

SEGUNDA
EDICIÓN
REVISADA Y
ACTUALIZADA

MACHINE LEARNING

Sebastian Raschka · Vahid Mirjalili

Python Machine Learning

Aprendizaje automático y aprendizaje profundo
con Python, scikit-learn y TensorFlow

Marcombo

Aprendizaje automático con Python

Aprendizaje automático y aprendizaje profundo
con Python, scikit-learn y TensorFlow

Segunda edición

Acceda a www.marcombo.info

para descargar gratis

códigos de ejemplo

complemento imprescindible de este libro

Código:

PYTHON3

Aprendizaje automático con Python

Aprendizaje automático y aprendizaje profundo
con Python, scikit-learn y TensorFlow

**Sebastian Raschka
Vahid Mirjalili**



Segunda edición original publicada en inglés por Packt Publishing Ltd. con el título: *Python Machine Learning*, © 2017 Sebastian Raschka y Vahid Mirjalili

Título de la edición en español: *Aprendizaje automático con Python*

Segunda edición en español, año 2019

© 2019 MARCOMBO, S.A.

www.marcombo.com

Traducción: Sònia Llena

Revisor técnico: Ferran Fàbregas

Correctora: Anna Alberola

Directora de producción: M.^a Rosa Castillo

«Cualquier forma de reproducción, distribución, comunicación pública o transformación de esta obra solo puede ser realizada con la autorización de sus titulares, salvo excepción prevista por la ley. Diríjase a CEDRO (Centro Español de Derechos Reprográficos, www.cedro.org) si necesita fotocopiar o escanear algún fragmento de esta obra».

ISBN: 978-84-267-2720-6

D.L.: B-27539-2018

Impreso en Servicepoint

Printed in Spain

1

Dar a los ordenadores el poder de aprender de los datos

En mi opinión, el **aprendizaje automático** –la aplicación y ciencia de los algoritmos que da sentido a los datos– es el campo más apasionante de todas las ciencias computacionales. Vivimos en una época en la cual los datos llegan en abundancia; utilizando algoritmos de autoaprendizaje del campo del aprendizaje automático podemos convertir estos datos en conocimiento. Gracias a las múltiples y potentes librerías de código abierto que han sido desarrolladas en los últimos años, probablemente no ha habido un momento mejor para acceder al campo del aprendizaje automático y aprender cómo utilizar potentes algoritmos para detectar patrones de datos y hacer predicciones sobre acontecimientos futuros.

En este capítulo, aprenderás los principales conceptos y los diferentes tipos de aprendizaje automático. Junto con una introducción básica a la terminología más importante, sentaremos las bases para utilizar con éxito técnicas de aprendizaje automático para la resolución práctica de problemas.

En este capítulo, trataremos los siguientes temas:

- Los conceptos generales del aprendizaje automático.
- Los tres tipos de aprendizaje y la terminología básica.
- La construcción de bloques para diseñar sistemas de aprendizaje automático.
- La instalación y configuración de Python para el análisis de datos y el aprendizaje automático.

Crear máquinas inteligentes para transformar datos en conocimiento

En esta época de tecnología moderna, existe un recurso que tenemos en abundancia: gran cantidad de datos estructurados y no estructurados. En la segunda mitad del siglo veinte, el aprendizaje automático evolucionó como un subcampo de la **Inteligencia Artificial (AI)** que involucraba algoritmos de autoaprendizaje que derivaban el conocimiento a partir de datos para crear predicciones. En lugar de necesitar al hombre para derivar de forma manual las reglas y crear modelos a partir del análisis de grandes cantidades de datos, el aprendizaje automático ofrece una alternativa más eficiente para capturar el conocimiento en datos, mejorar gradualmente el rendimiento de los modelos predictivos y tomar decisiones basadas en esos datos. El aprendizaje automático no solo es cada vez más importante en la investigación de ciencia computacional, sino que juega un papel cada vez más importante en nuestra vida diaria. Gracias al aprendizaje automático, disfrutamos de filtros potentes para el correo no deseado, *software* práctico de reconocimiento de voz y texto, motores de búsqueda fiables, desafiantes programas para jugar al ajedrez y, esperemos que muy pronto, eficientes coches de conducción autónoma.

Los tres tipos de aprendizaje automático

En esta sección, echaremos un vistazo a los tres tipos de aprendizaje automático: **aprendizaje supervisado**, **aprendizaje no supervisado** y **aprendizaje reforzado**. Vamos a aprender las diferencias fundamentales entre los tres tipos distintos de aprendizaje y, mediante ejemplos conceptuales, desarrollaremos una intuición para los ámbitos de problemas prácticos donde pueden ser aplicados:

