FernUniversität in Hagen

– Fakultät für Mathematik und Informatik –

Dynamic Pricing in der Luftfahrt: Analyse von Machine-Learning-Algorithmen zur Vorhersage von Flugpreisen

Abschlussarbeit zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

vorgelegt von

Dominik Lewin

Matrikelnummer: 3274837

Erstgutachter	:	Dr. Fabio Valdés
Zweitgutachterin	:	Prof. Dr. Uta Störl

Hinweis:
Zur besseren Lesbarkeit wird in der gesamten Abschlussarbeit das generische Maskulinum verwendet. Die verwendeten Personenbezeichnungen beziehen sich demnach auf alle Geschlechter, soweit nichts anderes erwähnt wird.

ABSTRACT

The aim of this bachelor's thesis was to determine how machine learning can be used to achieve cost savings in flight bookings. This is motivated by the constant price differences of flights caused by dynamic pricing, which can change multiple times a day.

The bachelor thesis gives an overview of seven selected machine learning algorithms, whose functionalities are presented in detail and applied to an example. For this purpose, a data set consisting of 237 flights with a total of 47,233 recorded airfares was obtained in advance.

During the training and optimization of the models, two problem types were addressed: firstly, the regression problem, which aims to predict an exact flight price, and secondly, the classification problem, which provides a booking recommendation (book / do not book). The best model was then used to simulate flight bookings and calculate cost savings.

It was found that the machine learning models for the regression problem were not suitable for predicting exact flight prices. However, good results were achieved for the classification problem. The Random Forest model performed the best, with an accuracy of 91.66 % on the test data. In 70.10 % of the test bookings, immediate booking was recommended by the model. In 27.10 % of the cases, savings were achieved through the model. The average saving was 23.99 %.

ZUSAMMENFASSUNG

Ziel der vorliegenden Bachelorarbeit war es, herauszufinden, wie sich Machine Learning einsetzen lässt, um Kosteneinsparungen bei Flugbuchungen zu erzielen. Grund hierfür sind die durch das Dynamic Pricing bedingten ständigen Preisdifferenzen der Flüge, die sich teilweise mehrfach täglich ändern.

Die Bachelorarbeit gibt einen Überblick über sieben ausgewählte Machine-Learning-Algorithmen, deren Funktionsweisen dabei ausführlich dargestellt und schließlich an einem Beispiel angewendet werden. Hierfür wurde zuvor ein Datensatz beschafft, der aus 237 Flügen mit insgesamt 47.233 aufgezeichneten Flugpreisen besteht.

Beim Trainieren und Optimieren der Modelle wurden zwei Problemarten behandelt: Zum einen das Regressionsproblem, bei dem ein genauer Flugpreis vorhersagt werden soll und zum anderen das Klassifikationsproblem, bei dem lediglich eine Buchungsempfehlung gegeben wird (buchen / nicht buchen). Das beste Modell wurde anschließend genutzt, um Flugbuchungen zu simulieren und die Kostenersparnis zu berechnen.

Dabei konnte festgestellt werden, dass sich die Machine-Learning-Modelle des Regressionsproblems nicht für die Vorhersage exakter Flugpreise einsetzen lassen. Für das Klassifikationsproblem hingegen konnten gute Ergebnisse erzielt werden. Die besten Ergebnisse erzielte der Random Forest mit einer Genauigkeit von 91,66 % auf den Testdaten.

Bei der Simulation wurde in 70,10 % der Test-Buchungen die sofortige Buchung empfohlen. In 27,10 % konnten durch das Modell Einsparungen in Höhe von durchschnittlich 23,99 % erzielt werden.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

1 EINLEITUNG

In der Luftfahrtbranche sind stark schwankende Flugpreise üblich, wodurch es vorkommen kann, dass für denselben Flug mit gleichen Leistungen unterschiedliche Preise angeboten werden. Grund hierfür ist das seit 1978 von Airlines eingesetzte *Revenue Management*. Dieses bezeichnet ein wirtschaftliches Konzept zur Gewinnmaximierung unter anderem mittels geeigneter Preisstrategien. Eine solche Preisstrategie ist das *Dynamic Pricing*, bei dem Flugpreise nicht für den gesamten Verkaufszeitraum festgelegt werden, sondern jederzeit dynamisch angepasst werden können. Ziele des Dynamic Pricing sind beispielsweise die Ausnutzung der verändernden Preissensibilität potenzieller Kunden oder die Möglichkeit, auf Nachfrageschwankungen reagieren zu können.

Aufgrund der Komplexität der Preisgestaltungsmodelle der Airlines ist es für Kunden schwierig zu entscheiden, wann ein Flug gebucht werden sollte oder nicht, um zu einen möglichst günstigen Preis zu buchen. Weit verbreitet ist der Grundsatz:

> „Tickets bought in advance are cheaper“ [Abd+21, S. 376].

Diesem Grundsatz kann mittlerweile allerdings nicht mehr zwingend zugestimmt werden [GG11; Abd+21; Jos+20]. Daher stellt sich die Frage, wie Kunden bei der Buchung unterstützt werden können, um einen finanziellen Vorteil zu erhalten.

Eine vielversprechende Lösung stellt der Einsatz von *Machine Learning* dar. Durch den Einsatz geeigneter *Machine-Learning-Algorithmen* können komplexe Muster in historischen Preisdaten erkannt werden, um mit Hilfe dieser Muster Vorhersagen für kommende Flüge zu treffen.

Ziel der Arbeit ist es, herauszufinden, wie Machine Learning eingesetzt werden kann, um Kunden bei der Buchung eines günstigen Flugpreises zu unterstützen. Hierfür befasst sich die Arbeit mit der Frage, welche Machine-Learning-Algorithmen sich am besten zur Vorhersage eignen und wie hoch die Einsparungen durch deren Einsatz sind. Dabei liegt der Fokus auf ausgewählten Algorithmen, mit denen in der Literatur bereits gute Ergebnisse erzielt werden konnten.

Zusätzlich wird untersucht, ob es vorteilhaft ist, den genauen Flugpreis vorherzusagen oder ob eine Buchungsempfehlung (buchen / nicht buchen) vielversprechender ist.

Nachfolgend soll ein Überblick über ähnliche Arbeiten auf dem Gebiet der Flugpreisvorhersage sowie die Methodik der vorliegenden Arbeit gegeben werden.

1.1 STAND DER WISSENSCHAFT

In der Literatur sind bereits einige Artikel zu finden, bei denen Machine Learning eingesetzt wurde, um zum einen den genauen Flugpreis vorherzusagen und zum anderen eine Buchungsempfehlung (buchen / nicht buchen) abzugeben.

Zur Vorhersage genauer Flugpreise verwendeten Ren, Yang und Yuan unter anderem die lineare Regression, bei der sich eine Vorhersagegenauigkeit von 70,06 % erzielen ließ [RYY15]. Papadakis konnte mit der Support Vector Machine eine Genauigkeit von 69,40 % erzielen [Pap14]. Auch die Algorithmen k-Nearest Neighbor [PEK22] und Multilayer Perceptron [Tzi+17] wurden bereits erfolgreich verwendet.

In [Sub+22] werden unterschiedliche Algorithmen auf unterschiedliche Datensätze angewendet. Es ist ersichtlich, dass die Ergebnisse stark vom verwendeten Datensatz abhängen. Für die lineare Regression konnte beispielsweise nur eine Genauigkeit von 35,15 % erzielt werden, während der Random Forest für denselben Datensatz 84,45 % erreichen konnte.

Nach Chen u. a. kann mit einer binären Klassifikation (buchen / nicht buchen) ein besseres Ergebnis erzielt werden als mit der Vorhersage eines exakten Preises [Che+15]. Dies hat unter anderem den Grund, dass das Preisverhalten einzelner Flüge nicht statisch ist und jederzeit von den Airlines situationsabhängig angepasst werden kann. Selbst die Preisgestaltung innerhalb derselben Airline kann von Flug zu Flug variieren, was das Finden eindeutiger Muster stark erschwert [Che+15].

1.2 METHODIK

Die vorliegende Arbeit setzt sich aus einem theoretischen Teil (Kapitel 2 und Kapitel 3) und einem praktischen Teil (Kapitel 4) zusammen.

Der theoretische Teil basiert auf einer umfangreichen Literaturrecherche und soll zunächst in die Themen Revenue Management, Dynamic Pricing und Machine Learning einführen. Zudem werden gängige Abstandsmetriken und ein aktuelles Vorgehensmodell für Machine-Learning-Projekte erläutert. Nach dieser Einführung folgt eine ausführliche Darstellung ausgewählter Machine-Learning-Algorithmen.

Im praktischen Teil werden die zuvor erläuterten Machine-Learning-Algorithmen an einem ausführlichen Beispiel angewendet. Vorab wurden bereits Flugpreise verschiedener Airlines für einen Zeitraum von vier Wochen gesammelt. Diese Daten werden genutzt, um die vorgestellten Algorithmen zu trainieren. Die Modelle werden anschließend getestet und durch Anpassung verschiedener Parameter optimiert, um das Vorhersageergebnis zu verbessern. Der praktische Teil wird in zwei separaten *Jupyter Notebooks*[1] ([Lew23a] und [Lew23b]) umgesetzt. In der vorliegenden Abschlussarbeit wird das Projekt lediglich zusammenfassend dargestellt.

1 Hierbei handelt es sich um eine webbasierte Entwicklungsumgebung.

2 GRUNDLAGEN

In diesem Kapitel werden die Grundlagen dargestellt, auf die im weiteren Verlauf dieser Arbeit aufgebaut werden soll.

Zunächst erfolgt eine Einführung in die Themen Revenue Management und Dynamic Pricing. Anschließend folgt eine ausführliche Einführung in das Thema Machine Learning, in der allgemein dargestellt wird, worum es sich beim Machine Learning handelt, was dieses ausmacht und welche Einsatzgebiete und Arten unterschieden werden können. Hierauf folgt eine kurze Darstellung von Abstandsmetriken, die für einige Machine-Learning-Algorithmen relevant sind.

Zum Ende dieses Kapitels wird ein Vorgehensmodell für Machine-Learning-Projekte vorgestellt, anhand dessen der praktische Teil dieser Arbeit durchgeführt wird.

2.1 EINFÜHRUNG IN DAS REVENUE MANAGEMENT

2.1.1 *Definition*

Das Revenue Management, zu deutsch *Ertragsmanagement*, bezeichnet ein wirtschaftliches Konzept zur Gewinnmaximierung, das unter anderem von Airlines verwendet wird und seinen Ursprung in der Luftfahrtindustrie findet [HT14]. Eine Gewinnmaximierung soll durch eine geeignete Steuerung von Verfügbarkeiten (Kapazitätssteuerung) einer Dienstleistung oder eines Produktes und der Preisgestaltung (Preissteuerung) erzielt werden. Ziel dieser Steuerungselemente ist es, die richtige Dienstleistung oder das richtige Produkt zum richtigen Zeitpunkt und zum richtigen Preis an den richtigen Kunden zu verkaufen [Cro98], indem Prognosen über das künftige Kaufverhalten von Kunden getroffen werden.

In der Literatur sind unterschiedliche Definitionen des Revenue Managements zu finden, weshalb im Folgenden eine häufig zitierte Definition wiedergegeben wird. Klein definiert das Revenue Management als

> „[...] eine Reihe von quantitativen Methoden zur Entscheidung über Annahme oder Ablehnung unsicherer, zeitlich verteilt eintreffender Nachfrage unterschiedlicher Wertigkeit. Dabei wird das Ziel verfolgt, die in einem begrenzten Zeitraum verfügbare, unflexible Kapazität möglichst effizient zu nutzen“ [Kle01, S. 248].

Bezogen auf die Luftfahrt bedeutet dies, dass sich eine Airline aufgrund schwankender Buchungsnachfrage und begrenzter Sitzplatzkapazitäten immer entscheiden muss, welchen Personen sie einen Flug zu welchem Zeit-

punkt und zu welchem Preis anbieten will, sodass keine Kapazitäten ungenutzt bleiben, der Erlös aber dennoch maximiert wird.

2.1.2 *Entstehung*

Nach Klein und Steinhardt begannen die ersten Ansätze des Revenue Managements mit der Deregulierung[1] des amerikanischen kommerziellen Luftverkehrs im Jahr 1978 [KS08]. Durch die Deregulierung war es den Airlines erstmals möglich, selbst über die Preisgestaltung und angebotene Flugrouten zu entscheiden. Durch die neu gewonnenen Freiheiten bekamen etablierte Airlines Konkurrenz durch günstigere Anbieter. Diese verzichteten teilweise auf Serviceleistungen und konnten auf diese Weise günstigere Preise anbieten.

Es entstand ein Preisdruck zwischen den Airlines, weshalb American Airlines die Preisdifferenzierung (vgl. Unterabschnitt 2.1.4) einführte, bei der für ein und denselben Flug unterschiedliche Preise verlangt wurden. Um schlecht ausgelastete Flüge füllen zu können, bot American Airlines Spezialtarife an.

Im Gegensatz zu Klein und Steinhardt datieren McGill und Van Ryzin die ersten Anzeichen des Revenue Managements auf die Anfänge der 1970er Jahre zurück [MVR99]. In diesen Jahren führte die Airline BOAC (British Overseas Airways Corporation; heutige British Airways) Frühbucherrabatte ein, bei denen ein Rabatt gewährt wurde, wenn mindestens 21 Tage vor Abflug gebucht wurde.

Durch das Revenue Management war es den Airlines möglich, Mehrerlöse zu erzielen. American Airlines schaffte es durch die Preisdifferenzierung, das Passagieraufkommen um ca. 15 % zu steigern. Auch die deutsche Airline Lufthansa konnte durch die Anwendung des Revenue Managements einen Mehrerlös von 1,4 Mrd. DM erzielen [KS08].

2.1.3 *Anwendungsmerkmale*

Das Revenue Management eignet sich besonders für Unternehmen und Branchen, auf die gewisse Merkmale zutreffen. Klein und Steinhardt nennen in [KS08] die folgenden Anwendungsvoraussetzungen. Diese werden zur besseren Verständlichkeit anhand mehrerer Beispiele auf die Luftfahrtbranche übertragen.

- *Weitgehend fixe Kapazitäten*

  Beispiel 2.1. Eine Airline bietet einen Flug von Hamburg nach London mit einem Airbus A320 an und kann insgesamt 180 Sitzplätze für diesen Flug anbieten. Diese Kapazitäten sind im Allgemeinen fix und könnten lediglich durch einen Wechsel des Flugzeugtyps verringert

1 Abbau staatlicher Vorschriften

oder gesteigert werden. In der Praxis ist ein Flugzeugwechsel allerdings mit weiteren Problemen verbunden und wird daher nur in Ausnahmefällen vorgenommen.

- *Verderblichkeit von Kapazitäten bei nicht erfolgter Nutzung*

  Beispiel 2.2. Nachdem der Flug aus Beispiel 2.1 durchgeführt wurde, kann die Airline die Sitzplätze für genau diesen (vergangenen) Flug nicht erneut anbieten.

- *Hohe Fixkosten für die Kapazitätsbereitstellung mit vergleichsweise geringen Grenzkosten[2] für die Leistungserstellung*

  Beispiel 2.3. Für die Durchführung des in Beispiel 2.1 genannten Fluges entstehen der Airline hohe Fixkosten (Kosten für Kerosin, Flugzeugwartung und Personal, Flughafengebühren etc.). Diese Kosten fallen unabhängig davon an, wie viele Personen den Flug tatsächlich buchen. Würde die Airline anstelle der 180 verfügbaren Sitzplätze nur 150 Sitzplätze verkaufen wollen, wären die daraus folgenden Kosteneinsparungen verhältnismäßig gering, da beispielsweise die Kosten für Personal sowie Flughafengebühren in gleicher Höhe anfallen. Die Grenzkosten sind dementsprechend gering.

- *Starke Nachfrageschwankungen*

  Beispiel 2.4. Bei Flugreisen bestehen hohe Schwankungen der Nachfrage. Während der Schulferien ist die Nachfrage nach Flügen in Urlaubsgebiete voraussichtlich höher als außerhalb der Schulferien.

- *Möglichkeit der Vorausbuchung*

  Beispiel 2.5. Flugtickets können meist weit im Voraus, etwa ein Jahr vor Abflugdatum, gebucht werden.

- *Möglichkeit der segmentorientierten Preisdifferenzierung*

  Beispiel 2.6. Kunden können kategorisiert werden, beispielsweise in Geschäfts- und Urlaubsreisende. Auf diese Weise können verschiedenen Gruppen verschiedene Preise angeboten werden.

Neben der Luftfahrt bestehen weitere Anwendungsfelder des Revenue Managements. Beispielsweise wird dieses auch von Hotels, Autovermietungen, Reiseveranstaltern oder Kreuzfahrtanbietern angewendet. Auch in diesen Branchen treffen o.g. Anforderungen meist zu.

2 Kosten, die durch die Bereitstellung weiterer Kapazitäten entstehen

2.1.4 *Instrumente des Revenue Managements*

Es gibt verschiedene Instrumente des Revenue Managements, die Unternehmen dabei unterstützen können, ihren Erlös zu maximieren. Nachfolgend werden drei dieser Instrumente dargestellt, wobei die Kapazitätssteuerung nach [KS08] das Kernelement des Revenue Managements bildet.

Preisdifferenzierung

Ein Unternehmen bietet in ihrer Kernleistung identische Produkte bzw. Dienstleistungen an, die zu unterschiedlichen Preisen an unterschiedliche Kundengruppen verkauft werden[3]. Die Preisdifferenzierung kann durch Marktsegmentierung erfolgen, indem Kunden in verschiedene Segmente eingeordnet werden. Häufig erfolgt eine Unterscheidung zwischen Geschäfts- und Freizeitreisenden, wodurch sich die unterschiedlichen Zahlungsbereitschaften ausnutzen lassen. Zudem können je nach Strecke, Aufenthaltsdauer, Anzahl der Flüge (One-Way Flug vs. Hin- und Rückflug) etc. unterschiedliche Preiskategorien aufgerufen werden.

Diese unterschiedlichen Preiskategorien spiegeln sich bei den Airlines in den Buchungsklassen wider. Airlines legen für die verschiedenen Beförderungsklassen (z.B. First Class, Business Class, Economy Class) jeweils mehrere Buchungsklassen mit fixen Preisen fest.

Im Luftverkehrsmarkt kann innerhalb der Preisdifferenzierung eine Abgrenzung der Teilmärkte erfolgen [Pom07; FR07]. Die Arten der Preisdifferenzierung können nach

- räumlichen (Abreiseflughafen, Zielflughafen),
- zeitlichen (Aufenthaltsdauer, Buchungszeitpunkt, Abflugzeitraum),
- mengenbezogenen (Gruppentarife, Vielfliegerrabatte, Rückflugermäßigung),
- personenbezogenen (Studenten- oder Seniorentarife) und
- sachlichen (spezielle Tarife für Urlaubs- oder Geschäftsreisende)

Kriterien unterschieden werden und auch in Kombination eingesetzt werden.

Beispiel 2.7. Eine Airline vertreibt Tickets für die Economy Class zu unterschiedlichen zuvor festgelegten Preisen. Dabei kann die Airline eine personenbezogene Preisdifferenzierung in Betracht ziehen, bei der z.B. je nach Alter, Geschlecht, Beruf oder Einkommen verschiedene Preiskategorien angeboten werden. Zu beachten ist, dass eine Unterscheidung zwischen günstigeren Economy-Class-Tickets und höherpreisigen First-Class-Tickets der Produktdifferenzierung zuzuordnen ist, da es sich um unterschiedliche Dienstleistungen handelt.

3 Abzugrenzen ist allerdings der Fall der Produktdifferenzierung, bei der ein höherer Preis für ein höherwertiges Produkt verlangt wird.

Kapazitätssteuerung

Die Kapazitätssteuerung ist ein Kernelement des Revenue Managements. Mit Hilfe verschiedenster Prognosemodelle versuchen die Unternehmen, ihre verfügbaren Kapazitäten einzuteilen und zu steuern. Nach [HT14] soll dabei geplant werden, welche Anteile den jeweiligen Marktsegmenten zu welchem Zeitpunkt zur Verfügung gestellt werden sollen.

Beispiel 2.8. Eine Airline geht davon aus, dass Geschäftsreisende im Allgemeinen eine höhere Zahlungsbereitschaft aufweisen und kurzfristig buchen. Freizeitreisende hingegen buchen schon weit im Voraus zu günstigen Konditionen. Im Rahmen der Kapazitätssteuerung soll geplant werden, wie viele Sitzplätze für zahlungskräftigere Kunden bis kurz vor Abflug zurückgehalten werden müssen. Verkauft die Airline zu viele Tickets zu einem vergünstigten Preis, kann es zu Umsatzeinbußen kommen.

Überbuchungssteuerung

Die Überbuchungssteuerung ist eines der ältesten Instrumente des Revenue Managements. Ziel der Überbuchungssteuerung ist es nach [KS08], ungenutzte Kapazitäten zu vermeiden. Diese entstehen dadurch, dass bereits verkaufte Leistungen kurzfristig nicht in Anspruch genommen werden. Unternehmen planen dementsprechend ein, dass einige Reisende ihre bezahlte Leistung nicht in Anspruch nehmen werden und bieten genau diese Leistung weiteren Kunden zum Verkauf an.

Beispiel 2.9. Eine Airline hat analog Beispiel 2.1 180 Sitzplätze für den Verkauf zur Verfügung. Die Airline rechnet beispielsweise mit fünf No-Show's[4] und verkauft daher 185 Flugtickets, obwohl insgesamt nur 180 Sitzplätze zur Verfügung stehen. Auf diese Weise wird der Umsatz erhöht, da selbst im Falle des Nichterscheinens einiger Passagiere mit einer vollen Auslastung zu rechnen ist. Sollten alle 185 Passagiere ihren Flug tatsächlich antreten wollen, müsste die Airline allerdings fünf Passagiere auf andere Flüge umbuchen.

2.2 EINFÜHRUNG IN DAS DYNAMIC PRICING

2.2.1 *Definition und Ziele*

Das Dynamic Pricing, zu deutsch *dynamische Preissetzung*, ist eine Preisstrategie, die besonders in der Luftfahrt weit verbreitet ist. Abdella u. a. beschreiben das Dynamic Pricing in der Luftfahrt als ein Spiel zwischen Airline und Kunde:

> „[...] dynamic pricing can be considered as a game between the retailer and consumers where each party tries to maximize its own profit" [Abd+21, S. 376].

4 Passagiere, die einen Flug gebucht haben, ihren Flug aber nicht antreten. Eine Stornierung des Fluges erfolgte jedoch nicht.

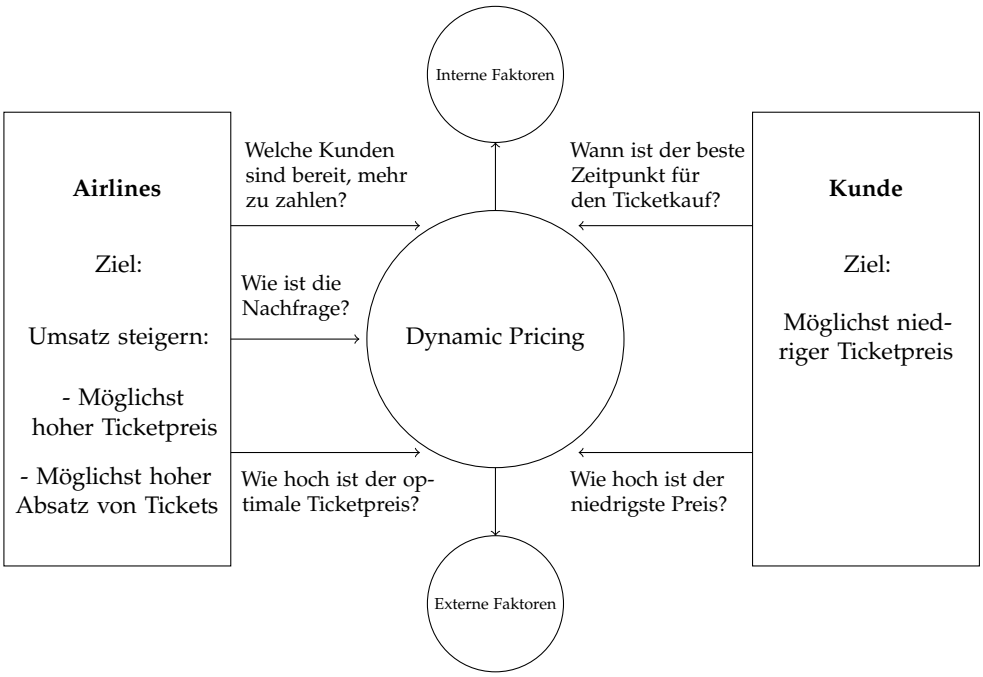

Abbildung 2.1: Dynamic Pricing. Nachempfunden von [Abd+21] und [Jos+20].

