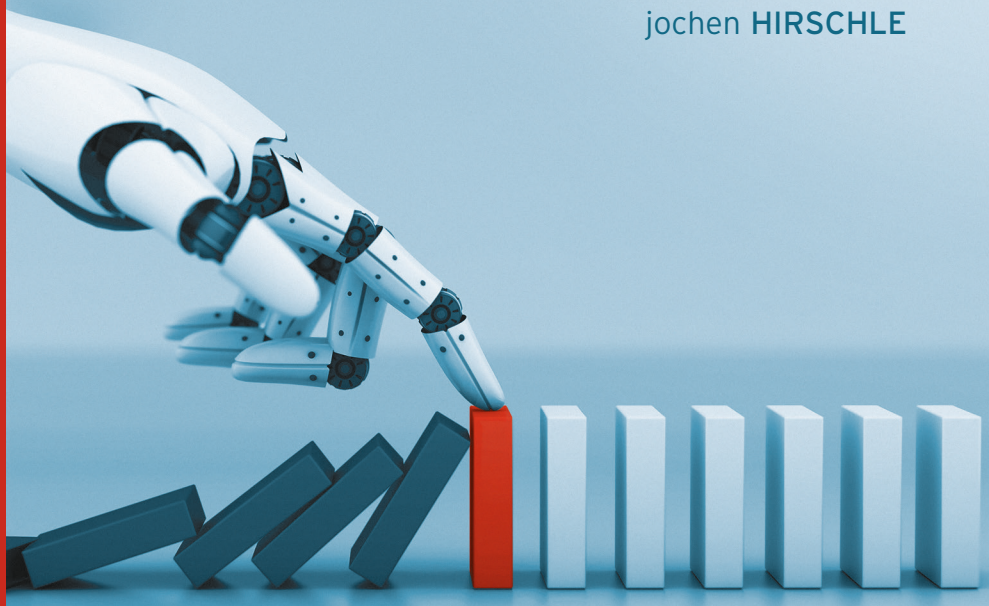
jochen HIRSCHLE



MACHINE LEARNING für ZEITREIHEN

Einstieg in Regressions-, ARIMA- und Deep-Learning-Verfahren mit Python

Im Internet:
GitHub-Repository zum Buch

HANSER

Hirschle

Machine Learning für Zeitreihen

Jochen Hirschle

Machine Learning für Zeitreihen

Einstieg in Regressions-, ARIMA- und Deep-Learning-Verfahren mit Python

HANSER

Der Autor:

Dr. Jochen Hirschle, Braunschweig

Bibliografische Information der Deutschen Nationalbibliothek:

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

Lektorat: Sylvia Hasselbach
Copy editing: Walter Saumweber, Ratingen
Umschlagdesign: Marc Müller-Bremer, *www.rebranding.de*, München
Umschlagrealisation: Max Kostopoulos
Titelmotiv: © *shutterstock.com/*NESPIX
Layout: Manuela Treindl, Fürth
Druck und Bindung: Kösel, Krugzell
Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702
Printed in Germany

Print-ISBN: 978-3-446-46726-2
E-Book-ISBN: 978-3-446-46814-6
E-Pub-ISBN: 978-3-446-46830-6

Inhalt

Vorwort

Zeitreihendaten entstehen heute in einer Vielzahl von Geschäftsfeldern – ob im Einzelhandel, in der Finanzbranche oder in der industriellen Produktion. Überall werden Messdaten aufgezeichnet und mit Zeitstempeln versehen abgelegt.

Solche Daten bilden nicht nur isolierte Zustände ab, sie sind wie Filme, die den Verlauf eines Vorgangs mit einer Serie von Momentaufnahmen nachzeichnen. Die zeitliche Dimension, die dadurch darstellbar wird, ist in einer datengetriebenen Ökonomie mehr als nur eine Spielart der Data Sciences. Der analytische Mehrwert von Zeitreihen wird bereits heute in einer Vielzahl von Geschäftsfeldern gewinnbringend eingesetzt und in einer noch größeren Zahl von Geschäftsfeldern wird das in der Zukunft geschehen.

Um diesen analytischen Mehrwert nutzen zu können, braucht man die richtigen Instrumente. Man braucht Konzepte und Techniken, muss verstehen, welche Muster in Zeitreihen typischerweise auftreten, welche Möglichkeiten es gibt, sie zu analysieren, und wie man sie zur Erzeugung von Prognosen einsetzen kann.

Genau das sind die Themen dieses Einführungsbuchs. Es bietet einen Einstieg in die Grundlagen und in die Praxis der Zeitreihenanalyse. Es zeigt anhand einer Vielzahl von Anwendungsbeispielen, wie sich Zeitreihen mit Python aufbereiten und analysieren lassen.

Dieses Buch ist aber kein Handbuch. Erwarten Sie also nicht, dass alle Verfahrensweisen, die es im Bereich der Zeitreihenanalyse gibt, besprochen werden. Es ist deshalb insbesondere für LeserInnen empfehlenswert, die neu in diesem Bereich sind und die sowohl eine Einführung in die wichtigsten Konzepte als auch in die zentralen Verfahren der Zeitreihenanalyse suchen. Wenn das Ihre Ziele sind und Sie darüber hinaus die nötigen Skills entwickeln möchten, um selbstständig mit Ihren eigenen Daten arbeiten zu können, bietet Ihnen dieses Buch einen soliden Einstieg.

Jetzt wünsche ich Ihnen eine anregende Lektüre. Ich hoffe, dass Sie währenddessen auch die Ehrfurcht vor der Materie verlieren, und dass das Buch Sie dazu inspiriert, eigene kreative Machine-Learning-Lösungen zu entwickeln!

Jochen Hirschle

Braunschweig im Oktober 2020

1 Einleitung

Die Fortschritte, die mit Machine-Learning-Verfahren in den letzten Jahren erzielt wurden, sind insbesondere in zwei Bereichen greifbar geworden:

- In der *Sprach- und Textverarbeitung* sind durch die Entwicklung von Technologien wie Word-Embedding, rekurrente neuronale Netze und Sequence-to-Sequence-Architekturen nachhaltige Verbesserungen bei der semantischen und syntaktischen Analyse erzielt worden. Automatische Übersetzungen werden immer präziser, Chatbots sind in der Lage, Sätze zu verstehen und angemessene Antworten auf Fragen von BenutzerInnen zu geben.
- Die Analyse von *Foto- und Videomaterial* hat sich mit konvolutionalen Netzen, die wiederkehrende Formen und Muster an verschiedenen Positionen in Bildern lokalisieren können, völlig verändert. Computer-Vision-Technologien wie Gesichtserkennung oder autonomes Fahren sind dadurch erst möglich geworden.

Auch wenn das vielleicht weniger offensichtlich ist, vollzieht sich im Bereich der Analyse von Zeitreihen gegenwärtig ein ähnlicher Umbruch. Womöglich geht dieser Umbruch etwas leiser vonstatten als in den anderen Bereichen. Das heißt aber nicht, dass dessen Auswirkungen weniger weitreichend wären.

Schließlich sind Zeitreihendaten der Schlüssel zur Prognose der Zukunft. Mit ihrer Hilfe lässt sich der Verlauf eines Prozesses nachzeichnen – der Kurs einer Aktie, die Nachfrage nach einem Produkt, der Stromkonsum oder der Verlauf einer Krankheit oder des Klimas. Wer aber die Historie eines Prozesses und dessen typische Muster versteht, kann die Fortschreibung dieses Prozesses über die Gegenwart hinaus überblicken. Zustände und Entwicklungen in der Zukunft werden prognostizierbar.

