Lehrstuhl für Informationstechnologie

Masterarbeit

# Vorhersage der Verpackungsgröße einer Lieferung in einem E-Commerce-Unternehmen mittels Machine Learning

Michael Josef Heininger, BSc

Mai 2021

**MONTANUNIVERSITÄT LEOBEN**
www.unileoben.ac.at

## EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt, und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Ich erkläre, dass ich die Richtlinien des Senats der Montanuniversität Leoben zu "Gute wissenschaftliche Praxis" gelesen, verstanden und befolgt habe.

Weiters erkläre ich, dass die elektronische und gedruckte Version der eingereichten wissenschaftlichen Abschlussarbeit formal und inhaltlich identisch sind.

Datum 23.05.2021

Unterschrift Verfasser/in
Michael Josef Heininger

## Gleichheitsgrundsatz

Aus Gründen der Lesbarkeit wurde in dieser Arbeit darauf verzichtet, geschlechtsspezifische Formulierungen zu verwenden. Es wird ausdrücklich festgehalten, dass die bei Personen verwendeten maskulinen Formen für beide Geschlechter zu verstehen sind.

# Danksagung

Mein Dank gebührt all jenen Personen, die mich im Zuge der Erstellung meiner Arbeit unterstützt und motiviert, sowie den Weg dorthin erst möglich gemacht haben.

Zuerst möchte ich mich beim Unternehmen niceshops GmbH bedanken, welches mir die Möglichkeit geboten hat, die Arbeit durchzuführen. Besonders möchte ich mich dabei bei meinem Betreuer Herrn Dipl.-Ing. Sebastian Mandl bedanken, welcher mich von Anfang an stets unterstützt hat. Ein weiterer Dank geht an meine Kollegen vom Team der Softwareentwicklung, die mir bei all meinen Fragen weitergeholfen haben.

Bei Herrn Prof. Dr. Ronald Ortner, der meine Masterarbeit betreut und begutachtet hat, möchte ich mich besonders bedanken. Er hatte stets ein offenes Ohr für meine Anliegen und seine konstruktive Kritik und sein Wissen haben mir bei der Erstellung der Arbeit sehr geholfen.

Zudem möchte ich mich bei Herrn Univ.-Prof. Dipl.-Ing. Dr.techn. Peter Auer für die Genehmigung der Verfassung des Themas der Masterarbeit am Lehrstuhl für Informationstechnologie bedanken.

Außerdem möchte ich mich bei meiner Schwester Eva Heininger, sowie bei Herrn Wilfried Mittendrein für das Korrekturlesen meiner Arbeit bedanken.

Abschließend gilt mein Dank meiner gesamten Familie sowie meinen Freunden. Insbesondere bedanke ich mich bei meinen Eltern Gertraud und Josef. Sie haben es mir durch ihre Unterstützung erst ermöglicht, mein Studium abzuschließen.

Michael Josef Heininger

Graz, im Mai 2021

# Kurzfassung

In einem E-Commerce-Unternehmen ist der Verpackungsprozess ein wesentlicher Bestandteil des Liefervorgangs. Wird dem Lagerarbeiter während dieses Prozesses die entsprechende Verpackungsgröße bereits vorgeschlagen, kann dadurch Zeit gespart werden.

Diese Masterarbeit wurde in Zusammenarbeit mit dem Unternehmen niceshops GmbH verfasst, welches Onlineshops in verschiedenen Produktsegmenten entwickelt. Ziel dieser Arbeit ist es zu untersuchen, ob es möglich ist, die Verpackungsgröße einer Lieferung mit einem Machine-Learning-Modell anhand der Daten historischer Lieferungen vorherzusagen. Es wurden die relevanten Daten für die Vorhersage analysiert und für jeden Onlineshop Modelle zur Vorhersage mit verschiedenen Algorithmen wie zum Beispiel k-nächste-Nachbarn, Random Forest oder Backpropagation trainiert und evaluiert. Die Genauigkeit der Vorhersagen dieser Modelle variiert von Shop zu Shop. Dies ist auf die unterschiedliche Produktvielfalt der Shops sowie auf die Datenqualität bei der Auswahl der richtigen Verpackungsgröße in früheren Lieferungen zurückzuführen.

Insgesamt lieferte der Random-Forest-Algorithmus die besten Resultate. Ein entsprechendes Modell wurde in einem Shop in den Verpackungsprozess integriert und wird dazu verwendet, dem Lagerarbeiter eine aus über 40 verschiedenen Verpackungsgrößen vorzuschlagen.

# Abstract

In an e-commerce company, the packaging process is an essential part of the delivery process. If the appropriate packaging size is proposed to the warehouse worker during this process, time can be saved. In addition, the prediction enables the estimation of the delivery volume in advance.

This master's thesis was carried out in cooperation with the company niceshops GmbH, which develops online shops in various product segments. The goal of this thesis is to examine whether it is possible to predict the packaging size of a delivery using a machine learning model based on the data of historical deliveries. The relevant data for the prediction was analyzed and prediction models were trained with different algorithms such as k-nearest neighbor, random forest or backpropagation. The models were evaluated for each online shop. The accuracy of the predictions made by these models varies from shop to shop. This is due to the different variety of products in the shops as well as the data quality regarding the selection of the correct packaging size in earlier deliveries.

Overall, the random forest algorithm provided the best results. A corresponding model was integrated into the packaging process of one of the shops and is used to propose one out of over 40 different packaging sizes to the warehouse worker.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Quelltextverzeichnis

# 1 Einleitung

Im Zeitalter der Digitalisierung entstehen durch die zunehmende Vernetzung von Prozessen in der Logistik große Mengen an verfügbaren Daten. *Maschinelles Lernen* oder *Machine Learning*[1] ermöglicht es, in diesem Datenberg versteckte Muster und wertvolle Informationen sichtbar zu machen. Machine Learning als Teilbereich der künstlichen Intelligenz entwickelt und analysiert Algorithmen, welche historische Daten als Eingabe verwenden und aus diesen lernen, um beispielsweise Prognosen zu tätigen.

Machine Learning findet im E-Commerce bereits großen Anklang und wird erfolgreich zur Erkennung von betrügerischem Verhalten, Personalisierung von Produktangeboten sowie zur Unterstützung im Kundensupport eingesetzt.[2] In dieser Arbeit wird eine weitere mögliche Einsatzmöglichkeit von Machine Learning zur Vorhersage der richtigen Verpackungsgröße für eine Lieferung mit ein oder mehreren Produkten beschrieben. Häufig gibt es in diesem Bereich noch keine Verpackungsgrößenvorschläge und der Lagerarbeiter wählt die entsprechende Verpackungsgröße auf Basis von Gefühl und Erfahrung aus. Die Verpackungsgrößenauswahl wird auch im österreichischen E-Commerce-Unternehmen niceshops GmbH in dieser Form durchgeführt.

## 1.1 Über niceshops GmbH

Die niceshops GmbH ist ein Unternehmen, welches Onlineshops in verschiedenen Produktsegmenten und mehreren europäischen Märkten entwickelt. Der Hauptsitz des Unternehmens befindet sich in der Steiermark in Saaz bei Feldbach. Hier befindet sich auch das Zentrallager, aus welchem jährlich mehr als 950.000 Pakete versendet werden. Neben dem Standort in Saaz verfügt das Unternehmen über weitere Standorte in Graz und Wien. Das Unternehmen beschäftigt über 400 Mitarbeiter.[3] Bei der niceshops GmbH handelt es sich um ein E-Commerce-Unternehmen mit starkem Fokus auf Logistik und Informationstechnologie. Die Software zur Betreibung der Onlineshops, Bearbeitung von Stammdaten und Bestellungen sowie zur Verwaltung des Lagers wird im Unternehmen selbst entwickelt. Das Unternehmen befindet sich seit Jahren in ständigem Wachstum. Um im hart umkämpften Online-Handel mithalten zu können, werden Prozesse laufend automatisiert und optimiert. Maschinelles Lernen ist eine Möglichkeit, um diese Automatisierung von Prozessen voranzutreiben.

[1] Im Rahmen dieser Arbeit werden die Begriffe maschinelles Lernen und Machine Learning synonym verwendet

[2] vgl. Große Holtforth 2018.

[3] vgl. Niceshops GmbH o. D.

## 1.2 Problemstellung

Im derzeitigen Verpackungsprozess muss der Lagerarbeiter beim Verpacken von Produkten einer Lieferung die verwendete Verpackungsgröße manuell auswählen. Diese Auswahl kann entweder durch einen Scan der *GTIN (Global Trade Item Number)* der Verpackung oder über ein User-Interface getätigt werden. Das ist ein manueller Schritt im Verpackungsprozess, welcher eine Entscheidung und Abschätzung des Lagerarbeiters bedingt und daher Zeit benötigt. Da es Tage gibt, an denen über 15.000 Pakete das Lager verlassen und verpackt werden müssen, ist die Zeit dieser Entscheidungsfindung nicht unwesentlich. Im folgenden Abschnitt wird der bestehende Verpackungsprozess genauer erläutert. Darüber hinaus wird auf die Gründe eingegangen, warum die Auswahl der Verpackungsgröße mittels Machine Learning automatisiert werden soll.

### 1.2.1 Ausgangslage

Um einen besseren Einblick zu bekommen, welche Prozesse und Prozessschritte durch das Vorhersagemodell betroffen sind, wird sowohl der dem Verpackungsprozess vorgelagerte Kommissionierprozess, als auch der Verpackungsprozess selbst erläutert. Im Unternehmen wird eine zweistufige Kommissionierung eingesetzt.[4] Die Entnahme der Produkte und die Zusammenstellung für einen spezifischen Kundenauftrag werden getrennt. In der ersten Stufe werden mehrere Kundenbestellungen zu einer sogenannten Pickliste auf Produktebene gruppiert. Diese Pickliste enthält die Produkte mit der Gesamtanzahl, welche für die Kundenbestellungen benötigt werden und definiert eine Reihenfolge, in der die Produkte entnommen werden. Dadurch wird der Lagerplatz des Produktes in einem Entnahmevorgang optimalerweise nur einmal angelaufen. Die entnommenen Produkte werden im nächsten Schritt zum Packtisch gebracht und der Lagerarbeiter startet den Verpackungsprozess. Am Packtisch befinden sich die vorgedruckten Rechnungen sowie die Paketlabel für alle Bestellungen, die in diesem Schritt verpackt werden sollen. Der Verpackungsprozess wird durch den Scan des Barcodes auf der Rechnung gestartet. Es werden am Monitor die benötigten Produkte für diese Lieferung angezeigt. Daraufhin werden diese Produkte zunächst auf dem Tisch zwischengelagert, um einen Überblick über die Menge und das Volumen der Produkte für die Lieferung zu bekommen. Der Lagerarbeiter muss nun die Entscheidung treffen, in welche Verpackung die Produkte verpackt werden sollen. Die entsprechende Größe wird im nächsten Schritt aus dem Stapel der Verpackungen über dem Packtisch entnommen und die Produkte zusammen mit der Rechnung in die Verpackung gegeben. Zuletzt wird das Paket verschlossen, mit einem Paketlabel versehen und weiter zum Lagerausgang gebracht. Von dort verlassen die Pakete mit dem jeweiligen

[4] vgl. Gleißner und Möller 2009.

Fahrzeug des Lieferdienstes das Lager.

### 1.2.2 Vorteile durch die Vorhersage

Aus der Beschreibung des Verpackungsprozesses geht hervor, dass die Entscheidung darüber, welche Verpackung entnommen werden soll, zu zusätzlichen Prozessschritten führt. Zum einen müssen die zu verpackenden Produkte zuerst auf den Packtisch geräumt werden, um eine Entscheidung treffen zu können. Zusätzlich muss der Lagerarbeiter überlegen, in welche Verpackungsgröße die Produkte verpackt werden können. Wird eine zu kleine Verpackungsgröße ausgewählt, müssen alle Produkte umgeräumt werden. Umgekehrt entsteht ungenutzter Freiraum und zusätzliches Transportvolumen, wenn eine zu große Verpackung ausgewählt wird. Dies soll durch eine Vorauswahl der Verpackungsgröße optimiert werden. Ist die Verpackungsgröße vorausgewählt, kann der Lagerarbeiter bereits im ersten Schritt die entsprechende Verpackungsgröße entnehmen und muss nicht überlegen, welche Größe benötigt wird. Zusätzlich müssen die Produkte nicht auf dem Packtisch zwischengelagert werden, sondern können direkt in die Verpackung gegeben werden. Es kann jedoch trotzdem passieren, dass eine zu große oder zu kleine Verpackung vorgeschlagen wird. Darauf kann der Lagerarbeiter reagieren und die richtige Verpackung auswählen.

Durch die Vorhersage der Verpackungsgrößen ist es auch möglich, das Transportvolumen vorherzusagen. Da vorab bekannt ist, welche Lieferungen das Lager verlassen, kann aus dem Volumen der vorhergesagten Verpackungsgrößen das gesamte Transportvolumen und -gewicht für jeden Lieferdienst bestimmt werden. Das ermöglicht dem Unternehmen die Transporte zu optimieren. Es kann definiert werden, wie viele Fahrzeuge von welchem Lieferdienst benötigt werden. Die Bestellung einer fix vorgegebenen Anzahl an Fahrzeugen auf Basis von Erfahrungswerten hat in der Vergangenheit einerseits dazu geführt, dass Lieferungen aufgrund von Platzmangel oder zu großem Gesamtgewicht das Lager nicht verlassen konnten. Anderseits verließen Transporter das Lager halb leer. Eine Vorhersage des Transportvolumens würde eine direkte Anbindung der Versanddienstleister ermöglichen. Dabei könnte das benötigte Volumen übermittelt und Fahrzeuge automatisch bestellt und bereitgestellt werden.

Weiters wird in Zeiten mit einer überdurchschnittlichen Anzahl an Lieferungen am Tag (wie zum Beispiel am Black Friday oder vor Weihnachten) die sonst verpflichtende Auswahl der Verpackungsgröße aufgrund der resultierenden Zeitersparnis deaktiviert. Dadurch wird das Gewicht der Verpackungen beim Gesamtgewicht nicht mit eingerechnet. Das führt dazu, dass es beim Lagerausgang zu großen Gewichtsdifferenzen für den Transport kommt. Weiters wird die Entnahme der Verpackungen nicht

aufgezeichnet, sodass eine kontinuierliche Bestandsführung nicht möglich ist. Wird die Verpackungsgröße vorausgewählt, muss die verpflichtende Auswahl in solchen Zeiten nicht mehr deaktiviert werden.

### 1.2.3 Warum Machine Learning?

Es stellt sich die Frage, warum Machine Learning verwendet werden soll, um die Verpackungsgröße vorherzusagen. Das beschriebene Problem kann auch als dreidimensionales Problem beim Verpacken von Behältern angesehen werden, bei dem eine bestimmte Menge von Produkten in einen Behälter mit beschränkten Abmessungen verpackt werden muss. Diese Problemstellung wird als *Behälterproblem* bezeichnet.[5] Bei den untersuchten Methoden zur Lösung dieses Problems müssen die Dimensionen Höhe, Breite und Tiefe der Produkte verfügbar sein. Ein Einsatz dieser Methoden ist allerdings keine Option, da im Unternehmen die Maße für die Produkte meist nicht bekannt sind. Darüber hinaus sind neben dem Volumen auch andere Eigenschaften der Produkte, wie zum Beispiel, ob ein Produkt zerbrechlich ist, relevant für die auszuwählende Verpackungsgröße. Somit ist die Auswahl der Verpackungsgröße ein komplexer Entscheidungsprozess, in den Eigenschaften von Produkten, Lieferung und Verpackung einfließen. Einer der Hauptvorteile von Machine Learning ist es, solche Zusammenhänge aus großen Datenmengen extrahieren zu können.[6]

Für die Implementierung eines Vorhersagesystems auf Basis von Machine Learning sind weiters keine zusätzlichen Investitionen notwendig. Die technologische Infrastruktur ist im Unternehmen bereits vorhanden, ebenso wie historische Daten vergangener Lieferungen inklusive der verwendeten Verpackungsgrößen.

## 1.3 Zielsetzung und Forschungsfrage

Im Rahmen dieser Masterarbeit soll geprüft werden, ob eine Vorhersage der Verpackungsgröße auf Basis von Machine Learning mit den im Unternehmen vorhandenen Daten möglich ist. Dabei sollen verschiedene Methoden und Algorithmen des maschinellen Lernens untersucht und beschrieben werden. Insbesondere soll identifiziert werden, welche dieser Methoden und Algorithmen für den Einsatz zur Verpackungsgrößenvorhersage geeignet sind. Die mit diesen Algorithmen gelernten Vorhersagemodelle müssen evaluiert werden und abschließend soll eines dieser Modelle in den bestehenden Prozessablauf eingebunden werden. Basierend auf der Zielsetzung lässt sich folgende Forschungsfrage mit den zugehörigen Unterfragen formulieren:

[5] vgl. Wäscher, Haußner und Schumann 2007.
[6] vgl. Manhart Klaus 2020.

Ist es möglich, mittels Machine Learning eine Vorhersage der entsprechenden Verpackungsgrößen für Lieferungen in einem E-Commerce Unternehmen zu treffen?

- Welche Machine-Learning-Systeme und -Algorithmen sind für die Vorhersage von Verpackungsgrößen geeignet?
- Welche Daten werden benötigt und wie müssen diese aufbereitet werden, um Machine-Learning-Algorithmen darauf anwenden zu können?
- Ab welcher Vorhersagegenauigkeit ist es möglich, ein trainiertes Modell in den bestehenden Verpackungsprozess zu integrieren? Kann diese Voraussetzung erreicht werden?

## 1.4 Vorgehensweise und Aufbau dieser Arbeit

Um diese Forschungsfrage zu beantworten, wurde zunächst eine Literaturrecherche durchgeführt, um die möglichen Machine-Learning-Algorithmen zum Trainieren eines Vorhersagemodells für Verpackungsgrößen zu identifizieren. Im Praxisteil wurden die Erkenntnisse aus dieser Literaturrecherche verwendet, um solche Modelle zu entwickeln.

Der Aufbau dieser Masterarbeit wird durch die im vorigen Kapitel festgelegte Forschungsfrage definiert.

Zuerst werden nach diesem einleitenden Kapitel in Kapitel 2 die Grundlagen des Machine Learnings erläutert. Dabei werden die relevanten Begriffe definiert, sowie verschiedene Machine-Learning-Systeme beschrieben. Weiters werden die einzelnen Schritte des Verpackungsprozesses dargelegt und Faktoren aufgezeigt, welche die Auswahl der entsprechenden Verpackungsgröße beeinflussen. Am Ende dieses Kapitels werden die Grundlagen des Machine Learnings und des Verpackungsprozesses kombiniert und die für die Vorhersage von Verpackungsgrößen relevanten Machine-Learning-Systeme erläutert. Dabei wird im Detail auf folgende Algorithmen zur Mehrklassenklassifikation eingegangen: k-nächste-Nachbarn, Naive Bayes, Entscheidungsbäume und Random Forest, Neuronale Netze sowie Support Vector Machines.

Kapitel 3 gliedert sich in die drei Schritte, die notwendig sind, um ein Machine-Learning-System zu entwickeln. Hierbei werden zunächst die Daten zu den Lieferungen und Verpackungsgrößen des Unternehmens analysiert und relevante Zusammenhänge identifiziert. Auf diesen Daten aufbauend werden anschließend Modelle zur Vorhersage mit den im vorherigen Kapitel beschriebenen Algorithmen trainiert und evaluiert. Schließlich wird beschrieben, wie ein Modell des Random-Forest-Algorithmus in einen

der Shops des Unternehmens integriert wurde.

Abschließend werden die wesentlichen Inhalte dieser Arbeit zusammengefasst. Darüber hinaus wird ein Ausblick gegeben, wie die im Rahmen dieser Arbeit erstellten Modelle zur Vorhersage von Verpackungsgrößen in Zukunft weiter eingesetzt und verbessert werden können.

# 2 Theoretische Grundlagen

In diesem Abschnitt werden die Grundlagen und Systeme des maschinellen Lernens erklärt und klassifiziert, um ein besseres Verständnis dieser Masterarbeit zu erlangen. Zusätzlich werden relevante Begriffe und Abläufe im Rahmen der Verpackungsplanung definiert. Abschließend werden auf Basis der vorhergehenden Kapitel die für die Vorhersage von Verpackungsgrößen relevanten Algorithmen kurz erklärt.

## 2.1 Machine Learning

Machine Learning oder maschinelles Lernen beschreibt im Allgemeinen die Generierung von Wissen aus Erfahrung und umfasst eine Sammlung von Lernalgorithmen mit welchen Modelle aus Daten gelernt werden. Diese Modelle können anschließend auf neue Daten angewendet und zum Beispiel dazu benutzt werden, um Vorhersagen für diese unbekannten Daten zu tätigen.[7] Eine Vorhersage der Verpackungsgröße für eine Lieferung ist daher eine mögliche Anwendung des maschinellen Lernens. Es gibt viele Definitionen für maschinelles Lernen. Im folgenden Text werden zwei mögliche Definitionen präsentiert.

### 2.1.1 Definition Machine Learning

Die erste Definition von Machine Learning wurde von Artur Samuel 1959 getätigt. Dieser beschreibt Machine Learning als

> „Forschungsgebiet, das Computer in die Lage versetzen soll, zu lernen, ohne explizit darauf programmiert zu sein."

Er kam zu dem Schluss, dass das Programmieren von Computern, um aus Erfahrungen zu lernen, letztendlich die Notwendigkeit eines Großteils des detaillierten Programmieraufwands beseitigen sollte.[8]

Tom Mitchell definierte maschinelles Lernen 1997 allgemein als jedes Computerprogramm, das seine Leistung bei bestimmten Aufgaben durch Erfahrung verbessert. Im Detail:

> „Ein Computerprogramm soll aus Erfahrung E in Bezug auf eine Klasse von Aufgaben T und Leistungsmaß P lernen, wenn sich seine Leistung bei Aufgaben T, gemessen durch P, mit Erfahrung E verbessert."[9]

[7] vgl. Döbel et al. 2018, S. 8.
[8] vgl. Samuel 1959.
[9] Mitchell 1997, S. 2.

### 2.1.2 Begriffsabgrenzung: Machine Learning, Artificial Intelligence und Deep Learning

Heutzutage gibt es viele verschiedene Begriffe, wenn es um neue Datenverarbeitungstechniken geht. Die Abgrenzung der Begriffe *künstliche Intelligenz (KI)*, *maschinelles Lernen (ML)* und *tiefes Lernen (DL)* ist dabei nicht immer klar und diese Begriffe werden oft synonym verwendet.[10] Generell gilt jedoch: Deep Learning ist eine Teilmenge des maschinellen Lernens und maschinelles Lernen ist eine Teilmenge der künstlichen Intelligenz. Mit anderen Worten: Alles maschinelle Lernen ist KI, aber nicht jede KI ist maschinelles Lernen. Analog dazu ist alles tiefe Lernen maschinelles Lernen, aber nicht alles maschinelle Lernen ist tiefes Lernen. Dieser Zusammenhang wird in Abbildung 1 dargestellt und die Begriffe im folgenden Abschnitt erklärt.

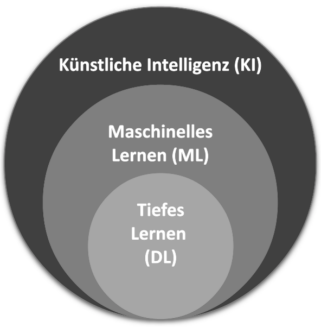


Abbildung 1: Künstliche Intelligenz, Maschinelles Lernen und Tiefes Lernen, Quelle: Eigene Darstellung

**Artificial Intelligence (AI), oder künstliche Intelligenz (KI)** hat es ermöglicht, Computersysteme zu schaffen, die Aufgaben ausführen können, welche normalerweise menschliche Intelligenz erfordern. Diese Aufgaben sind zum Beispiel visuelle Wahrnehmung, Spracherkennung, Entscheidungsfindung und Übersetzung zwischen Sprachen.[11] Für den Begriff "Künstliche Intelligenz"gibt es eine Reihe möglicher Definitionen, die von Kontext, Zeit und Anwendung abhängen. Eine eher allgemeine Definition ist „die von Maschinen demonstrierte Intelligenz im Gegensatz zur natürlichen Intelligenz von Menschen und Tieren". Da jedoch Algorithmen immer komplexere Aufgaben lösen, werden diejenigen, von denen angenommen wird, dass sie „Intelligenz" erfordern, manchmal aus dem Bereich der KI entfernt. Dies führt zur Behauptung, dass KI das ist, was noch nicht getan wurde.[12]

[10] vgl. Wehle 2017, S. 2.
[11] vgl. Tiwari, Tiwari und Tiwari 2018, S. 1.
[12] vgl. Visvikis et al. 2019, S. 2630.

Wie bereits zuvor erwähnt ist **maschinelles Lernen (ML), oder Machine Learning** ein Teilgebiet der Künstlichen Intelligenz und bezieht sich auf jede Art von Softwaresystem, das von selbst „lernen“ kann, ohne explizit von einem Menschen programmiert werden zu müssen. Heutzutage ist maschinelles Lernen ein weit verbreiteter Begriff, der viele Arten von Programmen umfasst, welche in Big Data Analytics und Data Mining verwendet werden. Letztendlich sind die „Gehirne“, die die meisten Vorhersageprogramme unterstützen - einschließlich Spamfilter, Produktempfehlungen und Betrugsdetektoren - Algorithmen für maschinelles Lernen. Üblicherweise wird zwischen drei verschiedenen Lernmodellen unterschieden: *Überwachtes Lernen*, *unüberwachtes Lernen* und *verstärkendes Lernen*.[13] Die Unterschiede werden im Detail in Kapitel 2.2.1 erläutert.

**Deep Learning (DL)** ist eine Form des maschinellen Lernens, bei der entweder überwachte, unüberwachte Algorithmen oder beides verwendet werden können. Obwohl es nicht unbedingt neu ist, hat Deep Learning in letzter Zeit einen Anstieg der Popularität erfahren, um die Lösung bestimmter Arten schwieriger Computerprobleme zu beschleunigen, insbesondere in den Bereichen Computer Vision und Verarbeitung natürlicher Sprache (NLP - Natural Language Processing). Ein Deep-Learning-Modell lernt die Features, welche wichtig sind, selbst, anstatt dass die relevanten Features manuell ausgewählt werden müssen. Die „Tiefe“ des tiefen Lernens ergibt sich aus den vielen Schichten, aus denen die Deep-Learning-Modelle bestehen. Dabei handelt es sich typischerweise um neuronale Netze. Ein Convolutional Neural Network (CNN) kann aus vielen Schichten von Modellen bestehen, wobei jede Schicht Eingaben von der vorherigen Schicht aufnimmt, verarbeitet und an die nächste Schicht ausgibt.[14]

## 2.2 Klassifikation von Machine-Learning-Systemen

In Bezug auf Machine Learning gibt es viele verschiedene Arten von Lernsystemen, sodass es nützlich ist, diese in Kategorien einzuteilen:

- Überwachtes, unüberwachtes oder verstärkendes Lernen
- Instanzbasiertes oder modellbasiertes Lernen
- Online- oder Batch-Lernen

Diese Kriterien sind nicht exklusiv, das heißt, sie können beliebig kombiniert werden. Im folgenden Abschnitt werden diese Kriterien im Detail untersucht.[15]

[13] vgl. Wehle 2017, S. 2.
[14] vgl. Wehle 2017, S. 3.
[15] vgl. Géron 2019, S. 7.

### 2.2.1 Überwachtes, unüberwachtes und verstärkendes Lernen

Maschinelle Lernsysteme werden hauptsächlich auf Basis der folgenden drei Gruppen unterschieden:

**Überwachtes Lernen**
Beim überwachten Lernen erhält der Computer Trainingsbeispiele, die mit dem richtigen Ergebnis, dem Label, gekennzeichnet sind. Der Zweck dieser Methode besteht darin, dass der Algorithmus lernen kann, indem er die tatsächlichen und prognostizierten Ergebnisse vergleicht, um Fehler zu finden und das Modell entsprechend modifiziert. Das überwachte Lernen verwendet daher Muster, um die Labels für zusätzliche, neue Daten vorherzusagen.[16] Überwachtes Lernen kann weiter eingeteilt werden in:

- Klassifizierung
- Regression

Die *Klassifizierung* ist ein überwachtes Lernproblem, bei dem eine kategorische Zielgröße vorhergesagt wird. Auf der anderen Seite ist die *Regression* ein Lernproblem, bei dem eine kontinuierliche Zielgröße vorhergesagt wird. Ein Klassifizierungsproblem kann mit Hilfe eines Spamfilters erläutert werden. Dieser wird durch viele Beispiele an E-Mails zusammen mit ihrer Kategorie (Spam, kein Spam) trainiert und soll für neue E-Mails bestimmen, ob es sich um Spams handelt oder nicht. Ein Beispiel für ein Regressionsproblem wäre der Bostoner Immobilienpreisdatensatz[17], bei dem die Eingaben Variablen sind, die ein Gebiet in Boston beschreiben, und die Ausgabe ein Hauspreis in Dollar ist.
Beispiele für überwachte Lernalgorithmen sind:

- k-nächste-Nachbarn-Algorithmus
- Lineare Regression
- Logistische Regression
- Support Vector Machines
- Entscheidungsbäume und Random Forests
- Neuronale Netze

Einige Algorithmen kommen speziell für die Klassifizierung (wie logistische Regression) oder Regression (wie lineare Regression) zum Einsatz. Andere können für beide Arten von Problemen mit geringfügigen Modifikationen

[16] vgl. Tiwari, Tiwari und Tiwari 2018, S. 3.
[17] vgl. Harrison Jr und Rubinfeld 1978.

verwendet werden (wie künstliche neuronale Netze).[18]

**Unüberwachtes Lernen**
Beim unüberwachten Lernen werden die Trainingsdaten ohne Label bereitgestellt, sodass der Lernalgorithmus Gemeinsamkeiten zwischen seinen Eingabedaten finden muss. Da in der Praxis häufig Daten ohne Label vorliegen, sind Methoden des maschinellen Lernens, die unbeaufsichtigtes Lernen ermöglichen, besonders wertvoll. Ohne eine richtige Antwort zu erhalten, betrachten unüberwachte Lernmethoden komplexe Daten, die umfangreich sind und versuchen, sie auf potenziell sinnvolle Weise zu organisieren.[19] Es gibt viele verschiedene Typen des unüberwachten Lernens:

- Clustering
- Visualisierung und Dimensionsreduktion
- Ausreißererkennung
- Assoziationsanalyse

Die häufigste unbeaufsichtigte Lernaufgabe ist das *Clustering*. Clustering ist das Erkennen nützlicher Cluster von Trainingsbeispielen.[20] Clustering wird oft dafür eingesetzt, Kunden anhand von ihren Daten in Gruppen einzuteilen. Daraufhin kann zum Beispiel eine spezifische Marketingkampagne für eine Gruppe erstellt werden. Ein weiterer Anwendungsbereich des unbeaufsichtigten Lernens ist die *Visualisierung und Dimensionsreduktion*. Die Aufgabe besteht darin, von einem hochdimensionalen Raum in zwei oder drei Dimensionen zu projizieren.[21]
Beispiele für unüberwachte Lernalgorithmen sind:

- k-Means (Clustering)
- Principal Component Analysis, kurz PCA (Dimensionsreduktion)

**Verstärkendes Lernen (Reinforcement Learning)**
Beim verstärkenden Lernen interagieren Maschinen, oft auch Agenten genannt, mit ihrer Umwelt. Die Agenten können Aktionen ausführen und bekommen dafür Feedback in Form von Belohnungen oder Bestrafungen. Dadurch lernen die Agenten, welche Aktionen ausgeführt werden müssen, um das Feedback zu maximieren.[22] Ein Beispiel für ein Problem des verstärkenden Lernens ist das Spielen eines Spiels, bei dem der Agent das Ziel hat, eine hohe Punktzahl zu erzielen. Dabei kann der Agent Züge im

[18] vgl. Brownlee 2019a.
[19] vgl. Tiwari, Tiwari und Tiwari 2018, S. 3.
[20] vgl. Stuart J. Russell and Peter Norvig 2016, S. 694f.
[21] vgl. Bishop 2006, S. 3.
[22] vgl. Döbel et al. 2018, S. 10.

Spiel ausführen und erhält Feedback in Form von Punkten. Einige beliebte Algorithmen für das verstärkende Lernen sind:

- Q-Learning
- Temporal Difference Learning
- Deep Reinforcement Learning

Abbildung 2 gibt einen Überblick über die in diesem Kapitel behandelten Lerntypen des maschinellen Lernens.

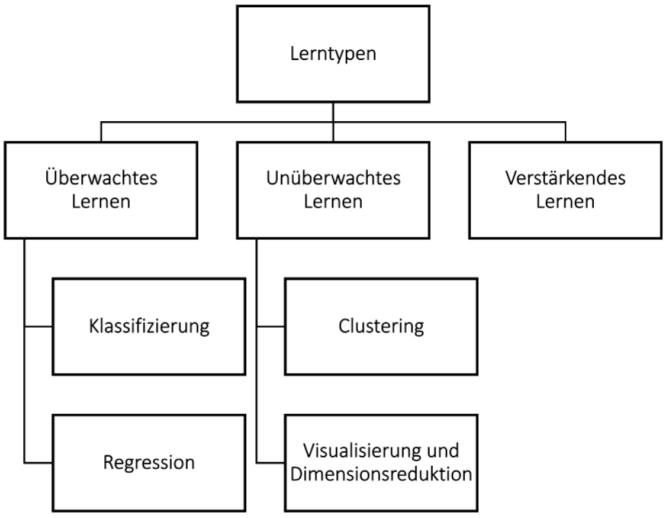


Abbildung 2: Drei Hauptkategorien (Lerntypen) von Machine Learning, Quelle: Eigene Darstellung

Die verschiedenen Lerntypen und damit verbundenen Algorithmen wurden in diesem Kapitel kurz erläutert und aufgezählt. In Kapitel 2.4 werden einige ausgewählte Algorithmen auf die Relevanz für die Vorhersage von Verpackungsgrößen untersucht.

