_deployment = df_deployment.rename(columns={'MMP-Bits':␣
↪'MAPPING_NO', 'Id Deployment Preference': 'DEPLOYMENT_ID',␣
↪'Deployment Preference': 'DEPLOYMENT'})
```

Zusammenführung

[3]:
```
# Zusammenführung der Tabellen „Produkte" und „Funktionsumfang" über␣
 ↪das Attribut „PRODUCT_NO".
df_products_solution_capability = pd.merge(df_products,␣
 ↪df_solution_capability, how='left', on='PRODUCT_NO')

# Zusammenführung der Tabellen „Lizenzmodell" und „Empfehlungen" über␣
 ↪das Attribut „MAPPING_NO".
df_deployment_recommendations = pd.merge(df_deployment,␣
 ↪df_recommendations, on='MAPPING_NO')

# Zusammenführung der Tabellen „Produkte", „Funktionsumfang",␣
 ↪„Lizenzmodell" und „Empfehlungen" über die Attribute „PRODUCT_NO",␣
 ↪„SOLUTION_CAPABILITY_ID" und „INDUSTRY_ID".
# Es entsteht der Kontextdatensatz.
df_context = pd.merge(df_deployment_recommendations,␣
 ↪df_products_solution_capability, how='inner', on=['PRODUCT_NO',␣
 ↪'SOLUTION_CAPABILITY_ID', 'INDUSTRY_ID'])

# Entfernen der Duplikate. Es liegen keine Duplikate vor.
df_context = df_context.drop_duplicates()
```

Aufbereitung der zusammengeführten Daten

[4]:
```
# Datensatz „df_context" aufbereiten, indem die Spalten umbenannt␣
 ↪werden und die Reihenfolge geändert wird.
df_context = df_context.rename(columns={'VALID_FROM_QUARTER_x':␣
 ↪'VALID_FROM_QUARTER_PRODUCT', 'VALID_TO_QUARTER_x':␣
 ↪'VALID_TO_QUARTER_PRODUCT', 'VALID_FROM_QUARTER_y':␣
 ↪'VALID_FROM_QUARTER_RECOM', 'VALID_TO_QUARTER_y':␣
 ↪'VALID_TO_QUARTER_RECOM'})
df_context = df_context[['PRODUCT_NO', 'PRODUCT_NAMES',␣
 ↪'PRODUCT_NAME', 'COMMERCIAL_PRODUCT_NAME',␣
 ↪'VALID_FROM_QUARTER_PRODUCT', 'VALID_TO_QUARTER_PRODUCT',␣
 ↪'INDUSTRY_ID', 'INDUSTRY_NAME', 'SOLUTION_CAPABILITY_ID',␣
 ↪'SOLUTION_CAPABILITY_NAME', 'RATIONALE', 'BUSINESS_CAPABILITY_ID',␣
 ↪'BUSINESS_CAPABILITY_NAME', 'BUSINESS_CAPABILITY_ID_RECOMMENDED',␣
 ↪'VALID_FROM_QUARTER_RECOM', 'VALID_TO_QUARTER_RECOM',␣
 ↪'DEPLOYMENT_ID', 'DEPLOYMENT', 'MAPPING_NO']]
```

```
# Nicht benötigte Spalten wie beispielsweise freie Texte werden␣
 ↪gelöscht.
df_context = df_context.drop(['PRODUCT_NAMES',␣
 ↪'COMMERCIAL_PRODUCT_NAME', 'RATIONALE', 'DEPLOYMENT', 'MAPPING_NO',␣
 ↪'PRODUCT_NAME', 'INDUSTRY_NAME', 'BUSINESS_CAPABILITY_NAME',␣
 ↪'SOLUTION_CAPABILITY_NAME'], axis='columns')

# Endgültiger Kontextdatensatz
df_context.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2185 entries, 0 to 2184
Data columns (total 10 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   PRODUCT_NO                            2185 non-null   object
 1   VALID_FROM_QUARTER_PRODUCT            2185 non-null   int64
 2   VALID_TO_QUARTER_PRODUCT              2185 non-null   int64
 3   INDUSTRY_ID                           2185 non-null   object
 4   SOLUTION_CAPABILITY_ID                2185 non-null   object
 5   BUSINESS_CAPABILITY_ID                2185 non-null   object
 6   BUSINESS_CAPABILITY_ID_RECOMMENDED  2185 non-null   object
 7   VALID_FROM_QUARTER_RECOM              2185 non-null   int64
 8   VALID_TO_QUARTER_RECOM                2185 non-null   int64
 9   DEPLOYMENT_ID            2185 non-null   object
dtypes: int64(4), object(6)
memory usage: 187.8+ KB
```

Zusammenführen der Hintergrunddaten Unternehmen und Vertriebspipelinedaten

Vorbereitung der Zusammenführung

[5]:
```
# Laden der benötigten Excel-Dateien.
df_pipeline = pd.read_pickle('./pipl-with-products.pkl')
df_accounts = pd.read_pickle('./cst.pkl')

# Umbenennung der Spalten.
```

```
df_accounts = df_accounts.rename(columns={'PMASTERC':␣
↪'SAP_MASTER_CODE', 'IND_CODE': 'INDUSTRY_CODE', 'PCRM_IMS':␣
↪'MARKET_SEGMENT', 'PISLSGTM4': 'SALES_SEGMENT', 'PBPOWNDFG':␣
↪'BUSINESS_PARTNER_OWNNED', 'PTARGACC': 'ACCOUNT_CLASSIFICATION',␣
↪'PPBPINDI': 'PRE_PB_INDICATOR', 'PBPACTST': 'RELATIONSHIP_STATUS',␣
↪'PPLNENTY_FINAL': 'PLANNING_ENTITY'})
df_pipeline = df_pipeline.rename(columns={'IC_ACCOUNT_ID':␣
↪'ACCOUNT_ID', 'IC_OBJECT_ID': 'PIPELINE_ID', 'IC_STATUS':␣
↪'STATUS_PIPELINE', 'IC_STATUS_SINCE': 'STATUS_SINCE_PIPELINE',␣
↪'IC_PHASE_TXT': 'PHASE', 'CA_STATUS_SINCE_DATE':␣
↪'STATUS_SINCE_DATE_PIPELINE', 'PRODUCT_ID': 'MATERIAL_NO',␣
↪'PPLNENTY_FINAL': 'PLANNING_ENTITY', 'CREATED_AT':␣
↪'CREATED_AT_PIPELINE', 'CREATED_AT_DATE':␣
↪'CREATED_AT_DATE_PIPELINE'})

# Umformatieren der Spalte „PHASE" des DataFrames „df_pipeline" zu␣
↪einer Spalte mit sprechendem Inhalt.
df_pipeline['BOUGHT'] = np.where((df_pipeline['PHASE']=='A -␣
↪Purchase'), True, False)

# Entfernen der Spalten, die für den Anwendungsfall nicht benötigt␣
↪werden.
df_accounts = df_accounts.drop(['PRE_PB_INDICATOR',␣
↪'BUSINESS_PARTNER_OWNNED'], axis='columns')
df_pipeline = df_pipeline.drop(['PHASE'], axis='columns')
```

Zusammenführung

[6]:
```
# Zusammenführen der beiden Tabellen über das Attribut␣
↪„PLANNING_ENTITY". Es entsteht der Hintergrunddatensatz.
df_background = pd.merge(df_accounts, df_pipeline,␣
↪on='PLANNING_ENTITY')
```

Aufbereitung der zusammengeführten Daten

[7]:
```
# Entfernen der Duplikate in dem Hintergrunddatensatz. Es wurden 2.919.
↪180 Duplikate gelöscht.
df_background = df_background.drop_duplicates()

