Benny Botsch

Maschinelles Lernen - Grundlagen und Anwendungen

Mit Beispielen in Python

MOREMEDIA
Springer Spektrum

Maschinelles Lernen – Grundlagen und Anwendungen

Benny Botsch

Maschinelles Lernen – Grundlagen und Anwendungen

Mit Beispielen in Python

Springer Spektrum

Benny Botsch
Berlin, Deutschland

ISBN 978-3-662-67276-1 ISBN 978-3-662-67277-8 (eBook)
https://doi.org/10.1007/978-3-662-67277-8

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Planung/Lektorat: Andreas Ruedinger
Springer Spektrum ist ein Imprint der eingetragenen Gesellschaft Springer-Verlag GmbH, DE und ist ein Teil von Springer Nature.
Die Anschrift der Gesellschaft ist: Heidelberger Platz 3, 14197 Berlin, Germany

Inhaltsverzeichnis

Einführung

1

Zusammenfassung

Maschinelles Lernen erfreut sich in vielen Bereichen immer größerer Beliebtheit. Das resultiert aufgrund der fortschreitenden Computertechnologie. Durch immer größere und schnellere Speichermedien können große Datenmengen lokal bzw. über ein Computernetzwerk effizient gespeichert und verarbeitet werden. Ebenso verbessert sich stetig die Performance der Prozessoren in CPU und GPU, wodurch sich die Trainingszeiten der Algorithmen immer weiter reduzieren lassen. Kap. 1 beschäftigt sich zunächst mit den verschiedenen Verfahren des maschinellen Lernens. Darüber hinaus werden einige Begriffe erläutert, welche in den späteren Kapiteln noch wichtig werden. In diesem Buch wird es neben den Übungen auch Programmierbeispiele geben, daher werden in Abschn. 1.8 einige Werkzeuge und Ressourcen vorgestellt, die für die Entwicklung des Python-Codes benötigt werden.

1.1 Was ist maschinelles Lernen

Maschinelles Lernen (ML) ist ein Teilgebiet der künstlichen Intelligenz (KI) und befasst sich mit der Entwicklung lernfähiger Systeme und Algorithmen. Die Algorithmen lernen anhand von ausreichend Daten verschiedene Zusammenhänge. Das somit entstandene Modell kann anschließend auf neue, unbekannte Daten derselben Art angewendet werden. Maschinelles Lernen wird mittlerweile in vielen Bereichen erfolgreich eingesetzt. Speziell, wenn bestimmte Prozesse zu kompliziert sind, um sie analytisch zu beschreiben. Eine der häufigsten Anwendungen von maschinellem Lernen ist die Erkennung von Mustern in großen Datenmengen. Diese Technologie wird in vielen Bereichen wie Finanzen, Marketing, Medizin und Wissenschaft eingesetzt, um die Ergebnisse aus Daten zu interpretieren. Ein weiteres Anwendungsgebiet ist die automatisierte Spracherkennung. Mit Hilfe von maschinellem Lernen können Computerprogramme dazu befähigt werden, menschliche Sprache zu verstehen und auf bestimmte Befehle zu reagieren. Diese Technologie wird häufig in der Kundenun-

B. Botsch, *Maschinelles Lernen – Grundlagen und Anwendungen*,
https://doi.org/10.1007/978-3-662-67277-8_1

Abb. 1.1 Schematische Darstellung des ML-Prozesses

terstützung, im Gesundheitswesen und in der Robotik eingesetzt. Schließlich wird maschinelles Lernen auch in der Computer Vision eingesetzt. Diese Technologie kann Computerprogramme dazu befähigen, Bilder und Videos zu analysieren und bestimmte Objekte zu erkennen. Diese Technologie wird in vielen Bereichen eingesetzt, z. B. in der autonomen Fahrzeugtechnologie, in der Sicherheitstechnologie und in der Medizintechnik.

Wie man in Abb. 1.1 sehen kann, werden für das Training eines Lernalgorithmus Eingabedaten benötigt. Das sind im Allgemeinen wert- bzw. zeitdiskrete Daten. Diese nutzt der jeweilige Algorithmus, um eine mathematische Funktion zu erlernen. Funktionen bzw. Abbildungen sind eindeutige Zuordnungen zwischen zwei Mengen X und Y.

$$f : X \to Y \tag{1.1}$$

Es wird öfters vorkommen, dass aufgrund von Messfehlern oder statistischen Effekten Widersprüche in den Daten auftreten. So kann z. B. einem Element aus X mehrere Elemente aus Y zugeordnet werden. Aus diesem Grund werden verschiedene Metriken (Verlustfunktion oder auch „loss function“ genannt) benötigt, wonach sich der Algorithmus bei seiner Optimierung richten kann.

Es gibt im Wesentlichen vier verschiedene Kategorien von Lernverfahren, welche in den nachfolgenden Kapiteln ausführlich beschrieben werden.

1.2 Überwachtes Lernen

Beim überwachten Lernen wird dem Algorithmus eine hinreichend große Menge von Eingabe- und Ausgabedaten zur Verfügung gestellt, siehe Abb. 1.2. Man spricht bei diesen Datensätzen von gelabelten oder markierten Datensätzen. Beim Training lernt der Algorithmus die Muster und Zusammenhänge bezüglich der Eingangsdaten und der Ausgangsdaten. Der gewünschte Funktionswert kann nun eine Klasse (Hund oder Katze) oder ein numerischer Wert (Aktienkurs) sein. Nach dem erfolgreichen Training kann das Modell dazu verwendet werden, um die Funktionswerte von neuen Eingangsdaten vorherzusagen.

Das überwachte Lernen lässt sich in zwei Problemstellungen einteilen, nämlich die Regression und die Klassifikation.

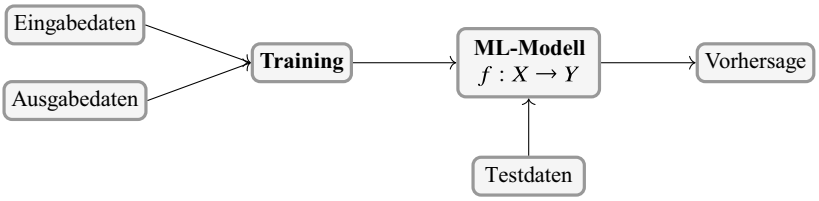

Abb. 1.2 Überwachtes Lernen mit Eingabe- und Ausgabedaten beim Training

1.2.1 Klassifikation und Regression

Die Hauptaufgabe der Klassifikation besteht darin, aus einer vordefinierten Liste von Möglichkeiten eine Klassenzugehörigkeit vorherzusagen. Die Klassifikation kann in eine binäre Klassifikation unterteilt werden, also die Unterscheidung zwischen zwei Klassen, oder in eine Mehrklassen-Klassifikation. Die Klassifikation von E-Mails als Spam oder kein Spam ist beispielsweise ein binäres Klassifikationsproblem. Ein Beispiel für die Mehrklassen-Klassifikation ist die Identifizierung von Objekten in Bildern. Das könnte z. B. die Unterscheidung von verschiedenen Tieren oder Pflanzen sein. Wollen wir nun Bilder klassifizieren und verwenden dabei die klassische Mustererkennung, so müssen zunächst sogenannte Feature (Merkmale) aus den Daten berechnet werden. Das können im Falle von Bildern z. B. Texturmerkmale oder eine Gradientenverteilung sein. Sind die Feature linear trennbar, wie in Abb. 1.3 dargestellt, dann können auch lineare Lernalgorithmen verwendet werden. Ist dies nicht der Fall, müssen Nichtlinearitäten dem Lernalgorithmus hinzugefügt werden. Nichtlinearitäten kann man beispielsweise durch polynominale Feature erzeugen. Dabei werden die Feature untereinander multipliziert und die so neu entstandenen Feature werden zu den bereits vorhandenen hinzugefügt.