Diese zutreffende Beschreibung wird in Abbildung 2.1 veranschaulicht.

Das Dynamic Pricing ist, wie auch schon das Revenue Management, in der Literatur nicht eindeutig definiert und es gibt verschiedene Ansichten, was unter diesem Begriff zu verstehen ist. Einig ist man sich allerdings, dass der Begriff die Situation beschreibt, dass den Kunden zu verschiedenen Zeitpunkten verschiedene Preise angeboten werden [WB19], obwohl sich die Leistung nicht ändert.

Nach Auffassung von Klein und Steinhardt lässt sich das Dynamic Pricing

> „[...] als das planvolle Vorgehen eines Anbieters, seine einseitigen Preisvorgaben zu beliebigen Zeitpunkten innerhalb des Verkaufsprozesses („dynamisch“) zu ändern, um so auf veränderte nachfrage- oder konkurrenzbezogene Rahmenbedingungen mit dem Ziel der Maximierung des Gesamterlöses zu reagieren“ [KS08, S. 176]

definieren. Diese Auffassung wird auch in der vorliegenden Arbeit vertreten.

Beim Dynamic Pricing gelten die Preise nicht für den gesamten Verkaufszeitraum, wie es bei der statischen Preisgestaltung der Fall ist. Die Preise werden stattdessen entsprechend aktueller interner und externer Faktoren angepasst (vgl. Beispiel 2.10).

Beispiel 2.10. Bezogen auf die Luftfahrtbranche können interne Faktoren einer Fluggesellschaft, die den Ticketpreis beeinflussen, sein:

- Buchungsdatum,
- historische Preisdaten,

- Abflugdatum/Saison/Ferienzeiten,
- Anzahl verfügbarer Flüge/Sitzplätze auf dieser Route oder
- Flugdistanz.

Externe Faktoren können sein:

- Events am Zielort (Konzert, Konferenz),
- Geschehnisse (politische Lage, Naturkatastrophen),
- Wetterbedingungen oder
- wirtschaftliche Verhältnisse.

Weitere Ursachen, die eine dynamische Preisanpassung notwendig machen, sind nach [KS08]:

- Die verändernde Zahlungsbereitschaft der Kunden und/oder
- der Markteintritt von Niedrigpreis-Konkurrenten.

Das Ziel des Dynamic Pricing ist es, Preise in Echtzeit anpassen zu können, um auf diese Weise auf das Kundenverhalten, die Nachfrage oder die Konkurrenz reagieren zu können. Durch ständige Preisanpassungen soll insgesamt der Erlös gesteigert werden, indem beispielsweise die verändernde Zahlungsbereitschaft der Kunden gewinnbringend ausgenutzt wird.

2.2.2 *Anforderungen und Anwendungsbeispiele*

Klein und Steinhardt stellen Anforderungen vor, um das Dynamic Pricing erfolgreich einsetzen zu können [KS08]. Zum einen muss die Möglichkeit einer einfachen und kostengünstigen Änderung von Preisen bestehen. Wäre jede Preisänderung mit hohen Kosten verbunden, würde das Ziel des Dynamic Pricing, die Gewinnmaximierung, vermutlich nicht erreicht werden. Zum anderen darf keine Notwendigkeit bestehen, Preise bereits im Voraus festzulegen oder Preise für eine längere Zeitspanne fixieren zu müssen. Einsatzgebiete, auf die diese Anforderungen zutreffen sind neben der Luftfahrt beispielsweise die Hotellerie, der Mietwagenverleih oder der Online-Einzelhandel.

Beispiel 2.11. Ein Reiseveranstalter, der seine Reisen ausschließlich über gedruckte Kataloge[5] anbietet, wird es schwer haben, die Preise dynamisch anzupassen und auf interne oder externe Einflüsse zu reagieren. Eine Preisanpassung würde einen Neudruck der Kataloge erfordern, weshalb die Preise zuvor für den Verkaufszeitraum festgelegt und fixiert werden müssen. In

5 Eine zeitgleiche Bereitstellung der Angebote im Internet, wie es in der Realität üblich ist, ist für dieses Beispiel nicht vorgesehen.

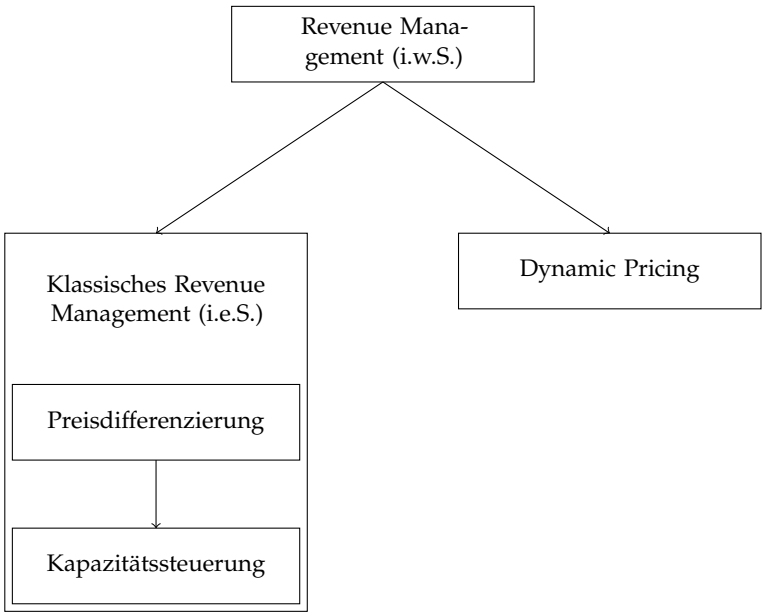

Abbildung 2.2: Abgrenzung Revenue Management und Dynamic Pricing. Nachempfunden von [KS08].

diesem Fall treffen die o.g. Anforderungen nicht zu und der Einsatz von Dynamic Pricing würde sich für diesen Reiseveranstalter aufgrund der damit verbundenen Kosten vermutlich nicht lohnen. Kommt unerwartet neue Konkurrenz auf den Markt, die deutlich günstigere Reisen anbieten kann, wäre es für den Reiseveranstalter schwierig, darauf selbst mit günstigeren Preisen zu reagieren.

Beispiel 2.12. Eine Fluggesellschaft bietet ihre Flüge ausschließlich online über deren Webseite zum Verkauf an. Es ist nicht erforderlich, die Preise vorab festzulegen und zu fixieren. Angenommen, die Airline bietet Flüge auf einer Strecke an, die auch von einer konkurrierenden Airline angeboten wird. Würde beispielsweise aufgrund eines Streiks der Flug der Konkurrenz gestrichen werden, wird die Nachfrage von Flugtickets auf dieser Strecke plötzlich deutlich ansteigen. Um nun den höchstmöglichen Gewinn zu erwirtschaften und die höhere Zahlungsbereitschaft der Kunden auszunutzen, kann die Airline die Preise kurzfristig deutlich anheben. Dies ist durch den Verkauf über die Webseite ohne hohen Aufwand möglich.

2.2.3 *Abgrenzung zum Revenue Management*

Ebenso wie die Definitionen selbst ist auch die Abgrenzung zwischen Dynamic Pricing und Revenue Management schwierig. Während einige Autoren die beiden Begriffe als gleichbedeutend ansehen und beide Begriffe der Nachfragesteuerung zuordnen [BB03], sehen andere Autoren wie Klein und Steinhardt das Dynamic Pricing neben der Kapazitätssteuerung und der Preisdifferenzierung als eine weitere Ausprägung des Revenue Managements an (vgl. Abbildung 2.2)[KS08].

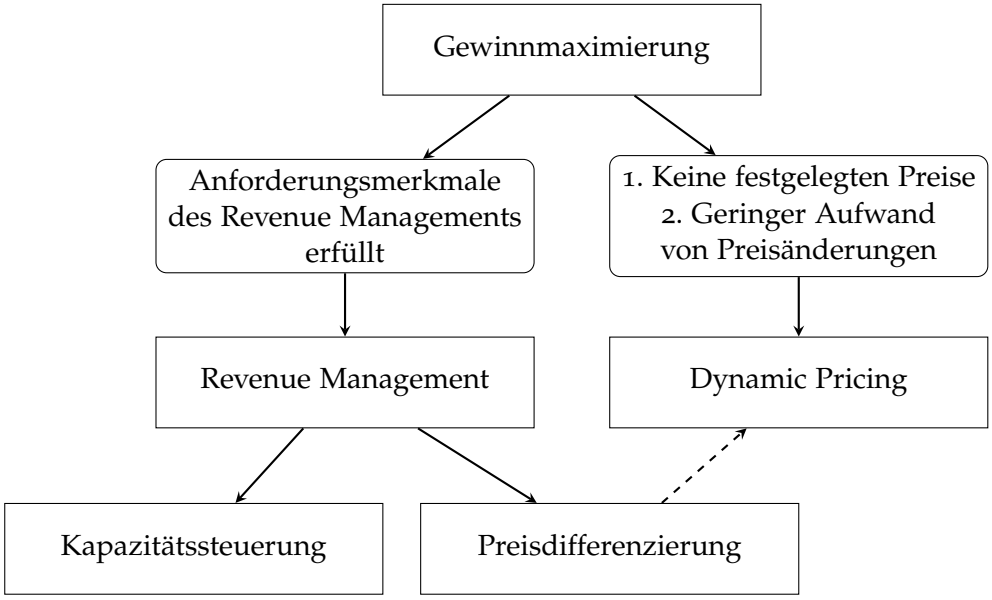

Abbildung 2.3: Angepasste Abgrenzung zwischen Revenue Management und Dynamic Pricing. Eigene Darstellung.

Das klassische Revenue Management, wie es in Abschnitt 2.1 dargestellt wurde, bezieht sich auf die mengenorientierte Kapazitätssteuerung, die beispielsweise durch verschiedene Buchungsklassen umgesetzt werden kann, denen jeweils ein fester Preis zugeordnet wurde. Das Dynamic Pricing hingegen ist preisorientiert. Es wird versucht, den Gewinn durch eine regelmäßige Anpassung der Preise zu maximieren. Die Bestimmung der Preise kann kontinuierlich erfolgen und der Situation jederzeit angepasst werden, während beim klassischen Revenue Management von Beginn an festgelegte Preise für die jeweiligen Buchungsklassen gelten.

Nach [KS08] kann das Dynamic Pricing implizit als eine Art der zeitlichen Preisdifferenzierung angesehen werden, wobei die Preisunterschiede lediglich auf den Buchungszeitpunkt, nicht jedoch auf den Zeitpunkt der Leistungserbringung zurückzuführen sind.

Die Ansichtsweise von Klein und Steinhardt, das Dynamic Pricing als eine Art der Preisdifferenzierung zu betrachten, bringt den Verdacht mit sich, dass Dynamic Pricing nur gewinnbringend eingesetzt werden könne, wenn die Anwendungsmerkmale des Revenue Managements (vgl. Unterabschnitt 2.1.3) erfüllt seien. Dies ist aber aus heutiger Sicht nicht zwingend der Fall, denn das Dynamic Pricing findet in vielen Bereichen Anwendung, in denen die Merkmale nicht erfüllt sind. Als Beispiel sei der Onlinehandel genannt, bei dem die Preise dynamisch an die Kundennachfrage und Angebote der Konkurrenz angepasst werden können, obwohl die Kapazitäten nicht zwingend begrenzt sein müssen.

Die geschilderte Problematik dieser Ansichtsweise macht eine alternative Abgrenzung der beiden Begriffe in Anlehnung an [Pri22] nötig. So kann das Dynamic Pricing als eigenständige Strategie der Gewinnmaximierung betrachtet werden, die nicht zwingend dem Revenue Management unter-

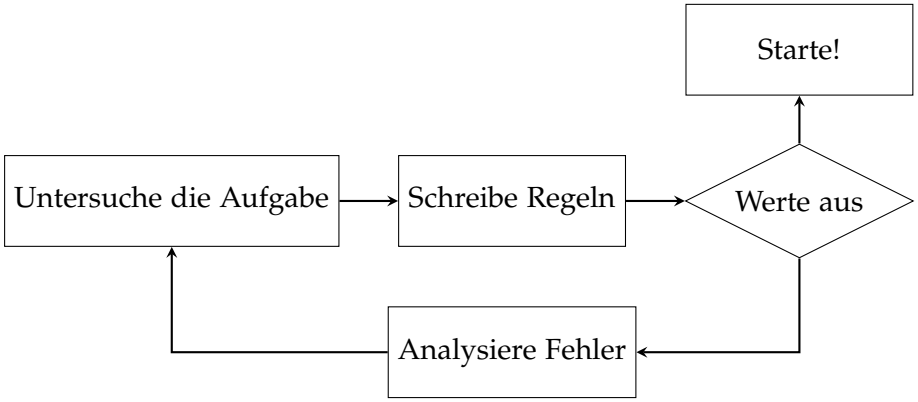

Abbildung 2.4: Die herkömmliche Herangehensweise von Lernprozessen. Nachempfunden von [Gér20].

geordnet werden muss. Es wird angenommen, dass das Dynamic Pricing dem Revenue Management sehr ähnlich ist und auch im Allgemeinen als eine Art der Preisdifferenzierung angesehen werden *kann*, aber *nicht zwangsläufig muss*. Demnach ist die zuvor genannte Betrachtungsweise von Klein und Steinhardt zwar anwendbar, der Begriff des Dynamic Pricing kann aber auch auf alternative Anwendungsgebiete ausgeweitet werden. Eine zusammenfassende Darstellung der Abgrenzung zwischen den beiden Begriffen kann Abbildung 2.3 entnommen werden.

Im Rahmen der vorliegenden Arbeit ist eine strikte Trennung zwischen den beiden Begriffen nicht erforderlich, da der Verkauf von Flugtickets durch Fluggesellschaften bereits die Anwendungsmerkmale des Revenue Managements erfüllt. Daher kann das Dynamic Pricing in diesem Fall auch als zeitliche Preisdifferenzierung betrachtet werden, es sei allerdings darauf hingewiesen, dass das Dynamic Pricing auch außerhalb der Luftfahrt Anwendung findet.

2.3 EINFÜHRUNG IN MACHINE LEARNING

In diesem Abschnitt wird in das Fachgebiet des Machine Learning (maschinelles Lernen) eingeführt. Nach einer allgemeinen Einführung in das Thema werden einige Anwendungsbeispiele gegeben. Es folgt eine ausführliche Beschreibung der verschiedenen Arten von Machine-Learning-Systemen, die sich an [Gér20] orientieren. Nachdem verschiedene Herausforderungen in diesem Zusammenhang dargestellt wurden, wird im letzten Unterabschnitt eine Auswahl geeigneter Methoden für den praktischen Teil dieser Arbeit getroffen. Es wird ausgearbeitet, welche Arten von Systemen sinnvoll sind und eingesetzt werden sollen.

2.3.1 *Was ist Machine Learning?*

Machine Learning (ML) ist ein Teilgebiet der künstlichen Intelligenz. Während das Gebiet der künstlichen Intelligenz sehr breit aufgestellt ist, verfolgt

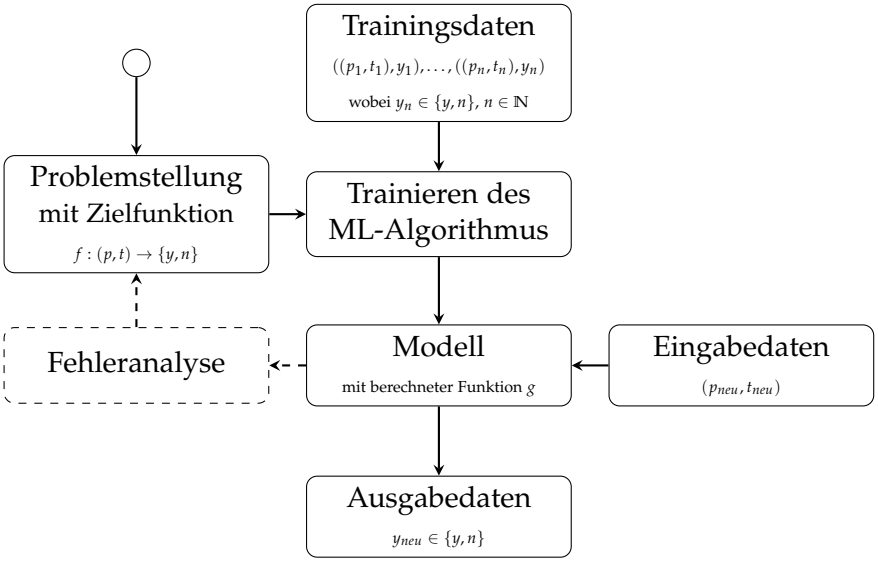

Abbildung 2.5: Der Lernprozess von Machine-Learning-Problemen. Eigene Darstellung in Anlehnung an [Gér20] und [AMMIL12].

Machine Learning nur einen Ansatz: Computer werden so programmiert, dass sie aus vorhandenen Datenmengen lernen, um bestimmte Aufgaben zu lösen [RRC19; AMMIL12]. Ziel von Machine Learning ist es daher, neues Wissen aus Daten zu gewinnen, indem Strukturen und Muster innerhalb großer Datenmengen erkannt werden [RRC19]. Nach [Gér20] ist Machine Learning durch dessen Eigenschaften geeignet für:

- Aufgaben, deren Lösung in herkömmlicher Weise (vgl. Abbildung 2.4) eine lange Liste von Regeln beinhalten würde.
- Komplexe Aufgaben, für die es mit herkömmlichen Methoden keine Lösung gibt.
- Fluktuierende Umgebungen, da sich Machine-Learning-Systeme neuen Daten anpassen können.
- Große Datenmengen, aus denen Erkenntnisse gewonnen werden sollen.

Der Lernprozess eines Machine-Learning-Problems ist in Abbildung 2.5 am Beispiel der Klassifikation dargestellt. Zu Beginn existiert eine Problemstellung, die gelöst werden soll. Diese wird in der vorliegenden Arbeit durch die Frage dargestellt, ob ein bestimmter Flug mit einem Flugpreis p zu einem bestimmten Zeitpunkt t gebucht werden soll oder nicht. Ausgang ist eine bisher unbekannte Zielfunktion $f : (p, t) \to \{y, n\}$. Mit Hilfe von historischen Trainingsdaten können Machine-Learning-Algorithmen trainiert werden. Auf diese Weise entsteht ein Modell, das in diesem Beispiel eine Funktion g ist, die alle Trainingsdaten bestmöglich erfüllt, aber nicht zwingend der optimalen Zielfunktion f entsprechen muss. Die Funktion f würde im

Gegensatz zu g alle Trainingsbeispiele fehlerfrei abdecken und für jedes Tupel (p, t) eine korrekte Antwort bestimmen. Eine solche optimale Funktion f ist allerdings oft schwer zu bestimmen und kann auch zu Nachteilen wie dem Overfitting (vgl. Unterabschnitt 2.3.7) führen.

Das auf diese Weise entstandene Modell kann anschließend optimiert und getestet werden. Sollten Anpassungen nötig sein, kann der Prozess iterativ optimiert werden. Andernfalls kann dieses Modell nun auf unbekannte Eingabedaten angewendet werden. Es werden Tupel bestehend aus Flugpreis und Zeitpunkt als Eingabeparameter übergeben, für die als Ausgabe eine berechnete Empfehlung zurückgegeben wird.

2.3.2 *Anwendungsgebiete*

Machine Learning findet aktuell in vielen Bereichen Anwendung und kann unter anderem durch Regression, Klassifikation, Vorhersage oder Clustering[6] umgesetzt werden.

Ein klassisches Beispiel für eine Klassifikation ist der Spamfilter für E-Mail-Programme [GC09]. Aber auch in der Medizin bieten Machine-Learning-Ansätze großes Potenzial. In [Kou+15] wird dargestellt, wie die Krebserkennung durch Machine Learning unterstützt werden kann.

Weitere Einsatzgebiete sind die Betrugserkennung, die Produktempfehlung in Online-Shops oder auch die dynamische Preisgestaltung, die in dieser Arbeit behandelt wird.

2.3.3 *Überwachtes vs. Unüberwachtes Lernen*

Überwachtes Lernen ist eine Art des Machine Learning. Es zeichnet sich dadurch aus, dass dem Algorithmus Trainingsdaten übergeben werden, die bereits die gewünschte Ausgabe enthalten. Diese Lösung wird als Label bezeichnet. Algorithmen des überwachten Lernens sind beispielsweise die *lineare Regression* (vgl. Abschnitt 3.1), *Entscheidungsbäume* (vgl. Abschnitt 3.3), *k-nächste Nachbarn* (vgl. Abschnitt 3.5), oder *Support Vector Machines* (vgl. Abschnitt 3.6).

Beispiel 2.13. Anhand verschiedener Merkmale (Prädiktoren) soll der günstigste Flugpreis vorhergesagt werden. Dem Lernalgorithmus werden Daten übergeben, die neben den Merkmalen *Abflugdatum*, *Abflughafen* und *Zielflughafen* auch den günstigsten *Flugpreis* $p \in \mathbb{N}^+$ als Label enthalten.

Beim **unüberwachten Lernen** werden für das Training Daten verwendet, die nicht gelabelt sind. Dem Algorithmus ist dementsprechend keine Lösung bekannt, weshalb dieser ohne Anleitung lernen muss. Der Algorithmus versucht selbständig Muster und Zusammenhänge in den Daten zu erkennen.

6 Gruppierung von Daten, indem ähnliche Elemente durch ein Ähnlichkeitsmaß zusammengefasst werden.

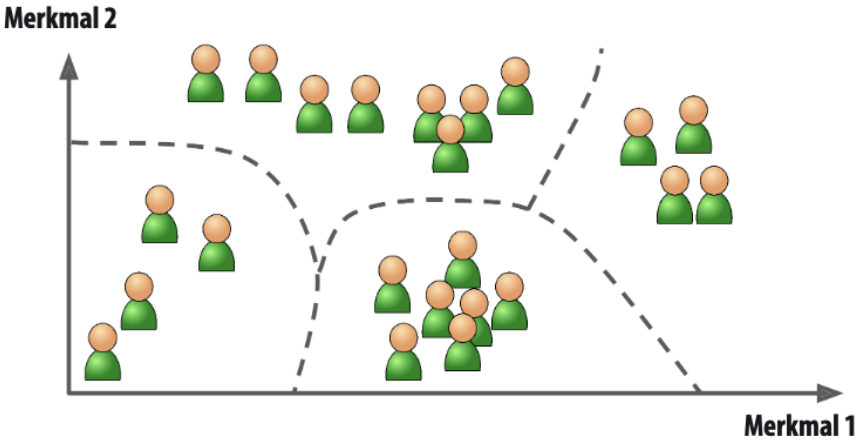

Abbildung 2.6: Clustering potenzieller Kunden in verschiedene Kundengruppen. Entnommen aus [Gér20].

Algorithmen des unüberwachten Lernens sind beispielsweise *k-Means* und *DBSCAN*, die im Rahmen dieser Arbeit allerdings nicht vorgestellt werden.

Beispiel 2.14. Eine Airline möchte die Zahlungsbereitschaft ihrer Kunden analysieren, um die Preise dynamisch an verschiedene Kundengruppen anzupassen. Es können persönliche Daten verwendet werden, um bestimmte Eigenschaften zu finden, die auf eine höhere Zahlungsbereitschaft hindeuten (Berufsgruppe, Einkommen, Grund der Reise etc.). Auf diese Weise können Kunden in unterschiedliche Gruppen (sogenannte Cluster) eingeordnet werden. Abbildung 2.6 zeigt mögliche Cluster.

2.3.4 *Batch- vs. Online-Learning*

Beim **Batch-Learning** lernt das System zunächst anhand der gesamten Trainingsdaten. Erst nach Abschluss des Lernprozesses kann dieses eingesetzt werden. Während des Einsatzes kann das System nicht inkrementell trainiert werden. Kommen neue Daten hinzu, um beispielsweise die Genauigkeit von Vorhersagen verbessern zu können, beginnt ein neuer Lernprozess, bei dem alte sowie neue Daten als Trainingsdaten verwendet werden. Da ein vollendeter Lernprozess dauerhaft in sich abgeschlossen ist, wird diese Art des maschinellen Lernens auch als *Offline-Learning* bezeichnet.