# Nicht benötigte Spalten wie beispielsweise freier Text wird gelöscht.
```

```
df_background = df_background.drop(['STATUS_SINCE_DATE_PIPELINE',␣
↪'CREATED_AT_DATE_PIPELINE'], axis='columns')
```

Datenreduktion

Im weiteren Verlauf werden lediglich nur Produkte des Levels 3 betrachtet. Dies hat den Grund, dass im Marketingbereich der SAP den Kunden im ersten Schritt die Produkte auf Level 3 vermarktet werden. Erst, wenn sich der Kunde für ein Produkt entschieden hat, kümmert sich der Vertriebler um die genauen Details und zieht Produkte vom Level 3 und höher in Betracht.

[8]:
```
df_background = df_background.
↪drop(df_background[(df_background['LEVEL_NAME']=='LEVEL2') |␣
↪(df_background['LEVEL_NAME']=='LEVEL4') |␣
↪(df_background['LEVEL_NAME']=='LEVEL5') |␣
↪(df_background['LEVEL_NAME']=='LEVEL6') |␣
↪(df_background['LEVEL_NAME']=='LEVEL7')].index)
```

Im Zuge der Datenreduktion werden nur Kunden des Marktsegments „M“ betrachtet. Aufgrund des persönlichen Kontakts und den engen Kundenbeziehungen zu den Unternehmen des Marktsegments „L“ ist dem Unternehmen SAP der Bedarf an Software in diesem Bereich bekannt. Demnach wird sich im Folgenden auf die unbekannten Bereiche konzentriert. Da das Marktsegment „S“ nur in wenigen Fällen vertreten ist, wird sich ledliglich auf das Marktsegment „M“ fokussiert.

[9]:
```
df_background = df_background.
↪drop(df_background[(df_background['MARKET_SEGMENT']=='L') |␣
↪(df_background['MARKET_SEGMENT']=='S')].index)
```

Datenanreicherung

Damit der `INDUSTRY_CODE` des Hintergrunddatensatzes mit der `INDUSTRY_ID` des Kontextdatensatzes in Verbindung gesetzt werden kann, wird der Datensatz „sap-mastercodes-industries“ hinzugezogen.

[10]:
```
# Laden des benötigten Datensatzes.
df_sap_mastercodes = pd.read_csv('./sap-mastercodes-industries.csv',␣
↪sep=';')
df_sap_mastercodes = df_sap_mastercodes[['SAP_MASTER_CODE',␣
↪'INDUSTRY_ID_INPUT']]
```

```
# Zusammenführung des Hintergrunddatensatzes und des Datensatzes␣
 ↪„df_sap_mastercodes" über den „SAP_MASTER_CODE".
df_background = pd.merge(df_background, df_sap_mastercodes,␣
 ↪how='left', on='SAP_MASTER_CODE')

# Löschen nicht benötigter Spalten.
df_background = df_background.drop(columns={'MARKET_SEGMENT',␣
 ↪'LEVEL_NAME', 'SAP_MASTER_CODE', 'INDUSTRY_CODE'}, axis='columns')

# Endgültiger Hintergrunddatensatz
df_background.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 724433 entries, 0 to 724432
Data columns (total 12 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   SALES_SEGMENT           724433 non-null  object
 1   ACCOUNT_CLASSIFICATION  724433 non-null  object
 2   RELATIONSHIP_STATUS     724433 non-null  object
 3   PLANNING_ENTITY         724433 non-null  object
 4   ACCOUNT_ID              724433 non-null  object
 5   PIPELINE_ID             724433 non-null  object
 6   STATUS_PIPELINE         724433 non-null  object
 7   STATUS_SINCE_PIPELINE   724433 non-null  object
 8   LEVEL_CODE              724433 non-null  object
 9   CREATED_AT_PIPELINE     724433 non-null  float64
 10  BOUGHT                  724433 non-null  bool
 11  INDUSTRY_ID_INPUT       472858 non-null  object
dtypes: bool(1), float64(1), object(11)
memory usage: 72.5+ MB
```

Zusammenführen der Hintergrund- und Kontextdaten

[11]:
```
# Laden der Mappingtabelle.
df_map = pd.read_excel('./product_to_level3.xlsx')
df_map.info()

# Zusammenführung der Kontextdaten und der Mappingdaten
df_context['PRODUCT_NO'] = df_context['PRODUCT_NO'].astype('int')
df_context_map = pd.merge(df_context, df_map, how='inner', on =␣
 ↪'PRODUCT_NO')
```

```
# Zwischenspeichern der Datensätze.
path_context_map = './context_map.csv'
df_context_map.to_csv(path_context_map, encoding='UTF-8', index=False)
path_background = './background.csv'
df_background.to_csv(path_background, encoding='UTF-8', index=False)

# Laden der Pandas DataFrames als Spark DataFrames.
dfs_background = spark.read.csv(path_background, header=True)
dfs_context_map = spark.read.csv(path_context_map, header=True)

# Zusammenführung des Hintergrunds- und Kontextdatensatzes.
dfs_data = dfs_background.join(dfs_context_map, ['LEVEL_CODE'],␣
 ↪how='inner')

# Zwischenspeichern des Datensatzes.
path_data = './data.parquet'
dfs_data.write.parquet(path_data, encoding='UTF-8', header=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   LEVEL_CODE  9 non-null      object
 1   PRODUCT_NO  9 non-null      int64
dtypes: int64(1), object(1)
memory usage: 272.0+ bytes
```

B.2 Aufbereitung der konsolidierten Daten

Aufgrund des Zusammenführen der Hintergrund- und Kontextdaten, werden einer Pipeline ID mehrere Datensätze zugewiesen. Damit eine Kundenpipeline durch genau eine Zeile repräsentiert wird, werden zusammengehörige Zeilen vereint.

[1]:
```
# Importieren von Spark und pandas und Starten einer Spark Session.
import pandas as pd
from pyspark.sql import SparkSession
import findspark
findspark.init()
spark = SparkSession.builder.appName('DataMergings')\
.config('spark.eventLog.enabled', 'false')\
```

```
.config('spark.executor.memory', '8g')\
.config('spark.executor.cores', '4')\
.config('spark.driver.memory','8g')\
.getOrCreate()
```

[2]:
```
# Imports
from pyspark.sql.window import Window
from pyspark.sql.functions import collect_list, array_contains, col,␣
 ↪when, struct

# Einlesen des Datensatzes.
dfs_data = spark.read.parquet('./data/')
```

Speichern der Industrien einer Empfehlung in einer Spalte des Typs Array

Eine Empfehlung kann mehreren Industrien zugeordnet sein. Damit alle Industrien einer Empfehlung in einer Zeile gespeichert werden können, wird für jede Industrie eine eigene Spalte erstellt. Eine 1 signalisiert in dem Falle, dass die Empfehlung der in der Spalte beschriebenen Industrie zugeordnet ist.

[3]:
```
# Speichern aller möglichen Industrien in einem Array.
industry_list = [
    'I_AERODE',
    'I_AUTOM',
    'I_BANKIN',
    'I_CHEMIC',
    'I_CP',
    'I_CROSS',
    'I_DEFSEC',
    'I_ECO',
    'I_ENTERT',
    'I_HEALTH',
    'I_HIGHRE',
    'I_HIGHTE',
    'I_IMC',
    'I_INSURN',
    'I_LIFE_S',
    'I_MEDIA',
    'I_MILLPR',
    'I_MINING',
    'I_OIL_GA',
    'I_PROFSV',
```

```
    'I_PUB_SE',
    'I_RETAIL',
    'I_S4HANA',
    'I_SPORTS',
    'I_TELECO',
    'I_TRANSP',
    'I_UTILIT',
    'I_WHOLSA',
    'I_WSTENV',
    'I_Z_INT'
]


# Übertragung der Industrien einer Empfehlung in eine Spalte.
aggAccountID  = Window.partitionBy('ACCOUNT_ID')
dfs_data = dfs_data.withColumn('INDUSTRIES',␣
 ↪collect_list('INDUSTRY_ID').over(aggAccountID))

# Erstellung einer Spalte für jede „INDUSTRY_ID" und Übertragung der␣
 ↪Industrien einer Empfehlung in die entsprechende Spalte.
for industry in industry_list:
    for cur_industry in dfs_data.select('INDUSTRIES'):
        dfs_data = dfs_data.withColumn(industry,␣
 ↪when(array_contains(cur_industry, industry), 1).otherwise(0))

dfs_data = dfs_data.drop('INDUSTRIES', 'INDUSTRY_ID')
```

Speichern der Lizenzmodelle einer Pipeline in einer Spalte des Typs Array

Ein Kunde kann sich sowohl für die On-Premise- als auch die Cloud-Variante eines Produktes interessieren. Genauso wie bei den Industrien wird für jedes Lizenzmodell eine eigene Spalte angelegt. Eine 1 signalisiert, dass der Kunde sich für das Lizenzmodell interessiert und eine 0 besagt, dass der Kunde kein Interesse für das Lizenzmodell gezeigt hat.

[4]:
```
# Speichern aller möglichen Lizenzmodelle in einem Array.
deployment_list = [
    'ONPREM',
    'PUBCLOUD'
]