Aprendizaje supervisado

- Datos etiquetados
- Feedback directo
- Predicción de resultados/futuro

Aprendizaje no supervisado

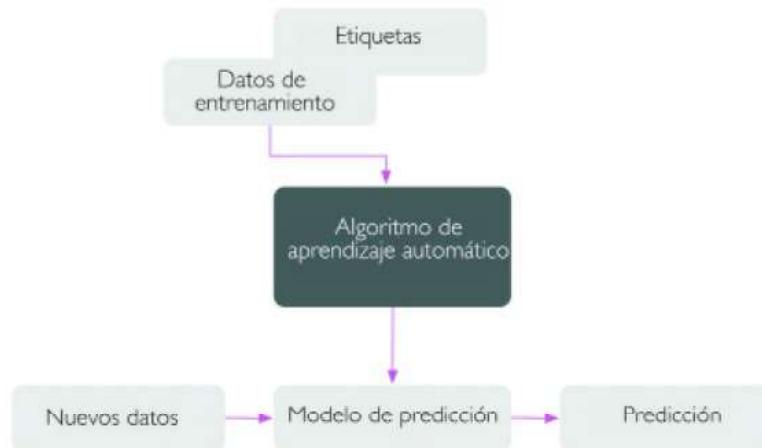
- Sin etiquetas
- Sin feedback
- Encontrar estructuras ocultas en los datos

Aprendizaje reforzado

- Proceso de decisión
- Sistema de recompensa
- Aprender series de acciones

Hacer predicciones sobre el futuro con el aprendizaje supervisado

El objetivo principal del aprendizaje supervisado es aprender un modelo, a partir de datos de entrenamiento etiquetados, que nos permite hacer predicciones sobre datos futuros o no vistos. Aquí, el término **supervisado** se refiere a un conjunto de muestras donde las señales de salida deseadas (etiquetas) ya se conocen.



Considerando el ejemplo del filtro de correo no deseado, podemos entrenar un modelo utilizando un algoritmo de aprendizaje automático supervisado en un cuerpo de correos electrónicos etiquetados –correos que están correctamente marcados como «correo no deseado» o como «no correo no deseado»– para predecir si un nuevo correo electrónico pertenece a una u otra categoría. Una tarea de aprendizaje supervisado con etiquetas de clase discretas, como en el ejemplo anterior del filtro de correo no deseado, también se conoce como **tarea de clasificación**. Otra subcategoría del aprendizaje supervisado es la **regresión**, donde la señal resultante es un valor continuo.

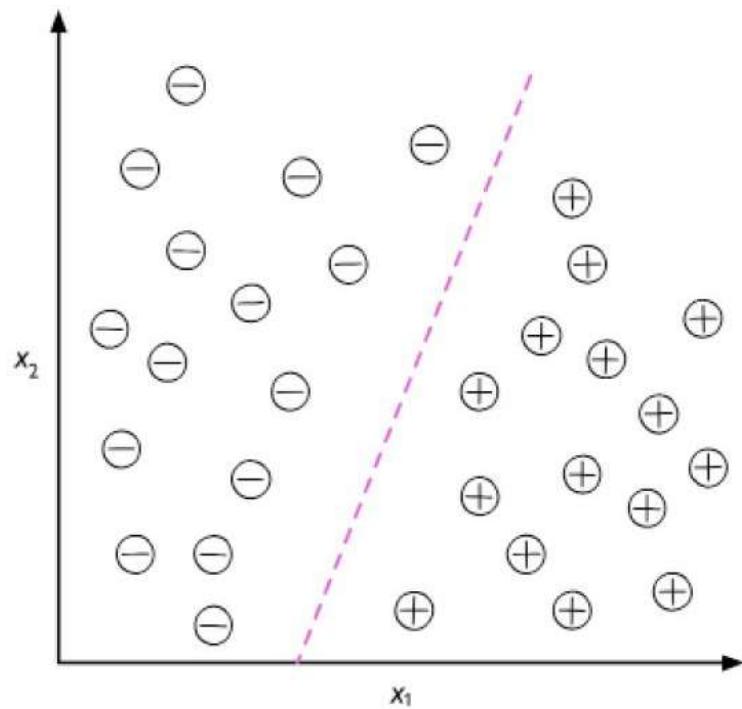
Clasificación para predecir etiquetas de clase

La clasificación es una subcategoría del aprendizaje supervisado cuyo objetivo es predecir las etiquetas de clase categórica de nuevas instancias, basadas en observaciones pasadas. Estas etiquetas de

clase son discretas, valores desordenados que se pueden entender como membresías grupales de las instancias. El ejemplo que hemos mencionado anteriormente de la detección de correo no deseado representa un típico ejemplo de una tarea de clasificación binaria, donde el algoritmo de aprendizaje automático aprende un conjunto de reglas para distinguir entre dos posibles clases: mensajes que son o no son correo no deseado.

Sin embargo, el conjunto de etiquetas de clase no tiene que ser de naturaleza binaria. El modelo predictivo aprendido mediante un algoritmo de aprendizaje supervisado puede asignar cualquier etiqueta de clase que se presente en el conjunto de datos de entrenamiento a una nueva instancia sin etiqueta. Un ejemplo típico de una tarea de **clasificación multiclas** es el reconocimiento de un carácter manuscrito. Aquí, podemos recoger un conjunto de datos de entrenamiento que consiste en múltiples ejemplos manuscritos de cada letra del alfabeto. Ahora, si un usuario proporciona un nuevo carácter manuscrito desde un dispositivo de entrada, nuestro modelo predictivo será capaz de predecir la letra correcta del alfabeto con cierta precisión. Sin embargo, nuestro sistema de aprendizaje automático no sería capaz de reconocer de forma correcta ningún dígito del cero al nueve, por ejemplo, si no formaran parte de nuestro conjunto de datos de entrenamiento.