Werden die Daten regelmäßig aktualisiert oder treffen diese kontinuierlich ein (z.B. Aktienkurse), ist das Batch-Learning ungeeignet. Der immer wieder neu beginnende Lernprozess dauert lange und beansprucht Rechenkapazitäten. Für diese Fälle eignet sich das **Online-Learning** besser, bei dem der Algorithmus inkrementell trainiert wird. Stehen neue Daten zur Verfügung, kann das System sofort mit den neuen Daten lernen, ohne einen neuen Lernprozess starten zu müssen. Es können einzelne Daten oder kleine Datenpakete (sogenannte Mini-Batches) nacheinander hinzugefügt werden. Ein Vorteil ist, dass Daten, die bereits für den Lernprozess verwendet wurden, nicht mehr benötigt werden, wodurch Speicherkapazitäten gespart werden.

2.3.5 *Instanzbasiertes vs. Modellbasiertes Lernen*

Das **instanzbasierte Lernen** basiert auf dem Auswendiglernen von Trainingsbeispielen. Anhand eines Ähnlichkeitsmaßes ist es dem System anschließend möglich, das Gelernte zu verallgemeinern und auf unbekannte Daten anzuwenden.

Beispiel 2.15. Seien in einem Trainingsdatensatz die in Beispiel 2.13 aufgeführten Merkmale und zusätzlich das Merkmal *Wochentag* vorhanden. Beim instanzbasierten Lernen kann nun der Flugpreis für einen neuen Flug bestimmt werden, indem im Trainingsdatensatz nach dem ähnlichsten Flug gesucht wird. Dieser kann anhand eines Ähnlichkeitsmaßes ausgewählt werden, indem beispielsweise der Flug mit gleichem Abflughafen, Zielflughafen und Wochentag betrachtet wird. Als Vorhersage kann dann der Flugpreis des ähnlichsten Fluges zurückgegeben werden.

Beim **modellbasierten Lernen** hingegen wird anhand der Trainingsdaten ein Modell erstellt, das diese Daten verallgemeinert. Unbekannte Daten werden nicht mit den Trainingsbeispielen abgeglichen, sondern es wird mit Hilfe des erstellten Modells eine Vorhersage bestimmt. Dieses Modell kann beispielsweise graphisch betrachtet eine Gerade sein, die alle Datenpunkte bestmöglich schneidet. Anhand dieser Geraden kann auch für neue Daten ein Wert abgelesen werden (vgl. lineare Regression in Abschnitt 3.1).

2.3.6 *Trainieren und Validieren*

Um einen Algorithmus trainieren zu können, sollte der Datensatz zunächst in einen Trainings- und Testdatensatz aufgeteilt werden. Der Trainingsdatensatz wird dem Algorithmus für den Lernprozess übergeben. Anhand dieser Daten wird das Modell erstellt.

Der Testdatensatz dient der Validierung und soll nach [Gér20] ca. 20% der vorhandenen Daten ausmachen. Es wird überprüft, ob das Modell für bisher unbekannte Daten die richtigen Vorhersagen trifft. Auf diese Weise kann die Qualität des Modells anhand eines Qualitätsmaßes gemessen werden.

Für die verschiedenen Machine-Learning-Algorithmen gibt es abhängig vom jeweiligen Algorithmus verschiedene sogenannte Hyperparameter, deren Werte den Lernprozess steuern. Bei der Auswahl dieser Hyperparameter kann es vorkommen, dass die Parameter zu sehr an die Trainingsdaten angepasst werden und das Modell auf dem Testdatensatz keine guten Ergebnisse liefert (vgl. Unterabschnitt 2.3.7). Um dieses Problem zu umgehen, kann die *k-Folds-Kreuzvalidierung* verwendet werden.

Hierfür wird der Trainingsdatensatz erneut aufgeteilt. Dieser teilt sich in k zufällige Teilmengen, die sogenannten Folds. Anschließend werden k Trainingsdurchläufe gestartet, bei denen jeweils $k-1$ Folds zusammengefasst als Trainingsbeispiele an den Algorithmus übergeben werden. Mit dem verbliebenen Fold (Validierungsdatensatz) kann im Anschluss die Qualität des Modells bestimmt werden. Auf diese Weise ist es möglich, die Qualität ver-

schiedener Modelle zu bestimmen, ohne den tatsächlichen Testdatensatz anrühren zu müssen.

Nachdem das beste Modell mit dessen Hyperparametern bestimmt wurde, wird das Modell erneut mit dem gesamten zu Beginn erzeugten Trainingsdatensatz (inkl. Validierungsdatensatz) trainiert. Mit Hilfe des Testdatensatzes kann im Anschluss die abschließende Beurteilung der Modellqualität erfolgen.

Als Qualitätsmaß kann bei der Klassifikation beispielsweise eine Konfusionsmatrix erzeugt werden. Diese stellt die tatsächlichen Kategorien (Zeilen) mit den vorhergesagten Kategorien (Spalten) gegenüber.

Beispiel 2.16. Anhand eines Testdatensatzes, der 10.000 Daten enthält, wurden Vorhersagen einer Kategorie (*true/false*) getroffen. Die Ergebnisse der Klassifikation sind in Tabelle 2.1 dargestellt. Es ist zu erkennen, dass 3.800 Werte korrekt als *false* klassifiziert wurden und 1.200 Werte fälschlicherweise als *true* klassifiziert wurden, obwohl diese eigentlich der Kategorie *false* angehören.

	false	true
false	3.800	1.200
true	2.000	3.000

Tabelle 2.1: Konfusionsmatrix. Eigene Darstellung.

2.3.7 *Herausforderungen und Overfitting*

Herausforderungen beim Trainieren von Machine-Learning-Algorithmen können entweder im Zusammenhang mit dem Datensatz oder dem Algorithmus selbst entstehen.

Herausforderungen in Bezug auf die Daten sind die Datenqualität und -quantität. Minderwertige Daten sollten vor dem Trainieren des Modells bereits im Rahmen der Vorverarbeitung (vgl. Unterabschnitt 2.5.3) beseitigt worden sein, da es ansonsten zu Fehlschlüssen kommen kann und das Finden von Zusammenhängen erschwert wird. Minderwertige Daten sind beispielsweise Fehler, Ausreißer oder Rauschen. Auch die Merkmalsauswahl beeinflusst die Datenqualität und sollte daher mit Bedacht erfolgen. Irrelevante Merkmale, die die Vorhersagegenauigkeit nicht beeinflussen, sollten nicht verwendet werden. Falls nötig, können auch neue Merkmale durch das Kombinieren vorhandener Merkmale erzeugt werden.

In Bezug auf die Datenquantität muss der zur Verfügung stehende Datensatz groß genug sein, um diesen in einen repräsentativen Trainings- und Testdatensatz aufteilen zu können (vgl. Unterabschnitt 2.3.6).

Eine Herausforderung im Zusammenhang mit dem Algorithmus ist neben der Wahl passender Algorithmen insbesondere das sogenannte **Overfitting** (Überanpassung). Dieses bezeichnet ein unerwünschtes Verhalten, bei dem das trainierte Modell zwar gut auf den Trainingsdaten funktioniert, für den

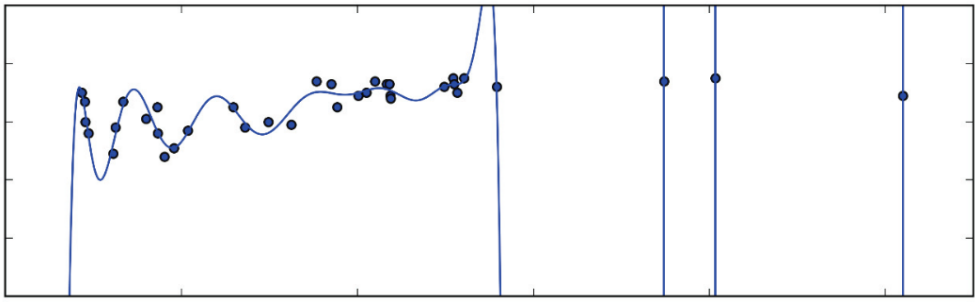

Abbildung 2.7: Graphische Darstellung des Overfittings. Entnommen aus [Gér20].

Testdatensatz (und somit auch für neue Daten) aber schlechte Vorhersagen trifft. Die Beispieldaten konnten nicht gut durch das Modell verallgemeinert werden. Dieses Verhalten ist in Abbildung 2.7 dargestellt. Es ist gut zu erkennen, dass die Modellfunktion zwar die Datenpunkte sehr gut trifft, für zwischenliegende Punkte würden sich aber fehlerhafte Werte ergeben.

Auf der anderen Seite kann es aber auch zum Underfitting kommen, bei dem das Modell zu einfach ist und selbst auf den Trainingsdaten schlechte Vorhersagen erzielt.

Ziel ist es, ein Modell auf die Weise zu trainieren, dass es nicht zu einfach, aber auch nicht zu komplex ist.

2.3.8 *Auswahl geeigneter Ansätze*

Für den praktischen Teil dieser Arbeit stehen ca. 47.000 Flugpreisdaten zur Verfügung (vgl. Abschnitt 4.2). Bei dieser Datenmenge kann es herausfordernd sein, Muster oder Zusammenhänge mit herkömmlichen Methoden zu entdecken, da sich eine umfangreiche Liste an Regeln ergeben würde, um Werte vorhersagen zu können. Dementsprechend ist es sinnvoll, Machine Learning und dessen Vorteile für die Flugpreisvorhersagen zu nutzen.

Der Datensatz enthält bereits die Flugpreise vergangener Flüge (vgl. Kapitel 4, Unterabschnitt 4.2.2). Es handelt sich um einen gelabelten Datensatz, was auf überwachtes Lernen schließen lässt. Dies ist sinnvoll, da bereits alle Merkmale und auch die Zielvariable bekannt sind.

In diesem Vorhaben sind bereits alle benötigen Daten vorhanden und es ist nicht absehbar, dass weitere Daten hinzukommen werden, weshalb der Ansatz des Batch-Learning verfolgt werden kann. Außerdem sind ausreichend Speicher- und Rechenkapazitäten vorhanden, was das Online-Learning in diesem Fall entbehrlich macht.

Um eine Auswahl zwischen instanzbasiertem und modellbasiertem Lernen treffen zu können, muss zunächst abgewogen werden, ob es für Vorhersagen ausreichend ist, neue Daten lediglich mit den Trainingsdaten abzugleichen. Anschließend würde der Preis des ähnlichsten Fluges als Vorhersagewert zurückgegeben werden. Ziel dieser Arbeit ist es allerdings, Muster und Zusammenhänge in vergangenen Daten zu finden, um ein möglichst

allgemeines Modell zur Vorhersage zu erstellen. Daher wird der Ansatz des modellbasierten Lernens verfolgt.

2.4 ABSTANDSMETRIKEN

Einige Machine-Learning-Algorithmen (unter anderem der in Abschnitt 3.5 dargestellte Algorithmus *k-Nearest Neighbors*) basieren auf dem Vergleich verschiedener Datenobjekte. Um diese verschiedenen Datenobjekte miteinander vergleichen zu können, werden Abstandsmaße benötigt.

Es gibt verschiedene geeignete Abstandsmaße, von denen im Folgenden zunächst drei Maße für numerische Attribute[7] und ein Abstandsmaß für nominale Attribute[8] vorgestellt werden sollen.

Abstandsmaße für numerische Attribute

Sei T ein Trainingsdatensatz mit $x, y \in T$ als einzelne Datenpunkte des Trainingsdatensatzes, bestehend aus n unterschiedlichen numerischen Attributen mit den Attributausprägungen $x_1, \ldots, x_n$ und $y_1, \ldots, y_n$. Der Abstand zwischen den beiden Datenpunkten x und y wird mit $d(x, y)$ bezeichnet.

Als Abstandsmaße für numerische Attribute eignen sich Metriken, da diese die folgenden drei **Eigenschaften einer Metrik** erfüllen:

- Reflexivität: $d(x, x) = 0$,
- Nicht-Negativität: $d(x, y) \geq 0$,
- Symmetrie: $d(x, y) = d(y, x)$.

Die *Euklidische Distanz*, die *Manhattan-Distanz* und auch die *Minkowski-Distanz* sind geeignete Metriken, die o.g. Eigenschaften erfüllen und daher oft als Abstandsmaß verwendet werden. Die Distanzen lassen sich wie folgt berechnen:

Euklidische Distanz:

$$d_{euklid}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}\,, \tag{2.1}$$

Manhattan-Distanz:

$$d_{manhattan}(x, y) = \sum_{i=1}^{n} \mid x_i - y_i \mid\,, \tag{2.2}$$

7 Dies können ganzzahlige oder reellwertige Werte sein, die messbar sind und auf die sich quantitative Methoden anwenden lassen.

8 Dies können Namen, Symbole oder Bezeichnungen sein, für die sich keine quantitativen Methoden wie die Mittelwertberechnung durchführen lassen.

Minkowski-Distanz:

$$d_{minkowski}(x,y) = \left(\sum_{i=1}^{n} \mid x_i - y_i \mid^{\alpha} \right)^{\frac{1}{\alpha}} . \tag{2.3}$$

Anzumerken ist, dass die Euklidische Distanz genau der Minkowski-Distanz mit $\alpha = 2$ und die Manhattan-Distanz genau der Minkowski-Distanz mit $\alpha = 1$ entspricht.

Abstandsmaß für nominale Attribute

Sei T ein Trainingsdatensatz mit $x, y \in T$ als einzelne Datenpunkte des Trainingsdatensatzes, bestehend aus n unterschiedlichen nominalen Attributen mit den Attributausprägungen $x_1, \ldots, x_n$ und $y_1, \ldots, y_n$. Der Abstand zwischen den beiden Datenpunkten x und y wird mit $d(x,y)$ bezeichnet.

Der Abstand kann wie folgt berechnet werden:

$$d(x,y) = \frac{n-m}{n}, \text{ wobei } m = |\{k \in 1, \ldots, n\} : x_k = y_k| . \tag{2.4}$$

Dabei entspricht n der Anzahl der verschiedenen Attribute und m entspricht der Anzahl an gleichen Attributen in den Datenpunkten x und y. Sind alle Attributausprägungen von x und y identisch, gilt $m = n$ und $d(x,y) = 0$. Sind hingegen alle Ausprägungen verschieden, gilt $m = 0$ und dementsprechend $d(x,y) = 1$.

2.5 VORGEHENSMODELL CRISP-DM

Unter einem Vorgehensmodell versteht Gnatz einen *„standardisierten organisatorischen Rahmen für den idealen Ablauf eines Projektes“* [Gna05, S. 2]. Ein Prozess wird in verschiedene zeitliche und inhaltlich abgegrenzte Phasen unterteilt, wodurch dieser besser geplant und durchgeführt werden kann.

Ein in der Literatur oft zu findendes Vorgehensmodell im Zusammenhang mit Data Mining[9] ist der von Chapman u. a. beschriebene *Cross Industry Standard Process for Data Mining* (CRISP-DM) [Cha+00]. Dieses Vorgehensmodell wurde von Clark in [Cla18] auf Machine-Learning-Prozesse übertragen und angepasst, wodurch sich dieser für den praktischen Teil dieser Arbeit eignet. Das iterative Modell lässt sich in sechs Phasen aufteilen, die in Abbildung 2.8 dargestellt sind. Nachfolgend werden allgemeine Erläuterungen zu den einzelnen Phasen gegeben, um ein Verständnis für das Vorgehensmodell zu entwickeln. Im praktischen Teil dieser Arbeit wird dieses Modell umgesetzt, um eine strukturierte Lösung zu erarbeiten.

9 Data Mining beschreibt die Extraktion von neuem Wissen aus großen Datenbeständen, indem Muster und Zusammenhänge erkannt werden und Regeln aus diesen Mustern abgeleitet werden [BH09].

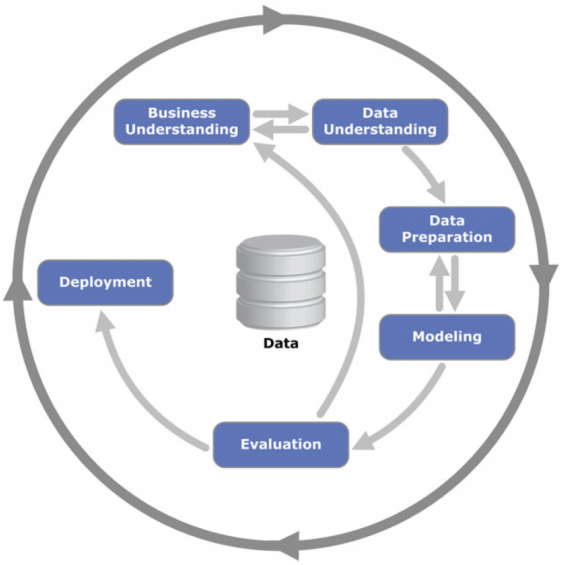

Abbildung 2.8: Das CRISP-DM Modell nach Chapman u. a. Entnommen aus [Cha+00].

2.5.1 *Business Understanding*

In dieser Phase wird die Problemstellung erarbeitet und es werden die Ziele festgelegt, die durch die Anwendung von Machine Learning erreicht werden sollen. Es stellt sich die Frage, was durch Machine Learning optimiert oder herausgefunden werden soll. Zudem werden Hintergrundinformationen (ggf. mit Hilfe von Domänenexperten) gesammelt, die für die Umsetzung des Projektes nötig sind.

2.5.2 *Data Understanding*

In der zweiten Phase geht es um die Datenbeschaffung und um erste Einsichten in die Daten. Es werden relevante Datensätze beschafft und ein allgemeines Verständnis für die Daten erlangt, sodass ein Überblick über die vorhandenen Daten besteht und alle Merkmale des Datensatzes nachvollzogen werden können. Erste Einsichten in die Daten erfolgen durch eine explorative Datenanalyse[10].

2.5.3 *Data Preparation*

In dieser Phase werden die vorhandenen Daten in eine für Machine-Learning-Algorithmen nutzbare Form gebracht und der finale Datensatz erzeugt. Fehlende oder fehlerhafte Daten müssen bereinigt werden und es können weitere Merkmale erzeugt oder entfernt werden (Merkmalsauswahl). Zudem kann es nötig sein, Daten zu transformieren oder zu skalieren.

10 Erforschung vorhandener Daten nach bisher unbekannten Zusammenhängen und Mustern.

2.5.4 *Modeling*

Der finale Datensatz wird zur weiteren Verwendung in einen Trainings- und einen Testdatensatz aufgeteilt (vgl. Unterabschnitt 2.3.6). Es wird eine Auswahl der vielversprechendsten Machine-Learning-Algorithmen getroffen, mit denen anschließend das Modell erstellt, trainiert und nach Messung der Modellqualität optimiert werden kann. Auf diese Weise können die optimalen Hyperparameter bestimmt werden. Zudem können die wichtigsten Merkmale herausgearbeitet werden, um ggf. unwichtige Merkmale in weiteren Iterationen zu vernachlässigen.

2.5.5 *Evaluation*

Im vorletzten Schritt wird geprüft, ob die Ergebnisse des Modells die in der ersten Phase festgelegten Ziele erreichen. Beispielsweise kann zur Bewertung die Genauigkeit von Vorhersageergebnissen herangezogen werden. Es wird entschieden, ob das Modell verwendet wird oder ob Anpassungen durch weitere Iterationen notwendig sind.

2.5.6 *Deployment*

Im letzten Schritt werden die Erkenntnisse zur Wieder- und Weiterverwendung aufbereitet. Das Modell wird in der Praxis auf unbekannten Daten eingesetzt.

3

MACHINE-LEARNING-ALGORITHMEN

In diesem Kapitel sollen die für die vorliegende Arbeit relevanten Machine-Learning-Algorithmen vorgestellt werden. Für den praktischen Teil dieser Arbeit liegt ein gelabelter Datensatz vor, weshalb ausschließlich überwachte Lernalgorithmen dargestellt werden. Für alle Algorithmen kann somit angenommen werden, dass die Trainingsdaten bereits das korrekte Label enthalten.

Es wurden insgesamt sieben Machine-Learning-Algorithmen ausgewählt, mit denen in der Literatur bereits gute Vorhersageergebnisse erzielt wurden. Die verschiedenen Machine-Learning-Algorithmen wurden mit Hilfe mehrerer Quellen erarbeitet, die nachfolgend angegeben werden. Die ausgewählten Algorithmen sind:

- Lineare Regression [Gér20],
- Naiver Bayes-Klassifikator [CB17; Sci23a],
- Entscheidungsbäume [Gér20; BKI03; Bur19],
- Random Forest [Gér20],
- k-Nearest Neighbors [Bur19; Sci23b],
- Support Vector Machine [Bur19; Hea+98; Wil18],
- Multilayer Perceptron [Alp22; BTL08; Gér20; Sci23c].

3.1 LINEARE REGRESSION

Bei der linearen Regression handelt es sich um einen überwachten Lernalgorithmus, der zur Vorhersage quantitativer Zielwerte (z.B. Flugpreise, Größen, Häufigkeiten) dient. Ein Zielwert kann berechnet werden, indem die gewichtete Summe der Eingabemerkmale gebildet und eine Konstante, der sogenannte Bias-Term, hinzuaddiert wird.

Die Berechnung des Zielwerts hat folgende allgemeine Form:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n \, . \tag{3.1}$$

Dabei gilt:

$\hat{y}$ ist der Vorhersagewert,

x_i ist das i-te Merkmal für $1 \leq i \leq n$,

n ist die Anzahl der Merkmale,

θ_j ist für $j = 0$ der Bias-Term und für $j > 0$ der Modellparameter.

In Vektorschreibweise lässt sich dies noch kompakter darstellen:

$$\hat{y} = \theta^\top x \, . \tag{3.2}$$

Dabei gilt:

$\hat{y}$ ist der Vorhersagewert,

x ist der Spaltenvektor mit den Merkmalen x_i für $0 \leq i \leq n$, wobei $x_0 = 1$ gilt,

$\theta^\top$ ist der transponierte Spaltenvektor der Modellparameter.

Es wird eine lineare Beziehung zwischen den unabhängigen Variablen (Eingabemerkmale) und der abhängigen Variable (Zielwert) angenommen und zur Vorhersage ausgenutzt. Die unabhängigen Variablen werden auch Prädiktoren genannt, da diese zur Berechnung herangezogen und zur Vorhersage genutzt werden.

Da es sich um einen überwachten Lernalgorithmus handelt, werden Trainingsbeispiele für das Training verwendet. Graphisch betrachtet ist das Ziel des Lernprozesses der linearen Regression, eine Gerade (im eindimensionalen Raum), eine Ebene (im zweidimensionalen Raum) oder eine Hyperebene (im n-dimensionalen Raum mit n > 2) durch Bestimmung der optimalen Modellparameter zu erzeugen, die so nah wie möglich an allen Trainingsbeispielen liegt. Dies lässt sich am folgenden Beispiel nachvollziehen.

Beispiel 3.1. Seien für einen fiktiven Flug die in Tabelle 3.1 dargestellten Flugpreise gegeben. Bei der Anzahl der Tage vor Abflug handelt es sich um die unabhängige Variable (Prädiktor), während es sich bei dem Preis um die abhängige Variable handelt. Es ist ein linearer Zusammenhang zu erkennen: Je niedriger die Anzahl der Tage vor Abflug ist, desto höher ist der Flugpreis. Die Regressionsgerade ist dann genau die Gerade, die so nah wie möglich an allen Trainingsbeispielen liegt.

Tage vor Abflug	45	40	35	30	25	20	15	10	5
Flugpreis	120	139	159	159	164	179	185	205	230

Tabelle 3.1: (Fiktive) Flugpreise. Eigene Darstellung.