Die Zeichen für diesen stillen Umbruch stehen jedenfalls gut. Das hat natürlich mit der Tatsache zu tun, dass Zeitreihendaten heute in großen Mengen und in ganz unterschiedlichen Geschäftsfeldern entstehen – in der Produktion (der sogenannten Industrie 2.0), im Finanzsektor, im Zuge der Entwicklung des „Internets der Dinge“. Überall werden Messreihen produziert, die Vorgänge in ihrem Verlauf beschreiben.

Und es hat damit zu tun, dass immer mehr Ressourcen in die Entwicklung von Analysetechnologien und neue Module und Frameworks fließen, die den praktischen Einsatz bewährter und neuerer Technologien vereinfachen. Sie vereinfachen es, Zeitreihen in auswertbare Strukturen zu gießen, und sie vereinfachen es, Lernarchitekturen aufzubauen, die auf die Analyse von Zeitreihen zugeschnitten sind.

Genau wie in den anderen Bereichen auch, sind dabei alle Augen auf neuronale Netze gerichtet. Mit ihnen gehen die größten Erwartungen, Hoffnungen und wahrscheinlich auch Befürchtungen einher. Und wirklich gibt es einige vielversprechende Ansätze in diesem Be-

reich. Vorwiegend mit Deep-Learning-Verfahren, die auf Basis rekurrenter Schichten arbeiten, deren Lernarchitekturen aber auf die Eigenarten von Zeitreihen eingestellt werden [Lai2018].

Aber auch wenn die Aufmerksamkeit groß und berechtigt ist, darf man sich nicht täuschen. Neuronale Netze sind zwar Schlüsseltechnologie, aber sie revolutionieren das maschinelle Lernen nicht im Alleingang. Auch wenn sie ihrer Grundveranlagung nach beliebig komplexe Probleme lösen können, heißt das noch lange nicht, dass sie das in der Praxis auch wirklich tun. Neuronale Netze sind keine Allesfresser, in die man oben Daten einfüllen und unten schlaue Ergebnisse abzapfen kann.

Sie sind sensibel im Hinblick auf die Struktur der Informationen, die wir ihnen präsentieren. Und sie reagieren sensibel auf die Art und Weise, wie wir ihre Lernarchitektur einstellen. Das hat damit zu tun, dass sie keine Vorkenntnisse und keine Vorstellungen von Problemen aus ihrem Alltagswissen schöpfen können. Sie wissen nicht, welche Datenpartitionen für eine Lernaufgabe relevant sind und welche nicht. Sie können wichtige von unwichtigen Informationen nicht unterscheiden und keine systematischen, hypothesengeleiteten Versuche unternehmen, um ein Problem anzugehen. Deshalb arbeiten sie am besten, wenn sie auf eine Lernaufgabe hin getrimmt werden, und wenn die Daten, mit denen wir sie füttern, eine bestimmte Art der Interpretation nahelegt.

Sie werden im Laufe der Lektüre und an den Anwendungsbeispielen feststellen, dass weder Machine-Learning-Verfahren im Allgemeinen, noch Deep-Learning-Ansätze im Besonderen und schon gar nicht ARIMA-Modelle, eine Lernaufgabe ohne weitreichende menschliche Vorbereitung zuverlässig lösen können. Das heißt aber natürlich nicht, dass neuronale Netze deswegen in ihrer Prognosekraft eingeschränkt wären. Es heißt nur, dass wir ihre Eigenschaften kennen müssen, um sie auf eine Lernaufgabe einstellen zu können.

1.1 Eigenschaften von Zeitreihen

Bevor wir uns aber mit den Lernalgorithmen und ihren Eigenheiten beschäftigen, deren konzeptuelle Grundlagen und praktische Einsatzmöglichkeiten ansehen, loten wir zu Beginn einige Hintergründe der Materie aus. Wir beschäftigen uns mit den Eigenschaften und Problemen des Materials, mit dem wir arbeiten.

Zeitreihen sind – etwa im Unterschied zu Text- oder Bilddaten – kein spezifischer Datentyp. Es handelt sich um ein Strukturmerkmal von Daten, um eine Ordnung, in der Informationen vorliegen. In diesem Buch interessieren wir uns in erster Linie für diskrete Zeitreihen. Diskrete Zeitreihen sind Zeitreihen, in denen die Messwerte, die über einen Untersuchungsgegenstand gewonnen wurden, in regelmäßigen Intervallen organisiert sind – zum Beispiel in Sekunden-, Stunden oder Tagesintervallen.

Diese Ordnung der Daten ermöglicht es, die Veränderung des Untersuchungsgegenstandes über die Zeitachse zu analysieren. Das ist einer der Hauptvorteile von Zeitreihen. Im Wesentlichen geht es darum, herauszufinden, inwiefern ein zeitlich späterer Messwert durch zeitlich frühere Messwerte beeinflusst wird. Treten wiederkehrende Muster in Erscheinung, spricht das dafür, dass die Reihe einem bestimmten Regelsystem unterliegt. Dessen Kenntnis

kann dann dazu verwendet werden, die Historie einer spezifischen Reihe mit einer gewissen Fehlertoleranz in die Zukunft hinein fortzuschreiben.

Leider beginnen an dieser Stelle auch schon die Unwägbarkeiten. Zwar sind die meisten davon nicht auf Zeitreihen beschränkt, treten aber in Zeitreihen im besonderen Maß in Erscheinung.

Kausale Beziehungen und Korrelationen

Eine dieser Fragen, mit denen wir praktisch immer konfrontiert werden, wenn wir mit Machine-Learning-Algorithmen arbeiten, ist die Frage der Unterscheidung zwischen *Korrelation* und *Ursache-Wirkungsbeziehung*. Auch wenn die Analyse von Zeitreihen immerhin den Vorteil hat, dass wir kontrollieren können, ob ein Ereignis vor oder nach einem anderen Ereignis auftritt, sind wir dadurch noch lange nicht sorgenfrei.

Unproblematisch ist das nur, solange wir uns an die idealtypischen Beispiele aus dem Lehrbuch halten, zum Beispiel der Aufzug dunkler Wolken, die nicht nur Vorbote oder Symptom des Regens sind, sondern den Regen tatsächlich auslösen. Was ist aber die Ursache für einen Unfall, den ein betrunkener Autofahrer ausgelöst hat? Zu spätes Bremsen, Tunnelblick, verlangsamtes Reaktionsvermögen, die Tatsache, dass am Straßenrand ein Reh stand, die Trinkkultur der Gesellschaft oder vielleicht doch der Bossa Nova?

Auch gehen die allermeisten Ereignisse nicht nur auf eine einzige Ursache zurück. Sie gehen auf ein ganzes Bündel von Ursachen zurück, das erst in Kombination zu einem auslösenden Faktor wird – das Entladen von Elektrizität zwischen Wolken, das sich als Blitz manifestiert, die Ausbreitung einer Protestbewegung, die auf vielfältige, mehr oder weniger koordinierte Handlungen von Individuen zurückgeht.

Davon abgesehen haben wir oft überhaupt nicht die Möglichkeit, das verursachende Phänomen direkt zu messen. Entweder weil wir es nicht kennen oder weil wir keine Messinstrumente besitzen, mit denen das möglich wäre. Also benutzen wir Indikatoren und Proxys. Wo wir auch hinsehen, wird nicht das eigentliche Phänomen gemessen, für das wir uns interessieren, sondern auf Stellvertreter gesetzt.

Dazu muss man keine abstrakten Beispiele erfinden. Schon der Abfall des Benzindrucks, den wir aus den Sensordaten auslesen und aus dem wir schließen, dass die Turbine bald ausfällt, ist nicht die Ursache für deren Ausfall, sondern nur Epiphänomen, das wir als Indikator benutzen. Ursache ist etwas anderes. Vielleicht ein Steinschlag, der die Benzinleitung beschädigt hat.