La siguiente figura ilustra el concepto de una tarea de clasificación binaria que da 30 muestras de entrenamiento; 15 de estas muestras están etiquetadas como clase negativa (signo menos) y otras 15 como clase positiva (signo más). En este caso, nuestro conjunto de datos es bidimensional, lo que significa que cada muestra tiene dos valores asociados: x_1 y x_2 . Ahora, podemos utilizar un algoritmo de aprendizaje automático supervisado para aprender una regla -el límite de decisión está representado con una línea discontinua- que puede separar las dos clases y clasificar nuevos datos dentro de cada categoría dados sus valores de x_1 y x_2 :



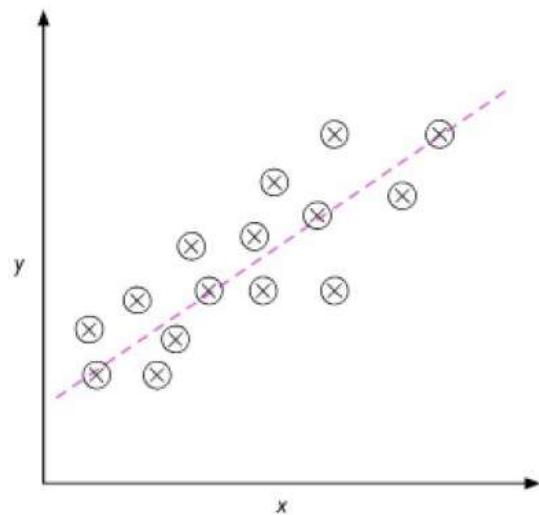
Regresión para predecir resultados continuos

En la sección anterior hemos aprendido que la tarea de clasificación consiste en asignar etiquetas categóricas y sin orden a instancias. Un segundo tipo de aprendizaje supervisado es la predicción de resultados continuos, también conocida como **análisis de regresión**. En el análisis de regresión, tenemos un número de variables predictoras (**explicativas**) y una variable de respuesta continua (**resultado o destino**), y tenemos que encontrar una relación entre estas variables que nos permita predecir un resultado.

Por ejemplo, supongamos que queremos predecir los resultados del examen de selectividad de matemáticas de nuestros alumnos. Si existe una relación entre el tiempo que han pasado estudiando para la prueba y los resultados finales, podríamos utilizarla como dato de entrenamiento para aprender un modelo que utilice el tiempo de estudio para predecir los resultados de la prueba de futuros estudiantes que deseen pasar este examen.

[ El término *regresión* fue ideado por Francis Galton en su artículo *Regression towards Mediocrity in Hereditary Stature* [Regresión hacia la mediocridad en estatura hereditaria] en 1886. Galton describió el fenómeno biológico según el cual la variación de altura en una población no aumenta con el tiempo. Él observó que la altura de los padres no pasa a los hijos, pero que, en cambio, la altura de los hijos está retrocediendo hacia la media de la población.]

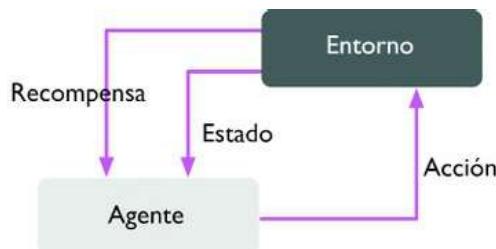
La siguiente figura ilustra el concepto de regresión lineal. Dada una variable predictora x y una variable de respuesta y , aplicamos una línea fina a este dato, que minimiza la distancia –normalmente, la distancia cuadrada de promedio– entre los puntos de muestra y la línea aplicada. Ahora podemos utilizar la intersección y la pendiente aprendidas de este dato para predecir la variable de resultado del nuevo dato:



Resolver problemas interactivos con aprendizaje reforzado

Otro tipo de aprendizaje automático es el **aprendizaje reforzado**. En este tipo de aprendizaje, el objetivo es desarrollar un sistema (**agente**) que mejore su rendimiento basado en interacciones con el entorno. Como la información sobre el estado actual del entorno normalmente también incluye una **señal de recompensa**, podemos pensar en el aprendizaje reforzado como un campo relacionado con el aprendizaje supervisado. Sin embargo, en el aprendizaje reforzado este *feedback* no es el valor o la etiqueta correctos sobre el terreno, sino una medida de cómo ha sido medida la acción por parte de una función de recompensa. A través de su interacción con el entorno, un agente puede utilizar el aprendizaje reforzado para aprender una serie de acciones que maximicen esta recompensa mediante un enfoque experimental de ensayo-error o una planificación deliberativa.

Un conocido ejemplo de aprendizaje reforzado es un motor de ajedrez. Aquí, el agente elige entre una serie de movimientos según el estado del tablero (el entorno), y la recompensa se puede definir como «ganas» o «pierdes» al final del juego:



Existen diferentes subtipos de aprendizaje reforzado. Sin embargo, un esquema general es que el agente en aprendizaje reforzado intenta maximizar la recompensa mediante una serie de interacciones con el entorno. Cada estado puede estar asociado a una recompensa positiva o negativa, y una recompensa se puede definir como el logro de un objetivo general (como ganar o perder una partida de ajedrez). Por ejemplo, en ajedrez, el resultado de cada movimiento podría ser un estado distinto del entorno. Para explorar un poco más el ejemplo del ajedrez, pensemos en ciertas

jugadas del tablero asociadas a un evento positivo (por ejemplo, eliminar una pieza del contrincante o amenazar a la reina). Sin embargo, otras jugadas están asociadas a un evento negativo (como perder una pieza para el contrincante en el siguiente turno). Ahora, no todos los turnos dan como resultado la eliminación de una pieza del tablero, y el aprendizaje reforzado se centra en aprender las series de pasos maximizando una recompensa basada en el *feedback* inmediato y diferido.