### 2.2.2 Batch- und Online-Lernen

Ein weiteres Kriterium zur Klassifizierung von Systemen für maschinelles Lernen ist, ob das System schrittweise aus eingehenden Daten lernen kann oder nicht.

**Batch-Lernen**
Beim Batch-Lernen kann das System nicht schrittweise lernen: Es muss

unter Verwendung aller verfügbaren Daten trainiert werden. Dies nimmt im Allgemeinen viel Zeit und Rechenressourcen in Anspruch, sodass dies normalerweise offline erfolgt. Zuerst wird das System trainiert und in das Produktivsystem integriert. Dort ist es im Einsatz und lernt nichts Neues, sondern wendet nur das an, was es gelernt hat. Wenn das Batch-Lernsystem mit neuen Daten trainiert werden soll, muss eine neue Version des Systems von Grund auf den gesamten Datensatz trainieren (nicht nur auf den neuen Daten, sondern auch auf den alten Daten). Dabei wird das alte System gestoppt und durch das neue ersetzt. Das Lernen mit dem gesamten Datensatz kann jedoch viele Stunden dauern. Wenn sich das System an sich schnell ändernde Daten anpassen muss, wird eine reaktivere Lösung benötigt.

**Online-Lernen**
Beim Online-Lernen wird das System schrittweise trainiert, indem dem System Dateninstanzen nacheinander entweder einzeln oder in kleinen Gruppen, sogenannten Mini-Batches, zugeführt werden. Online-Lernen eignet sich für Systeme, die Daten als Datenstrom empfangen (wie zum Beispiel Aktienkurse) und sich anpassen müssen, um sich schnell oder autonom zu ändern. Ein wichtiger Parameter von Online-Lernsystemen ist, wie schnell sie sich an sich ändernde Daten anpassen: Dies wird als Lernrate bezeichnet. Wenn eine hohe Lernrate festgelegt wird, passt sich das System schnell an neue Daten an, vergisst aber auch schnell die alten Daten. Bei einer niedrigen Lernrate lernt das System zwar langsamer, ist aber auch weniger empfindlich gegenüber Rauschen in den neuen Daten oder nicht repräsentativer Datenpunkte (Ausreißer).[23]

### 2.2.3 Instanzbasiertes und modellbasiertes Lernen

Eine weitere Möglichkeit, maschinelle Lernsysteme zu kategorisieren, besteht darin, wie sie verallgemeinern. Bei den meisten Aufgaben des maschinellen Lernens geht es darum, Vorhersagen zu treffen. Die *Verallgemeinerung* bezieht sich auf die Fähigkeit eines maschinellen Lernsystems, bei neuen Daten eine gute Leistung bezüglich der Vorhersage zu treffen und nicht nur bei den Daten, auf denen es trainiert wurde. Dies bedeutet, dass das System anhand von Trainingsbeispielen in der Lage sein muss, auf Beispiele zu verallgemeinern, die es noch nie zuvor gesehen hat. Es gibt zwei Hauptansätze zur Verallgemeinerung: instanzbasiertes Lernen und modellbasiertes Lernen.[24]

**Instanzbasiertes Lernen**
Beim instanzbasierten Lernen wird die zu klassifizierende, neue Instanz

[23] vgl. zu diesem Abschnitt Géron 2019, S. 15ff.
[24] vgl. Géron 2019, S. 16.

mit den gegebenen Beispielen verglichen, um eine Vorhersage zu treffen. Zur Klassifikation wird der Funktionswert eines gegebenen Beispiels, welches der neuen Instanz am ähnlichsten ist, genommen. Dies erfordert ein Maß für die Ähnlichkeit zwischen zwei Objekten. Die Lernphase bei instanzbasierten Verfahren ist gering und besteht oft nur aus dem Abspeichern der Beispiele. Deswegen werden solche Verfahren oft als *lazy learners* bezeichnet. Der eigentliche Aufwand beim instanzbasierten Verfahren entsteht bei der Klassifikation von neuen Instanzen, da diese mit allen gespeicherten Beispielen verglichen werden müssen.[25] Beispiele für instanzbasierte Lernalgorithmen sind der *k-nächste-Nachbarn-Algorithmus*, *Kernelmaschinen* und *RBF-Netzwerke*.

**Modellbasiertes Lernen**
Eine weitere Form der Verallgemeinerung ist das modellbasierte Lernen. Modellbasierte Lernalgorithmen verwenden die Trainingsdaten, um ein Modell zu erstellen, dessen Parameter aus den Trainingsdaten gelernt wurden. Der Unterschied zu instanzbasierten Lernverfahren besteht darin, dass die Parameter vorher feststehen und beim Training die richtigen Werte für diese Parameter gefunden werden.

In modellbasierten Lernalgorithmen können Regeln in Form eines Modells verallgemeinert werden, das gespeichert werden kann. Beim instanzbasierten Lernen erfolgt die Verallgemeinerung für jede Instanz einzeln, erst wenn diese neue Instanz eintrifft. Deshalb ist das Klassifizieren von neuen Instanzen beim modellbasierten Lernen in der Regel schneller als beim instanzbasierten Lernen.[26]

[25] vgl. Görz und Schneeberger 2010, S. 533.
[26] vgl. Tomar 2020.

## 2.3 Verpackungsprozess im Detail

In Kapitel 1.2.1 wurde der Kommissionier- und Verpackungsprozess überblicksmäßig beschrieben. In diesem Kapitel wird im Detail auf die einzelnen Schritte des Verpackungsprozesses als auch auf den Entscheidungsprozess der Auswahl der Verpackungsgröße eingegangen. Dazu werden zunächst einige grundlegende Begriffe definiert.

### 2.3.1 Begriffsdefinitionen

Der Prozess des Verpackens einer Lieferung von Produkten für einen Kunden umfasst mehrere Elemente, welche in ihren Eigenschaften und ihrem Zweck unterschieden werden müssen.

**Lieferung**
Eine Lieferung ist ein übergeordneter Begriff, welcher den Übergang von Waren von einem Lieferanten an einen Kunden beschreibt.

**Paketelement**
Ein Paketelement besteht allgemein aus dem Produkt mit den zugehörigen Produkteigenschaften und der Menge dieses Produkts in der Lieferung. Weiters werden Eigenschaften festgehalten, wie zum Beispiel, ob es sich um Frischware, eine Geschenkbox oder um Proben handelt.

**Rechnung**
Die Rechnung ist ein Dokument, welches der Lieferung beim Transport beigelegt wird. Es beinhaltet eine Auflistung der Anzahl der gelieferten Produkte, deren Preis und dient als Beweis, dass eine Sendung geliefert wurde.[27]

**Paketlabel**
Das Paketlabel wird auf das Paket geklebt. Darauf befindet sich ein Barcode, Bestellnummer sowie Informationen, wie Abhol- und Empfangsadresse.[28]

**Versandverpackung**
Die Versandverpackung dient dazu, dass die Paketelemente zusammen gruppiert und unversehrt zum Kunden transportiert werden können. Dazu wird oft auch Füllmaterial benötigt, um empfindliche oder zerbrechliche Ware zu schützen.[29]

Im Unternehmen gibt es über 40 verschiedene Versandverpackungen, welche in übergeordnete Typen gegliedert sind und dem Mitarbeiter zur

[27] vgl. LetMeShip 2021.
[28] vgl. LetMeShip 2021.
[29] vgl. Behrend 2019.

Auswahl stehen. Diese Typen werden in Tabelle 1 aufgelistet und erläutert.

| **Verpackungstyp** | **Beschreibung** |
|---|---|
| Karton | Standardverpackungen in einer Vielzahl von verschiedenen Größen |
| Palette | Mehrweg- oder Einwegpaletten für den Versand sehr großer, sperriger Produkte |
| Foodmailer | Spezielle Kartons in verschiedenen Größen für den Versand von Frisch- und Kühlware |
| Flaschenkarton | Spezielle Kartons für den Versand von Flaschen |
| Versandkuvert | Kuverts, welche häufig für den Versand von einzelnen, kleinen Produkten verwendet werden |
| Sonstiges | Verpackungen, welche nur für bestimmte Produkte in bestimmten Shops verwendet werden |

Tabelle 1: Vorhandene Verpackungstypen und ihre Beschreibung

**Paket**
Ein Paket beschreibt die Gesamtheit aller Paketelemente in einer Versandverpackung mit Füllmaterial, die zum Kunden gesendet wird.

### 2.3.2 Prozessschritte

Wie in der Unternehmensbeschreibung in Kapitel 1.1 erläutert, entwickelt die niceshops GmbH Onlineshops in verschiedenen Nischenmärkten. Diese Onlineshops unterscheiden sich stark in ihrem Produktsegment, welches von Naturkosmetik über Frischware bis hin zu Pools und Pferdefutter reicht. Jeder Shop ist für sich eigenständig. Das heißt, als Kunde ist es derzeit nicht möglich, Produkte verschiedener Shops in einer gemeinsamen Bestellung aufzugeben. Das führt dazu, dass alle Prozesse, welche eine Lieferung eines Shops betreffen, wie zum Beispiel die Kommissionierung und das Verpacken, immer im Kontext eines einzelnen Onlineshops ablaufen. Jede Lieferung kann genau einem Shop zugeordnet werden. Der Verpackungsprozess einer Lieferung eines Shops ist in Abbildung 3 dargestellt. In dieser Prozessbeschreibung wird davon ausgegangen, dass die Produkte für die zu verpackenden Lieferungen bereits aus den Regalen geholt wurden und neben dem Packtisch auf einem sogenannten Pickwagen zwischengelagert sind. Zusätzlich liegen die für die Lieferungen notwendigen Rechnungen und Paketlabel bereits am Packtisch.

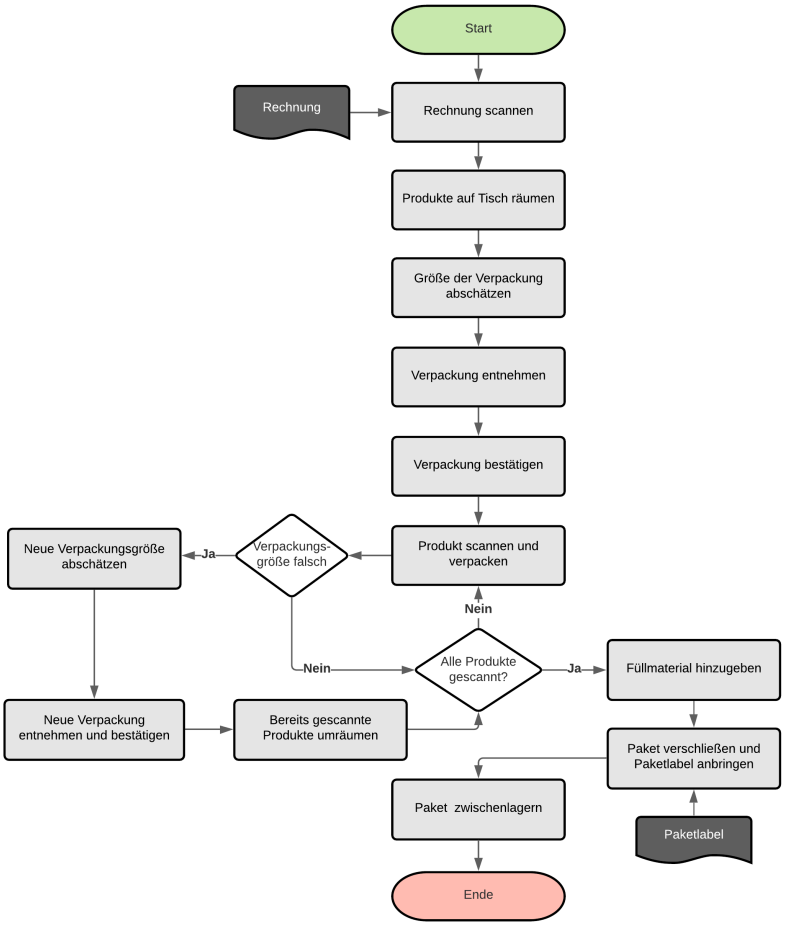


Abbildung 3: Verpackungsprozess einer Lieferung, Quelle: Eigene Darstellung

Der Prozess startet damit, dass der Mitarbeiter im Lager eine Rechnung beziehungsweise einen Lieferschein aus dem Stapel am Packtisch entnimmt und scannt. Wenn die Lieferadresse von der Rechnungsadresse abweicht, wird der Lieferschein gescannt und beigelegt, ansonsten die Rechnung. Aufgrund der Tatsache, dass es sich meist um die Rechnung handelt, wird der Start im Diagramm allgemein mit dem Scan der Rechnung definiert. Daraufhin öffnet sich die in Abbildung 4 dargestellte zentrale Benutzeroberfläche des Verpackungsprozesses.

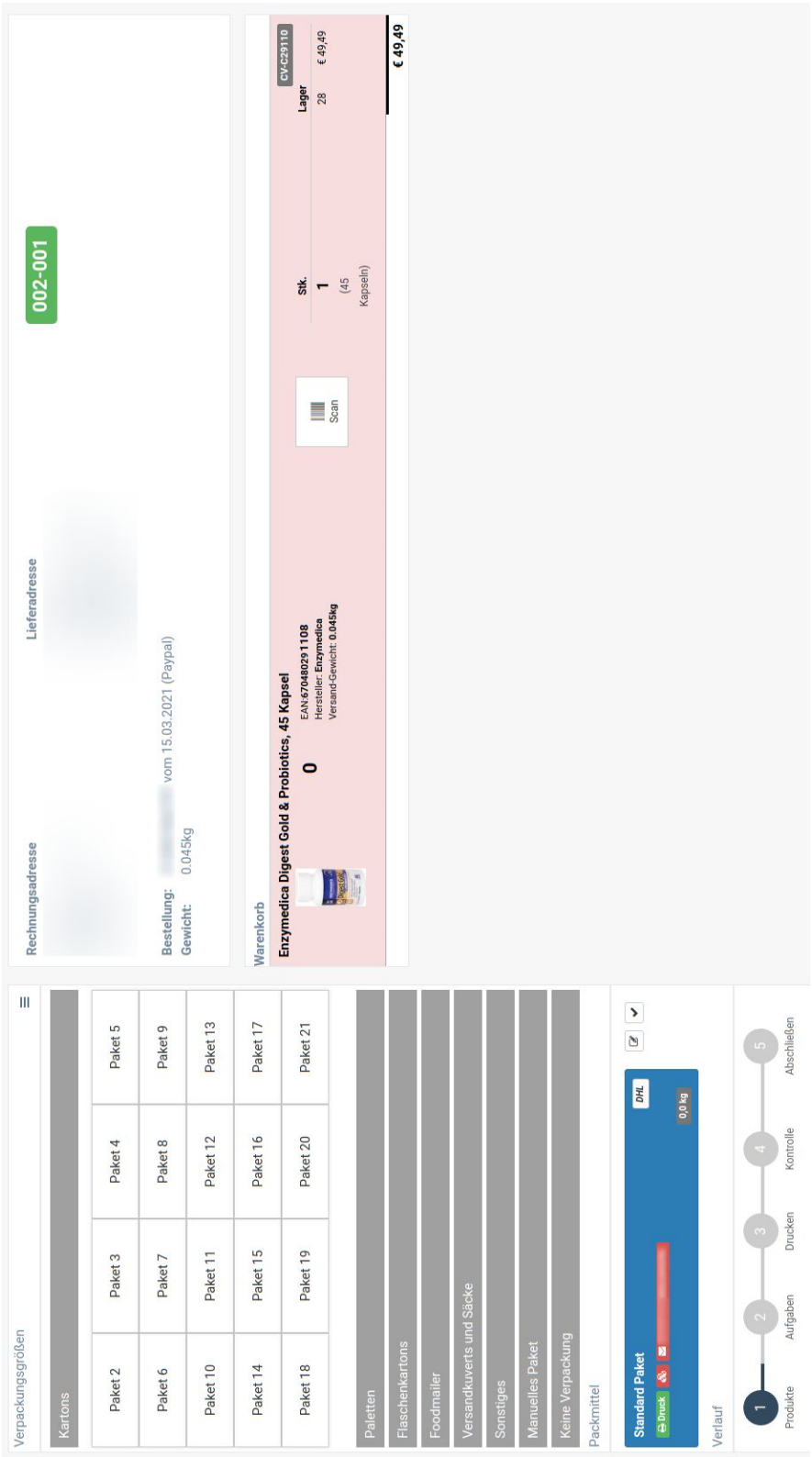


Abbildung 4: Benutzeroberfläche Verpackungsprozess, Quelle: niceshops GmbH

Im rechten oberen Bereich dieser Oberfläche werden dem Lagerarbeiter Standardinformationen der Lieferung wie zum Beispiel Rechnungsadresse, Lieferadresse, Gewicht und Bestellnummer angezeigt. Darunter befindet sich der Warenkorb, in welchem die zu verpackenden Produkte gemeinsam mit ihrer Anzahl und weiteren Informationen wie Titel, GTIN/EAN, Gewicht und Lagerstand aufgelistet sind. Im linken oberen Bereich befinden sich die verfügbaren Verpackungsgrößen. In dieser Abbildung sind nur die Verpackungsgrößen vom Typ Karton sichtbar. Alle anderen Verpackungsgrößen mit den zugehörigen Typen sind eingeklappt. Unter den Verpackungsgrößen sind die Eigenschaften des Pakets und der derzeitige Fortschritt im Prozess dargestellt.

Sobald diese Oberfläche angezeigt wird, beginnt der Lagerarbeiter die für die Lieferung benötigten Produkte aus dem Pickwagen neben dem Packtisch zu entnehmen und auf den Packtisch zwischenzulagern. Befinden sich alle benötigten Produkte auf dem Packtisch, beginnt der Entscheidungsprozess zur Auswahl der Verpackungsgröße. Dieser wird durch verschiedene Faktoren auf Basis der dargestellten Informationen beeinflusst, welche in Kapitel 2.3.3 näher untersucht werden. Hat der Arbeiter seine Entscheidung getroffen, entnimmt er die entsprechende Größe aus dem Stapel der verfügbaren Verpackungsgrößen. Die Bestätigung der Entnahme erfolgt entweder, falls vorhanden, durch einen Scan der GTIN auf der Verpackung oder durch einen Klick auf die entsprechende Verpackungsgröße in der Benutzeroberfläche. Nachdem die Versandverpackung ausgewählt wurde, beginnt der Lagerarbeiter die Produkte zu scannen und in die Versandverpackung zu geben. Gegebenenfalls müssen Produkte zuvor noch zusätzlich durch Füllmaterial geschützt werden. Wird beim Verpacken der Produkte bemerkt, dass eine falsche Verpackung ausgewählt wurde, müssen die Produkte in eine neue Verpackung umgeräumt werden. Je nachdem, ob im ersten Schritt eine zu kleine oder zu große Verpackung ausgewählt wurde, muss eine entsprechende größere beziehungsweise kleinere Verpackung ausgewählt werden. Die Auswahl der entsprechenden Versandverpackung muss demnach angepasst werden. Befinden sich alle zu verpackenden Produkte in der Versandverpackung, wird der noch vorhandene Freiraum mit Füllmaterial ausgefüllt. Danach wird das Paket verschlossen und das Paketlabel angebracht. Abschließend wird das fertig verpackte Paket in einem separaten Bereich zwischengelagert, wo es auf den Weitertransport zum Lagerausgang wartet.

Es sei zu erwähnen, dass der beschriebene Verpackungsprozess den Standardprozess darstellt. Es kann in der Praxis während des Prozesses vorkommen, dass eine geplante Lieferung in mehrere Verpackungen aufgeteilt werden muss. Ist dies der Fall, wird das Paket in der Datenbank mit dem Typ *label* versehen. Der Standardtyp ist *delivery*, da ein Paket nur den

Typ *label* erhält, wenn eine die Lieferung aufgeteilt werden muss. Wie oft dies der Fall ist, wird im Rahmen der Datenanalyse in Kapitel 3 untersucht.

**Zeitgliederung des Verpackungsprozesses**
Der Verpackungsprozess gliedert sich in mehrere Zeitabschnitte, welche zusammen die Gesamtdauer des Verpackungsprozesses ergeben. Die für die Berechnung der Gesamtzeit benötigten Zeiten und Variablen sind in Tabelle 2 dargestellt.

| **Abkürzung** | **Bezeichnung (engl.)** | **Bedeutung** |
|---|---|---|
| ST | Setup Time to Start Packing a New Order | Einrichtzeit, um das Verpacken einer weiteren Lieferung zu beginnen |
| TTSB | Total Time to Select and Prepare a Shipment Box Type for an Order | Gesamtzeit, um eine geeignete Versandverpackung auszuwählen und vorzubereiten |
| TTWP | Total Time to Wrap, Print and Stick Shipment Label for an Order | Gesamtzeit, um die Versandverpackung zu schließen, sowie das Paketlabel zu drucken und aufzubringen |
| TTSI | Time to Scan an Item and Put into the Box | Zeit, welche benötigt wird, um ein Produkt zu scannen und in die Versandverpackung zu geben |
| MIL | #Multi-Item Order Lines | Anzahl der zu verpackenden Produkte |

Tabelle 2: Gesamtzeit bestimmende Größen des Verpackungsprozesses, Quelle: Musagol 2017[30]

Die Gesamtverpackungszeit für eine Lieferung errechnet sich nach folgender Formel:

$$Gesamtverpackungszeit = ST + TTSB + TTWB + (TTSI * MIL)$$

Dieser Wert gibt die Gesamtzeit an, die erforderlich ist, um eine Lieferung mit mehreren Produkten zu verpacken. Diese Zeit setzt sich zusammen aus der Einrichtzeit für den Start, der Zeit für die Auswahl der Versandverpackung, des Scannens der Produkte, sowie des Verschließen des Pakets und Anbringen des Paketlabels.[31]

[30] vgl. Musaoglu 2017.
[31] vgl. Musaoglu 2017.

### 2.3.3 Faktoren, welche die Auswahl der Verpackungsgröße beeinflussen

Aus dem zuvor skizzierten Prozess geht hervor, dass die Auswahl der richtigen Versandverpackung ein wesentlicher Prozessschritt ist. Zum einen führt die Auswahl einer falschen Versandverpackung zu zusätzlichen Prozessschritten, da die Produkte umgeräumt werden müssen. Zum anderen benötigt die Entscheidung des Lagerarbeiters darüber, welche Verpackung gewählt werden muss, und die Bestätigung dieser zusätzliche Zeit. Der Entscheidungsprozess der Auswahl der richtigen Verpackungsgröße ist im Unternehmen unter anderem von folgenden Faktoren abhängig:

**Shop - Verfügbare Versandverpackungen**
Die verfügbaren Versandverpackungen werden shopübergreifend verwaltet, und in der Regel sind alle Verpackungen für jeden Shop verfügbar. Jedoch gibt es auch spezielle Versandverpackungen, welche nur in bestimmten Shops oder für bestimmte Produkte benötigt werden. Deshalb besteht die Möglichkeit, einzelne Größen in der Software für bestimmte Shops zu aktivieren oder zu deaktivieren. Wie in Abbildung 3 dargestellt und bereits erläutert, werden die verfügbaren Größen dem Lagermitarbeiter während des Verpackungsprozesses angezeigt und es können nur verfügbare Versandverpackungen ausgewählt werden.

**Produkte - Anzahl und Eigenschaften**
Das Hauptkriterium bei der Auswahl der entsprechenden Versandverpackung ist die Anzahl der jeweiligen Produkte und deren Eigenschaften. Diese Eigenschaften umfassen die Dimensionen, das Gewicht sowie weitere Merkmale eines Produkts, wie zum Beispiel die Tatsache, dass dieses zerbrechlich ist.[32] Zerbrechliche Produkte benötigen einen zusätzlichen Schutz in Form von Füllmaterial, was dazu führen kann, dass eine größere Verpackung verwendet werden muss.[33] Weiters gibt es in verschiedenen Shops Produkte, welche bereits versandfertig verpackt sind und einzeln an den Kunden versendet werden. In diesem Fall entfällt die Auswahl der Verpackungsgröße und es wird keine Verpackung ausgewählt. Zusätzlich zu den Dimensionen bestimmt auch die geometrische Form der Produkte die endgültige Verpackungsgröße, in der diese versendet werden. Sind Produkte zum Beispiel stapelbar, können diese in einem gemeinsamen Stapel versendet werden und benötigen dadurch weniger Platz. Enthält eine Lieferung Frischprodukte, darf die Kühlkette nicht unterbrochen werden und es muss eine Verpackung des Verpackungstyps *Foodmailer* ausgewählt werden. Diese Verpackungen in verschiedenen Größen wurden speziell für den Frischversand entworfen und es müssen zusätzlich zu den Produkten

[32] vgl. Knoll et al. 2019, S 577f.
[33] vgl. Behrend 2019.

auch Kühlakkus in die Verpackung gegeben werden. Dieser zusätzliche Platzbedarf der Kühlakkus muss bei der Auswahl berücksichtigt werden.

**Lieferland**
Werden Pakete nach Übersee transportiert, müssen diese besonders gut verpackt und die Produkte mit zusätzlichem Füllmaterial geschützt werden. Dadurch werden Transportschäden vermieden.

**Lieferdienst**
Im Unternehmen werden mit Hilfe verschiedener Lieferdienste und Logistikdienstleister die Pakete vom Lager zum Kunden gebracht. Bei einigen dieser Lieferdienste ist es jedoch nicht möglich, bestimmte Versandverpackungen zu verwenden. Manche Lieferdienste transportieren zum Beispiel keine Kartons, andere wiederum keine Paletten. Zusätzlich besteht im Unternehmen die Möglichkeit der Selbstabholung bei einem Lagerstandort oder in einem Geschäft eines Shops. Ist dies der Fall, ist der Lieferdienst *Selbstabholung* und die Produkte werden nicht in eine der verfügbaren Versandverpackungen gegeben, sondern in einem Papiersack ausgegeben.

**Mitarbeiter**
Eine weitere Einflussgröße auf die Auswahl der entsprechenden Versandverpackung ist der Mitarbeiter selbst, welcher den Verpackungsprozess durchführt. Je nach Shop und Erfahrung der Mitarbeiter kann es zu Unterschieden bezüglich der Auswahl der Verpackungsgröße kommen. In der Regel wird jedoch im Unternehmen eine einheitliche Auswahl der Versandverpackung mit möglichst wenig Freiraum bei entsprechendem Schutz der Produkte angestrebt.

### 2.3.4 Änderungen im Verpackungsprozess nach Implementierung des Vorhersagemodells

In Abschnitt 2.3.2 wurde der Verpackungsprozess mit den zugehörigen Schritten im Detail erläutert. Ziel dieser Arbeit ist es, durch den Einsatz eines Vorhersagemodells für die entsprechende Versandverpackung den Verpackungsprozess zu vereinfachen. Die in Abbildung 3 dargestellten Prozessschritte *Produkte auf Tisch räumen*, *Größe der Verpackung abschätzen* sowie *Verpackung bestätigen* sollen dadurch nicht mehr notwendig sein. Im Optimalfall ist die entsprechende Verpackungsgröße beim Start des Verpackungsprozesses vorausgewählt und der Mitarbeiter scannt die Produkte direkt vom Pickwagen in den ausgewählten Karton. Bei der Vorhersage einer falschen Verpackungsgröße führt dies zu denselben zusätzlichen Prozessschritten, wie im Fall, wo der Mitarbeiter im Lager eine falsche Verpackungsgröße auswählt. Dabei muss die tatsächliche Verpackungsgröße manuell ausgewählt und bestätigt werden. Der potentielle neue Verpackungspro-

zess mit Vorauswahl der Verpackungsgröße durch Machine Learning und die einhergehenden Schritte für den Mitarbeiter im Lager sind in Abbildung 5 dargestellt.

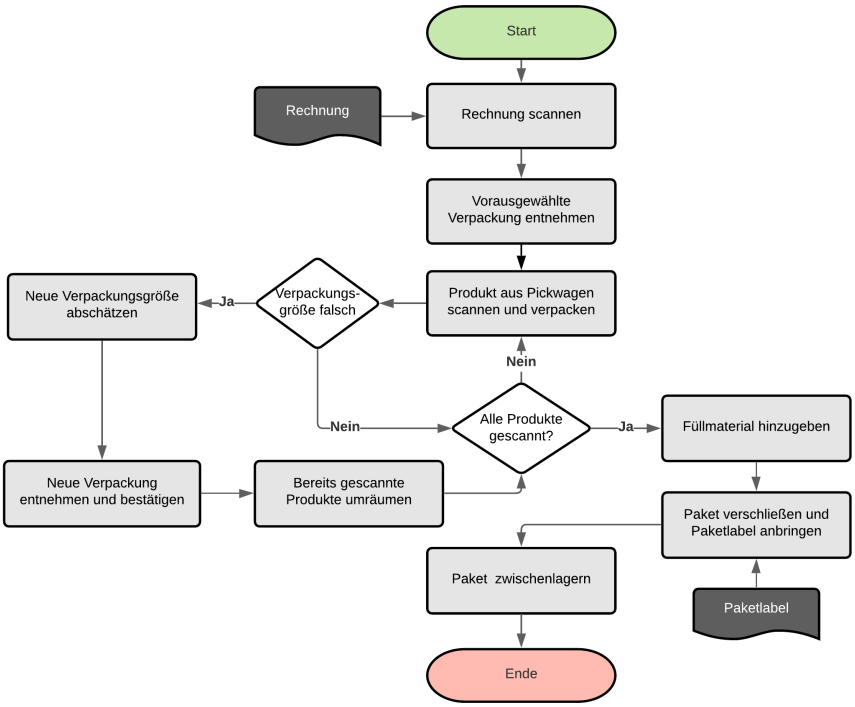


Abbildung 5: Verpackungsprozess einer Lieferung mit Vorhersagemodell, Quelle: Eigene Darstellung

Aufgrund der Implementierung des Vorhersagemodells und der vorhin genannten Tatsache, dass Prozesschritte des Lagerarbeiters automatisiert werden, wird erwartet, dass die in Abschnitt 2.3.2 genannte Gesamtverpackungszeit verringert wird. Dies ist jedoch nur der Fall, wenn die Vorhersagegenauigkeit groß genug ist, sodass die Zeitersparnis durch die Vorauswahl größer ist als die zusätzlich notwendige Zeit für die Korrektur einer falsch vorhergesagten Verpackungsgröße. In Bezug auf die Zeitgliederung wird speziell erwartet, dass die TTSB (Total Time to Select and Prepare a Shipment Box Type for an Order)[34], also die Gesamtzeit, um eine geeignete Versandverpackung auszuwählen und vorzubereiten, durch ein Vorhersagemodell verringert wird.

Um eine möglichst große Vorhersagegenauigkeit zu erreichen, muss das Machine-Learning-Modell die in Kapitel 2.3.3 diskutierten Faktoren

[34] vgl. Musaoglu 2017.

des Entscheidungsprozesses so gut wie möglich abbilden. Darüber hinaus kann ein Vorhersagemodell auf Basis von Machine Learning auch Zusammenhänge zwischen Eigenschaften der Produkte, Lieferung und Verpackungsgröße erkennen, welche womöglich nicht auf den ersten Blick ersichtlich sind. Dafür werden im nächsten Kapitel mögliche Machine-Learning-Modelle und -Algorithmen hinsichtlich ihrer Relevanz für eine Verpackungsgrößenvorhersage untersucht. In Abschnitt 3.2 werden einige dieser Modelle und Algorithmen implementiert und die Ergebnisse hinsichtlich der Vorhersagegenauigkeit untersucht.

## 2.4 Machine-Learning-Systeme für die Vorhersage einer Verpackungsgröße

In den beiden vorherigen Abschnitten wurden zunächst die Grundlagen des maschinellen Lernens erklärt und ein Überblick über die verschiedenen Systeme des maschinellen Lernens gegeben. Danach wurde der Verpackungsprozess beschrieben und mit Hilfe von Prozessdiagrammen visualisiert und erläutert, wie eine Vorauswahl der Verpackungsgröße den Prozess beeinflussen würde. Nun werden diese beiden Abschnitte kombiniert und Machine-Learning-Systeme vorgestellt, welche zum Einsatz für ein Vorhersagemodell von Verpackungsgrößen verwendet werden können.

Ziel dieser Arbeit ist es, zu prüfen, ob mittels Einsatz eines Machine-Learning-Systems eine Vorhersage der Verpackungsgröße für eine Lieferung möglich ist. Wie bereits beschrieben, definiert Tom Mitchell maschinelles Lernen als ein Computerprogramm, welches aus Erfahrung E in Bezug auf eine Klasse von Aufgaben und einem Leistungsmaß P lernt, wenn sich die Leistung bei den Aufgaben T, gemessen durch P, mit Erfahrung E verbessert.[35] Diese Definition enthält drei Größen: Die Aufgabe T, das Leistungsmaß P und die Erfahrung E. In Bezug auf die Problemstellung der Vorhersage einer Verpackungsgröße sind diese drei Größen folgende:

- **Aufgabe T:** Vorhersage einer Verpackungsgröße für eine Lieferung
- **Leistungsmaß P:** Prozent der korrekt vorhergesagten Verpackungsgrößen
- **Erfahrung E:** Datenbank, welche die Eigenschaften der Lieferungen und zugehörigen Verpackungsgrößen enthält.

Das heißt, ein Computerprogramm, welches lernt, die richtige Verpackungsgröße vorherzusagen, kann die Leistung, gemessen durch den Prozentsatz der korrekt vorhergesagten Verpackungsgrößen, durch die gewonnene Erfahrung aus historischen Daten von verpackten Lieferungen verbessern. Dafür werden

[35] vgl. Mitchell 1997, S. 2.

im Praxisteil in Kapitel 3 zunächst die historischen Daten, welche zur Vorhersage benötigt werden, definiert und im Detail analysiert. Um jedoch die für die Vorhersage relevanten Algorithmen einzuschränken, ist zu erwähnen, dass in der Datenbank für jede Lieferung die zugeordnete Verpackungsgröße als Klasse vorhanden ist. Dies wird in Abbildung 6 dargestellt.