Für Regressionsaufgaben versucht man dagegen, eine kontinuierliche Zahl vorherzusagen („floating-point“ in der Programmierung oder reelle Zahl in der Mathematik). Diese Form der Analyse schätzt die Koeffizienten einer Gleichung mit einem oder mehreren unabhängigen Features. Betrachtet man die lineare Regression, dann liefert diese eine Linie, welche die Abweichungen zwischen den vorhergesagten und

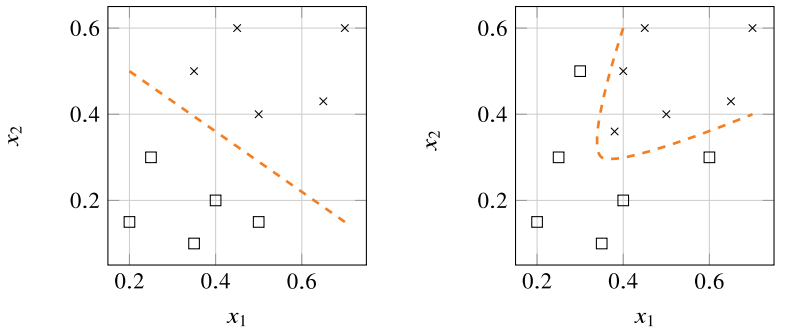

Abb. 1.3 Lineare und nichtlineare Trennung einer binären Klassifikation

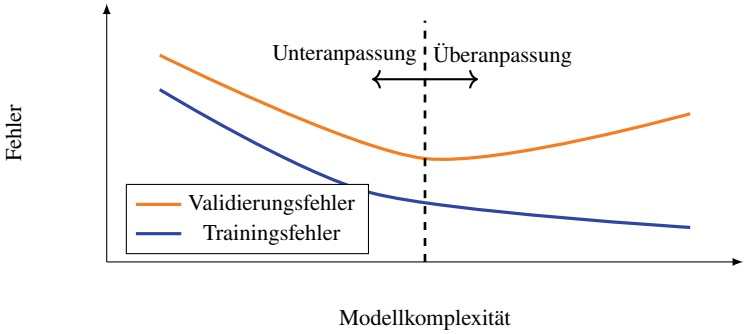

Abb. 1.4 Darstellung der Unter- bzw. Überanpassung anhand einer Lernkurve

den tatsächlichen Ausgabewerten minimiert. Die lineare Regression kann z. B. verwendet werden, um das durchschnittliche Jahresgehalt eines Arbeitnehmers (abhängige Variable) anhand von unabhängigen Variablen wie Alter, Bildung oder Berufserfahrung vorherzusagen. Eine weitere Anwendung ist die Analyse im Sport. So hängt beispielsweise die Anzahl der gewonnenen Spiele eines Fußballteams in einer Saison eng mit der durchschnittlichen Punktzahl der Mannschaft pro Spiel zusammen. Steigt die Anzahl der gewonnenen Spiele, sinkt die durchschnittliche Punktzahl, die der Gegner erzielt hat. Diese Beziehung kann verwendet werden, um vorherzusagen, wie viele Spiele die Teams gewinnen werden.

1.2.2 Generalisierung, Überanpassung und Unteranpassung

Das Ziel beim maschinellen Lernen besteht darin, Vorhersagen für neue unbekannte Daten zu treffen. Dazu muss zunächst ein Modell mit einem Trainingsdatensatz angelernt werden. Anschließend verwendet man einen Validierungsdatensatz, also Daten, die das Modell noch nicht kennt, um die Vorhersagequalität von neuen Daten zu überprüfen. Ist das Modell nun in der Lage, eine ähnlich gute Vorhersagequalität wie bei den Trainingsdaten zu erreichen, dann spricht man von der Generalisierung.

Die bestmögliche Generalisierung eines Modells erhalten wir, wenn die Komplexität des Modells eine ähnliche Mächtigkeit aufweist wie die des zu untersuchenden Problems bzw. der Funktion. Ist das Modell weniger komplex als die Funktion, dann spricht man von einer Unteranpassung („underfitting“), siehe Abb. 1.4. Verwendet man beispielsweise ein lineares Modell für einen Datensatz, welcher von einem Polynom dritten Grades stammt, dann wird man sowohl einen hohen Trainings- als auch Validierungsfehler bekommen. Wird die Komplexität nun stetig erhöht, verringert sich der Trainings- und auch der Validierungsfehler. Die Komplexität kann aber nicht ewig vergrößert werden, da ab einem bestimmten Punkt der Validierungsfehler anfängt zu steigen. Das ist der Punkt, wenn das Modell komplexer als die Funktion ist. Sollte zusätzlich Rauschen in den Daten vorhanden sein, dann würde das komplexe Modell dieses ebenfalls lernen. Die Folge ist eine schlechtere Generalisierung. Man spricht folglich von einer Überanpassung („overfitting“).

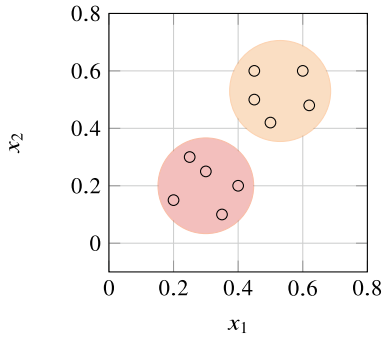

Abb. 1.5 Beispiel einer Clusteranalyse, bei der sich zwei signifikannte Cluster finden lassen

Bei allen Lernalgorithmen gibt es einen Kompromiss zwischen der Komplexität eines Modells, die Menge an vorhandenen Daten und dem Generalisierungsfehler bei neuen Daten. Nimmt die Menge an Daten zu, dann nimmt der Generalisierungsfehler ab. Erhöht sich die Komplexität des Modells, verringert sich zunächst der Generalisierungsfehler, nimmt dann aber dann wieder zu. Erhöht man allerdings die Anzahl der Trainingsdaten, dann kann ein Ansteigen des Generalisierungsfehlers reduziert werden [10].

1.3 Unüberwachtes Lernen

Im Vergleich zum überwachten Lernen sind beim unüberwachten Lernen die Ausgabedaten nicht bekannt. Es sind lediglich Eingabedaten vorhanden, wobei man versuchen möchte, Regelmäßigkeiten in den Eingabedaten zu finden. Man überprüft quasi den Eingaberaum auf mögliche Strukturen und ob diese öfters auftreten als andere. Ein Ansatz ist die sogenannte Dichteschätzung.