Aunque esta sección ofrece una visión básica del aprendizaje reforzado, ten en cuenta que las aplicaciones de este tipo de aprendizaje están fuera del alcance de este libro, que prioriza la clasificación, el análisis de regresión y el agrupamiento.

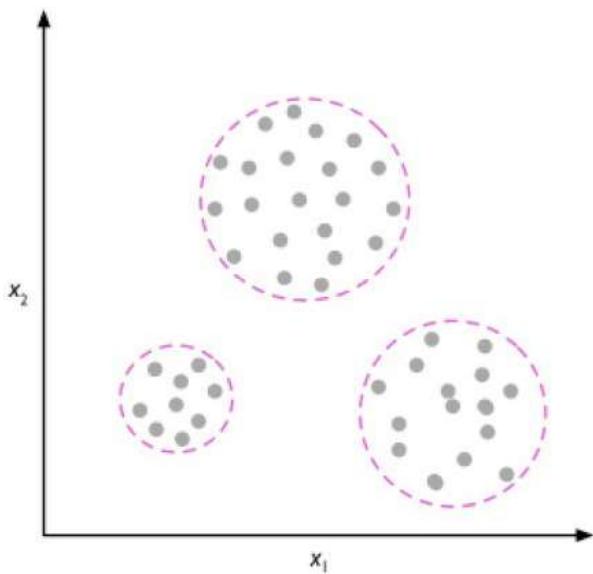
Descubrir estructuras ocultas con el aprendizaje sin supervisión

En el aprendizaje supervisado, cuando entrenamos nuestro modelo sabemos la respuesta correcta de antemano, y en el reforzado definimos una medida de recompensa para acciones concretas mediante el agente. Sin embargo, en el aprendizaje sin supervisión tratamos con datos sin etiquetar o datos de estructura desconocida. Con las técnicas de aprendizaje sin supervisión, podemos explorar la estructura de nuestros datos para extraer información significativa sin la ayuda de una variable de resultado conocida o una función de recompensa.

Encontrar subgrupos con el agrupamiento

El **agrupamiento** es una técnica exploratoria de análisis de datos que nos permite organizar un montón de información en subgrupos significativos (*clústers*) sin tener ningún conocimiento previo de los miembros del grupo. Cada *clúster* que surge durante el análisis define un grupo de objetos que comparten un cierto grado de semejanza pero difieren de los objetos de otros *clústers*, razón por la cual el agrupamiento también se denomina a veces **clasificación sin supervisión**. El agrupamiento es una excelente técnica para estructurar información y derivar relaciones significativas de los datos. Por ejemplo, permite a los vendedores descubrir grupos de clientes basados en sus intereses, con el fin de desarrollar programas de *marketing* exclusivos.

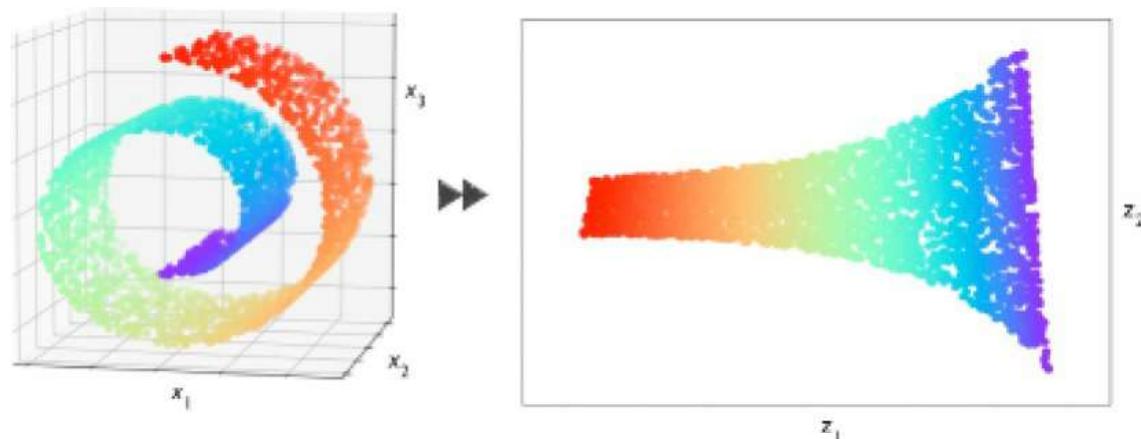
La siguiente figura muestra cómo se puede aplicar el agrupamiento para organizar datos sin etiquetar en tres grupos distintos, basados en la similitud de sus características x_1 y x_2 :