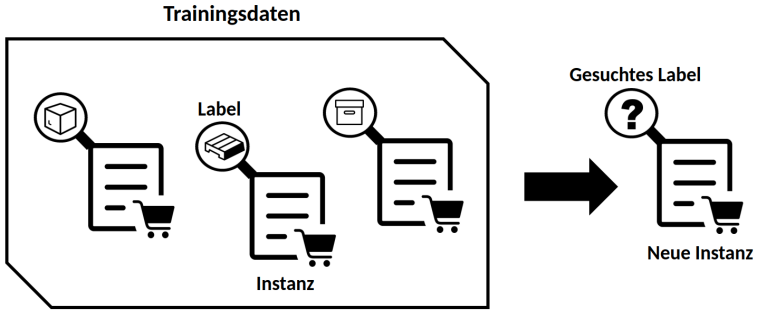


Abbildung 6: Machine-Learning-System für die Vohersage von Verpackungsgrößen, Quelle: In Anlehnung an Geron, 2019[36]

Wie in der Abbildung zu erkennen ist, ist jedem Trainingsdatensatz, jeder Instanz, eine Klasse als Label zugeordnet. Es handelt sich daher um überwachtes Lernen.[37] Die Klasse ist hierbei die Verpackungsgröße, welche der Lieferung zugeordnet ist. Da die Verpackungsgröße eine kategorische Zielgröße ist (Palette, Karton etc.) und das Vorhersagemodell mittels historischer Lieferungen und deren zugeordneter Verpackungsgröße trainiert wird, spricht man in diesem Fall von einem **Klassizifierungsproblem**. Genauer gesagt, handelt es sich hierbei um ein Lernproblem der *Mehrklassen-Klassifikation*. Im Gegensatz zur *binären Klassifikation*, bei welcher jede Instanz genau eine Klasse aus zwei möglichen Klassen erhalten kann (zum Beispiel Geschlecht: männlich/weiblich), kann bei der Mehrklassen-Klassifikation einer Instanz genau eine Klasse, aber aus mehr als zwei möglichen Klassen zugeordnet werden. In der vorliegenden Problemstellung kann einer neuen Lieferung eine aus über 40 verschiedenen Verpackungsgrößen zugeordnet werden. Sowohl die binäre als auch die Mehrklassen-Klassifikation sind Untergruppen der *Single-Label-Klassifikation*, da einer Instanz immer genau eine Klasse zugeordnet wird.[38]

Bei der Beschreibung des Verpackungsprozesses in Abschnitt 2.3.2 wurde darauf hingewiesen, dass in manchen Fällen die Lieferung in mehrere Pakete aufgeteilt werden muss. Dies würde wiederum bedeuten, dass einer

[36] vgl. Géron 2019, S. 8.
[37] vgl. Tiwari, Tiwari und Tiwari 2018, S. 3.
[38] vgl. Tsoumakas und Katakis 2007, S. 1.

Lieferung auch mehrere Verpackungsgrößen als Klasse zugeordnet werden können. Dies wird als *Multi-Label-Klassifikation* bezeichnet. Wie jedoch die Datenanalyse im Praxisteil zeigen wird, handelt es sich hierbei um Ausnahmefälle. Daher wird auf diese Art von Klassifizierungsproblem im folgenden Text nicht näher eingegangen. Es ist jedoch zu erwähnen, dass Multi-Label-Klassifikationsmethoden in zwei Kategorien eingeteilt werden können. Bei Methoden der Problemtransformation werden Multi-Label-Problemstellungen in Single-Label-Problemstellungen transformiert. Weiters gibt es Methoden der Anpassung von Single-Label-Algorithmen, sodass diese mit Multi-Label-Datensätzen umgehen können.[39]

Im folgenden Abschnitt liegt der Fokus auf der Single-Label-Klassifikation und speziell werden Algorithmen zur **Mehrklassen-Klassifikation (Multi-Class-Klassifikation)** untersucht. Dabei wird zwischen zwei Gruppen von Algorithmen unterschieden: Algorithmen, welche bereits mit mehr als zwei Klassen umgehen können, wie zum Beispiel *k-nächste-Nachbarn, Naive Bayes Klassifikatoren, Entscheidungsbäume (inklusive Random Forrest)* oder *neuronale Netze*. Andere Algorithmen, wie zum Beispiel *Support Vector Machines (SVMs)* sind nur binäre Klassifikatoren. Problemstellungen der Mehrklassen-Klassifikation können jedoch auch mit Hilfe von mehreren binären Klassifikatoren gelöst werden. Dabei kann zum Beispiel eine SVM eingesetzt werden.[40] [41] Bevor jedoch diese Algorithmen hinsichtlich der Problemstellung untersucht werden, werden im nächsten Abschnitt die Gemeinsamkeiten und Grundlagen von Machine-Learning-Algorithmen definiert.

### 2.4.1 Grundlagen der Machine-Learning-Algorithmen zur Klassifikation

Maschinelles Lernen, speziell überwachtes Lernen, kann so beschrieben werden, dass verfügbare Daten verwendet werden, um eine Zielfunktion $f$ zu lernen, welche Eingaben $x$ aus dem sogenannten *Instanzraum* $X$ auf Ausgaben $y$ abbildet.[42]

$$y = f(x) \tag{1}$$

Ziel ist es, die Zielfunktion so gut zu approximieren, dass bei neuen Eingaben $x$ die Ausgaben $y$ für diese Daten vorhergesagt werden können.[43]

[39] vgl. Tsoumakas und Katakis 2007, S. 3f.
[40] vgl. Géron 2019, S. 100.
[41] vgl. Aly 2005, S. 4.
[42] vgl. Brownlee 2019c.
[43] vgl. Shaikh 2018.

Für die Erklärung der Algorithmen zur Klassifikation in diesem Kapitel werden die Eingaben als Instanzen und die Ausgaben, Labels, als Klassen bezeichnet.[44] Eine *Instanz* ist ein Vektor an sogenannten *Features*. Ein *Instanzraum* ist der Raum aller möglichen Instanzen für eine Lernaufgabe. Der Instanzraum entspricht häufig einem geometrischen Raum, wobei jedes Feature einer Dimension entspricht.[45] Bei einem Klassifikator zur Vorhersage von Verpackungsgrößen kann es sich bei den Features $x = (x_1, ..., x_n)$ der Instanzen zum Beispiel um das Gewicht der Lieferung oder die Anzahl der Produkte in der Lieferung handeln. Bei der Klasse $y$ handelt es sich um die Verpackungsgröße, welche der Lieferung zugewiesen ist.

**Hypothese und Hypothesenraum**
Ein Modell, welches die Zielfunktion approximiert und Instanzen auf Klassen abbildet, wird als *Hypothese (h)* bezeichnet.[46] Beim maschinellen Lernen wird mit Hilfe von Algorithmen ein Raum möglicher Hypothesen durchsucht, der *Hypothesenraum (H)*, um jene Hypothese zu bestimmen, welche die Zielfunktion am besten approximiert.[47]

Die Wahl des Algorithmus und der Algorithmuskonfiguration umfasst die Wahl eines Hypothesenraums, von dem angenommen wird, dass er eine Hypothese enthält, die zumindest eine gute Annäherung für die Zielfunktion darstellt.[48]

**Verallgemeinerung**
Ein wichtiger Aspekt beim Lernen der Zielfunktion aus den Trainingsdaten ist, wie gut das Modell auf neue Daten verallgemeinert.[49] Die Verallgemeinerung wurde bereits in Abschnitt 2.2.3 erklärt und beschreibt die Fähigkeit eines Modells bei unbekannten Daten eine gute Leistung hinsichtlich der Vorhersage zu treffen und nicht nur bei den Daten, anhand der es trainiert wurde. Sie ist wichtig, da die gesammelten Daten meist nur eine Stichprobe, unvollständig und verrauscht sind.[50] Der sogenannte *Generalisierungsfehler* eines maschinellen Lernmodells wird bestimmt, indem die Leistung des Modells bei einem Testdatensatz von Beispielen gemessen wird, die getrennt vom Trainingdatensatz gesammelt wurden.[51]

[44] vgl. Utgoff et al. 2011a.
[45] vgl. Utgoff et al. 2011b.
[46] vgl. Brownlee 2019c.
[47] vgl. Mitchell 1997, S. 14f.
[48] vgl. Brownlee 2019c.
[49] vgl. Brownlee 2019b.
[50] vgl. Brownlee 2019b.
[51] vgl. Goodfellow, Bengio und Courville 2016, S. 110.

Eine Gefahr beim maschinellen Lernen besteht in der *Überanpassung (engl. overfitting)*. Dabei wird ein Modell erstellt, das eine gute Leistung für die Daten erbringt, auf denen es trainiert wurde, jedoch schlecht auf neue Daten verallgemeinert.[52] Eine Überanpassung tritt auf, wenn das Modell im Vergleich zur Menge und zum Rauschen der Trainingsdaten zu komplex ist. Dieses Problem kann durch eine Auswahl eines einfacheren Modells, der Bereitstellung von mehr Trainingsdaten oder durch die Bereinigung der Daten verringert werden.[53]

*Unteranpassung (engl. underfitting)* ist das Gegenteil von Überanpassung. Sie tritt auf, wenn das Modell zu einfach ist, um die zugrunde liegende Struktur der Daten zu lernen. Dies kann auch vorkommen, wenn wesentliche Features nicht berücksichtigt werden. Eine Auswahl von weiteren Features und die Verwendung eines komplexeren Modells mit mehr Parametern kann diesen Effekt reduzieren.[54]

Einige Algorithmen sind anfälliger für Überanpassungen als andere. Diese und andere Tatsachen werden in den folgenden Kapiteln erörtert, in denen einige Machine-Learning-Algorithmen vorgestellt werden.

### 2.4.2 k-nächste-Nachbarn-Algorithmus

Der k-nächste-Nachbarn (kNN) Algorithmus wurde in Abschnitt 2.2.3 als ein Algorithmus des instanzbasierten Lernens angeführt. Er basiert auf dem Prinzip, dass die Instanzen innerhalb eines Datensatzes in der Nähe von anderen Instanzen mit ähnlichen Feature-Werten liegen.[55] Sind diese Instanzen mit einer Klasse versehen, kann die Klasse einer nicht klassifizierten Instanz durch Betrachtung der Klassen ihrer k-nächsten-Nachbarn bestimmt werden. Die Klasse der nicht klassifizierten Instanz wird durch die häufigste Klasse dieser Nachbarn definiert.[56] Dieser Zusammenhang wird in Abbildung 7 dargestellt. Bei Betrachtung der 3-nächsten-Nachbarn (k=3) würde dem zu klassifizierenden Objekt die *Klasse 1* als Klasse zugewiesen werden, bei den 9-nächsten-Nachbarn (k=9) die *Klasse 2*.

In Bezug auf die Vorhersage der Verpackungsgröße einer Lieferung würde eine neue, nicht klassifizierte Lieferung die am häufigsten verwendete Verpackungsgröße der k-nächsten-Nachbarn als Klasse erhalten. Hierbei sei zu erwähnen, dass in Abbildung 7 nur drei verschiedene Klassen (Klasse 1, 2 und 3) und zwei verschiedene Features (x1 und x2) dargestellt sind.

[52] vgl. Grus 2019, S. 155.
[53] vgl. Géron 2019, S. 28.
[54] vgl. Géron 2019, S. 29.
[55] vgl. Cover und Hart 1967, S. 21.
[56] vgl. Kotsiantis 2007, S. 259.

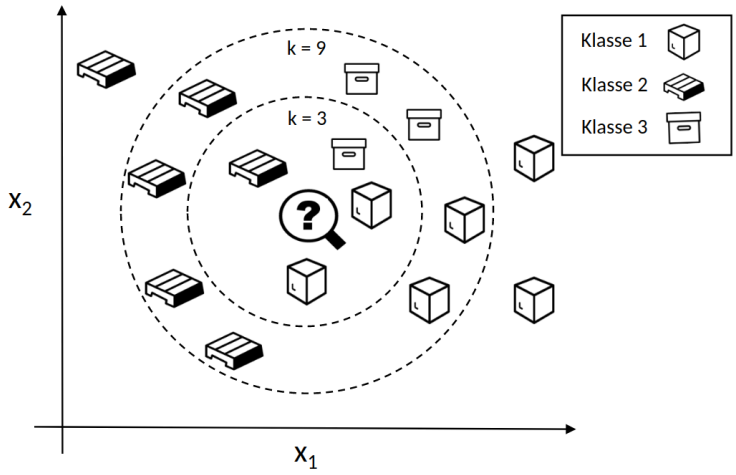


Abbildung 7: k-nächste-Nachbarn-Algorithmus, Quelle: Eigene Darstellung

Diese Features könnten, wie bereits erwähnt, das Gewicht der Lieferung und die Anzahl der Produkte in der Lieferung sein. Tatsächlich hat eine Lieferung mehr als zwei Features und ihr kann auch eine Klasse aus mehr als zwei möglichen Klassen zugeordnet werden. Um den kNN-Algorithmus anwenden zu können, benötigt man eine Größe, welche die Nähe zwischen zwei Instanzen ausdrückt. Instanzen können als Punkte innerhalb eines n-dimensionalen Instanzraums betrachtet werden. Jede dieser n Dimensionen entspricht einem der n Features, die zur Beschreibung einer Instanz verwendet werden. Die Nähe von zwei Instanzen kann durch den relativen Abstand zwischen diesen Instanzen unter Verwendung einer Abstandsmetrik bestimmt werden. Es gibt verschiedene Arten von Metriken, welche den Abstand zwischen zwei Instanzen beschreiben. Diese sind unter anderem der *Minkowski-*, *Manhatten-* und *Euklid-Abstand*.[57]

Ein wesentlicher Unterschied zwischen dem kNN-Algorithmus und anderen Algorithmen besteht darin, dass instanzbasierte Algorithmen für jede einzelne Instanz, die klassifiziert werden muss, eine eigene Approximation der Zielfunktion erstellen. Der kNN-Algorithmus bildet also keine explizite, allgemeine Hypothese $h$ bezüglich der Zielfunktion $f$.[58]

Beim kNN-Algorithmus müssen die Features aufgrund der Verwendung einer Abstandsmetrik numerisch und skaliert sein.[59] Der Klassifikator

[57] vgl. Kotsiantis 2007, S. 259.
[58] vgl. Mitchell 1997, S. 231f.
[59] vgl. Bhatia und Vandana 2010, S. 303.

kann sowohl die entsprechende Klasse, als auch die Wahrscheinlichkeit, mit welcher eine Instanz zu einer bestimmten Klasse gehört, bestimmen. Die Klassenwahrscheinlichkeit kann zum Beispiel durch den Anteil einer Klasse unter den Klassen der k-nächsten-Nachbarn bestimmt werden.[60]

Die **Stärken und Schwächen** des kNN-Algorithmus werden bedingt durch die Tatsache, dass dieser ein *lazy learning* Algorithmus ist. Wie in Abschnitt 2.2.3 erklärt, entsteht der Aufwand erst, wenn eine neue Instanz klassifiziert und mit allen gespeicherten Beispielen verglichen werden muss. Das führt dazu, dass das Training sehr schnell ist. Weiters ist dieser Algorithmus einfach zu verstehen und zu implementieren, da er nur zwei Parameter hat: die Anzahl der Nachbarn k und die Art der Abstandsmetrik. Die Klassifikation kann jedoch langsam sein, wenn der Datensatz sehr groß ist. Zudem ist der Algorithmus empfindlich gegenüber irrelevanten Features.[61]

### 2.4.3 Naive-Bayes-Klassifikator

Naive Bayes ist ein Klassifizierungsalgorithmus, welcher für binäre und Mehrklassen-Klassifizierungsprobleme eingesetzt wird.[62] Er basiert auf dem Satz von Bayes und wird als *naiv* bezeichnet, da er die vereinfachende Annahme enthält, dass die Features einer Instanz zur Klassifizierung dieser Instanz unabhängig sind.[63]

**Der Satz von Bayes**
Wie zuvor erwähnt, geht es beim maschinellen Lernen darum, die beste Hypothese (h) für die gegebenen Daten (D) aus dem Hypothesenraum (H) auszuwählen. Eine der einfachsten Möglichkeiten, eine Hypothese auszuwählen, ist die Auswahl der wahrscheinlichsten Hypothese auf Basis der vorliegenden Daten. Der Satz von Bayes bietet eine Möglichkeit, die Wahrscheinlichkeit einer Hypothese anhand von Vorwissen in Form von Daten zu berechnen. Er ist der Eckpfeiler der Bayes'schen Lernmethoden, da er eine Möglichkeit bietet, die *A-Posteriori-Wahrscheinlichkeit P(h|D)* aus der *A-Priori-Wahrscheinlichkeit P(h)* zusammen mit *P(D)* und *P(D|h)* zu berechnen. Der Satz von Bayes wird in der folgenden Gleichung dargestellt:

$$\mathrm{P(h|D)} = \frac{\mathrm{P(D|h)} \cdot \mathrm{P(h)}}{\mathrm{P(D)}} \tag{2}$$

- *P(h|D)* ist die Wahrscheinlichkeit der Hypothese h bei gegebenen Daten D.

[60] vgl. Jiang et al. 2007, S. 681.
[61] vgl. Bhatia und Vandana 2010, S. 303.
[62] vgl. Brownlee 2016, S. 83.
[63] vgl. Mitchell 1997, S. 197.

- *P(D|h)* ist die Wahrscheinlichkeit der Daten D, wenn die Hypothese h richtig ist.
- *P(h)* ist die Wahrscheinlichkeit, unabhängig von den Daten D, dass die Hypothese h richtig ist.
- *P(D)* ist die Wahrscheinlichkeit der Daten, unabhängig von der Hypothese h.

Beim Bayes'schen Lernen wird eine Hypothese *mit maximaler A-Posteriori-Wahrscheinlichkeit (engl. maximum a posteriori hypothesis – MAP)* gesucht. Nachdem die A-Posteriori-Wahrscheinlichkeiten für verschiedene Hypothesen berechnet wurden, kann die Hypothese mit der höchsten Wahrscheinlichkeit ausgewählt werden.

$$\begin{aligned} h_{MAP} &= \underset{h \in H}{\operatorname{argmax}} \frac{\mathrm{P(D|h)} \cdot \mathrm{P(h)}}{\mathrm{P(D)}} \\ &= \underset{h \in H}{\operatorname{argmax}} \mathrm{P(D|h)} \cdot \mathrm{P(h)} \end{aligned} \tag{3}$$

Im letzten Schritt wird der Term P(D) gestrichen, da er eine von h unabhängige Konstante ist.[64]

**Klassifizierung**
Naive Bayes ist ein Klassifikator, welcher auf dem Prinzip der maximalen A-Posteriori-Wahrscheinlichkeit basiert. Angenommen ein Klassifizierungsproblem hat die Klassen $\mathrm{k}_1$,...,$\mathrm{k}_i$ mit A-Priori-Wahrscheinlichkeiten $\mathrm{P}(\mathrm{k}_1)$,...,.$\mathrm{P}(\mathrm{k}_i)$ und die zu klassifizierenden Instanzen haben die Merkmale x = ($x_1$,...,$x_N$). Die Gleichung der maximalen A-Posteriori-Wahrscheinlichkeit kann nun folgendermaßen formuliert werden:

$$k_{MAP} = \underset{k \in K}{\operatorname{argmax}} \frac{\mathrm{P}(x_1,...,x_n|\mathrm{k}) \cdot \mathrm{P(k)}}{\mathrm{P}(x_1,...,x_N)} \tag{4}$$

Wie zuvor erwähnt, ist der Nenner für alle Klassen derselbe und kann aus der Gleichung entfernt werden. Weiters kann aufgrund der Tatsache, dass die Features als voneinander unabhängig angenommen werden, $\mathrm{P}(x_1,...,x_N|\mathrm{k})$ als $\mathrm{P}(x_1|\mathrm{K=k}) \cdot ... \cdot \mathrm{P}(x_N|\mathrm{K=k})$ formuliert werden. Das führt zu folgender Gleichung:

$$k_{MAP} = \underset{k \in K}{\operatorname{argmax}} \mathrm{P}(x_1|\mathrm{k}) \cdot \ ... \ \cdot \mathrm{P}(x_N|\mathrm{k}) \cdot \mathrm{P(k)} \tag{5}$$

Im Zuge der Klassifizierung einer neuen Instanz wird dieser jene Klasse k zugewiesen, welche die A-Posteriori-Wahrscheinlichkeit maximiert.[65]

[64] vgl. zu diesem Abschnitt Mitchell 1997, S. 197ff.
[65] vgl. zu diesem Abschnitt Aly 2005, S. 3.

Bei der Vorhersage von Verpackungsgrößen werden die Wahrscheinlichkeiten der Klassen berechnet und nach dem Prinzip der maximalen A-Posteriori-Wahrscheinlichkeit jener Verpackungsgröße zugewiesen, welche die höchste Wahrscheinlichkeit bedingt.

Je nach Implementierung können die Features kategorisch als auch numerisch sein, müssen jedoch nicht skaliert werden. Wie beim kNN-Klassifikator kann die entsprechende Klasse und eine Wahrscheinlichkeit für die Klassenzugehörigkeit zurückgegeben werden.[66]

Eine **Stärke** des naiven Bayes-Klassifikators ist, dass nur eine kleine Menge an Trainingsdaten erforderlich ist, um die für die Klassifizierung erforderlichen Parameter zu bestimmen.[67] Weiters geht es einfach und schnell, eine Klasse von Testdatensätzen vorherzusagen, und der Klassifikator kann auch zur Mehrklassen-Klassifikation eingesetzt werden. Eine **Schwäche** dieses Klassifikators ist die Annahme unabhängiger Features. Tatsächlich ist diese Annahme in der Praxis nie erfüllt. Dies kann zu einer geringeren Genauigkeit der Vorhersage führen.[68] Jedoch funktioniert der Klassifikator oft auch gut, wenn diese Voraussetzung nicht erfüllt ist.[69]

### 2.4.4 Entscheidungsbäume (Random Forests)

Ein Entscheidungsbaum verwendet eine Baumstruktur, um eine Anzahl möglicher Entscheidungspfade und ein Ergebnis für jeden Pfad darzustellen.[70] Gelernte Bäume können auch als Wenn-Dann-Regeln angesehen und daher einfach interpretiert werden. Diese Lernmethoden gehören zu den beliebtesten Algorithmen und werden erfolgreich für verschiedene Aufgabenstellungen, wie zum Beispiel zur medizinischen Diagnose, angewendet.[71] Entscheidungsbäume sind sowohl zur binären als auch zur Mehrklassen-Klassifikation geeignet und können auch angepasst werden, um Regressionsprobleme zu lösen.[72]

Ein Entscheidungsbaum besteht im wesentlichen aus drei Arten von Knoten:

- Einem *Wurzelknoten* ohne eingehender Verbindung und mit keiner oder mehr ausgehenden Kanten.

[66] vgl. Scikit-learn developers (BSD License) o. D.
[67] vgl. Iqbal und Yan 2015, S. 949.
[68] vgl. Kotsiantis 2007, S. 258.
[69] vgl. Mitchell 1997, S. 181.
[70] vgl. Grus 2019, S. 189.
[71] vgl. Mitchell 1997, S. 52.
[72] vgl. Kingsford und Salzberg 2008, S. 2.

- *Innere Knoten*, von denen jeder genau eine eingehende Kante und zwei oder mehrere ausgehende Kanten hat.
- *Blätter*, von denen jedes genau eine eingehende Kante und keine ausgehenden Kanten hat.

**Klassifikation**
Jeder Knoten in einem Entscheidungsbaum repräsentiert ein Feature einer zu klassifizierenden Instanz, und jede ausgehende Kante repräsentiert einen Wertebereich, den der Knoten annehmen kann. Instanzen werden beginnend mit dem Wurzelknoten klassifiziert und anhand ihrer Feature-Werte getrennt.[73] Genauer wird einer Instanz eine Klasse zugewiesen, indem man den Pfad von der Wurzel bis zu einem Blatt gemäß den Feature-Werten folgt.[74] In einem Entscheidungsbaum wird jedem Blatt eine Klassenbezeichnung und -wahrscheinlichkeit zugewiesen. Die Wahrscheinlichkeit, dass eine Instanz zu einer bestimmten Klasse gehört, wird durch den Anteil der Instanzen dieser Klasse in einem Blatt bestimmt.[75] Abbildung 8 ist ein Beispiel eines Entscheidungsbaums für die in Tabelle 3 definierten Trainingsdaten und soll darstellen, wie die Klassifizierung einer Lieferung hinsichtlich der Verpackungsgröße mit einem Entscheidungsbaum funktionieren könnte.[76]

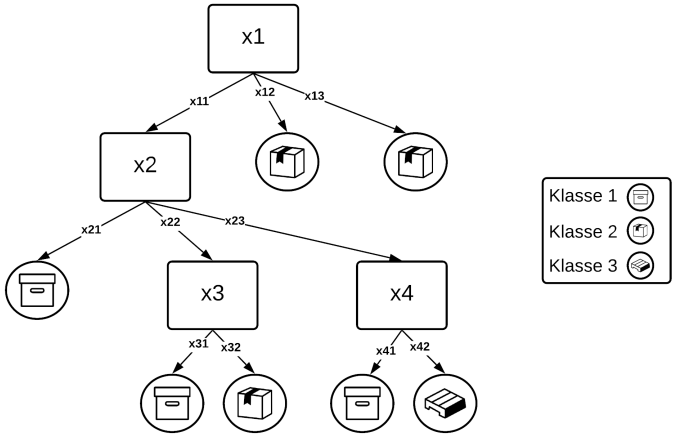


Abbildung 8: Beispiel eines Entscheidungsbaums, Quelle: In Anlehnung an Kotsiantis, 2007[77]

[73] vgl. Kotsiantis 2007, S. 251.
[74] vgl. Kingsford und Salzberg 2008, S. 1011.
[75] vgl. Géron 2019, S. 178.
[76] vgl. Kotsiantis 2007, S. 251.
[77] vgl. Kotsiantis 2007, S. 251.

| **x1** | **x2** | **x3** | **x4** | **Klasse** |
|---|---|---|---|---|
| x11 | x21 | x31 | x41 | Klasse 1 |
| x11 | x21 | x31 | x42 | Klasse 1 |
| x11 | x22 | x31 | x41 | Klasse 1 |
| x11 | x22 | x32 | x42 | Klasse 2 |
| x11 | x23 | x31 | x41 | Klasse 1 |
| x11 | x23 | x31 | x42 | Klasse 3 |
| x12 | x22 | x32 | x42 | Klasse 2 |
| x13 | x22 | x32 | x42 | Klasse 2 |

Tabelle 3: Trainingsdaten Entscheidungsbaum

Einer neuen Instanz mit den Feature-Werten x1 = x11, x2 = x22, x3 = x31 und x4 = x42 würde mit Hilfe des Entscheidungsbaums in Abbildung 8 die *Klasse 1* zugewiesen werden.

**Algorithmen**
Entscheidungsbaumalgorithmen erstellen einen Entscheidungsbaum aus einem bestimmten Datensatz. Dies wird als sogenanntes *NP-schweres Problem* angesehen, da es schwierig ist, einen Entscheidungsbaum zu finden, welcher möglichst viele Trainingsdaten korrekt klassifiziert.[78] Folglich sind heuristische Methoden zur Lösung des Problems erforderlich.[79] Die meisten Algorithmen, die zum Lernen von Entscheidungsbäumen entwickelt wurden, sind Variationen eines Algorithmus, der eine Top-Down-Suche durch mögliche Entscheidungsbäume verwendet, wie zum Beispiel *ID3, C4.5* und *CART*.[80] [81] Beim ID3-Algorithmus wird als Erstes am Wurzelknoten das beste Feature hinsichtlich einer statistischen Eigenschaft, die als *Informationsgewinn* bezeichnet wird, ausgewählt und als Test am Wurzelknoten des Baums verwendet. Der Informationsgewinn gibt an, wie gut ein Feature die Trainingsbeispiele gemäß ihrer Klasse trennt. Für jeden Wertebereich dieses Features wird dann ein dem Wurzelknoten untergeordneter Knoten erstellt, und die Trainingsbeispiele werden nach dem entsprechenden Knoten getrennt. Der gesamte Vorgang wird dann anhand der Trainingsbeispiele, die jedem untergeordneten Knoten zugeordnet sind, wiederholt. Jenes Feature, welches die Trainingsbeispiele am besten trennt, wird im entsprechenden Schritt ausgewählt.[82]

Der von Entscheidungsbaumalgorithmen wie dem ID3 durchsuchte

[78] vgl. Hancock et al. 1996, S. 115.
[79] vgl. Rokach und Maimon 2006, S. 167.
[80] vgl. Rokach und Maimon 2006, S. 167.
[81] vgl. Mitchell 1997, S. 55.
[82] vgl. zu diesem Abschnitt Mitchell 1997, S. 55.

Hypothesenraum ist die Menge aller möglichen Entscheidungsbäume. ID3 führt eine sogenannte *Hill-climbing-Suche* nach dem Entscheidungsbaum, welcher möglichst viele Trainingsdaten korrekt klassifiziert, in diesem Hypothesenraum durch. Ausgehend von einem leeren Entscheidungsbaum werden immer komplexere Entscheidungsbäume durchsucht.[83] Entscheidungsbäume erfordern nur sehr wenig Datenaufbereitung, können kategoriale und numerische Features verwenden und benötigen keine Skalierung oder Zentrierung der Features. Wie die zuvor beschriebenen Klassifikatoren können Entscheidungsbäume die Klasse sowie die Klassenwahrscheinlichkeit einer Instanz bestimmen.[84]

Wie anfangs erwähnt, ist die **Stärke** von Entscheidungsbäumen die Tatsache, dass sie meist besser interpretierbar sind als andere Klassifikatoren, wie zum Beispiel neuronale Netze oder Support Vector Machines. Weiters können neue Instanzen mit Hilfe von Entscheidungsbäumen schnell klassifiziert werden. Sie können sowohl Instanzen mit numerischen und kategorischen Features als auch Instanzen mit einigen fehlenden Feature-Werten klassifizieren. Ein **Schwäche** ist, dass kleine Änderungen der Eingabedaten zu großen Änderungen des erstellten Baums führen können.[85] Weiters neigen Entscheidungsbäume zur Überanpassung, was dazu führt, dass diese nicht gut auf unbekannte Instanzen verallgemeinern.[86] Dieses Problem kann mit Hilfe von Random Forests verringert werden.

**Random Forests**

Obwohl einzelne Entscheidungsbäume ausgezeichnete Klassifikatoren sein können, kann durch Kombinieren der Ergebnisse einer Sammlung von Entscheidungsbäumen meist eine erhöhte Genauigkeit erreicht werden. Sogenannte *Ensembles* von Entscheidungsbäumen gehören zu den leistungsstärksten Klassifikatoren. Random Forests sind eine Strategie zum Kombinieren von Entscheidungsbäumen.[87] Der Random-Forest-Algorithmus führt eine zusätzliche Zufälligkeit beim Erstellen der Bäume ein. Anstatt, wie zuvor erklärt, beim Aufteilen eines Knotens nach dem besten Feature unter allen Features zu suchen, wird nach dem besten unter einer zufälligen Teilmenge von Features gesucht. Dies führt zu einer größeren Baumvielfalt und ergibt im Allgemeinen ein insgesamt besseres Modell.[88]

[83] vgl. Mitchell 1997, S. 60f.
[84] vgl. Géron 2019, S. 178f.
[85] vgl. Kingsford und Salzberg 2008, S. 1f.
[86] vgl. Grus 2019, S. 216.
[87] vgl. Kingsford und Salzberg 2008, S. 3.
[88] vgl. Géron 2019, S. 197.

### 2.4.5 Neuronale Netze

Ein *künstliches neuronales Netzwerk* (engl. artificial neural network, ANN) oder auch *neuronales Netz* ist ein Vorhersagemodell, das von den Netzwerken biologischer Neuronen in unserem Gehirn inspiriert ist. Neuronale Netze bestehen aus künstlichen Neuronen und können auf eine Vielzahl von Problemstellungen, wie Handschrifterkennung und Gesichtserkennung angewendet werden. Sie bilden den Kern des *Deep Learnings*.[89] Künstliche Neuronen orientieren sich zwar an der Funktion realer Neuronen, jedoch werden diese vereinfacht dargestellt.[90] Sie bestehen im Wesentlichen aus zwei Hauptkomponenten: einer *Summierungsfunktion* und einer *Aktivierungsfunktion*. Dies ist in Abbildung 9 dargestellt.[91]

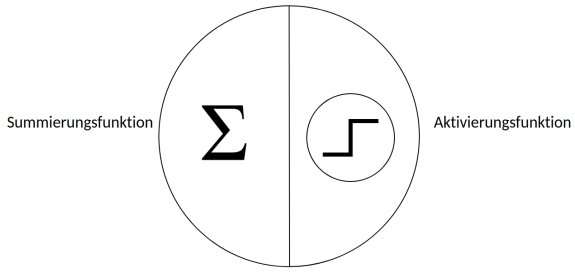


Abbildung 9: Künstliches Neuron, Quelle: In Anlehnung an Singh, 2021[92]

Die Summierungsfunktion ist für die Summierung aller Eingangssignale verantwortlich. Die Aktivierungsfunktion entscheidet anhand des Schwellenwerts, ob das Neuron auslöst oder nicht.