In manchen Fällen spielt es womöglich keine Rolle, ob wir Ursache oder Begleiterscheinungen messen. Vollkommen irrelevant ist es aber so gut wie nie. Allein schon, weil ein Epiphänomen von sehr unterschiedlichen Faktoren ausgelöst werden kann und nicht nur von dem einen Faktor, der das Ereignis zur Folge hat, das wir prognostizieren wollen. Wir können uns also nie sicher sein, ob am Ende wirklich das erwartete Ereignis eintritt, nur weil wir das Epiphänomen beobachtet haben.

Messungen

Damit sind wir auch schon bei einem zweiten Aspekt angelangt: bei der Messung bzw. der Gewinnung von Informationen. Darüber machen wir uns als Data Scientists nur selten Gedanken. Jedenfalls nicht, wenn es um die Frage der Erhebung geht. Ein Vorgang, bei dem wir auf sehr konkrete Weise mit der Realität und ihren Eigenarten konfrontiert werden.

In der empirischen Praxis delegieren wir die Erhebung von Informationen an Messinstrumente: An Thermometer, Spektrometer, Cookies, Tracker, Formulare oder Fragebögen. Dabei nehmen wir stillschweigend an, dass solche Informationen zuverlässig oder sogar objektiv sind – was immer das heißt. Zum Beispiel, dass sie nicht den typischen Verzerrungen unterliegen, die unsere Sinnesorgane und unser Erinnerungsvermögen produzieren.

Das ist ein Fehler. Denn Messungen gehen in den meisten Fällen mit einer Vielzahl von Problemen einher. Auch wenn wir das den polierten Datenmatrizen, mit denen wir später arbeiten, nicht mehr ansehen.

Da wäre zum Beispiel, wie wir oben gesehen haben, die Tatsache, dass wir die Dinge, für die wir uns eigentlich interessieren, häufig nicht direkt erheben können. Wir weichen auf Proxys aus, mit all den Konsequenzen, die daraus im Hinblick auf die Frage der Analyse von Ursache-Wirkungsbeziehungen resultieren.

Da wären aber auch die Ungenauigkeiten des Messinstruments oder der Einfluss sich verändernder Rahmenbedingungen bei mehrmaligen Messungen. Wenn zum Beispiel die Vibrationen, die beim Ausfahren der Landeklappen eines Flugzeugs entstehen, die Aufzeichnungen der Schwingungen einer Turbine beeinflussen oder wenn das Messinstrument über die Zeit hinweg verkalkt, undicht wird oder oxidiert, sind unsere Messungen verzerrt. Auch ein Tracker misst nur so lange die Aktivitäten einer Person im Internet, bis ihn jemand löscht oder bis sich die Person entscheidet, mit einem anderen Browser die interessanten Seiten zu öffnen. Hinzu kommt, dass solche Ungenauigkeiten und Ausfälle, wie wir später sehen werden, bei der Analyse von Zeitreihen besonders schwer ins Gewicht fallen, weil Algorithmen häufig schon kleinste Schwankungen im Verlauf als Anzeichen für ein Ereignis deuten.

Veränderung

Darüber hinaus haben wir es in der Zeitreihenanalyse mit einem ganz allgemeinen und wissenschaftstheoretisch weitgehend ungelösten Kernproblem zu tun, nämlich mit der Frage, was Veränderung überhaupt bedeutet. Das klingt vielleicht erst einmal seltsam. Aber was ist Veränderung? Wenn sich etwas verändert, wird es dann durch die Veränderung nicht zu etwas Anderem? Ist Veränderung nicht ein Paradoxon? Wenn es Veränderung gibt, dann muss das Objekt, das sich verändert, doch gleichzeitig unverändert bleiben, jedenfalls soweit unverändert, dass wir es nach der Änderung immer noch als das gleiche Objekt erkennen können.

Ist ein Mensch, der altert, noch derselbe Mensch wie früher, der Erwachsene der gleiche Mensch wie das Kind? Die Frage können wir vielleicht aus unserer Erfahrung heraus noch eindeutig mit *Ja* beantworten – obwohl dem einen oder anderen schon gewisse Zweifel kommen mögen. Und die Grenzen sind ja fließend. Wie ist es mit einer Raupe, die sich zu einem Schmetterling verpuppt, ein Mensch, der stirbt, oder eine Maschine, die sich in ihre Bestandteile auflöst? Jetzt wird es schon schwieriger. In den ersten beiden Fällen würde man vermutlich eher von Verwandlung oder von Übergang und im zweiten von Desintegration sprechen [Popper2016, S. 45].

1.2 Daten, Korrelationen und das „Ende der Theorie“

Das Schöne oder auch das Problem der Data Sciences – je nachdem aus welcher Perspektive man das betrachten möchte – ist, dass sie sich um all diese Dinge in der Regel wenig Gedanken machen muss.

Zum einen überlassen wir die Erhebung der Informationen anderen. Wir arbeiten mit fertigen Datenmatrizen, die wir höchstens noch für die Analyse aufbereiten und in Form bringen müssen. Wenn Daten fehlen oder ungenau sind, dann sind das für uns pragmatische Fragen, dir wir mit technischen Mitteln, mit Imputationsverfahren oder Kontrollvariablen, die wir in Lernalgorithmen einsetzen, – soweit wie möglich – in den Griff bekommen müssen.

Zum anderen haben wir uns eine gewisse Ignoranz gegenüber der Frage von Ursache-Wirkungsbeziehungen zu eigen gemacht. Manche haben daraus sogar eine Grundhaltung entwickelt, die sich in etwa so zusammenfassen lässt: Warum sollten wir uns überhaupt dafür interessieren, ob die Modelle auf Basis von Epiphänomenen angelernt werden oder nicht, solange wir damit nur zuverlässige Prognosen erstellen können?

> *Google's founding philosophy is that we don't know why this page is better than that one: If the statistics of incoming links say it is, that's good enough. No semantic or causal analysis is required. That's why Google can translate languages without actually „knowing“ them. [Anderson2008]*

Das ist natürlich nur eine Art der Grundhaltung. Andere sehen das anders. Andere denken, dass wir uns, indem wir die methodische Not zur beruflichen Tugend machen, mittelfristig die Zukunftschancen verbauen.

Zum Beispiel Judea Pearl, einer der Pioniere in der Entwicklung intelligenter Systeme: *„All the impressive achievements of deep learning amount to just curve fitting“*, sagte er 2018 in einem Interview [TheAtlantic2018] und wollte damit zum Ausdruck bringen, dass wir uns seit geraumer Zeit im Kreis drehen.

Solange künstliche Intelligenz nicht mehr leistet als Regelmäßigkeiten in großen Datensätzen aufzuspüren und wir den Algorithmen nicht den Unterschied zwischen Epiphänomenen und kausalen Ursache-Wirkungsbeziehungen beibringen können, werden wir schon bald keine weiteren Fortschritte mehr erzielen [Pearl2018]:

> *The key, he [Judea Pearl] argues, is to replace reasoning by association with causal reasoning. Instead of the mere ability to correlate fever and malaria, machines need the capacity to reason that malaria causes fever. [TheAtlantic2018]*

Zuletzt, und das ist vielleicht die wichtigste Erkenntnis, können wir das Paradoxon von Veränderung und Identität in gewisser Weise für unsere Zwecke lösen, nämlich indem wir daraus eine statistische Eigenschaft machen. Vielleicht ist das nicht die Antwort auf die philosophische Frage, die dahintersteht, aber immerhin eine Antwort auf die Frage, wie sich Veränderung bei gleichzeitiger Beständigkeit überhaupt konzeptualisieren lässt.