Reducción de dimensionalidad para comprimir datos

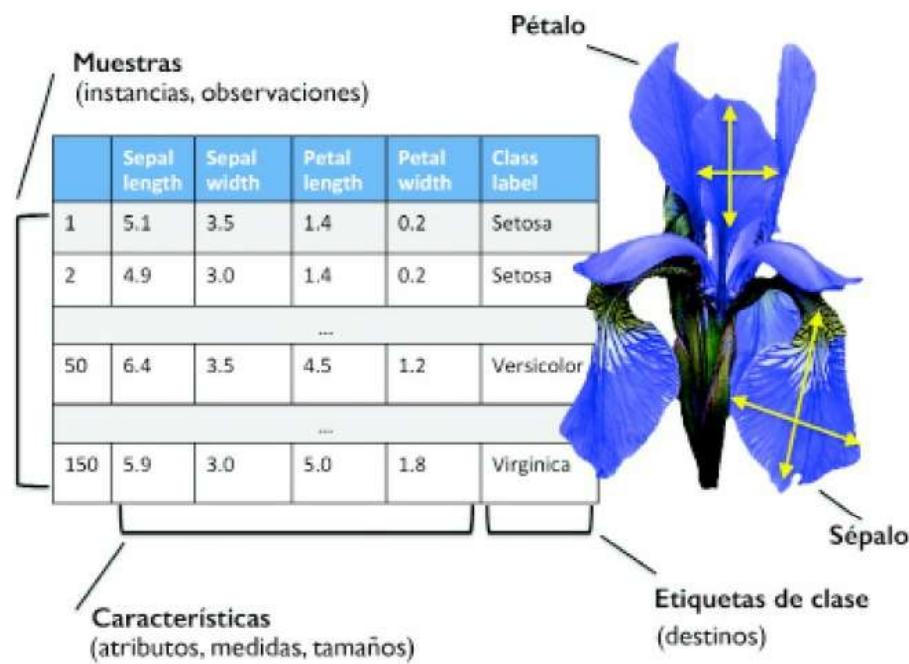
Otro subcampo del aprendizaje sin supervisión es la **reducción de dimensionalidad**. Muchas veces trabajamos con datos de alta dimensionalidad (cada observación muestra un elevado número de medidas), lo cual puede suponer un reto para el espacio de almacenamiento limitado y el rendimiento computacional de los algoritmos del aprendizaje automático. La reducción de dimensionalidad sin supervisión es un enfoque utilizado con frecuencia en el preprocesamiento de características para eliminar ruido de los datos; también puede degradar el rendimiento predictivo de ciertos algoritmos y comprimir los datos en un subespacio dimensional más pequeño, manteniendo la mayor parte de la información importante.

A veces, la reducción de dimensionalidad también puede ser útil para visualizar datos; por ejemplo, un conjunto de características dimensionales pueden ser proyectadas en un espacio de características de una, dos o tres dimensiones para visualizarlas mediante gráficos de dispersión o histogramas 2D o 3D. Las siguientes figuras muestran un ejemplo donde la reducción de dimensionalidad no lineal se ha aplicado para comprimir un brazo de gitano tridimensional en un subespacio con características 2D:



Introducción a la terminología básica y las notaciones

Ahora que ya hemos tratado las tres categorías principales de aprendizaje automático –supervisado, sin supervisión y reforzado–, vamos a echar un vistazo a la terminología básica que utilizaremos en este libro. La tabla siguiente muestra un extracto del conjunto de datos Iris, un ejemplo clásico en el campo del aprendizaje automático. El conjunto de datos Iris contiene las medidas de 150 flores iris de tres especies distintas: Setosa, Versicolor y Virginica. Cada muestra de flor representa una fila de nuestro conjunto de datos y las medidas de la flor en centímetros se almacenan en columnas, que también denominamos **características** del conjunto de datos:



Para que la notación sea simple a la vez que eficiente, utilizaremos algunos de los términos básicos de álgebra lineal. En los siguientes capítulos, utilizaremos una matriz y una notación vectorial para referirnos a nuestros datos. Seguiremos la convención común para representar cada muestra como una fila independiente en una matriz de características \mathbf{X} , donde cada característica se almacena en una columna independiente.

Así, el conjunto de datos Iris que contiene 150 muestras y cuatro características también se puede escribir como una matriz 150×4

$$\mathbf{X} \in \mathbb{R}^{150 \times 4} :$$

$$\begin{bmatrix}x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)}\end{bmatrix}$$

Para el resto del libro, si no se indica de otro modo, utilizaremos el superíndice i para indicar la muestra de entrenamiento i , y el subíndice j para indicar la dimensión j del conjunto de datos de entrenamiento.

Utilizamos letras en negrita y minúsculas para referirnos a vectores ($x \in \mathbb{R}^{n \times 1}$) y letras en negrita y mayúsculas para hablar de matrices ($X \in \mathbb{R}^{n \times m}$). Para referirnos a elementos individuales en un vector o matriz, escribimos las letras en cursiva ($x^{(n)}$ o $x_{(m)}^{(n)}$, respectivamente).

