

Segunda
edición

Estadística práctica para ciencia de datos con R y Python



Peter Bruce, Andrew Bruce
y Peter Gedeck

Marcombo



SEGUNDA EDICIÓN

Estadística práctica para ciencia de datos con R y Python

Más de 50 conceptos esenciales

Peter Bruce, Andrew Bruce y Peter Gedeck

Marcombo

Segunda edición original publicada en inglés por O'Reilly con el título *Practical Statistics for Data Scientists*, ISBN 978-1-492-07294-2 © Peter Bruce, Andrew Bruce y Peter Gedeck, 2020. *This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.*

Título de la edición en español:

Estadística práctica para ciencia de datos con R y Python

Segunda edición en español, 2022

© 2022 MARCOMBO, S.L.

www.marcombo.com

Diseño de portada: Karen Montgomery

Ilustración: Rebecca Demarest

Traducción: Francisco Martínez Carreño

Corrección: Nuria Barroso

«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. La presente publicación contiene la opinión del autor y tiene el objetivo de informar de forma precisa y concisa. La elaboración del contenido, aunque se ha trabajado de forma escrupulosa, no puede comportar una responsabilidad específica para el autor ni el editor de los posibles errores o imprecisiones que pudiera contener la presente obra.»

ISBN: 978-84-267-3443-3

D.L.: B 1061-2022

Impreso en Servicepoint

Printed in Spain

Nos gustaría dedicar este libro al recuerdo de nuestros padres, Victor G. Bruce y Nancy C. Bruce, que cultivaron la pasión por las matemáticas y la ciencia. También a nuestros primeros tutores John W. Tukey, Julian Simon y a nuestro amigo de toda la vida Geoff Watson, que nos animaron a seguir la carrera de estadística.

Peter Bruce y Andrew Bruce

Me gustaría dedicar este libro a Tim Clark y Christian Kramer, a los que agradezco profundamente su investigación científica y su amistad.

Peter Gedeck

Contenidos

Prefacio	xii
1. Análisis exploratorio de datos.....	1
Elementos de datos estructurados	2
Lecturas complementarias	4
Datos rectangulares.....	4
Marcos de datos e índices.....	6
Estructuras de datos no rectangulares	6
Lecturas complementarias.....	7
Estimación de la localización	7
Media.....	9
Estimación de medianas robustas.....	10
Ejemplo: estimaciones de localización de la población y tasas de homicidios.....	12
Lecturas complementarias.....	13
Estimación de la variabilidad	13
Desviación estándar y estimaciones relacionadas	15
Estimación basada en percentiles.....	17
Ejemplo: estimaciones de variabilidad de la población estatal	18
Lecturas complementarias.....	19
Exploración de la distribución de datos.....	19
Percentiles y diagramas de caja	20
Tablas de frecuencias e histogramas	22
Diagrama y estimación de la curva de densidad.....	24
Lecturas complementarias.....	26
Exploración de datos binarios y categóricos	27
Moda	29
Valor esperado	29
Probabilidad	30
Lecturas complementarias.....	30
Correlación	30
Diagramas de dispersión	34
Lecturas complementarias.....	36
Exploración de dos o más variables.....	36
Agrupación hexagonal y contornos (representación numérica frente a datos numéricos).....	36

Dos variables categóricas	39
Datos categóricos y numéricos	41
Visualización de varias variables	43
Lecturas complementarias	45
Resumen	45
2. Distribuciones de datos y muestreo	47
Muestreo aleatorio y sesgo de la muestra	48
Sesgo	50
Selección aleatoria	51
Tamaño frente a calidad: ¿cuándo importa el tamaño?	52
Media muestral frente a media poblacional	53
Lecturas complementarias	53
Sesgo de selección	54
Regresión a la media	55
Lecturas complementarias	57
Distribución muestral del estadístico	57
Teorema del límite central	60
Error estándar	60
Lecturas complementarias	61
Bootstrap	61
Remuestreo frente a bootstrapping	65
Lecturas complementarias	65
Intervalos de confianza	65
Lecturas complementarias	68
Distribución normal	69
Normal estándar y diagramas QQ	70
Distribuciones de cola larga	72
Lecturas complementarias	74
Distribución t de Student	74
Lecturas complementarias	77
Distribución binomial	77
Lecturas complementarias	79
Distribución chi cuadrado	79
Lecturas complementarias	80
Distribución F	81
Lecturas complementarias	81
La distribución de Poisson y distribuciones relacionadas	81
Distribución de Poisson	82
Distribución exponencial	83
Estimación de la tasa de fallos	83
Distribución de Weibull	84
Lecturas complementarias	85
Resumen	85

3. Experimentos estadísticos y pruebas significativas	87
Prueba A/B.....	88
¿Por qué tener un grupo de control?.....	90
¿Por qué solo A/B? ¿Por qué no C, D, ...?	91
Lecturas complementarias	92
Pruebas de hipótesis	93
La hipótesis nula.....	94
Hipótesis alternativa	95
Pruebas de hipótesis unidireccionales o bidireccionales.....	95
Lecturas complementarias.....	96
Remuestreo	96
Prueba de permutación	97
Ejemplo: adherencia de la web	98
Pruebas de permutación exhaustiva y de bootstrap	102
Pruebas de permutación: el resultado final de la ciencia de datos	102
Lecturas complementarias.....	103
Significación estadística y valores p.....	103
Valor p	106
Alfa	107
Errores de tipo 1 y 2	108
Ciencia de datos y valores p	109
Lecturas complementarias.....	109
Pruebas t.....	109
Lecturas complementarias.....	111
Pruebas múltiples.....	111
Lecturas complementarias.....	115
Grados de libertad	115
Lecturas complementarias.....	117
ANOVA	117
Estadístico F.....	120
ANOVA bidireccional	122
Lecturas complementarias.....	123
Prueba de chi cuadrado.....	123
Prueba de chi cuadrado: enfoque de remuestreo	124
Prueba de chi cuadrado: teoría estadística.....	126
Prueba exacta de Fisher	127
Relevancia para la ciencia de datos	129
Lecturas complementarias.....	130
Algoritmo Multi-Arm Bandit.....	130
Lecturas complementarias.....	134
Potencia y tamaño de la muestra	134
Tamaño de la muestra	135
Lecturas complementarias	138
Resumen	138

4. Regresión y pronóstico	139
Regresión lineal simple.....	139
La ecuación de regresión.....	141
Valores ajustados y residuos.....	143
Mínimos cuadrados.....	145
Pronóstico frente a explicación (elaboración de perfiles)	146
Lecturas complementarias.....	147
Regresión lineal múltiple	147
Ejemplo: datos de las viviendas del condado de King	148
Evaluación del modelo	149
Validación cruzada	151
Selección del modelo y regresión escalonada	152
Regresión ponderada	156
Lecturas complementarias.....	157
Pronóstico mediante la regresión	157
Los peligros de la extrapolación.....	158
Intervalos de confianza y de pronóstico	158
Variables de tipo factor en la regresión	160
Representación de variables ficticias.....	161
Variables de tipo factor con muchos niveles	163
Variables de tipo factor ordenadas.....	165
Interpretación de la ecuación de regresión	166
Predictoras correlacionadas.....	166
Multicolinealidad	168
Variables de confusión	168
Interacciones y efectos principales	170
Diagnósticos de regresión	172
Valores atípicos	173
Valores influyentes.....	175
Heterocedasticidad, anormalidad y errores correlacionados	177
Diagramas de residuos parciales y falta de linealidad	180
Regresión polinomial y por spline	183
Polinomial	183
Splines	185
Modelos aditivos generalizados.....	187
Lecturas complementarias.....	189
Resumen.....	189
5. Clasificación.....	191
Bayes ingenuo	192
Por qué la clasificación bayesiana exacta no es práctica	193
La solución ingenua.....	194
Variables predictoras numéricas.....	196
Lecturas complementarias.....	197

Análisis discriminante.....	197
Matriz de covarianza.....	198
Discriminante lineal de Fisher.....	199
Un ejemplo sencillo.....	200
Lecturas complementarias.....	203
Regresión logística.....	203
Función de respuesta logística y logit	204
Regresión logística y GLM	206
Modelos lineales generalizados	207
Valores pronosticados de regresión logística	208
Interpretación de los coeficientes y de la razón de oportunidades.....	208
Regresión lineal y logística: similitudes y diferencias	210
Evaluación del modelo	211
Lecturas complementarias.....	214
Evaluación de modelos de clasificación	215
Matriz de confusión	216
El problema de las clases raras	218
Precisión, exhaustividad y especificidad	218
Curva ROC	219
AUC.....	221
Sustentación.....	222
Lecturas complementarias.....	224
Estrategias para datos que no están equilibrados	224
Submuestreo	225
Sobremuestreo y aumento/disminución de la ponderación	226
Generación de datos	228
Clasificación basada en los costes.....	228
Exploración de pronósticos	229
Lecturas complementarias.....	230
Resumen.....	230
6. Aprendizaje automático estadístico.....	231
K-vecinos más cercanos.....	232
Un pequeño ejemplo: pronóstico del incumplimiento de préstamos ...	233
Métricas de distancia	235
Codificador One-Hot	236
Estandarización (normalización, puntuación z)	237
Elección de K	240
KNN como motor de características.....	241
Modelos de árbol.....	243
Un ejemplo sencillo.....	244
Algoritmo de partición recursiva.....	246
Medición de la homogeneidad o la impureza.....	248
Detención del crecimiento del árbol.....	249
Pronóstico de un valor continuo	251

Cómo se utilizan los árboles.....	252
Lecturas complementarias	253
Métodos de bagging y bosque aleatorio	253
Bagging.....	254
Bosque aleatorio	255
Importancia de la variable.....	259
Hiperparámetros.....	262
Boosting.....	263
El algoritmo boosting	264
XGBoost.....	265
Regularización: evitación del sobreajuste.....	268
Hiperparámetros y validación cruzada	272
Resumen	275
7. Aprendizaje no supervisado.....	277
Análisis de componentes principales.....	278
Un ejemplo sencillo	279
Cálculo de los componentes principales	282
Interpretación de componentes principales	282
Análisis de correspondencias	285
Lecturas complementarias	287
Agrupación K-means.....	287
Un ejemplo sencillo	288
Algoritmo K-means	290
Interpretación de los grupos	291
Selección del número de grupos	293
Agrupación jerárquica.....	296
Un ejemplo sencillo	296
El dendrograma	297
El algoritmo de aglomeración.....	299
Medidas de disimilitud	299
Agrupación basada en el modelo.....	301
Distribución normal multivariante	301
Mezclas de distribuciones normales.....	303
Selección del número de grupos	305
Lecturas complementarias	308
Variables categóricas y escalado.....	308
Escalado de variables.....	309
Variables dominantes	310
Datos categóricos y distancia de Gower.....	311
Problemas con la agrupación de datos mixtos.....	314
Resumen	316
Bibliografía	317

Prefacio

Este libro está dirigido a científicos de datos familiarizados de algún modo con los lenguajes de programación *R* y/o *Python*, y con una formación básica (quizás irregular o efímera) a la estadística. Dos de los autores de este libro llegamos al mundo de la ciencia de datos desde el mundo de la estadística y apreciamos en cierta medida la contribución que la estadística puede hacer al arte de la ciencia de datos. Al mismo tiempo, somos muy conscientes de las limitaciones de la enseñanza de la estadística tradicional: la estadística como disciplina tiene un siglo y medio de vida, y la mayoría de los libros de texto y cursos de estadística están cargados con el impulso y la inercia de un transatlántico. Todos los métodos de este libro tienen alguna conexión, histórica o metodológica, con la disciplina de la estadística. No se incluyen los métodos que evolucionaron principalmente a partir de la informática, como es el caso de las redes neuronales.

El libro tiene dos objetivos:

- Presentar, en forma digerible, navegable y de fácil referencia, conceptos clave de estadística que son relevantes para la ciencia de datos.
- Explicar qué conceptos son importantes y útiles desde la perspectiva de la ciencia de datos, cuáles lo son menos y por qué.

Convenciones que se utilizan en el libro

Términos clave

La ciencia de datos es la fusión de varias disciplinas, entre las que se incluyen la estadística, la informática, las tecnologías de la información y campos específicos de este ámbito. Como consecuencia, se pueden utilizar varios términos diferentes para referirse a un concepto dado. Los términos clave y sus sinónimos se destacarán a lo largo del libro en un recuadro como este.



Este elemento indica un consejo o una sugerencia.



Este elemento indica una nota general.



Este elemento indica una advertencia o precaución.

Uso de los ejemplos de código

En todos los casos, el libro proporciona ejemplos de código, en primer lugar en *R* y a continuación en *Python*. Para evitar repeticiones innecesarias, generalmente solo se muestran los resultados y gráficos que genera el código *R*. También omitimos el código necesario para cargar tanto los paquetes como los conjuntos de datos requeridos. Se puede acceder al código completo y los conjuntos de datos para su descarga en www.marcombo.info con el código DATOS22.

Este libro está aquí para ayudarte a hacer tu trabajo. En general, si en el libro se facilitan códigos de ejemplo, puedes usarlos en tus programas y documentación. No necesitas ponerte en contacto con nosotros para pedir permiso a menos que vayas a reproducir una parte importante del código. Por ejemplo, escribir un programa que use varios fragmentos de código del libro no requiere permiso. Vender o distribuir ejemplos de los libros de O'Reilly y de Marcombo requiere permiso. Responder a una pregunta proporcionando la referencia del libro y citar el código de un ejemplo no requiere permiso. La incorporación de una cantidad importante del código de los ejemplos del libro en la documentación de tu producto requiere permiso.

Generalmente no pedimos que se incluya una atribución, pero apreciamos que se haga. Una atribución contiene el título, autor, editor e ISBN. Por ejemplo: "Estadística práctica para ciencia de datos con *R* y *Python* de Peter Bruce, Andrew Bruce y Peter Gedeck (Marcombo y O'Reilly). Copyright 2022 Peter Bruce, Andrew Bruce y Peter Gedeck, 978-84-267-3443-3".

Si crees que el uso por tu parte de los ejemplos de código no está justificado o no respeta el permiso otorgado más arriba, no dudes en ponerte en contacto con nosotros en info@marcombo.com.

Agradecimientos

Los autores queremos expresar nuestro agradecimiento al numeroso grupo de personas que han contribuido a hacer realidad este libro.

Gerhard Pilcher, director ejecutivo de la empresa de minería de datos Elder Research, revisó los primeros borradores del libro y nos facilitó sus correcciones, así como detallados y útiles comentarios. Del mismo modo, Anya McGuirk y Wei Xiao, estadísticos de SAS, y Jay Hilfiger, autor de O'Reilly, proporcionaron comentarios útiles sobre los borradores iniciales del libro. Toshiaki Kurokawa, que tradujo la primera edición al japonés, realizó un completo trabajo de revisión y corrección durante la traducción. Aaron Schumacher y Walter Paczkowski revisaron minuciosamente la segunda edición del libro y aportaron numerosas sugerencias útiles y valiosas por las cuales les estamos muy agradecidos. No hace falta decir que cualquier error que exista es solo nuestro.