Die Bestimmung der optimalen Modellparameter kann auf verschiedene Weisen erfolgen. Als Beispiel wird das Verfahren anhand der *Methode der kleinsten Quadrate* erläutert. Bei dieser Methode wird die Summe der quadrierten Abweichungen zwischen Vorhersagewert (Regressionsgerade) und tatsächlichem Wert minimiert. Die Formel der mittleren quadratischen Abweichung (MSE - mean squared error) lautet:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (\theta^\top x^{(i)} - y^{(i)})^2 \, . \tag{3.3}$$

Dabei bezeichnet $\theta^\top x^{(i)}$ den Vorhersagewert, der mit Hilfe der Modellparameter und der Prädiktoren bestimmt wird, und $y^{(i)}$ den tatsächlichen Wert des i-ten Datenpunktes für $1 \leq i \leq n$ mit $0 < n$ Datenpunkten. Zu beachten ist, dass für jeden Datenpunkt $x_0^{(i)} = 1$ gilt. Für das Beispiel aus Tabelle 3.1 wird die mittlere quadratische Abweichung wie folgt bestimmt. Diesen Wert gilt es durch Bestimmung passender Parameter für θ zu minimieren.

$$\begin{aligned} MSE = \frac{1}{9}((\theta_0 + \theta_1 45 - 120)^2 + (\theta_0 + \theta_1 40 - 139)^2 + \\ \ldots + (\theta_0 + \theta_1 10 - 205)^2 + (\theta_0 + \theta_1 5 - 230)^2) \, . \end{aligned} \tag{3.4}$$

Wie genau dieses Minimierungsproblem gelöst werden kann, hängt von der Art der Modellfunktion ab. Für das eindimensionale Beispiel lassen sich die Werte für θ_0 und θ_1 auch direkt ohne schrittweise Approximation des Minimums der mittleren quadratischen Abweichung wie folgt berechnen. Dabei bezeichne x_i die Anzahl der Tage vor Abflug, y_i den dazugehörigen Flugpreis und $\bar{x}$ sowie $\bar{y}$ jeweils das arithmetische Mittel aller x_i bzw. y_i.[1]

$$\theta_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \, , \tag{3.5}$$

$$\theta_0 = \bar{y} - \theta_1 \bar{x} \, . \tag{3.6}$$

Werden die Werte aus Tabelle 3.1 eingesetzt, ergeben sich die gerundeten Werte $\theta_1 \approx -2,367$ und $\theta_0 \approx 230,285$. Es entsteht eine Regressionsgerade der Form $f(x) = 230,285 - 2,367x$. Nun können Werte mit Hilfe dieser Geraden vorhergesagt werden, indem die Prädiktoren in die Funktion f eingesetzt werden. Beispielsweise lässt sich der Flugpreis mit einer Vorlaufzeit von acht Tagen vor Abflug vorhersagen: $f(8) = 230,285 - 2,367 \cdot 8 = 211,349$. Der vorhergesagte Flugpreis beträgt demnach ungefähr 211 Euro.

3.2 NAIVER BAYES-KLASSIFIKATOR

Der naive Bayes-Klassifikator ist ein überwachter Lernalgorithmus, der für Klassifikationsprobleme verwendet wird. Der Algorithmus basiert auf dem *Satz von Bayes*, der der Wahrscheinlichkeitsrechnung zuzuordnen ist und es ermöglicht, bedingte Wahrscheinlichkeiten zu berechnen. Der Algorithmus gilt als naiv, weil die Attribute als statistisch unabhängig betrachtet werden und alle Attribute zur Berechnung der Wahrscheinlichkeit herangezogen werden. Dass die Attribute stochastisch unabhängig sind, bedeutet, dass das Wissen über eine konkrete Attributausprägung nichts über andere Attribute aussagt.

1 Es handelt sich nicht mehr um eine Vektorschreibweise.

Seien A und B Ereignisse und $P(A)$ bzw. $P(B)$ die Wahrscheinlichkeiten für deren Auftreten, dann kann die bedingte Wahrscheinlichkeit $P(A|B)$ für das Auftreten des Ereignisses A unter der Voraussetzung, dass B zutrifft, wie folgt bestimmt werden:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \ . \tag{3.7}$$

Dies kann auf einen Datensatz wie folgt übertragen werden: Sei T ein Trainingsdatensatz und $x \in T$ ein Datenpunkt des Trainingsdatensatzes, bestehend aus n unterschiedlichen Attributen mit den Attributausprägungen $x_1, \ldots, x_n$ und dem Label y. Ein Datenpunkt hat die Form $x = (x_1, \ldots, x_n, y)$. Dann gilt:

$$P(y|(x_1, \ldots, x_n)) = \frac{P((x_1, \ldots, x_n)|y) \cdot P(y)}{P((x_1, \ldots, x_n))} \ , \tag{3.8}$$

wobei weiter gilt:

- $P(y)$ bezeichnet die sogenannte *A-priori*-Wahrscheinlichkeit. Das ist die Wahrscheinlichkeit, dass ein beliebiger Datenpunkt der Klasse y angehört.
- $P((x_1, \ldots, x_n))$ bezeichnet die Wahrscheinlichkeit für das Auftreten eines Datenpunktes mit der Attributbelegung $x_1, \ldots, x_n$.
- $P((x_1, \ldots, x_n)|y)$ bezeichnet die Wahrscheinlichkeit für das Auftreten eines Datenpunktes mit der Attributbelegung $x_1, \ldots, x_n$ unter der Voraussetzung, dass dieser Datenpunkt der Klasse y angehört. Da alle Attribute stochastisch unabhängig sind, lässt sich die Wahrscheinlichkeit durch die Einzelwahrscheinlichkeiten der Attribute folgendermaßen berechnen: $P((x_1, \ldots, x_n)|y) = \prod_{k=1}^{n} P(x_k|y)$.
- $P(y|(x_1, \ldots, x_n))$ bezeichnet die sogenannte *A-posteriori*-Wahrscheinlichkeit, nach der gesucht wird. Das ist die Wahrscheinlichkeit, dass ein Datenpunkt der Klasse y angehört, für den bekannt ist, dass dieser die Attributausprägungen $x_1, \ldots, x_n$ besitzt.

Beim naiven Bayes-Klassifikator wird die A-posteriori-Wahrscheinlichkeit für alle vorhandenen Klassen bestimmt. Anschließend wird genau die Klasse hervorgesagt, dessen A-posteriori-Wahrscheinlichkeit am höchsten ist. Sei m die Anzahl verschiedener Klassen und $y_1, \ldots, y_m$ die einzelnen Klassen. Dann wird genau das y_i als Klasse für den Datenpunkt $(x_1, \ldots, x_n)$ bestimmt, für das gilt:

$$P(y_i|(x_1, \ldots, x_n)) > P(y_j|(x_1, \ldots, x_n)) \text{ für alle } 1 \leq j \leq m \text{ und } i \neq j \ .$$

Da für alle zu vergleichenden Werte der Nenner aus Gleichung 3.8 gleich ist, kann dieser bei der Bestimmung des Maximums ignoriert werden und

es genügt, $\prod_{k=1}^{n} P(x_k|y_j) \cdot P(y_j)$ für alle $1 \leq j \leq m$ zu bestimmen. Diese Wahrscheinlichkeit $P(x_k|y_j)$ kann abhängig vom Attributtyp berechnet werden:

Nichtnumerische Attribute

$P(x_k|y_j)$ kann berechnet werden, indem der Anteil der Datenpunkte in T mit Klasse y_j und Attributausprägung x_k von allen Datenpunkten in T mit Klasse y_i bestimmt wird. Um Wahrscheinlichkeiten von genau 0 zu vermeiden, kann das sogenannte *Laplace-Smoothing* eingesetzt werden, durch das alle Attributausprägungen berücksichtigt werden können, auch wenn sie im begrenzten Trainingsdatensatz nicht auftreten. Die Glättung erfolgt, indem bei der o.g. Berechnung des Anteils der Datenpunkte eine Konstante im Zähler hinzuaddiert wird und im Nenner die Anzahl der möglichen Ausprägungen dieses Attributs hinzuaddiert wird. Typischerweise wird die Konstante im Zähler mit 1 belegt. Dadurch werden nicht auftretende Attributausprägungen zwar berücksichtigt, der Einfluss auf die Vorhersage ist aber im Vergleich zu den tatsächlich vorhandenen Daten gering. Es sei allerdings darauf hingewiesen, dass es noch weitere Glättungsmethoden gibt, die an dieser Stelle nicht vertieft werden. Eine mögliche Anwendung des naiven Bayes-Klassifikators für nominale Attribute ist in Beispiel 3.2 dargestellt. Das Laplace-Smoothing wird in diesem Beispiel allerdings nicht verwendet, sondern lediglich die relativen Häufigkeiten.

Numerische Attribute

Für die Berechnung der Wahrscheinlichkeit $P(x_k|y_j)$ für numerische Attribute gibt es verschiedene Möglichkeiten, von denen an dieser Stelle lediglich der *gauß'sche* Ansatz vorgestellt wird (gauß'scher naiver Bayes).

Für die Berechnung werden zunächst die Parameter μ_{y_j} (Erwartungswert) und σ_{y_j} (Standardabweichung) der jeweiligen Klasse bestimmt. Mit Hilfe dieser Werte kann anschließend die Wahrscheinlichkeit berechnet werden:

$$P(x_k|y_j) = \frac{1}{\sqrt{2\pi}\sigma_{y_j}} \cdot \exp\left(-\frac{(x_k - \mu_{y_j})^2}{2\sigma_{y_j}^2}\right) \tag{3.9}$$

Beispiel 3.2. Seien folgende Attribute mit verschiedenen Attributausprägungen gegeben:

- Abflugtag ∈ {Montag, Samstag},
- Buchungstag ∈ {Dienstag, Sonntag},
- Tage bis Abflug ∈ {0-7, 8-30, >30},
- Preisklasse ∈ {günstig, mittel, teuer}.

Sei weiter der in Tabelle 3.2 dargestellte gelabelte Datensatz und ein neuer zu klassifizierender Datenpunkt x_{neu} = *(Montag, Dienstag, 7-30, teuer)* gegeben, für den eine Vorhersage getroffen werden soll. Dann können die Wahrscheinlichkeiten direkt bestimmt werden:

$P(Buchen = ja) = \frac{7}{12}$,

$P(Buchen = nein) = \frac{5}{12}$,

$P(Abflugtag = Montag|Buchen = ja) = \frac{3}{7}$,

$P(Buchungstag = Dienstag|Buchen = ja) = \frac{5}{7}$,

$P(Tage\ bis\ Abflug = 8 - 30|Buchen = ja) = \frac{3}{7}$,

$P(Preisklasse = teuer|Buchen = ja) = \frac{2}{7}$,

$P(Abflugtag = Montag|Buchen = nein) = \frac{2}{5}$,

$P(Buchungstag = Dienstag|Buchen = nein) = \frac{1}{5}$,

$P(Tage\ bis\ Abflug = 8 - 30|Buchen = nein) = \frac{2}{5}$,

$P(Preisklasse = teuer|Buchen = nein) = \frac{2}{5}$.

Mit Hilfe dieser Wahrscheinlichkeiten lassen sich

$P(x_{neu}|Buchen = ja) = \frac{3}{7} \cdot \frac{5}{7} \cdot \frac{3}{7} \cdot \frac{2}{7} \approx 0,037$ und

$P(x_{neu}|Buchen = nein) = \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{2}{5} \cdot \frac{2}{5} \approx 0,013$

berechnen. Es folgt:

$P(x_{neu}|Buchen = ja) \cdot P(Buchen = ja) \approx 0,022$,

$P(x_{neu}|Buchen = nein) \cdot P(Buchen = nein) \approx 0,005$ und

$0,022 > 0,005$,

somit würde der neue Datenpunkt x_{neu} mit $Buchen = ja$ klassifiziert werden.

3.3 ENTSCHEIDUNGSBÄUME

Entscheidungsbäume können sowohl für Klassifikationsprobleme als auch für Regressionsprobleme eingesetzt werden. Bei einem Entscheidungsbaum handelt es sich um einen azyklischen Graphen, mit dessen Hilfe Vorhersagen getroffen werden können.

Ein Entscheidungsbaum (Graph) wird anhand von Trainingsdaten erzeugt und besteht aus einer Wurzel, mehreren Knoten, Kanten und Blattknoten. Ein vereinfachtes Beispiel ist in Abbildung 3.1 zu finden, bei dem für einen ausgewählten Flug mit einem aktuellen Preis abgefragt werden kann, ob gebucht werden soll oder nicht.

Ausgehend von der Wurzel wird in jedem Knoten ein Merkmal untersucht. Abhängig von dem Ergebnis wird einer der ausgehenden Kanten zum nächsten Knoten gefolgt. Handelt es sich bei dem nächsten Knoten um einen Blattknoten, kann ein Anfrageergebnis bzw. eine Klassenbezeichnung zurückgegeben werden.

ID	Abflugtag	Buchungstag	Tage bis Abflug	Preisklasse	Buchen
x_1	Montag	Dienstag	0-7	günstig	ja
x_2	Montag	Sonntag	7-30	günstig	ja
x_3	Samstag	Sonntag	>30	günstig	ja
x_4	Montag	Dienstag	0-7	mittel	nein
x_5	Samstag	Dienstag	0-7	mittel	ja
x_6	Montag	Dienstag	7-30	mittel	ja
x_7	Samstag	Sonntag	7-30	mittel	nein
x_8	Samstag	Dienstag	7-30	teuer	ja
x_9	Samstag	Sonntag	7-30	teuer	nein
x_{10}	Montag	Sonntag	>30	mittel	nein
x_{11}	Samstag	Dienstag	0-7	teuer	ja
x_{12}	Samstag	Sonntag	0-7	teuer	nein

Tabelle 3.2: Beispielhafter (fiktiver) Datensatz eines Fluges. Eigene Darstellung.

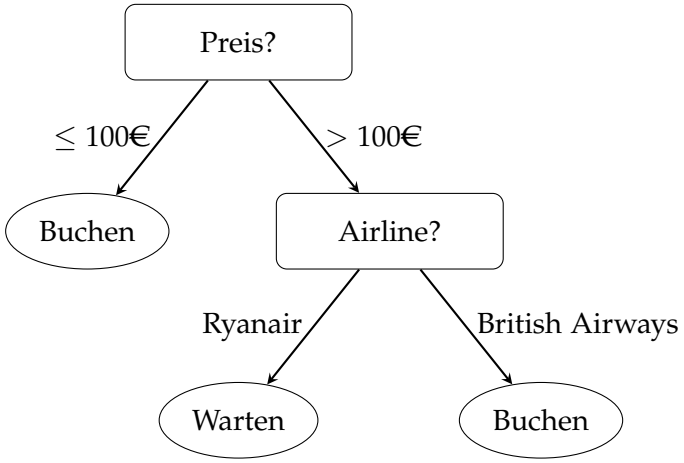

Abbildung 3.1: Entscheidungsbaum für ein vereinfachtes Beispiel. Eigene Darstellung.

Beispiel 3.3. Ein fiktiver Flug der Airline Ryanair kostet zum aktuellen Zeitpunkt 150 Euro. Mit Hilfe von Abbildung 3.1 kann für diesen Flug entschieden werden, dass der Flug nicht gebucht werden sollte. Zunächst wird ausgehend vom Wurzelknoten der rechten Kante gefolgt und anschließend der linken Kante.

Für die Erzeugung eines Entscheidungsbaumes wird ein Maß benötigt, um entscheiden zu können, welches Attribut in welchem Knoten untersucht werden soll. Je nach Art des Baumes gibt es unterschiedliche Algorithmen, die einen Entscheidungsbaum erzeugen und hierfür verschiedene Maße verwenden. Im Folgenden wird der Algorithmus *CART* für binäre Entscheidungsbäume und der Algorithmus *ID3* für Entscheidungsbäume, deren Knoten auch mehr als zwei Kinder haben können, vorgestellt.

3.3.1 *CART-Algorithmus*

Der *CART*-Algorithmus (Classification and Regression Tree) erzeugt einen binären Entscheidungsbaum, indem die Menge der Trainingsdaten mit Hilfe eines Spaltkriteriums rekursiv in zwei disjunkte Teilmengen geteilt wird. Zu Beginn existiert ein Trainingsdatensatz T. Für diesen, und auch für die erzeugten Teilmengen, wird folgendermaßen rekursiv vorgegangen:

1. Gehören alle Trainingsbeispiele derselben Kategorie an, wird ein Blattknoten mit genau dieser Kategorie erzeugt.
2. Gehören die Trainingsbeispiele verschiedenen Kategorien an, ohne, dass es weitere Merkmale gibt, die diese Beispiele unterscheiden, wird die am häufigsten vorkommende Kategorie als Blattknoten erzeugt.
3. Gehören die Trainingsbeispiele verschiedenen Kategorien an und gibt es weitere Merkmale, die die Beispiele unterscheiden, dann wird ein Merkmal als Spaltkriterium bestimmt, das nach Spaltung der Beispiele die reinsten Untermengen erzeugt. Eine Untermenge heißt *rein*, wenn alle in der Menge enthaltenen Datenpunkte der gleichen Kategorie angehören.

Um die Unreinheit einer Menge zu bestimmen, wird der *Gini-Index* (auch *Gini-Unreinheit*) berechnet. Der Gini-Index $gini(T)$ kann für die Menge T mit folgender Formel berechnet werden, wobei k die Kategorie, n die Anzahl verschiedener Kategorien und p_k der Anteil von Instanzen der Kategorie k aller Datenpunkte aus T ist:

$$gini(T) = 1 - \sum_{k=1}^{n} p_k^2 \,. \tag{3.10}$$

Beträgt der Gini-Index eines Knotens 0, dann besteht dieser Knoten aus Datenpunkten derselben Kategorie und gilt als *rein*. Dieser Knoten kann als Blattknoten gekennzeichnet werden, da bereits eine Kategorie als Ergebnis ausgegeben werden kann.

Beispiel 3.4. Angenommen, das Attribut *Abflugtag* teilt den in Tabelle 3.3 dargestellten Datensatz in die disjunkten Teilmengen $A_1 = \{x_1, x_2\}$ und $A_2 = \{x_3, x_4, x_5\}$. Dann kann der Gini-Index für die beiden Mengen berechnet werden:

$$gini(A_1) = 1 - (\tfrac{2}{2})^2 = 0 \,,$$

$$gini(A_2) = 1 - ((\tfrac{2}{3})^2 + (\tfrac{1}{3})^2) \approx 0,44 \,.$$

Um das Merkmal auszuwählen, das die Menge der Datenpunkte bestmöglich aufteilt, sodass der Gini-Index für die Kindknoten minimal wird, enthält der CART-Algorithmus eine Kostenfunktion $gini_A(T)$, die minimiert wird.

ID	Abflugtag	Abflugzeit	Buchungstag	Preis	Buchen
x_1	Montag	10:00	Dienstag	100	ja
x_2	Montag	10:00	Sonntag	200	ja
x_3	Samstag	18:00	Sonntag	200	nein
x_4	Samstag	18:00	Sonntag	100	nein
x_5	Samstag	18:00	Dienstag	200	ja

Tabelle 3.3: Beispielhafter (fiktiver) Datensatz eines Fluges mit jeweils zwei Ausprägungen verschiedener Attribute. Eigene Darstellung.

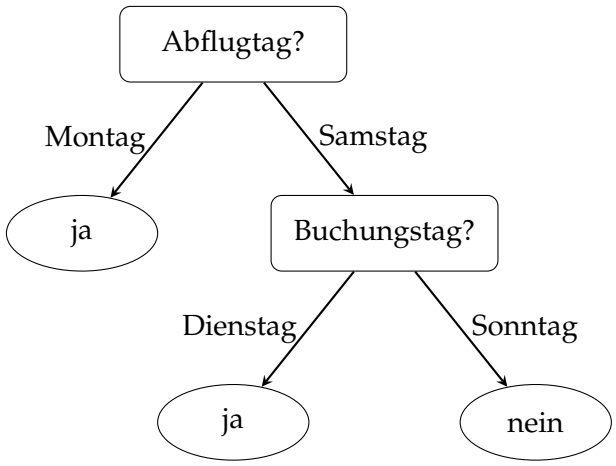

Abbildung 3.2: Entscheidungsbaum zu Beispiel 3.5. Eigene Darstellung.

Es bezeichne T die Trainingsdaten und A das Attribut/Merkmal, das die Datenpunkte in disjunkte Teilmengen (Kindknoten) A_1 und A_2 teilt. Dann lässt sich die zu minimierende Kostenfunktion wie folgt darstellen:

$$gini_A(T) = \frac{|A_1|}{|T|} gini(A_1) + \frac{|A_2|}{|T|} gini(A_2) \; . \tag{3.11}$$

Beispiel 3.5. Sei T der in Tabelle 3.3 dargestellte Datensatz. Es soll nun ein Merkmal bestimmt werden, das die o.g. Kostenfunktion minimiert. Als Beispiel wird der Wert für das Attribut *Abflugtag* berechnet, für das in Beispiel 3.4 bereits die Gini-Indexe für die Teilmengen A_1 und A_2 bestimmt wurden:

$$gini_{Abflugtag}(T) = \tfrac{2}{5} \cdot gini(A_1) + \tfrac{3}{5} \cdot gini(A_2) = \tfrac{2}{5} \cdot 0 + \tfrac{3}{5} \cdot 0,44 = 0,264 \; .$$

Angenommen, das Merkmal *Abflugtag* erzeuge die reinsten Teilmengen, dann wird dieses Attribut als Wurzelknoten erzeugt. Anschließend wird auf gleiche Weise mit den daraus entstehenden Teilmengen A_1 und A_2 rekursiv fortgefahren. Der daraus resultierende Entscheidungsbaum ist in Abbildung 3.2 dargestellt.

3.3.2 *ID3-Algorithmus*

Der *ID3*-Algorithmus (Iterative Dichotomiser 3) erzeugt einen nicht-binären Entscheidungsbaum und ähnelt vom Aufbau dem CART-Algorithmus (vgl. Unterabschnitt 3.3.1). Es wird ebenfalls das Attribut gesucht, das die Trainingsdaten bestmöglich spaltet. Allerdings wird beim ID3-Algorithmus anstelle des Gini-Indexes der Informationsgehalt verwendet, um das passende Attribut zu bestimmen.

Der Informationsgehalt, auch Entropie genannt, ist, genau wie der Gini-Index, ein Maß für die Unreinheit einer Datenmenge und gibt an, wie gut die Datenpunkte anhand ihrer Kategorie gespalten werden können. Bei einem Informationsgehalt von 1 ist die Wahrscheinlichkeit des Auftretens der verschiedenen Kategorien gleich hoch.

Sei ein Trainingsdatensatz T gegeben und es gebe k verschiedene Kategorien. Seien $T_1, \ldots, T_k$ die disjunkten Teilmengen von T, die der Kategorie k zugeordnet werden können, dann bezeichnet p_{T_i} für $1 \leq i \leq k$ die Wahrscheinlichkeit des Auftretens der jeweiligen Kategorie. Die Entropie H lässt sich wie folgt berechnen:

$$H(T) = -\sum_{i=1}^{k} p_{T_i} \cdot \log_2 p_{T_i} = -\sum_{i=1}^{k} \frac{|T_i|}{|T|} \cdot \log_2 \frac{|T_i|}{|T|} \,. \tag{3.12}$$

Beispiel 3.6. Die in Tabelle 3.3 dargestellten Datenpunkte werden um zwei weitere Zeilen ergänzt, wodurch sich der in Tabelle 3.4 dargestellte Datensatz ergibt. Es gilt $k = 2$, da es zwei Kategorien gibt. Die Datenpunkte lassen sich anhand der Kategorie in die Teilmengen $T_1 = \{x_1, x_2, x_5, x_7\}$ sowie $T_2 = \{x_3, x_4, x_6\}$ spalten. Die Gesamt-Entropie für T lässt sich wie folgt berechnen:

$$H(T) = -(\tfrac{4}{7} \cdot \log_2 \tfrac{4}{7} + \tfrac{3}{7} \cdot \log_2 \tfrac{3}{7}) \approx 0,985 \,.$$

ID	Abflugtag	Abflugzeit	Buchungstag	Preis	Buchen
x_1	Montag	10:00	Dienstag	100	ja
x_2	Montag	10:00	Sonntag	200	ja
x_3	Samstag	18:00	Sonntag	200	nein
x_4	Samstag	18:00	Sonntag	100	nein
x_5	Samstag	18:00	Dienstag	200	ja
x_6	Sonntag	18:00	Freitag	300	nein
x_7	Sonntag	18:00	Dienstag	200	ja

Tabelle 3.4: Erweiterter (fiktiver) Datensatz eines Fluges mit mehr als zwei Ausprägungen der jeweiligen Attribute. Eigene Darstellung.

Die Entropie lässt sich nicht nur für den gesamten Datensatz T berechnen, sondern auch für dessen Teilmengen. Sei A ein Attribut mit v verschiedenen Ausprägungen, das den Datensatz T in die disjunkten Teilmengen

$A_1, \ldots, A_v$ teilt, wobei $\bigcup_{i=1}^{v} A_i = T$ gilt. Dann lässt sich die Entropie analog Gleichung 3.12 berechnen, wobei k weiterhin die Anzahl verschiedener Kategorien/Klassen bezeichnet und nicht mit den v Attributausprägungen verwechselt werden darf:

$$H(A_i) = -\sum_{j=1}^{k} \frac{|A_{i_j}|}{|A_i|} \cdot \log_2 \frac{|A_{i_j}|}{|A_i|} \,. \tag{3.13}$$

Für den Fall, dass nach einer Spaltung keine Trainingsbeispiele einer bestimmten Kategorie mehr existieren, beträgt $A_{i_j} = 0$. Da aber $\log_2 0$ nicht definiert ist, wird dieser an dieser Stelle wie folgt definiert: $0 \cdot \log_2 0 := 0$.