Por ejemplo, x_1^{150} se refiere a la primera dimensión de las 150 muestras de flores, *largo de sépalo*. Así, cada fila de la matriz de características representa una instancia de flor y puede ser escrita como un vector de fila de cuatro dimensiones $x^{(i)} \in \mathbb{R}^{1 \times 4}$:



$$x^{(i)} = \begin{bmatrix} x_1^{(i)} & x_2^{(i)} & x_3^{(i)} & x_4^{(i)} \end{bmatrix}$$

Y cada dimensión de características es un vector de columna de 150 dimensiones $x_j \in \mathbb{R}^{150 \times 1}$. Por ejemplo:

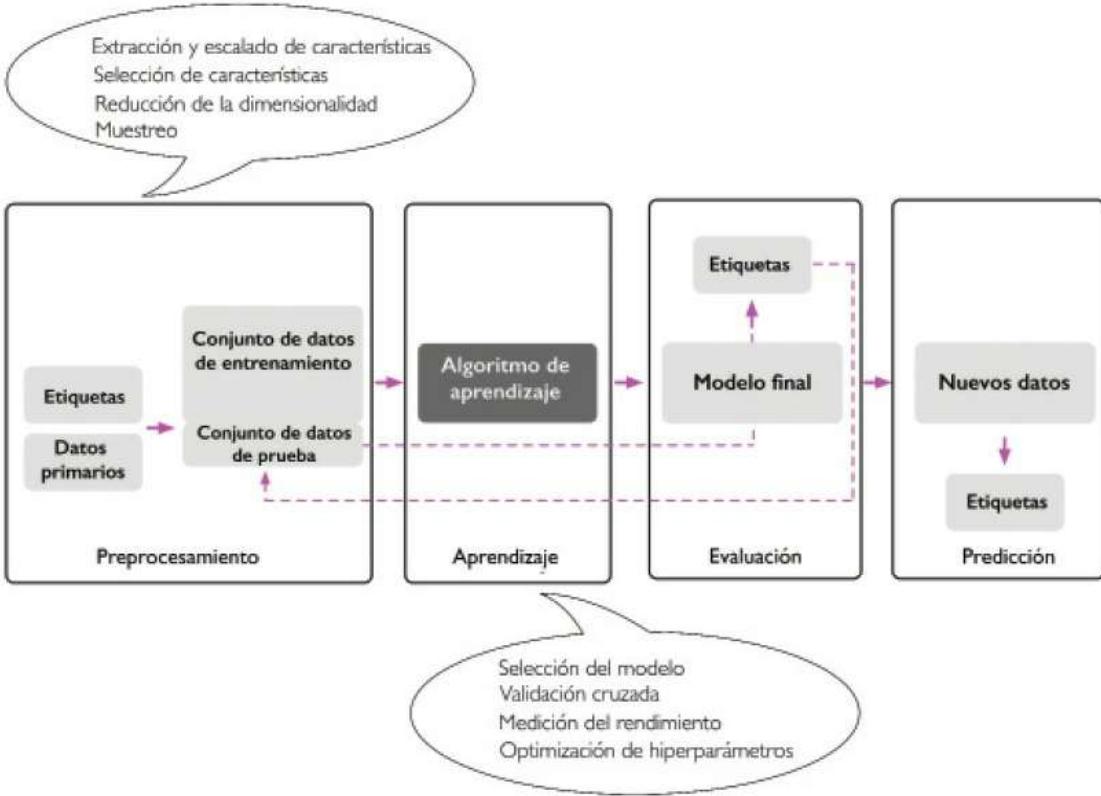
$$x_j = \begin{bmatrix} x_j^{(1)} \\ x_j^{(2)} \\ \vdots \\ x_j^{(150)} \end{bmatrix}$$

De forma similar, almacenamos las variables de destino (aquí, etiquetas de clase) como un vector de columna de 150 dimensiones:

$$y = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(150)} \end{bmatrix} (y \in \{\text{Setosa, Versicolor, Virginica}\})$$

Una hoja de ruta para crear sistemas de aprendizaje automático

En secciones anteriores, hemos hablado de los conceptos básicos del aprendizaje automático y de los tres tipos distintos de aprendizaje. En esta sección, hablaremos de las otras partes importantes del sistema de aprendizaje automático que acompañan al algoritmo de aprendizaje. El siguiente diagrama muestra un flujo de trabajo típico para el uso del aprendizaje automático en modelado predictivo, que trataremos en las siguientes subsecciones:



Preprocesamiento: Dar forma a los datos

Vamos a empezar hablando de la hoja de ruta para crear sistemas de aprendizaje automático. No es habitual que los datos primarios se presenten en la forma necesaria para un rendimiento óptimo del algoritmo de aprendizaje. Así, el preprocesamiento de los datos es uno de los pasos más importantes en cualquier aplicación de aprendizaje automático. Si tomamos como ejemplo el conjunto de datos de flores Iris de la sección anterior, podemos pensar en los datos primarios como una serie de imágenes de flores de las cuales queremos extraer características significativas. Estas características útiles pueden ser el color, el tono, la intensidad de las flores, o la altura, la longitud y anchura de la flor. Hay algoritmos de aprendizaje automático que, además, necesitan que las características seleccionadas tengan el mismo tamaño para conseguir un rendimiento óptimo, el cual normalmente se consigue transformando las características en el rango [0, 1] o con una distribución normal estándar con media cero y variación unitaria, como veremos más adelante.