Nehmen wir an, ein Objekt verfügt über gewisse Eigenschaften, die es zu dem machen, was es ist. Und nehmen wir weiter an, dass wir diese Eigenschaften mithilfe eines oder mehrerer Indikatoren messen können. Wenn wir die Identität eines Objekts nicht als etwas außerhalb

dieser Eigenschaften stehendes betrachten, dann drückt sich Identität offensichtlich in der Korrelation dieser Eigenschaften über die Zeit aus.

Das Phänomen nennt sich Autokorrelation. Es bedeutet zum Beispiel, dass der Messwert einer Eigenschaft zum Zeitpunkt t vermutlich sehr ähnlich dem Messwert zum Zeitpunkt $t+1$ ist. Zumindest dann, wenn man davon ausgeht, dass der zeitliche Abstand zwischen t und $t+1$ sehr klein Ist. Veränderung drückt sich dann in einer abnehmenden Korrelation über die Zeit aus. Die Korrelation selbst zeigt aber immer noch an, dass zwischen dem Objekt zum Zeitpunkt t und dem Objekt zum Zeitpunkt $t+1$, $t+10$ oder $t+1000$ eine Verbindung besteht.

Nehmen wir zum Beispiel die Drehgeschwindigkeit eines Triebwerkteils, die wir zu zwei Zeitpunkten messen. Wenn der Abstand zwischen den beiden Messzeitpunkten hinreichend gering ist, ist die Divergenz zwischen der Drehzahl sehr klein. Die Geschwindigkeit zum Zeitpunkt t_0 beinhaltet also bereits einen hohen Anteil der Information über die Drehgeschwindigkeit zum Zeitpunkt t_1.[1]

Die Veränderung der Messwerte zwischen zwei Zeitpunkten zeigt entsprechend an, dass sich das Objekt verändert. Darüber hinaus erhalten wir über den Vergleich der Messwerte aber auch Informationen über die Art der Veränderung. Bleiben wir bei unserem Triebwerk: Wenn der Messwert zum Zeitpunkt t_0 kleiner ist als zum Zeitpunkt t_1, deutet das darauf hin, dass die Maschine beschleunigt. Ist der Wert zum Zeitpunkt t_1 kleiner als zum Zeitpunkt t_0, verlangsamt sich die Drehgeschwindigkeit.

Auch lässt sich die Stärke der Differenz interpretieren, zum Beispiel weil ein Schwungrad eine gewisse Trägheit an den Tag legt und eine abrupte Verlangsamung oder Beschleunigung nur unter hoher Krafteinwirkung zustande kommt. Ein vollkommenes Degradieren der Maschine würde sich gegebenenfalls in einer Nullkorrelation relevanter Messwerte über zwei kurze Zeitpunkte hinweg widerspiegeln. Wenn man das wollte, könnte man dann von der Auflösung der Identität des Objekts sprechen.

Ob es über diese Art der Analyse hinaus sinnvoll ist, Identität mithilfe von Korrelationen auszudrücken, ist eine andere Frage. Tatsächlich ist die Trägheit von Objekten beziehungsweise die Abhängigkeit von Zuständen eines Objekts in der Zukunft von Zuständen in der Vergangenheit ja die einzige Möglichkeit überhaupt, einen Blick in die Zukunft zu werfen. Wenn wir also für den Kurs einer Aktie keine Korrelation zwischen den Kursen am Vortag und Folgetag feststellen, können wir auch keine Prognosen erstellen. Für den Börsenmakler oder den Anleger ist das allerdings kein Grund, die Identität der Aktie in Zweifel zu ziehen – zumal die daran gekoppelten Gewinne und Verluste sehr reale Konsequenzen dieser Identität sind. Das liegt aber streng genommen nicht daran, dass wir Identität falsch operationalisieren, sondern dass die Identität eines Objekts im Zweifelsfall über ein Außenkriterium hergestellt wird, dass sich per Definition nicht verändern kann.

In der Zeitreihenanalyse sind Autokorrelationen jedenfalls ein wichtiges Konzept, das für verschiedenste Zwecke herangezogen wird. Auch saisonale Schwankungen sind aus statistischer Sicht nicht viel mehr als Korrelationen zweier Messwerte über die Zeit hinweg. Allerdings kommen jetzt andere zeitliche Bezüge ins Spiel. Wir müssen uns daher nicht nur um zeitversetzte Messungen kümmern, sondern auch um die Abstände zwischen den Messzeitpunkten.

[1] Ganz anders ist das, wenn wir diese Messwerte mit den Messwerten eines anderen Triebwerks vergleichen, die so gut wie unkorreliert sind.

Die Temperatur im März hängt vielleicht weniger von der Temperatur im Februar als von der Temperatur im März des Vorjahres ab. Die Verkaufszahlen von Weihnachtsmännern im Dezember 2021 lassen sich präziser mithilfe der Zahlen der Verkäufe von Weihnachtsmännern im Dezember 2020 vorhersagen als mit den verkauften Weihnachtsmännern im November des gleichen Jahres.

Prognosen erzeugen heißt also im Wesentlichen, die wiederkehrenden Muster entdecken und sie in die Zukunft fortzuschreiben. Solche Muster können natürlich mehr oder weniger komplex sein: Trägheit, saisonale Schwankungen, einfache Trends, exponentielles Wachstum. Und sie können in unterschiedlichen Messdaten zum Ausdruck kommen.

Als Menschen vor unserer Zeit versuchten, die Temperatur von morgen vorherzusagen, verwendeten sie intuitiv alle verfügbaren Indikatoren und nicht nur die Temperatur am heutigen Tag. Sie warfen einen Blick in den Himmel und deuteten aufziehende Wolken als Vorboten für Regen, der Kälte bringt. Sie leiteten aus der Windrichtung ab, ob Luftmassen aus dem Süden oder Norden zu ihnen strömten. Sie orientierten sich womöglich am Verhalten der Tiere, an den Schwalben, die bei aufziehendem Regen, so sagt man, tiefer fliegen.

Komplexe statistische Modelle gehen ganz ähnlich vor. Wir können sie unter Verwendung aller verfügbaren Messdaten trainieren, und sie versuchen aus all diesen Daten – aus Verläufen, Trends, Interaktionen einzelner Messindikatoren untereinander – Muster zu erlernen, die Hinweise auf Zielwerte in der Zukunft geben.

1.3 Inhalt

Damit haben wir uns einen ersten Eindruck über die Hintergründe und Probleme verschafft. Wir werden sehen, dass wir einige dieser Probleme nicht loswerden. Trotzdem sollten wir sie nicht ignorieren, da die Modelle ab einem bestimmten Punkt nicht mehr besser werden, wenn wir nicht in der Lage sind, die tieferliegenden Schwachstellen unserer Daten oder Ansätze zu beheben.

Kommen wir jetzt aber zu den Inhalten dieses Buches. Es geht schließlich nicht darum, was wir nicht können, sondern um das, was möglich ist. Was wir also aus unseren Daten machen, welche Ansätze für welche Arten von Daten geeignet sind, wie wir Daten vorbereiten, Modelle aufsetzen, einstellen und anlernen können. Das ist Thema dieses für die Praxis geschriebenen Einführungsbuches.

Wir beginnen im *zweiten Kapitel* mit einer Auseinandersetzung mit dem Datenmaterial. Die Aufbereitung von Zeitreihendaten birgt in der Praxis einige Fallstricke. Wir müssen mit Datumsangaben arbeiten, unsere Daten in gleichmäßige Intervalle unterteilen, in Sekunden-, Minuten-, Stunden-, Tages- oder Monatsrhythmen. Wir müssen unser Datenmaterial, bevor wir Modelle anlernen können, verstehen, Dynamiken und Bewegungen visuell erfassen, um das richtige Modell für eine Aufgabe auswählen zu können und auch, um unsere Modelle beim Anlernen auf die richtige Spur zu bringen. Denn der Erfolg von Lernalgorithmen hängt neben der Frage, ob die Daten überhaupt relevante Informationen beinhalten, auch von der Wahl der richtigen Lernarchitektur ab.