En O'Reilly, Shannon Cutt nos orientó con buen ánimo y la adecuada dosis de insistencia en el proceso de publicación. Por otra parte, Kristen Brown dirigió con éxito la fase de producción del libro. Rachel Monaghan y Eliahu Sussman, cuidadosa y pacientemente, corrigieron y mejoraron la redacción, mientras que Ellen Troutman-Zaig preparó el índice. Nicole Tache tomó las riendas de la segunda edición y dirigió el proceso de manera eficaz, proporcionando muchas y buenas sugerencias editoriales para mejorar la legibilidad del libro para una amplia audiencia. También agradecemos a Marie Beaugureau, que inició nuestro proyecto en O'Reilly, así como a Ben Bengfort, autor de O'Reilly e instructor de Statistics.com, que nos presentó a O'Reilly.

Nosotros, y los contenidos de este libro, también nos hemos beneficiado de las muchas conversaciones que Peter ha tenido a lo largo de los años con Galit Shmueli, coautor de otros proyectos editoriales.

Finalmente, nos gustaría agradecer especialmente a Elizabeth Bruce y Deborah Donnell su paciencia y apoyo para hacer posible este proyecto.

CAPÍTULO 1

Análisis exploratorio de datos

Este capítulo se centra en el primer paso de cualquier proyecto de ciencia de datos: la exploración de los datos.

La estadística clásica se ocupó casi exclusivamente de la inferencia, un conjunto de procedimientos, a veces complejo, para sacar conclusiones sobre grandes poblaciones a partir de muestras de pequeño tamaño. En 1962, John W. Tukey (figura 1.1) propugnó una reforma de la estadística en su trabajo académico de investigación "The Future of Data Analysis" [Tukey, 1962]. Propuso una nueva disciplina científica llamada *análisis de datos* (*data analysis*) que incluía la inferencia estadística como un componente más. Tukey forjó vínculos con los colectivos profesionales de ingeniería e informática (acuñó los términos *bit*, abreviatura de dígito binario, y *software*), y sus principios originales son sorprendentemente duraderos y forman parte de los fundamentos de la ciencia de datos. El campo del análisis exploratorio de datos se estableció con el que es ahora un libro clásico de Tukey publicado en 1977: *Exploratory Data Analysis* [Tukey, 1977]. Tukey presentó diagramas sencillos (por ejemplo, diagramas de caja, diagramas de dispersión) que, junto con resúmenes estadísticos (media, mediana, cuantiles, etc.), ayudan a dibujar la imagen de un conjunto de datos.

Con la disponibilidad de forma inmediata de la capacidad de cálculo y el potente software de análisis de datos, el análisis exploratorio de datos ha evolucionado mucho más allá de lo que fue su alcance original. Los impulsores clave de esta disciplina han sido el rápido desarrollo de nuevas tecnologías, el acceso a una mayor cantidad de datos y más importantes, así como la mayor utilización del análisis cuantitativo en numerosas disciplinas. David Donoho, profesor de estadística de la Universidad de Stanford y exalumno de Tukey, es autor de un excelente artículo inspirado en su exposición para el taller del centenario del nacimiento de Tukey (Tukey Centennial) en Princeton, Nueva Jersey [Donoho, 2015]. Donoho localiza la génesis de la ciencia de datos en el trabajo pionero de Tukey sobre el análisis de datos.



Figura 1.1 John Tukey, el eminente estadístico cuyas ideas, que desarrolló hace más de 50 años, constituyen la base de la ciencia de datos.

Elementos de datos estructurados

Los datos proceden de muchas fuentes: mediciones de sensores, eventos, texto, imágenes y videos. *Internet de las cosas* (*Internet of Things* [IoT]) vomita caudales de información. Gran parte de estos datos no están estructurados: las imágenes son una colección de píxeles, y cada píxel contiene información sobre el color RGB (rojo, verde, azul). Los textos son secuencias de palabras y caracteres que no son palabras, y a menudo están organizados por secciones, subsecciones, etc. Los flujos de clics son secuencias de acciones de un usuario que interactúa con una aplicación o una página web. De hecho, un desafío importante de la ciencia de datos es convertir este torrente de datos sin procesar en información útil. Para aplicar los conceptos estadísticos tratados en el libro, los datos en bruto, no estructurados, deben procesarse y manejarse en un formato estructurado. Una de las formas más habituales en la que se presentan los datos estructurados es una tabla con filas y columnas, bien porque los datos pueden proceder de una base de datos relacional o porque los recopilamos así para su estudio.

Hay dos tipos básicos de datos estructurados: numéricos y categóricos. Los datos numéricos se presentan en dos modalidades: datos de carácter *continuo* (*continuous*), como es el caso de la velocidad del viento o la duración del tiempo, y *discreto* (*discrete*), como el recuento de las veces que ocurre un evento. Los datos *categóricos* (*categorical*) adoptan solo un conjunto fijo de valores, como pueden ser los tipos de pantallas de TV (plasma, LCD, led, etc.) o los nombres de los estados (Alabama, Alaska, etc.). Los datos *binarios* (*binary*) son un importante caso especial de datos categóricos que adoptan solo uno de dos valores, como 0/1, sí/no o verdadero/falso. Otro tipo muy útil de datos categóricos son los datos *ordinales* (*ordinal*), con los que se ordenan las categorías. Un ejemplo de estos datos es una clasificación numérica (1, 2, 3, 4 o 5).

¿Por qué nos molestamos en analizar la taxonomía de los tipos de datos? Pues porque resulta que a los efectos del análisis de datos y del modelado predictivo, el tipo de datos es importante para ayudar a determinar qué tipo de presentación visual utilizar, si la de análisis de datos o la de modelos estadísticos. De hecho, el software de la ciencia de datos, como *R* y *Python*, utiliza los tipos de datos para mejorar el rendimiento computacional. Y lo que es más importante, el tipo de datos de una variable determina cómo realizará el software los cálculos para esa variable.

Términos clave de los tipos de datos

Numéricos

Datos que se expresan en una escala numérica.

Continuos

Datos que pueden adoptar cualquier valor dentro de un intervalo. (*Sinónimos:* intervalo, flotante, numérico)

Discretos

Datos que solo pueden adoptar valores enteros, como los valores contables. (*Sinónimos:* entero, contable)

Categóricos

Datos que solo pueden adoptar un conjunto específico de valores que representan un conjunto de categorías posibles. (*Sinónimos:* enumeraciones, enumerado, factores, nominal)

Binarios

Son un caso especial de datos categóricos con solo dos categorías de valores, por ejemplo, 0/1, verdadero/falso. (*Sinónimos:* dicotómico, lógico, indicador, booleano)

Ordinales

Datos categóricos que tienen un orden explícito. (*Sinónimos:* factor ordenado)

Los ingenieros de software y los programadores de bases de datos pueden preguntarse por qué incluso necesitamos la noción de datos *categóricos* (*categorical*) y *ordinales* (*ordinal*) para el análisis. Después de todo, las categorías son simplemente una colección de valores de texto (o numéricos) y la base de datos subyacente maneja automáticamente la representación interna. Sin embargo, la identificación explícita de datos como categóricos, como algo diferente al texto, ofrece algunas ventajas:

- Saber que los datos son categóricos puede actuar como una señal que indica al software cómo deben comportarse los procedimientos estadísticos, como crear un gráfico o ajustar un modelo. En particular, los datos ordinales se pueden representar mediante `ordered.factor` en R, conservando el orden especificado por el usuario en gráficos, tablas y modelos. En Python, scikit-learn admite datos ordinales con `sklearn.preprocessing.OrdinalEncoder`.
- El almacenamiento y la indexación se pueden optimizar (como en una base de datos relacional).
- Los posibles valores que puede adoptar una variable categórica dada se aplican al software (como puede ser una enumeración).

El tercer "beneficio" puede conducir a un comportamiento no deseado o inesperado: el comportamiento predeterminado de las funciones de importación de datos en R (por ejemplo, `read.csv`) es convertir automáticamente una columna de texto en un factor.

Las operaciones posteriores en esa columna supondrán que los únicos valores permitidos para la columna son los que se importaron originalmente, y la asignación de un nuevo valor de texto introducirá una advertencia y producirá un NA (valor ausente). El paquete pandas de *Python* no realizará dicha conversión automáticamente. Sin embargo, podemos especificar una columna como categórica explícitamente con la función `read_csv`.

Ideas clave

- Los datos se clasifican en el software generalmente por tipos.
- En los tipos de datos están incluidos los numéricos (continuos, discretos) y categóricos (binarios, ordinales).
- La tipificación de datos en el software actúa como una señal que le indica al software cómo debe procesarlos.

Lecturas complementarias

- Los tipos de datos pueden crear confusión, ya que estos pueden superponerse, y la taxonomía de un tipo de software puede diferir de la de otro. El sitio web de R-Tutorial (www.r-tutor.com/) incluye la taxonomía de *R*. La documentación de pandas (https://pandas.pydata.org/docs/user_guide/) describe los diferentes tipos de datos y cómo se pueden manejar en *Python*.
- La clasificación de tipos de datos de las bases de datos es más detallada al incorporar consideraciones de niveles de precisión, campos de longitud fija o variable, entre otras cosas. Consultar la guía de W3Schools para SQL (www.w3schools.com/sql/).

Datos rectangulares

El marco de referencia típico para un análisis en ciencia de datos es el objeto de *datos rectangulares* (*rectangular data*), como puede ser una hoja de cálculo o una tabla de base de datos.

Datos rectangulares (*rectangular data*) es el término general asociado a una matriz bidimensional con filas que indican registros (casos) y columnas que indican características (variables). El *marco de datos* (*data frame*) es el formato específico en *R* y *Python*. Los datos no siempre tienen esta apariencia al principio: los datos no estructurados (por ejemplo, texto) deben procesarse y manipularse para que se puedan representar como un conjunto de características en la matriz de datos rectangulares (consultar "Elementos de datos estructurados" en la página 2). Para la mayor parte de las tareas de análisis y modelado de datos, los datos de las bases de datos relacionales deben extraerse y colocarse en una sola tabla.

Términos clave de los datos rectangulares

Marco de datos

Los datos rectangulares (como puede ser una hoja de cálculo) son la estructura básica de datos para los modelos estadísticos y de aprendizaje automático.

Característica

Una columna de una tabla se denomina generalmente *característica* (*feature*).

Sinónimos

atributo, entrada, predictor, variable

Resultado

Muchos proyectos de ciencia de datos implican pronosticar un *resultado* (*outcome*), a menudo un resultado de sí/no (en la tabla 1.1, es si "la subasta ha sido competitiva o no").

A veces, las *características* (*features*) se utilizan para pronosticar el *resultado* (*outcome*) de un estudio.

Sinónimos

variable dependiente, respuesta, objetivo, salida

Registros

A una fila dentro de una tabla se le denomina generalmente *registro* (*register*).

Sinónimos

caso, ejemplo, instancia, observación, patrón, muestra

Tabla 1.1 Típico formato del marco de datos

Categoría	divisa	sellerRating	Duración	endDay	ClosePrice	OpenPrice	¿Competitiva?
Música/Películas/Juegos	Dólar	3249	5	Lun	0.01	0.01	0
Música/Películas/Juegos	Dólar	3249	5	Lun	0.01	0.01	0
Automoción	Dólar	3115	7	Mar	0.01	0.01	0
Automoción	Dólar	3115	7	Mar	0.01	0.01	0
Automoción	Dólar	3115	7	Mar	0.01	0.01	0
Automoción	Dólar	3115	7	Mar	0.01	0.01	0
Automoción	Dólar	3115	7	Mar	0.01	0.01	1
Automoción	Dólar	3115	7	Mar	0.01	0.01	1

En la tabla 1.1, hay una combinación de datos medidos o contabilizados (por ejemplo, duración y precio) y datos categóricos (por ejemplo, categoría y divisa). Como se ha mencionado anteriormente, una forma especial de variable categórica es una variable binaria (sí/no o 0/1), como se ve en la columna de la derecha en la tabla 1.1, una variable indicadora que muestra si una subasta ha sido competitiva (si ha tenido varios postores) o no. Esta variable indicadora también resulta ser una variable de *resultado* (*outcome*), cuando el escenario es pronosticar si una subasta es competitiva o no.

Marcos de datos e índices

Las tablas de bases de datos tradicionales tienen una o más columnas designadas como índice, esencialmente un número de fila. Esta funcionalidad puede mejorar enormemente la eficiencia de determinadas consultas a bases de datos. En *Python*, con la biblioteca de pandas, la estructura básica de datos rectangulares es el objeto `DataFrame`. Por defecto, se crea un índice de enteros automático para un `DataFrame` basado en el orden de las filas. En pandas, también es posible establecer índices jerárquicos/multinivel para mejorar la eficiencia de ciertas operaciones.

En *R*, la estructura básica de datos rectangulares es el objeto `data.frame`. `data.frame` también tiene un índice implícito de enteros basado en el orden de las filas. El `data.frame` nativo de *R* no admite índices especificados por el usuario o multinivel, aunque se puede crear una clave personalizada mediante el atributo `row.names`. Para superar esta deficiencia, hay dos nuevos paquetes que se están utilizando de forma generalizada: `data.table` y `dplyr`. Ambos admiten índices multinivel y ofrecen importantes aumentos de velocidad cuando se trabaja con `data.frame`.



Diferencias terminológicas

La terminología de datos rectangulares puede resultar confusa. Los estadísticos y los científicos de datos usan términos diferentes para referirse al mismo concepto. Para un estadístico, las *variables predictoras* (*predictor variables*) se utilizan en un modelo para pronosticar una *respuesta* (*response*) o *variable dependiente* (*dependent variable*). Para un científico de datos, las *características* (*features*) se utilizan para pronosticar un *objetivo* (*target*). Los sinónimos resultan particularmente confusos: los informáticos usarán el término *muestra* (*sample*) para una única fila. Una *muestra* (*sample*) para un estadístico es una colección de filas.

Estructuras de datos no rectangulares

Existen otras estructuras de datos además de los datos rectangulares.

Los datos de serie de tiempo registran mediciones sucesivas de la misma variable. Es la materia prima de los métodos de pronóstico estadístico y también es un componente clave de los datos generados por los dispositivos del Internet de las cosas.

Las estructuras de datos espaciales, que se utilizan en cartografía y análisis de la localización, son más complejas y variadas que las estructuras de datos rectangulares. En la representación del *objeto* (*object*), el foco de los datos es un objeto (por ejemplo, una casa) y sus coordenadas espaciales. La vista de *campo* (*field*), por el contrario, se centra en pequeñas unidades de espacio y el valor de la métrica correspondiente (brillo de píxeles, por ejemplo).

Las estructuras de datos en forma de gráficos (o redes) se utilizan para representar relaciones físicas, sociales y abstractas. Por ejemplo, un gráfico de una red social, como Facebook o LinkedIn, puede representar conexiones entre personas en la red. Los centros de distribución conectados por carreteras son un ejemplo de una red física. Las estructuras de gráficos son útiles para ciertos tipos de problemas, como la optimización de redes o los sistemas de recomendación.

Cada uno de estos tipos de datos tiene su metodología especializada en ciencia de datos. El enfoque de este libro se centra en los datos rectangulares, el bloque de construcción fundamental del modelado predictivo.



Gráficos en estadística

En informática y tecnologías de la información, el término *gráfico (graph)* hace referencia normalmente a la descripción de las conexiones entre entidades y a la estructura de datos subyacente. En estadística, *gráfico (graph)* se utiliza para referirse a una variedad de diagramas y *visualizaciones (visualizations)*, no solo de conexiones entre entidades, aunque el término se aplica solo a la visualización, no a la estructura de datos.