Algunas de las características seleccionadas pueden estar altamente relacionadas y, por tanto, pueden ser redundantes hasta un cierto punto. En estos casos, las técnicas de reducción de dimensionalidad son muy útiles para comprimir las características en un subespacio dimensional más pequeño. Reducir la dimensionalidad de nuestro espacio de características tiene la ventaja de que se requiere menos espacio de almacenamiento y el algoritmo de aprendizaje funciona mucho más rápido. En algunos casos, la reducción de dimensionalidad también puede mejorar el rendimiento predictivo de un modelo si el conjunto de datos contiene un gran número de características irrelevantes (o ruido), es decir, si el conjunto de datos tiene una relación baja entre señal y ruido.

Para determinar si nuestro algoritmo de aprendizaje automático no solo funciona bien en el conjunto de entrenamiento sino que también se generaliza en datos nuevos, también nos interesa dividir de forma aleatoria el conjunto de datos en un conjunto de prueba y de entrenamiento individual. Utilizamos el conjunto de

entrenamiento para entrenar y optimizar nuestro modelo de aprendizaje automático, al tiempo que mantenemos el conjunto de prueba hasta el final para evaluar el modelo final.

Entrenar y seleccionar un modelo predictivo

Como veremos en capítulos próximos, se han desarrollado diferentes tipos de algoritmos de aprendizaje automático para resolver distintas tareas problemáticas. Un punto importante que se puede resumir de los famosos teoremas de *No hay almuerzo gratis* de David Wolpert es que no podemos aprender «gratis». Algunas publicaciones más importantes son *The Lack of A Priori Distinctions Between Learning Algorithms* [La falta de distinciones *a priori* entre los algoritmos de aprendizaje], D. H. Wolpert (1996); *No free lunch theorems for optimization* [Teoremas de no hay almuerzo gratis para la optimización], D. H. Wolpert y W.G. Macready (1997).

Intuitivamente, podemos relacionar este concepto con la popular frase: «Si tu única herramienta es un martillo, tiendes a tratar cada problema como si fuera un clavo» (Abraham Maslow, 1966). Por ejemplo, cada algoritmo de clasificación tiene sus sesgos inherentes, y ninguna clasificación individual es superior si no hacemos suposiciones sobre la tarea. En la práctica, resulta esencial comparar como mínimo un puñado de algoritmos distintos para entrenar y seleccionar el mejor modelo de rendimiento. Pero antes de comparar los diferentes modelos, debemos decidir una unidad para medir el rendimiento. Una unidad de medida que se utiliza con frecuencia es la precisión de la clasificación, que se define como la proporción de instancias clasificadas correctamente.

Una cuestión legítima que podemos preguntarnos es: «¿Cómo podemos saber qué modelo funciona bien en el conjunto de datos final y los datos reales si no utilizamos este conjunto de prueba para la selección del modelo, pero sí lo mantenemos para la evolución del modelo final?». Para abordar el problema incluido en esta cuestión, se pueden utilizar diferentes técnicas de validación cruzada, donde el conjunto de datos de entrenamiento se divide en subconjuntos de validación y entrenamiento para estimar el rendimiento de generalización del modelo. Al final, no podemos

esperar que los parámetros predeterminados de los diferentes algoritmos de aprendizaje proporcionados por las librerías de los programas sean óptimos para nuestra tarea problemática concreta. Por lo tanto, utilizaremos a menudo técnicas de optimización de hiperparámetros que nos ayudarán, en próximos capítulos, a afinar el rendimiento de nuestro modelo. Intuitivamente, podemos pensar en dichos hiperparámetros como parámetros que no se aprenden de los datos sino que representan los botones de un modelo que podemos girar para mejorar su rendimiento. Todo esto quedará más claro en capítulos posteriores, donde veremos ejemplos reales.

Evaluar modelos y predecir instancias de datos no vistos

Después de haber seleccionado un modelo instalado en el conjunto de datos de entrenamiento, podemos utilizar el conjunto de datos de prueba para estimar cómo funciona con los datos no vistos para estimar el error de generalización. Si su rendimiento nos satisface, ya podemos utilizar este modelo para predecir nuevos y futuros datos. Es importante observar que los parámetros para los procedimientos mencionados anteriormente, como el escalado de características y la reducción de dimensionalidad, solo pueden obtenerse a partir de conjuntos de datos de entrenamiento, y que los mismos parámetros vuelven a aplicarse más tarde para transformar el conjunto de datos de prueba, así como cualquier nueva muestra de datos. De otro modo, el rendimiento medido en los datos de prueba puede ser excesivamente optimista.

Utilizar Python para el aprendizaje automático

Python es uno de los lenguajes de programación más populares para la ciencia de datos y, por ello, cuenta con un elevado número de útiles librerías complementarias desarrolladas por sus excelentes desarrolladores y su comunidad de código abierto.

Aunque el rendimiento de los lenguajes interpretados –como Python– para tareas de cálculo intensivo es inferior al de los lenguajes de bajo nivel, se han desarrollado librerías como NumPy

y SciPy sobre implementaciones de C y Fortran de capa inferior para operaciones rápidas y vectorizadas en matrices multidimensionales.

Para tareas de programación de aprendizaje automático, haremos referencia sobre todo a la librería scikit-learn, que actualmente es una de las librerías de aprendizaje automático de código abierto más popular y accesible.