**Perzeptron**
Das Perzeptron ist eines der einfachsten künstlichen neuronalen Netze und wurde 1957 von Frank Rosenblatt entwickelt.[93] Die Features und die Ausgaben des Neurons, welche zur Bestimmung der Klasse verwendet werden, sind Zahlen und jedem Neuroneneingang ist ein Gewicht zugewiesen. Dieses Gewicht definiert den Beitrag des jeweiligen Features zur Perzeptronausgabe.[94] Das Neuron berechnet eine gewichtete Summe der Features, wendet dann eine Aktivierungsfunktion auf diese Summe an und gibt das Ergebnis aus.[95] In Abbildung 10 wird ein Beispiel der Berechnung

[89] vgl. Grus 2019, S. 227.
[90] vgl. Gershenson 2003, S. 2.
[91] vgl. zu diesem Abschnitt Singh 2021, S. 24f.
[92] vgl. Singh 2021, S. 25.
[93] Rosenblatt 1958.
[94] vgl. Mitchell 1997, S. 86.
[95] vgl. Géron 2019, S. 284.

eines künstlichen Neurons dargestellt.

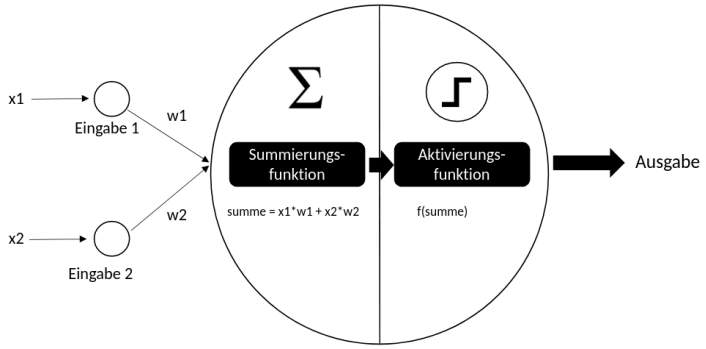


Abbildung 10: Berechnung eines künstlichen Neurons, Quelle: In Anlehnung an Singh, 2021[96]

Es existieren verschiedene Typen von Aktivierungsfunktionen. Diese sind unter anderem: *Schwellenwertfunktion*, *Sigmoidfunktion* und *Rectifier (ReLU)*.[97] [98] Ein Perzeptron kann zur einfachen linearen binären Klassifizierung verwendet werden, indem es eine lineare Kombination der Features berechnet. Je nachdem ob das Ergebnis einen gewissen Schwellenwert überschreitet, gibt es die eine oder die andere Klasse aus.[99]

Das Lernen eines Perzeptrons besteht aus der Auswahl von Werten für die Gewichte. Das bedeutet, der Hypothesenraum H, welcher für die Auswahl der Hypothese h berücksichtigt werden muss, ist die Menge aller möglichen realwertigen Gewichtsvektoren. Diese Gewichtsvektoren definieren zusammen mit dem Schwellwert eine Hyperebene im n-dimensionalen Instanzraum. Das Perzeptron weist eine entsprechende Klasse jenen Instanzen zu, die auf einer Seite der Hyperebene liegen, und eine andere Klasse jenen Instanzen, die auf der anderen Seite liegen. Die Klassifikation mit Hilfe von Hyperebenen ist nicht auf neuronale Netze beschränkt. Im nächsten Kapitel zu Support Vector Machines wird eine weitere Methode erläutert, die Klassen mit Hilfe von Hyperebenen bestimmt.

**Mehrlagiges Perzeptron und der Backpropagation-Algorithmus**
Perzeptronen können nur linear trennbare Instanzen klassifizieren. Wenn die Instanzen nicht linear trennbar sind, werden diese nicht richtig klassifiziert. Das *mehrlagige Perzeptron* (engl. mulitlayer perceptron, MLP) kann dieses

[96] vgl. Singh 2021, S. 25.
[97] vgl. Mitchell 1997, S. 86.
[98] vgl. Singh 2021, S. 25.
[99] vgl. Géron 2019, S. 285.

Problem mit Hilfe von Zwischenschichten lösen.[100]

Künstliche neuronale Netze können als gewichtete und gerichtete Graphen angesehen werden, in denen die Knoten die künstlichen Neuronen sind. Die Verbindungen zwischen den Neuronenausgängen und -eingängen entsprechen den Kanten mit dem zugewiesenen Gewicht im Graph. Künstliche neuronale Netze können, je nachdem wie die Ausgänge und Eingänge verbunden sind, in zwei Kategorien gruppiert werden: *Feedforward-Netze*, bei denen Graphen keine Schleifen aufweisen, und *rekurrente neuronale Netze*, bei denen Schleifen aufgrund von Rückkopplungen auftreten. Das mehrlagige Perzeptron ist eine Art eines Feedforward-Netzes, bei dem Neuronen in Schichten organisiert sind, zwischen denen unidirektionale Verbindungen bestehen.[101] Ein mehrlagiges Perzeptron besteht aus einer *Eingabeschicht*, ein oder mehreren *Zwischenschichten* und einer *Ausgabeschicht*. Sind zahlreiche Zwischenschichten vorhanden, wird dieses neuronale Netz als *tiefes neuronales Netz* bezeichnet.[102] In Abbildung 11 ist ein mehrlagiges Perzeptron dargestellt. Die *Softmax-Aktivierungsfunktion* wird im Abschnitt der Klassifikation mittels MLPs erklärt.

Der *Backpropagation-Algorithmus* (Rumelhart und McClelland, 1986)[103] wird zum Trainieren von MLPs verwendet. Das Training neuronaler Netze umfasst in erster Linie das Anpassen der Gewichte von Verbindungen, sodass das Modell mit einer höheren Genauigkeit vorhersagen kann. Für jede Trainingsinstanz tätigt der Backpropagation-Algorithmus zuerst eine Vorhersage und misst den Fehler. Dieser Fehler wird von der Ausgabeschicht zur Eingabeschicht zurückgegeben (Backpropagation) und der Fehlerbeitrag jeder Verbindung wird gemessen. Schließlich werden die Gewichte der Verbindungen angepasst, um den Fehler zu reduzieren. Bei der ersten Vorhersage werden alle Gewichte der Zwischenschichten zufällig initialisiert.[104]

**Klassifikation mittels MLP**

MLPs können sowohl für Regressions- als auch für Klassifizierungsaufgaben verwendet werden. Für ein binäres Klassifizierungsproblem wird nur ein einzelnes Neuron mit einer logistischen Aktivierungsfunktion in der Ausgabeschicht benötigt. Das Ergebnis dieser Aktivierungsfunktion ist eine Zahl zwischen 0 und 1, welche die Wahrscheinlichkeit der Klassenzugehörigkeit definiert. Weiters können MLPs problemlos zur Multi-Label-Klassifikation eingesetzt werden, indem die entsprechenden Klassen als Neuronen in

[100] vgl. Kotsiantis 2007, S. 255.
[101] vgl. Jain, Mao und Mohiuddin 1996, S. 34.
[102] vgl. Géron 2019, S. 298.
[103] Rumelhart, Hinton und Mcclelland 1986.
[104] vgl. Singh 2021, S. 32-34.

der Ausgabeschicht definiert werden. Es ist dabei zu beachten, dass in diesem Fall die Ausgabewahrscheinlichkeiten aller Neuronen in Summe nicht 1 ergeben müssen. Auf diese Weise kann das Modell eine beliebige Kombination von Klassen ausgeben. Liegt jedoch, wie bei der Vorhersage einer Verpackungsgröße, ein Mehrklassen-Klassifikationproblem vor, kann jeder Instanz nur eine aus drei oder mehr Klassen zugeordnet sein. Zur Darstellung dieses Problems wird pro Klasse (Verpackungsgröße) ein Ausgabeneuron und eine sogenannte *Softmax-Aktivierungsfunktion* in der Ausgabeschicht benötigt. Die Softmax-Aktivierungsfunktion stellt sicher, dass alle Ausgabewahrscheinlichkeiten der Neuronen zwischen 0 und 1 liegen und in Summe 1 ergeben. Dies ist erforderlich, wenn ein Mehrklassen-Klassifikationsproblem vorliegt.[105] In Abbildung 11 wird ein Beispiel eines MLP für die Vorhersage einer Verpackungsgröße skizziert.

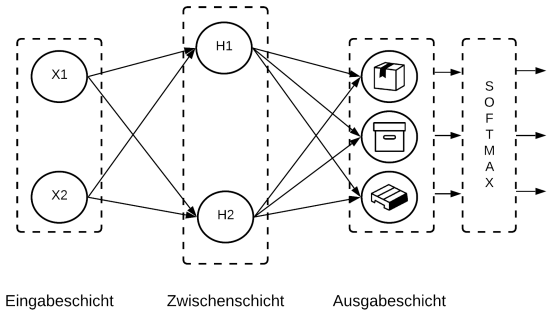


Abbildung 11: Neuronales Netz (MLP) mit Softmax-Funktion, Quelle: In Anlehnung an Geron, 2019[106]

Neuronale Netze sind flexibel und können sowohl für Regressions- als auch für Klassifizierungsprobleme verwendet werden. Eine **Stärke** besteht darin, dass neuronale Netze gut mit nichtlinearen Daten und einer großen Anzahl von Features (zum Beispiel Bilder) modelliert werden können. Ist ein neuronales Netz einmal trainiert, können Vorhersagen schnell getätigt werden.[107] Die größte **Schwäche** von künstlichen neuronalen Netzen stellt das sogenannte *Blackbox-Verhalten* dar. Die gelernte Hypothese ist im Allgemeinen schwer interpretierbar. Weiters gibt es keine Regel zur Bestimmung der Struktur künstlicher neuronaler Netze, und diese muss auf Basis von Erfahrung ausgewählt werden. Zusätzlich ist das Trainieren mit herkömmlichen CPUs rechenintensiv und zeitaufwändig.[108]

[105] vgl. zu diesem Abschnitt Géron 2019, S. 298.
[106] vgl. Géron 2019, S. 294.
[107] vgl. Ciaburro und Venkateswaran 2017.
[108] vgl. Mijwil 2018, S. 1f.

### 2.4.6 Support Vector Machines

Support Vector Machines (SVMs) sind überwachte Lernmethoden, die zur Klassifizierung und Regression verwendet werden können.[109]. Wie zu Anfang dieses Kapitels erwähnt, ist die Support Vector Machine ein Algorithmus, welcher ursprünglich zur binären Klassifikation eingesetzt wird. Daher werden in diesem Abschnitt zunächst SVMs in Bezug auf die binäre Klassifikation erklärt und am Ende des Kapitels erläutert, wie diese auch zur Mehrklassen-Klassifikation eingesetzt werden können.

Wie die im vorigen Abschnitt definierten neuronalen Netze basieren SVMs auf der Idee, eine *Hyperebene (Hyperplane)* im n-dimensionalen Instanzraum zu finden, welche die Klassen am besten trennt.[110] Deshalb wird zunächst der Begriff der Hyperebenen genauer definiert und die Suche der optimalen Hyperebene bei SVMs erläutert.

**Hyperebene und Support Vectors**
Allgemein können Hyperebenen als Trennflächen beschrieben werden, welche dazu verwendet werden, um Instanzen zu klassifizieren. Je nachdem, auf welcher Seite der Trennfläche sich eine Instanz befindet, wird diese der jeweiligen Klasse zugeordnet. Die Dimension dieser Hyperebene ist Abhängig von der Features, welche die Instanz beschreiben. Ist die Anzahl der Features zum Beispiel 2, dann stellt die Hyperebene eine Trennlinie dar. Bei drei Features handelt es sich um eine zweidimensionale Ebene.[111] In Abbildung 12 wird dargestellt, wie Hyperebenen zur Trennung von zwei Klassen verwendet werden können. In diesem Fall handelt es sich konkret um Trennlinien im zweidimensionalen Raum. Die zu klassifizierenden Objekte werden als Punkte in einem hochdimensionalen Raum behandelt und es wird versucht eine Hyperebene zu finden, die sie trennt. Wie in Abbildung 12 ersichtlich, gibt es im Allgemeinen mehrere Möglichkeiten, eine Hyperebene auszuwählen. Die Menge aller möglichen Hyperebenen definieren den Hypothesenraum H, in welchem die Hypothese h gesucht wird.

[109] vgl. Iqbal und Yan 2015, S. 950.
[110] vgl. Grus 2019, S. 211.
[111] vgl. Gandhi 2018.

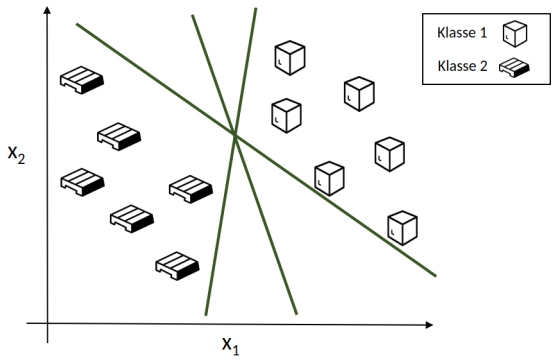


Abbildung 12: Klassifikation durch Hyperebenen, Quelle: In Anlehnung an Ghandi, 2018[112]

Die SVM unterscheidet sich von anderen auf Hyperebenen basierenden Klassifikatoren darin, wie die Hyperebene ausgewählt wird.[113] Bei der Auswahl der Hyperebene soll der Abstand dieser Hyperebene zum naheliegendsten Punkt jeder Klasse maximiert werden.[114] Dieser Zusammenhang wird in Abbildung 13 dargestellt.

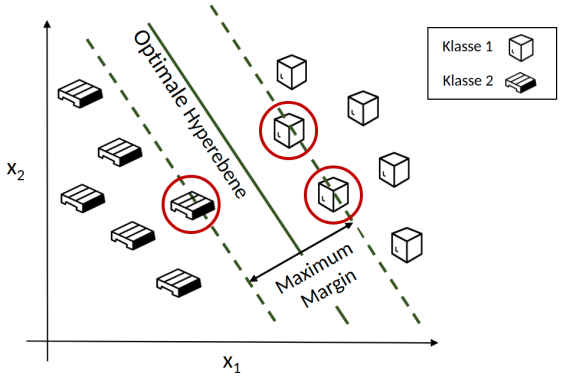


Abbildung 13: Optimale, Maximal-Margin-Hyperebene bei SVMs, Quelle: In Anlehnung an Ghandi, 2018[115]

Im zweidimensionalen Raum beschreibt die Hyperebene eine Linie und der Abstand zwischen der Linie und den nächstgelegenen Datenpunkten wird als *Margin* bezeichnet. Die Hyperebene, welche den größten Margin

[112] vgl. Gandhi 2018, S. 251.
[113] vgl. Noble 2006, S. 1565.
[114] vgl. Grus 2019, S. 211.
[115] vgl. Gandhi 2018, S. 251.

hat und die Klassen somit am besten trennt, wird als *Maximal-Margin-Hyperebene* bezeichnet. Im zweidimensionalen Fall wird der Margin als senkrechter Abstand von der Linie zu den nächstgelegenen Punkten berechnet. Nur diese Punkte sind für die Definition der Linie relevant und werden *Support-Vektoren (Support Vectors)* genannt. Die Hyperebene wird aus Trainingsdaten unter Verwendung eines Optimierungsverfahrens gelernt, welche den Margin maximiert.[116] Da Support Vector Machines ein Entfernungsmaß verwenden, um einer Instanz eine Klasse zuzuweisen, ist es wichtig, dass die Features numerisch und skaliert sind.

**Kernel-Trick und Kernelfunktionen**
In der Praxis sind reale Daten oft verrauscht und können mit einer Hyperebene nicht perfekt getrennt werden. Instanzen mit unterschiedlichen Klassen teilen sich also den Instanzraum auf eine Weise, die verhindert, dass eine lineare Hyperebene die verschiedenen Klassen korrekt trennt.[117] Eine Lösung für dieses Problem besteht darin, die Daten in einen höherdimensionalen Raum abzubilden und dort eine trennende Hyperebene zu definieren. Dies wird als der *Kernel-Trick* bezeichnet.[118] Die Funktion, welche die Daten aus einem niedrigdimensionalen Raum in einen Raum mit höherer Dimension projiziert, wird *Kernelfunktion* genannt. Die Auswahl einer geeigneten Kernelfunktionen ist wichtig, da durch die Projektion mittels einer geeigneten Kernelfunktion, die Daten im resultierenden höherdimensionalen Raum besser trennbar werden.[119] [120] Es gibt eine Reihe von verschiedenen Kernelfunktionen und die Auswahl der Kernelfunktion hängt von den zu klassifizierenden Daten ab. Unter anderem gibt es *lineare Kernel, RBF Kernel, polynomiale Kernel* und *Sigmoid Kernel*.[121]

**Support Vector Machine für Mehrklassen-Klassifikation**
Bei der Support Vector Machine handelt es sich ursprünglich um einen binären Klassifikator, welcher die entsprechende Klasse, aber keine Wahrscheinlichkeiten ausgibt. Jedoch können beispielsweise Methoden wie die *Platt-Skalierung* und *isotonische Regression* verwendet werden, um die Ausgabe einer SVM in Klassenwahrscheinlichkeiten umzuwandeln.[122] Eine Mehrklassen-Klassifikation mittels SVM kann durch eine Zerlegung des Mehrklassen-Problems in mehrere binäre Klassifizierungsaufgaben erreicht werden. Diese werden einzeln mit Hilfe binärer Klassifikatoren wie der SVM gelöst. Dabei gibt es unter anderem die Methoden *One-vs-Rest (One-*

[116] vgl. zu diesem Abschnitt Brownlee 2016, S. 166.
[117] vgl. Awad und Khanna 2015, S. 42.
[118] vgl. Kotsiantis 2007, S. 261.
[119] vgl. Noble 2006, S. 1567.
[120] vgl. Kotsiantis 2007, S. 261.
[121] vgl. Awad und Khanna 2015, S. 42.
[122] vgl. Arat 2019.

*vs-All)* oder *One-vs-One*[123]: Beim One-vs-All-Ansatz werden bei einem Klassifikationsproblem mit K Klassen K Binärklassifikatoren trainiert. Dabei wird die betrachtete Klasse k als positiv (eine eigene Klasse) und alle weiteren K-1 Klassen als negativ betrachtet. Zur Klassifikation wird die wahrscheinlichste Klasse aus allen K Binärklassifikatoren ausgewählt.[124] In Bezug auf die Vorhersage einer Verpackungsgröße bei 40 verschiedenen Verpackungsgrößen würde das bedeuten, dass 40 Binärklassifikatoren trainiert werden müssen. Eine weitere Möglichkeit ist der One-vs-One Ansatz. Hierbei werden bei K Klassen für jedes Klassenpaar, also insgesamt K(K-1)/2, Binärklassifikatoren trainiert. Eine neue Instanz wird dann mit Hilfe einer Mehrheitsabstimmung unter den binären Klassifikatoren klassifiziert. Bei 40 verschiedenen Verpackungsgrößen müssen bei diesem Ansatz 40x(40-1)/2 = 780 Binärklassifikatoren (780 SVMs) trainiert werden.

Im Allgemeinen gehören Support Vector Machines zu den robustesten und erfolgreichsten Klassifizierungsalgorithmen.[125] Eine **Stärke** der SVM-Klassifikatoren ist, dass diese in hochdimensionalen Räumen gut funktionieren. Weiters sind SVM-Klassifikatoren für Lernaufgaben geeignet, bei denen die Anzahl der Features größer ist als die Anzahl der Trainingsinstanzen. Aufgrund der Tatsache, dass sich die Parameter der SVM nur auf einige Support Vektoren und nicht auf den gesamten Trainingsdatensatz beziehen, ist eine schnelle Klassifikation möglich.[126] Eine **Schwäche** des SVM-Klassifikators ist die Tatsache, dass es in der Praxis oft schwierig und zeitaufwändig ist, den richtigen Kernel zu finden. Weiters ist es im Vergleich zum bereits erklärten Entscheidungsbaum kompliziert, das Modell zu verstehen und zu interpretieren.[127]

### 2.4.7 Zusammenfassung

Aus diesem Kapitel geht hervor, dass es eine Reihe von Algorithmen des maschinellen Lernens gibt, welche für ein System zur Vorhersage von Verpackungsgrößen eingesetzt werden können. Die genannten Vor- und Nachteile der Algorithmen haben einen Einfluss auf die Genauigkeit der Vorhersage und auch darauf, wie die Daten für den jeweiligen Algorithmus vorverarbeitet werden müssen. Eine Aussage darüber, wie gut ein Algorithmus für eine konkrete Problemstellung funktioniert, kann nur getätigt werden, wenn dieser tatsächlich implementiert wird. Dies wird im nächsten Kapitel durchgeführt.

[123] vgl Aly 2005, S. 4.
[124] vgl Murphy 2012, S. 503.
[125] vgl. Cortes und Vapnik 1995, S. 290.
[126] vgl. Kotsiantis 2007, S. 261.
[127] vgl. Bennett und Campbell 2000, S. 11.

# 3 Entwicklung und Implementierung des Vorhersagesystems

Die in Kapitel 1 definierte Problemstellung und die daraus resultierende Forschungsfrage beschreiben das Ziel, festzustellen, ob es mittels Machine Learning möglich ist, eine Verpackungsgröße für eine Lieferung vorherzusagen. Dafür wurde in Kapitel 2 zunächst ein grober Überblick über das Themenfeld des Machine Learnings gegeben sowie die vorhandenen Machine-Learning-Systeme kategorisiert und erläutert. Danach wurde der Verpackungsprozess skizziert und beschrieben, wie ein Vorhersagesystem der Verpackungsgrößen diesen Prozess verändern würde. Mit den vorhandenen Informationen über verschiedene Machine-Learning-Systeme und dem Verpackungsprozess wurden im nächsten Schritt verschiedene Machine-Learning-Algorithmen untersucht und erklärt, wie diese zur Vorhersage einer Verpackungsgröße eingesetzt werden können.

In diesem Kapitel soll mit Hilfe der gewonnenen Erkenntnisse aus Kapitel 2 ein Machine-Learning-System zur Vorhersage entwickelt und somit die definierte Forschungsfrage beantwortet werden. Dazu werden die untersuchten Algorithmen implementiert und die Vorhersagegenauigkeit aller erstellten Machine-Learning-Systeme untersucht.

Der Prozess, ein Machine-Learning-System zu entwerfen wie in Abbildung 14 dargestellt, kann in drei übergeordnete Schritte eingeteilt werden. Diese Schritte sind: *Data Engineering*, *Model Engineering* und *Model Deployment*.[128]


Abbildung 14: Drei Hauptschritte der Machine-Learning-Modell-Implementierung, Quelle: Eigene Darstellung

Für die Implementierung des Vorhersagemodells müssen jedoch zunächst die dafür notwendigen Daten erfasst und aufbereitet werden (*Data Engineering*). Danach werden diverse Machine-Learning-Modelle auf Basis der aufbereiteten Daten trainiert und evaluiert (*Model Engineering*). Abschließend wird ein trainiertes Modell in die bestehende Lagerverwaltungssoftware integriert (*Model Deployment*). Dieses Kapitel ist nach den genannten drei übergeordneten Schritten gegliedert. Jeder dieser Schritte wird im Detail im jeweiligen Abschnitt untersucht.

[128] vgl. Visengeriyeva et al. 2021.

## 3.1 Datenaufbereitung

Da Machine-Learning-Algorithmen aus bestehenden Daten Muster und Zusammenhänge erkennen, hat die Qualität dieser Beispieldaten einen großen Einfluss auf das Ergebnis des Machine-Learning-Algorithmus. Inkorrekte Daten können zu falschen Schlüssen und Entscheidungen führen.[129] In Bezug auf die Problemstellung der Vorhersage der Verpackungsgröße ist die Genauigkeit der Vorhersage abhängig von der Qualität der Eingabedaten. Deshalb wird in der Praxis ein Großteil der gesamten Zeit der Implementierung eines Modells dafür verwendet, Daten zu erfassen, aufzubereiten und zu bereinigen.[130] Der Datenaufbereitungsprozess unterteilt sich in den in Abbildung 15 dargestellten relevanten Subprozessen *Datenerfassung*, *Explorative Datenanalyse*, *Datenvorverarbeitung* und *Datentransformation*.[131]


Abbildung 15: Prozesse der Datenaufbereitung, Quelle: Eigene Darstellung

### 3.1.1 Datenerfassung

Ziel der Datenerfassung ist es, Datensätze zu finden, mit denen Modelle des maschinellen Lernens trainiert werden können. Im konkreten Fall geht es darum, Daten zu erfassen, welche für die Erstellung eines Vorhersagemodells auf Basis von Machine Learning relevant sind. Es wird dabei zwischen drei Ansätzen unterschieden: *Datenentdeckung*, *Datenerweiterung* und *Datengenerierung*. Die Datenentdeckung ist erforderlich, wenn neue Datensätze gesucht werden müssen. Dieser Ansatz gewinnt an Relevanz, da immer mehr Datensätze im Web verfügbar sind. Die Datenerweiterung ergänzt die Datenentdeckung, bei der vorhandene Datasets durch Hinzufügen weiterer externer Daten erweitert werden. Die Datengenerierung kann verwendet werden, wenn kein externer Datensatz verfügbar ist.[132] Im Rahmen dieser Arbeit werden nur die bereits in relationalen Datenbanken vorhandenen Datensätze der in Abbildung 16 dargestellten Datenstruktur für das Trainieren des Modells verwendet. Dies wird als Datenintegration bezeichnet und kann auch als Datenerweiterungstechnik betrachtet werden. Die meisten Systeme für maschinelles Lernen ignorieren die Tatsache, dass aufgrund der Normalisierung in der Regel mehrere Tabellen in einer Datenbank vorhanden sind. Die Daten aus den unterschiedlichen Tabellen

[129] vgl. Ilyas und Chu 2019, S. 13.
[130] vgl. Boehmke 2016, S. 5.
[131] vgl. Opidi 2019.
[132] vgl. Roh, Heo und Whang 2021, S. 1.

müssen daher vor dem Trainieren des Modells verknüpft werden.[133] Für das in dieser Arbeit diskutierte Modell werden die für die Vorhersage relevanten Tabellen zusammengefügt und die Daten in Form einer einzigen CSV-Datei pro Shop gespeichert.

**Datenstruktur**
Auf Basis der Problemstellung und Zielsetzung müssen die für die Vorhersage notwendigen Daten erfasst werden. Jeder Shop besitzt seine eigene relationale Datenbank, in welcher die Daten aufgrund der Normalisierung in mehreren Tabellen gespeichert sind. Aus Gründen der Übersichtlichkeit werden die für die Vorhersage potentiell relevanten Tabellen und Spalten nicht einzeln, sondern gruppiert als Eigenschaften der in Abschnitt 2.3.1 definierten Elemente des Verpackungsprozess beschrieben. Als Datengrundlage werden dabei die Daten in Bezug auf das *Paketelement*, das *Paket* und die *Versandverpackung* definiert. Zur Erinnerung: Ein Paket (Package) beschreibt die Gesamtheit aller Paketelemente in einer Verpackung (Packaging). Ein Paketelement (PackageItem) besteht allgemein aus einem Produkt mit den zugehörigen Eigenschaften und der Menge dieses Produkts. Diese Datengrundlage des Vorhersagemodells ist in Abbildung 16 dargestellt.

Die Kardinalitäten zwischen dem Paket und der Verpackung sowie zwischen dem Paket und dem Paketelement geben an, dass in einem Paket mindestens ein Paketelement enthalten ist und ein Paket genau eine Verpackung hat. Die Eigenschaften der jeweiligen Elemente, welche in Abbildung 16 dargestellt sind, werden in den darauf folgenden Tabellen näher erläutert.

Tabelle 4 beschreibt die Eigenschaften einer Verpackung, Tabelle 5 beschreibt die Eigenschaften eines Pakets, und Tabelle 6 beschreibt die Eigenschaften eines Paketelements.

[133] vgl. Roh, Heo und Whang 2021, S. 6.

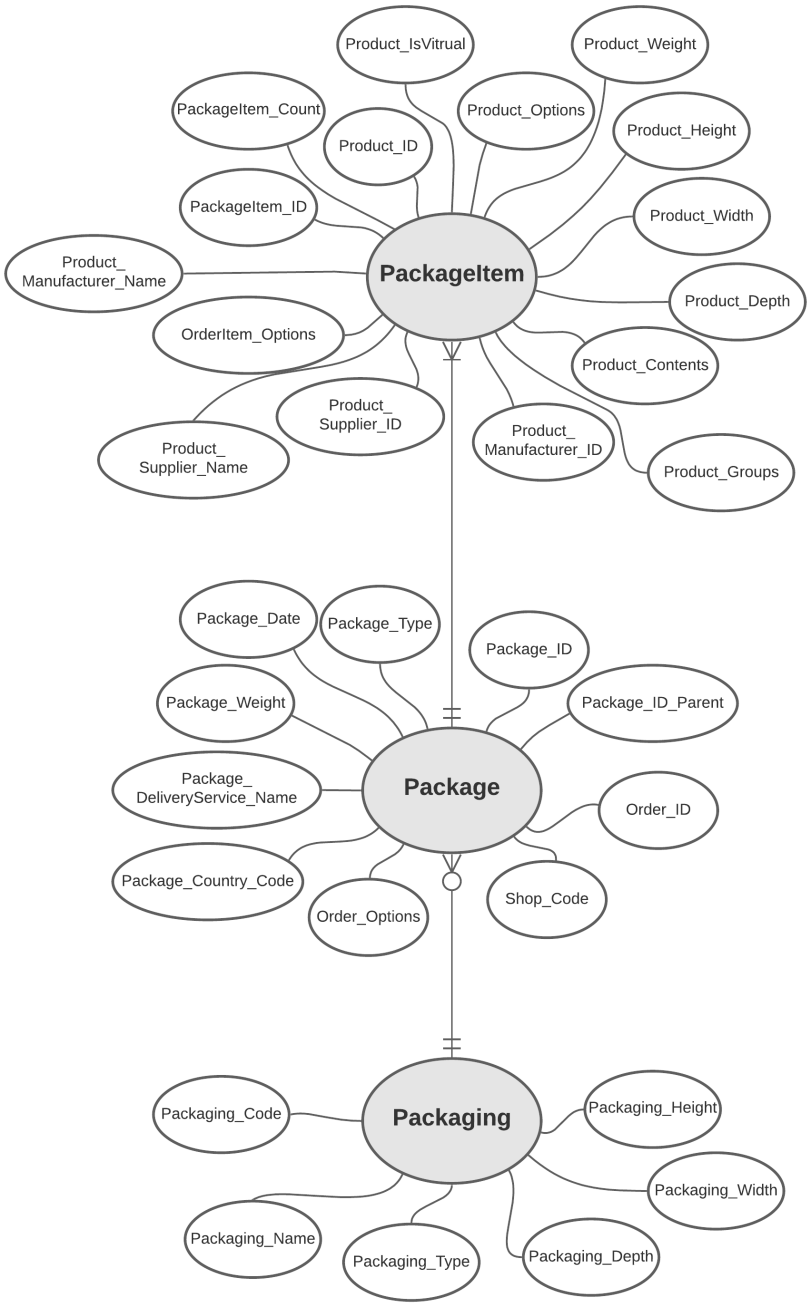


Abbildung 16: Eigenschaften von Paketen, Paketelementen und Verpackungen, Quelle: Eigene Darstellung

| **Attribut** | **Bezeichnung** | **Erklärung** |
|---|---|---|
| Packaging_Code | Verpackungscode | Code der Verpackung, in welcher das Paket versendet wird |
| Packaging_Name | Verpackungs-bezeichnung | Bezeichnung der Verpackung, in welcher das Paket versendet wird |
| Packaging_Type | Verpackungstyp | Typ der Verpackung (Palette, Karton, Frischbox,...) |
| Packaging_Height | Verpackungshöhe | Höhe der Verpackung (nicht vorhanden bei Paletten) |
| Packaging_Width | Verpackungsbreite | Breite der Verpackung |
| Packaging_Depth | Verpackungstiefe | Tiefe der Verpackung |

Tabelle 4: Eigenschaften einer Verpackung

| **Attribut** | **Bezeichnung** | **Erklärung** |
|---|---|---|
| Package_ID | Lieferung/ Packmittel ID | Es kann sich hier je nach Typ (*delivery*, *label*) um die ID der Lieferung (Typ *delivery*) oder um die ID des Packmittels (Typ *label*) handeln. |
| Package_ID_ Parent | Rechnung/ Lieferung ID | Handelt es sich um ein Paket vom Typ *delivery*, ist die Package_ID_Parent die ID der Rechnung. Ist das Paket vom Typ *label*, ist dies die ID des Lieferscheins. |
| Package_ID_ Type | Lieferungstyp | Zeigt an, ob es sich um eine Lieferung oder um ein Packmittel (Teil einer aufgeteilten Lieferung) handelt |
| Package_Date | Verpackungs-datum | Datum, an dem das Paket verpackt wurde |
| Package_Weight | Paketgewicht | Gesamtgewicht des Paketes (Gewicht aller Paketelemente + Gewicht der Verpackung) |
| Package_Delivery Service_Name | Lieferservice | Name des Lieferservices, mit dem das Paket verschickt wurde |
| Package_ Country_Code | Ländercode | Kurzbezeichnung des Landes, in welches das Paket verschickt wurde |
| Order_ID | Bestellungs ID | ID der Bestellung |
| Order_Options | Bestelloptionen | Optionen, welche auf Bestellebene gesetzt werden können (Geschenkbox, Eigenentnahme,...) |
| Shop_Code | Shop Bezeichnung | Bezeichnung des Shops, in dem die Lieferung aufgegeben wurde |

Tabelle 5: Eigenschaften eines Pakets

| **Attribut** | **Bezeichnung** | **Erklärung** |
|---|---|---|
| PackageItem_ID | ID des Paketelements | Dient zur Identifizierung eines Paketelements |
| PackageItem_ Count | Anzahl der Produkte | Beschreibt die Anzahl, wie oft dieses konkrete Produkt in der Lieferung enthalten ist |
| Product_ID | ID des Produktes | Dient zur Identifizierung eines Produktes |
| Product_ IsVirtual | Virtuelles Produkt | Zeigt an, ob es sich um ein virtuelles Produkt, zum Beispiel einen Gutscheincode, handelt |
| Product_Options | Produktoptionen | Spezifizieren das Produkt näher (zerbrechlich, entzündlich, bereits verpackt, Speditionsware) |
| Product_Weight | Produktgewicht | Gesamtgewicht des Produktes |
| Product_Height | Produkthöhe | Höhe des Produktes |
| Product_Width | Produktbreite | Breite des Produktes |
| Product_Depth | Produkttiefe | Tiefe des Produktes |
| Product_ Contents | Inhalt | Volumen in Liter/Milliliter oder Stückzahl |
| Product_Groups | Produktgruppe | Ein Produkt ist mehreren Gruppen und in der Regel einer Hauptwarengruppe zugeordnet |
| Product_ Manufacturer_ID | Hersteller ID | ID des Herstellers des Produkts |
| Product_ Manufacturer_Name | Hersteller Name | Name des Herstellers des Produkts |
| Product_ Supplier_ID | Lieferant ID | ID des Lieferanten des Produkts |
| Product_ Supplier_Name | Lieferant Name | Name des Lieferanten des Produkts |
| OrderItem_ Options | Paketelement Optionen | Optionen,welche zusätzlich noch auf Paketelement Ebene gesetzt werden können (Frischware, Kühlware, Geschenkbox, Probe) |

Tabelle 6: Eigenschaften eines Paketelements

Obwohl diese Eigenschaften im nächsten Schritt aus den jeweiligen Tabellen geholt werden, werden nicht alle davon für das endgültige Modell verwendet. Sie beschreiben lediglich die Datengrundlage, aus welcher relevante Eigenschaften, sogenannte *Features*, ausgewählt werden. Es ist vorab bekannt, dass Eigenschaften wie die Dimensionen oder der Inhalt des Produkts nur selten gepflegt sind. Diese Tatsache und weitere Eigenschaften werden in der explorativen Datenanalyse genauer untersucht.