Eine Möglichkeit zur Dichteschätzung ist die Clusteranalyse (siehe Abb. 1.5). Man versucht, dabei Häufungen bzw. Cluster in den Eingabedaten zu finden. Ein Anwendungbeispiel ist die Verwendung im Bereich des Marketings. Unternehmen wollen ihre neuen Kunden möglichst genau analysieren und in bestimmte Zielgruppen einordnen. Es werden daraufhin ähnliche Kunden aus dem gesamten Kundenbestand identifiziert, um individuelle Werbestrategien für diesen Kunden zu entwickeln.

Eine weitere Anwendung ist die Anomalie-Erkennung. Unüberwachtes Lernen wird an dieser Stelle eingesetzt, um Abweichungen von bestimmten Normen in Echtzeit zu erkennen und direkt eingreifen zu können. Selbst komplexe, automatisierte Prozesse können so durchgehend überwacht werden.

1.4 Bestärkendes Lernen

Bestärkendes Lernen beschäftigt sich damit, automatisch optimale Entscheidungen mit der Zeit zu treffen. Dieses allgemeine Problem wird in vielen wissenschaftlichen und technischen Fachgebieten untersucht. Viele Aufgaben beim maschinellen Lernen

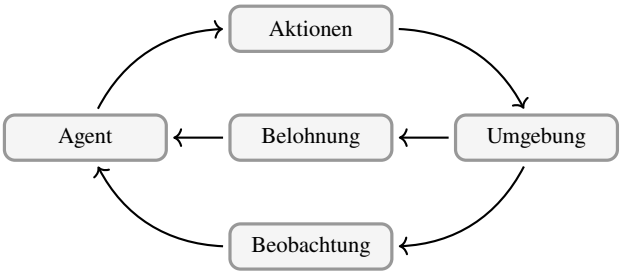

Abb. 1.6 Interaktion eines Agenten mit seiner Umgebung

besitzen eine verborgene zeitliche Dimension. Diese kann bei einem Produktivsystem allerdings zu Problemen führen. Bestärkendes Lernen ist ein Ansatz, der diese zusätzliche Dimension (im Allgemeinen die Zeit) beim Training berücksichtigt.

Im Gegensatz zum überwachten Lernen wissen wir nicht bei jedem Schritt genau, was das Richtige ist. Wir wissen nur das letztendliche Ziel, welches wir erreichen wollen. Wie man dieses Ziel erreicht, lernt der Algorithmus im Laufe der Zeit. Betrachten wir ein kleines Beispiel. Es sei ein Agent (Robotermaus) gegeben, der in einer Umgebung (Labyrinth) bestimmte Aktionen ausführen soll. Die Robotermaus kann sich dabei nach links bzw. rechts drehen oder sich vorwärts bewegen. Des Weiteren ist sie in der Lage, zu jedem Zeitpunkt den kompletten Zustand der Umgebung zu beobachten, siehe Abb. 1.6. Anhand der Beobachtung kann sie nun entscheiden, welche Aktion ausgeführt werden soll. Der Agent hat nun das Ziel, seine Belohnung zu maximieren. Es können ebenfalls negative Belohnungen auftreten, beispielsweise wenn der Agent Aktionen ausführt, die nicht erwünscht sind, oder in einen Zustand gelangt, den man nicht erreichen möchte. Bestärkendes Lernen bietet hier nun eine Lösung, die sich vom überwachten und unüberwachten Lernen unterscheidet. Es gibt keine vordefinierten Labels wie beim überwachten Lernen.

1.5 Teilüberwachtes Lernen

Teilüberwachtes Lernen ist ein Ansatz, der während des Trainings eine kleine Menge gelabelter Daten mit einer großen Menge nicht gelabelter Daten kombiniert. Teilüberwachtes Lernen fällt zwischen unüberwachtes Lernen (ohne gelabelten Trainingsdaten) und überwachtes Lernen (nur mit gelabelten Trainingsdaten).

In Abb. 1.7 ist das Prinzip des teilüberwachten Lernens dargestellt. Ausgangsbasis sind wenig gelabelte Daten. Mit diesen wird zunächst ein Modell trainiert, das kann z. B. ein SVM-Klassifizierer sein. Eine genaue Erläuterung zu diesem Klassifizierer finden Sie in Kap. 5. Dieses Modell wird dann anschließend verwendet, um die vielen verfügbaren ungelabelten Daten zu klassifizieren und somit zu kennzeichnen. Die neu gelabelten Daten werden mit den ursprünglich verfügbaren Daten kombiniert, um das Modell mit viel mehr Daten neu zu trainieren und somit ein besseres Modell zu erhalten [4].

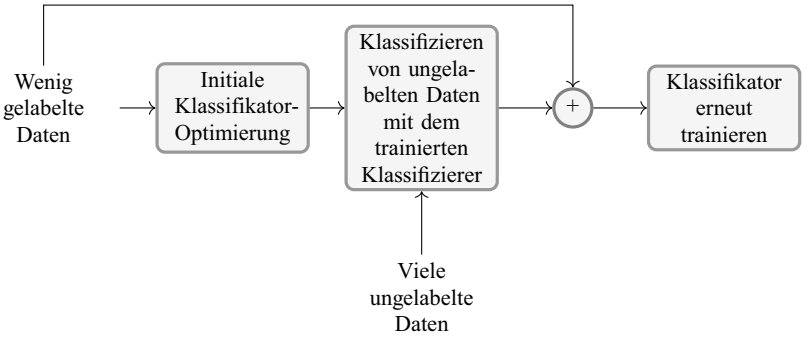

Abb. 1.7 Prinzip des teilüberwachten Lernens

1.6 Herausforderungen des maschinellen Lernens

Das Maschinelle Lernen hat in den letzten Jahren eine unglaubliche Entwicklung erfahren und wird zunehmend in vielen Bereichen eingesetzt, von der Spracherkennung bis hin zur Gesichtserkennung. Dennoch sind viele Herausforderungen und Schwierigkeiten vorhanden, die es zu überwinden gilt, um das volle Potenzial des maschinellen Lernens zu nutzen. Eine solche Herausforderung ist die Qualität der Daten, die für das Training von Modellen benötigt werden. Ungenaue oder unvollständige Daten können zu unzuverlässigen Ergebnissen führen und das Vertrauen in das System verringern. Ein weiteres Problem ist die Übertragbarkeit von Modellen auf neue Daten und Situationen. Modelle, die in einem bestimmten Kontext gut funktionieren, können in einem anderen Kontext völlig unbrauchbar sein. Schließlich ist auch die Interpretierbarkeit von Modellen ein wichtiger Faktor. Wenn wir nicht verstehen, wie ein Modell zu seinen Entscheidungen kommt, kann es schwierig sein, es in kritischen Anwendungen wie der Medizin oder dem Rechtswesen einzusetzen. Diese und andere Herausforderungen sind Gegenstand intensiver Forschung und Entwicklung, um das maschinelle Lernen weiter voranzutreiben und seine Anwendung in einer Vielzahl von Anwendungen zu verbessern.