# Übertragung der Lizenzmodelle einer Pipeline jeweils in eine Spalte.
aggPipelineID  = Window.partitionBy('PIPELINE_ID')
```

```
dfs_data = dfs_data.withColumn('DEPLOYMENTS',␣
 ↪collect_list('DEPLOYMENT_ID').over(aggPipelineID))

# Erstellung einer Spalte für jede „DEPLOYMENT_ID" und Übertragung der␣
 ↪Lizenzmodelle einer Pipeline in die entsprechende Spalte.
for deployment in deployment_list:
    for cur_deployment in dfs_data.select('DEPLOYMENTS'):
        dfs_data = dfs_data.withColumn(deployment,␣
 ↪when(array_contains(cur_deployment, deployment), 1).otherwise(0))

dfs_data = dfs_data.drop('DEPLOYMENTS', 'DEPLOYMENT_ID')
```

Speichern der Unternehmensfunktion, empfohlenen Unternehmensfunktion und des Funktionsumfangs einer Pipeline in einer Spalte des Typs Array

Zu einer Kundenpipeline können mehrere Unternehmensfunktionen, empfohlene Unternehmensfunktionen und Funktionsumfänge vorliegen, welche im Folgenden in einer Spalte des ArrayTypes gespeichert werden.

[5]:
```
# Speichern des Validitätszeitraums der Unternehmensfunktion als␣
 ↪StructType.
dfs_data = dfs_data.withColumn('BUSINESS_CAPABILITIY',
    struct(col('BUSINESS_CAPABILITY_ID_RECOMMENDED'),
        struct(col('VALID_FROM_QUARTER_RECOM'),
            col('VALID_TO_QUARTER_RECOM')))).
 ↪drop('VALID_FROM_QUARTER_RECOM','VALID_TO_QUARTER_RECOM')

# Übertragung der Produktnummern, Funktionsumfänge,␣
 ↪Unternehmensfunktionen sowie empfohlene Unternehmensfunktionen␣
 ↪einer Pipeline in eine Spalte des ArrayTypes.
aggPipelineID  = Window.partitionBy('PIPELINE_ID')
dfs_data = dfs_data.withColumn('SOLUTION_CAPABILITIES',␣
 ↪collect_list('SOLUTION_CAPABILITY_ID').over(aggPipelineID))
dfs_data = dfs_data.withColumn('BUSINESS_CAPABILITIES',␣
 ↪collect_list('BUSINESS_CAPABILITY_ID').over(aggPipelineID))
dfs_data = dfs_data.withColumn('BUSINESS_CAPABILITIES_RECOMMENDED',␣
 ↪collect_list('BUSINESS_CAPABILITY_ID_RECOMMENDED').
 ↪over(aggPipelineID))
dfs_data = dfs_data.
 ↪withColumn('BUSINESS_CAPABILITIES_RECOMMENDED_VALIDITY',␣
 ↪collect_list('BUSINESS_CAPABILITIY').over(aggPipelineID))
```

```
dfs_data = dfs_data.withColumn('PRODUCTS', collect_list('PRODUCT_NO').
 ↪over(aggPipelineID))

dfs_data = dfs_data.drop('PRODUCT_NO', 'BUSINESS_CAPABILITY_ID',␣
 ↪'SOLUTION_CAPABILITY_ID', 'BUSINESS_CAPABILITY_ID_RECOMMENDED',␣
 ↪'BUSINESS_CAPABILITIY')
```

Sampling und Zwischenspeichern des Datensatzes

Im Rahmen der Datenreduktion wird der Datensatz mithilfe der von Spark bereitgestellten Sampling-Methode reduziert.

[6]:
```
# Sampeln des Datensatzes.
pipeline_ids = dfs_data.select("PIPELINE_ID").distinct()
pipeline_ids_sample = pipeline_ids.sample(0.10, 123)
dfs_data = pipeline_ids_sample.join(dfs_data, ['PIPELINE_ID'])

# Zwischenspeichern des Datensatzes.
path = './data.parquet'
dfs_data.write.parquet(path)
```

B.3 Datenanreicherung

Bei den zuvor definierten Kontextdaten handelt es sich zum Teil um abgeleitete Attribute, welche im Folgenden berechnet werden. Zu diesen Attributen zählen die Lizenzmodelltendenz (`TOTAL_DEPLOYMENT_ID`), die Anzahl der empfohlenen Unternehmensfunktionen (`NO_BUSINESS_CAPABILITES_RECOM`), die Anzahl der Produkte (`NO_PRODUCTS`), die Verweildauer eines Kunden in einer Vertriebspipeline (`TIME_IN_PIPELINE`), die durchschnittliche Verweildauer eines Kunden in einer Vertriebspipeline (`AVERAGE_TIME_IN_PIPELINE_ACCOUNT`), die durchschnittliche Verweildauer eines Produktes in einer Vertriebspipeline (`AVERAGE_TIME_IN_PIPELINE_PRODUCTS`) und die Summe der gekauften Lösungen im Verhältnis zu den betrachteten Lösungen (`BOUGHT_RATIO_PERCENTAGE`).

[1]:
```
# Importieren von Spark und pandas und Starten einer Spark Session.
import pandas as pd
from pyspark.sql import SparkSession
import findspark
findspark.init()
spark = SparkSession.builder.appName('DataMergings')\
.config('spark.eventLog.enabled', 'false')\
.config('spark.executor.memory', '8g')\
```

```
.config('spark.executor.cores', '4')\
.config('spark.driver.memory','8g')\
.getOrCreate()
```

[2]:
```
# Imports
from pyspark.sql.window import Window
from pyspark.sql.functions import col, when, sum, to_date, datediff,␣
 ↪size, expr

# Einlesen des Datensatzes.
dfs_data = spark.read.parquet('./data/')
```

Summe der gekauften Lösungen im Verhältnis zu der Summe der betrachteten Lösungen

[3]:
```
# Gruppierung des Datensatzes nach der „Account ID".
aggAccountID  = Window.partitionBy('ACCOUNT_ID')

# Umwandeln der Variable „Gekauft" zu einem Integer.
dfs_data = dfs_data.withColumn('BOUGHT', when(dfs_data['BOUGHT'] ==␣
 ↪'False', 0).otherwise(1))
dfs_data = dfs_data.withColumn('BOUGHT', col('BOUGHT').cast('int'))

# Berechnung der Summe der gekauften Produkte eines Kunden sowie die␣
 ↪Summe der betrachteten Kunden.
dfs_data = dfs_data.withColumn('SUM_BOUGHT_PRODUCTS_ACCOUNT',␣
 ↪sum(col('BOUGHT')).over(aggAccountID))
dfs_data = dfs_data.withColumn('INITIAL', when(dfs_data['BOUGHT'] ==␣
 ↪0, 1).otherwise(1))
dfs_data = dfs_data.withColumn('TOTAL_PRODUCTS_ACCOUNT',␣
 ↪sum(col('INITIAL')).over(aggAccountID))

# Berechnung der Kaufquote.
dfs_data = dfs_data.withColumn('BOUGHT_RATIO_PERCENTAGE',␣
 ↪(dfs_data['SUM_BOUGHT_PRODUCTS_ACCOUNT'] /␣
 ↪dfs_data['TOTAL_PRODUCTS_ACCOUNT']) * 100 )
dfs_data = dfs_data.drop('INITIAL', 'SUM_BOUGHT_PRODUCTS_ACCOUNT',␣
 ↪'TOTAL_PRODUCTS_ACCOUNT')
```

Anzahl der empfohlenen Unternehmensfunktionen und Produkte

[4]:
```
dfs_data = dfs_data.withColumn('NO_BUSINESS_CAPABILITES_RECOM',␣
 ↪when((size(col('BUSINESS_CAPABILITIES_RECOMMENDED')) == 1) &␣
 ↪(col('BUSINESS_CAPABILITIES_RECOMMENDED')[0] == '' ), 0).
 ↪otherwise(size(col('BUSINESS_CAPABILITIES_RECOMMENDED'))))
dfs_data = dfs_data.withColumn('NO_PRODUCTS',␣
 ↪when((size(col('PRODUCTS')) == 1) & (col('PRODUCTS')[0] == '' ), 0).
 ↪otherwise(size(col('PRODUCTS'))))
```

Verweildauer eines Kunden in einer Vertriebspipeline

[5]:
```
# Berechnung der Zeitdifferenz zwischen dem Erstellungsdatum der␣
 ↪Pipeline und dem Datum der letzten Aktualisierung der Pipeline.
dfs_data = dfs_data.withColumn('STATUS_SINCE_PIPELINE',␣
 ↪col('STATUS_SINCE_PIPELINE').cast('string'))
dfs_data = dfs_data.withColumn('CREATED_AT_PIPELINE',␣
 ↪expr('substring(CREATED_AT_PIPELINE, 1,␣
 ↪length(CREATED_AT_PIPELINE)-8)'))
dfs_data = dfs_data.withColumn('CREATED_AT_PIPELINE',␣
 ↪col('CREATED_AT_PIPELINE').cast('string'))
dfs_data = dfs_data.withColumn('STATUS_SINCE_PIPELINE_DATE',␣
 ↪to_date(col('STATUS_SINCE_PIPELINE'),'yyyyMMdd'))
dfs_data = dfs_data.withColumn('CREATED_AT_PIPELINE_DATE',␣
 ↪to_date(col('CREATED_AT_PIPELINE'),'yyyyMMdd'))
dfs_data = dfs_data.withColumn('TIME_IN_PIPELINE',␣
 ↪datediff(col('STATUS_SINCE_PIPELINE_DATE'),␣
 ↪col('CREATED_AT_PIPELINE_DATE')))