Ideas clave

- La estructura de datos básica en ciencia de datos es una matriz rectangular en la que las filas están formadas por registros y las columnas son las variables (características).
- La terminología puede resultar confusa. Existe una serie de sinónimos, procedentes de diferentes disciplinas, que contribuyen a la ciencia de datos (estadística, informática y tecnologías de la información).

Lecturas complementarias

- Documentación sobre marcos de datos en *R* (<https://stat.ethz.ch/R-manual/R-devel/library/base/html/data.frame.html>).
- Documentación sobre marcos de datos en *Python* (https://pandas.pydata.org/pandas-docs/stable/user_guide/dsintro.html).

Estimación de la localización

Las variables, con los datos medidos o procedentes de recuentos, pueden tener miles de valores distintos. Un paso fundamental para explorar los datos es obtener un "valor típico" para cada característica (variable): una estimación de dónde se encuentra la mayoría de los datos (es decir, su tendencia central).

Términos clave de la estimación de la localización

Media

Suma de todos los valores dividida por el número de valores.

Sinónimos

promedio

Media ponderada

Suma de todos los valores multiplicados por cada ponderación y dividida por la suma de las ponderaciones.

Sinónimo

promedio ponderado

Mediana

Valor tal que la mitad del número de datos se encuentra por encima y la otra mitad por debajo de dicho valor.

Sinónimo

Percentil 50

Percentil

Valor tal que el P por ciento de los datos se encuentra por debajo del mismo.

Sinónimo

cuantil

Mediana ponderada

Valor tal que la mitad de la suma de las ponderaciones se encuentra por encima y la otra mitad por debajo de los datos ordenados.

Media recortada

El promedio de todos los valores después de eliminar un número fijo de valores extremos.

Sinónimo

media truncada

Robusto

Insensible a valores extremos.

Sinónimo

resistente

Atípico

Valor de un dato que es muy diferente de la mayoría de los valores de datos.

Sinónimo

valor extremo

A primera vista, resumir los datos puede parecer bastante trivial: simplemente hay que extraer la *media* (*mean*) de los datos. De hecho, si bien la media es fácil de calcular y conveniente de usar, es posible que no siempre sea la mejor medida para representar un valor central. Por esta razón, los estadísticos han desarrollado y promovido varias estimaciones alternativas a la media.



Métricas y estimaciones

Los estadísticos a menudo utilizan el término *estimación* (*estimate*) para referirse a un valor calculado a partir de los datos disponibles, para establecer una distinción entre lo que vemos a partir de los datos y el verdadero estado teórico o exacto de las cosas. Es más probable que los científicos de datos y los analistas de negocios se refieran a este valor como *métrica* (*metric*). La diferencia refleja el enfoque de la estadística frente al de la ciencia de datos. Hay que tener en cuenta que la explicación de la incertidumbre se encuentra en el corazón de la disciplina de la estadística, mientras que el foco de la ciencia de datos son los objetivos concretos de las organizaciones o de las empresas. Por lo tanto, los estadísticos estiman y los científicos de datos miden.

Media

La estimación más elemental para la localización es el valor medio o *promedio* (*average*). La media es la suma de todos los valores dividida por el número de valores. Consideremos el siguiente conjunto de números: {3 5 1 2}. La media es $(3 + 5 + 1 + 2) / 4 = 11/4 = 2.75$. Nos encontraremos el símbolo \bar{x} (pronunciado "barra x") que se utiliza para representar la media de la muestra de una población. La fórmula para calcular la media de un conjunto de n valores x_1, x_2, \dots, x_n es:

$$\text{Media} = \bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$



N (o n) se refiere al número total de registros u observaciones. En estadística, se escribe con mayúscula si se refiere a una población y en minúscula si se refiere a una muestra de una población. En la ciencia de datos, esa distinción no es vital, por lo que se puede ver de las dos formas.

La variación de la media se conoce como *media truncada* (*trimmed mean*), que se calcula ignorando un número fijo, en cada extremo, de valores ordenados y a continuación se calcula el promedio de los valores restantes. Al representar los valores ordenados por $x_{(1)}, x_{(2)}, \dots, x_{(n)}$ donde $x_{(1)}$ es el valor más pequeño y $x_{(n)}$ el valor más grande, la fórmula para calcular la media recortada con los p valores más pequeños y más grandes omitidos es:

$$\text{Media truncada} = \bar{x} = \frac{\sum_{i=p+1}^{n-p} x_{(i)}}{n-2p}$$

La media truncada elimina la influencia de valores extremos. Por ejemplo, en el buceo internacional, se eliminan las puntuaciones máxima y mínima de cinco jueces, y la puntuación final es el promedio de las puntuaciones de los tres jueces restantes ([https://en.wikipedia.org/wiki/Diving_\(sport\)#Scoring_the_dive](https://en.wikipedia.org/wiki/Diving_(sport)#Scoring_the_dive)). Esto hace que sea difícil para un solo juez manipular la puntuación, tal vez para favorecer al concursante de su país. Las medias truncadas se utilizan habitualmente y, en muchos casos, son preferibles a la media ordinaria. En el apartado "Estimación de medianas robustas" de la página 10 se amplía esta información.

Otro tipo de media es la *media ponderada* (*weighted mean*), que se calcula multiplicando cada valor de datos x_i por el peso w_i especificado por el usuario y dividiendo su suma por la suma de las ponderaciones. La fórmula para una media ponderada es:

$$\text{Media ponderada} = \bar{x}_w = \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}$$

Hay dos motivos fundamentales para usar una media ponderada:

- Algunos valores son intrínsecamente más variables que otros, y las observaciones muy variables reciben un peso menor. Por ejemplo, si tomamos el promedio de varios sensores y uno de los sensores es menos preciso, entonces podríamos reducir la ponderación de los datos de ese sensor.
- Los datos recopilados no representan por igual a los diferentes grupos que nos interesa medir. Por ejemplo, debido a la forma en la que se ha realizado un experimento en línea, es posible que no tengamos un conjunto de datos que refleje con precisión todos los grupos en la base de datos de los usuarios. Para corregir eso, podemos dar un mayor peso a los valores de los grupos que tengan una menor representación.

Estimación de medianas robustas

La *mediana* (*median*) es el valor central de una lista de datos ordenados de menor a mayor. Si hay un número par de valores de datos, el valor medio es uno que no está realmente en el conjunto de datos, sino el promedio de los dos valores que dividen los datos ordenados en mitades superior e inferior. En comparación con la media, que utiliza todas las observaciones, la mediana depende solo de los valores situados en el centro de los datos ordenados. Si bien esto puede parecer una desventaja, dado que la media es mucho más sensible a los datos, hay muchos casos en los que la mediana es una mejor métrica para la localización. Supongamos que queremos analizar los ingresos familiares típicos en los vecindarios de los alrededores del lago Washington en Seattle. Al comparar el vecindario de Medina con el de Windermere, la utilización de la media produciría resultados muy diferentes porque Bill Gates vive en Medina. Si usamos la mediana, no importará lo rico que sea Bill Gates, porque la posición de la observación intermedia seguirá siendo la misma.

Por las mismas razones por las que se usa una media ponderada, también es posible calcular una *mediana ponderada* (*weighted median*). Al igual que con la mediana, primero ordenamos los datos, aunque cada valor de los datos tiene una ponderación asociada. En lugar del número del medio, la mediana ponderada es un valor tal que la suma de las ponderaciones es igual para las mitades inferior y superior de la lista ordenada. Como la mediana, la mediana ponderada es robusta a valores atípicos.

Valores atípicos

A la mediana se le conoce como una estimación *robusta* (*robust*) de la localización, ya que no está influenciada por *valores atípicos* (*outliers*) (casos extremos) que podrían sesgar los resultados. Un valor atípico es cualquier valor que está muy lejos de los otros valores en el conjunto de datos. La definición exacta de valor atípico es algo subjetiva, a pesar de que se utilizan ciertas convenciones en distintos resúmenes de datos y diagramas (consultar "Percentiles y diagramas de caja" en la página 20). Ser un valor atípico en sí mismo no hace que un valor de los datos no sea válido o sea erróneo (como en el ejemplo anterior con Bill Gates). Aun así, los valores atípicos son a menudo el resultado de errores de datos, como la combinación de datos de diferentes unidades (kilómetros en lugar de metros) o las lecturas incorrectas de un sensor. Cuando los valores atípicos son el resultado de datos incorrectos, la media dará como resultado una estimación deficiente de la localización, mientras que la mediana seguirá siendo válida. Los valores atípicos deben identificarse y, por lo general, merecen una investigación más profunda.



Detección de anomalías

En contraste con el análisis normal de datos, donde los valores atípicos son a veces informativos y a veces molestos, en la *detección de anomalías* (*anomaly detection*) los puntos que nos interesan son los valores atípicos, y la mayor masa de datos sirve principalmente para definir la "normalidad" con la que se miden las anomalías.

La mediana no es la única estimación robusta de la localización. De hecho, la media truncada se usa habitualmente para evitar la influencia de valores atípicos. Por ejemplo, truncar el 10% inferior y superior (una opción frecuente) de los datos proporcionará protección contra valores atípicos en todos los conjuntos de datos, excepto en los más pequeños. La media truncada se puede considerar un compromiso entre la mediana y la media: es robusta a los valores extremos de los datos, pero utiliza más datos para calcular la estimación de la localización.



Otras métricas robustas para la localización

Los estadísticos han desarrollado una pléthora de otros estimadores de localización, principalmente con el objetivo de desarrollar un estimador más robusto que la media y también más eficiente (es decir, más capaz de

discernir pequeñas diferencias de localización entre conjuntos de datos). Si bien estos métodos son potencialmente útiles para conjuntos de datos pequeños, no es probable que proporcionen algún beneficio adicional para conjuntos con cantidades grandes de datos o incluso de tamaño moderado.

Ejemplo: estimaciones de localización de la población y tasas de homicidios

La tabla 1.2 muestra las primeras filas del conjunto de datos que contienen la población y las tasas de homicidios (en unidades de homicidios por cada 100 000 habitantes y por año) para cada estado de EE. UU. (censo de 2010).

Tabla 1.2 Algunas filas de data. Frame de la situación de la población y la tasa de homicidios por estados

Estado	Población	Homicidios	Abreviatura
1 Alabama	4 779 736	5.7	AL
2 Alaska	710 231	5.6	AK
3 Arizona	6 392 017	4.7	AZ
4 Arkansas	2 915 918	5.6	AR
5 California	37 253 956	4.4	CA
6 Colorado	5 029 196	2.8	CO
7 Connecticut	3 574 097	2.4	CT
8 Delaware	897 934	5.8	DE

Utilizamos *R* para calcular la media, la media truncada y la mediana de la población:

```
> state <- read.csv('state.csv')
> mean(state[['Population']])
[1] 6162876
> mean(state[['Population']], trim=0.1)
[1] 4783697
> median(state[['Population']])
[1] 4436370
```

Para calcular la media y la mediana con *Python*, podemos emplear los métodos `pandas` del marco de datos. La media truncada requiere la función `trim_mean` de `scipy.stats`:

```
state = pd.read_csv('state.csv')
state['Population'].mean()
trim_mean(state['Population'], 0.1)
state['Population'].median()
```

La media es mayor que la media truncada, que es mayor que la mediana.

Esto se debe a que la media truncada excluye los cinco estados más grandes y más pequeños (`trim=0.1` ignora el 10% de cada extremo). Si queremos calcular la tasa de homicidios promedio para el país, necesitamos usar una media o mediana ponderadas para dar cuenta

de las diferentes poblaciones de los estados. Dado que el software básico de *R* no tiene una función para la mediana truncada, necesitamos instalar el paquete `matrixStats`:

```
> weighted.mean(state[['Murder.Rate']], w=state[['Population']])
[1] 4.445834
> library('matrixStats')
> weightedMedian(state[['Murder.Rate']], w=state[['Population']])
[1] 4.4
```

Con NumPy podemos disponer de la media ponderada. Para la mediana ponderada, podemos usar el paquete especializado `wquantiles` (<https://pypi.org/project/wquantiles/>):

```
np.average(state['Murder.Rate'], weights=state['Population'])
wquantiles.median(state['Murder.Rate'], weights=state['Population'])
```

En este caso, la media ponderada y la mediana ponderada son aproximadamente iguales.

Ideas clave

- La métrica básica para la localización es la media, pero puede ser sensible a valores extremos (valores atípicos).
- Otras métricas (mediana, media truncada) son menos sensibles a valores atípicos y a distribuciones inusuales y, por lo tanto, son más robustas.

Lecturas complementarias

- El artículo de Wikipedia sobre tendencia central contiene un amplio debate sobre varias medidas de localización.
- El clásico *Exploratory Data Analysis* (Pearson) de John Tukey de 1977 todavía se lee bastante.

Estimación de la variabilidad

La localización es solo una dimensión para extraer el resumen de una característica. Una segunda dimensión, la *variabilidad* (*variability*), también conocida como *dispersión* (*dispersion*), mide el grado de agrupación o dispersión de los valores de los datos. En el corazón de la estadística se encuentra la variabilidad: hay que medirla, reducirla, distinguir la variabilidad aleatoria de la real, identificar las diversas fuentes de variabilidad real y tomar decisiones teniéndola en cuenta.

Términos clave de métricas de variabilidad

Desviaciones

Diferencias entre los valores observados y la estimación de la localización.

Sinónimos

errores, residuales

Varianza

Suma de los cuadrados de las desviaciones de la media al cuadrado y dividida por $n - 1$, donde n es el número de valores de datos.

Sinónimo

error cuadrático medio

Desviación estándar

Raíz cuadrada de la varianza.

Desviación media absoluta

Media de los valores absolutos de las desviaciones de la media.

Sinónimos

norma L1, norma Manhattan

Desviación absoluta mediana de la mediana

Mediana de los valores absolutos de las desviaciones de la mediana.

Rango

La diferencia entre el mayor y el menor valor de un conjunto de datos.

Estadísticos ordinales

Métricas basadas en los valores de datos ordenados de menor a mayor.

Sinónimo

rangos

Percentil

Valor tal que el P por ciento de los valores toma este valor o un valor inferior y para $(100 - P)$ el porcentaje toma este valor o un valor superior.

Sinónimo

cuantil

Rango intercuartil

Diferencia entre el percentil 75 y el percentil 25.

Sinónimo

IQR

Así como existen diferentes formas de medir la localización (media, mediana, etc.), también existen diferentes formas de medir la variabilidad.

Desviación estándar y estimaciones relacionadas

Las estimaciones de la variación más utilizadas se basan en las diferencias o *desviaciones* (*deviations*) entre la estimación de la localización y los datos observados. Para un conjunto de datos {1, 4, 4}, la media es 3 y la mediana es 4. Las desviaciones de la media son las diferencias: $1 - 3 = -2$, $4 - 3 = 1$, $4 - 3 = 1$. Estas desviaciones nos dicen lo dispersos están los datos en torno al valor central.