1.6.1 Unzureichende Menge an Trainingsdaten

Ein zentrales Problem beim maschinellen Lernen ist die unzureichende Menge an Trainingsdaten. Im Allgemeinen gilt, dass der Erfolg von maschinellen Lernsystemen stark von der Menge und Qualität der Daten abhängt, die zur Verfügung stehen. Wenn nicht genügend Daten vorhanden sind, kann dies zu Überanpassung („overfitting“) führen, was bedeutet, dass das Modell zu komplex wird und nicht mehr in der Lage ist, neue Daten sinnvoll zu verarbeiten. Um dieses Problem zu lösen, gibt es verschiedene Ansätze. Einer davon ist das sogenannte Transfer Learning, bei dem Modelle auf einer großen Menge von Daten trainiert werden und anschließend auf

eine andere, kleinere Menge von Daten angewendet werden. Ein anderer Ansatz ist die Verwendung von generativen Modellen, die versuchen, neue Daten zu erzeugen, um die Menge der Trainingsdaten zu erweitern.

Ein weiteres Problem bei der Verwendung von unzureichenden Trainingsdaten ist das sogenannte Data Snooping. Hierbei werden die Modelle auf den Trainingsdaten überoptimiert, was dazu führen kann, dass das Modell auf zufällige Muster in den Daten reagiert, die keine tatsächliche Vorhersagekraft haben. Dies kann zu einer Verschlechterung der Leistung des Modells führen, wenn es auf neue Daten angewendet wird, da diese möglicherweise nicht die gleichen zufälligen Muster aufweisen. Eine Möglichkeit, Data Snooping zu vermeiden, besteht darin, die Daten in Trainings-, Validierungs- und Testdaten aufzuteilen. Die Trainingsdaten werden verwendet, um das Modell zu trainieren, während die Validierungsdaten verwendet werden, um die Hyperparameter des Modells (z. B. die Anzahl der versteckten Schichten in einem neuronalen Netzwerk) zu optimieren. Die Testdaten werden schließlich verwendet, um die Leistung des Modells auf neuen Daten zu bewerten.

1.6.2 Nicht repräsentative Trainingsdaten

Die Qualität der Trainingsdaten ist ein kritischer Faktor für den Erfolg von maschinellen Lernsystemen. Eine der Herausforderungen besteht darin, Trainingsdaten zu finden, die repräsentativ für die gesamte Datenverteilung sind. Wenn die Trainingsdaten nicht repräsentativ sind, kann dies zu Verzerrungen und schlechter Leistung des Modells führen. Ein Grund dafür, dass nicht repräsentative Trainingsdaten zu einer schlechten Leistung von maschinellen Lernsystemen führen können, ist der sogenannte Bias. Dies bezieht sich auf eine systematische Abweichung des Modells von der wahren Beziehung zwischen den Eingabedaten und den Ausgaben. Wenn das Modell auf Basis von Trainingsdaten trainiert wird, die nicht die volle Bandbreite der möglichen Eingabedaten abdecken, kann es zu einem Verzerrungseffekt kommen. Das Modell neigt dazu, die Eigenschaften der Trainingsdaten zu generalisieren, was zu Vorhersagen führt, die nicht auf die breitere Population von Eingabedaten zutreffen. Ein Beispiel für einen Bias-Effekt tritt auf, wenn eine Klassifikationsaufgabe mit einem ungleichmäßig verteilten Klassenverhältnis behandelt wird. Wenn die Trainingsdaten hauptsächlich eine Klasse enthalten und die andere Klasse nur sehr selten vorkommt, kann das Modell dazu neigen, die häufige Klasse zu bevorzugen und die seltenere Klasse zu ignorieren.

Ein weiterer Effekt von nicht repräsentativen Trainingsdaten ist die Varianz. Dieser Effekt tritt auf, wenn das Modell sehr empfindlich auf kleine Unterschiede in den Trainingsdaten reagiert. Wenn die Trainingsdaten nicht repräsentativ sind, kann es zu einer hohen Varianz kommen, da das Modell auf Muster in den Trainingsdaten reagiert, die nicht in der breiteren Population von Eingabedaten vorkommen. Dadurch wird das Modell anfällig für Überanpassung, was bedeutet, dass es sehr gut auf den Trainingsdaten abschneidet, aber schlecht auf neuen Daten. Um diese Effekte zu minimieren, gibt es verschiedene Ansätze. Ein Ansatz besteht darin, mehr Trainingsdaten zu sammeln, um sicherzustellen, dass die Trainingsdaten repräsentativ für die gesamte Datenverteilung sind. Dies kann jedoch schwierig sein, insbesondere wenn

die Daten teuer oder schwierig zu sammeln sind. Ein weiterer Ansatz besteht darin, die vorhandenen Trainingsdaten zu erweitern, indem synthetische Daten generiert werden, um sicherzustellen, dass das Modell auf eine breitere Palette von Eingabedaten reagieren kann. Die Generierung von synthetischen Daten kann durch Techniken wie Data Augmentation erreicht werden, bei der zufällige Transformationen auf den vorhandenen Trainingsdaten angewendet werden, um neue Datenpunkte zu generieren.

Es gibt auch Ansätze, die speziell darauf abzielen, den Bias-Effekt zu minimieren. Einer dieser Ansätze besteht darin, eine Gewichtung der Trainingsdaten vorzunehmen, bei der Datenpunkten, die in der breiteren Population seltener vorkommen, ein höheres Gewicht zugewiesen wird. Dadurch wird sichergestellt, dass das Modell gleichermaßen auf alle relevanten Kategorien von Eingabedaten reagiert.

1.6.3 Daten von schlechter Qualität

Ein weiteres Problem sind inkonsistente oder unvollständige Daten. Inkonsistente Daten können aufgrund von Fehlern bei der Datenerfassung oder bei der Integration von Daten aus verschiedenen Quellen auftreten. Unvollständige Daten können beispielsweise auftreten, wenn eine wichtige Eingabevariable nicht erfasst wurde oder wenn Daten nur für einen Teil der Beobachtungszeit verfügbar sind. Inkonsistente oder unvollständige Daten können dazu führen, dass das Modell falsche Beziehungen zwischen den Eingabevariablen und der Zielvariable herstellt und daher ungenaue Vorhersagen trifft.

Ein weiteres Problem sind Daten von geringer Qualität, bei denen die Messgenauigkeit oder die Messmethode selbst unsicher oder fehlerhaft ist. Daten von geringer Qualität können dazu führen, dass das Modell falsche Beziehungen zwischen den Eingabevariablen und der Zielvariable herstellt und daher ungenaue Vorhersagen trifft.

Um die Auswirkungen von Daten von schlechter Qualität auf das maschinelle Lernen zu minimieren, gibt es verschiedene Schritte, die unternommen werden können. Eine Möglichkeit besteht darin, die Daten sorgfältig zu bereinigen, indem Ausreißer und fehlende Daten entfernt werden. Eine andere Möglichkeit besteht darin, die Daten auf Inkonsistenzen zu prüfen und diese zu korrigieren. Darüber hinaus können verschiedene Techniken eingesetzt werden, um die Qualität der Daten zu verbessern, wie beispielsweise die Erhöhung der Messgenauigkeit oder die Verbesserung der Datenerfassungsmethoden.