Beispiel 3.7. Fortsetzung zu Beispiel 3.6: Als Attribut wird der *Abflugtag* mit den Ausprägungen *Montag, Samstag, Sonntag* gewählt. Der Datensatz T lässt sich nun in die drei Teilmengen spalten:

$$H(Abflugtag_{Montag}) = -(\tfrac{2}{2} \cdot \log_2 \tfrac{2}{2} + \tfrac{0}{2} \cdot \log_2 \tfrac{0}{2}) = 0 \,,$$

$$H(Abflugtag_{Samstag}) = -(\tfrac{1}{3} \cdot \log_2 \tfrac{1}{3} + \tfrac{2}{3} \cdot \log_2 \tfrac{2}{3}) \approx 0,918 \,,$$

$$H(Abflugtag_{Sonntag}) = -(\tfrac{1}{2} \cdot \log_2 \tfrac{1}{2} + \tfrac{1}{2} \cdot \log_2 \tfrac{1}{2}) = 1.$$

Um nun das passende Attribut für den Knoten des Entscheidungsbaums bestimmen zu können, wird der Informationsgewinn eingeführt. Der Informationsgewinn eines Attributes A bezeichnet die Differenz aus der Gesamt-Entropie (Gleichung 3.12) des Datensatzes T und der Summe der gewichteten Entropien nach Spaltung des Datensatzes mit Hilfe des Attributs A (Gleichung 3.13) in v disjunkte Mengen $A_1, \ldots, A_v$:

$$IG(T, A) = H(T) - \sum_{i=1}^{v} \frac{|A_i|}{|T|} \cdot H(A_i) \,. \tag{3.14}$$

Beispiel 3.8. Fortsetzung zu Beispiel 3.7: Der Informationsgewinn für das Attribut *Abflugtag* lässt sich mit Hilfe der soeben durchgeführten Berechnungen leicht bestimmen:

$$\begin{aligned} IG(T, Abflugtag) = H(T) - (&\tfrac{|Abflugtag_{Montag}|}{|T|} \cdot H(Abflugtag_{Montag}) + \\ &\tfrac{|Abflugtag_{Samstag}|}{|T|} \cdot H(Abflugtag_{Samstag}) + \tfrac{|Abflugtag_{Sonntag}|}{|T|} \cdot \\ &H(Abflugtag_{Sonntag})) = 0,985 - (\tfrac{2}{7} \cdot 0 + \tfrac{3}{7} \cdot 0,918 + \tfrac{2}{7} \cdot 1) \approx 0,306 \,. \end{aligned}$$

Nachdem der Informationsgewinn für jedes Attribut bestimmt wurde, wird genau das Attribut als Knoten des Entscheidungsbaumes verwendet, das den größten Informationsgewinn besitzt. Anschließend wird analog mit den sich ergebenden Teilmengen vorgegangen, bis der Entscheidungsbaum vollständig ist.

3.4 RANDOM FOREST

Der Random Forest ist ein sehr mächtiger Machine-Learning-Algorithmus, der, wie auch der Entscheidungsbaum, für Klassifikations- und Regressionsprobleme angewendet werden kann. Der Algorithmus basiert auf dem sogenannten *Ensemble-Learning* und besteht aus vielen einzelnen Entscheidungsbäumen.

Das Ensemble-Learning ist eine Lernmethode und bezeichnet die Aneinanderreihung mehrerer Machine-Learning-Modelle (Klassifikatoren oder Regressoren), die auf verschiedenen Daten trainiert wurden und gemeinsam eine Vorhersage treffen sollen. Diese Vorhersage des Ensembles hat oft eine höhere Genauigkeit als die einzelnen Modelle selbst. Diese treffen zunächst ihre Vorhersage, woraus anschließend eine aggregierte Vorhersage des Ensembles bestimmt wird, indem beispielsweise das am häufigsten vorkommende Ergebnis zur Vorhersage herangezogen wird. Bei den Modellen kann es sich um unterschiedliche Algorithmen handeln, beispielsweise eine Zusammensetzung von Entscheidungsbaum, Naiver Bayes-Klassifikator und k-nächste-Nachbarn-Klassifikator, aber es können auch gleiche Machine-Learning-Algorithmen verwendet werden, die auf unterschiedlichen Daten trainiert wurden, beispielsweise mehrere Entscheidungsbäume.

Genau dies beschreibt den Random Forest. Dieser erzeugt einen Wald aus vielen Entscheidungsbäumen, die gemeinsam eine Vorhersage treffen. Ziel der Random Forests ist es, eine bei Entscheidungsbäumen übliche Überanpassung zu vermeiden.

Der Random Forest wird auf Grundlage der Bagging-Methode trainiert, bei der als Trainingsdaten eine zufällige Teilmenge der gesamten Daten ausgewählt wird, um das Modell zu trainieren. Beim Bagging werden diese Teilmengen mit Zurücklegen gebildet, das heißt, einzelne Datenpunkte werden für das Training verschiedener Entscheidungsbäume verwendet.

Zudem wird die sogenannte *Feature-Randomness* eingesetzt: Bei der Erzeugung eines Entscheidungsbaums wird nicht wie in Abschnitt 3.3 dargestellt das Beste aller Merkmale als Spaltkriterium bestimmt, sondern es wird eine zufällige Auswahl einer Teilmenge aller Merkmale erzeugt, aus der schließlich das beste Merkmal ausgewählt wird. Dies führt zu einer höheren Diversität der einzelnen Bäume und einer geringen Korrelation zwischen den Entscheidungsbäumen.

3.5 k-NEAREST NEIGHBORS (k-NÄCHSTE-NACHBARN-KLASSFIKATION)

Der Algorithmus k-Nearest Neighbors (KNN) kann für Klassifikationsprobleme eingesetzt werden, lässt sich aber auch für Regressionsprobleme anwenden. Das Grundprinzip des KNN besagt, dass sich ähnliche Punkte nah beieinander befinden. Sei $k \in \mathbb{N}^+$. Für ein neu zu klassifizierendes Objekt wird nach den k nächsten Nachbarn dieses Datenpunktes gesucht, um eine Klasse oder einen Wert vorherzusagen. Diese Nachbarn sind Datenpunkte der Trainingsdaten. Für Klassifikationsprobleme wird anschließend die am

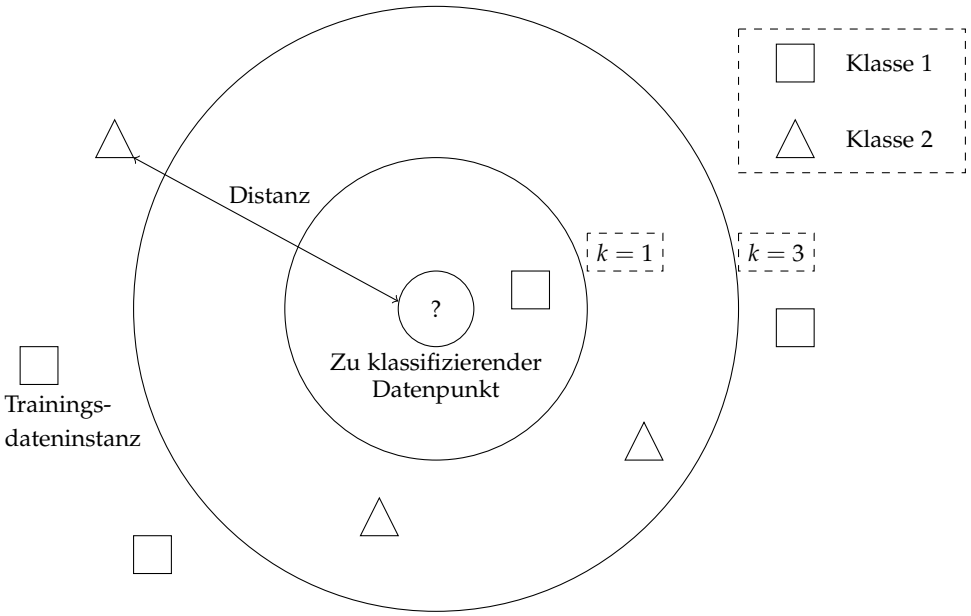

Abbildung 3.3: Darstellung des KNN-Algorithmus anhand eines Klassifikationsproblems mit $k = 1$ und $k = 2$. Nachempfunden von [Sri21].

häufigsten vorkommende Klasse der Nachbarn als Vorhersage verwendet. Für Regressionsprobleme wird zur Vorhersage der Mittelwert der Zielgröße dieser Nachbarn gebildet.

Um die k nächsten Nachbarn auswählen zu können, bedarf es zunächst eines Abstandsmaßes, um die k nächsten Nachbarn mit dem geringsten Abstand zum neuen Datenpunkt auswählen zu können. Geeignete Abstandsmaße wurden bereits in Abschnitt 2.4 vorgestellt. Zudem wird dem Algorithmus ein Wert für k übergeben, der die Anzahl der Nachbarn bestimmt. Dieser muss abhängig vom jeweiligen Datensatz bestmöglich bestimmt werden, um eine Über- bzw. Unteranpassung zu vermeiden.

Für Klassifikationsprobleme kann eine Klasse für den neuen Datenpunkt anhand einer *Mehrheitswahl* bestimmt werden. Es werden die k nächsten Nachbarn bestimmt und die unter den Nachbarn am häufigsten vorkommende Klasse wird als Klasse für den neuen Datenpunkt bestimmt. Es fällt auf, dass für k grundsätzlich eine ungerade Zahl gewählt werden sollte, um zu vermeiden, dass die Hälfte der Nachbarn der Klasse A und die andere Hälfte der Klasse B angehören. In diesem Fall liegt keine Mehrheit einer Klasse vor. In Abbildung 3.3 ist der KNN-Algorithmus dargestellt. Mit $k = 1$ wird für den neuen Datenpunkt die Klasse 1 bestimmt. Mit $k = 3$ hingegen wird dem neuen Datenpunkt die Klasse 2 zugeordnet.

3.6 SUPPORT VECTOR MACHINE (SVM)

Die Support Vector Machine (SVM) kann sowohl für Klassifikationsprobleme als auch für Regressionsprobleme eingesetzt werden. In dieser Arbeit wird lediglich die binäre Klassifikation dargestellt[2].

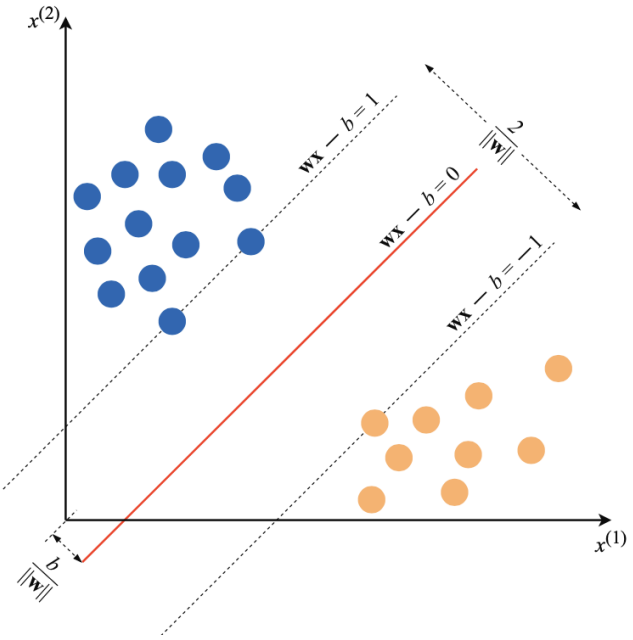

Abbildung 3.4: Zweidimensionales Beispiel für ein SVM-Modell. Entnommen aus [Bur19].

Ziel dieses Lernverfahrens ist es, eine Hyperebene zu bestimmen, die die Datenpunkte der beiden Klassen bestmöglich unterteilt bzw. voneinander abgrenzt. Die Besonderheit liegt darin, dass die Lücke bzw. der Abstand zwischen den Datenpunkten der beiden Klassen maximiert wird. Im zweidimensionalen Fall kann man sich dies als eine Gerade mit Rändern vorstellen, die den breitesten Abstand zwischen den Klassen darstellt. Eine beispielhafte Darstellung dieser Geraden ist in Abbildung 3.4 zu finden. Dieser Rand wird als *Margin* bezeichnet. Die Datenpunkte des Trainingsdatensatzes, die auf dem Rand liegen, bezeichnet man als *Stützpunkte (Support Vectors)*. Zu beachten ist, dass das Hinzufügen neuer Datenpunkte, die außerhalb des Randes liegen, keinen Einfluss auf die Hyperebene haben, da sich diese lediglich an den Stützpunkten orientiert.

Zu unterscheiden ist zwischen der *Hard-Margin-Klassifikation* (vgl. Unterabschnitt 3.6.1) und der *Soft-Margin-Klassifikation* (vgl. Unterabschnitt 3.6.2). Bei der Hard-Margin-Klassifikation werden die beiden Klassen exakt voneinander separiert. Dies ist allerdings anfällig für Ausreißer, da ein einziger Ausreißer die gesamte Hyperebene beeinflussen kann, wodurch es zu ungenauen Vorhersagen kommen kann.

2 Die Ausweitung des Algorithmus auf mehrere Klassen lässt sich durch ein Ensemble mehrerer SVM's allerdings leicht umsetzen.

Bei der Soft-Margin-Klassifikation hingegen wird ein gewisser Anteil sogenannter *Margin-Verletzungen* zugelassen, wodurch es vorkommen kann, dass sich Datenpunkte auch innerhalb der Ränder oder im Bereich der anderen Klasse befinden. Auf diese Weise kann einem möglichen Overfitting entgegengewirkt werden. Das Modell generalisiert besser.

Die Gleichung der Hyperebene lautet $wx_i - b = 0$, wobei x_i den Eingabevektor, b den Bias-Term und w den Vektor mit den Gewichten der einzelnen Merkmale beschreibt. Ziel ist es, w und b anhand von Trainingsdaten bestmöglich zu bestimmen.

3.6.1 *Hard-Margin-Klassifikation*

Sei T nun ein Trainingsdatensatz und $t_i \in T$ der $i-te$ Trainingsdatenpunkt aus T der Form (x_i, y_i), wobei x_i einen Eingabevektor mit verschiedenen Attributausprägungen darstellt. Sei y_i das Label einer binären Klassifikation, das die Werte $+1$ und -1 annehmen kann. Bezogen auf Abbildung 3.4 werden die blauen Punkte links der Entscheidungsgrenze somit der Klasse -1 und die gelben Punkte auf der rechten Seite der Entscheidungsgrenze der Klasse $+1$ zugeordnet.

Im Falle der Hard-Margin-Klassifikation, die in Abbildung 3.4 abgebildet ist, kann die Vorhersage für einen neuen zu klassifizierenden Eingabevektor x_{neu} wie folgt bestimmen:

$$\hat{y} = \begin{cases} -1, \text{ wenn } wx_{neu} - b \leq -1 \\ +1, \text{ wenn } wx_{neu} - b \geq 1 \end{cases} \tag{3.15}$$

Zudem müssen für eine SVM die folgenden beiden Bedingungen erfüllt sein:

$$wx_i - b \leq -1, \text{ für } y_i = -1 \qquad \text{und} \qquad wx_i - b \geq 1, \text{ für } y_i = +1 \, . \tag{3.16}$$

Weiter gilt:

$$y_i \cdot (wx_i - b) > 0 \, . \tag{3.17}$$

Dies ist offensichtlich, da für positive y_i des Trainingsdatensatzes auch $wx_i - b$ positiv sein muss und für negative y_i muss $wx_i - b$ negativ sein, weshalb das Produkt in jedem Fall positiv sein muss. Die Punkte, die am nächsten an der Hyperebene bzw. der Entscheidungsgrenze liegen (die Support Vectors), erfüllen die Gleichung:

$$|wx_i - b| = 1 \, . \tag{3.18}$$

Es folgt, dass Gleichung 3.17 noch weiter eingeschränkt werden kann. Es gilt nämlich

$$y_i \cdot (wx_i - b) \geq 1 \Leftrightarrow y_i \cdot (wx_i - b) - 1 \geq 0 \, . \tag{3.19}$$

Der Abstand zwischen den beiden Hyperebenen, die den Rand bilden und mit $wx_i - b = 1$ bzw. $wx_i - b = -1$ beschrieben werden können, beträgt $\frac{2}{||w||}$, wobei $||w||$ die euklidische Distanz (vgl. Abschnitt 3.5) von w beschreibt. Ziel ist es, den Abstand zwischen den Hyperebenen bzw. die Margin zu maximieren. Dies erfolgt durch Minimierung der euklidischen Distanz $||w||$, was äquivalent zur Minimierung von $\frac{1}{2}||w||^2$ ist.

Das zu lösende Optimierungsproblem lautet demnach:

$$\text{Minimiere } \frac{1}{2}||w||^2 \text{ bezüglich } y_i \cdot (wx_i - b) - 1 \geq 0 \text{ für } 1 \leq i \leq n, \quad (3.20)$$

wobei n die Anzahl der Trainingsdaten beschreibt.

3.6.2 *Soft-Margin-Klassifikation*

Die in Unterabschnitt 3.6.1 dargestellten Ausführungen bezogen sich auf einen Trainingsdatensatz, der sich linear separieren lässt. Da dies in der Realität nicht immer der Fall sein muss, kann die SVM auch mit nicht linear separierbaren Daten umgehen. Eine Möglichkeit ist die Soft-Margin-Klassifikation, die in diesem Unterkapitel beschrieben wird. Eine weitere Möglichkeit ist der Einsatz des sogenannten *Kernel-Tricks*, der in Unterabschnitt 3.6.3 dargestellt wird.

Die Soft-Margin-Klassifikation lässt sich einsetzen, sofern die Trainingsdaten lediglich aufgrund von Rauschen bzw. Ausreißern nicht linear separierbar sind. Um dennoch eine lineare Hyperebene als Entscheidungsgrenze bestimmen zu können, werden Margin-Verletzungen akzeptiert. Dies bedeutet, dass im Trainingsdatensatz Datenpunkte enthalten sein können, die nach Erzeugung einer Hyperebene falsch klassifiziert werden würden. Zusätzliches Ziel bei der Erzeugung der Hyperebene ist es somit, die Anzahl an Margin-Verletzungen zu minimieren, während der Abstand zwischen den Rändern der Hyperebene analog Unterabschnitt 3.6.1 maximiert werden soll.

Um dies umzusetzen bedarf es einer zu minimierenden Verlustfunktion, der sogenannten *Hinge-Verlustfunktion*:

$$max(0\ ,\ 1 - y_i \cdot (wx_i - b))\ . \quad (3.21)$$

Da auch bei der Soft-Margin-Klassifikation die Bedingungen der Gleichung 3.16 erfüllt sein müssen, ist offensichtlich, dass die Hinge-Verlustfunktion für einen Datenpunkt x_i genau dann 0 beträgt, wenn der Datenpunkt auf der richtigen Seite der Entscheidungsgrenze liegt, dieser somit korrekt klassifiziert wird. Sofern der Datenpunkt korrekt klassifiziert wird, beträgt $y_i \cdot (wx_i - b)$ mindestens 1 (vgl. Gleichung 3.19) und das Maximum demnach 0.

Sei C ein Hyperparameter, der den Kompromiss beschreibt, der zwischen einer Erhöhung der Margin und einer Verringerung von Margin-Verletzun-

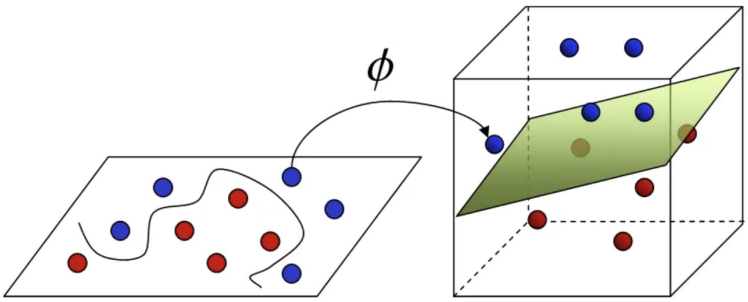

Abbildung 3.5: Kernel-Trick. Die linke Abbildung zeigt den ursprünglichen Raum, während die rechte Abbildung den transformierten Raum darstellt, in dem die Daten linear trennbar sind. Entnommen aus [Wil18].

gen eingegangen werden muss. Dann ist zur Erzeugung der Hyperebene die folgende Kostenfunktion zu minimieren:

$$C||w||^2 + \frac{1}{n}\sum_{i=1}^{n} max(0\ ,\ 1 - y_i \cdot (wx_i - b))\ . \tag{3.22}$$

Je höher der Wert für C festgelegt wird, desto mehr wird versucht, die Margin zu erhöhen. Bei einem niedrigen Wert für C wird versucht, wenig Margin-Verletzungen zuzulassen, was sich negativ auf die Breite der Ränder auswirken kann.

3.6.3 *Kernel-Trick*

Es gibt Daten, die in ihrem ursprünglichen Raum nicht linear separierbar sind, in einem Raum höherer Dimensionalität hingegen schon. Der Vektorraum kann in einen höherdimensionalen Raum überführt werden, in dem die Trainingsdaten anschließend linear separierbar sind und sich eine Hyperebene bestimmen lässt. Die Überführung erfolgt mit einer Abbildung $\Phi : x \mapsto \Phi(x)$. Diese Überführung ist zur Verdeutlichung in Abbildung 3.5 dargestellt.

Da es allerdings zu einem hohen Rechenaufwand führen würde, den gesamten Trainingsdatensatz in eine höhere Dimension abzubilden, können Kernel-Funktionen in höherdimensionalen Räumen eingesetzt werden, ohne die Transformation des Trainingsdatensatzes explizit ausführen zu müssen.

Das in Gleichung 3.20 dargestellte Optimierungsproblem lässt sich mit Hilfe verschiedener mathematischer Verfahren lösen, auf die in dieser Arbeit nicht näher eingegangen wird. Um aber die Funktionsweise des Kernel-Tricks verdeutlichen zu können, sei erwähnt, dass zur Lösung das Skalarprodukt $x_i x_j$ zweier verschiedener Punkte benötigt wird. Mit Hilfe des Kernel-Tricks kann nun das Skalarprodukt $x_i x_j$ des höherdimensionalen Raumes durch eine Kernel-Funktion bestimmt werden, ohne die Punkte x_i und x_j selbst in den höherdimensionalen Raum abbilden zu müssen.

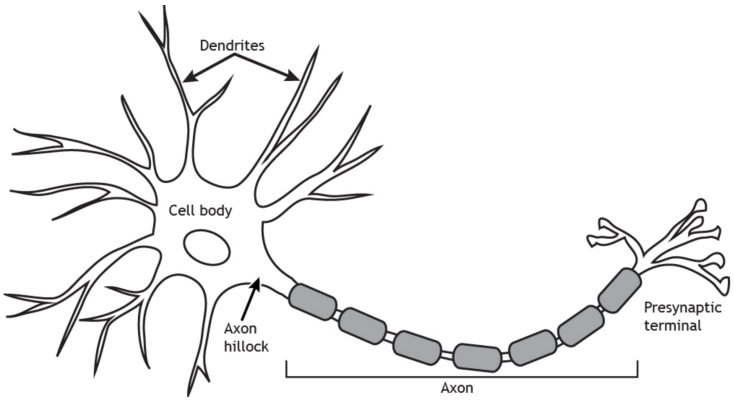

Abbildung 3.6: Das Neuron. Entnommen aus [Hen21].

Die am häufigsten verwendete Kernel-Funktion ist der RBF-Kernel (Radial Basis Functions Kernel), der das Skalarprodukt von x_i und x_j des höherdimensionalen Raumes wie folgt bestimmt:

$$k(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right), \tag{3.23}$$

wobei es sich bei σ um einen Hyperparameter des Kernels handelt, auf den an dieser Stelle nicht näher eingegangen werden soll. Mit Hilfe der auf diese Weise berechneten Werte lässt sich nun das Optimierungsproblem im höherdimensionalen Raum lösen.

3.7 MULTILAYER PERCEPTRON (MLP)

Beim Multilayer Perceptron (MLP) handelt es sich um einen überwachten Lernalgorithmus, der sich sowohl für Regressions- als auch für Klassifikationsprobleme eignet. Es handelt sich um ein künstliches neuronales Netzwerk, das dem menschlichen Gehirn nachempfunden wurde. Aus diesem Grund wird nachfolgend zunächst der biologische Hintergrund sehr allgemein dargestellt. Anschließend wird ein Perzeptron genauer dargestellt und zuletzt das Multilayer Perceptron, zu deutsch *mehrlagiges Perzeptron*.