Nachdem wir also im ersten Kapitel die Daten organisiert und visualisiert haben, beginnen wir im *dritten Kapitel* mit der Besprechung möglicher Prognoseverfahren. Wir starten mit den einfacheren Verfahren, die zum Grundstock des Machine Learning gehören. Es geht um lineare, logistische und Softmax-Regressionen. Verfahren, die zwar nicht speziell zur Analyse von Zeitreihendaten konzipiert sind, mit denen sich aber bestimmte Problemstellungen ohne Weiteres modellieren lassen. Davon abgesehen bilden sie die Grundlage aller anderen Verfahren, sowohl der ARIMA- als auch der Deep-Learning-Modelle, denen wir uns in den folgenden Kapiteln widmen. Daher nutzen wir die Gelegenheit und sehen uns an, wie sie arbeiten, wie sich lineare und nicht-lineare Zusammenhänge modellieren lassen und woran sie scheitern.

Von dort aus geht es ohne weitere Umwege zum *vierten Kapitel* und damit zu den ARIMA-Verfahren. Mit ihnen lassen sich latente Beziehungen, die erst auf den zweiten Blick in den Daten sichtbar werden, für Prognosen fruchtbar machen. Außerdem bieten sie Möglichkeiten, mit saisonalen und Trendkomponenten umzugehen. Auf der anderen Seite sind ARIMA-Modelle etwas widerspenstig in der Handhabung. Sie verlangen, dass wir die Daten vorbereiten und gut genug kennen. Wir müssen den Modellen mitteilen, in welchen Regionen der Reihe sie Muster anlernen sollen. Daher kümmern wir uns um Begriffe wie *Stationarität* und versuchen mithilfe von *Autokorrelationsdiagrammen* ein Auge für die verborgenen Muster zu entwickeln, die in unseren Daten enthalten sind.

Die Beschäftigung mit ARIMA-Verfahren ist die beste Schule, um das Handwerk der Zeitreihenanalyse zu lernen. Das brauchen wir auch, wenn es im *fünften Kapitel* um Deep-Learning-Verfahren geht. Von allen Verfahren sind neuronale Netze am leistungsstärksten. Sie können auf Grundlage eines Messwertes oder einer Vielzahl von Messwerten einer Zeitreihe eine Zielvariable prognostizieren. Sie können komplexe Muster aus Daten lernen. Sie haben das größte Potenzial und ihnen gehört die Zukunft. Aber sie verzeihen Fehler in der Handhabung noch weniger als ARIMA-Modelle. Sie sind anfällig für schlecht vorbereitete Daten, die sie nicht anlernen. Sie neigen zur Überanpassung an die Trainingsdaten und es gibt nur sehr wenige Anhaltspunkte dafür, wie wir ein neuronales Netz aufbauen sollten, über wie viele Schichten und Neuronen es verfügen sollte, um optimale Schätzergebnisse zu erzielen.

Wir müssen also erst verstehen, wie neuronale Netze arbeiten, um sie für unsere Problemstellungen einsetzen und zuschneiden zu können und um unsere Daten für den Anlernprozess vorbereiten zu können. Und wir brauchen vor allem Vergleichsmöglichkeiten. Wir müssen in der Lage sein, herauszufinden, ob ein angelerntes neuronales Netz wirklich besser funktioniert als eine einfache Regression, ein ARIMA-Modell oder ein Baseline-Schätzer, den wir auf Grundlage einer Faustformel erzeugt haben.

1.4 Voraussetzungen, Ressourcen und praktische Hinweise

Im praktischen Teil arbeiten wir dabei mit der Programmiersprache *Python*. Sie hat sich in den letzten Jahren zu einer der beliebtesten Programmiersprachen unter Data Scientists und Entwicklern gleichermaßen gemausert [Stackoverflow2019].

Python ist leicht zu lernen und bietet eine Reihe von Vorteilen im Umgang mit der Verarbeitung strukturierter und unstrukturierter Daten, im Umgang mit mehrdimensionalen Arrays und mit Datentypen aller Art. Außerdem bietet Python mit *Scikit-learn*, *statsmodels*, *TensorFlow* und *Keras* State-of-the-Art Machine-Learning-Bibliotheken, die ständig verbessert und weiterentwickelt werden.

Wenn Sie mit diesem Buch arbeiten, sollten Sie über Python-Grundlagenwissen verfügen. Und sie sollten wissen, wie sich Daten mit der Bibliothek *pandas* einlesen und verarbeiten lassen. Außerdem sind Kenntnisse statistischer Basiskonzepte wie Mittelwert, Standardabweichung, Korrelation oder lineare Regression hilfreich.

Die Inhalte der Kapitel des Buches bauen aufeinander auf. Sie sollten also die Inhalte des Kapitels zur Verarbeitung und Vorverarbeitung von Zeitreihen kennen, wenn Sie das Kapitel *Grundlagen maschinellen Lernens* lesen. Und Sie sollten die Grundlagen maschinellen Lernens beherrschen, wenn sie das Kapitel zum Thema neuronale Netze lesen. Eine Ausnahme bildet das ARIMA-Kapitel. Es ist nicht zwingend Voraussetzung, um das nachfolgende Kapitel zum Thema Deep Learning zu verstehen. Trotzdem finden Sie dort eine Reihe von Informationen über die Anatomie von Zeitreihen, die ihnen das Leben bei der Anwendung von Deep-Learning-Verfahren leichter macht.

Wenn Sie bereits Vorerfahrung in Machine Learning haben, können Sie gegebenenfalls das dritte Kapitel überspringen. Die Verfahren, die im Speziellen für die Analyse von Zeitreihendaten interessant sind, werden in den Kapiteln 4 und 5 vorgestellt.

Die Anwendungsbeispiele sind durchgehend mit Python 3.7.7 und unter Verwendung der folgenden Pakete/Versionen umgesetzt worden:

```
tensorflow=2.1.0
statsmodels=0.11.1
scikit-learn=0.22.1
pandas=1.1.1
seaborn=0.10.1
```

Wenn Sie die Beispiele selbst rechnen und dazu eine neue virtuelle Umgebung mit den relevanten Bibliotheken aufsetzen möchten, ist es ratsam, das mit *Anaconda* zu erledigen.[2] Wenn Sie stattdessen den Package-Installer *pip* zur Installation verwenden, laufen Sie Gefahr, auf eine Reihe von Versionskonflikten zu stoßen.

Begleitend zum Buch finden Sie den kompletten Code und die Daten zu den Anwendungsbeispielen im Internet auf einem GitHub-Repository unter: *https://github.com/tplusone/*

[2] Anaconda (*https://www.anaconda.com*) ist eine Data-Science-Plattform, die verschiedene nützliche Werkzeuge, unter anderem zum Anlegen virtueller Umgebungen und zur Installation von Bibliotheken, zur Verfügung stellt.

hanser_ml_zeitreihen. Sie können sich die Jupyter Notebooks mit dem Code zu den Kapiteln entweder im Browser ansehen oder das Repository klonen. Dafür benötigen Sie eine lauffähige git-Version, die sie kostenlos im Internet herunterladen und installieren können (*https://git-scm.com/downloads*). Mit dem folgenden Befehl werden alle Daten und Code-Beispiele auf ihren lokalen Rechner gezogen:

```
$ git clone https://github.com/tplusone/hanser_ml_zeitreihen.git
```

Hinweis zur Dezimalschreibweise:

Da in der Ausgabe der Programmiersprache Python durchgehend das amerikanische Zahlenformat zur Darstellung verwendet wird, haben wir uns – um Verwechslungen zu vermeiden – entschieden, auch im Text, in den Formeln und Tabellen dieses Zahlenformat zu verwenden. Mit einem Punkt werden daher immer Dezimalstellen, mit einem Komma Tausenderstellen abgetrennt. Ausgenommen davon sind Prozentangaben.