dfs_data = dfs_data.drop('STATUS_SINCE_PIPELINE_DATE',␣
 ↪'CREATED_AT_PIPELINE_DATE')
dfs_data = dfs_data.withColumn('STATUS_SINCE_PIPELINE',␣
 ↪col('STATUS_SINCE_PIPELINE').cast('int'))
dfs_data = dfs_data.withColumn('CREATED_AT_PIPELINE',␣
 ↪col('CREATED_AT_PIPELINE').cast('int'))
```

Durchschnittliche Verweildauer eines Kunden in einer Vertriebspipeline

[6]:
```
# Berechnung der Summe der Verweildauer eines Kunden in einer Pipeline␣
↪und der Summe der Pipelines eines Kunden.
dfs_data = dfs_data.withColumn('SUM_TIME_IN_PIPELINE',␣
↪sum(col('TIME_IN_PIPELINE')).over(aggAccountID))
dfs_data = dfs_data.withColumn('INITIAL', when(dfs_data['BOUGHT'] ==␣
↪0, 1).otherwise(1))
dfs_data = dfs_data.withColumn('NO_PIPELINES', sum(col('INITIAL')).
↪over(aggAccountID))

# Berechnung der durchschnittlichen Verweildauer.
dfs_data = dfs_data.withColumn('AVERAGE_TIME_IN_PIPELINE_ACCOUNT',␣
↪(dfs_data['SUM_TIME_IN_PIPELINE'] / dfs_data['NO_PIPELINES']))
dfs_data = dfs_data.drop('NO_PIPELINES', 'SUM_TIME_IN_PIPELINE',␣
↪'INITIAL')
```

Durchschnittliche Dauer einer Pipeline eines bestimmten Produktes

[7]:
```
# Gruppierung der Daten nach Produkten.
aggProducts  = Window.partitionBy('PRODUCTS')

# Berechnung der Summe der Verweildauer eines Produktes in einer␣
↪Pipeline und der Summe der Pipelines eines Produkts.
dfs_data = dfs_data.withColumn('SUM_TIME_IN_PIPELINE_PRODUCT',␣
↪sum(col('TIME_IN_PIPELINE')).over(aggProducts))
dfs_data = dfs_data.withColumn('INITIAL', when(dfs_data['BOUGHT'] ==␣
↪0, 1).otherwise(1))
dfs_data = dfs_data.withColumn('NO_PRODUCTS_TIME', sum(col('INITIAL')).
↪over(aggProducts))

# Berechnung der durchschnittlichen Dauer.
dfs_data = dfs_data.withColumn('AVERAGE_TIME_IN_PIPELINE_PRODUCT',␣
↪(dfs_data['SUM_TIME_IN_PIPELINE_PRODUCT'] /␣
↪dfs_data['NO_PRODUCTS_TIME']))
dfs_data = dfs_data.drop('SUM_TIME_IN_PIPELINE_PRODUCT',␣
↪'NO_PRODUCTS_TIME', 'INITIAL')
```

Lizenzmodelltendenz eines Kunden

[8]:
```
# Berechnung der Summe der On-Premise- sowie Cloud-Produkte, an denen␣
 ↪ein Kunde Interesse gezeigt hat.
dfs_data = dfs_data.withColumn('SUM_ONPREM', sum(col('ONPREM')).
 ↪over(aggAccountID))
dfs_data = dfs_data.withColumn('SUM_PUBCLOUD', sum(col('PUBCLOUD')).
 ↪over(aggAccountID))

# Das Lizenzmodell mit der größeren Summe stellt die präferierte␣
 ↪Lösung dar.
dfs_data = dfs_data.withColumn('TOTAL_DEPLOYMENT_ID',␣
 ↪when(dfs_data['SUM_ONPREM'] > dfs_data['SUM_PUBCLOUD'], 'ONPREM').
 ↪when(dfs_data['SUM_PUBCLOUD'] > dfs_data['SUM_ONPREM'], 'PUBCLOUD').
 ↪otherwise('NO_PREFERENCE'))
dfs_data = dfs_data.drop('SUM_ONPREM', 'SUM_PUBCLOUD')
```

B.4 Modellierung

Im Rahmen der Modellierung wird der zusammengeführte und angereicherte Datensatz für die Modellierung aufbereitet, um mithilfe des Random Forest Algorithmus ein Klassifikationsmodell aufzubauen. Das Klassifikationsmodell ordnet einen Datensatz anhand seiner Merkmale der Klasse „gekauft“ oder „nicht gekauft“ zu.

[1]:
```
# Importieren von Spark und pandas und Starten einer Spark Session.
import pandas as pd
from pyspark.sql import SparkSession
import findspark
findspark.init()
spark = SparkSession.builder.appName('DataMergings')\
.config('spark.eventLog.enabled', 'false')\
.config('spark.executor.memory', '8g')\
.config('spark.executor.cores', '4')\
.config('spark.driver.memory','8g')\
.getOrCreate()
```

[2]:
```
# Imports
import pyspark.sql.functions as F
from pyspark.sql.functions import col, to_date, expr, when, concat_ws,␣
 ↪round, col, udf
```

```
from pyspark.ml.feature import StringIndexer, OneHotEncoder,␣
 ↪VectorAssembler
from pyspark.ml import Pipeline
from pyspark.ml.classification import RandomForestClassifier
import sklearn
from sklearn.metrics import confusion_matrix
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from pyspark.sql.types import FloatType
import seaborn as sns
```

[3]:
```
# Einlesen des Datensatzes.
dfs_data = spark.read.parquet('./data/')
dfs_data.count()

# Konvertierung der Spalten, die Zahlen enthalten, zu einem␣
 ↪numerischen Datentypen, die durch den Import verloren gegangen sind.
dfs_data = dfs_data.withColumn('SALES_SEGMENT', col('SALES_SEGMENT').
 ↪cast('int'))
dfs_data = dfs_data.withColumn('PLANNING_ENTITY',␣
 ↪col('PLANNING_ENTITY').cast('int'))
dfs_data = dfs_data.withColumn('ACCOUNT_ID', col('ACCOUNT_ID').
 ↪cast('int'))
dfs_data = dfs_data.withColumn('PIPELINE_ID', col('PIPELINE_ID').
 ↪cast('int'))
dfs_data = dfs_data.withColumn('VALID_FROM_QUARTER_PRODUCT',␣
 ↪col('VALID_FROM_QUARTER_PRODUCT').cast('int'))
dfs_data = dfs_data.withColumn('VALID_TO_QUARTER_PRODUCT',␣
 ↪col('VALID_TO_QUARTER_PRODUCT').cast('int'))
```

[3]: 20684

Datenaufbereitung

Damit der konsolidierte Datensatz für die Modellierung angewandt werden kann, muss dieser die Anforderungen des Random Forest Algorithmus erfüllen. Zu diesen Anforderungen zählt die Analyse der Null-Werte, die Balance der Zielvariable Y `BOUGHT`, das Konvertieren der Zielvariable in einen numerischen Datentypen, das Konvertieren der Merkmale in binäre Vektoren und die Überprüfung, ob jede Spalte mindestens zwei verschiedene Inhalte vorweist.

Die Zielvariable Y wurde bereits im Rahmen der Datenanreicherung in einen numerischen Typen konvertiert. Nach Bildung des Trainings-, Test- und Validierungsdatensatzes werden die Merkmale in binäre Vektoren umgewandelt.

Analyse der Null-Werte

[4]:
```
# Zählen der Null-Werte. Aufgrund der Übersichtlichkeit werden nur␣
↪diejenigen Spalten angezeigt, die Null-Werte beinhalten.
dfs_data_null = dfs_data.agg(*[F.count(when(F.isnull(c), c)).alias(c)␣
↪for c in dfs_data.columns])
dfs_data_null.limit(1).toPandas()
```

[4]:
```
  RELATIONSHIP_STATUS  INDUSTRY_ID_INPUT
                    1               7928
```

[5]:
```
# Ersetzen der Null-Werte der Spalte „INDUSTRY_ID_INPUT" mit „I_NO".
dfs_data = dfs_data.na.fill('I_NO',['INDUSTRY_ID_INPUT'])