1.6.4 Irrelevante Merkmale

Das maschinelle Lernen nutzt die Merkmale oder Eigenschaften von Daten, um Vorhersagen bzw. Klassifizierungen durchzuführen. Allerdings können irrelevante Merkmale, die keinen Einfluss auf die Zielvariable haben, die Leistung des Modells erheblich beeinträchtigen. Irrelevante Merkmale können auf verschiedene Arten ent-

stehen. Ein häufiges Problem sind redundante Merkmale, die die gleiche Information wie andere Merkmale enthalten. Redundante Merkmale können dazu führen, dass das Modell überangepasst wird, da sie das Modell dazu verleiten, bestimmte Merkmale stärker zu gewichten als andere, obwohl sie die gleiche Information enthalten. Ein weiteres Problem sind uninformative Merkmale, die keinen Einfluss auf die Zielvariable haben und daher keine Vorhersagekraft besitzen. Uninformative Merkmale können dazu führen, dass das Modell weniger effektiv wird, da es sich auf irrelevante Informationen konzentriert.

Ein weiteres Problem sind kollineare Merkmale, die hoch miteinander korreliert sind. Kollinearität kann dazu führen, dass das Modell das Vorhandensein eines Merkmals überschätzt, wenn es in Verbindung mit anderen Merkmalen steht. Dies kann zu ungenauen Vorhersagen führen, wenn sich die Beziehung zwischen dem Merkmal und der Zielvariable ändert. Eine Möglichkeit, dieses Problem zu lösen, besteht darin, die Bedeutung der Merkmale zu gewichten, um diejenigen zu identifizieren, die am stärksten zur Zielvariable beitragen. Man schätzt sogenannte Merkmalsgewichte, indem eine Regressionsanalyse durchgeführt wird. Die Merkmalsgewichte können dann verwendet werden, um die wichtigen Merkmale zu identifizieren und die unwichtigen zu entfernen oder zu transformieren.

1.6.5 Explainable Artificial Intelligence

Das maschinelle Lernen hat in den letzten Jahren eine exponentielle Entwicklung erfahren und wird in vielen Bereichen eingesetzt, von der Spracherkennung bis hin zur Gesichtserkennung. Während diese Technologie enorme Chancen für die Gesellschaft bietet, sind auch zahlreiche Risiken und Herausforderungen damit verbunden. Eine dieser Herausforderungen betrifft die Interpretierbarkeit von Maschinen, auch bekannt als Explainable Artificial Intelligence (XAI).

XAI bezieht sich auf die Fähigkeit von Maschinen, ihre Entscheidungen und Handlungen auf eine verständliche und nachvollziehbare Weise zu erklären. Es geht darum sicherzustellen, dass Benutzer und andere Interessengruppen verstehen können, wie Maschinen zu ihren Entscheidungen kommen und welche Faktoren diese Entscheidungen beeinflussen. Dadurch können Benutzer das Vertrauen in die Maschinen und ihre Entscheidungen stärken und die Anwendung von Maschinen in kritischen Anwendungen wie der Medizin oder dem Rechtswesen verbessern.

Eine der wichtigsten Herausforderungen bei XAI ist die Komplexität von Maschinen. Moderne Maschinen sind oft sehr komplex und verwenden Algorithmen, die schwer zu verstehen sind. Um dieses Problem zu lösen, müssen Entwickler Techniken wie die Erklärbarkeit von Modellen und die Aufzeichnung von Entscheidungen verwenden, um sicherzustellen, dass Benutzer die Entscheidungen von Maschinen nachvollziehen können. Eine Möglichkeit, dies zu tun, besteht darin, Entscheidungsbäume zu erstellen, die zeigen, wie Maschinen zu ihren Entscheidungen kommen.

Ein weiteres Problem bei XAI ist die Balance zwischen Komplexität und Verständlichkeit. Um sicherzustellen, dass Maschinen erklärt werden können, müssen Entwickler ihre Modelle vereinfachen und abstrahieren. Eine Möglichkeit, dies zu

tun, besteht darin, Techniken wie Feature Selection zu verwenden, so dass nur relevante Informationen in das Modell einbezogen werden.

Ein weiteres Problem bei XAI ist die Sicherheit von Maschinen. Wenn wir Maschinen erklären können, wie sie zu ihren Entscheidungen kommen, kann dies auch dazu führen, dass diese Entscheidungen manipuliert werden können. Um dieses Problem zu lösen, müssen Entwickler sicherstellen, dass ihre Modelle sicher und robust sind und dass sie gegen Angriffe wie Adversarial Attacks geschützt sind.

XAI bezieht sich auch auf die Verantwortlichkeit von Entwicklern. Entwickler müssen sicherstellen, dass ihre Modelle ethisch und verantwortungsvoll eingesetzt werden und dass sie gegen Vorurteile und Diskriminierung geschützt sind. Dazu gehört auch die Gewährleistung der Privatsphäre und des Datenschutzes von Benutzern. Entwickler müssen sicherstellen, dass die Daten, die von Maschinen gesammelt werden, sicher und geschützt sind, um die Privatsphäre von Benutzern zu schützen.

Ein weiterer wichtiger Aspekt von XAI ist die Interaktion zwischen Benutzern und Maschinen. Benutzer müssen in der Lage sein, mit Maschinen zu interagieren und Feedback zu geben, um sicherzustellen, dass die Maschinen ihre Bedürfnisse und Anforderungen verstehen. Hierzu können Techniken wie Natural Language Processing (NLP) oder Dialogsysteme eingesetzt werden, um eine natürlichere Interaktion zwischen Benutzern und Maschinen zu ermöglichen.

Insgesamt gibt es viele Herausforderungen bei XAI, aber es gibt auch viele Möglichkeiten, um diese Herausforderungen zu bewältigen. Ein wichtiger Schritt besteht darin sicherzustellen, dass Entwickler sich der Bedeutung von XAI bewusst sind und dass sie entsprechende Techniken und Tools verwenden, so dass ihre Maschinen erklärt werden können. Darüber hinaus müssen Regulierungsbehörden und andere Interessengruppen sicherstellen, dass Maschinen ethisch und verantwortungsvoll eingesetzt werden und dass sie gegen Vorurteile und Diskriminierung geschützt sind. Durch eine gemeinsame Anstrengung können wir sicherstellen, dass Maschinen sicher, vertrauenswürdig und erklärt werden können und dass sie uns bei der Lösung wichtiger gesellschaftlicher Herausforderungen unterstützen können.

1.7 Bewertung und Vergleich von Algorithmen

Dieses Kapitel widmet sich der Frage, wie der Fehler eines Algorithmus bewertet werden kann. Speziell möchte man wissen, ob der zu erwartende Fehler bei neuen Daten sich innerhalb einer gewissen Toleranz befindet. Der Fehler ist grundsätzlich bei den Trainingsdaten immer kleiner als bei den Testdaten. Aufgrund dessen eignet sich der Trainingsfehler nicht für den Vergleich von Algorithmen. Es wird ein sogenannter Validierungsdatensatz benötigt, der sich vom Trainingsdatensatz unterscheidet.