2 Verarbeitung und Vorverarbeitung von Zeitreihen

Zeitreihen sind Sammlungen von Beobachtungen, die sich über einen bestimmten Zeitraum erstrecken. Die häufigste Form sind diskrete Zeitreihen, bei denen die Beobachtungen – meist Messungen – jeweils zu bestimmten Zeitpunkten erfolgen. Die Intervalle zwischen diesen Zeitpunkten können dabei regelmäßig oder unregelmäßig sein.

Wir interessieren uns für Zeitreihen mit regelmäßigen Intervallen, weil sich damit spezielle Analysen durchführen lassen. Allerdings können die meisten Zeitreihen mit unregelmäßigen Intervallen in Zeitreihen mit regelmäßigen Intervallen überführt werden – vorausgesetzt wir verfügen über genügend Datenmaterial und einen Zeitstempel, der die Zeit der Messung hinreichend genau spezifiziert.

In den folgenden Abschnitten sehen wir uns an, wie sich Zeitreihen in Form bringen lassen. Dabei arbeiten wir mit der Bibliothek *Pandas* und *NumPy*.

- Pandas ist ein Framework, das im Speziellen zur Arbeit mit strukturierten Daten entwickelt wurde. Hauptbestandteil ist die Klasse *DataFrame*, mit der sich zweidimensionale Tabellen, die aus Zeilen und Spalten bestehen, aufbereiten und umarbeiten lassen.
- Bei Pandas handelt es sich um eine High-Level-API, die die Daten unter der Haube mit *NumPy* organisiert. NumPy steht für *Numerical Python*. Zentrale Einheit des Frameworks ist die Klasse *ndarray*. Dabei handelt es sich um Behälter für datentypsichere Arrays, mit denen sich mehrdimensionale Datenstrukturen darstellen und mit einer Vielzahl von Funktionen bearbeiten lassen. NumPy ist auch deshalb so wichtig, weil faktisch alle hier verwendeten Machine-Learning-Bibliotheken Daten zum Anlernen als NumPy-Arrays erwarten.

Wir können an dieser Stelle keine umfassende Einführung in Pandas oder NumPy bieten. Das würde den Rahmen dieser Einführung sprengen. Stattdessen konzentrieren wir uns auf die Darstellung der Möglichkeiten, die Pandas und NumPy für die Aufarbeitung von Zeitreihen bieten. Darunter insbesondere das Arbeiten mit Datumsfunktionen und mit Datumsindexen. Mit ihrer Hilfe lassen sich Zeitreihen in Strukturen bringen, die für das Anlernen von Lernalgorithmen vorteilhaft sind.

Darüber hinaus kümmern wir uns um den Umgang mit fehlenden Werten und darum, wie wir Zeitreihendaten mit NumPy in dreidimensionale und andere mehrdimensionale Formate bringen können, die wir später, wenn es um die Fütterung von Deep-Learning-Modellen geht, benötigen.

2.1 Mit Datumsangaben arbeiten

Die Frage, ob man Beobachtungsdaten als Zeitreihen interpretieren kann, hängt zunächst und zuallererst von der technischen Frage ab, ob die Daten bei der Aufzeichnung mit einem Zeitstempel versehen werden, der Auskunft über den exakten Zeitpunkt der Beobachtung gibt.

Fortlaufende, womöglich theoretisch unendliche Zeitreihen werden zur Verarbeitung meist in Tabellen gegossen. Dabei stellen ID (Objektidentität, falls mehr als ein Objekt fortlaufend beobachtet wird) und die Bezeichnungen der Messwerte die Spaltenüberschriften, und die Messwerte pro Zeiteinheit die Inhalte jeder Zeile dar. Jede neue Messung wird in eine neue Zeile einer strukturell unveränderten Tabelle geschrieben, in der das Datum der Messung als Index festgelegt ist (Tabelle 2.1). In seltenen Fällen, zum Beispiel bei kurzen Zeitreihen, in der fixe Zeitpunkte erhoben werden, können die verschiedenen Messwerte auch in einer Zeile über verschiedene Spalten aufgezeichnet werden.

Tabelle 2.1 Beispiel einer Zeitreihe mit fixen Intervallen (Stunden), in der Messwerte (Luftdruck, Grad Celsius) für verschiedene Zeitpunkte aufgezeichnet wurden. Jede Zeile beinhaltet Messwerte, die sich auf die gleiche Einheit (Ort) für verschiedene Zeitpunkte beziehen.

Date	p (mbar)	T (degC)
2009-01-01 00:00:00	996.528	−8.304
2009-01-01 01:00:00	996.525	−8.065
2009-01-01 02:00:00	996.745	−8.763
2009-01-01 03:00:00	996.987	−8.897
2009-01-01 04:00:00	997.158	−9.348

Machine-Learning-Verfahren, die auf die Analyse von Zeitreihen spezialisiert sind, setzen in der Regel eine Datenstruktur voraus, die folgende Kriterien erfüllt:

- Jede Zeile bildet die Messwerte für eine Zeiteinheit ab, die sich auf ein einzelnes Objekt bezieht.
- Die Zeilen sind in festen, regelmäßigen Intervallen organisiert (z. B. monatliche, stündliche, sekündliche Messdaten).
- Es handelt sich um lückenlose Daten. Es liegen keine fehlenden Zeilen vor, die die Intervallstruktur unterbrechen würden, und es liegen keine Zeilen vor, in denen fehlende Werte auftreten.

Da Daten in den seltensten Fällen in genau dieser Form aufgezeichnet werden, bietet Pandas eine Reihe von Möglichkeiten, Daten nachträglich umzustrukturieren: Wir können Zeitreihen unter Verwendung von Zeitstempeln in regelmäßige Intervalle überführen, und wir können unvollständige oder lückenhafte Datenreihen schließen.

Wir werden im Folgenden die Möglichkeiten ausloten, mit Zeitstempeln zu arbeiten und Daten in unterschiedliche Strukturen zu bringen. Zur Veranschaulichung verwenden wir dabei vorzugsweise eine Zeitreihe mit Klimadaten, die in der Stadt Jena zwischen den Jahren 2009 und 2015 erhoben wurden.

2.1.1 Daten laden

Beginnen wir mit dem Laden der Daten als Data-Frame und der Ansicht der ersten Zeilen, um uns einen Überblick zu verschaffen:

```
import pandas as pd
df = pd.read_csv(file)
df.head()
```

Ausgabe:

```
   Date Time   p (mbar)  T (degC)
0  01.01.2009  00:10:00  996.52    -8.02
1  01.01.2009  00:20:00  996.57    -8.41
2  01.01.2009  00:30:00  996.53    -8.51
3  01.01.2009  00:40:00  996.51    -8.31
4  01.01.2009  00:50:00  996.51    -8.27
```

Mit der head-Methode, die wir auf den als *DataFrame* instanziierten Datensatz anwenden, erhalten wir eine Übersicht über die ersten fünf Zeilen. Die Daten enthalten insgesamt drei Variablen: einen Zeitstempel (*Date Time*) und die Messwerte für Luftdruck und Temperatur (*p (mbar), T (degC)*). Außerdem erzeugt Pandas automatisch einen Index, der – falls nicht anders spezifiziert – einer Durchnummerierung der einzelnen Zeilen beginnend mit dem Indexwert 0 entspricht.