Una forma de medir la variabilidad es estimar un valor típico para estas desviaciones. El promedio de las desviaciones en sí no nos diría mucho: las desviaciones negativas compensan las positivas. De hecho, la suma de las desviaciones de la media es precisamente cero. En cambio, un enfoque sencillo consiste en extraer el promedio de los valores absolutos de las desviaciones de la media. En el ejemplo anterior, el valor absoluto de las desviaciones es {2 1 1} y su promedio es $(2 + 1 + 1) / 3 = 1.33$. A esta medida se conoce como *desviación media absoluta* (*mean absolute deviation*) y se calcula con la fórmula:

$$\text{Desviación media absoluta} = \frac{\sum_{i=1}^n |x_i - \bar{x}|}{n}$$

donde \bar{x} es la media muestral.

Las estimaciones de variabilidad más conocidas son la *varianza* (*variance*) y la *desviación estándar* (*standard deviation*), que se calculan a partir del cuadrado de las desviaciones. La varianza es un promedio del cuadrado de las desviaciones y la desviación estándar es la raíz cuadrada de la varianza:

$$\text{Variancia} = s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}$$

$$\text{Desviación estándar} = s = \sqrt{\text{Variancia}}$$

La desviación estándar es mucho más fácil de interpretar que la varianza, ya que está en la misma escala que los datos originales. Aun así, con su fórmula más complicada y menos intuitiva, podría parecer peculiar que en estadística se prefiera la desviación estándar a la desviación media absoluta. Debe su supremacía a la teoría estadística, ya que trabajar matemáticamente con valores al cuadrado es mucho más conveniente que con valores absolutos, especialmente en el caso de modelos estadísticos.

Grados de libertad, y ¿n o n - 1?

En los libros de estadística, siempre se discute por qué tenemos $n - 1$ en el denominador en la fórmula de la varianza, en lugar de n , lo que conduce al concepto

de grados de libertad (*degrees of freedom*). Esta distinción no es importante ya que n es generalmente lo suficientemente grande como para que no haya mucha diferencia si dividimos entre n o $n - 1$. Pero en caso de que sea de interés, lo explicamos a continuación. Se fundamenta en la premisa de que deseamos hacer estimaciones sobre una población, basándonos en una muestra.

Si usamos el denominador intuitivo de n en la fórmula de la varianza, subestimaremos el valor real de la varianza y la desviación estándar en la población. Esto se conoce como estimación sesgada (*biased*). Sin embargo, si dividimos por $n - 1$ en lugar de n , la varianza se convierte en una estimación no sesgada (*unbiased*).

Explicar completamente por qué el uso de n conduce a una estimación sesgada implica la noción de grados de libertad, que tiene en cuenta el número de restricciones al calcular una estimación. En este caso, hay $n - 1$ grados de libertad, ya que hay una restricción: la desviación estándar depende del cálculo de la media muestral. Para la mayoría de los problemas, los científicos de datos no necesitan preocuparse por los grados de libertad.

La varianza, la desviación estándar y la desviación absoluta mediana son robustas a valores atípicos y extremos (consultar "Estimación de medianas robustas" en la página 10, donde se desarrolla un debate sobre estimaciones robustas para la localización). La varianza y la desviación estándar son especialmente sensibles a los valores atípicos, ya que se basan en las desviaciones al cuadrado.

Una estimación robusta de la variabilidad es la desviación absoluta mediana de la mediana (*median absolute deviation from the median*) o MAD:

$$\text{Desviación absoluta mediana} = \text{Mediana}(|x_1 - m|, |x_2 - m|, \dots, |x_N - m|)$$

donde m es la mediana. Al igual que la mediana, la MAD no se ve influenciada por valores extremos. También es posible calcular la desviación estándar truncada análoga a la media truncada (consultar "Media" en la página 9).



La varianza, la desviación estándar, la desviación media absoluta y la desviación absoluta mediana no son estimaciones equivalentes, incluso en el caso de que los datos provengan de una distribución normal. De hecho, la desviación estándar es siempre mayor que la desviación absoluta media, que a su vez es mayor que la desviación absoluta mediana. A veces, la desviación absoluta mediana se multiplica por un factor de escala constante para adaptarla a la misma escala que la desviación estándar en el caso de una distribución normal. El factor que se utiliza normalmente de 1.4826 significa que el 50% de la distribución normal cae dentro del rango \pm MAD.

Estimación basada en percentiles

Un enfoque diferente para estimar la dispersión se centra en observar la distribución de los datos ordenados. Los estadísticos que tienen como base los datos ordenados (clasificados) se denominan *estadísticos de orden* (*order statistics*). La medida más elemental es el *rango* (*range*): la diferencia entre los números de mayor y menor valor. Es de utilidad conocer los valores mínimos y máximos en sí, además de práctico, para identificar valores atípicos, pero el rango es extremadamente sensible a los valores atípicos y no es muy útil como medida general de la dispersión de datos.

Para evitar la sensibilidad a los valores atípicos, podemos observar el rango de los datos después de eliminar valores de cada extremo. Formalmente, este tipo de estimaciones se basan en diferencias entre *percentiles* (*percentiles*). En un conjunto de datos, el percentil P es un valor tal que al menos el P por ciento de los valores toman este valor o un valor inferior y al menos $(100 - P)$ por ciento de los valores toman este valor o un valor superior. Por ejemplo, para encontrar el percentil 80, ordenamos los datos. Luego, comenzando por el valor más pequeño, continuamos hasta el 80% del recorrido para llegar al mayor valor. Hay que tener en cuenta que la mediana es lo mismo que el percentil 50. El percentil es esencialmente lo mismo que el *cuantil* (*quantile*), con los cuantiles referenciados por porcentajes (por lo que el cuantil 0.8 es lo mismo que el percentil 80).

Una medida habitual de la variabilidad es la diferencia entre el percentil 25 y el percentil 75, al que se llama *rango intercuartílico* (*interquartile range*) (o IQR). Veamos un sencillo ejemplo: {3,1,5,3,6,7,2,9}. Los ordenamos para obtener {1,2,3,3,5,6,7,9}. El percentil 25 está en 2.5 y el percentil 75 está en 6.5, por lo que el rango intercuartílico es $6.5 - 2.5 = 4$. El software puede tener enfoques ligeramente diferentes que producen diferentes respuestas (consultar el consejo que aparece más abajo). Normalmente, estas diferencias son mínimas.

Para conjuntos de datos muy grandes, calcular percentiles exactos puede ser muy costoso desde el punto de vista del cálculo, ya que requiere ordenar todos los valores de los datos. El aprendizaje automático y el software estadístico utilizan algoritmos especiales [Zhang-Wang, 2007] para obtener un percentil aproximado que se puede calcular con mucha rapidez y tiene garantizada una cierta precisión.



Percentil: definición precisa

Si tenemos un número par de datos (n es par), entonces el percentil es ambiguo según la definición anterior. De hecho, podríamos tomar cualquier valor entre los estadísticos de orden $x_{(j)}$ y $x_{(j+1)}$, donde j satisface:

$$100 * \frac{j}{n} \leq P < 100 * \frac{j + 1}{n}$$

Formalmente, el percentil es el promedio ponderado:

$$\text{Percentil } (P) = (1 - w)x_{(j)} + wx_{(j+1)}$$

para alguna ponderación w entre 0 y 1. El software estadístico tiene enfoques ligeramente diferentes para elegir w . De hecho, la función `quantile` de *R* ofrece nueve alternativas diferentes para calcular el cuantil. A excepción de los conjuntos de datos pequeños, normalmente no es necesario preocuparse por la forma precisa en que se calcula el percentil. En el momento de escribir estas líneas, `numpy.quantile` de *Python* solo admite un enfoque: la interpolación lineal.

Ejemplo: estimaciones de variabilidad de la población estatal

La tabla 1.3 (repetición de la tabla 1.2 para hacerlo más cómodo) muestra las primeras filas del conjunto de datos que contienen las tasas de población y de los homicidios para cada estado.

Tabla 1.3 Algunas filas de `data.frame` de la situación de la población y la tasa de homicidios por estados

	Estado	Población	Tasa de homicidios	Abreviatura
1	Alabama	4 779 736	5.7	AL
2	Alaska	710 231	5.6	AK
3	Arizona	6 392 017	4.7	AZ
4	Arkansas	2 915 918	5.6	AR
5	California	37 253 956	4.4	CA
6	Colorado	5 029 196	2.8	CO
7	Connecticut	3 574 097	2.4	CT
8	Delaware	897 934	5.8	DE

Utilizando las funciones integradas de *R* para la desviación estándar, el rango intercuartílico (IQR) y la desviación absoluta mediana de la mediana (MAD), podemos calcular las estimaciones de variabilidad para los datos de la población estatal:

```
> sd(state[['Population']])
[1] 6848235
> IQR(state[['Population']])
[1] 4847308
> mad(state[['Population']])
[1] 3849870
```

El marco de datos de pandas proporciona métodos para calcular la desviación estándar y los cuantiles. Usando los cuantiles, podemos determinar fácilmente el IQR. Para la MAD robusta, usamos la función `robust.scale.mad` del paquete `statsmodels`:

```
state['Population'].std()
state['Population'].quantile(0.75) - state['Population'].quantile(0.25)
robust.scale.mad(state['Population'])
```

La desviación estándar es casi dos veces mayor que la MAD (en R, por defecto, la escala de la MAD se ajusta para estar en la misma escala que la de la media). Este hecho no es sorprendente, ya que la desviación estándar es sensible a valores atípicos.

Ideas clave

- La varianza y la desviación estándar son los estadísticos de variabilidad más difundidos y de los que más se informa de manera rutinaria.
- Ambos son sensibles a los valores atípicos.
- Entre las métricas más robustas se encuentran la desviación absoluta media, la desviación absoluta mediana y los percentiles (cuantiles).

Lecturas complementarias

- El recurso de estadísticos en línea de David Lane tiene una sección sobre percentiles (<https://onlinestatbook.com/2/introduction/percentiles.html>).
- Kevin Davenport tiene una publicación interesante en R-Bloggers (<https://www.r-bloggers.com/2013/08/absolute-deviation-around-the-median/>) sobre las desviaciones de la mediana y sus propiedades robustas.

Exploración de la distribución de datos

Cada una de las estimaciones que tratamos aquí resume los datos en una sola cifra para describir la localización o la variabilidad de los datos. También es interesante explorar cómo se distribuyen los datos en general.

Términos clave de la exploración de la distribución

Diagrama de caja

Diagrama presentado por Tukey para visualizar de forma rápida la distribución de datos.

Sinónimo

diagrama de caja y bigotes

Tabla de frecuencias

Registro del recuento de valores de datos numéricos que caen en un conjunto de intervalos (contenedores).

Histograma

Diagrama de la tabla de frecuencias con los contenedores o intervalos en el eje x y el recuento (o proporción) en el eje y. Aunque los gráficos de barras son visualmente similares, no deben confundirse con los histogramas. Consultar “Exploración de datos binarios y categóricos” en la página 27 para obtener más información sobre las diferencias entre ambas presentaciones.

Diagrama de densidad

Versión suavizada del histograma, a menudo basada en una *estimación de la densidad del núcleo (kernel density estimate)*.

Percentiles y diagramas de caja

En "Estimación basada en percentiles" en la página 16, exploramos cómo se pueden utilizar los percentiles para medir la dispersión de los datos. Los percentiles también son útiles para extraer un resumen de toda la distribución. Es habitual informar los cuartiles (percentiles 25, 50 y 75) y los deciles (percentiles 10, 20,..., 90). Los percentiles son especialmente indicados para extraer el resumen de las *colas (tails)* (partes de los extremos del rango) de la distribución. La cultura popular ha acuñado el término de los *uno por ciento (one-percenters)* para referirse a las personas con una riqueza superior al percentil 99.

La tabla 1.4 muestra algunos percentiles de la tasa de homicidios por estados. En R, esta información la facilita la función `quantile`:

```
quantile(state[['Murder.Rate']], p=c(.05, .25, .5, .75, .95))
5%    25%    50%    75%    95%
1.600  2.425  4.000  5.550  6.510
```

En Python la proporciona el método `quantile` del marco de datos de pandas:

```
state['Murder.Rate'].quantile([0.05, 0.25, 0.5, 0.75, 0.95])
```

Tabla 1.4 Percentiles de tasa de homicidios por estados

5%	25%	50%	75%	95%
1.60	2.42	4.00	5.55	6.51

La mediana es de 4 homicidios por cada 100 000 habitantes, aunque hay bastante variabilidad: el percentil 5 es solo 1.6 y el percentil 95 es 6.51.

Los *diagramas de caja (boxplots)*, presentados por Tukey [Tukey, 1977], utilizan percentiles y permiten visualizar la distribución de datos de una forma rápida. La figura 1.2 muestra el diagrama de caja de la población por cada estado, que proporciona R:

```
boxplot(state[['Population']] / 1000000, ylab='Population (millions)')
```

pandas proporciona una serie de gráficos exploratorios básicos para el marco de datos. Uno de ellos es el diagrama de caja:

```
ax = (state['Population']/1_000_000).plot.box()  
ax.set_ylabel('Population (millions)')
```

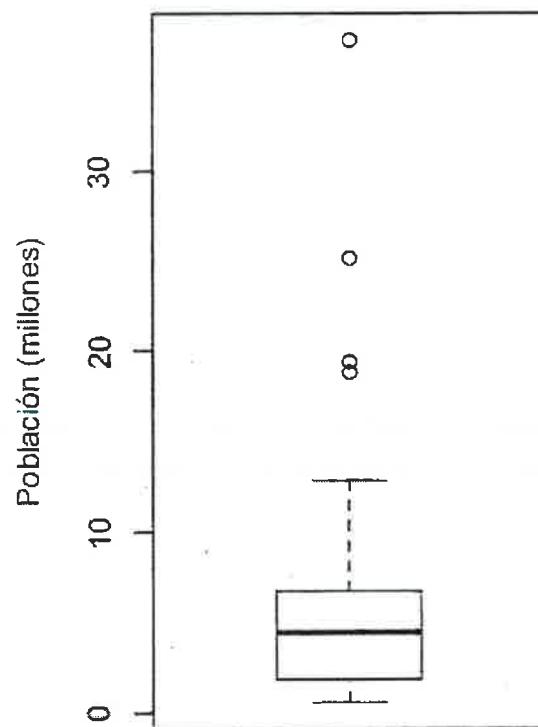


Figura 1.2 Diagrama de caja de la población por estados.

En este diagrama de caja, podemos ver de forma inmediata que la mediana de la población por estados es de alrededor de 5 millones, la mitad de los estados se encuentran entre aproximadamente 2 millones y 7 millones, y hay algunos valores atípicos de altos niveles de población. La parte superior e inferior del cuadro son los percentiles 75 y 25, respectivamente. La mediana se muestra mediante una línea horizontal dentro del cuadro. Las líneas discontinuas, denominadas *bigotes* (*whiskers*), se extienden desde las partes superior e inferior del cuadro para indicar el rango de la mayor parte de los datos. Hay muchas variaciones del diagrama de caja. Ver, por ejemplo, la documentación de la función `boxplot` de R [R-base, 2015]. Por defecto, la función R extiende los bigotes hasta el punto más alejado fuera de la caja, pero no van más allá de 1.5 veces el IQR. Matplotlib utiliza la misma implementación. Cualquier otro software puede usar una regla diferente.

Los datos que aparecen fuera de los bigotes se representan como puntos o círculos (a menudo considerados valores atípicos).

Tablas de frecuencias e histogramas

La tabla de frecuencias de una variable divide el rango de la variable en segmentos igualmente espaciados y nos dice cuántos valores caen dentro de cada segmento. La tabla 1.5 muestra la tabla de frecuencias de la población por cada estado calculada mediante *R*:

```
breaks <- seq(from=min(state[['Population']]),to=max(state[['Population']]), length=11)
pop_freq <- cut(state[['Population']], breaks=breaks, right=TRUE, include.lowest=TRUE)
table(pop_freq)
```

La función `pandas.cut` crea una serie que asigna los valores a los segmentos. Mediante el método `value_counts`, obtenemos la tabla de frecuencias:

```
binnedPopulation = pd.cut(state['Population'], 10)
binnedPopulation.value_counts()
```

Tabla 1.5 Tabla de frecuencias de la población por estados

Número de contenedor	Rango del contenedor	Recuento	Estados
1	563 626– 4 232 658	24	WY, VT, ND, AK, SD, DE, MT, RI, NH, ME, HI, ID, NE, WV, NM, NV, UT, KS, AR, MS, IA, CT, OK, OR
2	4 232 659– 7 901 691	14	KY, LA, SC, AL, CO, MN, WI, MD, MO, TN, AZ, IN, MA, WA
3	7 901 692– 11 570 724	6	VA, NJ, NC, GA, MI, OH
4	11 570 725– 15 239 757	2	PA, IL
5	15 239 758– 18 908 790	1	FL
6	18 908 791– 22 577 823	1	NY
7	22 577 824– 26,246,856	1	TX
8	26 246 857– 29 915 889	0	
9	29 915 890– 33 584 922	0	
10	33 584 923– 37 253 956	1	CA

El estado menos poblado es Wyoming, con 563 626 habitantes, y el más poblado es California, con 37 253 956 habitantes. Esto nos da un rango de $37\ 253\ 956 - 563\ 626 = 36\ 690\ 330$, que debemos dividir en contenedores de igual tamaño, digamos 10 contenedores. Con 10 contenedores del mismo tamaño, cada contenedor tendrá una anchura de 3 669 033, por lo que el primer contenedor incluirá desde 563 626 a 4 232 658. Por el contrario, el contenedor superior, desde 33 584 923 a 37 253 956, tiene un

solo estado: California. Los dos contenedores inmediatamente por debajo del de California están vacíos, hasta llegar a Texas. Es importante incluir los contenedores vacíos. El hecho de que no haya valores en esos contenedores constituye una información valiosa. También puede resultar conveniente experimentar con distintos tamaños de contenedores. Si son demasiado grandes, se pueden ocultar características importantes de la distribución. Si son demasiado pequeños, el resultado es demasiado granular y se pierde la capacidad de ver el panorama general.



Tanto las tablas de frecuencias como los percentiles, extraen el resumen de los datos mediante la creación de contenedores. En general, los cuartiles y deciles tendrán el mismo número de valores en cada contenedor (contenedores de igual número de valores), pero los tamaños de los contenedores serán diferentes. La tabla de frecuencias, por el contrario, tendrá diferente número de valores en los contenedores (contenedores de igual tamaño) y el tamaño de los contenedores será el mismo para todos.

El histograma es un modo de visualizar la tabla de frecuencias, con contenedores en el eje x y los valores de los datos en el eje y. En la figura 1.3, por ejemplo, el contenedor centrado en 10 millones ($1e + 07$) va de aproximadamente 8 millones a 12 millones, y hay seis estados en ese contenedor. Para crear mediante R el histograma correspondiente a la tabla 1.5, utilizamos la función `hist` con el argumento `breaks`:

```
hist(state[['Population']], breaks=breaks)
```

pandas soporta histogramas para marcos de datos con el método `DataFrame.plot.hist`. Utilizamos el argumento de palabra clave `bins` para definir el número de contenedores. Los diversos métodos de diagramas proporcionan como resultado un eje de objetos que permite mejorar el ajuste de la visualización mediante `Matplotlib`:

```
ax = (state['Population'] / 1_000_000).plot.hist(figsize=(4, 4))
ax.set_xlabel('Population (millions)')
```

El histograma se muestra en la figura 1.3. En general, los histogramas se representan gráficamente teniendo en cuenta que:

- Los contenedores vacíos se incluyen en el gráfico.
- Los contenedores tienen la misma anchura.
- El número de contenedores (o, de manera equivalente, el tamaño del contenedor) depende del usuario.
- Las barras son contiguas: no se muestran espacios vacíos entre las barras, a menos que haya un contenedor vacío.

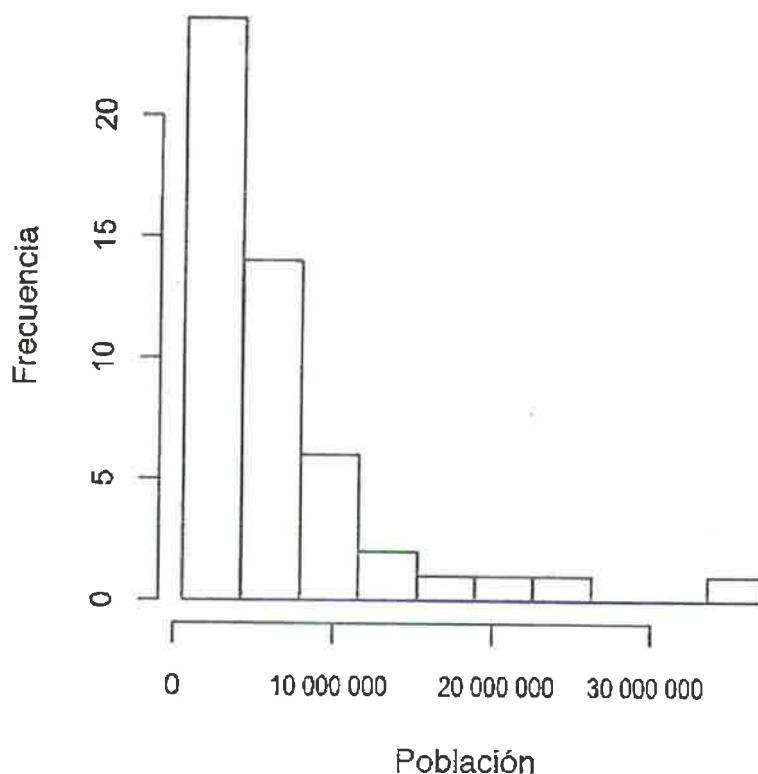


Figura 1.3 Histograma de población por estados.



Momentos de la distribución estadística

En teoría estadística, la localización y la variabilidad se conocen como el primer y segundo *momentos* (*moments*) de una distribución. Los momentos tercero y cuarto se denominan *asimetría* (*skewness*) y *curtosis* (*kurtosis*). La asimetría se refiere a si los datos están sesgados hacia valores mayores o menores, y la curtosis indica la propensión de los datos a tener valores extremos. Generalmente, no se utilizan métricas para medir la asimetría y la curtosis, sino que se descubren a través de presentaciones visuales como las figuras 1.2 y 1.3.

Diagrama y estimación de la curva de densidad

Relacionado con el histograma existe el diagrama de densidad, que muestra la distribución de los valores de los datos mediante una línea continua. Un diagrama de densidad se puede considerar como un histograma suavizado, aunque normalmente se calcula directamente a partir de los datos a través de una *estimación de la densidad del núcleo* (*kernel density estimate*) (ver [Duong, 2001], un breve tutorial). La figura 1.4 muestra la estimación de la densidad, que aparece superpuesta al histograma. En R, podemos calcular la estimación de la densidad utilizando la función `density`:

```
hist(state[['Murder.Rate']], freq=FALSE)
lines(density(state[['Murder.Rate']]), lwd=3, col='blue')
```

pandas proporciona el método `density` para crear el diagrama de densidad. Utilizamos el argumento `bw_method` para controlar la suavidad de la curva de densidad:

```
ax = state['Murder.Rate'].plot.hist(density=True, xlim=[0,12], bins=range(1,12))
state['Murder.Rate'].plot.density(ax=ax) ❶
ax.set_xlabel('Murder Rate (per 100,000)')
```

- ❶ Las funciones del diagrama a menudo adoptan opcionalmente el argumento de eje (`ax`), lo que hará que el diagrama se agregue al mismo gráfico.

Una distinción clave del histograma con diagrama de la figura 1.3 es la escala del eje y: el diagrama de densidad corresponde a la representación del histograma en porcentajes en lugar de hacerlo por el número de valores (esta opción la especificamos en R usando el argumento `freq=FALSE`). Hay que tener en cuenta que el área total por debajo de la curva de densidad es igual a 1 y, en lugar de aparecer el número de valores de los contenedores, calculamos el área que queda por debajo de la curva entre dos puntos cualesquiera en el eje x, que corresponde al porcentaje de la distribución que se encuentra entre esos dos puntos.

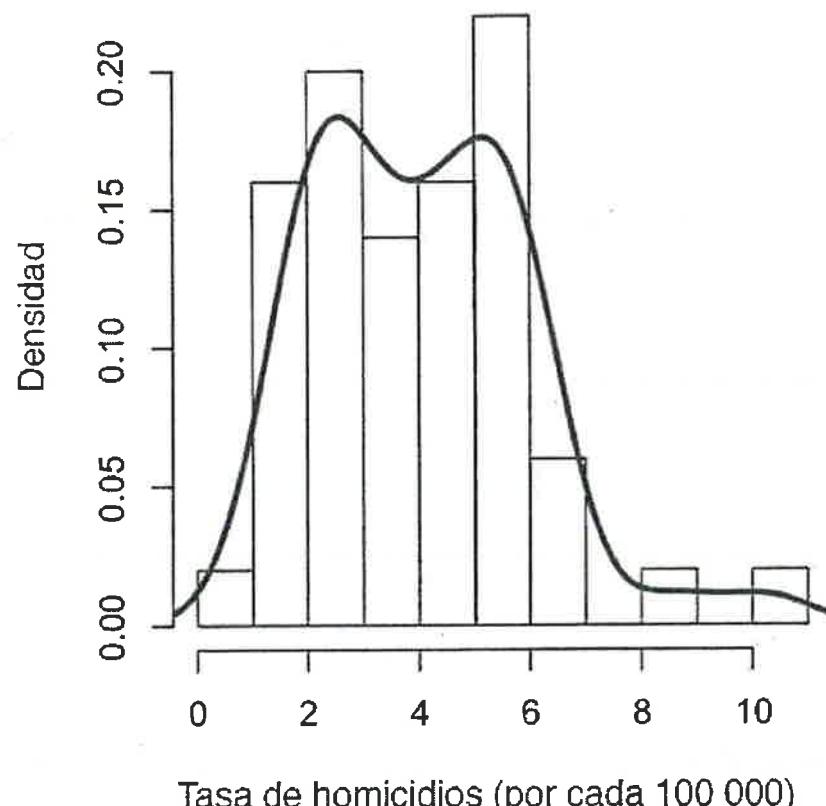


Figura 1.4 Densidad de las tasas de homicidios por estados.



Valoración de la densidad

La valoración de la densidad es un tema candente con una larga historia en la literatura estadística. De hecho, se han publicado más de 20 paquetes *R* que ofrecen funciones para la valoración de la densidad. [Deng-Wickham, 2011] proporciona una revisión completa de los paquetes *R*, con una recomendación particular para `ASH` o `KernSmooth`. Los métodos de valoración de la densidad en `pandas` y `scikit-learn` también proporcionan buenas aplicaciones. Para muchos problemas de ciencia de datos, no hay necesidad de preocuparse por los diversos tipos de estimaciones de la densidad, basta con utilizar las funciones básicas.

Ideas clave

- El histograma de frecuencias representa las frecuencias de los resultados (recuentos) en el eje y y los valores de la variable en el eje x. Proporciona de un vistazo una idea de la distribución de los datos.
- La tabla de frecuencias es una versión tabular de las frecuencias de los resultados que se encuentran en un histograma.
- Un diagrama de caja, con la parte superior e inferior de la caja en los percentiles 75 y 25, respectivamente, también da una idea rápida de la distribución de los datos. A menudo se utiliza en visualizaciones paralelas para comparar distribuciones.
- El diagrama de densidad es una versión suavizada del histograma. Requiere una función para estimar el diagrama basado en los datos (por supuesto, son posibles múltiples estimaciones).

Lecturas complementarias

- Un profesor de SUNY Oswego proporciona una guía paso a paso para crear un diagrama de caja (https://www.oswego.edu/~srp/stats/bp_con.htm).
- La valoración de la densidad en *R* se trata en el artículo del mismo nombre de Henry Deng y Hadley Wickham (<http://vita.had.co.nz/papers/density-estimation.pdf>).
- *R-Bloggers* tiene una publicación muy útil sobre histogramas con *R* (<https://www.r-bloggers.com/2012/12/basics-of-histograms/>), que incluye elementos de personalización, como es el agrupamiento (uso de `breaks()`).
- *R-Bloggers* también tiene una publicación similar sobre el diagrama de caja con *R* (<https://www.r-bloggers.com/2013/06/box-plot-with-r-tutorial/>).
- Matthew Conlen tiene una presentación interactiva (<https://mathisonian.github.io/kde/>) que demuestra el efecto de elegir diferentes núcleos y ancho de banda en las estimaciones de densidad del núcleo.

Exploración de datos binarios y categóricos

En el caso de los datos categóricos, las proporciones simples o porcentajes cuentan la historia de los datos.

Términos clave de la exploración de datos categóricos

Moda

Categoría o valor que ocurre con más frecuencia en un conjunto de datos.

Valor esperado

Cuando las categorías se pueden asociar con un valor numérico, el valor esperado proporciona un valor promedio basado en la probabilidad de ocurrencia de una categoría.

Gráficos de barras

Frecuencia o proporción de cada categoría representada en barras.

Gráficos en forma de tarta

Frecuencia o proporción de cada categoría representada en forma de cuña de un pastel.

Obtener el resumen de una variable binaria o una variable categórica con varias categorías es un asunto bastante fácil: sencillamente calculamos la proporción de 1 o las proporciones de las categorías importantes. Por ejemplo, la tabla 1.6 muestra el porcentaje de vuelos que han llegado con retraso a sus destinos como consecuencia de los retrasos en el aeropuerto de Dallas/Fort Worth en 2010. Los retrasos se clasifican como debidos a: factores bajo el control de la aerolínea, retrasos en el sistema de control del tráfico aéreo (CTA), el clima, la seguridad o las aeronaves que llegan tarde.

Tabla 1.6 Porcentaje de retrasos causados por el aeropuerto de Dallas/Fort Worth

Aerolínea	CTA	Clima	Seguridad	Llegadas
23.02	30.40	4.03	0.12	42.43

Los gráficos de barras, que se ven a menudo en la prensa de gran difusión, son una herramienta visual muy utilizada para mostrar una única variable categórica. Las categorías se enumeran en el eje x y las frecuencias o porcentajes, en el eje y. La figura 1.5 muestra los retrasos por año en el aeropuerto de Dallas/Fort Worth (DFW), y se genera con la función R `barplot`:

```
barplot(as.matrix(dfw) / 6, cex.axis=0.8, cex.names=0.7,
       xlab='Cause of delay', ylab='Count')
```

pandas también soporta gráficos de barras para marcos de datos:

```
ax = dfw.transpose().plot.bar(figsize=(4, 4), legend=False)
ax.set_xlabel('Cause of delay')
ax.set_ylabel('Count')
```

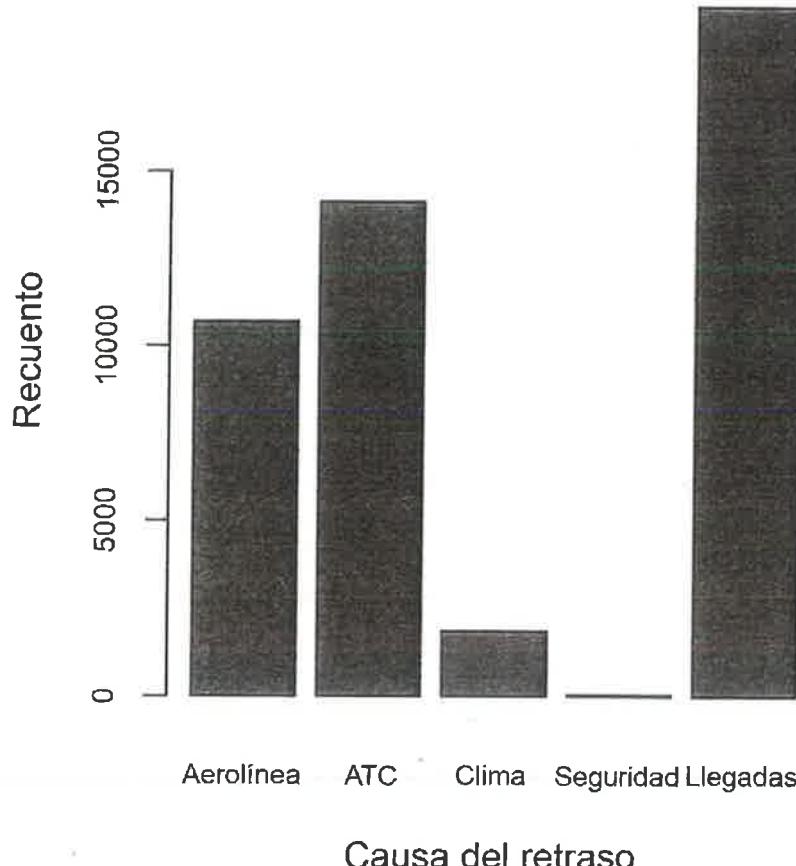


Figura 1.5 Gráfico de barras de retrasos de aerolíneas en el aeropuerto de DFW.

Hay que tener en cuenta que un gráfico de barras se parece a un histograma. En un gráfico de barras, el eje x representa diferentes categorías de una variable de tipo factor, mientras que, en un histograma, el eje x representa los valores de una sola variable en una escala numérica. En un histograma, las barras generalmente se muestran tocándose entre sí. Si hay espacios, estos indican valores ausentes en los datos. En un gráfico de barras, las barras se muestran separadas entre sí.

Los gráficos en forma de tarta son una alternativa a los gráficos de barras, aunque los estadísticos y los expertos en visualización de datos generalmente evitan los gráficos en forma de tarta por ser menos informativos visualmente (ver [Few, 2007]).



Datos numéricos como datos categóricos

En "Tablas de frecuencias e histogramas" en la página 22, analizamos las tablas de frecuencias basadas en la agrupación de datos. Esta acción convierte implícitamente los datos numéricos en un factor ordenado. En este sentido, los histogramas y los gráficos de barras son similares, excepto que las categorías del eje x en el gráfico de barras no están ordenadas. La conversión de datos numéricos en datos categóricos es un paso importante y se utiliza de forma generalizada en el análisis de datos, ya que reduce la complejidad (y el tamaño) de los datos. Esto ayuda a descubrir las relaciones entre las características, particularmente en las etapas iniciales del análisis.

Moda

La moda es el valor (o valores en caso de empate) que aparece con mayor frecuencia en los datos. Por ejemplo, la moda de la causa de los retrasos en el aeropuerto de Dallas/Fort Worth es "Inbound" (llegadas). Por citar otro ejemplo, en la mayor parte de Estados Unidos, la moda de la preferencia religiosa sería el cristianismo. La moda es un resumen estadístico simple para datos categóricos y, por lo general, no se utiliza para datos numéricos.

Valor esperado

Un tipo especial de datos categóricos son los datos en los que las categorías representan o pueden asignarse a valores discretos en la misma escala. Un vendedor de una nueva tecnología en la nube, por ejemplo, ofrece dos niveles de servicio, uno con un precio de 300 \$ al mes y otro de 50 \$ al mes. Este especialista en marketing ofrece seminarios web gratuitos para generar potenciales clientes, y la firma calcula que el 5% de los asistentes se inscribirá en el servicio de 300 \$, el 15% se inscribirá en el servicio de 50 \$ y el 80% no se inscribirá en nada. Estos datos se pueden resumir, con fines de tipo financiero, en un solo "valor esperado", que es una forma de media ponderada, en la que las ponderaciones son probabilidades.

El valor esperado se calcula de la siguiente manera:

1. Multiplicamos cada resultado por la probabilidad de que ocurra.
2. Sumamos esos valores.

En el ejemplo del servicio en la nube, el valor esperado de un asistente a un seminario web es, por lo tanto, de 22.50 \$ al mes, calculado de la siguiente manera:

$$EV = (0.05)(300) + (0.15)(50) + (0.80)(0) = 22.5$$

El valor esperado es realmente una forma de media ponderada: suma las ideas de expectativas futuras y las ponderaciones de probabilidad, a menudo basadas en juicios subjetivos. El valor esperado es un concepto fundamental en la valoración de empresas

y en los presupuestos de capital, por ejemplo, el valor esperado de los beneficios a los cinco años de una nueva adquisición, o los ahorros de costes esperados de un nuevo software de administración de pacientes de una clínica.

Probabilidad

Nos referimos anteriormente a la *probabilidad* (*probability*) de que ocurra un valor. La mayoría de las personas tiene una comprensión intuitiva de la probabilidad, y se encuentran con frecuencia con este concepto en los pronósticos meteorológicos (la probabilidad de que llueva) o en el análisis deportivo (la probabilidad de ganar). Los deportes y los juegos se expresan más a menudo como oportunidades, que se pueden convertir fácilmente en probabilidades (si las oportunidades de que un equipo gane son de 2 a 1, su probabilidad de ganar es $2 / (2 + 1) = 2/3$). Sin embargo, sorprendentemente, el concepto de probabilidad puede ser motivo de una profunda discusión filosófica a la hora de definirlo. Afortunadamente, aquí no necesitamos una definición matemática o filosófica formal. Para nuestros propósitos, la probabilidad de que suceda un evento es la proporción de veces que ocurriría si la situación pudiera repetirse una y otra vez, innumerables veces. La mayoría de las ocasiones se trata de una construcción imaginaria, pero es una comprensión funcional adecuada de la probabilidad.

Ideas clave

- Los datos categóricos generalmente se resumen en porcentajes y se pueden visualizar en un gráfico de barras.
- Las categorías pueden representar diferentes cosas (manzanas y naranjas, hombres y mujeres), niveles de una variable de tipo factor (bajo, medio y alto) o datos numéricos que se han agrupado.
- El valor esperado es la suma de los valores multiplicados por las probabilidades de que ocurran. A menudo se usa para totalizar los niveles de las variables de tipo factor.

Lecturas complementarias

Ningún curso de estadística está completo sin una lección sobre gráficos que inducen a error (https://en.wikipedia.org/wiki/Misleading_graph), que a menudo incluyen gráficos de barras y gráficos en forma de tarta.

Correlación

El análisis exploratorio de datos en muchos proyectos de modelado (ya sea en ciencia de datos o en investigación) implica examinar la correlación entre predictoras y entre

predictoras y una variable objetivo. Se dice que las variables X e Y (cada una con datos registrados) están correlacionadas positivamente si los valores altos de X acompañan a los valores altos de Y, y los valores bajos de X acompañan a los valores bajos de Y. Si los valores altos de X acompañan a los valores bajos de Y, y viceversa, las variables están correlacionadas negativamente.

Términos clave de la correlación

Coefficiente de correlación

Métrica que mide el grado en que las variables numéricas están asociadas entre sí (varía de -1 a $+1$).

Matriz de correlación

Tabla en la que las variables se muestran tanto en filas como en columnas, y los valores de las celdas son las correlaciones entre las variables.

Diagrama de dispersión

Diagrama en el que el eje x es el valor de una variable y el eje y es el valor de otra.

Consideremos estas dos variables, perfectamente correlacionadas en el sentido de que cada una va de menor a mayor:

$$v1: \{1, 2, 3\}$$

$$v2: \{4, 5, 6\}$$

La suma vectorial de los productos es $1 \cdot 4 + 2 \cdot 5 + 3 \cdot 6 = 32$. Ahora intentemos barajar una de ellas y volvamos a hacer el cálculo. La suma vectorial de productos nunca será mayor que 32. Por lo tanto, esta suma de productos podría utilizarse como métrica. Es decir, la suma observada de 32 podría compararse con muchas mezclas aleatorias (de hecho, esta idea se relaciona con una estimación basada en el remuestreo. Consultar "Prueba de permutación" en la página 97). Sin embargo, los valores generados por esta métrica no son tan importantes, excepto en lo referente a la distribución del remuestreo.

Más útil es una variante estandarizada: el *coeficiente de correlación* (*correlation coefficient*), que proporciona una estimación de la correlación entre dos variables que siempre tienen la misma escala. Para calcular el *coeficiente de correlación de Pearson* (*Pearson's correlation coefficient*), multiplicamos las desviaciones de la media de la variable 1 por las de la variable 2 y las dividimos por el producto de las desviaciones estándar:

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y}$$

Hay que tener en cuenta que dividimos por $n - 1$ en lugar de n . Consultar "Grados de libertad, y ¿ n o $n - 1$?" en la página 15 para ampliar detalles. El coeficiente de correlación siempre se encuentra entre +1 (correlación positiva perfecta) y -1 (correlación negativa perfecta). El 0 indica que no hay correlación.

Las variables pueden tener una asociación no lineal, en cuyo caso el coeficiente de correlación puede no ser una métrica útil. La relación entre las tasas impositivas y los ingresos recaudados es un ejemplo: a medida que las tasas impositivas aumentan desde cero, los ingresos recaudados también aumentan. Sin embargo, una vez que las tasas impositivas alcanzan un nivel alto y se acercan al 100%, la evasión fiscal aumenta y los ingresos fiscales en realidad disminuyen.

La tabla 1.7, denominada *matriz de correlación (correlation matrix)*, muestra la correlación entre las rentabilidades diarias de las acciones de telecomunicaciones desde julio de 2012 hasta junio de 2015. En la tabla, podemos ver que Verizon (VZ) y ATT (T) tienen la correlación más alta. Level 3 (LVLT), que es una empresa de infraestructuras, tiene la correlación más baja con las demás. Hay que tener en cuenta la diagonal de 1s (la correlación de una acción consigo misma es 1) y la redundancia de la información por encima y por debajo de la diagonal.

Tabla 1.7 Correlación entre la rentabilidad de las acciones de telecomunicaciones

	T	CTL	FTR	VZ	LVLT
T	1.000	0.475	0.328	0.678	0.279
CTL	0.475	1.000	0.420	0.417	0.287
FTR	0.328	0.420	1.000	0.287	0.260
VZ	0.678	0.417	0.287	1.000	0.242
LVLT	0.279	0.287	0.260	0.242	1.000

Por lo general, se presenta una tabla de correlaciones como la tabla 1.7 para mostrar visualmente la relación entre múltiples variables. La figura 1.6 muestra la correlación entre las rentabilidades diarias de los principales fondos cotizados en bolsa (ETF). En R, podemos crear esta figura fácilmente usando el paquete `corrplot`:

```
etfs <- sp500_px[row.names(sp500_px) > '2012-07-01',
                  sp500_sym[sp500_sym$sector == 'etf', 'symbol']]
library(corrplot)
corrplot(cor(etfs), method='ellipse')
```

Es posible crear el mismo gráfico con *Python*, pero no hay ninguna implementación en los paquetes convencionales. Sin embargo, la mayoría soporta la visualización de matrices de correlación mediante mapas de calor. El siguiente código lo realiza haciendo uso del paquete `seaborn.heat_map`. En el repositorio de código fuente adjunto, incluimos código *Python* para generar una visualización más completa:

```

etfs = sp500_px.loc[sp500_px.index > '2012-07-01',
                    sp500_sym[sp500_sym['sector']] == 'etf']['symbol']
sns.heatmap(etfs.corr(), vmin=-1, vmax=1,
            cmap=sns.diverging_palette(20, 220, as_cmap=True))

```

Los ETF para el S&P 500 (SPY) y el Índice Dow Jones (DIA) tienen una alta correlación. Del mismo modo, el QQQ y el XLK, compuestos en su mayoría por empresas de tecnología, están correlacionados positivamente. Los ETF defensivos, como los que rastrean los precios del oro (GLD), los precios del petróleo (USO) o la volatilidad del mercado (VXX), tienden a tener una correlación débil o negativa con los otros ETF. La orientación de la elipse indica si dos variables están correlacionadas positivamente (la elipse apunta hacia la parte superior derecha) o correlacionados negativamente (la elipse apunta hacia la parte superior izquierda). El sombreado y la anchura de la elipse indican la fuerza de la asociación: las elipses más delgadas y más oscuras corresponden a relaciones más fuertes (figura 1.6).

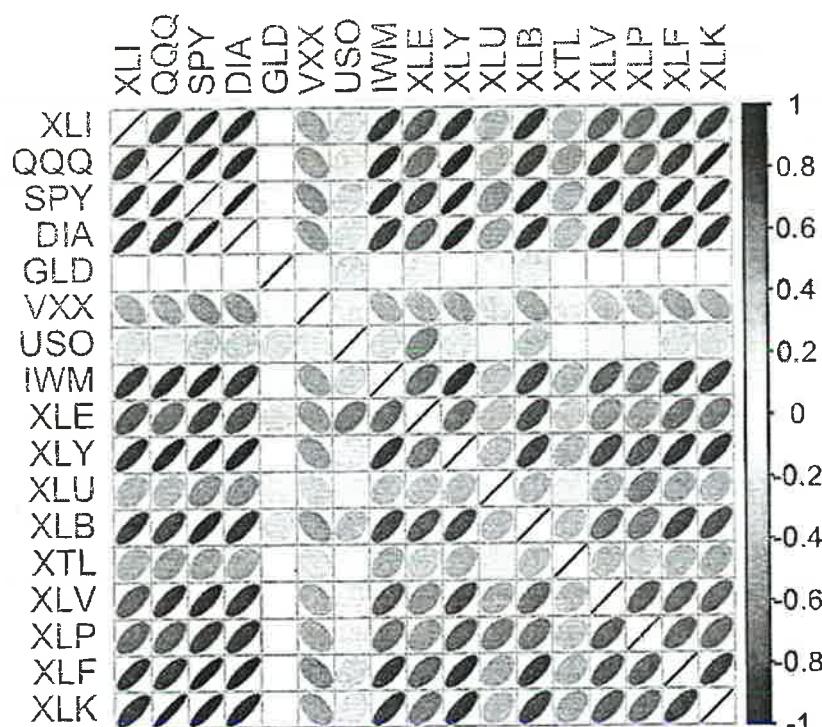


Figura 1.6 Correlación entre rentabilidades de ETF.

Al igual que la media y la desviación estándar, el coeficiente de correlación es sensible a valores extremos en los datos. Los paquetes de software ofrecen alternativas robustas al coeficiente de correlación clásico. Por ejemplo, el paquete robust de R (<https://cran.r-project.org/web/packages/robust/robust.pdf>) utiliza la función covRob para calcular una estimación robusta de correlación. Los métodos `sklearn.covariance` (<https://scikit-learn.org/stable/modules/classes.html#module-sklearn.covariance>) del módulo `scikit-learn` implementan diversos enfoques.



Otras estimaciones de la correlación

Los estadísticos propusieron hace mucho tiempo otros tipos de coeficientes de correlación, como *rho de Spearman* o *tau de Kendall*. Estos son coeficientes de correlación basados en el rango de los datos. Dado que trabajan con rangos en lugar de valores, estas estimaciones son robustas a valores atípicos y pueden manejar ciertos tipos de falta de linealidad. Sin embargo, los científicos de datos generalmente pueden ceñirse al coeficiente de correlación de Pearson y sus alternativas robustas para el análisis exploratorio. Las estimaciones basadas en rangos son atractivas principalmente para conjuntos de datos más pequeños y pruebas de hipótesis específicas.

Diagramas de dispersión

La forma tradicional de visualizar la relación entre dos variables de datos que hemos registrado es con un diagrama de dispersión. En el eje x se representa una variable y en el eje y otra, y cada punto del gráfico es un registro. En la figura 1.7 se muestra un gráfico de la correlación entre las rentabilidades diarias de ATT y Verizon. La visualización se consigue en *R* con el comando:

```
plot(telecom$T, telecom$VZ, xlab='ATT (T)', ylab='Verizon (VZ)')
```

El mismo gráfico se puede generar en *Python* usando el método de dispersión de pandas:

```
ax = telecom.plot.scatter(x='T', y='VZ', figsize=(4, 4), marker='$\u25ef$')
ax.set_xlabel('ATT (T)')
ax.set_ylabel('Verizon (VZ)')
ax.axhline(0, color='grey', lw=1)
ax.axvline(0, color='grey', lw=1)
```

Las rentabilidades tienen una relación positiva: mientras se agrupan alrededor del cero, la mayor parte de los días, las acciones suben o bajan conjuntamente (cuadrantes superior derecho e inferior izquierdo). Hay menos días en los que una acción baja significativamente mientras que la otra sube, o viceversa (cuadrantes inferior derecho y superior izquierdo).

Si bien el diagrama de la figura 1.7 muestra solo 754 puntos de datos, ya es obvio lo difícil que es identificar detalles en el centro del diagrama. Más adelante veremos cómo la adición de transparencia a los puntos, o el uso de agrupaciones hexagonales y diagramas de densidad, puede ayudar a encontrar una estructura complementaria en los datos.

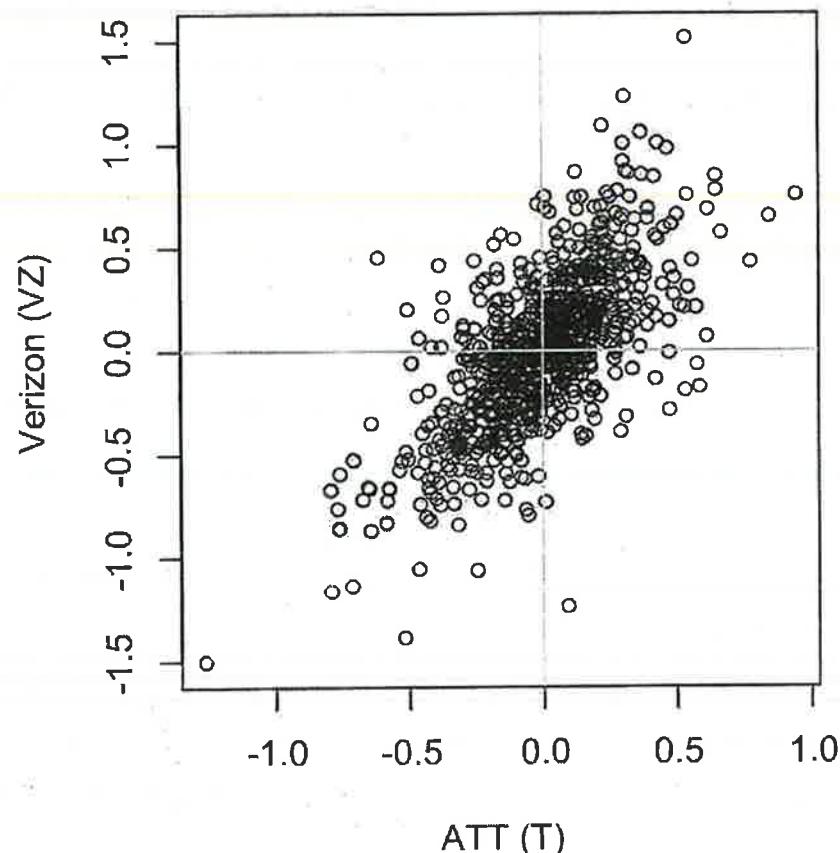


Figura 1.7 Diagrama de dispersión de la correlación entre las rentabilidades de ATT y Verizon.

Ideas clave

- El coeficiente de correlación mide el grado en que dos variables emparejadas (por ejemplo, la altura y el peso de los individuos) están asociadas entre sí.
- Cuando los valores altos de v1 acompañan a los valores altos de v2, v1 y v2 se asocian positivamente.
- Cuando los valores altos de v1 acompañan a los valores bajos de v2, v1 y v2 se asocian negativamente.
- El coeficiente de correlación es una métrica estandarizada, por lo que siempre varía de -1 (correlación negativa perfecta) a +1 (correlación positiva perfecta).
- Un coeficiente de correlación de cero indica que no hay correlación, pero hay que tener en cuenta que las disposiciones aleatorias de datos producirán valores tanto positivos como negativos para el coeficiente de correlación simplemente por casualidad.

Lecturas complementarias

Statistics, 4.^a ed., de David Freedman, Robert Pisani y Roger Purves (W. W. Norton, 2007) contiene una excelente discusión sobre correlación.

Exploración de dos o más variables

Los estimadores habituales como la media y la varianza analizan las variables una por una (*análisis univariable [univariate analysis]*). El análisis de correlación (consultar "Correlación" en la página 30) es un método importante que compara dos variables (*análisis bivariable [bivariate analysis]*). En esta sección, analizamos estimaciones y diagramas complementarios aplicados a más de dos variables (*análisis multivariante [multivariate analysis]*).

Términos clave de la exploración de dos o más variables

Tabla de contingencia

Registro del recuento entre dos o más variables categóricas.

Agrupación hexagonal

Diagrama de dos variables numéricas con los registros agrupados en hexágonos.

Diagrama de contorno

Diagrama que muestra la densidad de dos variables numéricas en forma de mapa topográfico.

Diagrama de violín

Similar a un diagrama de caja pero en el que se muestra la densidad estimada.

Al igual que el análisis univariable, el análisis bivariable implica tanto el cálculo de estadísticos de resumen como la generación de presentaciones visuales. El tipo apropiado de análisis bivariable o multivariable depende de la naturaleza de los datos: numéricos frente a categóricos.

Agrupación hexagonal y contornos (representación numérica frente a datos numéricos)

Los diagramas de dispersión son apropiados cuando hay un número relativamente pequeño de valores de datos. La gráfica de las rentabilidades de las acciones en la figura 1.7 solo involucra aproximadamente a 750 puntos. Para conjuntos de datos con cientos de miles o millones de registros, un diagrama de dispersión será demasiado denso, por lo que necesitamos visualizar la relación de un modo diferente. Para ilustrarlo,

consideremos el conjunto de datos `kc_tax`, que contiene los valores tasados por impuestos para propiedades residenciales en el condado de King, Washington. Para centrarnos en la parte principal de los datos, eliminamos las residencias muy caras y muy pequeñas o muy grandes utilizando la función `subset`:

```
kc_tax0 <- subset(kc_tax, TaxAssessedValue < 750000 &
                  SqFtTotLiving > 100 &
                  SqFtTotLiving < 3500)
nrow(kc_tax0)
432693
```

En pandas, filtramos el conjunto de datos de la siguiente manera:

```
kc_tax0 = kc_tax.loc[(kc_tax.TaxAssessedValue < 750000) &
                     (kc_tax.SqFtTotLiving > 100) &
                     (kc_tax.SqFtTotLiving < 3500), :]
kc_tax0.shape
(432693, 3)
```

La figura 1.8 es un diagrama de *agrupación hexagonal (hexagonal binning)* de la relación entre la superficie terminada en pies cuadrados y el valor tasado por impuestos para las viviendas en el condado de King. En lugar de trazar puntos, que aparecerían como una nube oscura monolítica, agrupamos los registros en contenedores hexagonales y rellenamos los hexágonos con un color que indica el número de registros en ese contenedor. En este gráfico, la relación positiva entre los pies cuadrados y el valor de tasación fiscal es clara. Una característica interesante es que se muestran ligeramente bandas adicionales por encima de la banda principal (la más oscura) en la parte inferior, que indica los hogares que tienen la misma superficie en pies cuadrados que los de la banda principal pero un valor fiscal más alto.

La figura 1.8 la ha generado el potente paquete de R `ggplot2`, desarrollado por Hadley Wickham [ggplot2]. `ggplot2` es una de las múltiples nuevas bibliotecas de software para el análisis visual exploratorio avanzado de datos. Consultar "Visualización de varias variables" en la página 43:

```
ggplot(kc_tax0, aes(x=SqFtTotLiving, y=TaxAssessedValue)) +
  stat_binhex(color='white') +
  theme_bw() +
  scale_fill_gradient(low='white', high='black') +
  labs(x='Finished Square Feet', y='Tax-Assessed Value')
```

En *Python*, se puede disponer fácilmente de los diagramas de agrupación hexagonal utilizando el método `hexbin` del marco de datos de pandas:

```
ax = kc_tax0.plot.hexbin(x='SqFtTotLiving', y='TaxAssessedValue',
                         gridsize=30, sharex=False, figsize=(5, 4))
ax.set_xlabel('Finished Square Feet')
ax.set_ylabel('Tax-Assessed Value')
```

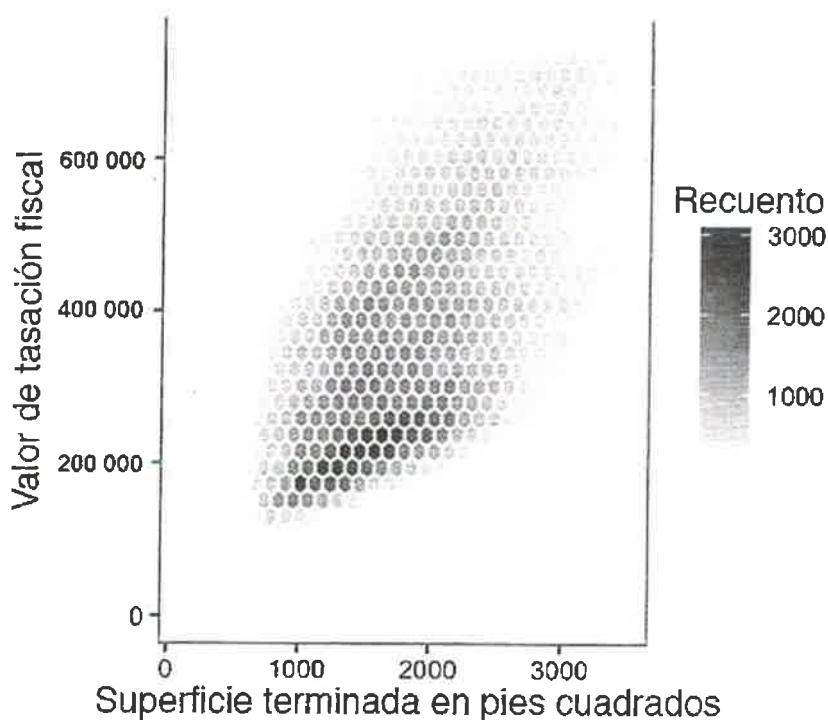


Figura 1.8 Agrupación hexagonal para el valor de tasación fiscal en relación con la superficie terminada en pies cuadrados.

La figura 1.9 utiliza contornos superpuestos a un diagrama de dispersión para visualizar la relación entre dos variables numéricas. Los contornos son esencialmente un mapa topográfico con dos variables. Cada banda de contorno representa una densidad específica de puntos, que aumenta a medida que nos acercamos a un "pico". Este gráfico muestra una historia similar a la de la figura 1.8: hay un pico secundario "al norte" del pico principal. Este cuadro también se ha creado mediante ggplot2 con la función incorporada geom_density2d:

```
ggplot(kc_tax0, aes(SqFtTotLiving, TaxAssessedValue)) +
  theme_bw() +
  geom_point(alpha=0.1) +
  geom_density2d(color='white') +
  labs(x='Finished Square Feet', y='Tax-Assessed Value')
```

La función seaborn kdeplot de Python crea la gráfica de contorno:

```
ax = sns.kdeplot(kc_tax0.SqFtTotLiving, kc_tax0.TaxAssessedValue, ax=ax)
ax.set_xlabel('Finished Square Feet')
ax.set_ylabel('Tax-Assessed Value')
```

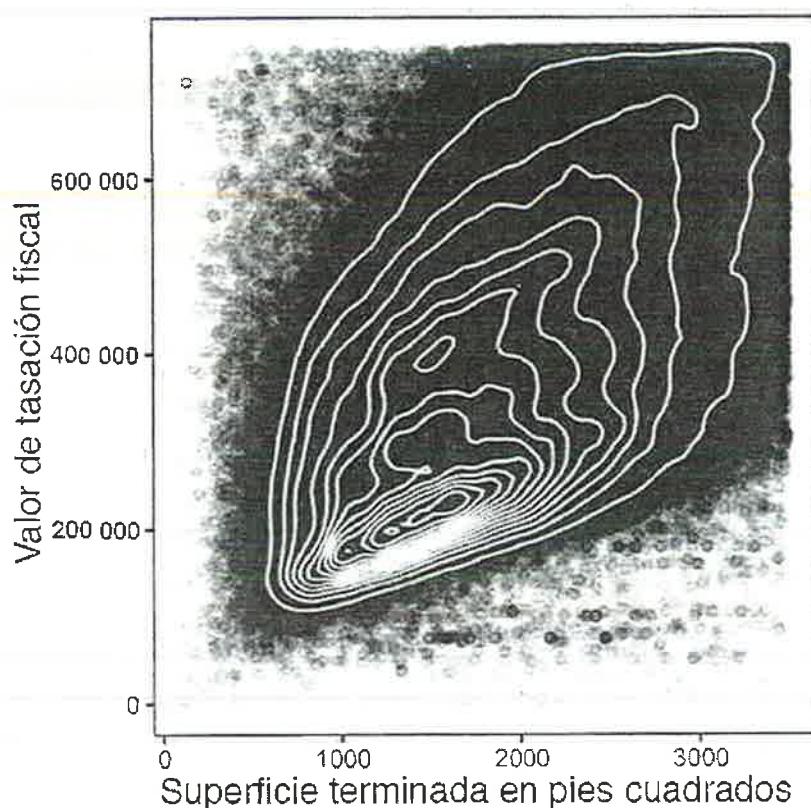


Figura 1.9 Gráfico de contorno para el valor de tasación fiscal en relación con la superficie terminada en pies cuadrados.