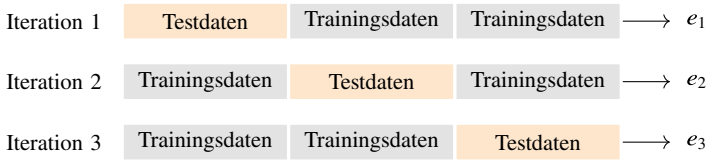

Abb. 1.8 Darstellung einer 3-fachen Kreuzvalidierung. Bei jeder Iteration resultiert ein Fehler e_i. Der Gesamtfehler errechnet sich als Durchschnitt aus den Einzelfehlern

1.7.1 Kreuzvalidierung

Die Kreuzvalidierung eignet sich zur Bewertung der Leistung eines Algorithmus. Es wird ein Teil des vorhandenen Datensatzes separiert, um damit die Güte der Vorhersage zu überprüfen. Jeder Durchlauf der Kreuzvalidierung umfasst eine zufällige Aufteilung des originalen Datensatzes in einem Trainings- und Testdatensatz. Mit dem Trainingsdatensatz wird der Algorithmus trainiert. Der Testdatensatz wird zur Bewertung der Leistung verwendet. Dieser Vorgang wird nun mehrmals wiederholt. Daraufhin resultiert ein mittlerer Kreuzvalidierungsfehler als Leistungsindikator.

Der Kreuzvalidierung stehen verschiedene Techniken zur Verfügung. Am häufigsten wird die k-fache Kreuzvalidierung verwendet. Die Daten werden zufällig in k ausgewählte Teilmengen ungefähr gleicher Größe aufgeteilt. Dabei wird nun eine Teilmenge zum Validieren verwendet und der Rest zum Trainieren. Dieser Vorgang wird k-mal wiederholt, so dass jede Teilmenge einmal zum Validieren verwendet wird, siehe Abb. 1.8.

Die Daten kann man ebenso für Training und Validierung in genau zwei Teilmengen aufteilen. Diese Technik nennt sich Holdout. Hier werden Training und Validierung nur einmal ausgeführt, was die Ausführungszeit bei großen Datensätzen beachtlich verkürzt, allerdings sollte der ausgegebene Fehler mit Bedacht interpretiert werden.

Eine weitere Technik ist die Leave-One-Out-Kreuzvaliderung. Die Leave-One-Out-Kreuzvalidierung ist eine Technik zur Bewertung der Leistung von ML-Modellen. Im Wesentlichen geht es darum, ein Modell auf einer Stichprobe von Daten zu trainieren und es dann auf einer anderen Stichprobe zu testen, um zu sehen, wie gut es generalisiert. Diese Prozedur wird für jede einzelne Beobachtung in den Daten durchgeführt, wobei jede Beobachtung einmal aus der Stichprobe entfernt wird und das Modell auf den verbleibenden Daten trainiert und getestet wird. Ein wichtiger Vorteil der Leave-One-Out-Kreuzvaliderung ist die Genauigkeit, da sie eine große Anzahl von Tests durchführt. Bei einer Stichprobe von n Beobachtungen wird das Modell n-mal trainiert und getestet, wobei jeweils eine andere Beobachtung aus der Stichprobe entfernt wird. Dies ermöglicht es, eine sehr genaue Schätzung der Vorhersageleistung des Modells zu erhalten. Ein Nachteil ist jedoch, dass sie sehr rechenintensiv sein kann, insbesondere bei großen Datensätzen. Da jedes Modell n-mal trainiert und getestet wird, kann dies sehr zeitaufwendig sein. Aus diesem Grund wird diese Technik in der Praxis oft nur bei kleineren Datensätzen verwendet.

Die Kreuzvalidierung ist eine rechenintensive Methode, da das Training und die Validierung mehrmals durchgeführt werden. Vor allem ist die Kreuzvaliderung in

Abb. 1.9 Darstellung der Konfusionsmatrix für eine binäre Klassifikation

		Vorhergesagte Klasse		
		Positive	Negative	Total
Tatsächliche Klasse	Positive	t_p	f_n	$t_p + f_n$
	Negative	f_p	t_n	$f_p + t_n$
	Total	$t_p + f_p$	$f_n + t_n$	n

der Modellentwicklung ein wichtiger Faktor, da eine mögliche Unter- bzw. Überanpassung identifiziert werden kann.

1.7.2 Messfehler

Um Modelle evaluieren zu können, werden sogenannte Metriken benötigt. Metriken geben einem Informationen darüber, wie gut die Performance eines Modells ist. Wir betrachten zunächst einige Metriken im Bereich der Klassifikation. Zur Bewertung von Klassifikationsergebnissen wird häufig die Konfusionsmatrix herangezogen. Die Zeilen stellen den tatsächlichen Wert dar, während die Spalten den vorhergesagten Wert ausdrücken. In Abb. 1.9 ist die allgemeine Konfusionsmatrix dargestellt. Diese beinhaltet als Elemente die Anzahl an korrekten positiven (True Positives t_p), korrekten negativen (True Negatives t_n), falschen positiven (False Positives f_p) und falschen negativen (False Negatives f_n) Vorhersagen. Durch die Konfusionsmatrix lassen sich nun verschiedene Metriken ableiten. Die wohl einfachste Performance-Metrik für eine Klassifikation ist die Accuracy (Genauigkeit). Diese lässt sich mit der Gleichung

$$\text{acc} = \frac{t_p + t_n}{t_p + f_p + f_n + t_n} \tag{1.2}$$

berechnen und gibt den Anteil an richtigen Vorhersagen an. Je näher der Wert an 1 liegt, desto besser.

Die Precision gibt den Anteil an richtig vorhergesagten positiven Ergebnissen (t_p) bezogen auf die Gesamtheit aller als positiv vorhergesagten Ergebnisse an. Zum Beispiel sollte beim Spam-Filter die Precision sehr hoch sein, da wichtige E-Mails besser nicht als Spam klassifiziert werden sollten.

$$\text{precision} = \frac{t_p}{t_p + f_p} \tag{1.3}$$

Der Recall gibt den Anteil der korrekt als positiv klassifizierten Ergebnisse (t_p) bezogen auf die Gesamtheit der tatsächlich positiven Ergebnisse an. Bezogen auf unseren Spam-Filter bedeutet ein hoher Recall, dass man nicht unbedingt jede Spam-Mail herausfiltern muss.

$$\text{recall} = \frac{t_p}{t_p + f_n} \tag{1.4}$$

Die Metriken haben unterschiedliche Stärken und Schwächen. Die Genauigkeit (acc) ist beispielsweise recht empfindlich gegenüber Klassenungleichgewichten. Der Matthews-Korrelationskoeffizient (mcc) wird als Maß für die Qualität von binären

und mehrklassigen Klassifikationen verwendet. Es gilt allgemein als ausgewogenes Maß, das auch bei sehr unterschiedlichen Klassengrößen eingesetzt werden kann. Der mcc ist im Wesentlichen ein Korrelationskoeffizientwert zwischen -1 und $+1$. Ein Koeffizient von $+1$ und -1 steht für eine perfekte Vorhersage, 0 für eine durchschnittliche zufällige Vorhersage.