# Entfernen der Einträge, wo die „RELATIONSHIP_STATUS" oder␣
↪„STATUS_PIPELINE" Null ist.
dfs_data = dfs_data.filter('RELATIONSHIP_STATUS is not null')
```

[5]:
```
20683
```

Ausbalancierung des Datensatzes

Damit die Zielvariable `BOUGHT` ausgeglichen dargestellt wird, wird der Datensatz ausbalanciert. Demnach wird der Datensatz so reduziert, sodass jeweils 5938 Datensätze der Klasse „gekauft“ (1) und „nicht gekauft“ (0) zugeordnet werden.

[6]:
```
# Überprüfung der Balance zwischen den Klassen „gekauft" und „nicht␣
↪gekauft".
dfs_data.groupBy('BOUGHT').count().show()
```

```
+------+-----+
|BOUGHT|count|
+------+-----+
|     1| 5938|
|     0|14745|
+------+-----+
```

[7]:
```
# Reduzierung der Datensätze, die der Klasse „nicht gekauft"␣
 ↪zugeordnet sind.
dfs_data_bought_false = dfs_data.filter(dfs_data['BOUGHT'] == 0)
dfs_data_bought_false = dfs_data_bought_false.sample(0.3985,123)
dfs_data_bought_false.count()
```

[7]: 5938

[8]:
```
# Zusammenführung der Datensätze der Klasse „gekauft" und des␣
 ↪reduzierten Datensatzes der Klasse „nicht gekauft".
dfs_data_bought_true = dfs_data.filter(dfs_data['BOUGHT'] == 1)
dfs_data = dfs_data_bought_false.unionAll(dfs_data_bought_true)
dfs_data.count()
```

[8]: 11876

Überprüfung, ob jede Spalte mindestens zwei verschiedene Inhalte vorweist

Es wird überprüft, wie viele verschiedene Inhalte die jeweiligen Spalten vorweisen. Es ist zu erkennen, dass die Spalten VALID_FROM_QUARTER_PRODUCT und VALID_TO_QUARTER_PRODUCT jeweils nur einen Inhalt beinhalten und somit gelöscht werden. Die Ergebnisse der Gruppierungen werden aufgrund von Vertraulichkeit und Datenschutzgründen nicht angezeigt.

[9]:
```
# Überprüfen, wie viele verschiedene Inhalte die Spalten␣
 ↪„VALID_FROM_QUARTER_PRODUCT" und „VALID_TO_QUARTER_PRODUCT"␣
 ↪vorweisen.
dfs_data.groupBy('VALID_FROM_QUARTER_PRODUCT').count().show()
dfs_data.groupBy('VALID_TO_QUARTER_PRODUCT').count().show()

# Die Spalten „VALID_FROM_QUARTER_PRODUCT" und␣
 ↪„VALID_TO_QUARTER_PRODUCT" werden gelöscht, da diese bei jedem␣
 ↪Eintrag den gleichen Wert enthalten.
dfs_data = dfs_data.drop('VALID_FROM_QUARTER_PRODUCT',␣
 ↪'VALID_TO_QUARTER_PRODUCT')
```

Training, Testen und Validieren des konsolidierten Datensatzes

Nachdem der Datensatz für die Modellierung aufbereitet wurde und somit den Anforderungen des Random Forest Algorithmus genügt, wird der Datensatz in einen Trainings-, Test-

sowie Validierungsdatensatz aufgeteilt. Auf Basis der Trainingsdaten wird das Klassifikationsmodell aufgebaut. Das Klassifikationsmodell wird anhand der Testdaten überprüft. Um sicherzustellen, dass das Modell keiner Überanpassung unterliegt, wird das Modell mithilfe des Validierungsdatensatzes kontrolliert.

Bildung eines Test-, Trainingsdatensatzes sowie eines Validierungsdatensatzes

Die Unterteilung des Datensatzes in Trainings, Test- und Validierungsdatensatz orientiert sich an einer Zeitspanne von einem Jahr. Der Trainings- sowie Testdatensatz umfasst das Jahr 2018, während sich der Validierungsdatensatz auf das Jahr 2019 beschränkt.

[10]:
```
# Unterteilung der Daten in Test- bzw. Trainingsdaten und␣
↪Validierungsdaten.
dfs_train_test = dfs_data.filter('STATUS_SINCE_PIPELINE between␣
↪20180101 and 20181231')
dfs_validate = dfs_data.filter('STATUS_SINCE_PIPELINE between 20190101␣
↪and 20191231')

# Unterteilung der Daten in Trainings- und Testdaten (30 % werden zum␣
↪Testen verwendet)).
(dfs_train, dfs_test) = dfs_train_test.randomSplit([0.7, 0.3], 5043)
print('Trainingsdatensatz: ' + str(dfs_train.count()))
print('Testdatensatz: ' + str(dfs_test.count()))
print('Validierungsdatensatz: ' + str(dfs_validate.count()))

# Zwischenspeichern des Test-, Trainings- und Validierungsdatensatzes.
path_train = './train.parquet'
path_test = './test.parquet'
path_validate = './validate.parquet'

dfs_train.write.parquet(path_train)
dfs_test.write.parquet(path_test)
dfs_validate.write.parquet(path_validate)

dfs_train = spark.read.parquet(path_train)
dfs_test = spark.read.parquet(path_test)
dfs_validate = spark.read.parquet(path_validate)
```

```
Trainingsdatensatz: 3448
Testdatensatz: 1544
Validierungsdatensatz: 3448
```

Konvertierung der Merkmale in binäre Vektoren

[11]:
```
# Die Merkmale werden in binäre Vektoren umgewandelt.
string_list = [
        'ACCOUNT_CLASSIFICATION',
        'RELATIONSHIP_STATUS',
        'INDUSTRY_ID_INPUT',
        'TOTAL_DEPLOYMENT_ID'
]

num_list = [
        'SALES_SEGMENT',
        'PLANNING_ENTITY',
        'STATUS_SINCE_PIPELINE',
        'CREATED_AT_PIPELINE',
        'I_AERODE',
        'I_AUTOM',
        'I_BANKIN',
        'I_CHEMIC',
        'I_CP',
        'I_CROSS',
        'I_DEFSEC',
        'I_ECO',
        'I_ENTERT',
        'I_HEALTH',
        'I_HIGHRE',
        'I_HIGHTE',
        'I_IMC',
        'I_INSURN',
        'I_LIFE_S',
        'I_MEDIA',
        'I_MILLPR',
        'I_MINING',
        'I_OIL_GA',
        'I_PROFSV',
        'I_PUB_SE',
        'I_RETAIL',
        'I_S4HANA',
        'I_SPORTS',
        'I_TELECO',
        'I_TRANSP',
        'I_UTILIT',
        'I_WHOLSA',
```

```
        'I_WSTENV',
        'I_Z_INT',
        'ONPREM',
        'PUBCLOUD',
        'TIME_IN_PIPELINE',
        'AVERAGE_TIME_IN_PIPELINE_ACCOUNT',
        'AVERAGE_TIME_IN_PIPELINE_PRODUCT',
        'BOUGHT_RATIO_PERCENTAGE',
        'NO_BUSINESS_CAPABILITES_RECOM',
        'NO_PRODUCTS'
]
stages = []
feature_list_assembler = []

# Alle kategorischen Variablen werden über den StringIndexer indiziert␣
 ↪und mittels des OneHotEncoder werden die Zeilen in binäre Vektoren␣
 ↪umgewandelt.
for feature in string_list:
        stringIndexer = StringIndexer(inputCol= feature, outputCol=␣
 ↪feature + '_INDEX', handleInvalid = 'skip')
        encoder = OneHotEncoder(inputCol=feature + '_INDEX',␣
 ↪outputCol=feature + '_OHE')
        feature_list_assembler += [feature + '_OHE']
        stages += [stringIndexer, encoder]

# Der Assembler fügt die Merkmale in einem Merkmalsvektor zusammen.
feature_list_assembler += num_list
assembler = VectorAssembler(inputCols = feature_list_assembler,␣
 ↪outputCol = 'features')
stages += [assembler]
```

Instantiierung des Random Forests

[12]:
```
# Instantiierung des Random Forests mit 1000 Entscheidungsbäumen
rf = RandomForestClassifier(labelCol = 'BOUGHT', featuresCol =␣
 ↪'features', numTrees = 1000)
stages += [rf]

# Erstellen der Pipeline
pipeline = Pipeline(stages = stages)
```

Trainieren des Modells mit den Trainingsdaten

[13]:
```
# Vorhersagen treffen mithilfe der Pipeline.
pipeline_model = pipeline.fit(dfs_train)
dfs_train = pipeline_model.transform(dfs_train)
```

Ermittlung von Vorhersagen mithilfe des Testdatensatzes

[14]:
```
# Entfernen der Zielvariable aus dem Testdatensatz.
dfs_test_unlabled = dfs_test.drop('BOUGHT')

# Die Daten werden mithilfe der Pipeline transformiert.
dfs_test_unlabled = pipeline_model.transform(dfs_test_unlabled)

# Zusammenführen der Testdaten mit der Zielvariable „BOUGHT".
dfs_test_model = dfs_test.join(dfs_test_unlabled, ['ACCOUNT_ID',␣
↪'LEVEL_CODE', 'PIPELINE_ID'], how = 'inner')
```

Ermittlung von Vorhersagen mithilfe des Validierungsdatensatzes

[15]:
```
# Entfernen der Zielvariable aus dem Validierungsdatensatz.
dfs_validate_unlabled = dfs_validate.drop('BOUGHT')

# Die Daten werden mithilfe der Pipeline transformiert.
dfs_validate_unlabled = pipeline_model.transform(dfs_validate_unlabled)

# Zusammenführen der Testdaten mit der Zielvariable „BOUGHT".
dfs_validate_model = dfs_validate.join(dfs_validate_unlabled,␣
↪['ACCOUNT_ID', 'LEVEL_CODE', 'PIPELINE_ID'], how = 'inner')
```

Training, Testen und Validieren des Hintergrunddatensatzes

Das Vorgehen der Modellierung des Hintergrunddatensatzes gleicht der Vorgehensweise zur Modellierung des konsolidierten Datensatzes. Damit der Hintergrunddatensatz und der konsolidierte Datensatz auf den gleichen Daten beruhen, werden dem konsolidierten Datensatz die Kontextdaten entnommen. Als Resultat entsteht der Hintergrunddatensatz entsteht.

Bildung eines Test-, Trainingsdatensatzes sowie eines Validierungsdatensatzes

Die Unterteilung des Datensatzes in Trainings, Test- und Validierungsdatensatz orientiert sich an einer Zeitspanne von einem Jahr. Der Trainings- sowie Testdatensatz umfasst das Jahr 2018, während sich der Validierungsdatensatz auf das Jahr 2019 beschränkt ist.

[16]:
```
# Löschen der Kontextspalten des konsolidierten Datensatzes.
context_columns = [
        'PRODUCTS',
        'INDUSTRY_ID',
        'SOLUTION_CAPABILITIES',
        'BUSINESS_CAPABILITIES',
        'BUSINESS_CAPABILITIES_RECOMMENDED',
        'BOUGHT_RATIO_PERCENTAGE',
        'NO_BUSINESS_CAPABILITES_RECOM',
        'NO_PRODUCTS',
        'TIME_IN_PIPELINE',
        'AVERAGE_TIME_IN_PIPELINE_ACCOUNT',
        'AVERAGE_TIME_IN_PIPELINE_PRODUCT',
        'TOTAL_DEPLOYMENT_ID',
        'I_AERODE',
        'I_AUTOM',
        'I_BANKIN',
        'I_CHEMIC',
        'I_CP',
        'I_CROSS',
        'I_DEFSEC',
        'I_ECO',
        'I_ENTERT',
        'I_HEALTH',
        'I_HIGHRE',
        'I_HIGHTE',
        'I_IMC',
        'I_INSURN',
        'I_LIFE_S',
        'I_MEDIA',
        'I_MILLPR',
        'I_MINING',
        'I_OIL_GA',
        'I_PROFSV',
        'I_PUB_SE',
        'I_RETAIL',
```

```
        'I_S4HANA',
        'I_SPORTS',
        'I_TELECO',
        'I_TRANSP',
        'I_UTILIT',
        'I_WHOLSA',
        'I_WSTENV',
        'I_Z_INT',
        'ONPREM',
        'PUBCLOUD'
]

for col in context_columns:
        dfs_train_background = dfs_train.drop(col)
        dfs_test_background = dfs_test.drop(col)
        dfs_validate_background = dfs_validate.drop(col)

# Zwischenspeichern der Datensätze.
path_train_background = './background_train.parquet'
path_test_background = './background_test.parquet'
path_validate_background = './background_validate.parquet'

dfs_train_background.write.parquet(path_train)
dfs_test_background.write.parquet(path_test)
dfs_validate_background.write.parquet(path_validate)

dfs_train_background = spark.read.parquet(path_train_background)
dfs_test_background = spark.read.parquet(path_test_background)
dfs_validate_background = spark.read.parquet(path_validate_background)
```

Konvertierung der Merkmale in binäre Vektoren

[17]:
```
# Die Merkmale werden in binäre Vektoren umgewandelt.
string_feature_list = [
        'ACCOUNT_CLASSIFICATION',
        'RELATIONSHIP_STATUS',
        'INDUSTRY_ID_INPUT'
]

num_list = [
        'SALES_SEGMENT',
        'PLANNING_ENTITY',
```

```
        'STATUS_SINCE_PIPELINE',
        'CREATED_AT_PIPELINE'
]
stages = []
feature_list_assembler = []

# Alle kategorischen Variablen werden über den StringIndexer indiziert␣
 ↪und mittels des OneHotEncoder werden die Zeilen in binäre Vektoren␣
 ↪umgewandelt.
for feature in string_feature_list:
        stringIndexer = StringIndexer(inputCol= feature, outputCol=␣
 ↪feature + '_INDEX', handleInvalid = 'skip')
        encoder = OneHotEncoder(inputCol=feature + '_INDEX',␣
 ↪outputCol=feature + '_OHE')
        feature_list_assembler += [feature + '_OHE']
        stages += [stringIndexer, encoder]

# Der Assembler fügt die Merkmale in einem Merkmalsvektor zusammen.
feature_list_assembler += num_list
assembler = VectorAssembler(inputCols = feature_list_assembler,␣
 ↪outputCol = 'features')
stages += [assembler]
```

Instantiierung des Random Forests

[18]:
```
# Instantiierung des Random Forests mit 1000 Entscheidungsbäumen.
rf = RandomForestClassifier(labelCol = 'BOUGHT', featuresCol =␣
 ↪'features', numTrees = 1000)
stages += [rf]

# Erstellen der Pipeline.
pipeline = Pipeline(stages = stages)
```

Trainieren des Modells mit den Trainingsdaten

[19]:
```
# Vorhersagen treffen mithilfe der Pipeline.
pipeline_model = pipeline.fit(dfs_train_background)
dfs_train_background = pipeline_model.transform(dfs_train_background)
```

Ermittlung von Vorhersagen mithilfe des Testdatensatzes

[20]:
```
# Entfernen der Zielvariable aus dem Testdatensatz.
dfs_test_unlabled_background = dfs_test_background.drop('BOUGHT')

# Die Daten werden mithilfe der Pipeline transformiert.
dfs_test_unlabled_background= pipeline_model.
 ↪transform(dfs_test_unlabled_background)