3.7.1 *Biologische Neuronen*

Die elementare Einheit des menschlichen Nervensystems ist die Nervenzelle, auch *Neuron* genannt. Das menschliche Gehirn besteht aus etwa 10^{11} Neuronen [Alp22], wobei diese im Wesentlichen aus Zellkörper, Dendriten und Axon bestehen (vgl. Abbildung 3.6). Dendriten sind Verzweigungen, über die ein Neuron elektrische Signale von anderen Neuronen oder von sensorischen Nervenzellen[3] aufnehmen kann. Das Axon besteht ebenfalls aus Ver-

3 Dies sind Nervenzellen, die beispielsweise Informationen von Sinnesorganen an das Gehirn weiterleiten.

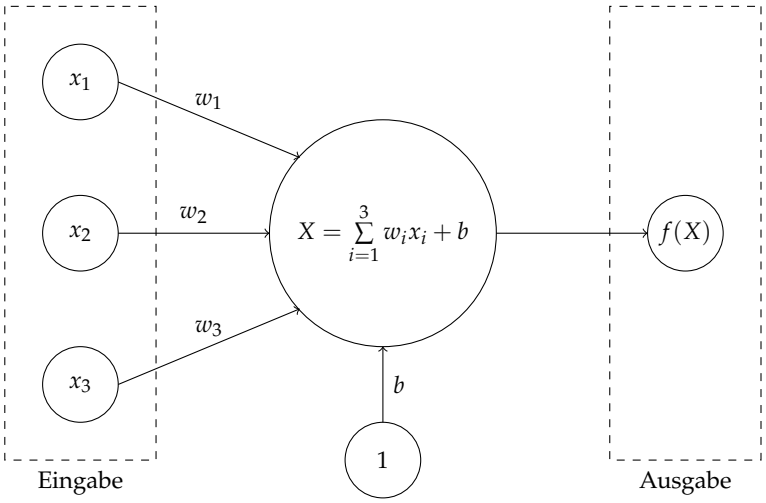

Abbildung 3.7: Darstellung eines Perzeptrons mit drei Eingaben und einer Ausgabe. Eigene Darstellung.

zweigungen, über die das Neuron Informationen mittels elektrischer Signale an andere Neuronen weitergeben kann. Ein Neuron ist mit etwa 10^4 anderen Neuronen verbunden [Alp22], wobei die Kontaktstelle zwischen zwei Neuronen *Synapse* genannt wird.

Die Funktionsweise kann stark vereinfacht wie folgt dargestellt werden: Ein Neuron empfängt elektrische Impulse über die Dendriten. Wird ein bestimmter Schwellwert erreicht, kann ein sogenanntes *Aktionspotential* ausgelöst werden. Ist dies der Fall, „feuert" das Neuron und gibt einen elektrischen Impuls an nachfolgende Neuronen weiter.

3.7.2 *Das Perzeptron*

Das Perzeptron ist ein künstliches Neuron, das in der Funktionsweise dem biologischen Neuron nachempfunden ist. Es nimmt Eingaben aus der Umgebung oder von anderen Perzeptronen auf, verarbeitet diese mit Hilfe einer Schwellwertfunktion (auch Aktivierungsfunktion genannt) und gibt je nach Ergebnis eine Ausgabe an alle nachfolgenden Perzeptronen.

Der Abbildung 3.7 kann die soeben beschriebene Funktionsweise entnommen werden. Das Perzeptron erhält einen gewichteten Eingabevektor, in diesem Fall die drei Werte x_1, x_2 und $x_3 \in \mathbb{R}$ mit den sogenannten *synaptischen Gewichten* w_1, w_2 und $w_3 \in \mathbb{R}$. Es wird zunächst die Summe des gewichteten Eingabevektors gebildet und ein Bias[4] $b \in \mathbb{R}$ hinzuaddiert. Abhängig von dem Ergebnis X und der Schwellwertfunktion $f(X)$ wird die Ausgabe berechnet. Eine Schwellwertfunktion ist beispielsweise die *Heaviside-Funktion*, die für den Fall $X > 0$ den Wert 1 zurückgibt und ansonsten den Wert 0.

4 Der Bias (auch Verzerrungseinheit genannt) ist nötigt, um das Modell zu verallgemeinern.

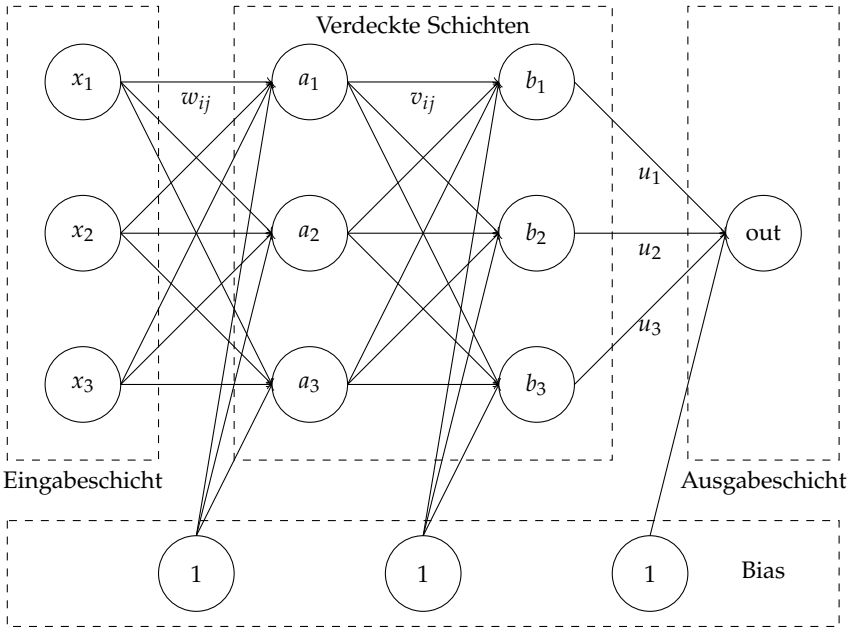

Abbildung 3.8: Darstellung mehrlagiger Perzeptronen mit drei Eingaben, zwei verdeckten Schichten und einer Ausgabe. Eigene Darstellung.

Bei Regressionsproblemen wird an dieser Stelle statt der Schwellwertfunktion die identische Abbildung ausgegeben [Sci23c], die genau der gewichteten Summe $X = \sum_{i=1}^{n} w_i x_i + b$ mit n Eingabewerten entspricht.

3.7.3 *Mehrlagige Perzeptronen*

Das Perzeptron lässt sich lediglich einsetzen, sofern ein linearer Zusammenhang zwischen den Eingabewerten und dem Ausgabewert besteht. Für nichtlineare Zusammenhänge eignet sich hingegen das MLP, bei dem statt eines Perzeptrons mehrere Perzeptronen in verschiedenen (verdeckten) Schichten verwendet werden.

Wie der Abbildung 3.8 entnommen werden kann, erhält jedes Perzeptron der ersten verdeckten Schicht die Eingabeparameter x_1, x_2 und $x_3 \in \mathbb{R}$, die auch in diesem Fall gewichtet sind. Die Gewichte wurden zur besseren Übersicht nicht vollständig dargestellt, jede Verbindung besitzt aber ein eigenes Gewicht: $w_{i,j} \in \mathbb{R}$ bezeichnet das Gewicht für die Verbindung von x_i nach a_j und $v_{ij} \in \mathbb{R}$ bezeichnet das Gewicht der Verbindung von a_i nach b_j. Zudem besitzt auch der Bias eigene Gewichte, die hier nicht dargestellt wurden.

Jedes Perzeptron der ersten verdeckten Schicht führt eine Berechnung durch und leitet das Ergebnis an alle Perzeptronen der nachgelagerten Schicht. Auch die Perzeptronen der zweiten verdeckten Schicht führen wieder Berechnungen durch und leiten das Ergebnis weiter. Zuletzt erhält schließlich die Ausgabeschicht als Eingabe die Ergebnisse der zweiten verdeckten Schicht und berechnet die Ausgabe. Da Informationen immer von

der Eingabeschicht über die verdeckten Schichten bis zur Ausgabeschicht weitergegeben werden, bezeichnet man diese Netze als *Feedforward-Netze*.

Die Perzeptronen der verdeckten Schichten berechnen zunächst die gewichtete Summe und wenden anschließend eine nichtlineare Funktion auf diese Summe an, beispielsweise die oft verwendete *Sigmoidfunktion*. Bezogen auf die erste verdeckte Schicht kann die Sigmoidfunktion folgendermaßen berechnet werden:

$$a_j = sigmoid\left(\sum_{i=1}^{n} w_{ij}x_i + b_j\right) = \frac{1}{1 + \exp\left(-\sum_{i=1}^{n} w_{ij}x_i + b_j\right)}, \tag{3.24}$$

wobei $1 \leq j \leq n$ gilt. Für das Beispiel in Abbildung 3.8 gilt $n = 3$.

Nach weiteren Verarbeitungsschritten berechnet schließlich das Perzeptron der Ausgabeschicht die Ausgabe. Im Falle der Regression wird an dieser Stelle lediglich die gewichtete Summe bestimmt und der Wert als Vorhersage ausgegeben [Sci23c]. Bei Klassifikationsproblemen existiert zusätzlich eine Aktivierungsfunktion, die die Ausgabe bestimmt.

3.7.4 *Backpropagation*

Das MLP lernt anhand von Trainingsdaten, indem die einzelnen Gewichte aller Verbindungen (in Abbildung 3.8 wären dies die Gewichte w_{ij}, v_{ij}, u_j sowie die nicht dargestellten Gewichte des jeweiligen Bias) so angepasst werden, dass der Fehler zwischen tatsächlichem und vorhergesagtem Wert möglichst gering ist. Der Lernprozess erfolgt durch *Backpropagation*, wobei immer mit einem *Mini-Batch*[5] gleichzeitig trainiert wird und der gesamte Datensatz mehrmals durchlaufen wird.

Im ersten Schritt werden alle Gewichte mit Zufallswerten besetzt. Als Zufallswerte eignen sich nach [Alp22] kleine Werte im Intervall $[-0,01; 0,01]$. Anschließend erfolgt ein Vorwärtsdurchlauf, bei dem jeweils die einzelnen Datenpunkte als Eingabe an die erste verdeckte Schicht übergeben und weiterverarbeitet werden, bis die Ausgabeschicht das Ergebnis bestimmt hat. Anhand einer Fehlerfunktion kann der Fehler zwischen tatsächlichem und berechnetem Wert bestimmt werden, wobei sich als Fehlerfunktion beispielsweise die mittlere quadratische Abweichung (vgl. Abschnitt 3.1, Gleichung 3.3) eignet. Diesen Wert der Fehlerfunktion gilt es zu minimieren, indem ein Rückwärtsdurchlauf von der Ausgabeschicht bis zur Eingabeschicht folgt, bei dem der jeweilige Beitrag zum Fehler für jede einzelne Verbindung bestimmt wird. Mit Hilfe des *Gradientenabstiegsverfahrens*, auf das in dieser Arbeit nicht näher eingegangen werden soll, werden alle Verbindungsgewichte im Netz angepasst. Es erfolgen mehrere Vorwärts- und Rückwärtsdurchläufe, bis der Wert der Fehlerfunktion minimal ist.

5 Eine geringe Anzahl an Datenpunkten des gesamten Datensatzes.

4 PRAKTISCHER TEIL: FLUGPREISVORHERSAGE

Im praktischen Teil dieser Arbeit soll ein Machine-Learning-Projekt umgesetzt werden, in dem alle in den vorherigen Kapiteln dargestellten Inhalte an einem praktischen Beispiel angewendet werden. Anhand eines Datensatzes mit historischen Flugpreisen sollen ausgewählte Machine-Learning-Algorithmen verwendet werden, um Modelle zu trainieren und anschließend Vorhersagen zu treffen. Bei den Vorhersagen wird unterschieden zwischen dem Regressionsproblem, bei dem ein genauer Flugpreis vorhersagt werden soll, und dem Klassifikationsproblem, bei dem eine Empfehlung (buchen / nicht buchen) vorhersagt werden soll.

Das Projekt wird in zwei *Jupyter Notebooks* ausführlich umgesetzt und jeder Schritt genau erläutert. Die Jupyter Notebooks sind Teil dieser Abschlussarbeit. Das erste Jupyter Notebook [Lew23a] stellt den Datensatz ausführlich vor und startet mit einer explorativen Datenanalyse, während es im zweiten Jupyter Notebook [Lew23b] um die Anwendung der Machine-Learning-Algorithmen gehen wird.

Um das Vorgehen strukturiert darstellen zu können, gliedert sich dieses Kapitel in die einzelnen Schritte des Vorgehensmodells *CRISP-DM*, das in Abschnitt 2.5 bereits erläutert wurde. Dabei werden diese sechs Schritte in den Unterkapiteln jeweils ausführlich dargestellt. In der vorliegenden schriftlichen Ausarbeitung werden lediglich Zusammenfassungen und wichtige Erkenntnisse dargestellt, weshalb für genauere Einblicke auf das genannte Projekt, bestehend aus den zwei Jupyter Notebooks, verwiesen sei.

4.1 BUSINESS UNDERSTANDING

4.1.1 *Problemstellung*

Fluggesellschaften verwenden schon seit vielen Jahren die Preisstrategie des *Dynamic Pricing* (vgl. Abschnitt 2.2). Abhängig von verschiedensten Faktoren legen Airlines stetig ihre Flugpreise fest, wodurch diese einer ständigen dynamischen Anpassung unterliegen. Dadurch schwanken die Flugpreise für ein und denselben Flug, auch wenn es sich um ähnliche Plätze in derselben Kabine handelt [Abd+21]. Da sich die Flugpreise teilweise mehrmals täglich ändern [Jos+20], ist es für Kunden sinnvoll, die Preise vor ihrer Buchung zu vergleichen und eine Buchung ggf. zu verzögern, wenn sie zu einem möglichst günstigen Preis buchen möchten. Es stellt sich die Frage, wie Machine Learning eingesetzt werden kann, um preisbewusste Kunden bei ihrer Entscheidung bestmöglich zu unterstützen.

4.1.2 *Hintergrundwissen*

Der verbreitete Grundsatz, dass eine weit im Voraus erfolgte Buchung am günstigsten sei, könne nach [GG11] heutzutage verworfen werden, da Airlines oft gegen diesen Grundsatz verstoßen, um ihren Profit zu steigern. Dementsprechend sei eine frühe Buchung nicht unbedingt sinnvoll [Abd+21; Jos+20]. Diese Aussage wird während der explorativen Datenanalyse in Unterabschnitt 4.2.3 untersucht.

Aufgrund der hohen Komplexität der Preisgestaltungs-Modelle ist es oft schwierig abzuschätzen, ob ein Flug zu einem bestimmten Zeitpunkt für einen bestimmten Preis gebucht werden sollte oder ob dieser in Zukunft günstiger angeboten wird [Tzi+17]. Grund hierfür ist zudem auch die mangelnde Verfügbarkeit von Airlineinternen Informationen wie beispielsweise die Anzahl verfügbarer Sitzplätze für einen Flug.

Airlines gestalten ihre Preise, indem sie auf viele verschiedene externe und interne Faktoren zurückgreifen [Abd+21]. Externe Faktoren sind beispielsweise:

- Wetterbedingungen,
- Events,
- Inflation,
- Naturkatastrophen,
- Terrorismus,
- politische Lage.

Interne Faktoren sind beispielsweise:

- Buchungsdatum,
- historische Preisdaten,
- Abflugdatum,
- Saison,
- Ferienzeiten,
- Verfügbare Flüge,
- Verfügbare Airlines,
- Buchungsklasse,
- Verfügbarkeit der Sitzplätze,
- Flugdistanz.

Zudem machen sich die Fluggesellschaften die verändernde Zahlungsbereitschaft der Kunden zu Nutze. Die Zahlungsbereitschaft macht sich beispielsweise bei den unterschiedlichen Zielgruppen bemerkbar: Geschäftsreisende buchen eher kurzfristig und sind im Allgemeinen bereit, mehr Geld für das Flugticket auszugeben, während Freizeitreisende eine niedrigere Zahlungsbereitschaft besitzen und eher dazu neigen, weit im Voraus zu buchen. Um den Gewinn zu erhöhen und die unterschiedlichen Zahlungsbereitschaften auszunutzen, ist es für die Airlines sinnvoll, die Preise kurz vor Abflug deutlich anzuheben. Dies bestätigen auch Groves und Gini in [GG11]: Es konnte herausgefunden werden, dass Flugpreise kurz vor Abflug angehoben werden. Diese Preiserhöhung erfolgte allerdings bei Businessflügen schon deutlich früher als bei Urlaubsflügen.

4.1.3 *Ziele*

Um nun preisbewusste Kunden bei der Buchung eines möglichst günstigen Fluges zu unterstützen, soll Machine Learning eingesetzt werden. Auf der Suche nach Flügen besteht für den Kunden Zugriff auf die folgenden Informationen:

- Buchungsdatum (aktuelles Datum der Preisabfrage),
- Abflugdatum,
- Airline (inkl. Flugnummer und Flugzeiten),
- Abflug- und Zielflughafen,
- aktueller Preis.

Anhand dieser Informationen sollen zwei Ansätze verfolgt werden. Zum einen soll für diesen bestimmten Flug der niedrigste zu erwartende Preis vorhergesagt werden (Regressionsproblem). Zum anderen soll Machine Learning eingesetzt werden, um eine Empfehlung auszusprechen, ob dieser bestimmte Flug gebucht werden soll oder ob dieser künftig voraussichtlich günstiger angeboten wird (Klassifikationsproblem). Hierbei wird zwischen den beiden Empfehlungen „buchen“ und „nicht buchen“ unterschieden.

Auf Basis des niedrigsten zu erwartenden Preises oder der Empfehlung (buchen / nicht buchen) können Kunden schließlich ihre Entscheidung treffen, einen Flug zu buchen oder nicht zu buchen.

4.1.4 *Vorgehensweise*

Mit Hilfe eines geeigneten Datensatzes sollen verschiedene Machine-Learning-Algorithmen trainiert werden. Dies erfolgt mit dem Tool Jupyter Notebook in der Programmiersprache Python. Es werden die Python-Bibliotheken *scikit-learn* [Ped+11], *Pandas* [McK10], *Numpy* [Har+20] und *Matplotlib* [Hun07] verwendet.

4.2 DATA UNDERSTANDING

4.2.1 *Datenbeschaffung*

Für die Datenbeschaffung eignen sich die Flugpreise einer Route, die mehrfach täglich von verschiedenen Airlines bedient wird. Zudem soll es keine reine Businessstrecke, aber auch keine reine Urlaubsstrecke sein.

Eine Route, die diese Voraussetzungen erfüllt, ist die Route von Hamburg nach London. Diese wird mehrfach täglich von verschiedenen Airlines angeboten, wobei vom Flughafen Hamburg (HAM) die Flughäfen London-Heathrow (LHR), London-Gatwick (LGW) und London-Stansted (STN) angeflogen werden. Zu diesen Airlines gehören die Billigfluggesellschaften

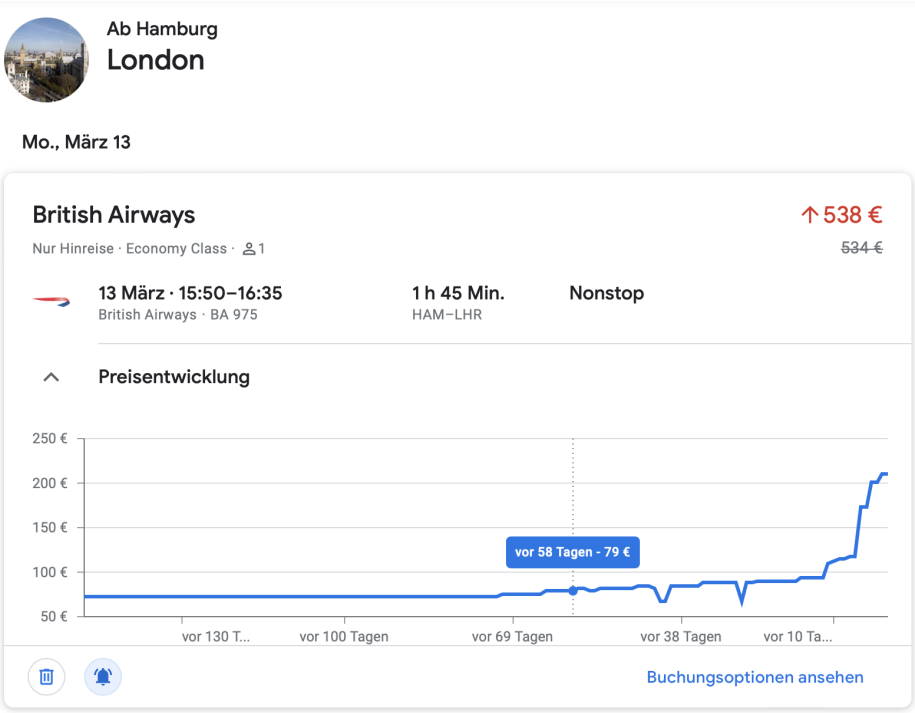

Abbildung 4.1: Google Flüge: Preisentwicklung für einen beispielhaften Flug von Hamburg nach London mit British Airways. Entnommen von https://www.google.com/travel/flights/ am 12.03.2023.

easyJet, Ryanair sowie die Lufthansa-Tochter Eurowings. Als klassische Airline bietet zudem British Airways Flüge zwischen Hamburg und London an.

Mit Hilfe von *Google Flüge*[1] ist es möglich, die Preisentwicklung für einen ausgewählten Flug zu betrachten. Abbildung 4.1 zeigt eine beispielhafte Preisentwicklung. Abhängig vom Flug kann für einen Zeitraum beginnend ab ca. 100 bis 381 Tage vor Abflug bis einen Tag vor Abflug jede einzelne Preisänderung angezeigt werden, wobei immer der *günstigste Tagespreis* angezeigt wird. Stündlich ändernde Preise werden dementsprechend nicht berücksichtigt.

Um den Datensatz zu erzeugen, werden die Preisänderungen für jeden verfügbaren Flug manuell erfasst und in einer CSV[2]-Datei gespeichert. Die Datenerfassung erfolgt täglich für alle Flüge auf genannter Strecke für einen Zeitraum von insgesamt vier Wochen.

4.2.2 *Vorstellung des Datensatzes*

Der Datensatz [Lew23c], der für die vorliegende Arbeit verwendet wird, besteht aus 237 Flügen mit insgesamt 47.233 aufgezeichneten Flugpreisen für Flüge von Hamburg nach London. Bei allen Flügen handelt es sich um

1 https://www.google.com/travel/flights/
2 Comma-seperated values

```
1 data.head()
```

	Buchungsdatum	Abflugdatum	Abflughafen	Zielflughafen	Abflugzeit	Ankunftszeit	Preis	Flugnummer	Airline
0	2022-11-09	2023-02-26	HAM	LHR	17:55:00	18:40:00	98	EW 7460	Eurowings
1	2022-11-10	2023-02-26	HAM	LHR	17:55:00	18:40:00	98	EW 7460	Eurowings
2	2022-11-11	2023-02-26	HAM	LHR	17:55:00	18:40:00	98	EW 7460	Eurowings
3	2022-11-12	2023-02-26	HAM	LHR	17:55:00	18:40:00	98	EW 7460	Eurowings
4	2022-11-13	2023-02-26	HAM	LHR	17:55:00	18:40:00	98	EW 7460	Eurowings

Abbildung 4.2: Ein Auszug der gesammelten Daten für einen Flug. Eigene Darstellung aus [Lew23a].

One-Way-Flüge (nur Hinflug), die non-stop (ohne Zwischenstopp oder Umstieg) durchgeführt werden. Im Datensatz ist immer der günstigste Tagespreis der Economy Class angegeben, unabhängig von airlinespezifischen Tarifen. Dementsprechend werden weder die Anzahl an Handgepäck- oder Aufgabegepäckstücken noch inkludierte Leistungen wie Verpflegung, Sitzplatzreservierung etc. berücksichtigt.

Abbildung 4.2 zeigt beispielhaft die ersten Zeilen des Datensatzes für einen ausgewählten Flug. Die Merkmale werden in Anhang A, Tabelle A.1 genauer dargestellt, indem die Attributausprägungen erläutert werden.

Beispiel 4.1. Bezogen auf Zeile 0 der Abbildung 4.2: Der Eurowings-Flug mit der Flugnummer EW 7460 und Abflug am 26.02.2023 um 17:55 Uhr vom Flughafen Hamburg und Ankunft um 18:40 Uhr am Flughafen London-Heathrow wurde am 09.11.2022 zu einem Preis von 98 € angeboten.

Tabelle 4.1 kann entnommen werden, wie sich die gesamten Flugpreise des Datensatzes auf die unterschiedlichen Airlines verteilen. Zudem ist interessant, wie sich die Anzahl der durchgeführten Flüge zur Anzahl der Flugpreise verhält. Daher wird in Tabelle 4.1 auch dargestellt, wie viele Flüge der jeweiligen Airline im Datensatz vorhanden sind, wobei sich ein Flug durch seine Flugnummer und das Abflugdatum identifizieren lässt. Auffällig ist, dass der Großteil der Flugpreise (über 50 %) auf British Airways zurückzuführen ist, gefolgt von Eurowings. Die Billigfluggesellschaften Ryanair und easyJet sind nur in geringem Umfang mit ca. 10 % im Datensatz vertreten.