$$\text{mcc} = \frac{t_p t_n - f_p f_n}{\sqrt{(t_p + f_p)(t_p + f_n)(t_n + f_p)(t_n + f_n)}} \tag{1.5}$$

Bisher wurden die Metriken für Klassifizierungsprobleme diskutiert. Jetzt werden verschiedene Regressionsmetriken betrachtet. Der mittlere absolute Fehler (mae) ist eine der Regressionsmetriken. Zuerst wird die Differenz zwischen dem tatsächlichen Wert y und dem vorhergesagten Wert $\hat{y}$ berechnet. Dann ergibt der Durchschnitt der Absolutwerte dieser Differenzen den mae. Je näher der Wert an 0 liegt, desto besser ist die Qualität des Modells.

$$\text{mae} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{1.6}$$

Der mittlere quadratische Fehler (*mse*) ist eine weitere beliebte Metrik. Wie bei *mae* wird die Differenz zwischen den realen Werten y und vorhergesagten Werten $\hat{y}$ berechnet. Aber in diesem Fall werden die Differenzen quadriert. Der Wert ist immer nicht negativ, und Werte, die näher an 0 liegen, sind besser.

$$\text{mse} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{1.7}$$

Beispiel

Gegeben sei eine Konfusionsmatrix mit zwei Klassen. Ein trainiertes Modell kann von 35 Hunden 30 exakt vorhersagen. Auf der anderen Seite wurden von maximal 4 Katzen 2 richtig vorhergesagt.

		Vorhergesagte Klasse		
		Hund	Katze	Total
Tatsächliche Klasse	Hund	30	5	35
	Katze	2	2	4
	Total	32	7	39

Berechnen wir zunächst die Genauigkeit für dieses Beispiel. Wir erreichen dabei einen Wert von 0.82 und man könnte nun vermuten, dass das Modell gut zwischen Hund und Katze unterscheiden kann. Allerdings sind die Klassen sehr unausgewogen. In diesem Beispiel sind deutlich mehr Daten bei den Hunden als bei den Katzen vorhanden.

$$\text{acc} = \frac{t_p + t_n}{t_p + f_p + f_n + t_n} = \frac{30 + 2}{39} = 0.82$$

Schauen wir uns nun den Matthews-Korrelationskoeffizienten an. Wir erreichen jetzt nur noch einen Wert von 0.28, welcher deutlich schlechter als die Genauigkeit ist.

$$\text{mcc} = \frac{t_p t_n - f_p f_n}{\sqrt{(t_p + f_p)(t_p + f_n)(t_n + f_p)(t_n + f_n)}} = \frac{60 - 10}{\sqrt{32 \cdot 35 \cdot 4 \cdot 7}} = 0.28$$

Es stellt sich nun die Frage, welche Metrik man für die Beurteilung eines Modells heranziehen sollte. Das hängt ganz von der Problemstellung ab. In diesem Beispiel sind die Klassen sehr unausgewogen. Daher bieten sich hier Metriken an, welche diese Unausgewogenheit berücksichtigen, wie beispielsweise der Matthews-Korrelationskoeffizient. ◀

1.7.3 Intervallschätzung

Intervallschätzung ist ein statistisches Verfahren, das verwendet wird, um eine Schätzung für einen unbekannten Parameter durchzuführen, wie z. B. den Erwartungswert oder die Standardabweichung einer Zufallsvariable. In der Praxis wird Intervallschätzung oft verwendet, um Vorhersagen für eine Zielvariable zu machen, wie z. B. die Wahrscheinlichkeit, dass ein Kunde ein bestimmtes Produkt kauft. Anstatt nur eine Vorhersage für die Wahrscheinlichkeit zu machen, wird eine Intervallschätzung erstellt, die angibt, wie sicher die Vorhersage ist.

Es gibt verschiedene Methoden zur Berechnung von Intervallschätzungen, die je nach Modell und Daten variieren können. Eine häufig verwendete Methode ist die Konfidenzintervallschätzung, die auf der Normalverteilung basiert. Das Konfidenzintervall gibt einen Bereich an, innerhalb dessen der wahre Wert des Parameters mit einer bestimmten Wahrscheinlichkeit liegt. Die Wahrscheinlichkeit wird durch das Konfidenzniveau angegeben, das üblicherweise als 95 % oder 99 % gewählt wird.

Eine weitere Methode zur Berechnung von Intervallschätzungen ist die Bootstrap-Methode, die auf der Wiederverwendung der vorhandenen Daten basiert. Die Bootstrap-Methode schätzt die Verteilung des Parameters, indem sie zufällig Stichproben mit Wiederholungen aus den vorhandenen Daten zieht und die Schätzwerte für jede Stichprobe berechnet. Das Konfidenzintervall kann dann aus der Verteilung der Schätzwerte abgeleitet werden.

Es ist wichtig zu beachten, dass Intervallschätzungen keine genauen Vorhersagen liefern, sondern nur eine Schätzung der Unsicherheit der Vorhersage darstellen. Je größer das Konfidenzniveau ist, desto breiter wird das Intervall und desto unsicherer wird die Vorhersage. Eine zu hohe Unsicherheit kann jedoch dazu führen, dass das Modell nicht sinnvoll verwendet werden kann.

1.7.4 Hypothesenprüfung

Die Hypothesenprüfung ist ein grundlegendes statistisches Konzept, das auch beim maschinellen Lernen Anwendung findet. Bei der Hypothesenprüfung geht es darum,

eine statistische Aussage darüber zu treffen, ob ein bestimmtes Merkmal oder eine bestimmte Eigenschaft in einer Population vorhanden ist oder nicht. Im maschinellen Lernen wird die Hypothesenprüfung häufig eingesetzt, um zu bestimmen, ob ein Modell signifikant bessere Ergebnisse liefert als ein anderes Modell oder ob ein bestimmtes Merkmal für die Vorhersagekraft des Modells wichtig ist.

Die Hypothesenprüfung basiert auf der Formulierung von Null- und Alternativhypothesen. Die Nullhypothese besagt, dass es keinen signifikanten Unterschied oder keinen Zusammenhang zwischen den untersuchten Variablen gibt, während die Alternativhypothese besagt, dass es einen signifikanten Unterschied oder Zusammenhang gibt. Das Ziel der Hypothesenprüfung besteht darin festzustellen, ob die Daten ausreichende Beweise für die Ablehnung der Nullhypothese zugunsten der Alternativhypothese liefern.

Es gibt verschiedene statistische Tests, die für die Hypothesenprüfung verwendet werden können, wie z. B. der t-Test oder der Chi-Quadrat-Test. Diese Tests verwenden eine statistische Metrik, um den Grad der Abweichung der beobachteten Daten von den erwarteten Daten unter der Nullhypothese zu messen. Das Ergebnis des Tests wird in der Regel durch einen p-Wert ausgedrückt, der angibt, wie wahrscheinlich es ist, die beobachteten Daten unter der Annahme der Nullhypothese zu erhalten. Ein kleiner p-Wert deutet darauf hin, dass die Daten stark genug sind, um die Nullhypothese abzulehnen.