Instalar Python y sus paquetes desde el Python Package Index

Python está disponible para los tres sistemas operativos principales –Microsoft Windows, macOS y Linux– y tanto el instalador como la documentación se pueden descargar desde el sitio web oficial de Python: <https://www.python.org>.

Este libro está escrito para Python versión 3.5.2 o posterior, y es recomendable que utilices la versión más reciente de Python 3 que esté disponible actualmente, aunque la mayoría de los ejemplos de código también son compatibles con Python 2.7.13 o superior. Si decides utilizar Python 2.7 para ejecutar los ejemplos de código, asegúrate de que conoces las diferencias principales entre ambas versiones. Puedes consultar un buen resumen de las diferencias entre Python 3.5 y 2.7 en

<https://wiki.python.org/moin/Python2orPython3>.

Los paquetes adicionales que se utilizarán en este libro se pueden instalar mediante el programa de instalación `pip`, que forma parte de la librería estándar de Python desde la versión 3.3. Puedes encontrar más información sobre el `pip` en

<https://docs.python.org/3/installing/index.html>.

Una vez hemos instalado Python con éxito, podemos ejecutar el `pip` desde el terminal para instalar los paquetes de Python adicionales:

```
pip install SomePackage
```

Los paquetes ya instalados pueden ser actualizados con el comando
`--upgrade`:

```
pip install SomePackage --upgrade
```

Utilizar la distribución y el gestor de paquetes Anaconda de Python

Una distribución de Python alternativa muy recomendada para cálculo científico es Anaconda, de Continuum Analytics. Anaconda es una distribución gratuita (incluso para uso comercial) de Python preparada para la empresa, que incluye todos los paquetes de Python esenciales para la ciencia de datos, matemáticas e ingeniería en una distribución multiplataforma fácil de usar. El instalador de Anaconda se puede descargar desde <http://continuum.io/downloads> y hay disponible una guía de inicio rápido de Anaconda en <https://conda.io/docs/test-drive.html>.

Tras haber instalado Anaconda con éxito, podemos instalar los nuevos paquetes de Python mediante el siguiente comando:

```
conda install SomePackage
```

Los paquetes existentes se pueden actualizar mediante el siguiente comando:

```
conda update SomePackage
```

Paquetes para cálculo científico, ciencia de datos y aprendizaje automático

En este libro, utilizaremos principalmente matrices multidimensionales de NumPy para almacenar y manipular datos. De forma ocasional, utilizaremos pandas, una librería creada sobre NumPy que proporciona herramientas de manipulación de datos de alto nivel que permiten trabajar con datos tabulados de un modo más conveniente. Para aumentar nuestra experiencia de aprendizaje y visualizar datos cuantitativos, que a menudo es extremadamente útil para dar sentido a todo esto de manera intuitiva, utilizaremos la librería Matplotlib, que se puede personalizar en muchos aspectos.

Las versiones de los principales paquetes de Python que se han utilizado para escribir este libro son las que aparecen en la siguiente lista. Asegúrate de que la versión de los paquetes que has instalado sea igual o superior a estas versiones para garantizar que los

ejemplos de código funcionen correctamente:

- NumPy 1.12.1
- SciPy 0.19.0
- scikit-learn 0.18.1
- Matplotlib 2.0.2
- pandas 0.20.1

Resumen

En este capítulo, hemos explorado el aprendizaje automático desde un nivel muy alto y nos hemos familiarizado con el panorama general y los principales conceptos que vamos a explorar con mayor detalle en los próximos capítulos. Hemos aprendido que el aprendizaje supervisado está compuesto por dos importantes subcampos: clasificación y regresión. Mientras que los modelos de clasificación nos permiten categorizar objetos en clases conocidas, podemos utilizar el análisis de regresión para predecir el resultado continuo de variables de destino. El aprendizaje sin supervisión no solo ofrece técnicas útiles para descubrir estructuras en datos sin etiquetar, sino que también puede ser útil para la compresión de datos en las etapas de preprocesamiento de características. Nos hemos referido brevemente a la hoja de ruta típica para aplicar el aprendizaje automático a problemas concretos, que usaremos como base para discusiones más profundas y ejemplos prácticos en los siguientes capítulos. Además, hemos preparado nuestro entorno Python e instalado y actualizado los paquetes necesarios para estar listos para ver en acción el aprendizaje automático.

Más adelante en este libro, además del aprendizaje automático en sí mismo, también presentaremos diferentes técnicas para preprocesar nuestros conjuntos de datos, que nos ayudarán a conseguir el mejor rendimiento de los distintos algoritmos de aprendizaje automático. Si bien cubriremos los algoritmos de clasificación de forma bastante extensa en todo el libro, también exploraremos diferentes técnicas para el análisis de regresión y la agrupación.

Tenemos un emocionante viaje por delante, en el que descubriremos potentes técnicas en el amplio campo del aprendizaje automático. Sin embargo, nos acercaremos al

aprendizaje automático paso a paso, generando nuestro conocimiento de forma gradual a lo largo de los capítulos que componen este libro. En el capítulo siguiente, empezaremos este viaje implementando uno de los algoritmos de aprendizaje automático para clasificación, que nos prepara para el *Capítulo 3, Un recorrido por los clasificadores de aprendizaje automático con scikit-learn*, donde nos acercaremos a algoritmos de aprendizaje automático más avanzados con la librería de código abierto scikit-learn.