2.1.2 Datumsangaben und Zeitstempel parsen

Pandas versucht beim Einlesen die Daten automatisch in Zahlenformate (Integer oder Floats) zu konvertieren, sofern dies möglich ist. Für die beiden Messparameter (Luftdruck und Temperatur) ist das gelungen. Wenn wir die Spalten anpacken und das Attribut *dtype* abfragen, erhalten wir den entsprechenden Datentyp zurück:

```
df['p (mbar)'].dtype
```

Ausgabe:

```
dtype('float64')
```

Bei der Variable *Date Time* hat Pandas das Datenformat nicht automatisch verstanden und deshalb einen gemischten Datentypus (O) produziert, der sich wie ein String verhält. Für die Verarbeitungsschritte, die uns noch bevorstehen, ist dieses Format ungeeignet. Wir wollen später mit Datumsangaben arbeiten, z. B. Mittelwerte über Stunden berechnen oder einzelne Datumsbereiche (Monate) extrahieren.

Deshalb wandeln wir das Datum zunächst in eine Basisinstanz der Klasse *datetime* um, mit der wir datumsspezifische Operationen durchführen können. Dafür gibt es in Pandas mehrere Möglichkeiten. Wir beschränken uns hier auf die einfachste, die nachträgliche Umwandlung, indem wir die Pandas-Funktion *to_datetime* einsetzen:

```
df['date'] = pd.to_datetime(df['Date Time'])
df['date'].dtype
```

Ausgabe:

```
dtype('<M8[ns]')
```

Nach dieser Aktion erhalten wir eine neue Spalte *date*, die vom Datentyp *M8* ist. Dabei handelt es sich um einen Pandas-spezifischen Datumstypus, der alle wesentlichen Funktionalitäten der Python-*datetime*-Klasse beinhaltet, darüber hinaus jedoch einige weitere Funktionalitäten bietet, die wir später benötigen werden.

Die Funktion *to_datetime* hat beim Aufruf den Wert jeder Zeile, der sich in der Variablen *Date Time* befindet, gelesen, interpretiert und automatisch die richtigen Direktiven auf die jeweiligen Positionen des Strings angewendet: Jahr, Monat, Tag, Stunde, Minute usw.

Auf diese Weise entsteht am Ende eine neue Spalte, in der jeder einzelne Wert in einen Datumstyp umgewandelt wurde. Dieser Automatismus funktioniert in vielen Fällen – aber nicht immer. Liegt ein Datumsstring in einem Format vor, das nicht automatisch gelesen und interpretiert werden kann, bekommen wir eine Fehlermeldung und wir müssen den Parse-Vorgang manuell anleiten. Dazu bietet die *to_datetime*-Funktion den optionalen Parameter *format* an.

Der Parameter *format* nimmt einen String, der das Parsing durch Angabe von Datumsdirektiven informiert. Bei den Direktiven handelt es sich um vorgegebene Abkürzungen für datumsrelevante Angaben. In unserem Fall müssten wir folgenden String zur Formatierung anwenden:

```
df['date'] = pd.to_datetime(
             df['Date Time'], format='%d.%m.%Y %H:%M:%S')
```

Pandas wird damit informiert, dass die zu lesenden Werte nacheinander Angaben zum Tag, Monat, Jahr, zur Stunde, Minute und Sekunde beinhalten. Außerdem spezifizieren wir damit das Format dieser Angaben (z. B. vierstellig vs. zweistellig). Die dafür vorgesehenen Direktiven sind vordefiniert. Eine Liste finden Sie in der Python-API der Klasse *datetime*, *https://docs.python.org/3/library/datetime.html#strftime-strptime-behavior*. Beachten Sie auch, dass der String, der das Parsing anleitet, auch alle Zeichen beinhalten muss, die beim Parsen übergangen werden sollen (z. B. Punkte, Doppelpunkte, Leerzeichen).

Nach der Umwandlung stehen uns einige einfache und weitreichendere Funktionen zur Verfügung, mit denen wir an unseren Daten arbeiten können.

Sehen wir uns zunächst die einfachen Funktionen an. Wir können jetzt zum Beispiel den Tag, Monat und das Jahr oder die Stunde aus dem geparsten Datum extrahieren, indem wir zuerst das Postfix *dt* angeben und dann das gewünschte Datums-Attribut abrufen: *day*, *month*, *year*, *hour* etc. Eine neue Variable, die den Monat des jeweiligen Datums als Integer beinhaltet, würden wir zum Beispiel auf folgende Weise erzeugen:

```
df['month'] = df['date'].dt.month
df.head()
```

Ausgabe:

```
        Date Time            p (mbar)  T (degC)  date                 month
69499   28.04.2010 15:40:00  996.51    20.53     2010-04-28 15:40:00   4
189060  04.08.2012 22:30:00  986.66    19.54     2012-08-04 22:30:00   8
229360  11.05.2013 19:10:00  987.09    12.25     2013-05-11 19:10:00   5
308450  11.11.2014 10:30:00  985.40    4.65      2014-11-11 10:30:00  11
395306  06.07.2016 14:30:00  992.49    18.96     2016-07-06 14:30:00   7
```

2.1.3 Indexbezogene Datums-Funktionen

Um die weitreichenderen Vorzüge der Datumsklasse für Zeitreihenanalysen kennenzulernen, müssen wir zunächst das Datum als Index des Data-Frames setzen. Im Augenblick arbeiten wir noch mit dem bestehenden Standard-Index, der beim Laden der CSV-Datei automatisch erzeugt wurde und der die Zeilen im Wesentlichen durchnummeriert. Das ändern wir, indem wir die *set_index*-Methode aufrufen, und die Spalte übergeben, die wir als Index festlegen möchten:

```
df = df.set_index('date')
df.head()
```

Ausgabe:

```
date                 p (mbar)  T (degC)
2009-01-01 00:10:00  996.52    -8.02
2009-01-01 00:20:00  996.57    -8.41
2009-01-01 00:30:00  996.53    -8.51
2009-01-01 00:40:00  996.51    -8.31
2009-01-01 00:50:00  996.51    -8.27
```

Der Index ist jetzt durch die Spalte *date* (Datentyp M8) ersetzt worden. Nun können wir die Zeitreihe unter die Lupe nehmen, einzelne Fenster extrahieren und datumsspezifische Berechnungen durchführen.

Beginnen wir mit Extraktionen auf Grundlage von Datumsangaben. Nehmen wir an, wir möchten alle Zeilen, die zwischen dem 01. Februar 2011 und dem 31. Juli 2012 aufgezeichnet wurden, filtern. Das lässt sich jetzt über das in Python bewährte *Slicing*-Verfahren bewerkstelligen – entweder, indem wir einfach den Datums-Auszug als String definieren, der dann geparst wird, oder indem wir vordefinierte Instanzen der *datetime*-Klasse übergeben:

```
# Slicing: Variante 1
df['2011-01-02':'2011-07-31']
# Slicing: Variante 2
from datetime import datetime
start = datetime(day=1, month=2, year=2011)
end = datetime(day=31, month=7, year=2011)
df[start:end]
```

Ausgabe:

```
Date                 p (mbar)  T (degC)
2011-01-02 00:00:00  990.02     0.03
2011-01-02 00:10:00  990.07     0.15
…
2011-07-31 23:40:00  989.98    13.34
2011-07-31 23:50:00  989.97    13.36
```

Wenn wir auf Grundlage einzelner Angaben (Wochentag, Monat, Jahr) Selektionen durchführen möchten, können wir dazu die Attribute des Indexobjekts ansprechen. Im Folgenden selektieren wir zum Beispiel alle Mai-Monate beliebiger Jahre:

```
df[df.index.month == 5]
```