Airline	Anzahl der Preisdaten	Anzahl durchgeführter Flüge
British Airways	25.528	122
Eurowings	12.355	71
easyJet	4.335	22
Ryanair	5.015	22
Gesamt	47.233	237

Tabelle 4.1: Anzahl der verfügbaren Flugpreisdaten und die Anzahl der durchgeführten Flüge in Abhängigkeit von der Airline. Eigene Darstellung.

4.2.3 *Explorative Datenanalyse*

Im Rahmen der explorativen Datenanalyse werden zunächst Datenpunkte mit NaN-Werten[3] entfernt, um beispielsweise Mittelwertberechnungen durchführen zu können. Anschließend werden statistische Lagemaße ausgewählt und für den Datensatz berechnet. Das Ergebnis ist in Tabelle 4.2 dargestellt. Der Datensatz wird zudem um das Attribut *Tage bis Abflug* erweitert, das sich aus den vorhandenen Daten ableiten lässt.

Lagemaß	**Preis in Euro**
Arithm. Mittel	77,53
Median	67,00
Modalwert	58,00
Standardabweichung	62,55
Minimum	17,00
25 % - Quantil	49,00
50 % - Quantil	67,00
75 % - Quantil	85,00
Maximum	927,00

Tabelle 4.2: Darstellung statistischer Lagemaße. Eigene Darstellung.

Diese Lagemaße werden auch bezogen auf die unterschiedlichen Airlines berechnet und können dem Unterkapitel 3.3. des ersten praktischen Teils entnommen werden.

Anschließend wird der Flugpreis in Abhängigkeit des Wochentages des Abflugs und des Wochentages der Buchung untersucht. Es stellt sich heraus, dass es einen Wochentag des Abflugs gibt, an dem die Flugpreise im Durchschnitt am günstigsten sind. Abbildung 4.3 kann entnommen werden, dass Abflüge an Dienstagen durchschnittlich am günstigsten sind, während Abflüge an Samstagen am teuersten sind. Für den Wochentag der Buchung kann herausgefunden werden, dass dieser irrelevant ist und keinen Einfluss auf den Flugpreis hat (bezogen auf den verwendeten Datensatz).

Insgesamt werden Flugpreise durchschnittlich betrachtet immer teurer, je näher das Abflugdatum kommt. Im Schnitt sind die Flüge zwischen 200 und 350 Tage vor Abflug am günstigsten. Diese Erkenntnis widerspricht der in Unterabschnitt 4.1.2 aufgestellten Behauptung (in Anlehnung an [Abd+21; Jos+20; GG11]), dass es nicht unbedingt sinnvoll sei, eine Buchung früh aufzugeben. Dies könnte zum einen daran liegen, dass Airlines ihre Preisstrategien ggf. ständig anpassen, um kein Muster bei der Preisgestaltung erkennen zu lassen. Zum anderen könnte der Datensatz zu klein sein, um allgemeine Behauptungen dieser Art verifizieren oder widerlegen zu können.

Zu beachten ist, dass British Airways am häufigsten im Datensatz vertreten ist und daher großen Einfluss auf die Durchschnittswerte hat. Aus

3 NaN steht für *Not a Number*. Dies sind Dateneinträge ohne Wert.

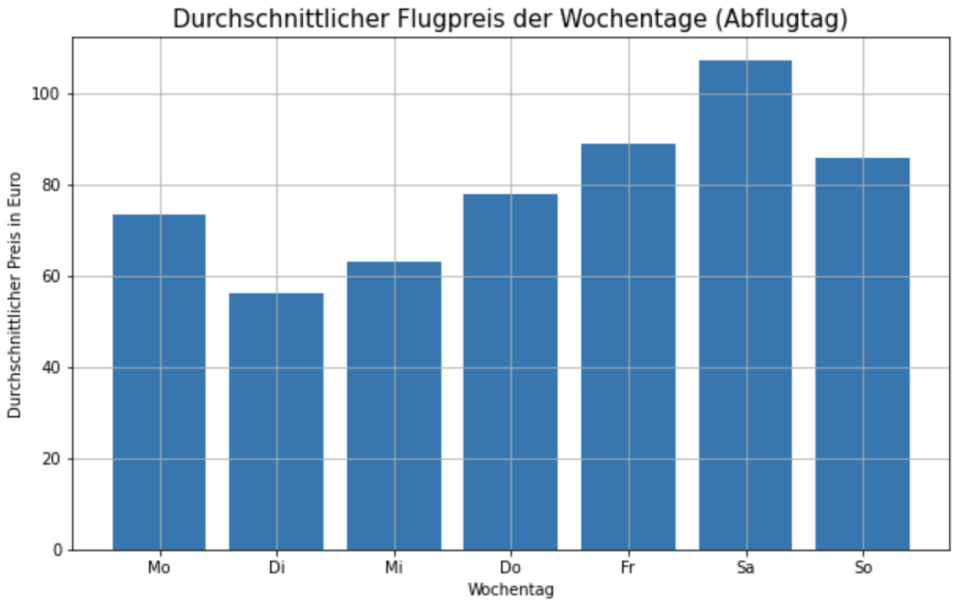

Abbildung 4.3: Durchschnittlicher Flugpreis der Wochentage des Abfluges. Eigene Darstellung aus [Lew23a].

diesem Grund werden die unterschiedlichen Airlines im Unterkapitel 3.6. des ersten praktischen Teils auch einzeln betrachtet und die Flugpreise im Zeitverlauf dargestellt.

Für numerische Attribute kann eine Korrelationsmatrix erzeugt werden, die die Korrelation zwischen jeweils zwei Merkmalen darstellt. Um auch die Airline und die Abflugzeit berücksichtigen zu können, müssen diese Merkmale in numerische Attribute umgewandelt werden. Hierfür werden zwei neue Spalten im Datensatz erzeugt, die das nicht-numerische Attribut codieren. Damit eine Korrelation sinnvoll bestimmt werden kann, müssen die Attribute eine Rangfolge besitzen. Während diese bei der Abflugzeit gegeben ist, ist eine Rangfolge der Airline nicht offensichtlich, weshalb diese nach subjektivem Empfinden von *klassischer Airline* (im Allgemeinen hochpreisig) nach *Billigairline* (im Allgemeinen günstig) geordnet wird. Die Codierungen lauten wie folgt:

- *Airlinecode* (0: British Airways, 1: Eurowings, 2: easyJet, 3: Ryanair): Die Airlines werden durch numerische Werte repräsentiert
- *Abflugzeit codiert* (0: früh morgens, 1: morgens, 2: mittags, 3: nachmittags, 4: abends): Die Abflugzeit wird in verschiedene Kategorien eingeteilt und ebenfalls durch einen numerischen Wert repräsentiert.

Anschließend wird der Pearson-Korrelationskoeffizient bestimmt. Hierbei handelt es sich um ein Maß für die lineare Beziehung zwischen zwei Variablen. Der Korrelationskoeffizient nimmt einen Wert zwischen -1 und 1 an, wobei -1 auf eine negative Korrelation, 1 auf eine positive Korrelation und 0 auf keine Korrelation hinweist. Die Korrelationsmatrix ist in Abbildung 4.4 dargestellt. Es fällt auf, dass die Anzahl der Tage bis Abflug, die

	Preis	Wochentag der Buchung	Wochentag des Abflugs	Tage bis Abflug	Airlinecode	Abflugzeit codiert	Zielflughafen codiert
Preis	1.000000	0.001322	0.175157	-0.362053	-0.249549	-0.143292	-0.271884
Wochentag der Buchung	0.001322	1.000000	-0.003340	-0.001697	-0.000144	0.000729	0.000400
Wochentag des Abflugs	0.175157	-0.003340	1.000000	0.054411	-0.039935	-0.044279	-0.012897
Tage bis Abflug	-0.362053	-0.001697	0.054411	1.000000	-0.074541	-0.018014	-0.005899
Airlinecode	-0.249549	-0.000144	-0.039935	-0.074541	1.000000	0.490138	0.903079
Abflugzeit codiert	-0.143292	0.000729	-0.044279	-0.018014	0.490138	1.000000	0.533411
Zielflughafen codiert	-0.271884	0.000400	-0.012897	-0.005899	0.903079	0.533411	1.000000

Abbildung 4.4: Korrelationsmatrix. Interessant ist dabei insbesondere die erste Spalte. Eigene Darstellung aus [Lew23a].

Airline und der Zielflughafen das größte Potenzial haben, um zur Vorhersage von Flugpreisen herangezogen zu werden. Der Wochentag der Buchung hat, wie schon zuvor ersichtlich war, keinen Einfluss auf den Preis.

4.3 DATA PREPERATION

4.3.1 *Auswahl geeigneter Merkmale*

Wie bereits in Unterabschnitt 4.2.3 dargestellt wurde, sollen die Merkmale *Tage bis Abflug*, *Airline* und *Zielflughafen* beibehalten werden. Die *Abflugzeit* und der *Wochentag des Abflugs* bleiben trotz geringerer Korrelation bestehen, da diese dennoch zu nicht-linearen Mustern beitragen könnten. Der *Preis* bleibt als Label erhalten. Die verbleibenden Attribute werden aus folgenden Gründen entfernt:

- Buchungsdatum: Das genaue Buchungsdatum ist für eine Vorhersage nicht relevant, da es zu einem Overfitting führen könnte. Das Attribut ist für einen kleinen Datensatz zu speziell und kann nicht verallgemeinert werden.
- Wochentag der Buchung: Dieses Attribut hat, wie bereits erwähnt, keinen nennenswerten Einfluss auf den Preis.
- Abflugdatum: Dieses Attribut ist (analog zum Buchungsdatum) zu speziell und nicht repräsentativ.
- Abflughafen: Da es sich immer um den Flughafen Hamburg handelt, kann dieses Attribut entfallen.
- Ankunftszeit: Die Ankunftszeit hat dieselbe Aussagekraft wie die Abflugzeit, weshalb ein Attribut ausreichend ist. Die Ankunftszeit ist lediglich die um die Flugzeit aufaddierte Abflugzeit.

4.3.2 *Datenbereinigung*

Das Attribut *Preis* besitzt fehlende Werte. Für den Umgang mit einzelnen fehlenden Werten gibt es im Allgemeinen mehrere Möglichkeiten. Diese sind u.a.:

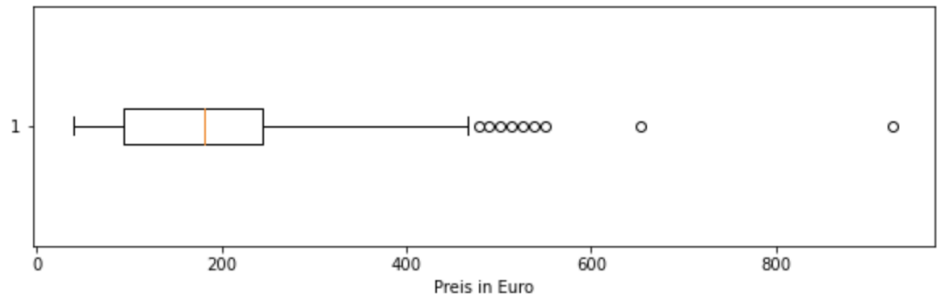

Abbildung 4.5: Boxplot zur Identifizierung von Ausreißern. Berücksichtigt wurden die gesamten Flugpreise von British Airways. Eigene Darstellung aus [Lew23b].

- Die entsprechenden Datenpunkte (Zeilen) werden ersatzlos entfernt,
- das Merkmal wird ersatzlos entfernt oder
- fehlende Werte werden ersetzt (z.B. durch arithmetisches Mittel, Median, null etc.).

Im Unterkapitel 3.2. des ersten praktischen Teils wird ausführlich diskutiert, wie fehlende Werte ersetzt werden könnten. Dies führt zu dem Entschluss, dass keine sinnvolle Ersetzungsstrategie gefunden werden kann, weshalb alle Datenpunkte ersatzlos entfernt werden, sofern diese einen fehlenden Wert besitzen.

Es lässt sich feststellen, dass für einige Airlines Ausreißer in den Flugpreisen zu finden sind. Um diese zu identifizieren und zu bereinigen, wird je Airline ein Boxplot erzeugt. Ein Beispiel ist in Abbildung 4.5 anhand der Flugpreise von British Airways dargestellt. Bei den als Kreis dargestellten Datenpunkten handelt es sich um Ausreißer. Diese werden für jede Airline bestimmt und anschließend mit dem Maximum ersetzt, das verbleiben würde, wenn alle Ausreißer entfernt werden würden. Auf diese Weise bleiben die Datenpunkte weiterhin hochpreisig, werden aber soweit abgeschwächt, dass sie nicht mehr als Ausreißer gelten.

Die gesamte Datenbereinigung ist Kapitel 2 des zweiten praktischen Teils zu entnehmen.

4.3.3 *Labelerzeugung*

Der Datensatz für die Flugpreisvorhersage (Regressionsproblem) besitzt bereits das Label *Preis* und ist demnach vollständig.

Für das Klassifikationsproblem soll ein eigener Datensatz mit einem zusätzlichen Label erzeugt werden. Das Label enthält die Empfehlung, einen Flug zu einem bestimmten Preis zu buchen oder nicht zu buchen. Es handelt sich um ein binäres Attribut, das durch die Zahlen 1 (buchen) und 0 (nicht buchen) repräsentiert wird.

Ein Flug soll genau dann gebucht werden, wenn dieser als *günstig* eingestuft werden kann. Ein Flug soll nicht gebucht werden, wenn absehbar ist,

	Flugnummer	Airline	Zielflughafen	Wochentag des Abflugs	Abflugzeit	Tage bis Abflug	Preis
0	EW 7460	Eurowings	LHR	Sonntag	nachmittags	109	98
1	EW 7460	Eurowings	LHR	Sonntag	nachmittags	108	98
2	EW 7460	Eurowings	LHR	Sonntag	nachmittags	107	98
3	EW 7460	Eurowings	LHR	Sonntag	nachmittags	106	98
4	EW 7460	Eurowings	LHR	Sonntag	nachmittags	105	98

Abbildung 4.6: Die ersten fünf Zeilen des finalen Regressionsdatensatzes. Eigene Darstellung aus [Lew23b].

	Flugnummer	Airline	Zielflughafen	Wochentag des Abflugs	Abflugzeit	Tage bis Abflug	Preis	Buchen
0	EW 7460	Eurowings	LHR	Sonntag	nachmittags	109	98	0
1	EW 7460	Eurowings	LHR	Sonntag	nachmittags	108	98	0
2	EW 7460	Eurowings	LHR	Sonntag	nachmittags	107	98	0
3	EW 7460	Eurowings	LHR	Sonntag	nachmittags	106	98	0
4	EW 7460	Eurowings	LHR	Sonntag	nachmittags	105	98	0

Abbildung 4.7: Die ersten fünf Zeilen des finalen Klassifikationsdatensatzes. Eigene Darstellung aus [Lew23b].

dass noch günstigere Flüge folgen werden. Hierfür wird ein *günstiger Preis* wie folgt definiert:

Definition 4.2. *Sei y der Flugpreis zu einem bestimmten Zeitpunkt x und sei P die Menge aller noch kommenden Preise für diesen Flug, für die gilt: Tage bis Abflug $\leq$ x (inklusive dem Preis y). Gesucht wird eine Teilmenge $G \subseteq P$ mit $|G| = |P| \cdot 0.15$, die die niedrigsten Preise der Menge P enthält. Der Preis y ist genau dann günstig, wenn $y \in G$ und y somit zu den* 15 % *der niedrigsten verbleibenden Flugpreisen gehört.*

Es wird eine Funktion (vgl. Kapitel 2, Zeile 16 des zweiten praktischen Teils) verwendet, die für jeden Datenpunkt einzeln entscheiden kann, ob dieser Flug zu den gegebenen Konditionen gebucht werden sollte oder nicht. Die Funktion bekommt den gesamten Datensatz, eine Flugnummer, ein Abflugdatum, die Anzahl der Tage bis Abflug (x) und den Flugpreis (y) übergeben. Anschließend wird die Menge P nach obiger Definition bestimmt. Diese Menge wird aufsteigend sortiert. Nun bilden die obersten 15 % der sortierten Menge P genau die Menge G. Es wird 1 zurückgegeben, wenn nun $y \in G$ ist, ansonsten wird 0 zurückgegeben. Wird diese Funktion auf jeden Datenpunkt angewendet, kann für den gesamten Datensatz das entsprechende Label erzeugt werden.

4.3.4 *Finale Datensätze*

Die ersten Zeilen der finalen Datensätze können Abbildung 4.6 (Regressionsproblem) und Abbildung 4.7 (Klassifikationsproblem) entnommen werden.

4.3.5 *One-Hot-Codierung*

Grundsätzlich funktionieren Machine-Learning-Algorithmen besser auf numerischen Daten. Daher ist es sinnvoll, die kategorischen Attribute in numerische Attribute umzuwandeln. Es handelt sich um die Attribute: Flugnummer, Airline, Zielflughafen, Wochentag des Abflugs und Abflugzeit. Eine Transformation erfolgt durch die sogenannte *One-Hot-Codierung*: Für jede Kategorie wird ein neues binäres Attribut erzeugt. Trifft diese jeweilige Kategorie für einen Datenpunkt zu, wird eine 1 notiert, ansonsten eine 0. Bei den neuen Merkmalen handelt es sich um sogenannte Dummy-Merkmale. Die One-Hot-Codierung soll auf alle zuvor genannten kategorischen Attribute angewendet werden. Die Umsetzung kann Kapitel 3 des zweiten praktischen Teils entnommen werden.

4.3.6 *Trainings- und Testdaten*

Es wird ein rein zufälliger Trainingsdatensatz ausgewählt, der 80 % des gesamten Datensatzes (vgl. Unterabschnitt 4.3.4) ausmacht. Die verbleibenden 20 % dienen als Testdatensatz. Der Testdatensatz bleibt unberührt, bis die Modelle endgültig trainiert wurden. Erst danach folgen die Tests anhand der Testdaten.

4.4 MODELING: TRAINING UND OPTIMIERUNG

4.4.1 *Regressionsproblem*

Als Qualitätsmaße dienen:

- R2-Score: Der R2-Score wird für Regressionsmodelle verwendet und liegt zwischen 0 und 1. Der Score gibt an, wie gut sich die Prädiktoren eignen, um das Label vorherzusagen, wobei ein Wert von 1 auf eine perfekte Modellanpassung hindeuten würde. Ein Wert von 0 würde bedeuten, dass das Modell genauso schlechte Vorhersagen trifft wie die reine Durchschnittsvorhersage der Zielvariablen. Der Score kann auch in Prozent angegeben werden.
- MAE (mean absolute error): Gibt das Mittel der absoluten Abweichung zwischen tatsächlichem und vorhergesagtem Wert an. Dieser Wert sollte möglichst klein sein.
- MSE (mean squared error): Vgl. Abschnitt 3.1, Gleichung 3.3.
- RMSE (root mean squared error): Dieser Wert entspricht der Wurzel des MSE und erlaubt eine bessere Interpretation als der MSE. Je kleiner der RMSE ist, desto näher liegen die vorhergesagten Werte an den tatsächlichen Werten.

Der Ablauf kann in sechs Schritte unterteilt werden, wobei die letzten drei Schritte optional sind und nur durchgeführt werden, sofern das Modell nicht zuvor aufgrund schlechter Ergebnisse verworfen wurde:

1. Der entsprechende Machine-Learning-Algorithmus wird von *scikit-learn* importiert. Das Modell wird ohne jegliche Optimierung mit Hilfe der Trainingsdaten trainiert.
2. Die berechneten Qualitätsmaße des Modells werden ausgegeben.
3. Es erfolgt eine Kreuzvalidierung (vgl. Kapitel 2, Unterabschnitt 2.3.6) mit Ausgabe der Ergebnisse.
4. Zur Optimierung des Modells wird eine Gittersuche durchgeführt, bei der verschiedene Belegungen der Hyperparameter automatisiert durchlaufen werden, um die beste Kombination dieser Parameter bestimmen zu können.
5. Ein optimiertes Modell wird mit den zuvor gefundenen Hyperparametern trainiert.
6. Erneute Berechnung der Qualitätsmaße und Durchführung der Kreuzvalidierung auf dem optimierten Modell.

Diese Schritte werden auf alle geeigneten Algorithmen angewendet und können dem Kapitel 5 des zweiten praktischen Teils entnommen werden.

4.4.2 *Klassifikationsproblem*

Für das Klassifikationsproblem wird ein vom Regressionsproblem abweichendes Verfahren gewählt. Grund hierfür ist, dass das Trainieren und Optimieren sämtlicher Modelle enorme Rechenzeiten benötigt. Basierend auf dieser Erfahrung wird das weitere Vorgehen entsprechend angepasst. Zunächst wird eine Kreuzvalidierung aller geeigneter Machine-Learning-Algorithmen auf dem Trainingsdatensatz durchgeführt. Als Qualitätsmaß wird der Anteil der korrekt klassifizierten Datenpunkte gewählt. Anschließend werden die drei besten Modelle ausgewählt und mittels Gittersuche optimiert. Das gesamte Vorgehen ist ebenfalls Kapitel 5 des zweiten praktischen Teils zu entnehmen.

4.5 EVALUATION

Nachdem die Modelle trainiert und optimiert wurden, sollen die endgültigen Modelle getestet werden. Hierfür wird der Testdatensatz herangezogen. Das entsprechende Modell berechnet die Vorhersagen für den gesamten Testdatensatz (ohne Labels). Da die Labels bekannt sind, können nun die Vorhersagen mit den tatsächlichen Werten verglichen werden. Erneut werden die o.g. Qualitätsmaße berechnet, um deuten zu können, wie gut das Modell auf unbekannten Daten funktioniert.

Eine Übersicht der vollständigen Ergebnisse ist in Anhang B zu finden. Tabelle B.1 und Tabelle B.2 stellen die Ergebnisse des Regressionsproblems dar, während Tabelle B.3 und Tabelle B.4 die Ergebnisse des Klassifikationsproblems darstellen. Es werden jeweils die zuerst trainierten Modelle und die optimierten Modelle gegenübergestellt.

Das für das Regressionsproblem am besten funktionierende Modell ist das MLP mit einem R2-Score von 61,75 % auf den Testdaten.

Das für das Klassifikationsproblem am besten funktionierende Modell ist der Random Forest mit einer Genauigkeit von 91,66 % auf den Testdaten.

4.6 DEPLOYMENT

Zum Deployment gehört unter anderem auch der Einsatz der endgültig trainierten Modelle in der Praxis auf unbekannten Daten. Da dies im Rahmen dieser Arbeit nicht umsetzbar ist, soll zumindest anhand von simulierten Buchungen herausgefunden werden, ob sich Machine-Learning einsetzen lässt, um bei der Flugbuchung Kosten zu sparen.

Hierfür werden Flugbuchungen von Passagieren simuliert und die Ersparnis in Prozent berechnet, wenn dieser Passagier auf die Empfehlung der Vorhersage hört. Verwendet wird an dieser Stelle wieder der gesamte Datensatz aus Unterabschnitt 4.3.4.

Es sei darauf hingewiesen, dass die Modelle bereits Teile des Datensatzes für das Training verwendet haben und daher keine vollständig unbekannten Daten vorliegen. Diese Vorgehensweise ist nicht optimal, allerdings sind lückenlose Flugpreisdaten für die Simulation erforderlich. Allein anhand des Testdatensatzes wäre die Simulation nicht umsetzbar.

Um dennoch ein repräsentatives Ergebnis zu erhalten, erfolgen jeweils 50 Durchläufe der Simulation, sodass am Ende 5.000 Flugbuchungen simuliert werden. Dies entspricht einem Anteil von ca. 10 % der gesamten Daten. Das Vorgehen unterscheidet sich nach der Art des Problems und wird in den kommenden Unterabschnitten erläutert.

4.6.1 *Regressionsproblem*

Das Kundenverhalten soll je Durchlauf wie folgt simuliert werden:

1. Es werden 100 zufällige Datenpunkte des Datensatzes ausgewählt. Dabei entspricht ein Datenpunkt dem Flug, den der Kunde buchen möchte und dem Zeitpunkt, zu dem er nach einem Flug sucht.

2. Es werden alle nachfolgenden Preise für diesen Flug bis einen Tag vor Abflug durch das Modell vorhergesagt.

3. Es wird genau an dem Tag gebucht, an dem der niedrigste Preis vorhergesagt wurde. Gebucht wird zum tatsächlichen Preis, der dem Datensatz für den entsprechenden Tag entnommen werden kann.

```
Anzahl der Buchungen, bei denen der Preis gleich blieb:  3833
Anzahl der Buchungen, bei denen draufgezahlt wurde:  488
Anzahl der Buchungen, bei denen gespart wurde:  679

Anteil der Buchungen, bei denen draufgezahlt wurde:  9.76 %
Der durchschnittliche Aufpreis beträgt:  21.01 %

Anteil der Buchungen, bei denen gespart wurde:  13.58 %
Die durchschnittliche Ersparnis beträgt:  22.84 %
```

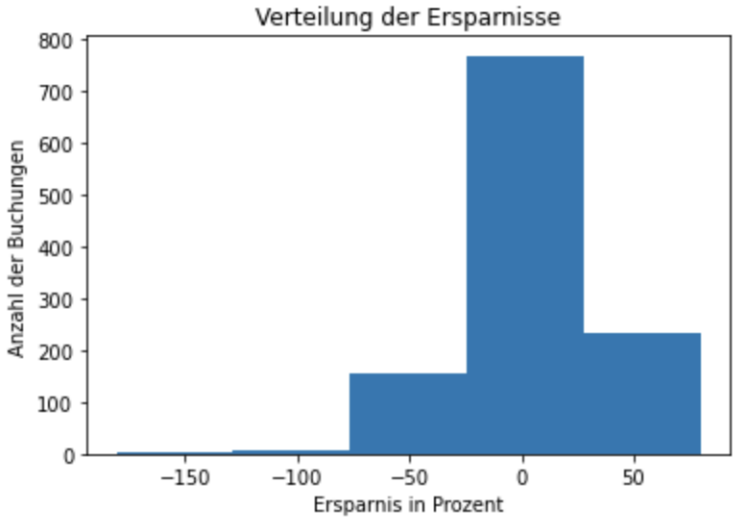

Abbildung 4.8: Ausgabe der Simulationsergebnisse des Regressionsproblems. Eigene Darstellung aus [Lew23b].

Nach 50 Durchläufen der Simulation ergeben sich die in Abbildung 4.8 dargestellten Ergebnisse. Hierbei handelt es sich um die Python-Ausgabe des zweiten praktischen Teils. Zunächst wird ausgegeben, bei wie vielen der 5.000 Buchungen der Preis gleichgeblieben, gestiegen oder gesunken ist. Diese Ergebnisse werden anschließend in Prozent ausgegeben und es wird die prozentuale Ersparnis bzw. der Aufpreis ausgegeben. Schließlich werden die Ergebnisse in einem Histogramm visualisiert. Hierbei wird die Anzahl der Buchungen (y-Achse) dargestellt, bei denen ein bestimmter Prozentsatz (x-Achse) eingespart wurde, indem auf die Empfehlung des Modells gehört wurde. Ein negativer Prozentsatz bedeutet, dass die Buchung am Ende zu einem höheren Preis getätigt wurde, als wenn die Buchung sofort erfolgt wäre.

Insgesamt ist zu erkennen, dass es nicht sinnvoll ist, auf die Vorhersagen des Modells zu vertrauen. In nur 13,58 % aller Test-Buchungen wurde überhaupt Geld gespart. In 9,76 % der Buchungen musste sogar mehr bezahlt werden.

4.6.2 *Klassifikationsproblem*

Das Kundenverhalten soll je Durchlauf wie folgt simuliert werden:

```
Anzahl der Buchungen, bei denen der Preis gleich blieb:  3545
Anzahl der Buchungen, bei denen draufgezahlt wurde:  100
Anzahl der Buchungen, bei denen gespart wurde:  1355

Anteil der Buchungen, bei denen draufgezahlt wurde:  2.0 %
Der durchschnittliche Aufpreis beträgt:  17.73 %

Anteil der Buchungen, bei denen gespart wurde:  27.1 %
Die durchschnittliche Ersparnis beträgt:  23.99 %
```

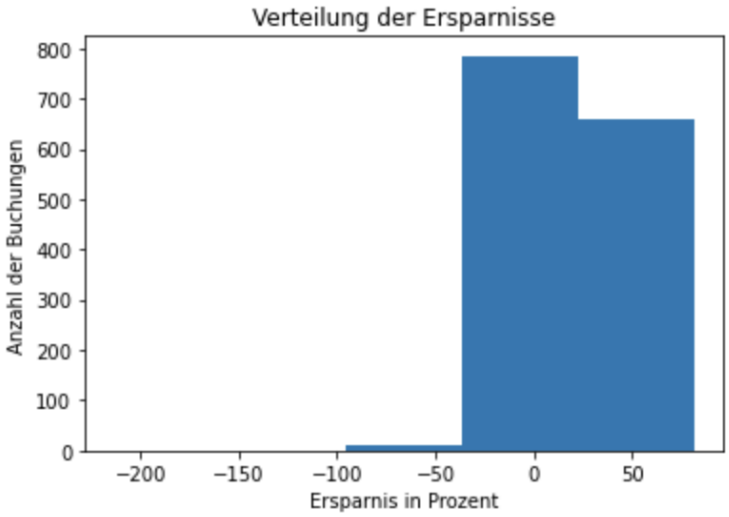

Abbildung 4.9: Ausgabe der Simulationsergebnisse des Klassifikationsproblems. Eigene Darstellung aus [Lew23b].

1. Für einen speziellen ausgewählten Flug mit einem bestimmten Preis wird die die Klasse 1 (buchen) oder 0 (nicht buchen) vorhergesagt.

2. Wenn die 1 bestimmt wurde, wird der Flug zum aktuellen Preis gebucht.

3. Wenn die 0 bestimmt wurde, wird der Flug nicht gebucht. Einen Tag später soll der tagesaktuelle Flugpreis erneut abgerufen werden. Es wird mit Punkt 1. fortgefahren, bis ein Flug gebucht wurde.

Analog Unterabschnitt 4.6.1 wird in Abbildung 4.9 die Python-Ausgabe der Simulationsergebnisse für das Klassifikationsproblem dargestellt.

Die Ergebnisse sehen sehr erfreulich aus. In 70,90 % aller Test-Buchungen hat das System die sofortige Buchung empfohlen. Auf diese Weise wurde kein Geld gespart, aber es wurde bereits ein günstiger Flug gebucht.

Interessanter ist die Erkenntnis, dass in 27,10 % aller Buchungen Einsparungen durch die Empfehlung des Modells erzielt werden konnten. Die Ersparnis liegt im Durchschnitt bei 23,99 %.

Lediglich in 2,00 % aller Buchungen musste zu einem höheren Preis gebucht werden, der im Schnitt 17,73 % teurer war. Allerdings sind dies nur 100 von 5.000 Buchungen, weshalb die geringe Anzahl von Fehlentscheidungen in Kauf genommen werden kann.

5 FAZIT UND AUSBLICK

Es wurden sieben unterschiedliche Machine-Learning-Algorithmen angewendet, um Modelle mit Hilfe von Trainingsdaten zu trainieren. Diese Modelle waren anschließend in der Lage, Vorhersagen für unbekannte Daten zu treffen.

Für die Vorhersage exakter Flugpreise (Regressionsproblem) eigneten sich sechs der sieben vorgestellten Algorithmen. Zwei dieser Algorithmen erzielten bereits auf den Trainingsdaten dermaßen schlechte Ergebnisse, dass diese sofort verworfen wurden. Dies ist zum einen die lineare Regression mit einem R2-Score von 32,29 % und zum anderen die SVM mit einem R2-Score von 17,27 %. Die verbliebenen Algorithmen (Entscheidungsbaum, Random Forest, k-Nearest Neighbors und MLP) erzielten auf den Trainingsdaten R2-Scores zwischen 61,94 % und 71,77 %. Angewendet auf die Testdaten konnte keines der Algorithmen überzeugen. Das beste Ergebnis erzielte das MLP mit einem R2-Score von 61,75 %.

Das trainierte MLP-Modell wurde anschließend zur Simulation von 5.000 Flugbuchungen verwendet. Es wurde getestet, wie hoch das Einsparpotenzial ist, wenn genau an dem Tag gebucht wird, für den das Modell den niedrigsten Preis vorhersagt. Das Ergebnis war nicht zufriedenstellend. Zwar konnten bei einigen Buchungen Einsparungen erzielt werden, allerdings musste in vielen Fällen mehr gezahlt werden, als wenn man sofort gebucht hätte. In der Praxis könnte dieses Modell daher nicht eingesetzt werden.

Machine Learning eignet sich demnach - zumindest bezogen auf den vorhandenen Datensatz - nicht, um Flugpreise exakt vorherzusagen, da die Abweichungen zu hoch sind. Dies kann damit begründet werden, dass die Preisgestaltungsstrategien der Airlines zu komplex und unterschiedlich sind, sodass keine Muster in den tatsächlichen Preisen gefunden werden können. Zudem ist es möglich, dass Airlines ihre Preisgestaltungsstrategien regelmäßig ändern, um derartige Muster nicht entdecken zu können.

Für das Klassifikationsproblem (buchen / nicht buchen) wurden ebenfalls sechs der sieben vorgestellten Algorithmen angewendet. Am schlechtesten schnitten der naive Bayes-Klassifikator und die SVM mit einer Genauigkeit von 66,49 % bzw. 68,70 % ab, wobei die Genauigkeit den Anteil korrekt klassifizierter Datenpunkte bezeichnet. Die verbliebenen Algorithmen (Entscheidungsbaum, Random Forest, k-Nearest Neighbors und MLP) konnten Genauigkeiten zwischen 84,15 % und 91,64 % auf dem Trainingsdatensatz erzielen. Angewendet auf den Testdatensatz konnte der Random Forest schließlich mit einer Genauigkeit von 91,66 % überzeugen.

Bei der Simulation von 5.000 Flugbuchungen konnte bei 27,10 % der Buchungen Geld gespart werden, indem auf die Empfehlung des Modells geachtet wurde. Dabei lag die Kostenersparnis durchschnittlich bei 23,99 %.

Lediglich in 2,00 % aller Buchungen musste zu einem höheren Preis gebucht werden, der im Schnitt 17,73 % teurer war. Allerdings waren dies lediglich 100 von 5.000 Buchungen, weshalb die geringe Anzahl von Fehlentscheidungen in Kauf genommen werden kann.

Das Modell lässt sich demnach erfolgreich einsetzen, um Kunden eine Buchungsempfehlung auszusprechen und Einsparungen zu erzielen.

Die Vorhersagegenauigkeit (sowohl für das Regressions- als auch für das Klassifikationsproblem) könnte gesteigert werden, indem die Modelle auf einem deutlich größeren Datensatz trainiert werden, der ggf. noch zusätzliche Routen, Airlines, Tarife etc. enthält. Zudem könnten zur Optimierung weitere Faktoren berücksichtigt werden, wie zum Beispiel Konkurrenzpreise, Ferienzeiten, Veranstaltungen etc. Auch das *Ensemble-Learning* wäre eine vielversprechende Möglichkeit, um die Vorhersagen zu verbessern. Hierbei wird auf die Ergebnisse mehrerer unterschiedlicher Modelle zurückgegriffen, um eine einheitliche Vorhersage zu treffen.

Zusammenfassend lässt sich aus den Ergebnissen schließen, dass sich Machine Learning nicht einsetzen lässt, um exakte Flugpreise vorherzusagen, da die Abweichungen von den tatsächlichen Preisen zu hoch sind. Jedoch konnten mit Hilfe von Machine Learning gute Vorhersagen einer Buchungsempfehlung (buchen / nicht buchen) getroffen werden, wodurch Kunden einen günstigeren Flug buchen können.

Teil I

ANHANG

A

ATTRIBUTE DES DATENSATZES

Attribut	**Skala**[1]	**Beschreibung / Ausprägungen**
Buchungsdatum	Intervallskala	Das Kalenderdatum des jeweils tagesaktuellen Preises für einen zukünftigen Flug. Das Datum hat das Format YYYY-MM-DD.
Abflugdatum	Intervallskala	Tägliche Kalenderdaten der Abflüge zwischen dem 26.02.2023 und dem 25.03.2023 der Form YYYY-MM-DD.
Abflughafen	Nominalskala	Der Abflughafen ist immer der Flughafen Hamburg (HAM).
Zielflughafen	Nominalskala	Als Zielflughafen gibt es drei unterschiedliche Flughäfen in London: Flughafen London-Heathrow (LHR), Flughafen London-Gatwick (LGW) und Flughafen London-Stansted (STN).
Abflugzeit	Intervallskala	Die Uhrzeit des Abfluges der Form HH:MM:SS. Es handelt sich um die lokale Uhrzeit in Hamburg.
Ankunftszeit	Intervallskala	Die Uhrzeit der Ankunft der Form HH:MM:SS. Es handelt sich um die lokale Uhrzeit in London.
Preis	Verhältnisskala	Der Flugpreis in Euro. Es handelt sich um eine natürliche Zahl.
Flugnummer	Nominalskala	Die Flugnummern bestehen aus einem Airlinecode U2 (easyJet), FR (Ryanair), EW (Eurowings), BA (British Airways), gefolgt von einer Zahlenkombination.
Airline	Nominalskala	Die Airlines sind: easyJet, Ryanair, Eurowings, British Airways.

Tabelle A.1: Attribute des Datensatzes. Eigene Darstellung.

1 Unterschieden wird an dieser Stelle zwischen der Nominalskala (Ausprägungen sind unterscheidbar und besitzen keine natürliche Rangfolge), der Intervallskala (Ausprägungen sind quantitativ, Differenzen können gebildet werden, Verhältnisse zwischen zwei Ausprägungen sind nicht interpretierbar) und der Verhältnisskala (Analog Intervallskala mit der Abweichung, dass ein natürlicher Nullpunkt vorhanden ist und der Quotient aussagekräftig ist).

B

ERGEBNISSE

B.1 REGRESSIONSPROBLEM

Erstes Modell:

	Lineare Regression	Entscheidungsbaum	Random Forest	k-Nearest Neighbors	SVM	MLP
Trainingsdaten-Score in %	32,29	71,77	71,41	67,74	17,27	61,94
∅ Kreuzvalidierung in %	47,50	48,00	44,01	39,43	——-	36,15
Testdaten-Score in %	——-	30,10	39,58	51,44	——-	60,89
Mean absolute error in €	——-	18,78	17,60	16,17	——-	16,15

Tabelle B.1: Ergebnisse des ersten Modells ohne Anpassung der Hyperparameter. Die Lineare Regression und die SVM wurden aufgrund schlechter Ergebnisse verworfen. Eigene Darstellung.

Optimiertes Modell:

	Lineare Regression	Entscheidungsbaum	Random Forest	k-Nearest Neighbors	SVM	MLP
Trainingsdaten-Score in %	——-	64,32	66,53	66,78	——-	64,92
∅ Kreuzvalidierung in %	——-	38,05	36,70	38,19	——-	35,59
Testdaten-Score in %	——-	57,74	59,58	55,55	——-	61,75
Mean absolute error in €	——-	16,41	15,83	16,02	——-	15,59

Tabelle B.2: Ergebnisse des optimierten Modells nach Anpassung der Hyperparameter. Eigene Darstellung.

B.2 KLASSIFIKATIONSPROBLEM

Erstes Modell:

	Naiver Bayes-Klassifikator	Entscheidungsbaum	Random Forest	k-Nearest Neighbors	SVM	MLP
∅ Kreuzvalidierung in %	66,49	90,44	91,64	84,86	68,70	84,15

Tabelle B.3: Ergebnisse der Kreuzvalidierung des ersten Modells ohne Anpassung der Hyperparameter. Trainings- und Testdaten-Scores wurden nicht gemessen. Eigene Darstellung.

Optimiertes Modell:

	Naiver Bayes-Klassifikator	Entscheidungsbaum	Random Forest	k-Nearest Neighbors	SVM	MLP
∅ Kreuzvalidierung in %	———	90,83	91,71	86,90	———	———
Testdaten-Score in %	———	90,67	91,66	87,87	———	———

Tabelle B.4: Ergebnisse der Kreuzvalidierung des optimierten Modells nach Anpassung der Hyperparameter sowie des Testdaten-Scores. Berücksichtigt wurden die besten drei Modelle. Eigene Darstellung.

[Abd+21] Juhar A. Abdella, Nazar M. Zaki, Khaled Shuaib und Fahad Khan. "Airline ticket price and demand prediction: A survey". In: *Journal of King Saud University-Computer and Information Sciences* 33.4 (2021), S. 375–391.

[AMMIL12] Yaser S. Abu-Mostafa, Malik Magdon-Ismail und Hsuan-Tien Lin. *Learning from data*. Bd. 4. AMLBook New York, 2012.

[Alp22] Ethem Alpaydin. *Maschinelles Lernen*. Walter de Gruyter GmbH & Co KG, 2022.

[BKI03] Christoph Beierle und Gabriele Kern-Isberner. *Methoden wissensbasierter Systeme*. Springer, 2003.

[BH09] Nicolas Bissantz und Jürgen Hagedorn. "Data Mining (Datenmustererkennung)". In: *Wirtschaftsinformatik* 51.1 (2009), S. 139–144.

[BTL08] David Bong, James Tan und Koon Lai. "Application of multilayer perceptron with backpropagation algorithm and regression analysis for long-term forecast of electricity demand: A comparison". In: *2008 International Conference on Electronic Design*. IEEE. 2008, S. 1–5.

[BB03] E. Andrew Boyd und Ioana C. Bilegan. "Revenue management and e-commerce". In: *Management science* 49.10 (2003), S. 1363–1386.

[Bur19] Andriy Burkov. *Machine Learning kompakt: Alles, was Sie wissen müssen*. MITP-Verlags GmbH & Co. KG, 2019.

[Cha+00] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer u. a. "CRISP-DM 1.0: Step-by-step data mining guide". In: *SPSS inc* 9.13 (2000), S. 1–73.

[Che+15] Yuwen Chen, Jian Cao, Shanshan Feng und Yudong Tan. "An ensemble learning based approach for building airfare forecast service". In: *2015 IEEE International Conference on Big Data*. IEEE. 2015, S. 964–969.

[CB17] Vincy Cherian und M.S. Bindu. "Heart disease prediction using Naive Bayes algorithm and Laplace Smoothing technique". In: *Int. Journal of Computer Science Trends and Technology* 5.2 (2017), S. 68–73.

[Cla18] Andrew Clark. "The machine learning audit-CRISP-DM Framework". In: *Isaca Journal* 1 (2018), S. 42–47.

[Cro98] Robert G. Cross. *Revenue management: Hard-core tactics for market domination*. Broadway Books, 1998.

[FR07] Mark Friesen und Sven Reinecke. "Wahrgenommene Preisfairness bei Revenue Management im Luftverkehr". In: *Marketing Review St. Gallen* 24 (2007), S. 34–39.

[Gér20] Aurélien Géron. *Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme. Aktuell zu TensorFlow 2*. O'Reilly, 2020.

[Gna05] Michael Andreas Josef Gnatz. "Vom Vorgehensmodell zum Projektplan". Diss. Technische Universität München, 2005.

[GG11] William Groves und Maria Gini. *A regression model for predicting optimal purchase timing for airline tickets*. Techn. Ber. University of Minnesota, 2011.

[GC09] Thiago S. Guzella und Walmir M. Caminhas. "A review of machine learning approaches to spam filtering". In: *Expert Systems with Applications* 36.7 (2009), S. 10206–10222.

[Har+20] Charles Harris, Jarrod Millman, Stéfan van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau u. a. "Array programming with NumPy". In: *Nature* 585.7825 (2020), S. 357–362.

[Hea+98] Marti A. Hearst, Susan T. Dumais, Edgar Osuna, John Platt und Bernhard Scholkopf. "Support vector machines". In: *IEEE Intelligent Systems and their applications* 13.4 (1998), S. 18–28.

[HT14] Wibke Heidig und Torsten Tomczak. "Revenue Management aus Kundensicht". In: *Revenue Management aus der Kundenperspektive: Grundlagen, Problemfelder und Lösungsstrategien* (2014), S. 1–17.

[Hen21] Casey Henley. *Foundations of Neuroscience*. Michigan State University Libraries, 2021.

[Hun07] John D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), S. 90–95.

[Jos+20] Neeraj Joshi, Gaurav Singh, Saurav Kumar, Rachna Jain und Preeti Nagrath. "Airline prices analysis and prediction using decision tree regressor". In: *Data Science and Analytics, Part I* (2020), S. 170–186.

[Kle01] Robert Klein. "Revenue Management: Quantitative Methoden zur Erlösmaximierung in der Dienstleistungsproduktion". In: *Betriebswirtschaftliche Forschung und Praxis* 53 (2001), S. 245–259.

[KS08] Robert Klein und Claudius Steinhardt. *Revenue management: Grundlagen und mathematische Methoden*. Springer-Verlag, 2008.

[Kou+15] Konstantina Kourou, Themis P. Exarchos, Konstantinos P. Exarchos, Michalis V. Karamouzis und Dimitrios I. Fotiadis. "Machine learning applications in cancer prognosis and prediction". In: *Computational and structural biotechnology journal* 13 (2015), S. 8–17.

[Lew23a] Dominik Lewin. *Teil I zum praktischen Teil der Bachelorarbeit*. URL: https://github.com/ddominik94/Bachelorarbeit-Dynamic-Pricing/blob/main/BA_erster_Teil.ipynb. 2023.

[Lew23b] Dominik Lewin. *Teil II zum praktischen Teil der Bachelorarbeit*. URL: https://github.com/ddominik94/Bachelorarbeit-Dynamic-Pricing/blob/main/BA_zweiter_Teil.ipynb. 2023.

[Lew23c] Dominik Lewin. *flight_price_data.csv*. URL: https://github.com/ddominik94/Bachelorarbeit-Dynamic-Pricing/blob/main/flight_price_data.csv. 2023.

[MVR99] Jeffrey I. McGill und Garrett J. Van Ryzin. "Revenue management: Research overview and prospects". In: *Transportation science* 33.2 (1999), S. 233–256.

[McK10] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. 2010, S. 56 –61.

[Pap14] Manolis Papadakis. "Predicting Airfare Prices". In: *Clerk Maxwell* (2014).

[Ped+11] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel u. a. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830.

[Pom07] Wilhelm Pompl. *Luftverkehr: Eine ökonomische und politische Einführung*. 5. Aufl. Springer-Verlag, 2007.

[PEK22] Naveen Prasath, Sherin Eliyas und Sathish Kumar. "A Prediction of Flight Fare Using K-Nearest Neighbors". In: *2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. IEEE. 2022, S. 1347–1351.

[Pri22] Anna Priester. "Grundlagen des Dynamic Pricing". In: *Dynamic Pricing aus Konsumentensicht: Theoretische Analyse und empirische Untersuchung von Einflussfaktoren und Konsequenzen* (2022), S. 10–96.

[RRC19] Gopinath Rebala, Ajay Ravi und Sanjay Churiwala. "Machine learning definition and basics". In: *An introduction to machine learning* (2019), S. 1–17.

[RYY15] Ruixuan Ren, Yunzhe Yang und Shenli Yuan. *Prediction of airline ticket price*. Techn. Ber. University of Stanford, 2015.

[Sci23a] Scikit-Learn.org. *Naive Bayes*. URL: https://scikit-learn.org/stable/modules/naive_bayes.html. Zugriffsdatum: 18 Mai 2023. 2023.

[Sci23b] Scikit-Learn.org. *Nearest Neighbors*. URL: https://scikit-learn.org/stable/modules/neighbors.html. Zugriffsdatum: 13 Mai 2023. 2023.

[Sci23c] Scikit-Learn.org. *Neural network models*. URL: https://scikit-learn.org/stable/modules/neural_networks_supervised.html. Zugriffsdatum: 10 Juli 2023. 2023.

[Sri21] Abhishek Srivastava. "Impact of k-nearest neighbour on classification accuracy in knn algorithm using machine learning". In: *Advances in Smart Communication and Imaging Systems: Select Proceedings of MedCom 2020*. Springer. 2021, S. 363–373.

[Sub+22] Raja Subramanian, Marisetty Murali, B. Deepak, P. Deepak, Hamsinipally Reddy und Raja Sudharsan. "Airline fare prediction using machine learning algorithms". In: *4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE. 2022, S. 877–884.

[Tzi+17] Konstantinos Tziridis, T. Kalampokas, George Papakostas und Kostas Diamantaras. "Airfare prices prediction using machine learning techniques". In: *25th European Signal Processing Conference (EUSIPCO)*. IEEE. 2017, S. 1036–1039.

[Wil18] Drew Wilimitis. *The Kernel Trick in Support Vector Classification*. URL: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f. Zugriffsdatum: 23 Mai 2023. 2018.

[WB19] Michael D. Wittman und Peter P. Belobaba. "Dynamic pricing mechanisms for the airline industry: A definitional framework". In: *Journal of Revenue and Pricing Management* 18 (2019), S. 100–106.