**Daten via REST abfragen**

Die relevanten Daten sind in verschiedenen Tabellen gespeichert. Der erste Schritt besteht darin, eine Möglichkeit zu schaffen, diese Daten für die Analyse und die Erstellung eines Modells abzufragen. Da jeder Shop seine eigene Datenbank mit den zugehörigen Tabellen hat, müssen die Daten

zunächst von allen Shops geholt werden. Dies kann über eine *RESTful API* gelöst werden. Eine RESTful API ist eine Programmierschnittstelle, welche die Kommunikation zwischen Client und Server in Netzwerken beschreibt.[134]

Der *REST-Client* ist in diesem Fall der MetaShop, welcher kein tatsächlicher Shop ist, bei dem man bestellen kann, sondern welcher dafür zuständig ist, shopübergreifende Daten zu pflegen. In diesem Fall fragt der Metashop bei allen tatsächlichen Shops (REST-Server) um die benötigten Daten an. Des Weiteren speichert der MetaShop als REST-Client die Daten vom Shop in eine CSV-Datei pro Shop und stellt die Dateien aller Shops als ZIP-Datei zum Download zur Verfügung.

Der jeweilige Shop als *REST-Server* setzt ein SQL-Statement auf seiner Datenbank ab und übermittelt die Daten. Aufgrund des großen Datenvolumens müssen die Daten von jedem Shop in Batches und nicht gesamt übermittelt werden.

### 3.1.2 Explorative Datenanalyse und Datenvorverarbeitung

Diese nun verfügbaren Datensätze mit den in Abbildung 16 definierten Eigenschaften müssen analysiert und vorverarbeitet werden, um sie später für das Lernen eines Modells verwenden zu können. Das Ziel jeder Datenanalyse ist es, Wissen zu entdecken, das zur Lösung von Problemen oder zur Entscheidungsfindung verwendet wird. Inkorrekte oder fehlende Daten können dies jedoch verhindern und werden oft erst bemerkt, wenn die Daten analysiert werden.[135] Aus diesem Grund wird in Bezug auf die Entwicklung des Vorhersagemodells die explorative Datenanalyse und Datenvorverarbeitung parallel durchgeführt und kann als iterativer Prozess angesehen werden. Werden im Rahmen der explorativen Datenanalyse Probleme in den Daten entdeckt, werden diese Probleme in der Datenvorverarbeitung behoben und die Daten danach weiter analysiert. Das Ziel in diesem Schritt ist es, mit Hilfe der Vorverarbeitung und Analyse der Daten Eigenschaften zu entdecken, welche der Lernalgorithmus nutzen kann, um die Verpackungsgröße vorherzusagen.

Die *explorative Datenanalyse*, auch explorative Statistik, ist ein Teilbereich der Statistik mit dem Ziel, einen Überblick über die zu untersuchenden Daten zu bekommen. Dabei werden die Daten hinsichtlich Muster und Zusammenhänge analysiert. Der Begriff *explorativ* kommt daher, dass man in den Daten nach relevanten Informationen forscht. Generell werden zur Analyse Grafiken wie Boxplots, Histogramme oder Streudiagramme

[134] vgl. Masse 2011, S. 5.
[135] vgl. Famili et al. 1997, S. 1.

verwendet.[136]

Viele Faktoren beeinflussen den Erfolg des maschinellen Lernens bei einer bestimmten Aufgabe. Die Repräsentation und Qualität der Daten ist dabei der wichtigste Faktor. Wenn viele irrelevante und redundante Informationen vorhanden sind oder verrauschte und unzuverlässige Daten vorliegen, ist die Entdeckung von Wissen während der Trainingsphase schwieriger.[137] Da die Auswahl der Verpackungsgröße für eine Lieferung derzeit ein manueller Prozess ist, welcher durch den Lagermitarbeiter durchgeführt wird, kann es auch hier zu Aufzeichnungsfehlern kommen. Es besteht die Möglichkeit, dass die Produkte zwar in eine richtige Versandverpackung gegeben werden, jedoch unabsichtlich eine falsche Verpackungsgröße ausgewählt wird. Andererseits können Produkte auch falsch oder von Arbeiter zu Arbeiter unterschiedlich verpackt werden. Dies führt dazu, dass es für das Modell schwierig ist, Zusammenhänge und Muster zu erkennen. Die erfassten Daten können auch in einem falschen Format vorliegen, zum Beispiel können diese unterschiedliche Datumsformate haben, unorganisiert oder sehr groß sein. Deshalb sind weitere Schritte notwendig, um die Qualität dieser Daten zu erhöhen – *Datenvorverarbeitung*. Die drei üblichen Teilgebiete der Datenvorverarbeitung sind Formatierung, Bereinigung und Sampling. Die Datenformatierung stellt sicher, dass alle Werte innerhalb desselben Attributs gleich geschrieben werden. Im Rahmen der Datenbereinigung werden unordentliche Daten entfernt und gegebenenfalls fehlende Werte ergänzt. In diesem Schritt können auch Duplikate und Ausreißer entfernt werden, wenn die explorative Datenanalyse zeigt, dass diese Daten definitiv fehlerhafte Daten sind. Sampling (Datenauswahl) kann notwendig sein, wenn viele Daten vorhanden sind und nur eine kleine Menge zum Trainieren des Modells verwendet wird, um Zeit und Kosten zu sparen.[138]

**Verwendete Technologien**
Die explorative Datenanalyse und Datenvorverarbeitung wurde mit Hilfe eines *Google Colab IPython Notebooks* mittels *Python* durchgeführt. Dabei wurden die notwendigen Bibliotheken zur Datenanalyse und -verarbeitung wie *Pandas*, *Numpy*, *Matlplotlib* und *Seaborn* verwendet.

**Überblick**
Der erste Schritt besteht darin, sich einen groben Überblick über die vorhandenen Daten zu verschaffen. Dabei sei zu erwähnen, dass die nachfolgende Analyse sich auf die Daten bezieht, welche am 19.03.2021 erfasst

[136] vgl. Schäfer 2010, S. 99.
[137] vgl. Kotsiantis, Kanellopoulos und Pintelas 2006, S. 111.
[138] vgl. Opidi 2019.

wurden. Weiters wurden die Namen der Onlineshops, welche nicht zur Niceshops Gruppe gehören, anonymisiert (Shop A, Shop B,...). Zunächst werden die Daten aus den einzelnen CSV-Dateien pro Shop geladen und in einem gemeinsamen *Pandas Dataframe* gespeichert. Ein Pandas Dataframe kann mit einer gewöhnlichen Tabelle verglichen werden, in welcher zweidimensionale Daten in Form von Zeilen und Spalten gespeichert werden. Wie in Abschnitt 3.1.1 erläutert, werden die Eigenschaften des Pakets, der Paketelemente und der Versandverpackung verknüpft. Das bedeutet, jede Zeile in diesem Dataframe repräsentiert ein Paketelement, also ein Produkt in einer bestimmten Menge in einem Paket. Das führt dazu, dass Eigenschaften des Pakets und der Versandverpackung redundant sind, da diese für jedes Paketelement gleich sind. Dieser Zusammenhang ist in Abbildung 17 dargestellt.

| Shop | Paket ID | Paket Gewicht | ... | Verp. Code | Verp. Typ | Verp. Volumen | ... | Paketel. ID | Paketel. Anzahl | Produkt ID | Produkt Titel | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Shop A | 1 | 20 | ... | EUR | Paletten | 120 | ... | 11 | 1 | 111 | Produkt A | ... |
| Shop A | 1 | 20 | ... | EUR | Paletten | 120 | ... | 12 | 2 | 222 | Produkt B | ... |
| Shop A | 2 | 5 | ... | PK_2 | Kartons | 12 | ... | 21 | 3 | 111 | Produkt C | ... |
| Shop B | 1 | 3 | ... | PK_3 | Kartons | 10 | ... | 11 | 1 | 111 | Produkt Z | ... |
| Shop B | 1 | 3 | ... | PK_3 | Kartons | 10 | ... | 12 | 4 | 222 | Produkt Y | ... |

Eigenschaften des Pakets | Eigenschaften der Verpackung | Eigenschaften des Paketelements und des Produkts

Abbildung 17: Dataframe mit den Eigenschaften von Paketen, Paketelementen und Verpackungen, Quelle: Eigene Darstellung

Wie in den Beispieldaten in Abbildung 17 erkannt werden kann, sind die Paket-, Paketelement- und Produkt-Identifizierungsnummern nur für einen Shop eindeutig. So können zum Beispiel zwei verschiedene Produkte unterschiedlicher Shops die gleiche Produkt ID haben.

Als Erstes wird untersucht, wie viele Daten zur Verfügung stehen. Der tatsächliche Dataframe mit derselben Struktur wie in Abbildung 17 enthält *5.937.821* Zeilen und *34* Spalten. Das heißt, zur Untersuchung stehen knapp *6 Millionen* Paketelemente mit *34* Eigenschaften zur Verfügung. Im Rahmen der Datenformatierung werden die richtigen Datentypen gesetzt. Des Weiteren wird der Shop Code länderunabhängig gemacht. Eine Lieferung im Shop *Biolindo* nach Deutschland hat beispielsweise den Shop Code *biolindo_de*, eine Lieferung nach Österreich *biolindo_at*. Da es jedoch für die Vorhersage und Analyse auch den Ländercode als Eigenschaft gibt, wird dieses Länderkürzel im Shop Code entfernt. Weiters wurden fehlende Werte der Bestell-, Paketelement- und Produktoptionen durch einen Leerstring ersetzt.

**Analyse auf Paketebene**

Der zuvor beschriebene Dataframe enthält alle Paketelemente. Da für die weitere Analyse die Details der Produkte in einem Paket nicht benötigt werden, wird ein eigener DataFrame erstellt. Dieser Dataframe enthält nur die Daten der Pakete und zugehörigen Verpackungen mit einer zusätzlichen Spalte für die Gesamtanzahl der Produkte in diesem Paket. Dieser Schritt fällt bereits unter das Stichwort *Feature Engineering* oder *Feature Extraction* und wird in Abschnitt 3.2.3 genauer erläutert. Für die Erstellung des Dataframes für die Pakete werden die Zeilen der Paketelemente gruppiert. Dieses Vorgehen wird in Abbildung 18 dargestellt.

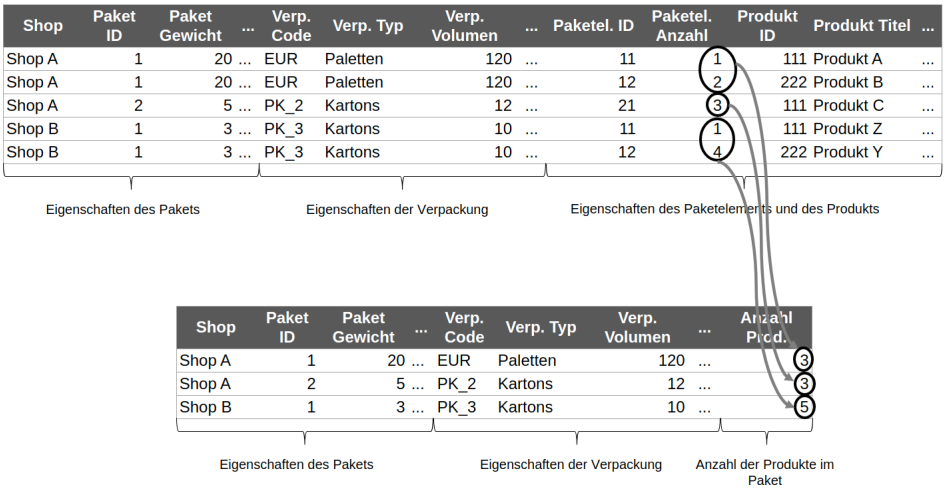


Abbildung 18: Dataframe mit den Eigenschaften von Paketen und Verpackungen, Quelle: Eigene Darstellung

Jede Zeile dieses Dataframes repräsentiert nun ein Paket, das versendet wurde. Die gewonnene Spalte beschreibt die Gesamtanzahl der Produkte im Paket und ist somit eine neue Eigenschaft des Pakets, ein neues Feature. Wie bereits definiert, handelt es sich bei der Problemstellung um überwachtes Lernen. Wie aus Abbildung 18 erkannt werden kann, sind die Trainingsbeispiele, Pakete, mit einem Label, dem Verpackungscode, gekennzeichnet. Aus den Zusammenhängen zwischen den Eigenschaften des Pakets und dem Verpackungscode soll das Modell in Kapitel 3.2 trainiert werden.

Zunächst werden jedoch die Daten im Detail analysiert und gegebenenfalls bereinigt. Die weitere Analyse umfasst folgende Aspekte:

- Gesamtanzahl der Pakete und Produkte
- Anteil der Pakete vom Typ *label*

- Analyse der Verpackungsgröße
  - Gesamtanzahl und Anzahl pro Shop
  - Aufteilung pro Shop und Lieferdienst
- Ausreißerbehandlung – Paketgewicht und Anzahl der Produkte

**Gesamtanzahl der Pakete und Produkte**
Es wurden gesamt *1.557.922* Pakete mit der zugehörigen Verpackungsgröße aufgezeichnet. Das bedeutet, es stehen etwas mehr als *1,5 Millionen* Trainingsbeispiele vor der weiteren Datenbereinigung zur Vorhersage der Versandverpackung zur Verfügung.

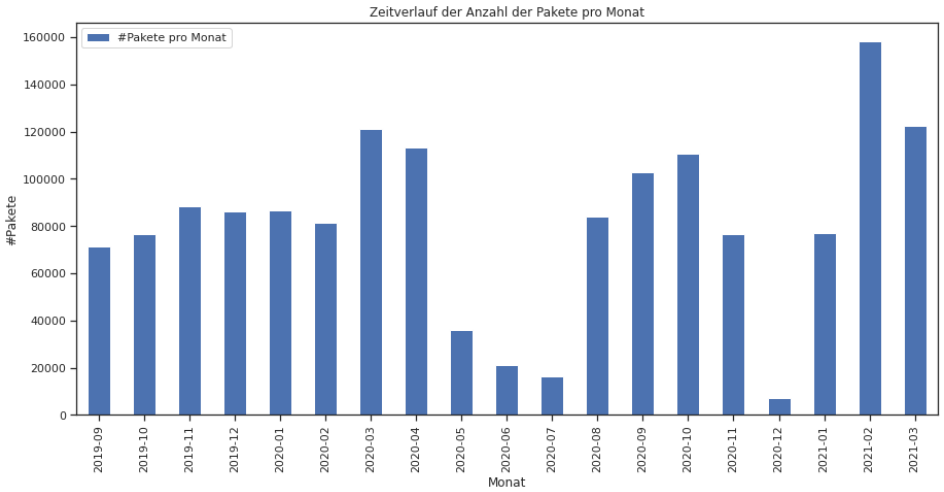


Abbildung 19: Zeitverlauf der Anzahl der Pakete pro Monat, Quelle: Eigene Darstellung

Aus den monatlichen Zahlen in Abbildung 19 geht vorher, dass die verpflichtende Auswahl der Paketgröße erst im September 2019 aktiviert wurde. Aufgrund des zusätzlichen Aufwandes im Lager, verbunden mit der Auswahl der Verpackungsgröße, wurde diese verpflichtende Auswahl Ende April 2020 aufgrund des hohen Bestellaufkommens wegen der Corona Pandemie deaktiviert. Anfang August 2020 wurde die Auswahl wieder aktiviert und Ende November wegen der vielen Bestellungen vor Weihnachten wieder deaktiviert. Seit Mitte Jänner 2021 ist die verpflichtende Auswahl der Verpackungsgröße wieder aktiv. Wie bereits in den Vorteilen des Vorhersagemodells erläutert, müsste durch die Vorauswahl der Verpackungsgröße in Zukunft die verpflichtende Auswahl nicht mehr deaktiviert werden.

Des Weiteren ist für die Entwicklung eines Vorhersagemodells von Relevanz, wie diese Daten in Bezug auf die verschiedenen Shops verteilt

sind. Abbildung 20 zeigt, welche Anzahl der aufgezeichneten 1,5 Millionen Pakete auf welche Shops fällt.

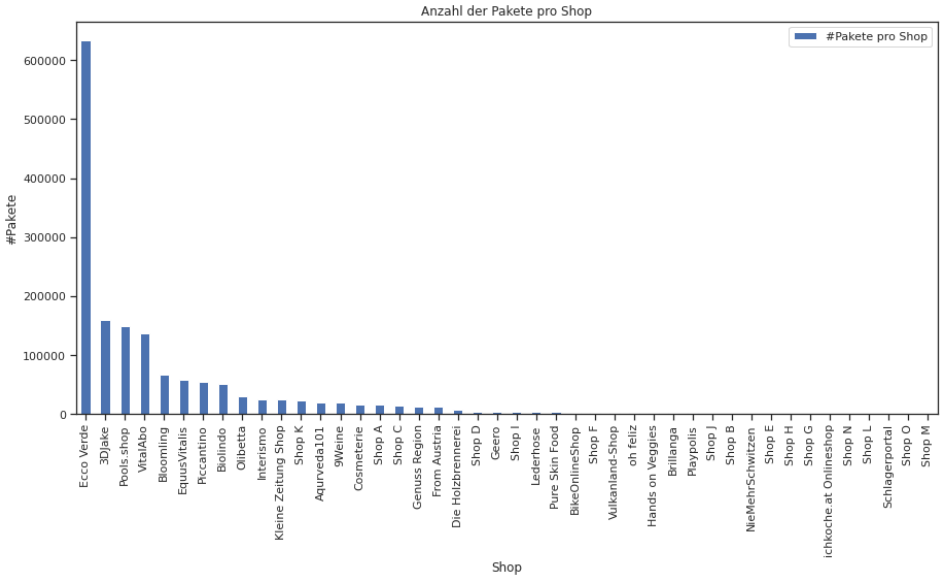


Abbildung 20: Anzahl der Pakete pro Shop, Quelle: Eigene Darstellung

Aus diesem Diagramm geht hervor, dass fast die Hälfte aller aufgezeichneten Lieferungen dem Shop *Ecco Verde* zugeordnet sind. Würde ein Modell über alle Shops hinweg trainiert werden, wäre es stark von den Daten dieses Shops beeinflusst. Eine Möglichkeit wäre, für jeden Shop ein eigenes Modell zu erstellen. Dies wird in Kapitel 3.2 näher diskutiert.

Auch die Anzahl der Produkte ist für die Vorhersage relevant. Denn je mehr Produkte vorhanden sind, desto schwieriger und aufwändiger ist es, aus den potentiellen Kombinationen von Produkten, welche in einem Paket vorhanden sein können, zu lernen. Insgesamt wurden im Unternehmen über alle Shops hinweg bereits *415.413* verschiedene Produkte versendet. Weiters wurde untersucht, bei wie vielen Produkten im Unternehmen die Dimensionen Höhe, Breite und Tiefe hinterlegt sind. Dabei wurde entdeckt, dass lediglich bei 5,67 % der Produkte diese Maße hinterlegt sind. Wie bereits in der Einleitung erwähnt, macht es diese Tatsache nicht möglich, die Verpackungsgröße auf Basis des Volumens der Produkte vorherzusagen. Des Weiteren kann aufgrund der fehlenden Daten das Volumen auch nicht als Eigenschaft zur Vorhersage verwendet werden. Im nächsten Schritt werden die in den Paketelementen vorhandenen Produkte pro Shop gezählt und im Diagramm in Abbildung 21 dargestellt. Es ist darauf hinzuweisen, dass es sich hierbei nur um die Produkte, welche bereits Bestandteil einer

Lieferung waren, handelt. Produkte, welche noch nie versendet wurden, scheinen nicht auf.

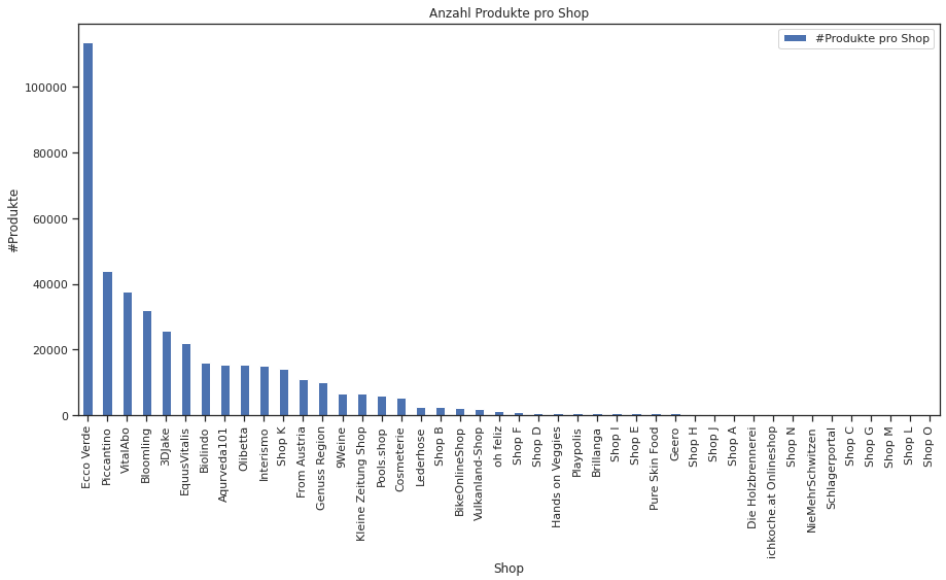


Abbildung 21: Anzahl der Produkte pro Shop, Quelle: Eigene Darstellung

Abbildung 21 zeigt, dass *Ecco Verde* nach den meisten Paketen auch die meisten Produkte hat. Der Shop *Piccantino* hat zwar die zweitmeisten Produkte im Sortiment, jedoch eine vergleichsmäßig geringe Anzahl an Paketen.

Dieser Zusammenhang zwischen der Anzahl der Pakete und der Anzahl der Produkte wird nun in Form der Kenngröße *Anzahl der Pakete bezogen auf die Anzahl der Produkte* genauer untersucht. Diese Größe könnte für die Vorhersagegenauigkeit eines Modells pro Shop relevant sein, denn je mehr Paketlieferungen und je weniger Produkte pro Shop vorhanden sind, desto leichter könnte es für diesen Shop sein, eine Verpackungsgröße vorherzusagen. Das folgende Diagramm in Abbildung 22 zeigt diesen Zusammenhang. Es wurde die Anzahl der Pakete durch die Anzahl der Produkte pro Shop dividiert und drei entsprechende Gruppen gebildet.

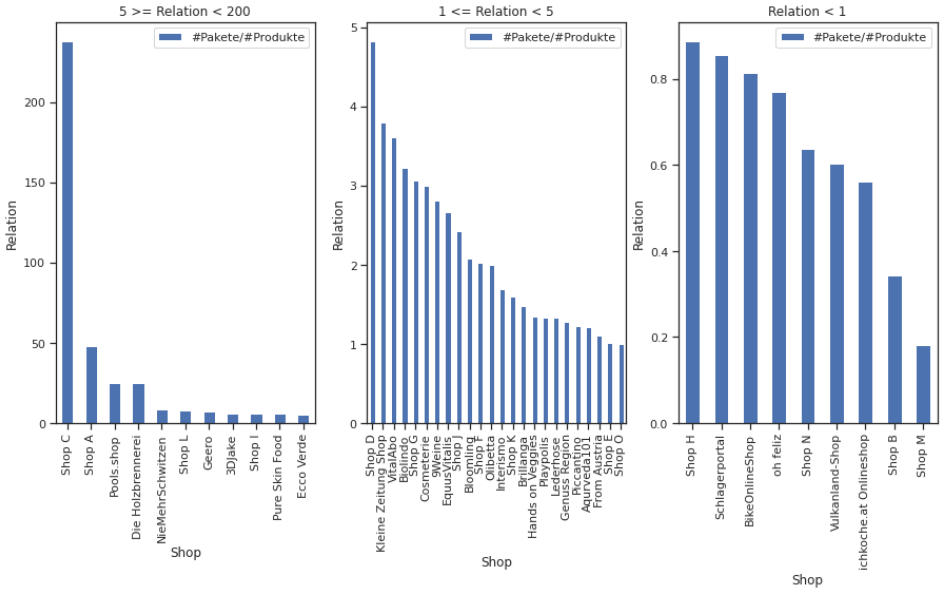


Abbildung 22: Verhältnis der Anzahl an Paketen und Produkten pro Shop, Quelle: Eigene Darstellung

*Shop C* ist ein Shop, welcher nur 9 Produkte und daher ein großes Verhältnis hat. Eine weitere relativ hohe Paket/Produkt-Relation haben die Shops *Die Holzbrennerei*, *Pools.shop* und *Shop A*. Dadurch, dass *Ecco Verde* eine sehr große Anzahl an Paketen hat, ist die Relation trotz der ebenso großen Anzahl an Produkten größer als 10. Shops, welche einen Relationswert unter 1 haben, sind einerseits Shops, welche nicht mehr online sind und dadurch wenig Daten haben. Andererseits sind es Shops wie *Shop B* oder *Shop D*, welche meist nur Produkte für andere Shops bereitstellen und nicht selbst ausliefern.

**Anteil der Pakete mit Typ *label***

Wie bereits erwähnt gibt es zwei verschiedene Typen von Paketen: *delivery* und *label*. Zur Erinnerung: Ein Paket erhält nur den Typ *label*, wenn eine geplante Lieferung während des Verpackungsprozesses in mehrere Pakete aufgeteilt werden muss. Erfahrungsgemäß geschieht dies jedoch nur in Ausnahmefällen. Da jedoch ein Modell, welches anstatt einer Verpackungsgröße mehrere Verpackungsgrößen vorhersagen muss, komplexer ist, wird nun untersucht, welchen Anteil diese aufgeteilten Pakete an der Gesamtanzahl der Pakete haben. Aus den Daten der Pakete wurde ermittelt, dass der Anteil der aufgeteilten Pakete mit dem Typ *label* bei *1,01 %* liegt. Da dies ein relativ geringer Prozentsatz ist, wird im Rahmen dieser Arbeit ein Modell entwickelt, welches nur eine Verpackungsgröße vorhersagen kann. Wie in Abbildung 23 dargestellt, variiert der Anteil der aufgeteilten

Lieferungen pro Shop.

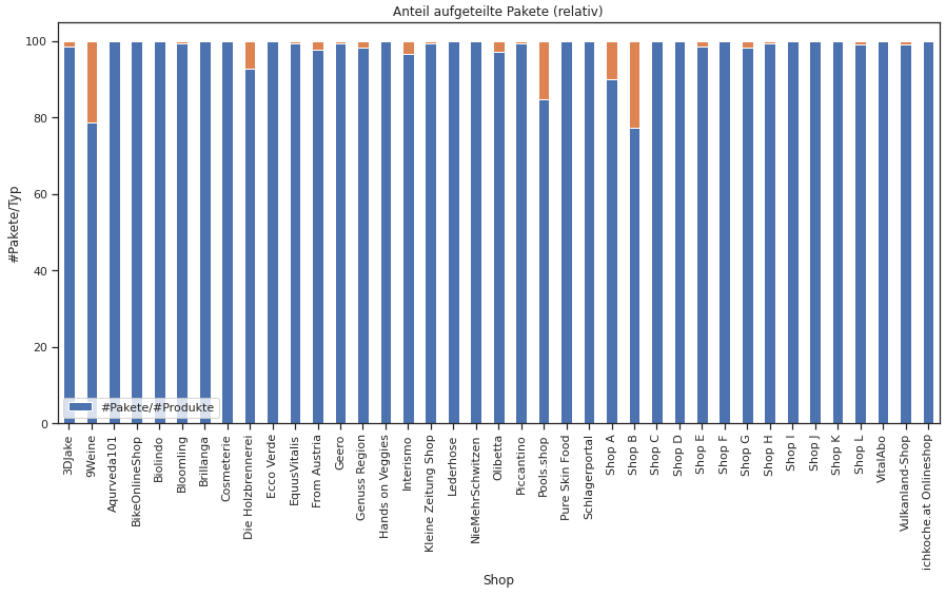


Abbildung 23: Anteil der aufgeteilten Pakete an der gesamten Anzahl, Quelle: Eigene Darstellung

Aus dem Diagramm ist zu entnehmen, dass es Shops gibt, bei denen die Anzahl der aufgeteilten Pakete bei über 20 % liegt. Dabei handelt es sich zum einen um Shops, welche schwere und sperrige Produkte anbieten (*Biovera*, *Holzbrennerei*, *Pools*, *Yaasa*) oder um Shops, welche viele Flaschen versenden (*9 Weine*), wofür spezielle Flaschenkartons verwendet werden.

**Analyse der Verpackungsgröße**

Da das Ziel darin besteht, Verpackungsgrößen vorherzusagen, müssen diese analysiert und Zusammenhänge identifiziert werden. Zunächst gilt es, sich anzusehen, in welchen Verpackungen die Pakete das Lager verlassen haben. In Abbildung 24 ist der Anteil der Anzahl für jede einzelne Verpackungsgröße an der Gesamtanzahl der versendeten Verpackungsgrößen dargestellt. Aus diesem Diagramm ist ersichtlich, dass es sich hauptsächlich um Kartons (Paket x) handelt, in denen die Produkte das Lager verlassen. Des Weiteren kann erkannt werden, dass eine Vielzahl verschiedener Versandverpackungen verfügbar sind, jedoch die zehn häufigsten Kartons über 80 % aller verwendeten Verpackungen ausmachen.

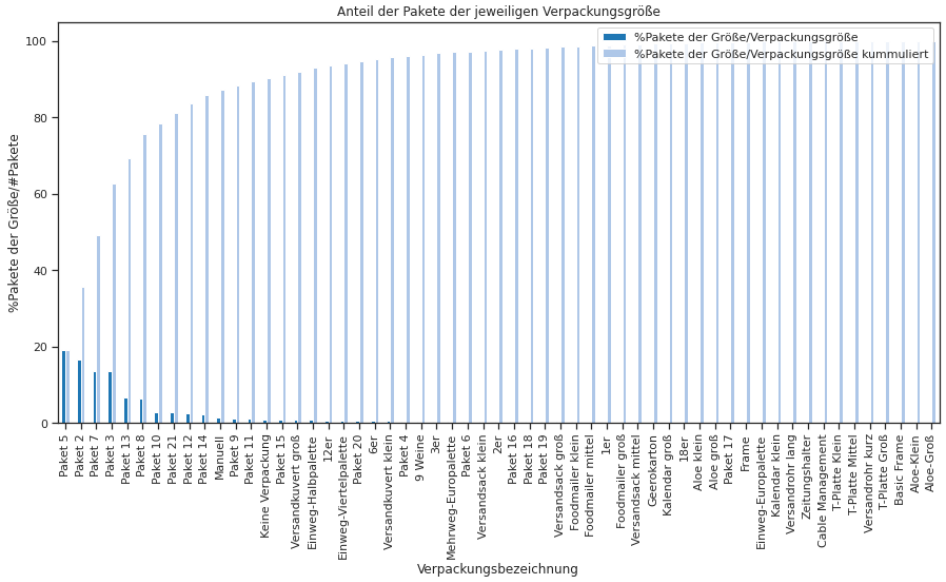


Abbildung 24: Anteil der Pakete der jeweiligen Verpackungsgröße, Quelle: Eigene Darstellung

Zusätzlich zur Gesamtanzahl wird im nächsten Schritt die Anzahl der Verpackungsgrößen pro Shop ermittelt, um gegebenenfalls zu erkennen, ob man von einem Shop auf gewisse Verpackungsgrößen schließen kann. Aufgrund der Übersichtlichkeit werden pro Shop nur die drei häufigsten Verpackungsgrößen angezeigt und die Anzahl der Shops auf 15 limitiert.