Darüber hinaus lassen sich jetzt alle möglichen datumsbezogenen Berechnungen und Umstrukturierungen durchführen. Dabei hilft uns die Instanzmethode *resample*. Zum Beispiel, wenn wir die Daten von der 10-Minuten-Frequenz in eine stündliche Frequenz *downsamplen* möchten. Der übergebene Parameter (*rule*) definiert die Art des Resamplings. Mit dem anschließenden *mean*-Befehl wird im Beispiel die Berechnung von Mittelwerten über die ursprünglichen Werte eingeleitet, sodass ein neuer Data-Frame entsteht, der aus einer lückenlosen Zeitreihe besteht, in der die Intervalle zwischen den Zeilen genau eine Stunde betragen.

```
df = df.resample(rule='H').mean()
df.head()
```

Ausgabe:

```
date                 p (mbar)  T (degC)
2009-01-01 00:00:00  996.528   -8.304
2009-01-01 01:00:00  996.525   -8.065
2009-01-01 02:00:00  996.745   -8.763
2009-01-01 03:00:00  996.987   -8.897
2009-01-01 04:00:00  997.158   -9.348
```

Der Übergabeparameter *rules* kann dabei mit unterschiedlichen Anweisungen arbeiten, die zu unterschiedlichen Intervallstrukturen führen. Die wichtigsten sind in Tabelle 2.2 verzeichnet.

Tabelle 2.2 Auswahl an Regeln, die die Reindexierung der Resample-Anweisung anleiten

Rule	Bedeutung
D	Kalendertag
B	Arbeitstage (spart Samstage und Sonntage aus)
W	Wochentag
M	Monatsende
H	Stunde
T	Minute
S	Sekunde

Sobald wir *resample* einsetzen bzw. immer dann, wenn wir davon ausgehen können, dass die Intervalle einer Zeitreihe fixiert sind, sollten wir dem Datumsindex des Data-Frames das mitteilen. Das hat im Moment noch keine weitreichenden Konsequenzen. Später, wenn wir mit ARIMA-Modellen arbeiten oder bestimmte Funktionen verwenden, die mit Datumsangaben arbeiten, kann das jedoch hilfreich sein.

Um diese Einstellung vorzunehmen, müssen wir den Index des Data-Frames noch einmal anfassen. Dazu greifen wir auf das Attribut *freq* des Index-Objekts des Data-Frames zu und stellen es entsprechend ein:

```
df.index.freq = 'H'
```

Wenn wir anschließend das Attribut *freq* abfragen, erhalten wir die Information *Hour* zurück. Damit signalisiert der Data-Frame nach außen, dass die Intervalle zwischen den Zeilen jeweils auf eine Stunde eingestellt sind.

2.2 Operationen entlang der Zeitachse

Zeitreihen unterscheiden sich von Querschnittsdaten im Wesentlichen dadurch, dass in Zeitreihen spezifische Korrelationsverhältnisse über die Zeitachse vorherrschen. Hintergrund ist die Annahme bzw. die Beobachtung, dass ein Objekt, das als Einheit betrachtet wird, den Zustand, in dem es sich zu einem Zeitpunkt befindet, nicht willkürlich wechseln kann.

Nehmen wir an, es handle sich bei dem Objekt um einen Ort und die Messgröße sei die Temperatur, die wir monatlich beobachten. Die Temperatur im Januar ist der Temperatur im Dezember oder im Februar wesentlich ähnlicher als der Temperatur im August. Je weiter wir uns von einem Zeitpunkt entfernen, so könnte man verallgemeinern, umso unterschiedlicher sind die Temperaturwerte.

Das trifft aber nicht ganz zu. Unsere Zeitreihe weist schließlich eine zyklische Struktur auf, in der die Korrelationen zwischen den gemessenen Temperaturwerten zunächst abnehmen, um dann wieder zuzunehmen. Die Temperatur zum Zeitpunkt t_{12} ist mit hoher Wahrscheinlich sehr ähnlich der Temperatur zum Zeitpunkt t_0, da sich die Erde in diesem Zeitraum einmal komplett um die Sonne gedreht hat.

Diese Korrelationsmuster beinhalten Informationen über die Struktur der Zeitreihe, die sich bestimmte Verfahren zur Prognose von Zuständen in der Zukunft zunutze machen. Wie man solche Muster analysiert und damit Modelle anlernt, sehen wir uns im dritten Kapitel zum Thema ARIMA-Verfahren genauer an. Was uns in diesem Abschnitt interessiert, sind die vorbereitenden Arbeiten an Zeitreihen, die wir auf dem Weg dorthin erledigen müssen.

2.2.1 Mit Lags arbeiten

Unter *Lags* versteht man zeitversetzte Werte einer Reihe. Das Lag erster Ordnung bezeichnet den Messwert des vorherigen Intervalls. In einer diskreten Zeitreihe mit konstanten Intervallen, die in einem Data-Frame abgelegt sind, ist das einfach der Wert aus der vorherigen Zeile.

In Pandas können wir auf die Lags mit der *shift*-Methode zugreifen. Der Parameter *periods* bestimmt dabei, um wie viele Zeitschritte (Intervalle) wir in die Vergangenheit zurückgreifen.

Nehmen wir wieder die Temperaturdaten der Stadt Jena, die wir diesmal in eine monatliche Intervallstruktur gezwungen haben. Wenn wir pro Zeile eine neue Variable mit dem Temperaturwert aus dem vorherigen Monat erhalten möchten, können wir wie folgt vorgehen:

```
df['Lag_1'] = df['T (degC)'].shift(periods=1)
df.head()
```

Ausgabe:

```
date        T (degC)  Lag_1
2009-01-31  -3.627    NaN
2009-02-28   0.170    -3.627
2009-03-31   3.990     0.170
2009-04-30  11.890     3.990
2009-05-31  13.434    11.890
```

2.2.2 Differenzen erster und höherer Ordnung

Die Tatsache, dass wir auf beliebige Lags zugreifen können, lässt sich für verschiedene Zwecke nutzen. Unter anderem dafür, um die Differenzen erster und höherer Ordnung zu bilden. Die Differenzbildung ist immer dann nützlich, wenn man eine Zeitreihe um saisonale oder Trend-Komponenten bereinigen möchte.

Obwohl man in der Praxis meistens mit den *First Differences* arbeitet, kann man theoretisch Differenzen beliebiger Ordnung bilden. Bei den *First Differences* geht man die Reihe der Messwerte Zeile für Zeile durch und subtrahiert vom aktuellen Messwert jeweils das Lag erster Ordnung. Bei Differenzen zweiter Ordnung wendet man das gleiche Verfahren über das Ergebnis der First Differences noch einmal an.

Sehen wir uns dazu ein einfaches Beispiel an. In Bild 2.1 (links) ist eine einfache Zeitreihe über einige Minuten abgedruckt, die einen sinusartigen Verlauf nimmt. Die Messwerte liegen in einem Data-Frame unter der Spalte *value* vor, und der Data-Frame weist einen Datumsindex auf, der in 10-Sekunden-Intervalle getaktet ist.

Setzen wir jetzt also die First Differences ein, um zu sehen, welche Varianzen übrig bleiben, wenn wir die zyklische Komponente aus der Reihe entfernen. In Pandas können wir dazu entweder mit dem *shift*-Operator arbeiten und damit auf das erste Lag zugreifen, das wir dann vom jeweils aktuellen Messwert subtrahieren. Oder wir schlagen den einfacheren Weg ein und verwenden gleich die *diff*-Methode, die uns Pandas anbietet. Sie erledigt die gleiche Arbeit im Hintergrund für uns: