Introduction\_Machine\_Learning\_Python\_Andreas\_C01

Introduction to Machine Learning

with Python

A Guide for Data Scientists

Andreas C. Müller and Sarah Guido

**CHAPTER 1**

**Introduction**

Machine learning is about extracting knowledge from data. It is a research field at the intersection of statistics, artificial intelligence, and computer science and is also known as predictive analytics or statistical learning. The application of machine learning methods has in recent years become ubiquitous in everyday life. From automatic recommendations of which movies to watch, to what food to order or which products to buy, to personalized online radio and recognizing your friends in your photos, many modern websites and devices have machine learning algorithms at their core. When you look at a complex website like Facebook, Amazon, or Netflix, it is very likely that every part of the site contains multiple machine learning models.

El aprendizaje automático consiste en extraer conocimiento de los datos. Es un campo de investigación en la intersección de la estadística, la inteligencia artificial y la informática y también se conoce como **análisis predictivo o aprendizaje estadístico**. En los últimos años, la aplicación de métodos de aprendizaje automático se ha vuelto omnipresente en la vida cotidiana. Desde recomendaciones automáticas sobre qué películas mirar, qué comida pedir o qué productos comprar, hasta radio en línea personalizada y reconocer a tus amigos en tus fotos, muchos sitios web y dispositivos modernos tienen algoritmos de aprendizaje automático en su núcleo. Cuando observa un sitio web complejo como Facebook, Amazon o Netflix, es muy probable que cada parte del sitio contenga múltiples modelos de aprendizaje automático.

Outside of commercial applications, machine learning has had a tremendous influence on the way data-driven research is done today. The tools introduced in this book have been applied to diverse scientific problems such as understanding stars, finding distant planets, discovering new particles, analyzing DNA sequences, and providing personalized cancer treatments.

Fuera de las aplicaciones comerciales, el aprendizaje automático ha tenido una enorme influencia en la forma en que se realiza hoy la investigación basada en datos. Las herramientas presentadas en este libro se han aplicado a diversos problemas científicos, como **comprender las estrellas, encontrar planetas distantes, descubrir nuevas partículas,** analizar secuencias de ADN y proporcionar tratamientos personalizados contra el cáncer.

Your application doesn’t need to be as large-scale or world-changing as these examples in order to benefit from machine learning, though. In this chapter, we will explain why machine learning has become so popular and discuss what kinds of problems can be solved using machine learning. Then, we will show you how to build your first machine learning model, introducing important concepts along the way.

Sin embargo, no es necesario que su aplicación sea de gran escala o que cambie el mundo como estos ejemplos para poder beneficiarse del aprendizaje automático. En este capítulo, explicaremos por qué el **aprendizaje automático se ha vuelto tan popular** y discutiremos qué tipos de problemas se pueden resolver utilizando el aprendizaje automático. Luego, le mostraremos cómo construir su primer modelo de aprendizaje automático, introduciendo conceptos importantes a lo largo del camino.

**Why Machine Learning?**

In the early days of “intelligent” applications, many systems used handcoded rules of “if ” and “else” decisions to process data or adjust to user input. Think of a spam filter whose job is to move the appropriate incoming email messages to a spam folder. You could make up a blacklist of words that would result in an email being marked as spam. This would be an example of using an expert-designed rule system to design an “intelligent” application. Manually crafting decision rules is feasible for some applications, particularly those in which humans have a good understanding of the process to model. However, using handcoded rules to make decisions has two major disadvantages:

En los primeros días de las aplicaciones "inteligentes", muchos sistemas utilizaban reglas codificadas a mano de decisiones "si" y "si no" para procesar datos o ajustarse a las entradas del usuario. Piense en un filtro de spam cuyo trabajo es mover los mensajes de correo electrónico entrantes apropiados a una carpeta de spam. Podrías crear una lista negra de palabras que daría como resultado que un correo electrónico se marque como spam. Este sería un ejemplo del uso de un sistema de reglas diseñado por expertos para diseñar una aplicación "inteligente". La elaboración manual de reglas de decisión es factible para algunas aplicaciones, particularmente aquellas en las que los humanos tienen una buena comprensión del proceso a modelar. Sin embargo, utilizar reglas codificadas a mano para tomar decisiones tiene dos desventajas principales:

* The logic required to make a decision is specific to a single domain and task. Changing the task even slightly might require a rewrite of the whole system.
* Designing rules requires a deep understanding of how a decision should be made by a human expert.

One example of where this handcoded approach will fail is in detecting faces in images. Today, every smartphone can detect a face in an image. However, face detection was an unsolved problem until as recently as 2001. The main problem is that the way in which pixels (which make up an image in a computer) are “perceived” by the computer is very different from how humans perceive a face. This difference in representation makes it basically impossible for a human to come up with a good set of rules to describe what constitutes a face in a digital image.

Un ejemplo de dónde fallará este enfoque codificado a mano es en la detección de rostros en imágenes. Hoy en día, todos los teléfonos inteligentes pueden detectar una cara en una imagen. Sin embargo, la detección de rostros era un problema sin resolver hasta 2001. El problema principal es que la forma en que la computadora "percibe" los píxeles (que forman una imagen en una computadora) es muy diferente de cómo los humanos perciben un rostro. . Esta diferencia en la representación hace que sea básicamente imposible que un ser humano encuentre un buen conjunto de reglas para describir lo que constituye un rostro en una imagen digital.

Using machine learning, however, simply presenting a program with a large collection of images of faces is enough for an algorithm to determine what characteristics are needed to identify a face.

Sin embargo, utilizando el aprendizaje automático, simplemente presentar un programa con una gran colección de imágenes de rostros es suficiente para que un algoritmo determine qué características se necesitan para identificar un rostro.

**Problems Machine Learning Can Solve**

The most successful kinds of machine learning algorithms are those that automate decision-making processes by generalizing from known examples. In this setting, which is known as supervised learning, the user provides the algorithm with pairs of inputs and desired outputs, and the algorithm finds a way to produce the desired output given an input. In particular, the algorithm is able to create an output for an input it has never seen before without any help from a human. Going back to our example of spam classification, using machine learning, the user provides the algorithm with a large number of emails (which are the input), together with information about whether any of these emails are spam (which is the desired output). Given a new email, the algorithm will then produce a prediction as to whether the new email is spam.

Los tipos de algoritmos de aprendizaje automático más exitosos son aquellos que automatizan los procesos de toma de decisiones generalizando a partir de ejemplos conocidos. En esta configuración, que se conoce como aprendizaje supervisado, el usuario proporciona al algoritmo pares de entradas y salidas deseadas, y el algoritmo encuentra una manera de producir la salida deseada dada una entrada. En particular, el algoritmo es capaz de crear una salida para una entrada que nunca antes había visto sin la ayuda de un humano. Volviendo a nuestro ejemplo de clasificación de spam, utilizando el aprendizaje automático, el usuario proporciona al algoritmo una gran cantidad de correos electrónicos (que son la entrada), junto con información sobre si alguno de estos correos electrónicos es spam (que es el resultado deseado). Dado un nuevo correo electrónico, el algoritmo producirá una predicción sobre si el nuevo correo electrónico es spam.

Machine learning algorithms that learn from input/output pairs are called supervised learning algorithms because a “teacher” provides supervision to the algorithms in the form of the desired outputs for each example that they learn from. While creating a dataset of inputs and outputs is often a laborious manual process, supervised learning algorithms are well understood and their performance is easy to measure. If your application can be formulated as a supervised learning problem, and you are able to create a dataset that includes the desired outcome, machine learning will likely be able to solve your problem.

Los algoritmos de aprendizaje automático que aprenden de pares de entrada/salida se denominan algoritmos de aprendizaje supervisado porque un "maestro" proporciona supervisión a los algoritmos en forma de los resultados deseados para cada ejemplo del que aprenden. Si bien la creación de un conjunto de datos de entradas y salidas suele ser un proceso manual laborioso, los algoritmos de aprendizaje supervisado se comprenden bien y su rendimiento es fácil de medir. Si su aplicación puede formularse como un problema de aprendizaje supervisado y puede crear un conjunto de datos que incluya el resultado deseado, es probable que el aprendizaje automático pueda resolver su problema.

**Examples of supervised machine learning tasks include:**

**Identifying the zip code from handwritten digits on an envelope**

Here the input is a scan of the handwriting, and the desired output is the actual digits in the zip code. To create a dataset for building a machine learning model, you need to collect many envelopes. Then you can read the zip codes yourself and store the digits as your desired outcomes.

Aquí la entrada es un escaneo de la escritura a mano y la salida deseada son los dígitos reales del código postal. Para crear un conjunto de datos para construir un modelo de aprendizaje automático, es necesario recopilar muchos sobres. Luego puede leer los códigos postales usted mismo y almacenar los dígitos según los resultados deseados.

**Determining whether a tumor is benign based on a medical image**

Here the input is the image, and the output is whether the tumor is benign. To create a dataset for building a model, you need a database of medical images. You also need an expert opinion, so a doctor needs to look at all of the images and decide which tumors are benign and which are not. It might even be necessary to do additional diagnosis beyond the content of the image to determine whether the tumor in the image is cancerous or not.

Aquí la entrada es la imagen y la salida es si el tumor es benigno. Para crear un conjunto de datos para construir un modelo, necesita una base de datos de imágenes médicas. También se necesita la opinión de un experto, por lo que un médico debe observar todas las imágenes y decidir qué tumores son benignos y cuáles no. Incluso podría ser necesario realizar un diagnóstico adicional más allá del contenido de la imagen para determinar si el tumor en la imagen es canceroso o no.

**Detecting fraudulent activity in credit card transactions**

Here the input is a record of the credit card transaction, and the output is whether it is likely to be fraudulent or not. Assuming that you are the entity distributing the credit cards, collecting a dataset means storing all transactions and recording if a user reports any transaction as fraudulent.

Aquí la entrada es un registro de la transacción con tarjeta de crédito y la salida es si es probable que sea fraudulenta o no. Suponiendo que usted es la entidad que distribuye las tarjetas de crédito, recopilar un conjunto de datos significa almacenar todas las transacciones y registrar si un usuario informa que alguna transacción es fraudulenta.

An interesting thing to note about these examples is that although the inputs and outputs look fairly straightforward, the data collection process for these three tasks is vastly different. While reading envelopes is laborious, it is easy and cheap. Obtaining medical imaging and diagnoses, on the other hand, requires not only expensive machinery but also rare and expensive expert knowledge, not to mention the ethical concerns and privacy issues. In the example of detecting credit card fraud, data collection is much simpler. Your customers will provide you with the desired output, as they will report fraud. All you have to do to obtain the input/output pairs of fraudulent and nonfraudulent activity is wait.

Un aspecto interesante a tener en cuenta sobre estos ejemplos es que, si bien las entradas y salidas parecen bastante sencillas, el proceso de recopilación de datos para estas tres tareas es muy diferente. Si bien leer sobres es laborioso, es fácil y económico. La obtención de imágenes y diagnósticos médicos, por otro lado, requiere no sólo maquinaria costosa sino también conocimientos expertos poco comunes y costosos, sin mencionar las preocupaciones éticas y las cuestiones de privacidad. En el ejemplo de la detección de fraudes con tarjetas de crédito, la recopilación de datos es mucho más sencilla. Sus clientes le proporcionarán el resultado deseado, ya que denunciarán el fraude. Todo lo que tienes que hacer para obtener los pares de entrada/salida de actividad fraudulenta y no fraudulenta es esperar.

Unsupervised algorithms are the other type of algorithm that we will cover in this book. In unsupervised learning, only the input data is known, and no known output data is given to the algorithm. While there are many successful applications of these methods, they are usually harder to understand and evaluate.

Los algoritmos no supervisados ​​son el otro tipo de algoritmo que cubriremos en este libro. En el aprendizaje no supervisado, sólo se conocen los datos de entrada y no se proporcionan datos de salida conocidos al algoritmo. Si bien existen muchas aplicaciones exitosas de estos métodos, generalmente son más difíciles de comprender y evaluar.

**Examples of unsupervised learning include:**

**Identifying topics in a set of blog posts**

If you have a large collection of text data, you might want to summarize it and find prevalent themes in it. You might not know beforehand what these topics are, or how many topics there might be. Therefore, there are no known outputs.

Si tiene una gran colección de datos de texto, es posible que desee resumirla y encontrar temas frecuentes en ella. Es posible que no sepa de antemano cuáles son estos temas o cuántos temas puede haber. Por lo tanto, no se conocen resultados.

**Segmenting customers into groups with similar preferences**

Given a set of customer records, you might want to identify which customers are similar, and whether there are groups of customers with similar preferences. For a shopping site, these might be “parents,” “bookworms,” or “gamers.” Because you don’t know in advance what these groups might be, or even how many there are, you have no known outputs.

Dado un conjunto de registros de clientes, es posible que desee identificar qué clientes son similares y si hay grupos de clientes con preferencias similares. Para un sitio de compras, estos podrían ser "padres", "ratones de biblioteca" o "jugadores". Como no se sabe de antemano cuáles podrían ser estos grupos, ni siquiera cuántos hay, no se conocen resultados.

**Detecting abnormal access patterns to a website**

To identify abuse or bugs, it is often helpful to find access patterns that are different from the norm. Each abnormal pattern might be very different, and you might not have any recorded instances of abnormal behavior. Because in this example you only observe traffic, and you don’t know what constitutes normal and abnormal behavior, this is an unsupervised problem.

Para identificar abusos o errores, suele resultar útil encontrar patrones de acceso que sean diferentes de la norma. Cada patrón anormal puede ser muy diferente y es posible que no tenga ningún caso registrado de comportamiento anormal. Debido a que en este ejemplo sólo observa el tráfico y no sabe qué constituye un comportamiento normal y anormal, se trata de un problema no supervisado.

For both supervised and unsupervised learning tasks, it is important to have a representation of your input data that a computer can understand. Often it is helpful to think of your data as a table. Each data point that you want to reason about (each email, each customer, each transaction) is a row, and each property that describes that data point (say, the age of a customer or the amount or location of a transaction) is a column. You might describe users by their age, their gender, when they created an account, and how often they have bought from your online shop. You might describe the image of a tumor by the grayscale values of each pixel, or maybe by using the size, shape, and color of the tumor.

Tanto para las tareas de aprendizaje supervisadas como para las no supervisadas, es importante tener una representación de los datos de entrada que una computadora pueda entender. A menudo resulta útil pensar en los datos como una tabla. Cada punto de datos sobre el que desea razonar (cada correo electrónico, cada cliente, cada transacción) es una fila, y cada propiedad que describe ese punto de datos (por ejemplo, la edad de un cliente o el monto o ubicación de una transacción) es una columna. Puede describir a los usuarios por su edad, sexo, cuándo crearon una cuenta y con qué frecuencia compraron en su tienda en línea. Podría describir la imagen de un tumor mediante los valores de escala de grises de cada píxel, o tal vez utilizando el tamaño, la forma y el color del tumor.

Each entity or row here is known as a sample (or data point) in machine learning, while the columns—the properties that describe these entities—are called features.

Cada entidad o fila aquí se conoce como muestra (o punto de datos) en el aprendizaje automático, mientras que las columnas (las propiedades que describen estas entidades) se denominan características.

Later in this book we will go into more detail on the topic of building a good representation of your data, which is called feature extraction or feature engineering. You should keep in mind, however, that no machine learning algorithm will be able to make a prediction on data for which it has no information. For example, if the only feature that you have for a patient is their last name, no algorithm will be able to predict their gender. This information is simply not contained in your data. If you add another feature that contains the patient’s first name, you will have much better luck, as it is often possible to tell the gender by a person’s first name.

Más adelante en este libro entraremos en más detalles sobre el tema de crear una buena representación de sus datos, lo que se denomina extracción de características o ingeniería de características. Sin embargo, debes tener en cuenta que ningún algoritmo de aprendizaje automático podrá hacer una predicción sobre datos de los que no tiene información. Por ejemplo, si la única característica que tiene para un paciente es su apellido, ningún algoritmo podrá predecir su sexo. Esta información simplemente no está contenida en sus datos. Si agrega otra característica que contenga el nombre del paciente, tendrá mucha mejor suerte, ya que a menudo es posible saber el género por el nombre de una persona.

**Knowing Your Task and Knowing Your Data**

Quite possibly the most important part in the machine learning process is understanding the data you are working with and how it relates to the task you want to solve. It will not be effective to randomly choose an algorithm and throw your data at it. It is necessary to understand what is going on in your dataset before you begin building a model. Each algorithm is different in terms of what kind of data and what problem setting it works best for. While you are building a machine learning solution, you should answer, or at least keep in mind, the following questions:

Posiblemente la parte más importante del proceso de aprendizaje automático sea comprender los datos con los que está trabajando y cómo se relacionan con la tarea que desea resolver. No será efectivo elegir un algoritmo al azar y arrojarle sus datos. Es necesario comprender qué sucede en su conjunto de datos antes de comenzar a construir un modelo. Cada algoritmo es diferente en términos de qué tipo de datos y para qué configuración de problema funciona mejor. Mientras crea una solución de aprendizaje automático, debe responder, o al menos tener en cuenta, las siguientes preguntas:

• What question(s) am I trying to answer? Do I think the data collected can answer that question?

• What is the best way to phrase my question(s) as a machine learning problem?

• Have I collected enough data to represent the problem I want to solve?

• What features of the data did I extract, and will these enable the right predictions?

• How will I measure success in my application?

• How will the machine learning solution interact with other parts of my research or business product?

In a larger context, the algorithms and methods in machine learning are only one part of a greater process to solve a particular problem, and it is good to keep the big picture in mind at all times. Many people spend a lot of time building complex machine learning solutions, only to find out they don’t solve the right problem.

En un contexto más amplio, los algoritmos y métodos del aprendizaje automático son solo una parte de un proceso mayor para resolver un problema particular, y es bueno tener presente el panorama general en todo momento. Muchas personas dedican mucho tiempo a crear soluciones complejas de aprendizaje automático, sólo para descubrir que no resuelven el problema correcto.

When going deep into the technical aspects of machine learning (as we will in this book), it is easy to lose sight of the ultimate goals. While we will not discuss the questions listed here in detail, we still encourage you to keep in mind all the assumptions that you might be making, explicitly or implicitly, when you start building machine learning models.

Al profundizar en los aspectos técnicos del aprendizaje automático (como lo haremos en este libro), es fácil perder de vista los objetivos finales. Si bien no discutiremos las preguntas enumeradas aquí en detalle, le recomendamos que tenga en cuenta todas las suposiciones que podría estar haciendo, explícita o implícitamente, cuando comience a crear modelos de aprendizaje automático.

**Why Python?**

Python has become the lingua franca for many data science applications. It combines the power of general-purpose programming languages with the ease of use of domain-specific scripting languages like MATLAB or R. Python has libraries for data loading, visualization, statistics, natural language processing, image processing, and more. This vast toolbox provides data scientists with a large array of general- and special-purpose functionality. One of the main advantages of using Python is the ability to interact directly with the code, using a terminal or other tools like the Jupyter Notebook, which we’ll look at shortly. Machine learning and data analysis are fundamentally iterative processes, in which the data drives the analysis. It is essential for these processes to have tools that allow quick iteration and easy interaction.

Python se ha convertido en la lengua franca de muchas aplicaciones de ciencia de datos. Combina el poder de los lenguajes de programación de propósito general con la facilidad de uso de lenguajes de programación de dominios específicos como MATLAB o R. Python tiene bibliotecas para carga de datos, visualización, estadísticas, procesamiento de lenguaje natural, procesamiento de imágenes y más. Esta amplia caja de herramientas proporciona a los científicos de datos una amplia gama de funciones generales y especiales. Una de las principales ventajas de usar Python es la capacidad de interactuar directamente con el código, usando una terminal u otras herramientas como Jupyter Notebook, que veremos en breve. El aprendizaje automático y el análisis de datos son procesos fundamentalmente iterativos, en los que los datos impulsan el análisis. Es fundamental que estos procesos cuenten con herramientas que permitan una rápida iteración y una fácil interacción.

As a general-purpose programming language, Python also allows for the creation of complex graphical user interfaces (GUIs) and web services, and for integration into existing systems.

Como lenguaje de programación de propósito general, Python también permite la creación de interfaces gráficas de usuario (GUI) y servicios web complejos, y la integración en sistemas existentes.

**scikit-learn**

scikit-learn is an open source project, meaning that it is free to use and distribute, and anyone can easily obtain the source code to see what is going on behind the scenes. The scikit-learn project is constantly being developed and improved, and it has a very active user community. It contains a number of state-of-the-art machine learning algorithms, as well as comprehensive documentation about each algorithm. scikit-learn is a very popular tool, and the most prominent Python library for machine learning. It is widely used in industry and academia, and a wealth of tutorials and code snippets are available online. scikit-learn works well with a number of other scientific Python tools, which we will discuss later in this chapter.

scikit-learn es un proyecto de código abierto, lo que significa que su uso y distribución son gratuitos, y cualquiera puede obtener fácilmente el código fuente para ver qué sucede detrás de escena. El proyecto scikit-learn se desarrolla y mejora constantemente y cuenta con una comunidad de usuarios muy activa. Contiene una serie de algoritmos de aprendizaje automático de última generación, así como documentación completa sobre cada algoritmo. scikit-learn es una herramienta muy popular y la biblioteca de Python más destacada para el aprendizaje automático. Se utiliza ampliamente en la industria y el mundo académico, y hay una gran cantidad de tutoriales y fragmentos de código disponibles en línea. scikit-learn funciona bien con otras herramientas científicas de Python, que discutiremos más adelante en este capítulo.

While reading this, we recommend that you also browse the scikit-learn user guide and API documentation for additional details on and many more options for each algorithm. The online documentation is very thorough, and this book will provide you with all the prerequisites in machine learning to understand it in detail.

Mientras lee esto, le recomendamos que también consulte la guía del usuario de scikit-learn y la documentación de la API para obtener detalles adicionales y muchas más opciones para cada algoritmo. La documentación en línea es muy completa y este libro le proporcionará todos los requisitos previos del aprendizaje automático para comprenderlo en detalle.

**Installing scikit-learn**

scikit-learn depends on two other Python packages, NumPy and SciPy. For plotting and interactive development, you should also install matplotlib, IPython, and the Jupyter Notebook. We recommend using one of the following prepackaged Python distributions, which will provide the necessary packages:

scikit-learn depende de otros dos paquetes de Python, NumPy y SciPy. Para el trazado y el desarrollo interactivo, también debe instalar matplotlib, IPython y Jupyter Notebook. Recomendamos utilizar una de las siguientes distribuciones de Python empaquetadas, que proporcionarán los paquetes necesarios:

**Anaconda**

A Python distribution made for large-scale data processing, predictive analytics, and scientific computing. Anaconda comes with NumPy, SciPy, matplotlib, pandas, IPython, Jupyter Notebook, and scikit-learn. Available on Mac OS, Windows, and Linux, it is a very convenient solution and is the one we suggest for people without an existing installation of the scientific Python packages. Anaconda now also includes the commercial Intel MKL library for free. Using MKL (which is done automatically when Anaconda is installed) can give significant speed improvements for many algorithms in scikit-learn.

Una distribución de Python creada para el procesamiento de datos a gran escala, análisis predictivo e informática científica. Anaconda viene con NumPy, SciPy, matplotlib, pandas, IPython, Jupyter Notebook y scikit-learn. Disponible en Mac OS, Windows y Linux, es una solución muy conveniente y es la que sugerimos para las personas que no tienen una instalación existente de los paquetes científicos de Python. Anaconda ahora también incluye la biblioteca comercial Intel MKL de forma gratuita. El uso de MKL (que se realiza automáticamente cuando se instala Anaconda) puede brindar mejoras de velocidad significativas para muchos algoritmos en scikit-learn.

**Enthought Canopy**

Another Python distribution for scientific computing. This comes with NumPy, SciPy, matplotlib, pandas, and IPython, but the free version does not come with scikit-learn. If you are part of an academic, degree-granting institution, you can request an academic license and get free access to the paid subscription version of Enthought Canopy. Enthought Canopy is available for Python 2.7.x, and works on Mac OS, Windows, and Linux.

Otra distribución de Python para informática científica. Viene con NumPy, SciPy, matplotlib, pandas e IPython, pero la versión gratuita no viene con scikit-learn. Si forma parte de una institución académica que otorga títulos, puede solicitar una licencia académica y obtener acceso gratuito a la versión de suscripción paga de Enthink Canopy. Enthink Canopy está disponible para Python 2.7.x y funciona en Mac OS, Windows y Linux.

**Python(x,y)**

A free Python distribution for scientific computing, specifically for Windows. Python(x,y) comes with NumPy, SciPy, matplotlib, pandas, IPython, and scikit-learn.

If you already have a Python installation set up, you can use pip to install all of these packages:

$ **pip install numpy scipy matplotlib ipython scikit-learn pandas pillow**

For the tree visualizations in Chapter 2, you also need the graphviz packages; see the accompanying code for instructions.

**Essential Libraries and Tools**

Understanding what scikit-learn is and how to use it is important, but there are a few other libraries that will enhance your experience. scikit-learn is built on top of the NumPy and SciPy scientific Python libraries. In addition to NumPy and SciPy, we will be using pandas and matplotlib. We will also introduce the Jupyter Notebook, which is a browser-based interactive programming environment. Briefly, here is what you should know about these tools in order to get the most out of scikit-learn.1

Es importante comprender qué es scikit-learn y cómo usarlo, pero existen algunas otras bibliotecas que mejorarán su experiencia. scikit-learn se basa en las bibliotecas científicas de Python NumPy y SciPy. Además de NumPy y SciPy, usaremos pandas y matplotlib. También presentaremos Jupyter Notebook, que es un entorno de programación interactivo basado en navegador. Brevemente, esto es lo que debe saber sobre estas herramientas para aprovechar al máximo scikit-learn.1

**Jupyter Notebook**

The Jupyter Notebook is an interactive environment for running code in the browser. It is a great tool for exploratory data analysis and is widely used by data scientists. While the Jupyter Notebook supports many programming languages, we only need the Python support. The Jupyter Notebook makes it easy to incorporate code, text, and images, and all of this book was in fact written as a Jupyter Notebook. All of the code examples we include can be downloaded from GitHub.

Jupyter Notebook es un entorno interactivo para ejecutar código en el navegador. Es una gran herramienta para el análisis de datos exploratorio y es ampliamente utilizada por los científicos de datos. Si bien Jupyter Notebook admite muchos lenguajes de programación, solo necesitamos compatibilidad con Python. Jupyter Notebook facilita la incorporación de código, texto e imágenes y, de hecho, todo este libro fue escrito como un Jupyter Notebook. Todos los ejemplos de código que incluimos se pueden descargar desde GitHub.

**NumPy**

NumPy is one of the fundamental packages for scientific computing in Python. It contains functionality for multidimensional arrays, high-level mathematical functions such as linear algebra operations and the Fourier transform, and pseudorandom number generators.

NumPy es uno de los paquetes fundamentales para la informática científica en Python. Contiene funcionalidad para matrices multidimensionales, funciones matemáticas de alto nivel como operaciones de álgebra lineal y la transformada de Fourier, y generadores de números pseudoaleatorios.

In scikit-learn, the NumPy array is the fundamental data structure. scikit-learn takes in data in the form of NumPy arrays. Any data you’re using will have to be converted to a NumPy array. The core functionality of NumPy is the ndarray class, a multidimensional (n-dimensional) array. All elements of the array must be of the same type. A NumPy array looks like this:

En scikit-learn, la matriz NumPy es la estructura de datos fundamental. scikit-learn toma datos en forma de matrices NumPy. Cualquier dato que esté utilizando deberá convertirse a una matriz NumPy. La funcionalidad principal de NumPy es la clase ndarray, una matriz multidimensional (n-dimensional). Todos los elementos de la matriz deben ser del mismo tipo. Una matriz NumPy se ve así:

1 If you are unfamiliar with NumPy or matplotlib, we recommend reading the first chapter of the SciPy Lecture Notes.

In[1]:

import numpy as np

x = np.array([[1, 2, 3], [4, 5, 6]])

print("x:\n{}".format(x))

Out[1]:

x:

[[1 2 3]

[4 5 6]]

We will be using NumPy a lot in this book, and we will refer to objects of the NumPy ndarray class as “NumPy arrays” or just “arrays.”

Usaremos mucho NumPy en este libro y nos referiremos a los objetos de la clase ndarray NumPy como "matrices NumPy" o simplemente "matrices".

**SciPy**

SciPy is a collection of functions for scientific computing in Python. It provides, among other functionality, advanced linear algebra routines, mathematical function optimization, signal processing, special mathematical functions, and statistical distributions. scikit-learn draws from SciPy’s collection of functions for implementing its algorithms. The most important part of SciPy for us is scipy.sparse: this provides sparse matrices, which are another representation that is used for data in scikit-learn. Sparse matrices are used whenever we want to store a 2D array that contains mostly zeros:

SciPy es una colección de funciones para informática científica en Python. Proporciona, entre otras funciones, rutinas avanzadas de álgebra lineal, optimización de funciones matemáticas, procesamiento de señales, funciones matemáticas especiales y distribuciones estadísticas. scikit-learn se basa en la colección de funciones de SciPy para implementar sus algoritmos. La parte más importante de SciPy para nosotros es scipy.sparse: proporciona matrices dispersas, que son otra representación que se utiliza para los datos en scikit-learn. Las matrices dispersas se utilizan siempre que queremos almacenar una matriz 2D que contiene principalmente ceros:

In[2]:

from scipy import sparse

# Create a 2D NumPy array with a diagonal of ones, and zeros everywhere else

eye = np.eye(4)

print("NumPy array:\n{}".format(eye))

Out[2]:

NumPy array:

[[ 1. 0. 0.

[ 0. 1. 0.

[ 0. 0. 1.

[ 0. 0. 0.

0.]

0.]

0.]

1.]]

In[3]:

# Convert the NumPy array to a SciPy sparse matrix in CSR format

# Only the nonzero entries are stored

sparse\_matrix = sparse.csr\_matrix(eye)

print("\nSciPy sparse CSR matrix:\n{}".format(sparse\_matrix))

Out[3]:

SciPy sparse CSR matrix:

(0, 0)

1.0

(1, 1)

1.0

(2, 2)

1.0

(3, 3)

1.0

Usually it is not possible to create dense representations of sparse data (as they would not fit into memory), so we need to create sparse representations directly. Here is a way to create the same sparse matrix as before, using the COO format:

Normalmente no es posible crear representaciones densas de datos dispersos (ya que no caben en la memoria), por lo que necesitamos crear representaciones dispersas directamente. Aquí hay una manera de crear la misma matriz dispersa que antes, usando el formato COO:

In[4]:

data = np.ones(4)

row\_indices = np.arange(4)

col\_indices = np.arange(4)

eye\_coo = sparse.coo\_matrix((data, (row\_indices, col\_indices)))

print("COO representation:\n{}".format(eye\_coo))

Out[4]:

COO representation:

(0, 0)

1.0

(1, 1)

1.0

(2, 2)

1.0

(3, 3)

1.0

More details on SciPy sparse matrices can be found in the SciPy Lecture Notes.

**matplotlib**

matplotlib is the primary scientific plotting library in Python. It provides functions for making publication-quality visualizations such as line charts, histograms, scatter plots, and so on. Visualizing your data and different aspects of your analysis can give you important insights, and we will be using matplotlib for all our visualizations. When working inside the Jupyter Notebook, you can show figures directly in the browser by using the %matplotlib notebook and %matplotlib inline commands. We recommend using %matplotlib notebook, which provides an interactive environment (though we are using %matplotlib inline to produce this book). For example, this code produces the plot in Figure 1-1:

matplotlib es la principal biblioteca de trazado científico en Python. Proporciona funciones para realizar visualizaciones con calidad de publicación, como gráficos de líneas, histogramas, diagramas de dispersión, etc. Visualizar sus datos y diferentes aspectos de su análisis puede brindarle información importante y usaremos matplotlib para todas nuestras visualizaciones. Cuando trabaja dentro de Jupyter Notebook, puede mostrar figuras directamente en el navegador utilizando los comandos %matplotlib notebook y %matplotlib en línea. Recomendamos usar %matplotlib notebook, que proporciona un entorno interactivo (aunque estamos usando %matplotlib en línea para producir este libro). Por ejemplo, este código produce el gráfico de la Figura 1-1:

In[5]:

%matplotlib inline

import matplotlib.pyplot as plt

# Generate a sequence of numbers from -10 to 10 with 100 steps in between

x = np.linspace(-10, 10, 100)

# Create a second array using sine

y = np.sin(x)

# The plot function makes a line chart of one array against another

plt.plot(x, y, marker="x")

Figure 1-1. Simple line plot of the sine function using matplotlib

**pandas**

pandas is a Python library for data wrangling and analysis. It is built around a data structure called the DataFrame that is modeled after the R DataFrame. Simply put, a pandas DataFrame is a table, similar to an Excel spreadsheet. pandas provides a great range of methods to modify and operate on this table; in particular, it allows SQL-like queries and joins of tables. In contrast to NumPy, which requires that all entries in an array be of the same type, pandas allows each column to have a separate type (for example, integers, dates, floating-point numbers, and strings). Another valuable tool provided by pandas is its ability to ingest from a great variety of file formats and data‐bases, like SQL, Excel files, and comma-separated values (CSV) files. Going into detail about the functionality of pandas is out of the scope of this book. However, Python for Data Analysis by Wes McKinney (O’Reilly, 2012) provides a great guide. Here is a small example of creating a DataFrame using a dictionary:

pandas es una biblioteca de Python para la manipulación y el análisis de datos. Está construido alrededor de una estructura de datos llamada DataFrame que se modela a partir del R DataFrame. En pocas palabras, un DataFrame de pandas es una tabla, similar a una hoja de cálculo de Excel. pandas proporciona una gran variedad de métodos para modificar y operar en esta tabla; en particular, permite consultas similares a SQL y uniones de tablas. A diferencia de NumPy, que requiere que todas las entradas de una matriz sean del mismo tipo, pandas permite que cada columna tenga un tipo independiente (por ejemplo, números enteros, fechas, números de punto flotante y cadenas). Otra herramienta valiosa proporcionada por pandas es su capacidad para ingerir desde una gran variedad de formatos de archivos y bases de datos, como archivos SQL, Excel y archivos de valores separados por comas (CSV). Entrar en detalles sobre la funcionalidad de los pandas está fuera del alcance de este libro. Sin embargo, Python para análisis de datos de Wes McKinney (O'Reilly, 2012) proporciona una excelente guía. Aquí hay un pequeño ejemplo de cómo crear un DataFrame usando un diccionario:

In[6]:

import pandas as pd

from IPython.display import display

# create a simple dataset of people

data = {'Name': ["John", "Anna", "Peter", "Linda"],

'Location' : ["New York", "Paris", "Berlin", "London"],

'Age' : [24, 13, 53, 33]

}

data\_pandas = pd.DataFrame(data)

# IPython.display allows "pretty printing" of dataframes

# in the Jupyter notebook

display(data\_pandas)

This produces the following output:

Age Location Name

0 24 New York John

1 13ParisAnna

2 53BerlinPeter

3 33LondonLinda

There are several possible ways to query this table. For example:

In[7]:

# Select all rows that have an age column greater than 30

display(data\_pandas[data\_pandas.Age > 30])

This produces the following result:

Age Location Name

2 53 Berlin

Peter

3 33

London

Linda

**mglearn**

This book comes with accompanying code, which you can find on GitHub. The accompanying code includes not only all the examples shown in this book, but also the mglearn library. This is a library of utility functions we wrote for this book, so that we don’t clutter up our code listings with details of plotting and data loading. If you’re interested, you can look up all the functions in the repository, but the details of the mglearn module are not really important to the material in this book. If you see a call to mglearn in the code, it is usually a way to make a pretty picture quickly, or to get our hands on some interesting data. If you run the notebooks published on Git‐Hub, the mglearn package is already in the right place and you don’t have to worry about it. If you want to call mglearn functions from any other place, the easiest way to install it is by calling pip install mglearn.

Este libro viene con un código adjunto, que puede encontrar en GitHub. El código adjunto incluye no solo todos los ejemplos que se muestran en este libro, sino también la biblioteca mglearn. Esta es una biblioteca de funciones de utilidad que escribimos para este libro, para no saturar nuestras listas de códigos con detalles de trazado y carga de datos. Si está interesado, puede buscar todas las funciones en el repositorio, pero los detalles del módulo mglearn no son realmente importantes para el material de este libro. Si ve una llamada a mglearn en el código, generalmente es una forma de crear una imagen bonita rápidamente o de tener en nuestras manos algunos datos interesantes. Si ejecuta los cuadernos publicados en Git‐Hub, el paquete mglearn ya está en el lugar correcto y no tiene que preocuparse por ello. Si desea llamar a funciones mglearn desde cualquier otro lugar, la forma más sencilla de instalarlas es llamando a pip install mglearn.

Throughout the book we make ample use of NumPy, matplotlib and pandas. All the code will assume the following imports:

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import mglearn

from IPython.display import display

We also assume that you will run the code in a Jupyter Notebook with the %matplotlib notebook or %matplotlib inline magic enabled to show plots. If you are not using the notebook or these magic commands, you will have to call plt.show to actually show any of the figures.

También asumimos que ejecutará el código en un Jupyter Notebook con el cuaderno %matplotlib o la magia en línea %matplotlib habilitada para mostrar gráficos. Si no está utilizando el cuaderno o estos comandos mágicos, tendrá que llamar a plt.show para mostrar cualquiera de las figuras.

**Python 2 Versus Python 3**

There are two major versions of Python that are widely used at the moment: Python 2 (more precisely, 2.7) and Python 3 (with the latest release being 3.5 at the time of writing). This sometimes leads to some confusion. Python 2 is no longer actively developed, but because Python 3 contains major changes, Python 2 code usually does not run on Python 3. If you are new to Python, or are starting a new project from scratch, we highly recommend using the latest version of Python 3 without changes. If you have a large codebase that you rely on that is written for Python 2, you are excused from upgrading for now. However, you should try to migrate to Python 3 as soon as possible. When writing any new code, it is for the most part quite easy to write code that runs under Python 2 and Python 3. 2 If you don’t have to interface with legacy software, you should definitely use Python 3. All the code in this book is written in a way that works for both versions. However, the exact output might differ slightly under Python 2.

**Versions Used in this Book**

We are using the following versions of the previously mentioned libraries in this book:

In[8]:

import sys

print("Python version: {}".format(sys.version))

import pandas as pd

print("pandas version: {}".format(pd.\_\_version\_\_))

import matplotlib

print("matplotlib version: {}".format(matplotlib.\_\_version\_\_))

import numpy as np

print("NumPy version: {}".format(np.\_\_version\_\_))

import scipy as sp

print("SciPy version: {}".format(sp.\_\_version\_\_))

import IPython

print("IPython version: {}".format(IPython.\_\_version\_\_))

import sklearn

print("scikit-learn version: {}".format(sklearn.\_\_version\_\_))

Out[8]:

Python version: 3.5.2 |Continuum Analytics, Inc.| (default,

Jul 2 2016, 17:53:06)

[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)]

pandas version: 0.20.1

matplotlib version: 2.0.1

NumPy version: 1.12.1

SciPy version: 0.19.0

IPython version: 5.1.0

scikit-learn version: 0.19

While it is not important to match these versions exactly, you should have a version of scikit-learn that is as least as recent as the one we used.

Now that we have everything set up, let’s dive into our first application of machine

learning.

This book assumes that you have version 0.18 or later of scikit-learn. The model\_selection module was added in 0.18, and if you use an earlier version of scikit-learn, you will need to adjust the imports from this module.

**A First Application: Classifying Iris Species**

In this section, we will go through a simple machine learning application and create our first model. In the process, we will introduce some core concepts and terms. Let’s assume that a hobby botanist is interested in distinguishing the species of some iris flowers that she has found. She has collected some measurements associated with each iris: the length and width of the petals and the length and width of the sepals, all measured in centimeters (see Figure 1-2).

En esta sección, analizaremos una aplicación simple de **aprendizaje automático** y crearemos nuestro primer **modelo**. En el proceso, introduciremos algunos **conceptos** y **términos** básicos. Supongamos que un botánico aficionado está interesado en distinguir las **especies** de algunas **flores de iris** que ha encontrado. Ha recopilado algunas medidas asociadas con cada iris: el largo y ancho de los **pétalos** y el largo y ancho de los **sépalos**, todos medidos en centímetros (ver Figura 1-2).

She also has the measurements of some irises that have been previously identified by an expert botanist as belonging to the species setosa, versicolor, or virginica. For these measurements, she can be certain of which species each iris belongs to. Let’s assume that these are the only species our hobby botanist will encounter in the wild.

También tiene las medidas de unos iris que han sido previamente identificados por un experto botánico como pertenecientes a las especies setosa, versicolor o virginica. Para estas mediciones, puede estar segura de a qué especie pertenece cada iris. Supongamos que estas son las únicas especies que nuestro botánico aficionado encontrará en la naturaleza.

Our goal is to build a machine learning model that can learn from the measurements of these irises whose species is known, so that we can predict the species for a new iris.

Nuestro objetivo es construir un modelo de aprendizaje automático que pueda aprender de las mediciones de estos iris cuyas especies se conocen, de modo que podamos predecir las especies de un nuevo iris.

Figure 1-2. Parts of the iris flower

Because we have measurements for which we know the correct species of iris, this is a supervised learning problem. In this problem, we want to predict one of several options (the species of iris). This is an example of a classification problem. The possible outputs (different species of irises) are called classes. Every iris in the dataset belongs to one of three classes, so this problem is a three-class classification problem.

Debido a que tenemos mediciones para las cuales conocemos la especie correcta de iris, este es un problema de aprendizaje supervisado. En este problema, queremos predecir una de varias opciones (la especie de iris). Este es un ejemplo de un problema de clasificación. Las posibles salidas (diferentes especies de lirios) se denominan clases. Cada iris del conjunto de datos pertenece a una de tres clases, por lo que este problema es un problema de clasificación de tres clases.

The desired output for a single data point (an iris) is the species of this flower. For a particular data point, the species it belongs to is called its label.

El resultado deseado para un único punto de datos (un iris) es la especie de esta flor. Para un punto de datos en particular, la especie a la que pertenece se denomina etiqueta.

**Meet the Data**

The data we will use for this example is the Iris dataset, a classical dataset in machine learning and statistics. It is included in scikit-learn in the datasets module. We can load it by calling the load\_iris function:

Los datos que utilizaremos para este ejemplo son el conjunto de datos Iris, un conjunto de datos clásico en aprendizaje automático y estadística. Está incluido en scikit-learn en el módulo de conjuntos de datos. Podemos cargarlo llamando a la función load\_iris:

In[9]:

from sklearn.datasets import load\_iris

iris\_dataset = load\_iris()

The iris object that is returned by load\_iris is a Bunch object, which is very similar to a dictionary. It contains keys and values:

In[10]:

print("Keys of iris\_dataset: \n{}".format(iris\_dataset.keys()))

Out[10]:

Keys of iris\_dataset:

dict\_keys(['target\_names', 'feature\_names', 'DESCR', 'data', 'target'])

The value of the key DESCR is a short description of the dataset. We show the beginning of the description here (feel free to look up the rest yourself):

In[11]:

print(iris\_dataset['DESCR'][:193] + "\n...")

Out[11]:

Iris Plants Database

====================

Notes

----

Data Set Characteristics:

:Number of Instances: 150 (50 in each of three classes)

:Number of Attributes: 4 numeric, predictive att

...

----

The value of the key target\_names is an array of strings, containing the species of flower that we want to predict:

El valor de la clave target\_names es una matriz de cadenas que contiene la especie de flor que queremos predecir:

In[12]:

print("Target names: {}".format(iris\_dataset['target\_names']))

Out[12]:

Target names: ['setosa' 'versicolor' 'virginica']

The value of feature\_names is a list of strings, giving the description of each feature:

In[13]:

print("Feature names: \n{}".format(iris\_dataset['feature\_names']))

Out[13]:

Feature names:

['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',

'petal width (cm)']

The data itself is contained in the target and data fields. data contains the numeric measurements of sepal length, sepal width, petal length, and petal width in a NumPy array:

In[14]:

print("Type of data: {}".format(type(iris\_dataset['data'])))

Out[14]:

Type of data: <class 'numpy.ndarray'>

The rows in the data array correspond to flowers, while the columns represent the four measurements that were taken for each flower:

In[15]:

print("Shape of data: {}".format(iris\_dataset['data'].shape))

Out[15]:

Shape of data: (150, 4)

We see that the array contains measurements for 150 different flowers. Remember that the individual items are called samples in machine learning, and their properties are called features. The shape of the data array is the number of samples multiplied by the number of features. This is a convention in scikit-learn, and your data will always be assumed to be in this shape. Here are the feature values for the first five samples:

Vemos que el conjunto contiene medidas para 150 flores diferentes. Recuerde que los elementos individuales se denominan muestras en el aprendizaje automático y sus propiedades se denominan características. La forma de la matriz de datos es el número de muestras multiplicado por el número de características. Esta es una convención en scikit-learn y siempre se asumirá que sus datos tienen esta forma. Estos son los valores de las características de las primeras cinco muestras:

In[16]:

print("First five rows of data:\n{}".format(iris\_dataset['data'][:5]))

Out[16]:

First five rows of data:

[[ 5.1 3.5 1.4 0.2]

[ 4.9 3.

1.4 0.2]

[ 4.7 3.2 1.3 0.2]

[ 4.6 3.1 1.5 0.2]

[ 5.

3.6 1.4 0.2]]

From this data, we can see that all of the first five flowers have a petal width of 0.2 cm and that the first flower has the longest sepal, at 5.1 cm.

The target array contains the species of each of the flowers that were measured, also

as a NumPy array:

In[17]:

print("Type of target: {}".format(type(iris\_dataset['target'])))

Out[17]:

Type of target: <class 'numpy.ndarray'>

target is a one-dimensional array, with one entry per flower:

In[18]:

print("Shape of target: {}".format(iris\_dataset['target'].shape))

Out[18]:

Shape of target: (150,)

The species are encoded as integers from 0 to 2:

In[19]:

print("Target:\n{}".format(iris\_dataset['target']))

Out[19]:

Target:

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

2 2]

The meanings of the numbers are given by the iris['target\_names'] array: 0 means setosa, 1 means versicolor, and 2 means virginica.

**Measuring Success: Training and Testing Data**

We want to build a machine learning model from this data that can predict the species of iris for a new set of measurements. But before we can apply our model to new measurements, we need to know whether it actually works—that is, whether we should trust its predictions.

Queremos construir un modelo de aprendizaje automático a partir de estos datos que pueda predecir las especies de iris para un nuevo conjunto de mediciones. Pero antes de que podamos aplicar nuestro modelo a nuevas mediciones, necesitamos saber si realmente funciona, es decir, si debemos confiar en sus predicciones.

Unfortunately, we cannot use the data we used to build the model to evaluate it. This is because our model can always simply remember the whole training set, and will therefore always predict the correct label for any point in the training set. This “remembering” does not indicate to us whether our model will generalize well (in other words, whether it will also perform well on new data).

Lamentablemente, no podemos utilizar los datos que utilizamos para construir el modelo para evaluarlo. Esto se debe a que nuestro modelo siempre puede recordar simplemente todo el conjunto de entrenamiento y, por lo tanto, siempre predecirá la etiqueta correcta para cualquier punto del conjunto de entrenamiento. Este "recordar" no nos indica si nuestro modelo se generalizará bien (en otras palabras, si también funcionará bien con datos nuevos).

To assess the model’s performance, we show it new data (data that it hasn’t seen before) for which we have labels. This is usually done by splitting the labeled data we have collected (here, our 150 flower measurements) into two parts. One part of the data is used to build our machine learning model, and is called the training data or training set. The rest of the data will be used to assess how well the model works; this is called the test data, test set, or hold-out set.

Para evaluar el rendimiento del modelo, le mostramos datos nuevos (datos que no ha visto antes) para los cuales tenemos etiquetas. Esto generalmente se hace dividiendo los datos etiquetados que hemos recopilado (aquí, nuestras 150 medidas de flores) en dos partes. Una parte de los datos se utiliza para construir nuestro modelo de aprendizaje automático y se denomina datos de entrenamiento o conjunto de entrenamiento. El resto de los datos se utilizará para evaluar qué tan bien funciona el modelo; esto se denomina datos de prueba, conjunto de prueba o conjunto de reserva.

scikit-learn contains a function that shuffles the dataset and splits it for you: the train\_test\_split function. This function extracts 75% of the rows in the data as the training set, together with the corresponding labels for this data. The remaining 25% of the data, together with the remaining labels, is declared as the test set. Deciding how much data you want to put into the training and the test set respectively is somewhat arbitrary, but using a test set containing 25% of the data is a good rule of thumb.

scikit-learn contiene una función que mezcla el conjunto de datos y lo divide por usted: la función train\_test\_split. Esta función extrae el 75% de las filas de los datos como conjunto de entrenamiento, junto con las etiquetas correspondientes para estos datos. El 25% restante de los datos, junto con las etiquetas restantes, se declara como conjunto de prueba. Decidir cuántos datos desea incluir en el conjunto de entrenamiento y de prueba respectivamente es algo arbitrario, pero usar un conjunto de prueba que contenga el 25% de los datos es una buena regla general.

In scikit-learn, data is usually denoted with a capital X, while labels are denoted by a lowercase y. This is inspired by the standard formulation f(x)=y in mathematics, where x is the input to a function and y is the output. Following more conventions from mathematics, we use a capital X because the data is a two-dimensional array (a matrix) and a lowercase y because the target is a one-dimensional array (a vector).

En scikit-learn, los datos generalmente se indican con una **X** mayúscula, mientras que las etiquetas se indican con una **y** minúscula. Esto está inspirado en la formulación estándar f(x)=y en matemáticas, donde x es la entrada de una función e y es la salida. Siguiendo más convenciones de las matemáticas, usamos una X mayúscula porque los datos son una matriz bidimensional (una matriz) y una y minúscula porque el objetivo es una matriz unidimensional (un vector).

Let’s call train\_test\_split on our data and assign the outputs using this nomenclature:

In[20]:

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(**

**iris\_dataset['data'], iris\_dataset['target'], random\_state=0)**

Before making the split, the train\_test\_split function shuffles the dataset using a pseudorandom number generator. If we just took the last 25% of the data as a test set, all the data points would have the label 2, as the data points are sorted by the label (see the output for iris['target'] shown earlier). Using a test set containing only one of the three classes would not tell us much about how well our model generalizes, so we shuffle our data to make sure the test data contains data from all classes.

Antes de realizar la división, la función **train\_test\_split** mezcla el conjunto de datos utilizando un generador de números pseudoaleatorios. Si simplemente tomamos el último 25% de los datos como conjunto de prueba, todos los puntos de datos tendrían la etiqueta 2, ya que los puntos de datos están ordenados por la etiqueta (consulte el resultado de iris['target'] mostrado anteriormente). Usar un conjunto de prueba que contenga solo una de las tres clases no nos dirá mucho sobre qué tan bien se generaliza nuestro modelo, por lo que mezclamos nuestros datos para asegurarnos de que los datos de prueba contengan datos de todas las clases.

To make sure that we will get the same output if we run the same function several times, we provide the pseudorandom number generator with a fixed seed using the random\_state parameter. This will make the outcome deterministic, so this line will always have the same outcome. We will always fix the random\_state in this way when using randomized procedures in this book.

Para asegurarnos de que obtendremos el mismo resultado si ejecutamos la misma función varias veces, proporcionamos al generador de números pseudoaleatorios una semilla fija utilizando el parámetro random\_state. Esto hará que el resultado sea determinista, por lo que esta línea siempre tendrá el mismo resultado. Siempre arreglaremos el estado\_aleatorio de esta manera cuando utilicemos procedimientos aleatorios en este libro.

The output of the train\_test\_split function is X\_train, X\_test, y\_train, and y\_test, which are all NumPy arrays. X\_train contains 75% of the rows of the dataset, and X\_test contains the remaining 25%:

In[21]:

print("X\_train shape: {}".format(X\_train.shape))

print("y\_train shape: {}".format(y\_train.shape))

Out[21]:

X\_train shape: (112, 4)

y\_train shape: (112,)

In[22]:

print("X\_test shape: {}".format(X\_test.shape))

print("y\_test shape: {}".format(y\_test.shape))

Out[22]:

X\_test shape: (38, 4)

y\_test shape: (38,)

**First Things First: Look at Your Data**

Before building a machine learning model it is often a good idea to inspect the data, to see if the task is easily solvable without machine learning, or if the desired information might not be contained in the data.

Antes de construir un modelo de aprendizaje automático, suele ser una buena idea inspeccionar los datos para ver si la tarea se puede resolver fácilmente sin el aprendizaje automático o si la información deseada podría no estar contenida en los datos.

Additionally, inspecting your data is a good way to find abnormalities and peculiarities. Maybe some of your irises were measured using inches and not centimeters, for example. In the real world, inconsistencies in the data and unexpected measurements are very common.

Además, inspeccionar sus datos es una buena forma de encontrar anomalías y peculiaridades. Quizás algunos de tus iris se midieron en pulgadas y no en centímetros, por ejemplo. En el mundo real, las inconsistencias en los datos y las mediciones inesperadas son muy comunes.

One of the best ways to inspect data is to visualize it. One way to do this is by using a scatter plot. A scatter plot of the data puts one feature along the x-axis and another along the y-axis, and draws a dot for each data point. Unfortunately, computer screens have only two dimensions, which allows us to plot only two (or maybe three) features at a time. It is difficult to plot datasets with more than three features this way.

Una de las mejores formas de inspeccionar datos es visualizarlos. Una forma de hacerlo es mediante el uso de un diagrama de dispersión. Un diagrama de dispersión de los datos coloca una característica a lo largo del eje x y otra a lo largo del eje y, y dibuja un punto para cada punto de datos. Desafortunadamente, las pantallas de computadora tienen sólo dos dimensiones, lo que nos permite trazar sólo dos (o tal vez tres) características a la vez. Es difícil trazar conjuntos de datos con más de tres características de esta manera.

One way around this problem is to do a pair plot, which looks at all possible pairs of features. If you have a small number of features, such as the four we have here, this is quite reasonable. You should keep in mind, however, that a pair plot does not show the interaction of all of features at once, so some interesting aspects of the data may not be revealed when visualizing it this way.

Una forma de solucionar este problema es realizar un gráfico de pares, que analice todos los pares posibles de características. Si tiene una pequeña cantidad de funciones, como las cuatro que tenemos aquí, esto es bastante razonable. Sin embargo, debe tener en cuenta que un gráfico de pares no muestra la interacción de todas las características a la vez, por lo que es posible que algunos aspectos interesantes de los datos no se revelen al visualizarlos de esta manera.

Figure 1-3 is a pair plot of the features in the training set. The data points are colored according to the species the iris belongs to. To create the plot, we first convert the NumPy array into a pandas DataFrame. pandas has a function to create pair plots called scatter\_matrix. The diagonal of this matrix is filled with histograms of each feature:

La Figura 1-3 es un gráfico de pares de las características del conjunto de entrenamiento. Los puntos de datos están coloreados según la especie a la que pertenece el iris. Para crear el gráfico, primero convertimos la matriz NumPy en un DataFrame de pandas. pandas tiene una función para crear gráficos de pares llamada scatter\_matrix. La diagonal de esta matriz está llena de histogramas de cada característica:

In[23]:

# create dataframe from data in X\_train

# label the columns using the strings in iris\_dataset.feature\_names

iris\_dataframe = pd.DataFrame(X\_train, columns=iris\_dataset.feature\_names)

# create a scatter matrix from the dataframe, color by y\_train

pd.plotting.scatter\_matrix(iris\_dataframe, c=y\_train, figsize=(15, 15),

marker='o', hist\_kwds={'bins': 20}, s=60,

alpha=.8, cmap=mglearn.cm3)

Figure 1-3. Pair plot of the Iris dataset, colored by class label

From the plots, we can see that the three classes seem to be relatively well separated using the sepal and petal measurements. This means that a machine learning model will likely be able to learn to separate them.

**Building Your First Model: k-Nearest Neighbors**

Now we can start building the actual machine learning model. There are many classification algorithms in scikit-learn that we could use. Here we will use a k-nearest neighbors classifier, which is easy to understand. Building this model only consists of storing the training set. To make a prediction for a new data point, the algorithm finds the point in the training set that is closest to the new point. Then it assigns the label of this training point to the new data point.

Ahora podemos comenzar a construir el modelo de aprendizaje automático real. Hay muchos algoritmos de clasificación en scikit-learn que podríamos usar. Aquí usaremos un clasificador de k vecinos más cercanos, que es fácil de entender. Construir este modelo sólo consiste en almacenar el conjunto de entrenamiento. Para hacer una predicción para un nuevo punto de datos, el algoritmo encuentra el punto en el conjunto de entrenamiento más cercano al nuevo punto. Luego asigna la etiqueta de este punto de entrenamiento al nuevo punto de datos.

The k in k-nearest neighbors signifies that instead of using only the closest neighbor to the new data point, we can consider any fixed number k of neighbors in the training (for example, the closest three or five neighbors). Then, we can make a prediction using the majority class among these neighbors. We will go into more detail about this in Chapter 2; for now, we’ll use only a single neighbor.

La k en k-vecinos más cercanos significa que en lugar de usar solo el vecino más cercano al nuevo punto de datos, podemos considerar cualquier número fijo k de vecinos en el entrenamiento (por ejemplo, los tres o cinco vecinos más cercanos). Luego, podemos hacer una predicción usando la clase mayoritaria entre estos vecinos. Entraremos en más detalles sobre esto en el Capítulo 2; Por ahora, usaremos solo un vecino.

All machine learning models in scikit-learn are implemented in their own classes, which are called Estimator classes. The k-nearest neighbors classification algorithm is implemented in the KNeighborsClassifier class in the neighbors module. Before we can use the model, we need to instantiate the class into an object. This is when we will set any parameters of the model. The most important parameter of KNeighbor sClassifier is the number of neighbors, which we will set to 1:

Todos los modelos de aprendizaje automático en scikit-learn se implementan en sus propias clases, que se denominan clases de Estimador. El algoritmo de clasificación de k vecinos más cercanos se implementa en la clase KNeighborsClassifier en el módulo de vecinos. Antes de que podamos usar el modelo, necesitamos crear una instancia de la clase en un objeto. Aquí es cuando estableceremos los parámetros del modelo. El parámetro más importante de KNeighbor sClassifier es el número de vecinos, que estableceremos en 1:

In[24]:

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors=1)

The knn object encapsulates the algorithm that will be used to build the model from the training data, as well the algorithm to make predictions on new data points. It will also hold the information that the algorithm has extracted from the training data. In the case of KNeighborsClassifier, it will just store the training set.

El objeto knn encapsula el algoritmo que se utilizará para construir el modelo a partir de los datos de entrenamiento, así como el algoritmo para hacer predicciones sobre nuevos puntos de datos. También contendrá la información que el algoritmo ha extraído de los datos de entrenamiento. En el caso de KNeighborsClassifier, solo almacenará el conjunto de entrenamiento.

To build the model on the training set, we call the fit method of the knn object, which takes as arguments the NumPy array X\_train containing the training data and the NumPy array y\_train of the corresponding training labels:

Para construir el modelo en el conjunto de entrenamiento, llamamos al método de ajuste del objeto knn, que toma como argumentos la matriz NumPy X\_train que contiene los datos de entrenamiento y la matriz NumPy y\_train de las etiquetas de entrenamiento correspondientes:

In[25]:

knn.fit(X\_train, y\_train)

Out[25]:

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=1, n\_neighbors=1, p=2,

weights='uniform')

The fit method returns the knn object itself (and modifies it in place), so we get a string representation of our classifier. The representation shows us which parameters were used in creating the model. Nearly all of them are the default values, but you can also find n\_neighbors=1, which is the parameter that we passed. Most models in scikit-learn have many parameters, but the majority of them are either speed optimizations or for very special use cases. You don’t have to worry about the other parameters shown in this representation. Printing a scikit-learn model can yield very long strings, but don’t be intimidated by these. We will cover all the important parameters in Chapter 2. In the remainder of this book, we will not show the output of fit because it doesn’t contain any new information.

El método fit devuelve el objeto knn en sí (y lo modifica en su lugar), por lo que obtenemos una representación de cadena de nuestro clasificador. La representación nos muestra qué parámetros se utilizaron en la creación del modelo. Casi todos ellos son los valores predeterminados, pero también puedes encontrar n\_neighbors=1, que es el parámetro que pasamos. La mayoría de los modelos en scikit-learn tienen muchos parámetros, pero la mayoría de ellos son optimizaciones de velocidad o para casos de uso muy especiales. No tiene que preocuparse por los demás parámetros que se muestran en esta representación. Imprimir un modelo scikit-learn puede generar cadenas muy largas, pero no se deje intimidar por ellas. Cubriremos todos los parámetros importantes en el Capítulo 2. En el resto de este libro, no mostraremos el resultado del ajuste porque no contiene ninguna información nueva.

**Making Predictions**

We can now make predictions using this model on new data for which we might not know the correct labels. Imagine we found an iris in the wild with a sepal length of 5 cm, a sepal width of 2.9 cm, a petal length of 1 cm, and a petal width of 0.2 cm. What species of iris would this be? We can put this data into a NumPy array, again by calculating the shape—that is, the number of samples (1) multiplied by the number of features (4):

Ahora podemos hacer predicciones utilizando este modelo sobre datos nuevos para los cuales es posible que no conozcamos las etiquetas correctas. Imaginemos que encontramos un iris en la naturaleza con una longitud de sépalo de 5 cm, un ancho de sépalo de 2,9 cm, una longitud de pétalo de 1 cm y un ancho de pétalo de 0,2 cm. ¿Qué especie de iris sería esta? Podemos poner estos datos en una matriz NumPy, nuevamente calculando la forma, es decir, el número de muestras (1) multiplicado por el número de características (4):

In[26]:

X\_new = np.array([[5, 2.9, 1, 0.2]])

print("X\_new.shape: {}".format(X\_new.shape))

Out[26]:

X\_new.shape: (1, 4)

Note that we made the measurements of this single flower into a row in a two- dimensional NumPy array, as scikit-learn always expects two-dimensional arrays for the data.

To make a prediction, we call the predict method of the knn object:

In[27]:

prediction = knn.predict(X\_new)

print("Prediction: {}".format(prediction))

print("Predicted target name: {}".format(

iris\_dataset['target\_names'][prediction]))

Out[27]:

Prediction: [0]

Predicted target name: ['setosa']

Our model predicts that this new iris belongs to the class 0, meaning its species is setosa. But how do we know whether we can trust our model? We don’t know the correct species of this sample, which is the whole point of building the model!

Nuestro modelo predice que este nuevo iris pertenece a la clase 0, lo que significa que su especie es setosa. Pero ¿cómo sabemos si podemos confiar en nuestro modelo? ¡No conocemos la especie correcta de esta muestra, que es el objetivo de construir el modelo!

**Evaluating the Model**

This is where the test set that we created earlier comes in. This data was not used to build the model, but we do know what the correct species is for each iris in the test set.

Aquí es donde entra en juego el conjunto de prueba que creamos anteriormente. Estos datos no se utilizaron para construir el modelo, pero sí sabemos cuál es la especie correcta para cada iris en el conjunto de prueba.

Therefore, we can make a prediction for each iris in the test data and compare it against its label (the known species). We can measure how well the model works by computing the accuracy, which is the fraction of flowers for which the right species was predicted:

Por lo tanto, podemos hacer una predicción para cada iris en los datos de prueba y compararla con su etiqueta (la especie conocida). Podemos medir qué tan bien funciona el modelo calculando la precisión, que es la fracción de flores para las cuales se predijo la especie correcta:

In[28]:

y\_pred = knn.predict(X\_test)

print("Test set predictions:\n {}".format(y\_pred))

Out[28]:

Test set predictions:

[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0 2]

In[29]:

print("Test set score: {:.2f}".format(np.mean(y\_pred == y\_test)))

Out[29]:

Test set score: 0.97

We can also use the score method of the knn object, which will compute the test set accuracy for us:

In[30]:

print("Test set score: {:.2f}".format(knn.score(X\_test, y\_test)))

Out[30]:

Test set score: 0.97

For this model, the test set accuracy is about 0.97, which means we made the right prediction for 97% of the irises in the test set. Under some mathematical assumptions, this means that we can expect our model to be correct 97% of the time for new irises. For our hobby botanist application, this high level of accuracy means that our model may be trustworthy enough to use. In later chapters we will discuss how we can improve performance, and what caveats there are in tuning a model.

Para este modelo, la precisión del conjunto de prueba es de aproximadamente 0,97, lo que significa que hicimos la predicción correcta para el 97% de los iris en el conjunto de prueba. Según algunos supuestos matemáticos, esto significa que podemos esperar que nuestro modelo sea correcto el 97% de las veces para iris nuevos. Para nuestra aplicación de botánico aficionado, este alto nivel de precisión significa que nuestro modelo puede ser lo suficientemente confiable como para usarlo. En capítulos posteriores discutiremos cómo podemos mejorar el rendimiento y qué advertencias existen al ajustar un modelo.

**Summary and Outlook**

Let’s summarize what we learned in this chapter. We started with a brief introduction to machine learning and its applications, then discussed the distinction between supervised and unsupervised learning and gave an overview of the tools we’ll be using in this book. Then, we formulated the task of predicting which species of iris a particular flower belongs to by using physical measurements of the flower. We used a dataset of measurements that was annotated by an expert with the correct species to build our model, making this a supervised learning task. There were three possible species, setosa, versicolor, or virginica, which made the task a three-class classification problem. The possible species are called classes in the classification problem, and the species of a single iris is called its label.

Resumamos lo que aprendimos en este capítulo. Comenzamos con una breve introducción al aprendizaje automático y sus aplicaciones, luego discutimos la distinción entre aprendizaje supervisado y no supervisado y brindamos una descripción general de las herramientas que usaremos en este libro. Luego, formulamos la tarea de predecir a qué especie de iris pertenece una flor en particular utilizando medidas físicas de la flor. Utilizamos un conjunto de datos de mediciones anotadas por un experto con las especies correctas para construir nuestro modelo, lo que la convierte en una tarea de aprendizaje supervisada. Había tres especies posibles, setosa, versicolor o virginica, lo que convertía la tarea en un problema de clasificación de tres clases. Las posibles especies se denominan clases en el problema de clasificación, y la especie de un solo iris se denomina etiqueta.

The Iris dataset consists of two NumPy arrays: one containing the data, which is referred to as X in scikit-learn, and one containing the correct or desired outputs, which is called y. The array X is a two-dimensional array of features, with one row per data point and one column per feature. The array y is a one-dimensional array, which here contains one class label, an integer ranging from 0 to 2, for each of the samples.

El conjunto de datos de Iris consta de dos matrices NumPy: una que contiene los datos, a la que se hace referencia como X en scikit-learn, y otra que contiene las salidas correctas o deseadas, que se llama y. La matriz X es una matriz bidimensional de características, con una fila por punto de datos y una columna por característica. La matriz y es una matriz unidimensional, que aquí contiene una etiqueta de clase, un número entero que va de 0 a 2, para cada una de las muestras.

We split our dataset into a training set, to build our model, and a test set, to evaluate how well our model will generalize to new, previously unseen data. We chose the k-nearest neighbors classification algorithm, which makes predictions for a new data point by considering its closest neighbor(s) in the training set. This is implemented in the KNeighborsClassifier class, which contains the algorithm that builds the model as well as the algorithm that makes a prediction using the model.

Dividimos nuestro conjunto de datos en un conjunto de entrenamiento para construir nuestro modelo y un conjunto de prueba para evaluar qué tan bien se generalizará nuestro modelo a datos nuevos nunca antes vistos. Elegimos el algoritmo de clasificación de k vecinos más cercanos, que hace predicciones para un nuevo punto de datos considerando sus vecinos más cercanos en el conjunto de entrenamiento. Esto se implementa en la clase KNeighborsClassifier, que contiene el algoritmo que construye el modelo, así como el algoritmo que realiza una predicción utilizando el modelo.

We instantiated the class, setting parameters. Then we built the model by calling the fit method, passing the training data (X\_train) and training outputs (y\_train) as parameters. We evaluated the model using the score method, which computes the accuracy of the model. We applied the score method to the test set data and the test set labels and found that our model is about 97% accurate, meaning it is correct 97% of the time on the test set.

Creamos una instancia de la clase, configurando parámetros. Luego construimos el modelo llamando al método de ajuste, pasando los datos de entrenamiento (X\_train) y las salidas de entrenamiento (y\_train) como parámetros. Evaluamos el modelo utilizando el método de puntuación, que calcula la precisión del modelo. Aplicamos el método de puntuación a los datos del conjunto de prueba y a las etiquetas del conjunto de prueba y descubrimos que nuestro modelo tiene aproximadamente un 97% de precisión, lo que significa que es correcto el 97% de las veces en el conjunto de prueba.

This gave us the confidence to apply the model to new data (in our example, new flower measurements) and trust that the model will be correct about 97% of the time. Here is a summary of the code needed for the whole training and evaluation procedure:

Esto nos dio la confianza para aplicar el modelo a nuevos datos (en nuestro ejemplo, nuevas medidas de flores) y confiar en que el modelo será correcto aproximadamente el 97% de las veces. Aquí hay un resumen del código necesario para todo el procedimiento de capacitación y evaluación:

In[31]:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

iris\_dataset['data'], iris\_dataset['target'], random\_state=0)

knn = KNeighborsClassifier(n\_neighbors=1)

knn.fit(X\_train, y\_train)

print("Test set score: {:.2f}".format(knn.score(X\_test, y\_test)))

Out[31]:

Test set score: 0.97

This snippet contains the core code for applying any machine learning algorithm using scikit-learn. The fit, predict, and score methods are the common interface to supervised models in scikit-learn, and with the concepts introduced in this chapter, you can apply these models to many machine learning tasks. In the next chapter, we will go into more depth about the different kinds of supervised models in scikit-learn and how to apply them successfully.

Este fragmento contiene el código central para aplicar cualquier algoritmo de aprendizaje automático utilizando scikit-learn. Los métodos de ajuste, predicción y puntuación son la interfaz común para los modelos supervisados ​​en scikit-learn y, con los conceptos introducidos en este capítulo, puede aplicar estos modelos a muchas tareas de aprendizaje automático. En el próximo capítulo, profundizaremos en los diferentes tipos de modelos supervisados ​​en scikit-learn y cómo aplicarlos con éxito.