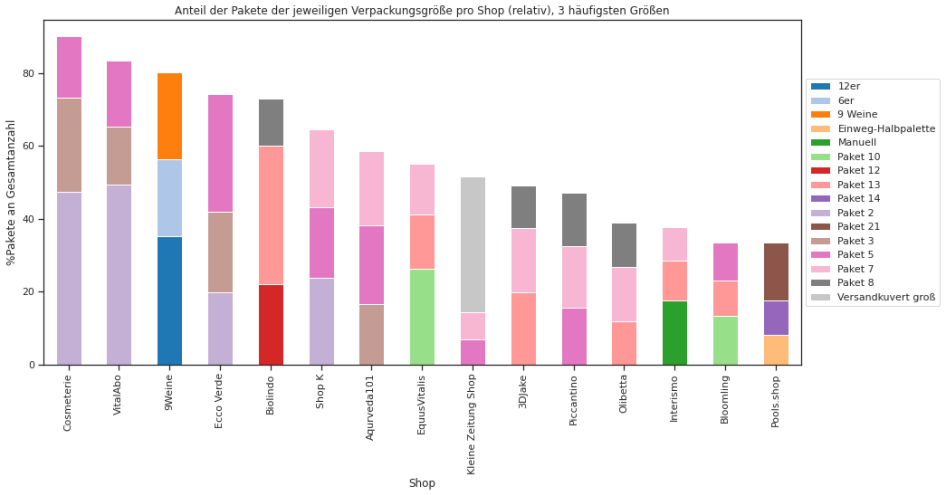


Abbildung 25: Anteil der Pakete der jeweiligen Verpackungsgröße pro Shop, drei häufigsten Größen, Quelle: Eigene Darstellung

Abbildung 25 zeigt, dass die Shops *VitalAbo* und *Ecco Verde* trotz der großen Anzahl an Paketen eine relativ geringe Anzahl an verschiedenen Verpackungsgrößen aufweisen. Weiters kann aus dem Diagramm erschlossen werden, dass bereits mit nur drei Verpackungsgrößen aus der Vielzahl an verfügbaren Größen oft ein Großteil der Pakete abgedeckt werden kann. In Shops, bei denen die Produkte im Durchschnitt eher klein sind (*Ecco Verde, Cosmeterie, VitalAbo*) werden die Lieferungen meist in den Paketen/Kartons 2, 3 oder 5 versendet.

Auch der Lieferdienst könnte für die Vorhersage einer Verpackungsgröße relevant sein, da manche Größen nicht oder nur mit einem bestimmten Lieferdienst versendet werden dürfen. Abbildung 26 zeigt die Gesamtanzahl der Pakete pro Lieferdienst und es ist ersichtlich, dass Lieferdienste, welche hauptsächlich Verpackungsgrößen des Typs Karton versenden (DHL, Post.at, BRT, Post.ch), die größte Anzahl der Pakete aufweisen.

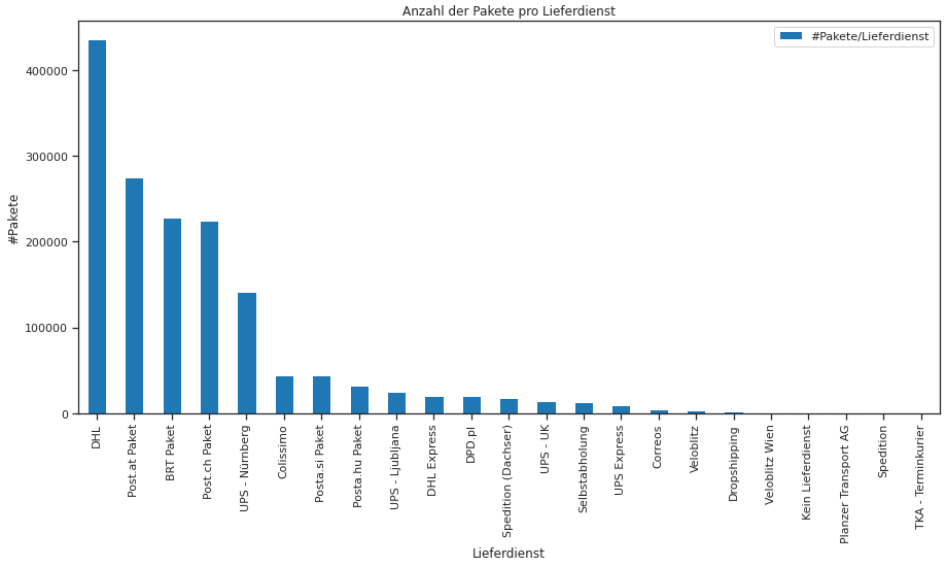


Abbildung 26: Anzahl der Pakete pro Lieferdienst, Quelle: Eigene Darstellung

Im nächsten Schritt wird die Anzahl der Pakete noch zusätzlich zum Lieferdienst auf die Verpackungsgröße bezogen. Dabei werden wie zuvor die Verpackungsgrößen auf die drei häufigsten Größen eingeschränkt.

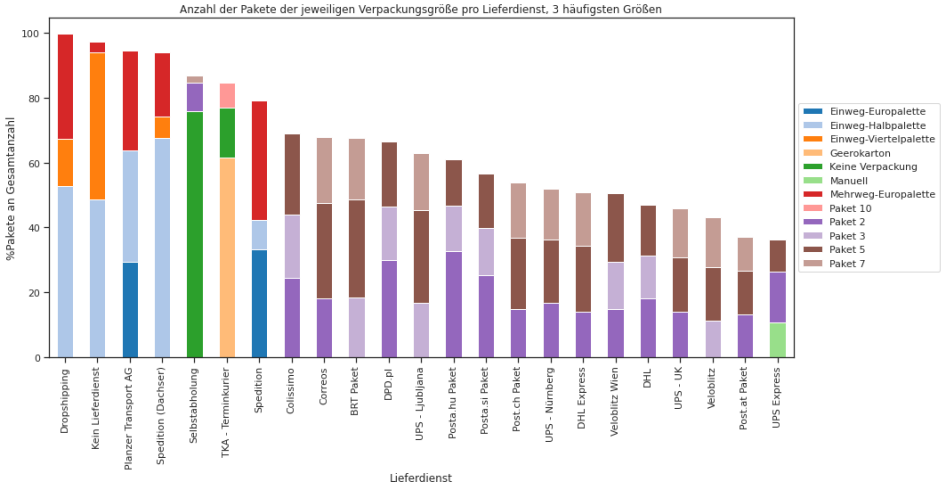


Abbildung 27: Anzahl der Pakete der jeweiligen Verpackungsgröße pro Lieferdienst, drei häufigsten Größen, Quelle: Eigene Darstellung

Wie bereits zuvor erwähnt, kann man aus dem Diagramm in Abbildung 27 entnehmen, dass Kartons den Großteil aller Verpackungsgrößen ausmachen. Weiters ist zu erkennen, dass die Lieferdienste *Dropshipping, Planzer* und *Spedition Dachser* hauptsächlich Paletten versenden. Selbstabholungen werden meist nicht verpackt. Auch auf den Lieferdienst bezogen werden über 50 % der Pakete nur in drei verschiedenen Größen verpackt. Jedoch ist zu beachten, dass der Lieferdienst oft auch sehr stark vom Shop abhängt.

**Ausreißerbehandlung – Gewicht und Anzahl der Produkte** Wie zuvor erwähnt kann das Volumen der Produkte nicht als Eigenschaft zur Vorhersage verwendet werden. Das Gewicht des Pakets sowie die Anzahl der Produkte könnten Größen sein, welche mit dem Volumen und in weiterer Form mit der Verpackungsgröße in Zusammenhang stehen. Ein Paket mit einem größeren Gewicht und einer höheren Anzahl an Produkten wird in den meisten Fällen in einer größeren Verpackungsgröße verpackt werden. Dieser Zusammenhang wird in Abbildung 28 dargestellt. In diesem Diagramm werden die drei verschiedenen Größen (klein, mittel, groß) der Frischboxen, auch *Foodmailer* genannt, als Histogramm dargestellt. Dabei bestätigt sich der zuvor definierte Zusammenhang: Je mehr Produkte im Paket und je größer das Gewicht des Pakets, desto größer muss die Verpackung sein, in welcher die Produkte verpackt werden.

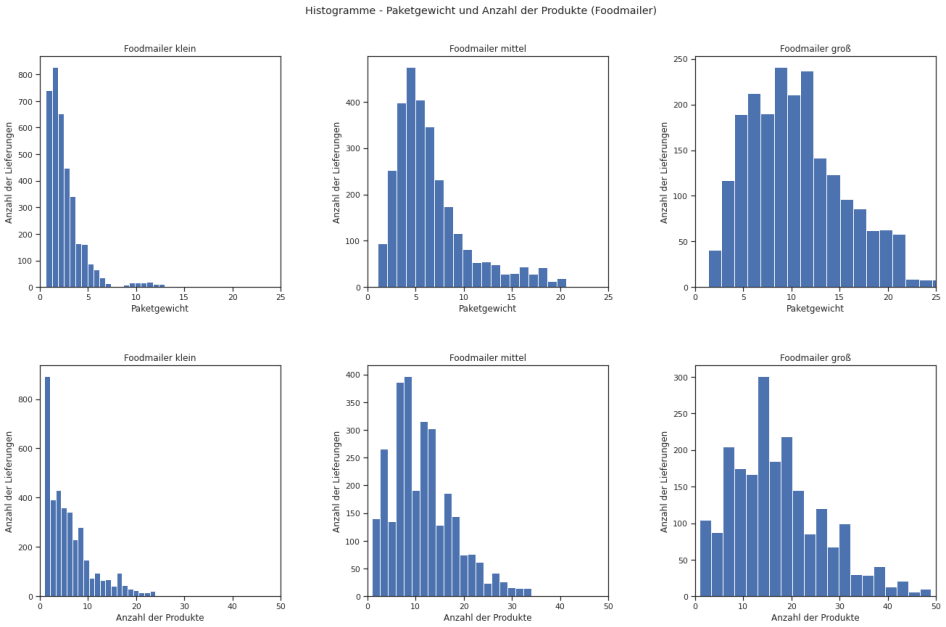


Abbildung 28: Histogramme - Paketgewicht und Anzahl der Produkte (Foodmailer), Quelle: Eigene Darstellung

Zu Beginn dieses Abschnitts wurde auch erwähnt, dass im Rahmen der Datenvorverarbeitung mit Hilfe der explorativen Datenanalyse auch Ausreißer entfernt werden können. Hierbei werden die numerischen Größen Paketgewicht und Produktanzahl im Paket hinsichtlich Ausreißer untersucht. Ausreißer sind Datenpunkte, die sich erheblich von anderen Beobachtungen in einem bestimmten Datensatz unterscheiden. Ausreißer können unter anderem durch menschliche Fehler während der Datenerfassung, -aufzeichnung oder -eingabe verursacht werden. In Bezug auf das Gewicht und die Anzahl der Produkte eines Pakets kann ein Ausreißer durch die Auswahl einer falschen Verpackungsgröße entstehen. Es gibt verschiedene Kategorien von Ausreißern und Methoden, wie Ausreißer identifiziert werden können. Wenn eine einzelne Dateninstanz in Bezug auf den Rest der Daten als anomal angesehen werden kann, wird die Instanz als Punktausreißer bezeichnet.[139] Punktausreißer können mit Hilfe von Grafiken wie zum Beispiel Boxplots, Histogramme oder Verteilungsplots visualisiert werden. Hinsichtlich des Paketgewichts und der Anzahl der Produkte muss jede Verpackungsgröße einzeln analysiert werden, da diese unterschiedliche Verteilungen in Bezug auf diese Größen aufweisen. In der Abbildung 29 werden die Boxplot Diagramme dreier Verpackungsgrößen dargestellt.

[139] vgl. Singh und Upadhyaya 2012, S. 307-310.

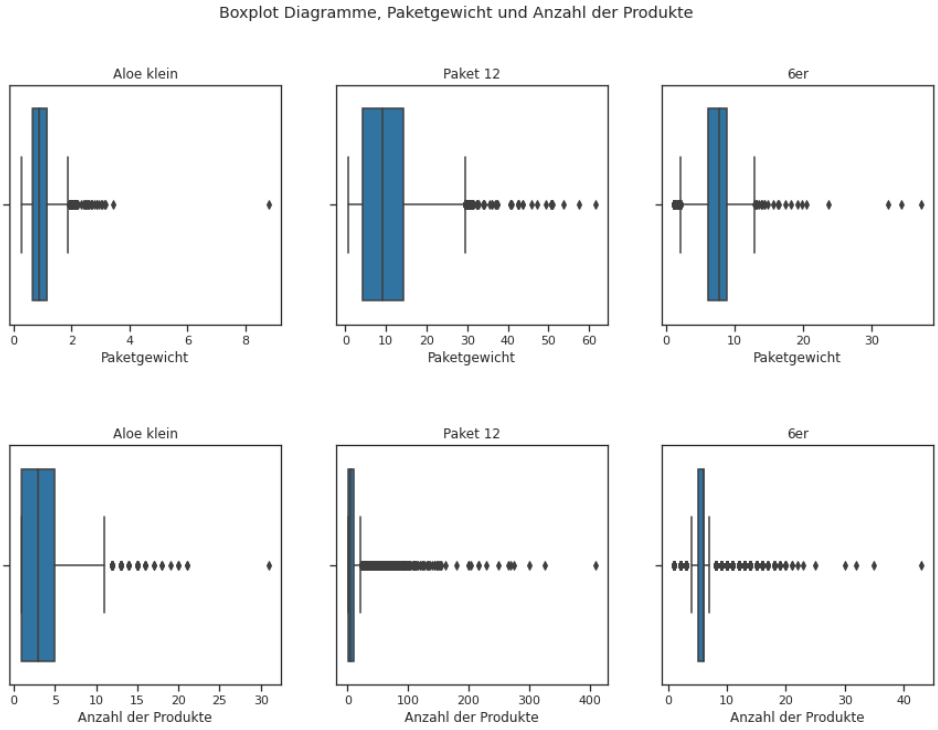


Abbildung 29: Boxplot Diagramme, Paketgewicht und Anzahl der Produkte, Quelle: Eigene Darstellung

Aus den Boxplot-Diagrammen geht hervor, dass die jeweiligen Verpackungsgrößen hinsichtlich des Paketgewichts als auch der Anzahl der Produkte in einem Paket Ausreißer aufweisen. Diese wurden daraufhin stichprobenartig untersucht. Dabei wurde entdeckt, dass es sich bei den Ausreißern in Bezug auf das Paketgewicht in den meisten Fällen um eine falsche Auswahl der Paketgröße seitens des Benutzers handelt. Bei den Ausreißern hinsichtlich der Anzahl der Produkte in einem Paket handelte es sich meist tatsächlich um valide Daten. Es gibt einige Produkte, welche relativ klein sind und in großen Mengen bestellt werden und dadurch von der Norm abweichen.

Eine weitere Möglichkeit, Ausreißer zu identifizieren, ist mit Hilfe des *Z-Werts*. Diese Methode setzt voraus, dass die Variable eine *Gaußsche Verteilung* hat. Der Z-Wert gibt mit Hilfe der Standardabweichung an, wie viel der Wert einer Beobachtung, ein Datenpunkt, vom Mittelwert abweicht. Ein Z-Wert von 3 bedeutet, dass der Datenpunkt 3 Standardabweichungen vom Mittelwert entfernt ist.

$$
\begin{aligned}
z &= \frac{(x - \mu)}{\sigma} \\
\text{mit } z &= \text{Z-Wert} \\
x &= \text{Datenpunkt} \\
\mu &= \text{Mittelwert} \\
\sigma &= \text{Standardabweichung}
\end{aligned}
\tag{6}
$$

Datenwerte, deren Z-Wert größer als ein Schwellenwert ist, werden als Ausreißer deklariert. In der Praxis wird dieser Schwellenwert oft als 3 angenommen. Für eine normalverteilte Größe tritt ein Z-Wert größer als 3 nur mit einer Wahrscheinlichkeit von 0,0027 (dies entspricht etwa 1 von 370 Beobachtungen).[140] [141]
Abbildung 28 zeigt, dass das Paketgewicht als gaußverteilt angenommen werden kann. Daher werden die Datenpunkte pro Verpackungsgröße, welche einen Z-Wert größer als 3 haben, aus den Trainingsdaten entfernt. Insgesamt werden dadurch *11.896* Datenpunkte, welche Pakete repräsentieren, entfernt. Dies entspricht *0,76 %* aller Pakete. Datenpunkte, welche einen Z-Wert größer als 3 beziehungsweise kleiner als -3 hinsichtlich der Anzahl der Produkte aufweisen, werden auf Grund der vorhin beschriebenen Erkenntnis nicht eliminiert.

### 3.1.3 Datentransformation

Die Datentransformation ist der letzte Schritt der Vorbereitung von Daten für maschinelle Lernaufgaben und wird auch als *Feature Engineering* bezeichnet. Beim Feature Engineering werden relevante Eigenschaften, Features, aus Rohdaten extrahiert und in Formate umgewandelt, die für das Modell des maschinellen Lernens besser geeignet sind. Dies ist ein entscheidender Schritt beim maschinellen Lernen, da die richtigen Features es ermöglichen, qualitativ hochwertigere Ergebnisse zu erreichen.[142] Die Daten können durch *Skalierung*, *Zerlegung* oder *Aggregation* transformiert werden. Die meisten Datensätze enthalten Werte, die sich in Bezug auf den Datenbereich oder Einheiten unterscheiden. Wenn solche nicht standardisierten Daten in Algorithmen für maschinelles Lernen eingespeist werden, führt dies zu einer schlechteren Leistung, da Attribute mit größeren numerischen Werten möglicherweise als wichtiger angesehen werden. Daher ist meist eine Skalierung erforderlich, um diesen Effekt zu unterdrücken, indem alle Merkmale auf ein ähnliches Maß normalisiert

[140] vgl. Ilyas und Chu 2019, S. 19.
[141] vgl. Frost o. D.
[142] vgl. Zheng und Casari 2018, S. 1.

werden.[143] In Bezug auf die Vorhersage der Verpackungsgröße werden das Paketgewicht sowie die Anzahl der Produkte im Paket normalisiert. Sind Werte einer Eigenschaft zusammengesetzt oder gruppiert, kann es für ein Machine-Learning-Modell sinnvoller sein, diese in verschiedene Bestandteile zu zerlegen. Zum Beispiel können einzelne wichtige Eigenschaften des Pakets durch eine komma-getrennte Liste an Eigenschaftswerten dargestellt werden. Zum Beispiel kann die Paketeigenschaft *Frischversand* aus der komma-getrennten Liste als Eigenschaften des Pakets rausgelöst werden. Andererseits kann Feature Aggregation dazu verwendet werden, verwandte Merkmale zusammenzuführen und die Dimensionalität eines Eingabesatzes zu verringern.[144] Eine Form der Feature Aggregation wurde in Form des Gruppierens und Summierens der Anzahl der einzelnen Produkte zur Gesamtanzahl der Produkte in einem Paket bereits durchgeführt.

Generell sei zu erwähnen, dass die in diesem Abschnitt beschriebenen Schritte der Datenaufbereitung, speziell das Feature Engineering, auch beim Entwickeln des spezifischen Modells im folgenden Kapitel gegebenenfalls nochmals durchlaufen werden müssen. Verschiedene Machine-Learning-Modelle und -Algorithmen benötigen unterschiedliche Eigenschaften in unterschiedlichen Formaten.

## 3.2 Modellentwicklung und -evaluierung

Zu Beginn dieses Kapitels wurde der Prozess, ein Machine-Learning-System zu entwickeln, in die drei Schritte *Data Engineering, Model Engineering* und *Model Deployment* eingeteilt. Nachdem die notwendigen Daten bezüglich der Lieferungen erfasst und aufbereitet sind, werden auf Basis dieser Daten Modelle zur Vorhersage von Verpackungsgrößen entworfen und evaluiert. Dabei ist zu erwähnen, dass aufgrund der Tatsache, dass die Shops voneinander unabhängig sind und Lieferungen immer im Kontext eines Shops verpackt werden, für jeden Shop ein eigenes Modell zur Vorhersage erstellt wird.

Die Modellentwicklung kann in zwei Subprozesse, die zu einem endgültigen Modell führen, eingeteilt werden: Modell-Training und Modell-Evaluierung. Der Zusammenhang dieser Subprozesse wird in Abbildung 30 dargestellt.

[143] vgl. Opidi 2019.
[144] vgl. Opidi 2019.

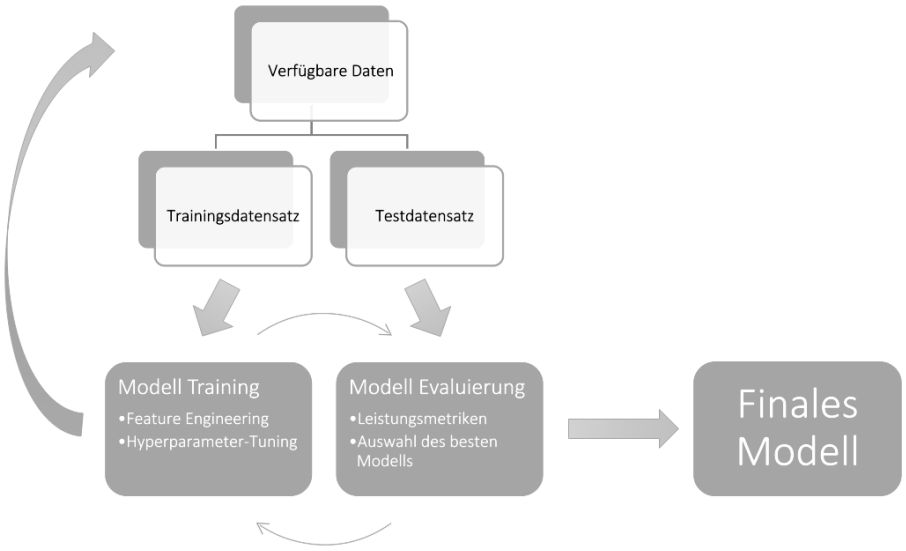


Abbildung 30: Prozess der Modellentwicklung und -evaluierung, Quelle: In Anlehnung an Visengeriyeva, 2021[145]

Im ersten Schritt werden beim **Modell-Training** Algorithmen für maschinelles Lernen auf Trainingsdaten angewendet, um ein Modell zu lernen. Dazu gehören auch das Feature Engineering, welches im nächsten Kapitel genauer in Bezug auf die Problemstellung erklärt wird, und das sogenannte *Hyperparameter-Tuning*. Ein Hyperparameter ist ein Parameter eines Lernalgorithmus, welcher vor dem Training festgelegt und vom Lernalgorithmus selbst nicht beeinflusst wird und konstant bleibt.[146] Hyperparameter beschreiben die Modellarchitektur und dienen der Steuerung des Lernalgorithmus. Das Finden geeigneter Hyperparameter ist ein wichtiger Schritt der Modellentwicklung, da diese einen großen Einfluss auf die Genauigkeit der Vorhersage haben können.[147] Je nach Algorithmus gibt es unterschiedliche Hyperparameter und einige davon werden in Abschnitt 3.2.6 beschrieben.

Nachdem das Modell trainiert wurde, findet die **Modell-Evaluierung** statt. Dabei wird sichergestellt, dass die Genauigkeit der Vorhersage der Verpackungsgröße bei neuen Lieferungen ausreicht, um das Modell tatsächlich dem Endbenutzer im Lager zur Verfügung stellen zu können. Die einzige Möglichkeit zu wissen, wie gut ein Modell auf unbekannte

[145] Visengeriyeva et al. 2021, vgl.
[146] vgl. Géron 2019, S. 29.
[147] vgl. Anonym 2018.

Lieferungen verallgemeinert, besteht darin, es tatsächlich an neuen Instanzen auszuprobieren. Dafür werden die Daten in einen Trainings- und Testdatensatz aufgeteilt. Das Modell wird mit dem Trainingsdatensatz trainiert und anschließend mit dem Testdatensatz evaluiert. Durch Auswerten des Modells auf dem Testdatensatz kann der in Kapitel 2.4.1 erläuterte Generalisierungsfehler, der angibt, wie sich das Modell auf unbekannten Instanzen verhält, geschätzt werden. Ein üblicher Ansatz in der Praxis, um ein Modell zu bewerten, besteht darin, 80 % der Daten für das Training zu verwenden und 20 % für die Evaluierung bereitzustellen. Diese Aufteilung ist jedoch abhängig vom Umfang der zur Verfügung stehenden Daten.[148]

Wie in Abbildung 30 dargestellt, handelt es sich bei der Modellentwicklung um einen iterativen Prozess. Ist die Leistung eines entwickelten Modells – gemessen durch eine sogenannte *Leistungsmetrik* – nicht ausreichend, wird das Modell angepasst oder ein neues Modell mit Hilfe eines anderen Algorithmus trainiert und danach wieder evaluiert. Dabei kann es auch notwendig sein, die Datengrundlage zu erweitern, um weitere Features zu erhalten. Das beste Modell hinsichtlich einer definierten Leistungsmetrik in Bezug auf die Vorhersage einer neuen Lieferung wird dann tatsächlich im Lager eingesetzt.

### 3.2.1 Leistungsmetriken

Es gibt einige Metriken, die sich als nützlich erweisen, um die Leistung eines Klassifikators mit mehreren Klassen zu testen. In diesem Abschnitt werden *Confusion-Matrix* und *Genauigkeit (Accuracy)* als mögliche Leistungsmetriken vorgestellt. Weiters wird erklärt, warum die Genauigkeit zur Bewertung der Modelle in Kapitel 3.2.7 verwendet wird.

**Confusion-Matrix**
Viele Metriken basieren auf der sogenannten *Confusion-Matrix*, da diese alle relevanten Informationen zur Klassifikatorleistung enthält. Die Confusion-Matrix entspricht einer Tabelle, welche die tatsächlichen und vorhergesagten Klassen gegenüberstellt. Die Klassen werden in den Zeilen wie in den Spalten in der gleichen Reihenfolge aufgelistet. Der Wert der Zelle in Zeile $i$ und Spalte $j$ entspricht der Anzahl, wie oft für eine Instanz der tatsächlichen Klasse $i$, die Klasse $j$ vorhergesagt wurde. Daher entsprechen die Werte der Zellen in der Hauptdiagonale der Anzahl, wie oft eine Klasse richtig vorhergesagt wurde. Die Werte der Zellen, welche nicht auf der Hauptdiagonale liegen, entsprechen der Anzahl der falsch vorhergesagten Klassen. Ein Beispiel einer Confusion-Matrix wird in Abbildung 31 in Kapitel 3.2.7 dargestellt.

[148] vgl. zu diesem Abschnitt Géron 2019, S. 30f.

**Accuracy**
Die *Accuracy (Genauigkeit)* ist eine der Metriken, welche für die Mehrklassen-Klassifizierung häufig verwendet wird. Sie ist definiert als der Prozentsatz der korrekten Vorhersagen. Wenn mit unausgeglichenen Datensätzen gearbeitet wird, bei denen die meisten Instanzen einer einzelnen Klasse zugeordnet sind, ist es nicht immer ratsam, die Genauigkeit als Metrik zu verwenden, da starke Klassifizierungsfehler für Klassen mit wenigen Instanzen vernachlässigt werden.[149] [150]

Obwohl die Datenanalyse zeigt, dass hinsichtlich der Klassen ein unausgeglichener Datensatz vorliegt, wird für den Vergleich und die Evaluierung der Modelle zur Vorhersage von Verpackungsgrößen die Genauigkeit als Metrik verwendet. Die Genauigkeit wird verwendet, da das Ziel ist, beim Verpackungsprozess durch eine korrekte Vorhersage der Verpackungsgröße Zeit zu sparen und so viele Größen wie möglich richtig vorherzusagen. Dabei sind selten vorkommende Größen tatsächlich weniger relevant.

### 3.2.2 Verwendete Technologien

Wie bei der explorativen Datenanalyse zuvor wurde die Modellentwicklung mit Hilfe eines *Google Colab IPython Notebooks* mittels *Python* durchgeführt. Dabei wurde die Open-Source-Bibliothek *Scikit-learn* für das Trainieren und Evaluieren der Modelle sowie zum Feature Engineering verwendet. Scikit-learn bietet state-of-the-art Implementierungen vieler bekannter Algorithmen für maschinelles Lernen und basiert auf den bereits genannten Python-Bibliotheken: *NumPy*, *SciPy* und *Matplotplib*. Mit Hilfe von Scikit-learn können Modelle für eine bestimmte Problemstellung relativ schnell und einfach erstellt werden.[151] In Abschnitt 3.2.6 wird ein Python Code präsentiert, in welchem die Scikit-learn Bibliothek verwendet wird, um ein Modell zur Vorhersage von Verpackungsgrößen zu entwickeln.

### 3.2.3 Feature Engineering

Im Abschnitt 3.1.3 wurde bereits erläutert, dass die Auswahl von relevanten Eigenschaften – den Features – aus den zur Verfügung stehenden Daten einen großen Einfluss auf die Leistung des Machine-Learning-Modells hat. Features sind von der Art der verfügbaren Daten und auch vom Modell abhängig, da einige Modelle für einige Arten von Features besser geeignet sind als andere. Beim Feature Engineering werden die am besten geeigneten Features unter Berücksichtigung der Daten, des Modells und der Problemstellung ausgewählt. Die Anzahl der Features ist ebenfalls

[149] vgl. zu diesem Abschnitt Jordan 2017.
[150] vgl. zu diesem Abschnitt Grandini, Bagli und Visani 2020, S. 2f.
[151] vgl. Pedregosa et al. 2012, S. 2826.

wichtig. Wenn nicht genügend relevante Features vorhanden sind, kann es zur bereits genannten Unteranpassung kommen. Wenn es zuviele Features gibt oder die meisten davon irrelevant sind, ist das Modell schwieriger zu trainieren und es kann zur Überanpassung kommen.[152]

Wie in der explorativen Datenanalyse in Abbildung 28 im Abschnitt 3.1.2 dargestellt, stehen die Verpackungsgrößen sowohl mit dem **Gewicht der Lieferung** als auch der **Anzahl der Produkte in der Lieferung** in Zusammenhang. Lieferungen mit einem größeren Gewicht und einer größeren Anzahl an Produkten werden in den meisten Fällen in eine größere Verpackungsgröße verpackt. Deshalb wurden diese beiden numerischen Größen als Features ausgewählt. Dabei ist jedoch zu beachten, dass diese je nach Algorithmus vor dem Training eventuell standardisiert werden müssen. Weiters wurde bereits erwähnt, dass es eigene Kartons speziell für den Versand von Frisch- und Kühlware gibt. Da diese sich hinsichtlich des Volumens nicht von anderen Kartons unterscheiden, wurde aus den Lieferoptionen die Option **Frischversand** als sogenanntes *Flag* extrahiert und als Feature zum Trainieren der Modelle verwendet. Dieses Flag gibt an, ob es sich um Frischversand handelt oder nicht. Auf dieselbe Weise wurde das **Selbstabholung-Flag** als Feature definiert. Dieses Flag ist für das Trainieren des Modells wichtig, da, wenn die Lieferung selbst abgeholt wird, in der Regel *keine Verpackung* verwendet wird.

**Approximiertes Liefervolumen**
Im Rahmen der Datenanalyse wurde festgestellt, dass nur für 5,67 % der Produkte die Dimensionen (Höhe, Breite, Tiefe) hinterlegt sind. Daher kann nur durch Summieren der vorhandenen Volumen der Produkte kein aussagekräftiges Volumen der Lieferung vorab bestimmt werden. Im Rahmen des Feature Engineerings entstand jedoch die Idee, das Volumen der Produkte auf Basis des bekannten Volumens der Verpackungsgrößen, in denen diese Produkte versendet wurden, zu approximieren. Die relevanten Größen sind dabei das Produktgewicht, das Gewicht der Lieferung und das Volumen der Verpackung. Zur Bestimmung des approximierten Volumens eines Produktes in einer Lieferung wird das Volumen der Verpackung durch das Gesamtgewicht des Pakets dividiert und mit dem Produktgewicht multipliziert. Das heißt, das Produktvolumen wird auf Basis des Gewichtes der Produkte approximiert. Diese Berechnung wird in den folgenden

[152] vgl. zu diesem Abschnitt Zheng und Casari 2018, S. 3.

Gleichungen dargestellt:

$$\begin{aligned} V_{i,k} &= \frac{V_k}{G_k} \cdot G_i \\ \text{mit } V_{i,k} &= \text{approximiertes Volumen eines Produktes i} \\ &\quad \text{in der Lieferung k} \\ V_k &= \text{Volumen der Verpackung der Lieferung k} \\ G_k &= \text{Gewicht der Lieferung k} \\ G_i &= \text{Gewicht des Produkts i} \end{aligned} \tag{7}$$

Da ein Produkt nicht nur in einer Lieferung, sondern in mehreren Lieferungen versendet wird, wird zur Bestimmung des endgültigen approximierten Produktvolumens $V_i$ der Median aller approximierten Produktvolumen verwendet, da dieser hinsichtlich Ausreißern weniger empfindlich ist als der Durchschnitt.

$$V_i = \underset{k}{\text{median}} \{V_{i,k}\} \tag{8}$$

Um nun das *approximierte Volumen einer Lieferung* als Feature für das Vorhersagemodell zu erhalten, wurden die approximierten Volumen $V_i$ aller Produkte einer Lieferung $L$ aufsummiert:

$$V_{approximiert} = \sum_{i \in L} V_i \tag{9}$$

**Ausgewählte Features**
Zusammenfassend wurden zum Training der Modelle in Abschnitt 3.2.6 die folgenden fünf Features verwendet:

- Gewicht der Lieferung
- Anzahl der Produkte in der Lieferung
- Approximiertes Liefervolumen
- Frischversand-Flag
- Selbstabholung-Flag

Dabei ist zu erwähnen, dass diese Features das Ergebnis eines iterativen Feature-Engineering-Prozesses sind. Es wurden Modelle mit unterschiedlichen Kombinationen aus diesen und auch anderen Features wie zum Beispiel dem Lieferdienst, dem Lieferland oder dem Gesamtpreis trainiert und hinsichtlich der Genauigkeit evaluiert. Dabei wurde die höchste Genauigkeit durch das Trainieren mit den ausgewählten fünf Features erreicht.

### 3.2.4 Anforderungen an die Vorhersagegenauigkeit

Die Datenanalyse zeigte, dass die Klassen der gegebenen Daten ungleichmäßig verteilt sind. Bei über 40 verschiedenen Verpackungsgrößen kann mit nur drei verschiedenen Größen in vielen Shops ein Großteil aller Pakete abgedeckt werden. Deshalb werden die in Abschnitt 3.2.6 trainierten Modelle, welche neben der vorhergesagten Klasse auch eine Wahrscheinlichkeit für die Klassenzugehörigkeit ausgeben, auf Basis zweier Kriterien evaluiert. Zunächst wird die Genauigkeit bestimmt, dass die tatsächliche Größe mit der vorhergesagten Klasse übereinstimmt. Weiters wird die Genauigkeit ermittelt, dass sich die tatsächliche Größe unter den drei Klassen mit den höchsten Wahrscheinlichkeiten befindet. Das ist für das Unternehmen in Bezug auf den Verpackungsprozess von großer Relevanz, da man dem Mitarbeiter aus der Vielzahl an verfügbaren Verpackungsgrößen die drei häufigsten zur Auswahl geben kann. Das heißt, auch wenn die vorhergesagte Verpackungsgröße nicht richtig ist, muss der Mitarbeiter während des Verpackens nicht lange nach der korrekten Verpackungsgröße suchen, sondern bekommt diese mit zwei weiteren Größen mit hoher Wahrscheinlichkeit bereits vorgeschlagen. Aus diesem Grund werden in Absprache mit dem Unternehmen und der Lagerleitung zwei Anforderungen an die Vorhersagegenauigkeit der entwickelten Modelle gestellt: Zum einen kann ein Modell generell zum Vorschlag von Verpackungsgrößen eingesetzt werden, wenn die Genauigkeit, dass sich die tatsächliche Größe unter den drei vorhergesagten Größen befindet, über 80 % liegt. Liegt zusätzlich die Genauigkeit bei der Vorhersage *einer* Größe bei über 80 %, kann diese im Verpackungsprozess bereits vorausgewählt werden (siehe Abbildung 3 in Abschnitt 2.3).

Für die Bewertung des Baseline-Algorithmus im folgenden Abschnitt sowie zum Vergleich der Modelle in Kapitel 3.2.7 werden die Ergebnisse für die Shops *9Weine, Vitalabo, Ecco Verde* und *Piccantino* präsentiert. Es wurden diese Shops ausgewählt, da diese repräsentativ das gesamte Spektrum an Shops bezüglich erreichter Genauigkeit recht gut abdecken.

### 3.2.5 Baseline

Bevor komplexere Modelle erstellt und hinsichtlich der Vorhersagegenauigkeit verglichen werden, ist es wichtig, eine sogenannte *Baseline* zu definieren. Eine Baseline ist eine einfache Methode, die Heuristiken, zusammenfassende Statistiken, Zufälligkeit oder maschinelles Lernen verwendet, um Vorhersagen zu treffen.[153] Die Genauigkeit (Accuracy) dieser Baseline dient zum Vergleich mit komplexeren Modellen. Ein Algorithmus, welcher häufig als Baseline für Klassifizierungsprobleme verwendet wird, ist der

[153] vgl. Aditya 2018.

sogenannte *Zero Rule (ZeroR, 0-Rule)* Algorithmus, der die am häufigsten vorkommende Klasse im Datensatz vorhersagt. Wenn 65 % der Instanzen diese Klasse haben, würde ZeroR davon ausgehen, dass alle Instanzen diese Klasse haben und die entsprechende Genauigkeit würde 65 % betragen. ZeroR ist ein einfacher und effektiver Baseline-Algorithmus. Wenn ein komplexeres Modell eine niedrigere Genauigkeit als der ZeroR aufweist, ist es als Machine-Learning-Modell für den Einsatz nicht geeignet.[154]

Für den Zero-R-Algorithmus wurde die Genauigkeit, dass die tatsächliche Größe mit der häufigsten vorkommenden Größen übereinstimmt, sowie die Genauigkeit, dass sich die tatsächliche Größe unter den drei häufigsten Größen befindet, bestimmt. Wie die Tabelle 7 zeigt, ist der ZeroR-Algorithmus ein guter Baseline-Algorithmus für einen Vergleich mit komplexeren Modellen, da die Genauigkeit bei der Vorhersage einer Verpackungsgröße trotz über 40 verschiedener Klassen aufgrund der ungleichmäßigen Klassenverteilung zum Beispiel bei *Vitalabo* fast 50 % beträgt. Die Genauigkeit, dass sich die tatsächliche Größe unter den drei häufigsten Größen befindet, liegt bei den Shops *9Weine* und *Vitalabo* bereits bei über 80 %, bei *Ecco Verde* trotz der hohen Anzahl an Lieferungen bei fast 75 %. Nur der Shop *Piccantino* liegt bei unter 50 %.

| Modell \ Shop | | 9Weine | VitalAbo | Ecco Verde | Piccantino |
|---|---|---|---|---|---|
| Baseline | Genauigkeit (1 Größe) | 35,41% | 49,25% | 32,49% | 17,36% |
| | Genauigkeit (3 Größen) | 81,76% | 83,20% | 74,21% | 47,33% |

Tabelle 7: Vorhersagegenauigkeit der Baseline

Es wurden auch alternative Baseline-Algorithmen, wie zum Beispiel eine zufällige Vorhersage der Verpackungsgröße oder eine Vorhersage auf Basis des approximierten Volumens entwickelt und evaluiert. Es zeigte sich jedoch, dass die Auswahl der häufigsten bzw. der drei häufigsten Klassen die höchste Genauigkeit aufweist. Deshalb werden diese Ergebnisse im nächsten Abschnitt dazu verwendet, um die Leistung der gelernten Modelle zu vergleichen.

### 3.2.6 Training der Modelle

Nachdem die Daten aufbereitet sowie die Features und Baseline definiert sind, können verschiedene Modelle zur Vorhersage trainiert und evaluiert werden. Dafür wurden die verfügbaren Daten bezüglich der Lieferung für jeden Shop zunächst, wie in Abbildung 30 dargestellt, in einen Trainings- und Testdatensatz aufgeteilt. Quelltext 2 zeigt, dass eine zufällige Aufteilung der Daten in einen Trainingsdatensatz, welcher 80 % und einen

[154] vgl. Msg systems ag - Applied Technology Research. o. D.

Testdatensatz, welcher 20 % umfasst, erfolgte. Mit diesen Trainingsdaten wurde für jeden in Kapitel 2.4.1 beschriebenen Algorithmus und für jeden Shop ein Modell gelernt.

**Training mit Scikit-learn und Google Colab**
Wie bereits erwähnt, bietet *Scikit-learn* zusammen mit *Google Colab* eine Möglichkeit, verschiedene Modelle einfach mittels bereits implementierter Algorithmen auf den Cloud-Servern von Google zu trainieren. Beim Training konnte dabei auf die Hardware-Ressourcen von Google wie zum Beispiel GPUs, TPUs sowie 35 GB RAM zugegriffen werden. Im Allgemeinen ermöglichten diese Ressourcen ein schnelles Training der Modelle. Das Training eines Modells des Shops *Ecco Verde* mit 581.048 Trainingsinstanzen benötigte mit den Algorithmen k-nächste-Nachbarn, Naive Bayes und Entscheidungsbaum nur wenige Sekunden. Der Random Forest benötigte dabei in etwa drei Minuten. Die in der Theorie genannte Tatsache, dass das Training von neuronalen Netzen und Support Vector Machines sehr viel Zeit in Anspruch nimmt, konnte bestätigt werden. So benötigte das Training eines sehr einfachen neuronalen Netzes bereits über eine Stunde. Das Training von Support Vector Machines nahm mehrere Stunden in Anspruch.

**kNN-Algorithmus**
Aufgrund der Tatsache, dass der kNN-Algorithmus einfach zu verstehen und das Training sehr schnell ist, wurde dieser als Erstes implementiert. Die Hyperparameter des kNN Algorithmus sind die Anzahl der Nachbarn k und die Art der Abstandsmetrik. Vor dem Training müssen die numerischen Features *Gewicht der Lieferung, Anzahl der Produkte* pro Lieferung und *Approximiertes Liefervolumen* skaliert werden. Die Implementierung des Algorithmus in Scikit-learn ermöglicht auch die Ausgabe der Klassenwahrscheinlichkeiten.

**Naive Bayes**
In der Beschreibung des Naive-Bayes-Klassifikators in Abschnitt 2.4.3 wurde erwähnt, dass es von der Implementierung abhängt, ob die Features kategorisch oder numerisch sein können. Der Naive Bayes kann durch Annahme einer Gaußschen Verteilung für das Training mittels numerischer Features erweitert werden. Diese Erweiterung wird *Gaussian Naive Bayes* genannt. Sie ist auch in Scikit-learn verfügbar und wird für das Vorhersagemodell verwendet.[155] Es gibt weitere Implementierungen des Naive Bayes, doch Gaussian Naive Bayes erreichte bei den meisten Shops die höchste Genauigkeit. Die numerischen Features müssen vor dem Training nicht skaliert werden.

[155] vgl. Brownlee 2020.

**Decision Tree und Random Forest**
Entscheidungsbäume sind für die Vorhersage der Verpackungsgröße ein interessantes Modell, da neue Instanzen rasch klassifiziert werden können und zudem leicht interpretiert werden kann, warum einer Lieferung eine Verpackungsgröße zugewiesen wurde. Scikit-learn verwendet den *CART-Algorithmus* zum Erstellen eines Entscheidungsbaums. Dieser Algorithmus hat eine Vielzahl von Hyperparametern, wie zum Beispiel die maximale Tiefe oder die minimale Anzahl an Instanzen in einem Blatt, auf welche hier im Detail nicht näher eingegangen wird. Es ist zu erwähnen, dass es unzählige Kombinationen von Hyperparametern gibt, mit denen Entscheidungsbäume gelernt werden können. Diese Kombinationen selbst auszuprobieren, würde viel Zeit beanspruchen. Dafür gibt es eine sogenannte *Rastersuche* oder *Grid Search*, welche für den Random Forest zur Hyperparameteroptimierung durchgeführt wurde und am Ende dieses Abschnitts erläutert wird.

Die Implementierung eines Random-Forest-Algorithmus wird im Detail zum Schluss dieses Abschnitts erläutert, da ein mit diesem Algorithmus trainiertes Modell schlussendlich in das bestehende Softwaresystem integriert wurde. Hierbei wird auch die Hyperparameteroptimierung mittels Grid Search genauer beschrieben. Der Random Forest hat wie die Entscheidungsbäume eine Vielzahl von verschiedenen Hyperparametern.

**Neuronales Netz**
Scikit-learn bietet auch die Möglichkeit, neuronale Netzwerke zu implementieren. Diese Implementierung eines MLP-Klassifikators, welche den Backpropagation-Algorithmus zum Trainieren verwendet, hat jedoch keine GPU-Unterstützung. Dies führte dazu, dass das Training bereits einfacher Modelle viel Zeit in Anspruch nahm und somit eine Optimierung der Hyperparameter wie beim Random-Forest-Algorithmus nicht möglich war. In Kapitel 2.4.5 wurde zusätzlich zum Blackbox-Verhalten die Tatsache, dass die Struktur neuronaler Netzwerke auf Basis von Erfahrung ausgewählt werden muss, als weitere Schwäche bezeichnet. Aus diesen Gründen wurde nur ein einfaches Modell mit drei Zwischenschichten zu je zehn Neuronen und der ReLU-Aktivierungsfunktion trainiert. Dabei mussten die numerischen Features skaliert werden. Zusätzlich zu der Anzahl an Zwischenschichten, deren Anzahl an Neuronen und der Aktivierungsfunktion, gibt es noch einige weitere Hyperparameter, welche hier nicht im Detail erläutert werden.

**Support Vector Machine**
Ähnlich wie beim Trainieren von neuronalen Netzen ist auch das Trainieren einer Support Vector Machine zur Mehrklassen-Klassifikation sehr aufwendig. Durch die Vielzahl der verschiedenen Verpackungsgrößen ist es aufgrund der limitierten Rechenkapazität nicht möglich, eine Mehrklassen-Klassifikation mit dem in Kapitel 2.4.6 beschriebenen One-vs-One-Ansatz

durchzuführen, da das Training von Modellen auf der Google-Cloud auf 12 Stunden beschränkt ist. Deshalb wurde das Modell mit dem One-vs-Rest-Ansatz trainiert. Das heißt, für jede Verpackungsgröße wurde ein eigenes Modell erstellt. Da eine Hyperparameteroptimierung bei über 40 Modellen sehr rechen- und zeitintensiv ist, wurde wiederum nur ein vereinfachtes Modell trainiert. Es wurden die verschiedenen Kerneltypen (linear, RBF, polynomial, Sigmoid) ausprobiert.

**Implementierung eines Random-Forest-Algorithmus**
Im folgenden Abschnitt wird die Implementierung eines Random-Forest-Klassifikators erläutert. Zunächst wird, wie im Quelltext 1 dargestellt, eine sogenannte *Pipeline* definiert, welche die Features – je nachdem, ob es sich um numerische oder kategorische Größen handelt – transformiert. Bei den numerischen Features kann die Art der Skalierung als Parameter definiert werden. Kategorische Features werden mittels dem sogenannten *One-Hot-Encoding* transformiert. One-Hot-Encoding erstellt für jeden Wert eines kategorischen Features eine eigene Spalte. Je nach Feature-Wert der Instanz bekommt eine dieser Spalten eine 1, die anderen Spalten eine 0.[156]

```
def getPipeline(features_num, features_cat, scaler):
        pipe_num = Pipeline([('std_scaler', scaler)])

        full_pipeline = ColumnTransformer([
                        ("num", pipe_num, features_num),
                        ("cat", OneHotEncoder(handle_unknown = 'ignore'),
                                features_cat),
        ])
        return full_pipeline
```

Quelltext 1: Beispiel einer Scikit-learn Pipeline

Quelltext 2 zeigt die Funktion *predictByRandomForest(...)*, welche es ermöglicht, einen Random-Forest-Klassifikator zu trainieren und zu evaluieren. Diese Funktion kann mit den entsprechenden Daten für jeden Shop aufgerufen werden. Zuerst werden die verfügbaren Instanzen mit der Funktion *train_test_split(...)* in einen zufälligen Trainings- und Testdatensatz aufgeteilt und die Features mit der zuvor definierten Pipeline transformiert. Der *random_state* Parameter sorgt dafür, dass die Aufteilung in Trainings- und Testdatensätze für jeden Aufruf der *train_test_split(...)* Methode mit diesem Parameter gleich ist. Dadurch wurden alle Algorithmen mit den gleichen Trainingsinstanzen trainiert und Testinstanzen evaluiert. Anschließend wird die Struktur (Hyperparameter) des Modells durch den *RandomForestClassi-*

[156] vgl. Géron 2019, S. 67.

*fier* definiert und das Modell mit dem Trainingsdatensatz durch den Aufruf der Funktion *model.fit(...)* trainiert.

```
def predictByRandomForest(df_data, num_attribs, cat_attribs):
        #Split in train and test data and prepare the data
        labels = df_data["Packaging_Name"].to_numpy()
        X_train, X_test, y_train, y_test = train_test_split(df_data,
                                                labels,
                                                test_size=0.20,
                                                random_state=1)
        full_pipeline = getPipeline(num_attribs, cat_attribs, None)
        X_train_prepared = full_pipeline.fit_transform(X_train)

        #Train the model
        model = RandomForestClassifier(random_state=42, bootstrap=True,
                                max_depth=110, max_features=3,
                                min_samples_leaf=4,
                                min_samples_split=10,
                                n_estimators=100)
        model.fit(X_train_prepared, y_train)

        #Predict
        X_test_prepared = full_pipeline.transform(X_test)
        y_pred = model.predict(X_test_prepared)
        probabilities = model.predict_proba(X_test_prepared)

        #Evaluate the model
        accuracy = accuracy_score(y_test, y_pred)
        accuracy3 = getAccuracyOfNMostLikely(model.classes_,
                                        probabilities,
                                        y_test, 3)
        return [len(X_train.index),len(X_test.index),accuracy,accuracy3]
```

Quelltext 2: Training und Evaluierung eines Random-Forest-Klassifikators

Die Hyperparameter, welche die Struktur des Modells bestimmen, wurden durch eine Optimierung mit Hilfe Scikit-learn's *GridSearchCV* gefunden. Dabei muss angegeben werden, welche Hyperparameter mit welchen Werten ausprobiert werden sollen. Die Grid Search wertet dann alle möglichen Kombinationen von Hyperparameterwerten aus und wählt jene Kombination, welche zur höchsten Genauigkeit beim Trainingsdatensatz führt.[157] Die

[157] vgl. Géron 2019, S. 76.

ausgewählten Hyperparameter und gefundenen Werte sind im Quelltext 2 dargestellt.

### 3.2.7 Vergleich der gelernten Modelle

Damit entschieden werden kann, welche Modelle in welchen Shops in das bestehende Softwaresystem eingebunden werden können, werden in diesem Abschnitt die Vorhersagegenauigkeiten der gelernten Modelle verglichen. Für den Vergleich werden die gelernten Modelle mit den zurückgehaltenen 20 % der Daten der Lieferungen, welche nicht für das Training verwendet wurden, evaluiert.

**Evaluierung mit Scikit-learn Google Colab**
Wie bereits erwähnt, können mit der im Quelltext 2 definierten Methode die Modelle auch evaluiert werden. Für die Vorhersage und Evaluierung bietet Scikit-learn weitere vordefinierte Methoden. Mit *model.predict(...)* wird eine Vorhersage getätigt, und als Ergebnis werden die entsprechenden Verpackungsgrößen der zu klassifizierenden Lieferungen zurückgegeben. Die Funktion *model.predict_proba(...)* gibt die Wahrscheinlichkeiten für jede Größe zurück. Weiters wird in der Methode im Quelltext 2 die Vorhersagegenauigkeit mittels der Funktion *accuracy_score(...)* ermittelt. Die Wahrscheinlichkeit, dass die tatsächliche Größe sich unter den drei wahrscheinlichsten Größen befindet, wird mit Hilfe der selbst implementierten Funktion *getAccuracyOfNMostLikely(...)* ermittelt. Wie beim Training der Modelle zuvor benötigte auch die Evaluierung der meisten Modelle nicht viel Zeit. Die Klassifizierung von 145.262 Testinstanzen des Shops *Ecco Verde* mit Modellen, welche mit Naive-Bayes-, Entscheidungsbaum- oder Backpropagation-Algorithmus trainiert wurden, konnte in unter einer Sekunde durchgeführt werden. Der Random-Forest-Klassifikator benötigte dafür rund 15 Sekunden, der kNN-Klassifikator 25 Sekunden. Für die Evaluierung der Support Vector Machine musste die Klassifkation für jedes der über 40 erstellten Modelle durchgeführt werden. Das führte dazu, dass neben dem Training auch die Klassifikation mit knapp über einer Stunde sehr lange dauerte.

**Ergebnisse**
Im Folgenden werden die Ergebnisse der gelernten Vorhersagemodelle in den ausgewählten Shops präsentiert. Die Vorhersagegenauigkeiten dieser Modelle werden in Tabelle 8 dargestellt. Hierbei ist zu erwähnen, dass die Modelle mehrmals mit verschiedenen Trainings- und Testdaten trainiert und evaluiert wurden. Die Vorhersagegenauigkeit und der Unterschied zwischen den Genauigkeiten der trainierten Modelle blieb dabei für jede Aufteilung annähernd gleich. Die Ergebnisse einer zufälligen Aufteilung in Tabelle 8 sind somit repräsentativ für alle weiteren Aufteilungen.

| Modell \ Shop | | 9Weine | VitalAbo | Ecco Verde | Piccantino |
|---|---|---|---|---|---|
| Anzahl | Trainingsinstanzen | 16.908 | 124.177 | 581.048 | 50.788 |
| | Testinstanzen | 4.228 | 31.045 | 145.262 | 12.698 |
| kNN | Genauigkeit (1 Größe) | 94,06% | 80,43% | 66,41% | 58,17% |
| | Genauigkeit (3 Größen) | 97,28% | 93,26% | 90,19% | 83,26% |
| Naive Bayes | Genauigkeit (1 Größe) | 86,68% | 70,00% | 62,53% | 7,14% |
| | Genauigkeit (3 Größen) | 97,73% | 94,04% | 96,32% | 10,32% |
| Decision Tree | Genauigkeit (1 Größe) | 92,08% | 76,86% | 59,39% | 51,60% |
| | Genauigkeit (3 Größen) | 92,64% | 83,43% | 60,87% | 57,18% |
| Random Forest | Genauigkeit (1 Größe) | **94,47%** | **80,49%** | **68,98%** | **60,51%** |
| | Genauigkeit (3 Größen) | **99,15%** | **98,76%** | 98,20% | **92,24%** |
| Neuronales Netz | Genauigkeit (1 Größe) | 92,57% | 76,91% | 68,69% | 53,66% |
| | Genauigkeit (3 Größen) | 99,08% | 98,15% | **98,24%** | 86,99% |
| SVM | Genauigkeit (1 Größe) | 87,18% | 66,75% | 65,39% | 22,90% |
| | Genauigkeit (3 Größen) | 98,77% | 97,28% | 97,76% | 82,27% |

Tabelle 8: Vergleich der Vorhersagegenauigkeit der Modelle

Die Tabelle zeigt, dass die Vorhersagegenauigkeit der Modelle im Allgemeinen für den Shop *9Weine* am höchsten, *VitalAbo* am zweithöchsten, *Ecco Verde* am dritthöchsten und *Piccantino* am niedrigsten ist. Wie die Abbildung 25 der Datenanalyse im Abschnitt 3.1.2 sowie die Baseline zeigen, hängt die Genauigkeit unter anderem von der Verpackungsgrößenvielfalt ab. Werden viele verschiedene Kartons verwendet, wie zum Beispiel bei *Piccantino*, ist die Genauigkeit geringer als zum Beispiel bei *Vitalabo* oder *9Weine*, wo eine Großzahl der Pakete in wenig verschiedene Größen verpackt werden. Dennoch zeigt diese Tabelle bei allen Modellen eine deutliche Steigerung der Vorhersagegenauigkeit gegenüber der Baseline. Speziell bei der Vorhersage von drei Größen liegt die Vorhersagegenauigkeit meist über 95 %. Weiters hängt die Vorhersagegenauigkeit auch mit der Produktvielfalt zusammen. Wie bereits in der Datenanalyse erwähnt, hat *Piccantino* am zweitmeisten Produkte im Sortiment, jedoch verglichen mit anderen Shops mit einer ähnlichen Anzahl an Produkten eine geringere Anzahl an Lieferungen. Zudem hat dieser Shop eine hohe Vielfalt an Produkten wie zum Beispiel Gewürze, Kochzubehör, bis hin zu Frischware. *9Weine* hat auf der anderen Seite eine relative geringe Anzahl an Produkten und verkauft fast ausschließlich Wein. Die Tabelle zeigt weiters den großen Unterschied bezüglich der Anzahl der Trainings- und Testinstanzen zwischen den Shops. *Ecco Verde* hat mit 581.048 Trainings- und 145.262 Testinstanzen fast 5 mal mehr Daten zur Verfügung als der zweitgrößte dargestellte Shop *VitalAbo*.

Aus der Tabelle geht hervor, dass der Random-Forest-Algorithmus über alle Shops hinweg die höchste Genauigkeit aufweist. Der kNN-Algorithmus und das neuronale Netz haben verglichen mit der Baseline ebenfalls eine hohe

Vorhersagegenauigkeit. Naive Bayes, Entscheidungsbäume und Support Vector Machines weisen verglichen mit den anderen Klassifikatoren eine geringere Genauigkeit auf, vor allem beim Shop *Piccantino*. Mögliche Interpretationen, warum manche Algorithmen eine höhere Vorhersagegenauigkeit als andere aufweisen, werden im Folgenden diskutiert.

**kNN-Algorithmus**
Trotz der Einfachheit des kNN-Algorithmus liegt die Genauigkeit speziell bei der Vorhersage einer Größe über alle Shops hinweg nur ein wenig hinter der des komplexeren Random-Forest-Klassifikators. Die höchste Genauigkeit wurde für k=5 Nachbarn und den euklidischen Abstand als Abstandsmetrik erreicht. Bei der Vorhersage der drei wahrscheinlichsten Größen liegt der kNN-Algorithmus bei den meisten Shops knapp hinter der Genauigkeit des Naive Bayes, dem neuronalen Netz und der Support Vector Machine. Aufgrund der überschaubaren Größe der vorliegenden Datensätze dauert die Klassifizierung nicht viel länger als zum Beispiel bei einem Modell, welches mit dem Random-Forest-Algorithmus trainiert wurde.

**Naive Bayes**
Es wurde bereits erwähnt, dass bei Naive-Bayes-Klassifikatoren die Features als voneinander unabhängig angenommen werden. Dies ist bei den Features *Gewicht der Lieferung, Anzahl der Produkte* pro Lieferung und *Approximiertes Liefervolumen* nicht der Fall, wie in der Datenanalyse in Abschnitt 3.1.2 gezeigt wurde. Deshalb weist der Naive-Bayes-Klassifikator bei der Vorhersage einer Größe in den meisten Shops die geringste Genauigkeit auf. Jedoch ist die Vorhersagegenauigkeit gegenüber der Baseline trotzdem hoch. Lediglich der Shop *Piccantino* weist eine sehr geringe Vorhersagegenauigkeit auf.

**Decision Tree und Random Forest**
Der Entscheidungsbaum-Klassifikator weist in den meisten Shops bei der Vorhersage einer Verpackungsgröße die vierthöchste Genauigkeit auf. Entscheidungsbäume sind in der Regel sehr gute Klassifikatoren. Dass die Vorhersagegenauigkeit jedoch geringer ist als zum Beispiel beim kNN-Algorithmus, kann daran liegen, dass Entscheidungsbäume zur Überanpassung neigen.

Wie in Abschnitt 2.4.4 erwähnt, können Random Forests dabei Abhilfe schaffen. Die Ergebnisse in der Tabelle zeigen, dass die mit dem Random Forest trainierten Modelle bei der Vorhersage einer Verpackungsgröße die größte Vorhersagegenauigkeit in allen Shops aufweisen. Die Vorhersagegenauigkeit bei drei Größen liegt bei fast allen Shops bis auf *Piccantino* bei über 95 %.

**Neuronales Netz**
Die Vorhersagegenauigkeit des implementierten neuronalen Netzes ist nur ein wenig geringer als die Genauigkeit des Random Forest und des kNN-Algorithmus, obwohl nur ein einfaches Modell ohne Hyperparameteroptimierung erstellt wurde. Beim Shop *Ecco Verde* liegt die Vorhersagegenauigkeit bei drei Größen sogar knapp vor jener des Random Forest. Dies lässt vermuten, dass durch eine bessere Struktur sowie einer Hyperparameteroptimierung die Vorhersagegenauigkeit weiter gesteigert und die Genauigkeit des Random Forest eventuell sogar übertroffen werden könnte. Aufgrund der geringen Erfahrung mit neuronalen Netzen und der Tatsache, dass die Rechenkapazitäten von Google Colab limitiert sind, wurde dies nicht weiter verfolgt. Es besteht jedoch Optimierungspotential für die Zukunft.

**Support Vector Machine**
Bei den Support Vector Machines (SVMs) führte das Modell mit dem linearen Kernel zur höchsten Vorhersagegenauigkeit. Generell liegt die Vorhersagegenauigkeit der SVM nur knapp über jener des Naive-Bayes-Klassifikators. Bei der Vorhersage der drei wahrscheinlichsten Größen liegt die Genauigkeit jedoch nur knapp hinter jener des Random Forests. Ähnlich wie der Naive-Bayes-Klassifikator hat die SVM eine sehr geringe Vorhersagegenauigkeit beim Shop *Piccantino*. Wie beim neuronalen Netz zuvor besteht durch Anpassen der Hyperparameter weiteres Optimierungspotential für die Vorhersage mittels der SVM.

**Analyse des Vorhersagemodells mittels Random Forest**
Beim Vergleich der Modelle in Tabelle 8 weist der Random-Forest-Klassifikator die höchste Vorhersagegenauigkeit unter allen Modellen auf. Wie Tabelle 9 zeigt, ist die Genauigkeit in allen ausgewählten Shops bei der Vorhersage einer Verpackungsgröße um mindestens 30 % höher. Bei der Vorhersage der drei wahrscheinlichsten Größen beträgt die Steigerung der Genauigkeit trotz der teilweise bereits hohen Vorhersagegenauigkeit der Baseline in jedem dargestellten Shop mindestens 15 %.

| Modell \ Shop | | 9Weine | VitalAbo | Ecco Verde | Piccantino |
|---|---|---|---|---|---|
| Baseline | Genauigkeit (1 Größe) | 35,41% | 49,25% | 32,49% | 17,36% |
| | Genauigkeit (3 Größen) | 81,76% | 83,20% | 74,21% | 47,33% |
| Random Forest | Genauigkeit (1 Größe) | 94,47% | 80,49% | 68,98% | 60,51% |
| | Genauigkeit (3 Größen) | 99,15% | 98,76% | 98,20% | 92,24% |
| Differenz gegenüber der Baseline | Genauigkeit (1 Größe) | + 59,05% | + 31,23% | + 36,49% | + 43,15% |
| | Genauigkeit (3 Größen) | + 17,38% | + 15,56% | + 23,99% | + 44,90% |

Tabelle 9: Vergleich der Vorhersagegenauigkeit des Random Forests gegenüber der Baseline, Quelle: Eigene Darstellung

Diese Tabelle zeigt, dass der Einsatz des Vorhersagemodells für die Vorhersage der drei wahrscheinlichsten Größen aufgrund einer Wahrscheinlichkeit von über 90 % in jedem der ausgewählten Shops möglich ist.

Abbildung 31 zeigt die Confusion-Matrix des Random-Forest-Klassifikators beim Shop *Piccantino*. Aus Gründen der Übersichtlichkeit sind nur die 15 häufigsten Größen dargestellt. Die Zeilen dieser Matrix beschreiben die tatsächliche Verpackungsgröße dieser Lieferung, die Spalten beschreiben die Größe, die der Klassifikator vorhergesagt hat. Die Zahlen in den Zellen beschreiben die Anzahl der falsch bzw. richtig vorhergesagten Größen.

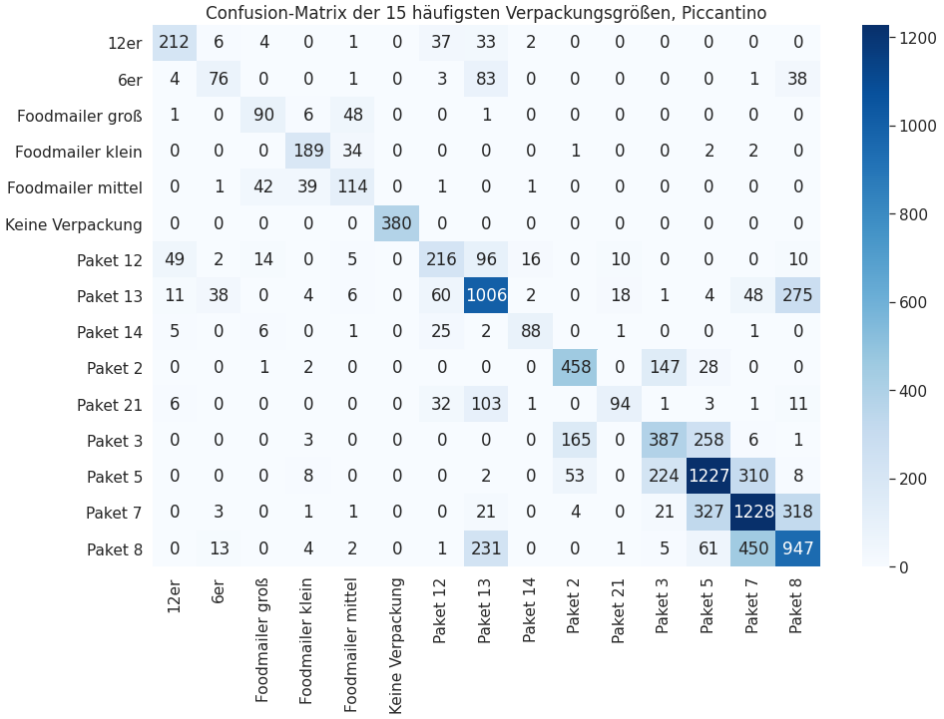


Abbildung 31: Confusion-Matrix der 15 häufigsten Größen beim Shop *Piccantino*, Quelle: Eigene Darstellung

Obwohl die Confusion-Matrix für fast alle Klassen die höchsten Werte in der Diagonale aufweist, existieren auch teilweise sehr hohe Werte in Zellen, die sich nicht auf der Diagonale befinden. Diese beschreiben falsch klassifizierte Lieferungen. Aus der Matrix geht hervor, dass Flaschenkartons (*12er, 6er*) oft mit Paketen verwechselt werden und umgekehrt. Ein zusätzliches Feature, welches angibt, ob es sich bei einem Produkt um eine Flasche handelt, könnte diesen Fehler reduzieren. Diese Daten sind im Unternehmen jedoch noch nicht verfügbar. Weiters zeigt die Matrix, dass die Features, welche angeben, dass es sich um eine Lieferung mit Frischware (*Foodmailer*)

oder um eine Selbstabholung (*Keine Verpackung*) handelt, relevant sind, da diese Verpackungen in den meisten Fällen richtig vorhergesagt werden. Jedoch fällt es dem Modell schwer, zwischen den verschiedenen Größen des Foodmailers zu unterscheiden. Speziell die mittlere Größe wird oft mit der größeren oder kleineren Größe dieser Verpackungsart verwechselt. Wie der rechte untere Bereich dieser Matrix zeigt, werden vor allem Kartons, welche ein ähnliches Volumen aufweisen, miteinander verwechselt. Dies lässt darauf schließen, warum die Genauigkeit bei der Vorhersage der drei wahrscheinlichsten Größen generell sehr hoch ist. Wie bei den Foodmailern wird eine Größe oft mit der nächstkleineren oder der nächstgrößeren Größe verwechselt. Diese Überschneidungen der Klassen sind auch in Abbildung 28 in Kapitel 3.1.2 sichtbar.