1.8 Werkzeuge und Ressourcen

In diesem Buch wird es neben den Übungen ebenfalls Programmierbeispiele geben. Es wurde sich hier für die Programmiersprache Python entschieden, da hierfür besonders viele Bibliotheken im Bereich des maschinellen Lernens vorhanden sind. Zunächst muss dafür ein Arbeitsumfeld in Form einer Entwicklungsumgebung für die Programmierung geschaffen werden. In den folgenden Kapiteln werden Sie erfahren, wie Sie Python auf Ihren Rechner installieren und die verschiedenen Python-Bibliotheken verwenden können.

1.8.1 Installation von Python mit Anaconda

Eine einfache Installation von Python ist die Verwendung der Anaconda-Distribution. Anaconda ist eine freie und betriebssystemunabhängige Open-Source -Software. Darüber hinaus bietet Anaconda einige hilfreiche Eigenschaften. Viele Data-Science-Module wie NumPy, scikit-learn, TensorFlow oder Keras können schnell und einfach in unterschiedlichen Entwicklungsumgebungen installiert werden. Ebenso sind viele Pakete in Anaconda integriert, die einem beim Entwickeln von Applikationen unterstützen.

Sie können die Anaconda-Distribution auf der Seite https://www.anaconda.com/products/distribution herunterladen. In diesem Buch wird durchgängig die Python-Version 3 verwendet, daher sollten Sie auch die Anaconda-Version für Pyhton 3.x

herunterladen. Nach der Installation starten Sie den Anaconda-Navigator. Dieses Programm erlaubt die Verwaltung der verschiedenen Python-Pakete. Auf der Startseite haben Sie die Möglichkeit, unterschiedliche Applikationen unter der aktuellen Umgebung (Environment) zu starten, wie z. B. jupyter notebook, qtconsole oder spyder.

In dem Reiter Environments können Sie die verschiedenen Umgebungen anschauen. Die Base ist die per Default von Anaconda angelegte Python-Umgebung. Auf Create können weitere Python-Umgebungen hinzugefügt werden. Dadurch können unterschiedliche Entwicklungen mit individuellen Versionen der Python-Pakete durchgeführt werden. Die Python-Pakete werden ebenfalls in diesem Reiter hinzugefügt, dazu gibt man den Modulnamen in das Feld Search Packages ein, aktiviert dieses und klickt anschließend auf Apply.

Eine weitere Installationsmöglichkeit ist die Verwendung von Kommandos in einem Terminal. Das Terminal rufen Sie in Windows beispielsweise mit cmd im Startmenü auf. In Anaconda erzeugen Sie eine neue virtuelle Python-Umgebung, indem folgender Befehl in das Terminal eingegeben wird:

```
conda create --name machinelearning_env python=3.7
```

Damit ist nun eine neue Umgebung mit den Namen `machinelearning_env` und der Python-Version 3.7 vorhanden. Mit dem Befehl

```
conda activate machinelearning_env
```

wird die Umgebung aktiviert. Nach dem Beenden der Arbeiten sollte die selektierte Umgebung wieder deaktiviert werden:

```
conda deactivate
```

Python-Pakete können Sie wie folgt installieren:

```
conda install numpy
```

Dadurch wird das NumPy-Paket installiert. Es können auch mehrere Pakete auf einmal installiert werden, indem man die verschiedenen Paketnamen mit einem Leerzeichen aneinanderreiht.

1.8.2 Entwicklungsumgebungen

Python-Code kann man prinzipiell mit einem Texteditor (z. B. Notepad++) editieren und im Terminal ausführen. Je größer und komplexer die Projekte werden, desto mühsamer wird die Entwicklung mit diesen einfachen Mitteln. Ab einem gewissen Grad an Komplexität wird eine programmierfreundliche Umgebung gebraucht. Eine IDE (Integrated Development Environment) unterstützt einen bei der Python-Entwicklung.

Mit Microsoft Visual Studio Code können sämtliche Programmiersprachen (C#, C++, Java, Python, JSON, PHP) editiert werden. Zusätzlich bietet Visual Studio

Code zahlreiche Erweiterungen und eine Debugging-Funktion an. Die Python-Erweiterung wird wie folgt hinzugefügt:

1. Wählen Sie in der Visual Studio Code-Menüleiste Ansicht>Erweiterungen aus, um die Ansicht „Erweiterungen“ zu öffnen. Dort werden einem die empfohlenen und beliebtesten Erweiterungen im Marketplace angezeigt. Zusätzlich werden die bereits installierten und aktivierten Erweiterungen aufgelistet.
2. Geben Sie python in das Suchfeld oben in der Ansicht „Erweiterungen“ ein, um die Liste der verfügbaren Erweiterungen zu filtern.
3. Wählen Sie die von Microsoft veröffentlichte Python-Erweiterung aus. Die Details zu dieser Erweiterung werden in einem Registerkartenbereich auf der rechten Seite angezeigt.
4. Wählen Sie im Bereich „Erweiterungen“ oder im Hauptbereich Installieren aus.

Wenn Sie bereits Anaconda installiert haben, dann können Sie die Anaconda-Umgebung mit Visual Studio Code verknüpfen. Drücken Sie CTRL+SHIFT+P, um die Einstellungen zu öffnen. Geben Sie „Select Interpreter“ in die Suchleiste und klicken anschließend auf „Python: Select Interpreter“. Klicken Sie nun auf „Enter Interpreter Path“ und dann auf „Find“. Dadurch wird ein Datei-Explorer-Fenster geöffnet, in dem Sie zum Speicherort des Python-Interpreters navigieren können. Sie können entweder die Basisinstallation von Anaconda „python.exe“ auswählen, oder Sie navigieren zum Ordner „envs“ und öffnen den Ordner für eine Umgebung und wählen die „python.exe“ aus. Die untere rechte Ecke von Visual Studio Code sollte jetzt aktualisiert werden und den Namen und die Python-Version der ausgewählten Umgebung anzeigen. Sollte bei der Ausführung eines Python-Skripts Fehler erscheinen, müssen eventuell noch Umgebungsvariablen angelegt werden. Geben Sie dazu Umgebungsvaiablen im Windows-Startmenü ein und wählen „Umgebungsvariablen bearbeiten“ aus. Nun gibt es die Möglichkeit, die Umgebungsvariablen zu bearbeiten. Wählen Sie „Path“ bei Systemvariablen und klicken auf „Bearbeiten“. Fügen Sie nun folgende zwei Pfade hinzu:

```
C:\Users\your-user-name\Anaconda3\
C:\Users\your-user-name\Anaconda3\Scripts
```

Darüber hinaus gibt es noch die Möglichkeit, die in Anaconda bereits integrierte IDE namens Spyder zu nutzen. Spyder steht für Scientific Python Development Envirenment. Die Umgebung ermöglicht das Editieren und Bearbeiten von Python-Code und bietet Funktionen zur Visualisierung und zum Debugging an. Sie können Spyder am schnellsten über den Anaconda-Navigator starten.

1.8.3 Python-Bibliotheken

In Python gibt es eine große Anzahl von Programmbibliotheken, die einem bei der Entwicklung die Arbeit erleichtern. In diesem Abschnitt werden einige dieser Bibliotheken kurz vorgestellt.