# Zusammenführen der Testdaten mit der Zielvariable „BOUGHT".
dfs_test_model_background = dfs_test_background.
 ↪join(dfs_test_unlabled_background, ['ACCOUNT_ID', 'LEVEL_CODE',␣
 ↪'PIPELINE_ID'], how = 'inner')
```

Ermittlung von Vorhersagen mithilfe des Validierungsdatensatzes

[21]:
```
# Entfernen der Zielvariable aus dem Validierungsdatensatz.
dfs_validate_unlabled_background = dfs_validate_background.
 ↪drop('BOUGHT')

# Die Daten werden mithilfe der Pipeline transformiert.
dfs_validate_unlabled_background = pipeline_model.
 ↪transform(dfs_validate_unlabled_background)

# Zusammenführen der Testdaten mit der Zielvariable „BOUGHT".
dfs_validate_model_background = dfs_validate_background.
 ↪join(dfs_validate_unlabled_background, ['ACCOUNT_ID', 'LEVEL_CODE',␣
 ↪'PIPELINE_ID'], how = 'inner')
```

Evaluation der Ergebnisse im Vergleich

Nachdem die Modelle für die konsolidierten Daten sowie Hintergrunddaten und Test- sowie Validierungsdurchläufe erstellt wurden, werden die Vorhersagen der vier entstandenen Modelle miteinander vergleichen. Die Modelle werden anhand der Konfusionsmatrix sowie der daraus zu berechnenden Metriken, der Treffergenauigkeit, der ROC-Kurve, der AUROC und der Wahrscheinlichkeitsdichtefunktion verglichen.

Konfusionsmatrix und Treffergenauigkeit der konsolidierten Daten

[22]:
```
# Ausgabe der Konfusionsmatrix und der Treffergenauigkeit der Testdaten
y_true_test = dfs_test_model.select(['BOUGHT']).collect()
y_pred_test = dfs_test_model.select(['prediction']).collect()

print(confusion_matrix(y_true_test, y_pred_test))
accuracy_evaluator =␣
 ↪MulticlassClassificationEvaluator(predictionCol='prediction',␣
 ↪labelCol='BOUGHT', metricName='accuracy')
accuracy = accuracy_evaluator.evaluate(dfs_test_model)
print('Treffergenauigkeit: %g' % (accuracy))
```

```
[[676  72]
[ 11 785]]
Treffergenauigkeit: 0.946244
```

[23]:
```
# Ausgabe der Konfusionsmatrix und der Treffergenauigkeit der␣
 ↪Validierungsdaten
y_true_validate = dfs_validate_model.select(['BOUGHT']).collect()
y_pred_validate = dfs_validate_model.select(['prediction']).collect()

print(confusion_matrix(y_true_validate, y_pred_validate))
accuracy_evaluator =␣
 ↪MulticlassClassificationEvaluator(predictionCol='prediction',␣
 ↪labelCol='BOUGHT', metricName='accuracy')
accuracy = accuracy_evaluator.evaluate(dfs_validate_model)
print('Treffergenauigkeit: %g' % (accuracy))
```

```
[[2100  236]
[  30 2300]]
Treffergenauigkeit: 0.942992
```

Konfusionsmatrix und Treffergenauigkeit der Hintergrunddaten

[24]:
```
# Ausgabe der Konfusionsmatrix und der Treffergenauigkeit der Testdaten
y_true_background_test = dfs_test_model_background.select(['BOUGHT']).
 ↪collect()
y_pred_background_test = dfs_test_model_background.
 ↪select(['prediction']).collect()

print(confusion_matrix(y_true_background_test, y_pred_background_test))
```

```
accuracy_evaluator =␣
 ↪MulticlassClassificationEvaluator(predictionCol='prediction',␣
 ↪labelCol='BOUGHT', metricName='accuracy')
accuracy = accuracy_evaluator.evaluate(dfs_test_model_background)
print('Treffergenauigkeit: %g' % (accuracy))
```

```
[[323 425]
[ 73 723]]
Treffergenauigkeit: 0.677461
```

[25]:
```
# Ausgabe der Konfusionsmatrix und der Treffergenauigkeit der␣
 ↪Validierungsdaten
y_true_background_validate = dfs_validate_model_background.
 ↪select(['BOUGHT']).collect()
y_pred_background_validate = dfs_validate_model_background.
 ↪select(['prediction']).collect()

print(confusion_matrix(y_true_background_validate,␣
 ↪y_pred_background_validate))
accuracy_evaluator =␣
 ↪MulticlassClassificationEvaluator(predictionCol='prediction',␣
 ↪labelCol='BOUGHT', metricName='accuracy')
accuracy = accuracy_evaluator.evaluate(dfs_validate_model_background)
print('Treffergenauigkeit: %g' % (accuracy))
```

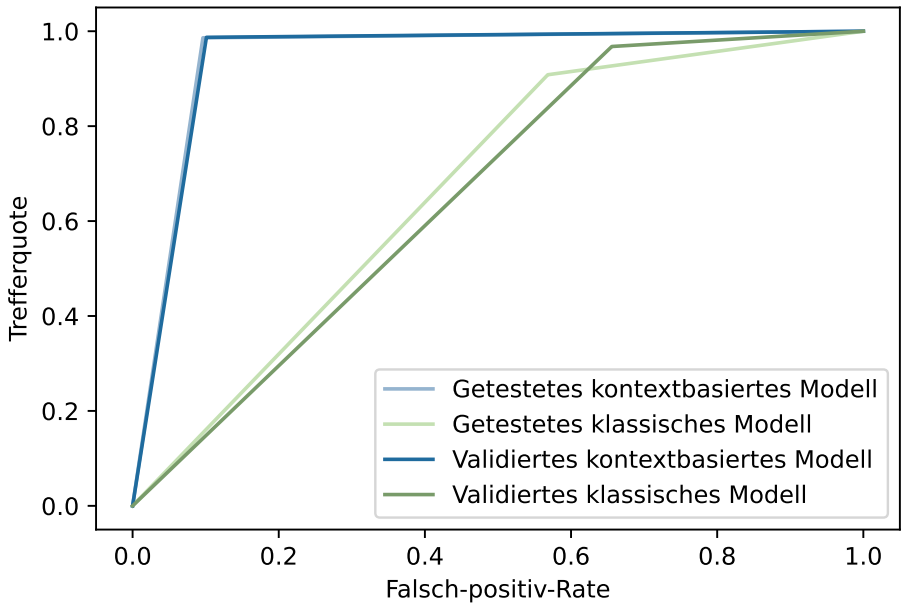
```
[[ 804 1532]
[  75 2255]]
Treffergenauigkeit: 0.655594
```

ROC-Kurven im Vergleich

[26]:
```
# Berechnung der ROC-Kurven.
fpr_test, tpr_test, thresholds = roc_curve(y_true_test, y_pred_test)
fpr_background_test, tpr_background_test, thresholds =␣
 ↪roc_curve(y_true_background_test, y_pred_background_test)

fpr_validate, tpr_validate, thresholds = roc_curve(y_true_validate,␣
 ↪y_pred_validate)
fpr_background_validate, tpr_background_validate, thresholds =␣
 ↪roc_curve(y_true_background_validate, y_pred_background_validate)

# Berechnung der AUROC.
```

```
auc_test = roc_auc_score(y_true_test, y_pred_test)
auc_background_test = roc_auc_score(y_true_background_test,␣
 ↪y_pred_background_test)
print('AUROC getestetes kontextbasiertes Modell: %.3f' % auc_test)
print('AUROC getestetes klassisches Modell: %.3f' %␣
 ↪auc_background_test)

auc_validate = roc_auc_score(y_true_validate, y_pred_validate)
auc_background_validate = roc_auc_score(y_true_background_validate,␣
 ↪y_pred_background_validate)
print('AUROC validiertes kontextbasiertes Modell: %.3f' % auc_validate)
print('AUROC validiertes klassisches Modell: %.3f' %␣
 ↪auc_background_validate)


# Zeichnen der ROC-Kurven.
plt.plot(fpr_test, tpr_test, label='Getestetes kontextbasiertes␣
 ↪Modell',  color='#95B4CF')
plt.plot(fpr_background_test, tpr_background_test, label='Getestetes␣
 ↪klassisches Modell',  color='#C4E0B2')

plt.plot(fpr_validate, tpr_validate, label='Validiertes␣
 ↪kontextbasiertes Modell',  color='#1F6B9E')
plt.plot(fpr_background_validate, tpr_background_validate,␣
 ↪label='Validiertes klassisches Modell',  color='#799B6A')