Generell gilt es zu untersuchen, ob in der Praxis wirklich alle verschiedenen Verpackungsgrößen benötigt werden. Ein größerer Unterschied der verschiedenen Größen von Kartons könnte die Vorhersagegenauigkeit im Allgemeinen erhöhen. Weiters sollte bei den Stammdaten gepflegt werden, ob es sich beim Produkt um eine Flasche handelt.

## 3.3 Einbindung des Modells in das bestehende System

Der letzte Schritt besteht darin, das trainierte Modell in das bestehende Softwaresystem einzubinden. Dieser Prozess wird auch als *Model-Deployment* bezeichnet. Im Allgemeinen gibt es verschiedene Arten, ein Machine-Learning-Modell in ein bestehendes System zu integrieren, auf welche hier jedoch nicht im Detail eingegangen wird. Stattdessen wird in diesem Abschnitt nur kurz das Model-Deployment der Modelle für die Vorhersage der Verpackungsgröße erläutert. Hierbei werden die trainierten Modelle auf der *Google Cloud AI Platform* bereitgestellt. Es wurde entschieden, dass die Vorhersage der Verpackungsgrößen gesammelt für eine Vielzahl von Lieferungen bereits kurz nach dem Erstellen dieser durchgeführt wird. Dabei wird eine Anfrage mit den Werten der zuvor definierten Features dieser Lieferungen an das Modell auf der Cloud gesendet. Das Modell verarbeitet diese Anfrage und gibt die drei wahrscheinlichsten Verpackungsgrößen für jede Lieferung zurück. Diese vorhergesagten Verpackungsgrößen werden in der Datenbank für die Lieferungen festgehalten.

Aufgrund der Tatsache, dass die Vorhersage vorab und nicht direkt beim Starten des Verpackungsprozesses durchgeführt wird, können auch Modelle eingesetzt werden, bei denen die Klassifizierung etwas länger dauert. Da der Random-Forest-Algorithmus für fast alle Shops die höchste Vorhersagegenauigkeit aufweist, wurde entschieden, ein Modell dieses Algorithmus in das Produktivsystem einzubinden. Zudem ist es aufgrund der hohen Genauigkeit dieser Modelle bei der Vorhersage von drei

Größen möglich, in jedem Shop einen relevanten Vorschlag für die drei wahrscheinlichsten Größen zu geben. Generell kann abhängig von der Vorhersagegenauigkeit für jeden Shop separat definiert werden, ob die wahrscheinlichste Verpackungsgröße im Verpackungsprozess vorausgewählt ist oder nur die drei wahrscheinlichsten Größen vorgeschlagen werden. Es wurde im ersten Schritt der Random-Forest-Klassifikator im Shop *Vitalabo* integriert, da dieser Shop verglichen mit anderen Shops eine hohe Anzahl an Lieferungen aufweist und der Klassifikator mit knapp über 80 % auch eine gute Vorhersagegenauigkeit hat, sodass die Verpackungsgröße vorausgewählt werden kann. In Abbildung 32 wird die zentrale Benutzeroberfläche des Verpackungsprozesses im Shop *Vitalabo* nach Einbindung des Vorhersagemodells dargestellt.

Beim Start des Verpackungsprozesses werden die vorgeschlagenen Verpackungsgrößen für die zu verpackende Lieferung aus der Datenbank geladen und dem Lagerarbeiter im linken oberen Bereich als eigener ausgeklappter Reiter angezeigt. Für die Lieferung in Abbildung 32 werden die Verpackungsgrößen *Paket 2, Paket 3* und *Paket 5* vorgeschlagen. *Paket 2* hat dabei die höchste Wahrscheinlichkeit und wird deshalb bereits vorausgewählt. Der Arbeiter muss nun die entsprechende Verpackungsgröße nicht mehr auswählen und kann direkt mit dem Scannen der Produkte beginnen. Der weitere Prozessablauf bleibt gleich wie im Kapitel 2.3.2 beschrieben.

Zur Zeit der Verfassung dieser Arbeit war das Modell in diesem Shop erst seit Kurzem im Einsatz. Daher können noch keine aussagekräftigen Angaben zur Vorhersagegenauigkeit gegeben werden. Es ist jedoch in Zukunft geplant, auch die Modelle anderer Shops in den Verpackungsprozess einzubinden.

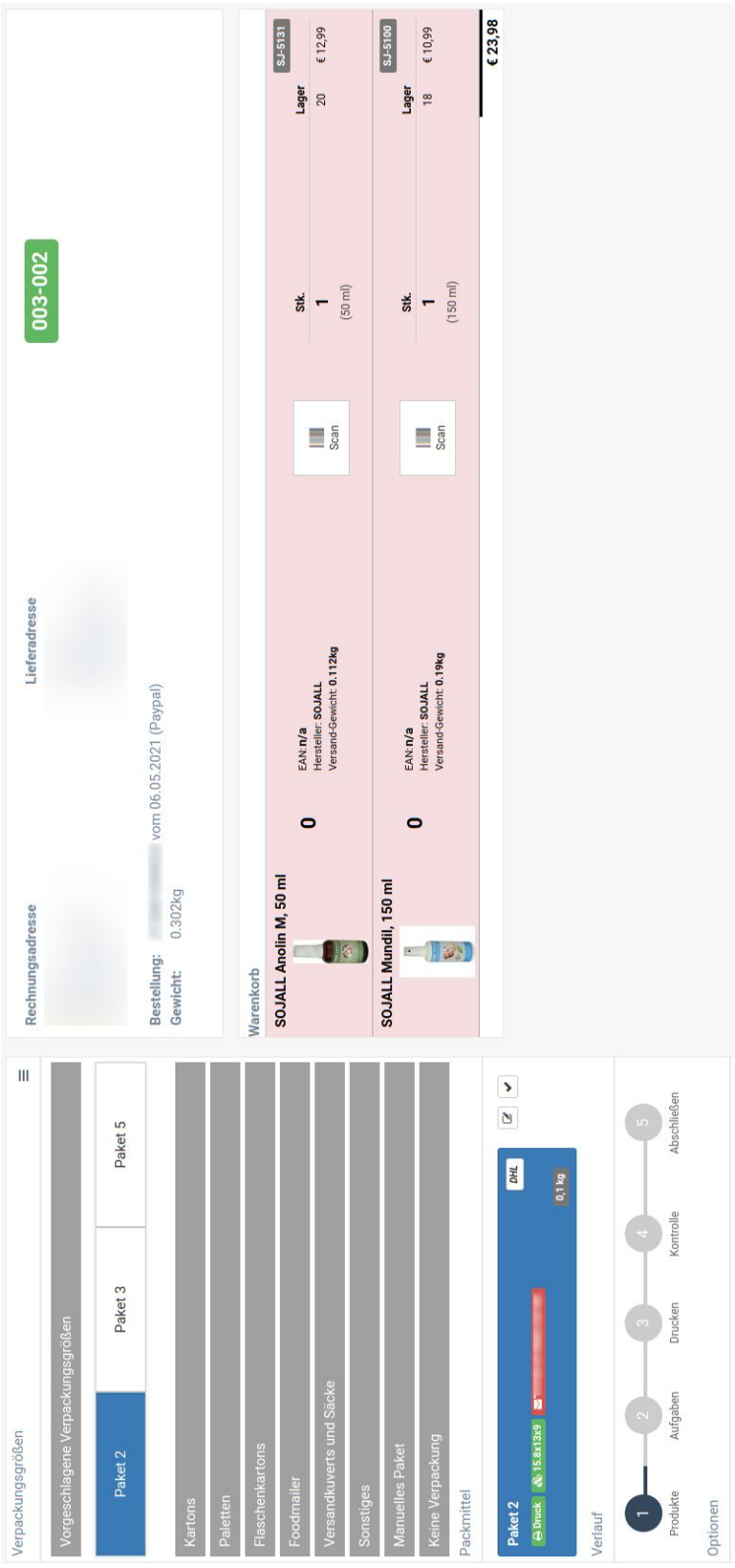


Abbildung 32: Benutzeroberfläche Verpackungsprozess nach der Implementierung des Vorhersagemodells, Quelle: niceshops GmbH

# 4 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war es zu prüfen, ob es möglich ist, mit Hilfe eines Machine-Learning-Modells die Verpackungsgrößen in einem E-Commerce-Unternehmen vorherzusagen. Dafür wurden zunächst die notwendigen Grundlagen des Machine Learnings in Hinblick auf die Entwicklung eines Vorhersagemodells erläutert und insbesondere Algorithmen identifiziert, welche zum Trainieren eines Vorhersagemodells zur Mehrklassen-Klassifikation verwendet werden können. Aufbauend auf diese theoretischen Grundlagen konnten anschließend für jeden Onlineshop des Unternehmens verschiedene Modelle trainiert und evaluiert werden. Der letzte Schritt bestand in der Integration eines dieser Modelle für einen Shop in das bestehende Softwaresystem. Es wird nun zu der in Kapitel 1 definierten Forschungsfrage inklusive Unterfragen Stellung genommen.

Zunächst erfolgte eine Einteilung der Machine-Learning-Systeme in Kategorien, um einen Überblick zu bekommen, welche dieser Systeme für die Vorhersage einer Verpackungsgröße relevant sind. Es konnte dabei der Unterschied zwischen überwachtem, unüberwachtem und verstärkendem Lernen erläutert, sowie Anwendungsbeispiele und Algorithmen für jede Kategorie genannt werden. Weiters wurden auch das Batch- und Online-Lernen sowie das instanzbasierte und modellbasierte Lernen gegenübergestellt. Die Tatsache, dass die Vorhersage einer Verpackungsgröße in den bestehenden Verpackungsprozess eingreift, bedingte eine detaillierte Beschreibung von diesem Prozess. Einer Erklärung und Abgrenzung der für den Verpackungsprozess relevanten Begriffe folgte eine Auflistung der im Unternehmen vorhandenen Verpackungstypen, welche unter anderem Kartons, Paletten oder Flaschenkartons umfasst. Weiters wurde der Prozess vor und nach der Implementierung eines Vorhersagemodells anhand von Ablaufdiagrammen dargestellt. Es konnte festgestellt werden, dass einzelne Prozessschritte wie zum Beispiel die Auswahl und Bestätigung der Verpackungsgröße durch eine Vorauswahl nicht mehr notwendig sind. Die Beschreibung der Faktoren, welche die Auswahl einer Verpackungsgröße beeinflussen, war relevant für die Entwicklung des Vorhersagemodells in Kapitel 3, da diese Faktoren Auswirkungen auf die Datenanalyse und Auswahl der Features hatten. Es konnten die tatsächlich verfügbaren Verpackungen, die Anzahl und Eigenschaften der Produkte der Lieferung, Lieferland, Lieferdienst und der Mitarbeiter, welcher den Verpackungsprozess durchführt, als Einflussfaktoren identifiziert werden. Den Abschluss dieses Kapitels bildete die Identifikation der für die Vorhersage von Verpackungsgrößen relevanten Machine-Learning-Systeme. Auf Basis der Erarbeitung der Grundlagen des Machine Learnings sowie der Beschreibung des Verpackungsprozesses konnte festgestellt werden, dass es sich um überwachtes Lernen handelt, da jeder Lieferung (Instanz) eine Verpackungsgröße (Klasse) zugewiesen

werden kann. Speziell wurde definiert, dass es sich um ein Problem der Mehrklassen-Klassifikation handelt. Diese Erkenntnis ermöglichte eine Auswahl an Algorithmen, welche dazu verwendet werden können, um Modelle zur Mehrklassen-Klassifikation zu trainieren. Die Auswahl umfasste die Algorithmen *k-nächste-Nachbarn, Naive Bayes, Entscheidungsbäume und Random Forest, Neuronale Netze* sowie *Support Vector Machines*. Neben einer überblicksmäßigen Erklärung, wie die Algorithmen funktionieren und zum Trainieren eines Modells zur Vorhersage von Verpackungsgrößen eingesetzt werden können, konnten Stärken und Schwächen, Gemeinsamkeiten sowie die Anforderungen an die Features für das Training der Algorithmen identifiziert werden. Diese Erkenntnisse waren in weiterer Folge wichtig für die Entwicklung des Vorhersagemodells in Kapitel 3, da unter anderem festgestellt werden konnte, dass der k-nächste-Nachbarn-Algorithmus ein sogenannter *lazy learning Algorithmus* ist und die Klassifikation neuer Instanzen, wenn ein großer Datensatz vorliegt, langsam ist. Aufgrund der Tatsache, dass der Naive-Bayes-Klassifikator trotz der Annahme unabhängiger Features oft auch gute Ergebnisse liefert, wenn diese Voraussetzung nicht erfüllt ist, wurde auch für diesen Algorithmus ein Modell erstellt. Dass Entscheidungsbäume zu den beliebtesten Algorithmen gehören, konnte darauf zurückgeführt werden, dass sie bereits auf viele verschiedene Aufgabenstellungen angewendet wurden und die Ergebnisse besser interpretierbar sind als für andere Algorithmen. Random Forests verringern das Risiko der Überanpassung bei Entscheidungsbaumalgorithmen, da diese die Ergebnisse einer Sammlung von Entscheidungsbäumen (Ensembles) kombinieren, um eine erhöhte Genauigkeit zu erreichen. Dabei wurde erkannt, dass die Bestimmung der Struktur neuronaler Netze viel Erfahrung bedingt, sowie das Trainieren der Algorithmen beider Methoden rechenintensiv ist.

Auf Basis der Grundlagen konnte das Vorhersagesystem entwickelt und implementiert werden. Der Prozess, ein Vorhersagesystem zu entwickeln, wurde in die Schritte *Data Engineering, Model Engineering* und *Model Deployment* eingeteilt. Im Rahmen des Data Engineerings wurde die Datengrundlage definiert, relevante Daten mit Hilfe einer explorativen Datenanalyse identifiziert und beschrieben, wie diese Daten aufbereitet werden müssen, um damit ein Machine-Learning-Modell trainieren zu können. Die Datengrundlage umfasst die Daten zu Paketen, Paketelementen und Verpackungen, wie zum Beispiel das Gewicht der Lieferung, die Anzahl der Produkte in einer Lieferung oder die Größe der Verpackung. Mit Hilfe der explorativen Datenanalyse konnte gezeigt werden, dass die Shops eine unterschiedlich hohe Anzahl an historischen Lieferungen aufweisen. Speziell führte die Tatsache, dass mehr als die Hälfte aller Lieferungen dem Shop *Ecco Verde* zugeordnet sind, zur Entscheidung, für jeden Shop ein eigenes Modell zu erstellen. Ein shopübergreifendes Modell würde zu stark

von den Daten von dem *Ecco Verde* beeinflusst werden. Die Analyse der Daten hinsichtlich der Verpackungsgrößen legte dar, dass ein Großteil der Lieferungen mit den drei am häufigsten verwendeten Verpackungsgrößen versendet werden. Die Analyse zeigte weiters, dass nur bei knapp über 5 % aller Produkte das Volumen hinterlegt ist und dieses somit nicht als Eigenschaft zur Vorhersage verwendet werden kann. Die Datenanalyse ergab jedoch, dass das Gewicht des Pakets sowie die Anzahl der Produkte im Zusammenhang mit der Verpackungsgröße stehen.

Nach der Aufbereitung der Daten mussten die relevanten Features für das Training der Modelle ausgewählt werden. Folgende Features wurden ausgewählt: das Gewicht der Lieferung, Anzahl der Produkte in der Lieferung, ein Flag, welches angibt, ob es sich um Frischversand handelt, ein weiteres Flag, welches definiert, ob es sich um eine Selbstabholung handelt, sowie ein approximiertes Volumen. Das approximierte Volumen einer Lieferung konnte aus der Summe der approximierten Volumen der Produkte in einer Lieferung bestimmt werden. Die Berechnung der approximierten Volumen der Produkte erfolgt wiederum mit Hilfe der Daten historischer Lieferungen. Diese Features ermöglichten es, Modelle für jeden Shop zu trainieren und zu evaluieren. Die Implementierung erfolgte mit Hilfe der vordefinierten Algorithmen der Python Bibliothek *Scikit-learn*. Die Vorhersagegenauigkeit der Modelle für die Shops *9Weine*, *Vitalabo*, *Ecco Verde* und *Piccantino* wurde mit jener des *ZeroR-Baseline-Algorithmus* verglichen, welcher einer neuen Lieferung immer die am häufigsten verwendete Verpackungsgröße als Klasse zuweist. Neben der Vorhersagegenauigkeit bei der Vorhersage einer Größe wurde auch die Wahrscheinlichkeit (Genauigkeit) bestimmt, dass sich die tatsächliche Größe unter den drei wahrscheinlichsten vorhergesagten Größen befindet. Die Analyse ergab, dass der Random-Forest-Algorithmus über alle Shops hinweg die höchste Vorhersagegenauigkeit bei der Vorhersage einer Größe aufweist. Die Genauigkeit des k-nächste-Nachbarn-Algorithmus lag meist nur ein wenig hinter jener des Random Forests. Allgemein zeigten die Ergebnisse, dass die Genauigkeit der Vorhersage aufgrund der unterschiedlichen Anzahl an historischen Lieferungen sowie der unterschiedlichen Verpackungs- und Produktvielfalt von Shop zu Shop variiert. Dies führt dazu, dass die Vorhersagegenauigkeit allgemein für den Shop *9Weine* am höchsten, *VitalAbo* am zweithöchsten, *Ecco Verde* am dritthöchsten und *Piccantino* am niedrigsten ist. Die Confusion Matrix zeigte, dass es den Modellen oft schwer fällt, zwischen benachbarten Verpackungsgrößen zu unterscheiden und lässt darauf schließen, warum im Allgemeinen die Genauigkeit bei der Vorhersage von drei Größen bei den meisten ausgewählten Shops über 95 % beträgt. Dies ermöglicht eine Implementierung eines Modells in jedem Shop, welches dem Lagerarbeiter die drei wahrscheinlichsten Größen vorschlägt. Ein solches mit dem Random-Forest-Algorithmus trainiertes Modell wurde

im Shop *Vitalabo* integriert. Zusammenfassend kann aufgrund der Verbesserung der Vorhersagegenauigkeit für jeden Shop gegenüber der Baseline und der erfolgreichen Implementierung eines Modells in einem Shop die Forschungsfrage beantwortet und bestätigt werden, dass es möglich ist, die Verpackungsgröße einer Lieferung mittels Machine Learning vorherzusagen.

## 4.1 Ausblick

In Zukunft könnten durch eine detailliertere Untersuchung der Daten eventuell shop-spezifische Features verwendet werden, um die Vorhersagegenauigkeit dieses Modells zu erhöhen. Zudem ist zu prüfen, ob durch das Trainieren von neuronalen Netzen mit anderen Strukturen eine höhere Genauigkeit erreicht werden kann. Die Einsatzmöglichkeit der Vorhersagesysteme ist nicht nur auf den Verpackungsprozess beschränkt. Ist vorab bekannt, welche Lieferungen am nächsten Tag das Lager verlassen, kann durch die Vorhersage einer Größe für jede Lieferung das gesamte Transportvolumen pro Lieferdienst abgeschätzt werden. Dadurch könnten die Transporte in Zukunft optimiert werden, da bestimmt werden kann, wie viele Fahrzeuge von welchem Lieferdienst benötigt werden. Durch die Möglichkeit, dem Lagerarbeiter in jedem Shop eine Teilmenge an Verpackungsgrößen zur Auswahl geben zu können, können durch die damit verbundene Zeitersparnis die ausgewählten Verpackungsgrößen auch in Monaten mit sehr hohem Bestellaufkommen, wie zum Beispiel vor Weihnachten, aufgezeichnet werden. Da bekannt ist, welcher Shop wie viel Stück von welcher Verpackung benötigt, wäre es weiters möglich, auf Basis dieser Vorhersagen den Bestand an Verpackungen zu optimieren. Im Unternehmen werden nun die Modelle sukzessive für alle weiteren Shops in den Verpackungsprozess integriert, sowie weitere Einsatzmöglichkeiten dieser Vorhersagemodelle geprüft.

# Literaturverzeichnis


Aditya (2018). *What does baseline mean in the context of machine learning?* URL: https://datascience.stackexchange.com/questions/30912/what-does-baseline-mean-in-the-context-of-machine-learning (Abruf am 24. 04. 2021).

Aly, Mohamed (2005). *Survey on multiclass classification methods.* Techn. Ber.

Anonym (2018). *Hyperparameter Tuning: Der Feinschliff eines Machine-Learning-Modells - News - TWT Interactive.* URL: https://www.twt.de/news/detail/hyperparameter-tuning-der-feinschliff-eines-machine-learning-modells.html (Abruf am 22. 04. 2021).

Arat, Mustafa Murat (2019). *Can you interpret probabilistically the output of a Support Vector Machine?* URL: https://mmuratarat.github.io/2019-10-12/probabilistic-output-of-svm (Abruf am 18. 04. 2021).

Awad, Mariette und Rahul Khanna (2015). „Support Vector Machines for Classification". In: *Efficient Learning Machines*, S. 39–66. ISBN: 978-1-4302-5989-3. DOI: 10.1007/978-1-4302-5990-9_3.

Behrend, Claudia (2019). *E-Commerce-Verpackung: Großer Karton, wenig Inhalt.* URL: https://www.dvz.de/rubriken/management-recht/detail/news/e-commerce-verpackung-grosser-karton-wenig-inhalt.html (Abruf am 12. 03. 2021).

Bennett, Kristin P. und Colin Campbell (2000). „Support Vector Machines: Hype or Hallelujah?" In: *SIGKDD Explor. Newsl.* 2.2, 1–13. ISSN: 1931-0145. DOI: 10.1145/380995.380999.

Bhatia, Nitin und Vandana (2010). „Survey of Nearest Neighbor Techniques". In: *International Journal of Computer Science and Information Security* 8.

Bishop, Christopher (2006). *Pattern Recognition and Machine Learning.* Springer. ISBN: 978-0-387-31073-2.

Boehmke, Bradley (2016). *Data Wrangling with R.* ISBN: 978-3-319-45598-3. DOI: 10.1007/978-3-319-45599-0.

Brownlee, Jason (2016). *Master Machine Learning Algorithms.*

— (2019a). *Different Types of Learning in Machine Learning.* URL: https://machinelearningmastery.com/types-of-learning-in-machine-learning/ (Abruf am 15. 02. 2020).

— (2019b). *Overfitting and Underfitting With Machine Learning Algorithms.* URL: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/ (Abruf am 17. 04. 2021).

— (2019c). *What is a Hypothesis in Machine Learning?* URL: https://machinelearningmastery.com/what-is-a-hypothesis-in-machine-learning/ (Abruf am 17. 04. 2021).

Brownlee, Jason (2020). *Naive Bayes for Machine Learning*. URL: https://machinelearningmastery.com/naive-bayes-for-machine-learning/ (Abruf am 30. 04. 2021).

Ciaburro, Giuseppe und Balaji Venkateswaran (2017). „Pros and Cons of Neural Networks“. In: *Neural Networks with R*. URL: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788397872/1/ch01lvl1sec27/pros-and-cons-of-neural-networks (Abruf am 11. 04. 2021).

Cortes, Corinna und Vladimir Vapnik (1995). „Support-Vector Networks“. In: *Machine Learning* 20.3, S. 273–297. ISSN: 15730565. DOI: 10.1023/A:1022627411411.

Cover, T. M. und P. E. Hart (1967). „Nearest Neighbor Pattern Classification“. In: *IEEE Transactions on Information Theory* 13.1, S. 21–27. ISSN: 15579654. DOI: 10.1109/TIT.1967.1053964.

Döbel, Inga, Miriam Leis, Manuel Vogelsang, Dmitry Neustroev, Henning Petzka, Angelika Voss, Martin Wegele und Juliane Welz (2018). *Maschinelles Lernen. Eine Analyse zu Kompetenzen, Forschung und Anwendungf*.

Famili, A., Wei Min Shen, Richard Weber und Evangelos Simoudis (1997). „Data preprocessing and intelligent data analysis“. In: *Intelligent Data Analysis* 1.1, S. 3–23. ISSN: 15714128. DOI: 10.3233/IDA-1997-1102.

Frost, Jim (o. D.). *5 Ways to Find Outliers in Your Data*. URL: https://statisticsbyjim.com/basics/outliers/ (Abruf am 02. 04. 2021).

Gandhi, Rohith (2018). *Support Vector Machine — Introduction to Machine Learning Algorithms*. URL: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (Abruf am 10. 04. 2021).

Géron, Aurélien (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2nd ed Edition. O'Reilly UK Ltd. ISBN: 978-1492032649.

Gershenson, Carlos (2003). „Artificial Neural Networks for Beginners“.

Gleißner, Harald und Klaus Möller (2009). *Fallstudien Logistik*. Gabler. DOI: 10.1007/978-3-8349-8365-7.

Goodfellow, Ian J., Yoshua Bengio und Aaron Courville (2016). *Deep Learning*. MIT Press.

Grandini, Margherita, Enrico Bagli und Giorgio Visani (2020). *Metrics for multi-class classification: An overview*. Techn. Ber. eprint: 2008.05756.

Große Holtforth, Dominik (2018). *Machine Learning im E-commerce*. URL: https://ecommerceinstitut.de/machine-learning-im-e-commerce/ (Abruf am 15. 05. 2021).

Grus, Joel (2019). *Data Science from Scratch*. O'Reilly Media, Inc. ISBN: 9781492041139.

Görz, Günther und Josef Schneeberger (2010). *Handbuch der Künstlichen Intelligenz*. Oldenbourg Wissenschaftsverlag. DOI: doi : 10 . 1524 / 9783486598834.

Hancock, Thomas, Tao Jiang, Ming Li und John Tromp (1996). „Lower Bounds on Learning Decision Lists and Trees“. In: *Information and Computation* 126.2, S. 114–122. ISSN: 08905401. DOI: 10.1006/inco.1996.0040.

Harrison Jr, David und Daniel L Rubinfeld (1978). *The Boston Housing Dataset*. URL: https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html (Abruf am 12. 03. 2021).

Ilyas, Ihab und Xu Chu (2019). *Data Cleaning*. Association for Computing Machinery. DOI: 10.1145/3310205.

Iqbal, Muhammad und Zhu Yan (2015). „SUPERVISED MACHINE LEARNING APPROACHES: A SURVEY“. In: *International Journal of Soft Computing* 5, S. 946–952. DOI: 10.21917/ijsc.2015.0133.

Jain, Anil K., Jianchang Mao und K. M. Mohiuddin (1996). „Artificial neural networks: A tutorial“. In: *Computer* 29.3, S. 31–44. ISSN: 00189162. DOI: 10.1109/2.485891.

Jiang, Liangxiao, Zhihua Cai, Dianhong Wang und Siwei Jiang (2007). „Survey of improving K-nearest-neighbor for classification“. In: *Proceedings - Fourth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2007*. Bd. 1, S. 679–683. ISBN: 0769528740. DOI: 10.1109/FSKD.2007.552.

Jordan, Jeremy (2017). *Evaluating a machine learning model*. URL: https://www.jeremyjordan.me/evaluating-a-machine-learning-model/ (Abruf am 23. 04. 2021).

Kingsford, Carl und Steven L. Salzberg (2008). *What are decision trees?* DOI: 10.1038/nbt0908-1011.

Knoll, Dino, Daniel Neumeier, Marco Prüglmeier und Gunther Reinhart (2019). „An automated packaging planning approach using machine learning“. In: *Procedia CIRP*. Bd. 81. Elsevier B.V., S. 576–581. DOI: 10.1016/j.procir.2019.03.158.

Kotsiantis, Sotiris (2007). *Supervised Machine Learning: A Review of Classification Techniques*. Techn. Ber., S. 249–268.

Kotsiantis, Sotiris, Dimitris Kanellopoulos und P. Pintelas (2006). „Data Preprocessing for Supervised Learning“. In: *International Journal of Computer Science* 1, S. 111–117.

LetMeShip (2021). *Versandlexikon*. URL: https://www.letmeship.com/de-at/versandlexikon/ (Abruf am 12. 03. 2021).

Manhart Klaus (2020). *Was Sie über Maschinelles Lernen wissen müssen*. URL: https://www.computerwoche.de/a/was-sie-ueber-maschinelles-lernen-wissen-muessen,3329560 (Abruf am 14. 05. 2021).

Masse, Mark (2011). *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. O'Reilly Media. ISBN: 978-1449310509.

Mijwil, Maad M. (2018). *Artificial Neural Networks Advantages and Disadvantages*.

Mitchell, Tom M. (1997). *Machine Learning*. New York: McGraw-Hill. ISBN: 978-0-07-042807-2.

Msg systems ag - Applied Technology Research., Hrsg. (o. D.). *Zero Rule*. URL: https://machinelearningcatalogue.com/algorithm/alg_zero-rule.html?s=zero (Abruf am 14. 05. 2021).

Murphy, Kevin P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press. ISBN: 978-0262018029.

Musaoglu, Erhan (2017). *Warehouse picking and packing processes: how much labor do you need?* URL: https://www.explorewms.com/warehouse-labor-for-picking-and-packing.html (Abruf am 13. 03. 2021).

Niceshops GmbH, Hrsg. (o. D.). *Das Unternehmen*. URL: https://www.niceshops.com/de/unternehmen (Abruf am 15. 01. 2021).

Noble, William Stafford (2006). „What is a support vector machine?" In: *Nature Biotechnology* 24, S. 1565–1567.

Opidi, Alfrick (2019). *Solving Data Challenges In Machine Learning With Automated Tools*. URL: https://www.topbots.com/data-preparation-for-machine-learning/ (Abruf am 18. 03. 2021).

Pedregosa, Fabian et al. (2012). „Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.

Roh, Yuji, Geon Heo und Steven Euijong Whang (2021). „A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective". In: *IEEE Transactions on Knowledge and Data Engineering* 33.4, S. 1328–1347. DOI: 10.1109/TKDE.2019.2946162.

Rokach, Lior und Oded Maimon (2006). „Decision Trees". In: *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag, S. 165–192. DOI: 10.1007/0-387-25465-x_9.

Rosenblatt, Frank (1958). „The perceptron: A probabilistic model for information storage and organization in the brain". In: *Psychological Review* 65.6, S. 386–408. ISSN: 0033295X. DOI: 10.1037/h0042519.

Rumelhart, D., G. Hinton und James Mcclelland (1986). „A General Framework for Parallel Distributed Processing". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* 1.

Samuel, A. (1959). „Some Studies in Machine Learning Using the Game of Checkers". In: *IBM J. Res. Dev.* 3, S. 210–229.

Schäfer, Thomas (2010). „Explorative Datenanalyse: Muster und Zusammenhänge erkennen". In: *Statistik I*. VS Verlag für Sozialwissenschaften, S. 99–128. DOI: 10.1007/978-3-531-92446-5_4.

Scikit-learn developers (BSD License), Hrsg. (o. D.). *1.9. Naive Bayes — scikit-learn 0.24.1 documentation*. URL: https://scikit-learn.org/

stable/modules/naive_bayes.html#categorical-naive-bayes (Abruf am 18. 04. 2021).

Shaikh, Raheel (2018). *The ABC of Machine Learning*. URL: https://towardsdatascience.com/the-abc-of-machine-learning-ea85685489ef (Abruf am 17. 04. 2021).

Singh, Karanjit und Shuchita Upadhyaya (2012). „Outlier Detection: Applications And Techniques". In: *International Journal of Computer Science Issues* 9.

Singh, Pramod (2021). *Deploy Machine Learning Models to Production*. Apress. ISBN: 978-1-4842-6546-8. DOI: 10.1007/978-1-4842-6546-8.

Stuart J. Russell and Peter Norvig (2016). *Artificial Intelligence: A Modern Approach, Global Edition*. ISBN: 9781292153964.

Tiwari, Tanya, Tanuj Tiwari und Sanjay Tiwari (2018). „How Artificial Intelligence, Machine Learning and Deep Learning are Radically Different?" In: *International Journal of Advanced Research in Computer Science and Software Engineering* 8. DOI: 10.23956/ijarcsse.v8i2.569.

Tomar, Laxman (2020). *Model-based vs Instance-based Learning*. URL: https://www.kaggle.com/getting-started/179177 (Abruf am 14. 02. 2021).

Tsoumakas, Grigorios und Ioannis Katakis (2007). „Multi-Label Classification: An Overview". In: *International Journal of Data Warehousing and Mining* 3.3, S. 1–13. DOI: 10.4018/jdwm.2007070101.

Utgoff, Paul E. et al. (2011b). „Instance Space". In: *Encyclopedia of Machine Learning*. Springer US, S. 549–549. DOI: 10.1007/978-0-387-30164-8_408.

— (2011a). „Instance". In: *Encyclopedia of Machine Learning*. Springer US, S. 549–549. DOI: 10.1007/978-0-387-30164-8_406.

Visengeriyeva, Larysa, Anja Kammer, Isabel Bär, Alexander Kniesz und Michael Plöd (2021). *An Overview of the End-to-End Machine Learning Workflow*. URL: https://ml-ops.org/content/end-to-end-ml-workflow (Abruf am 18. 03. 2021).

Visvikis, Dimitris, Catherine Cheze Le Rest, Vincent Jaouen und Mathieu Hatt (2019). „Artificial intelligence, machine (deep) learning and radio(geno)mics: definitions and nuclear medicine imaging applications". In: *European Journal of Nuclear Medicine and Molecular Imaging* 46.13, S. 2630–2637. ISSN: 16197089. DOI: 10.1007/s00259-019-04373-w.

Wäscher, Gerhard, Heike Haußner und Holger Schumann (2007). „An improved typology of cutting and packing problems". In: *European Journal of Operational Research* 183.3, S. 1109–1130. ISSN: 03772217. DOI: 10.1016/j.ejor.2005.12.047.

Wehle, Hans-Dieter (2017). *Machine Learning, Deep Learning, and AI: What's the Difference?*

Zheng, Alice und Amanda Casari (2018). *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. 1st. O'Reilly Media, Inc. ISBN: 1491953241.