# Beschriftung der Axen und des Graphen.
plt.xlabel('Falsch-positiv-Rate')
plt.ylabel('Trefferquote')
plt.legend()

# Anzeigen des Graphen.
figROC = plt.gcf()
plt.show()

# Speichern der Grafik.
figROC.savefig('./ROC.pdf', dpi=100)
```

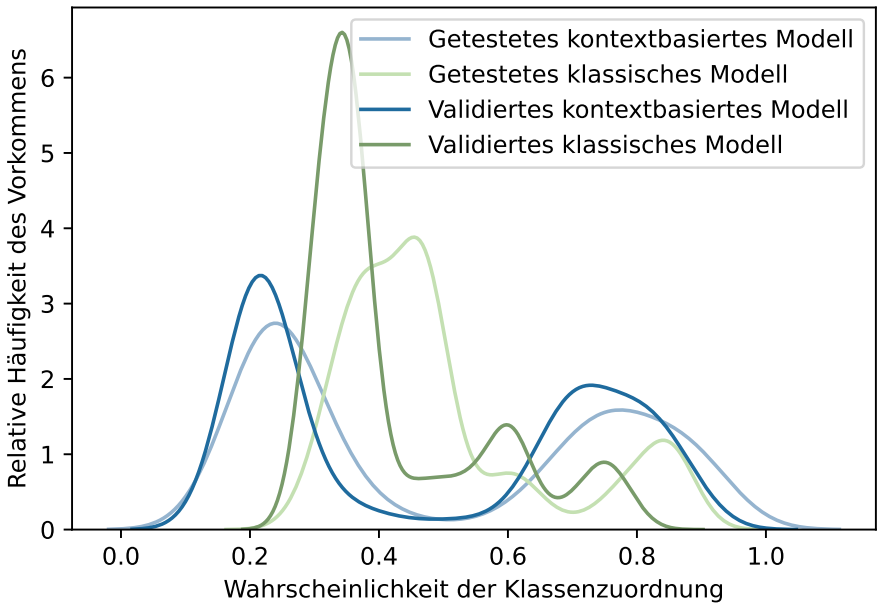
```
AUROC getestetes kontextbasiertes Modell: 0.945
AUROC getestetes klassisches Modell: 0.670
AUROC validiertes kontextbasiertes Modell: 0.943
AUROC validiertes klassisches Modell: 0.656
```



Wahrscheinlichkeitsdichtefunktionen im Vergleich

[27]:
```
# Speichern des ersten Eintrags der Spalte „probability" und Runden␣
 ↪des Eintrags.
firstelement = udf(lambda v:float(v[0]),FloatType())

# Berechnung der Anzahl des Vorkommens der Wahrscheinlichkeiten.
# Kontext-Test
dfs_prob_test = dfs_test_model.select([firstelement('probability')])
dfs_prob_test = dfs_prob_test.
 ↪withColumnRenamed('<lambda>(probability)','probablity_first')
probability_test = dfs_prob_test.select('probablity_first').collect()
val_test, count_test = np.unique(probability_test, return_counts=True)

# Hintergrund-Test
dfs_prob_background_test = dfs_test_model_background.
 ↪select([firstelement('probability')])
dfs_prob_background_test = dfs_prob_background_test.
 ↪withColumnRenamed('<lambda>(probability)','probablity_first')
probability_background_test = dfs_prob_background_test.
 ↪select('probablity_first').collect()
```

```
val_background_test, count_background_test = np.
 ↪unique(probability_background_test, return_counts=True)

# Kontext-Validierung
dfs_prob_validate = dfs_validate_model.
 ↪select([firstelement('probability')])
dfs_prob_validate = dfs_prob_validate.
 ↪withColumnRenamed('<lambda>(probability)','probablity_first')
probability_validate = dfs_prob_validate.select('probablity_first').
 ↪collect()
val_validate, count_validate = np.unique(probability_validate,␣
 ↪return_counts=True)

# Hintergrund-Validierung
dfs_prob_background_validate = dfs_validate_model_background.
 ↪select([firstelement('probability')])
dfs_prob_background_validate = dfs_prob_background_validate.
 ↪withColumnRenamed('<lambda>(probability)','probablity_first')
probability_background_validate = dfs_prob_background_validate.
 ↪select('probablity_first').collect()
val_background_validate, count_background_validate = np.
 ↪unique(probability_background_validate, return_counts=True)

# Ergebnisse grafisch darstellen lassen.
fig, ax = plt.subplots()
ax.set(xlabel='Wahrscheinlichkeit der Klassenzuordnung',␣
 ↪ylabel='Relative Häufigkeit des Vorkommens')
sns.kdeplot(val_test, ax=ax, color='#95B4CF', label='Getestetes␣
 ↪kontextbasiertes Modell', legend=True)
sns.kdeplot(val_background_test, ax=ax, color='#C4E0B2',␣
 ↪label='Getestetes klassisches Modell', legend=True)
sns.kdeplot(val_validate, ax=ax, color='#1F6B9E', label='Validiertes␣
 ↪kontextbasiertes Modell', legend=True)
sns.kdeplot(val_background_validate, ax=ax, color='#799B6A',␣
 ↪label='Validiertes klassisches Modell', legend=True)
ax.legend()
fig.savefig('./Wahrscheinlichkeitsdichtefunktionen.pdf')
```



Relevanz der einzelnen Merkmale

Zur Validierung der Kontextdaten wird der Einfluss der Kontextdaten auf die Vorhersagen des Modells analysiert. Es wird deutlich, dass lediglich das Attribut Industrie eine positive Auswirkung auf die Produktempfehlungen hat.

[28]:
```
# Berechnung der Relevanz der einzelnen Merkmale.
va = pipeline_model.stages[-2]
tree = pipeline_model.stages[-1]
list(zip(va.getInputCols(), tree.featureImportances))
```

[28]:
```
[('ACCOUNT_CLASSIFICATION_OHE', 0.004885783372537859),
('RELATIONSHIP_STATUS_OHE', 0.00456509362523879),
('INDUSTRY_ID_INPUT_OHE', 3.775287200926704e-05),
('TOTAL_DEPLOYMENT_ID_OHE', 0.02401804098305332),
('SALES_SEGMENT', 0.01632216467930783),
('PLANNING_ENTITY', 0.08938223269112996),
('STATUS_SINCE_PIPELINE', 0.026954396631976144),
('CREATED_AT_PIPELINE', 0.0006778609032316918),
('I_AERODE', 0.0006813366270440584),
('I_AUTOM', 5.945513913800263e-05),
('I_BANKIN', 0.0008937396110407713),
('I_CHEMIC', 0.01011667022675652),
```

```
('I_CP', 0.0005960748709193845),
('I_CROSS', 0.00044317895484331554),
('I_DEFSEC', 0.0005547236719339693),
('I_ECO', 0.0002966682720963612),
('I_ENTERT', 0.00028862457747374023),
('I_HEALTH', 0.0006262073058050284),
('I_HIGHRE', 0.0003360773049848121),
('I_HIGHTE', 0.0004916656073513368),
('I_IMC', 0.0002284220851651993),
('I_INSURN', 0.0003950727268778656),
('I_LIFE_S', 0.00039937040368662874),
('I_MEDIA', 0.0002896844373726698),
('I_MILLPR', 0.0002654588560434187),
('I_MINING', 0.00014212258085437876),
('I_OIL_GA', 0.0014195361506583286),
('I_PROFSV', 3.1536769047086224e-05),
('I_PUB_SE', 0.008231120084041968),
('I_RETAIL', 0.026777791602426472),
('I_S4HANA', 0.028627604705568185),
('I_SPORTS', 0.0),
('I_TELECO', 0.0),
('I_TRANSP', 0.0),
('I_UTILIT', 0.0),
('I_WHOLSA', 0.0),
('I_WSTENV', 0.0),
('I_Z_INT', 0.004289853223498061),
('ONPREM', 0.0),
('PUBCLOUD', 0.0),
('TIME_IN_PIPELINE', 0.0),
('AVERAGE_TIME_IN_PIPELINE_ACCOUNT', 0.0),
('AVERAGE_TIME_IN_PIPELINE_PRODUCT', 0.0),
('BOUGHT_RATIO_PERCENTAGE', 0.0),
('NO_BUSINESS_CAPABILITES_RECOM', 0.0),
('NO_PRODUCTS', 0.0)]
```

Ehrenwörtliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Thema: *Kontextbasierte Vorschläge von verkaufsfördernden Inhalten in der Enterprise-Softwareindustrie* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum Anna-Lena Blinken