Se utilizan otros tipos de gráficos para mostrar la relación entre dos variables numéricas, incluidos los *mapas de calor* (*heat maps*). Los mapas de calor, la agrupación hexagonal y los gráficos de contorno proporcionan una representación visual bidimensional de la densidad. En este sentido, son por su naturaleza semejantes a los histogramas y diagramas de densidad.

Dos variables categóricas

Una forma conveniente de resumir dos variables categóricas es mediante una tabla de contingencia, una tabla de recuentos por categorías. La tabla 1.8 muestra la tabla de contingencia con la calificación de un préstamo personal y el resultado de ese préstamo. Esta información se ha extraído de los datos proporcionados por Lending Club, líder en el negocio de préstamos entre particulares. La calificación va desde A (la más alta) a G (la más baja). El resultado es: totalmente pagado, al corriente de pago, atrasado o cancelado (no se espera que se cobre el importe del préstamo). La tabla muestra el recuento y las filas de porcentajes. Los préstamos de alta calificación tienen un porcentaje muy bajo de retrasos en el pago/cancelaciones en comparación con los préstamos de baja calificación.

Tabla 1.8 Tabla de contingencia de las calificaciones y estados de los préstamos

Calificación	Cancelado	Al corriente de pago	Totalmente pagado	Atrasado	Total
A	1562	50 051	20 408	469	72 490
	0.022	0.690	0.282	0.006	0.161
B	5302	93 852	31 160	2056	132 370
	0.040	0.709	0.235	0.016	0.294
C	6023	88 928	23 147	2777	120 875
	0.050	0.736	0.191	0.023	0.268
D	5007	53 281	13 681	2308	74 277
	0.067	0.717	0.184	0.031	0.165
E	2842	24 639	5949	1374	34 804
	0.082	0.708	0.171	0.039	0.077
F	1526	8444	2328	606	12 904
	0.118	0.654	0.180	0.047	0.029
G	409	1990	643	199	3241
	0.126	0.614	0.198	0.061	0.007
Total	22 671	321 185	97 316	9789	450 961

En las tablas de contingencia solo se suelen ver recuentos, aunque también pueden incluir porcentajes totales y de cada columna. Las tablas dinámicas en Excel son quizás la herramienta más utilizada para crear tablas de contingencia. En *R*, con la función *CrossTable* del paquete *descr* podemos generar tablas de contingencia. Para crear la tabla 1.8 se ha utilizado el siguiente código:

```
library(descr)
x_tab <- CrossTable(lc_loans$grade, lc_loans$status,
prop.c=FALSE, prop.chisq=FALSE, prop.t=FALSE)
```

El método *pivot_table* crea las tablas dinámicas en *Python*. El argumento *aggfunc* nos permite obtener los recuentos. Calcular los porcentajes es un poco más complicado:

```
crosstab = lc_loans.pivot_table(index='grade', columns='status',
aggfunc=lambda x: len(x), margins=True) ❶

df = crosstab.loc['A':'G'].copy() ❷
df.loc[:, 'Charged Off':'Late'] = df.loc[:, 'Charged Off':'Late'].div(df['All'], axis=0) ❸
df['All'] = df['All'] / sum(df['All']) ❹
perc_crosstab = df
```

- ❶ El argumento de palabra clave *margins* agregará las sumas de filas y columnas.
- ❷ Creamos una copia de la tabla dinámica ignorando las sumas de las columnas.
- ❸ Dividimos las filas por la fila suma.
- ❹ Dividimos la columna 'All' por su suma.

Datos categóricos y numéricos

Los diagramas de caja (consultar "Percentiles y diagramas de caja" en la página 20) son una forma sencilla de comparar visualmente las distribuciones de una variable numérica agrupada de acuerdo con una variable categórica. Por ejemplo, es posible que deseemos comparar cómo varía el porcentaje de retrasos en los vuelos entre las aerolíneas. La figura 1.10 muestra el porcentaje de vuelos que se retrasaron en un mes en el que el retraso lo controlaban las aerolíneas:

```
boxplot(pct_carrier_delay ~ airline, data=airline_stats, ylim=c(0, 50))
```

El método `boxplot` de pandas toma el argumento `by` que divide el conjunto de datos en grupos y crea los diagramas de caja individuales:

```
ax = airline_stats.boxplot(by='airline', column='pct_carrier_delay')
ax.set_xlabel('')
ax.set_ylabel('Daily % of Delayed Flights')
plt.suptitle('')
```

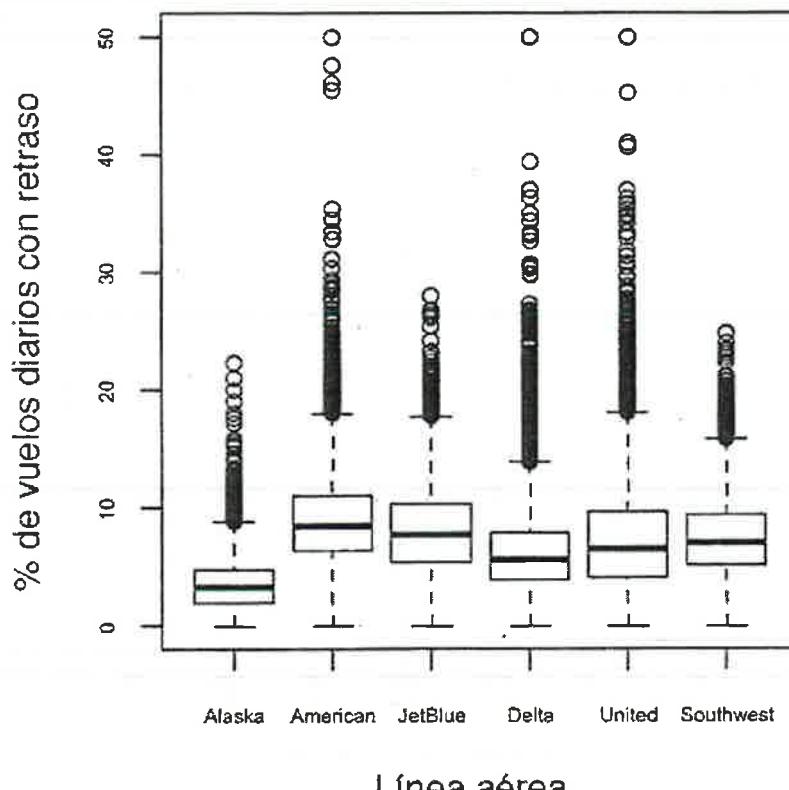


Figura 1.10 Diagrama de caja del porcentaje de retrasos por aerolínea.

Alaska se destaca por tener la menor cantidad de retrasos, mientras que American tiene la mayor cantidad de retrasos: el cuartil inferior de American está más alto que el cuartil superior de Alaska.

Estadística práctica para ciencia de datos con R y Python

El *diagrama de violín* (*violin plot*), introducido por [Hintze-Nelson, 1998], es una mejora del diagrama de caja y representa la estimación de la densidad, con la densidad asociada al eje y. Se obtiene la imagen espectral del diagrama de densidad, volcando a continuación ambas imágenes. La forma resultante se rellena, creando una imagen que se asemeja a un violín. La ventaja de un diagrama de violín es que puede mostrar matices en la distribución que no son perceptibles en un diagrama de caja. Por otro lado, el diagrama de caja muestra más claramente los valores atípicos de los datos. En `ggplot2`, la función `geom_violin` se puede usar para crear un diagrama de violín de la siguiente manera:

```
ggplot(data=airline_stats, aes(airline, pct_carrier_delay)) + ylim(0, 50) +
  geom_violin() +
  labs(x="", y='Daily % of Delayed Flights')
```

Los diagramas de violín están disponibles con el método `violinplot` del paquete `seaborn`:

```
ax = sns.violinplot(airline_stats.airline, airline_stats.pct_carrier_delay, inner='quartile',
                     color='white')
ax.set_xlabel('')
ax.set_ylabel('Daily % of Delayed Flights')
```

El diagrama correspondiente se muestra en la figura 1.11. El diagrama del violín muestra una concentración en la distribución cercana a cero para Alaska y, en menor medida, para Delta. Este fenómeno no es tan obvio en el diagrama de caja. Podemos combinar el diagrama de violín con el diagrama de caja agregando `geom_boxplot` al diagrama (aunque su funcionamiento mejora cuando se presenta en color).

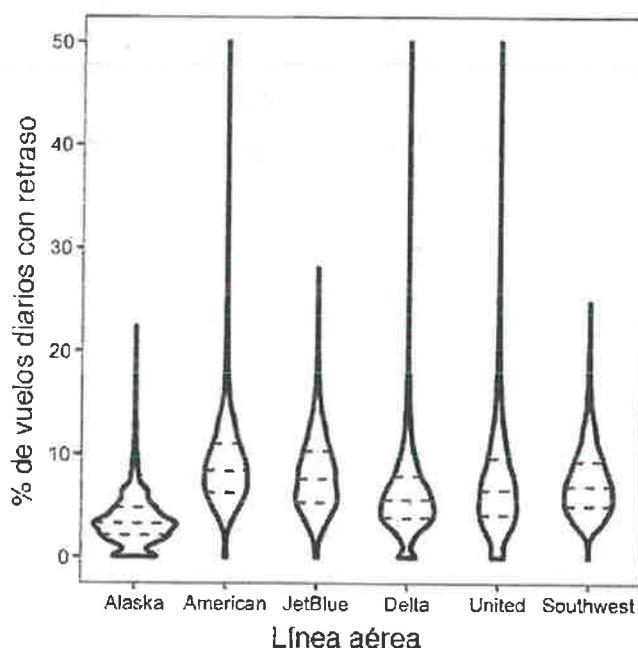


Figura 1.11 Diagrama de violín del porcentaje de retrasos por aerolínea.

Visualización de varias variables

Los tipos de gráficos que se utilizan para comparar dos variables (diagramas de dispersión, agrupación hexagonal y diagramas de caja) se extienden fácilmente a más variables mediante la noción de *acondicionamiento* (*conditioning*). Como ejemplo, veamos la anterior figura 1.8, que muestra la relación entre las superficies en pies cuadrados de las viviendas terminadas y sus valores de tasación fiscal. Observamos que parece haber un grupo de viviendas que tienen un valor fiscal más alto por pie cuadrado. Profundizando, la figura 1.12 explica el efecto de la localización al representar los datos para un conjunto de códigos postales. Ahora el panorama es mucho más claro: el valor fiscal es mucho más alto en algunos códigos postales (98105, 98126) que en otros (98108, 98188). Esta disparidad da lugar a los conglomerados observados en la figura 1.8.

Creamos la figura 1.12 usando `ggplot2` y el concepto de *facetas* (*facets*) o de una variable condicionante (en este caso, el código postal):

```
ggplot(subset(kc_tax0, ZipCode %in% c(98188, 98105, 98108, 98126)),  
       aes(x=SqFtTotLiving, y=TaxAssessedValue)) +  
  stat_binhex(color='white') +  
  theme_bw() +  
  scale_fill_gradient(low='white', high='blue') +  
  labs(x='Finished Square Feet', y='Tax-Assessed Value') +  
  facet_wrap('ZipCode') ❶
```

- ❶ Utilizamos las funciones `facet_wrap` y `facet_grid` de `ggplot` para especificar la variable condicionante.

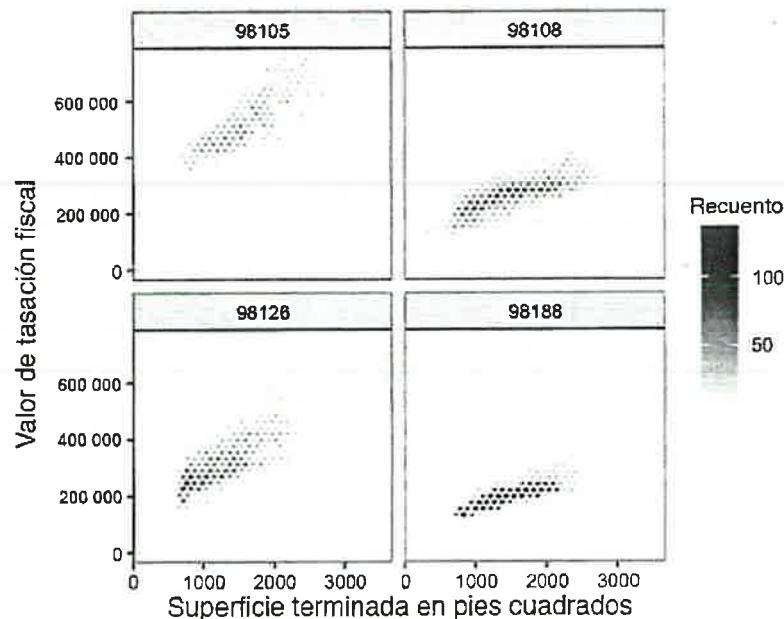


Figura 1.12 Valor de tasación fiscal en relación con la superficie terminada en pies cuadrados por código postal.

Estadística práctica para ciencia de datos con R y Python

La mayoría de los paquetes de *Python* utilizan `Matplotlib` para sus visualizaciones. Si bien en principio es posible crear gráficos facetados usando `Matplotlib`, el código puede complicarse. Afortunadamente, es posible crear gráficos de forma sencilla mediante `seaborn`:

```
zip_codes = [98188, 98105, 98108, 98126]
kc_tax_zip = kc_tax0.loc[kc_tax0.ZipCode.isin(zip_codes),:]
kc_tax_zip

def hexbin(x, y, color, **kwargs):
    cmap = sns.light_palette(color, as_cmap=True)
    plt.hexbin(x, y, gridsize=25, cmap=cmap, **kwargs)

g = sns.FacetGrid(kc_tax_zip, col='ZipCode', col_wrap=2) ❶
g.map(hexbin, 'SqFtTotLiving', 'TaxAssessedValue',
      extent=[0, 3500, 0, 700000]) ❷
g.set_axis_labels('Finished Square Feet', 'Tax-Assessed Value')
g.set_titles('Zip code {col_name:.0f}')
```

- ❶ Utilizamos los argumentos `col` y `row` para especificar las variables condicionantes. Para una sola variable condicionante, utilizamos `col` junto con `col_wrap` para empaquetar los gráficos facetados en varias filas.
- ❷ El método `map` llama a la función `hexbin` con los subconjuntos para los diferentes códigos postales del conjunto de datos original. `extent` define los límites de los ejes x e y.

El concepto de variables condicionantes en un sistema gráfico lo iniciaron los *gráficos Trellis* (*Trellis graphics*), desarrollados por Rick Becker y Bill Cleveland entre otros, en los Bell Labs [Trellis-Graphics]. Esta idea se ha extendido a varios sistemas gráficos modernos, como son los paquetes `lattice` [`lattice`] y `ggplot2` de *R* y los módulos `seaborn` [`seaborn`] y `Bokeh` [`bokeh`] de *Python*. Las variables condicionantes también son parte integral de las plataformas de inteligencia empresarial como Tableau y Spotfire. Con la llegada de equipos con una gran potencia informática, las plataformas de visualización modernas han ido mucho más allá de lo que fueron los humildes comienzos del análisis exploratorio de datos. Sin embargo, los conceptos y herramientas clave desarrolladas hace medio siglo (por ejemplo, los sencillos diagramas de caja) todavía constituyen la base de estos sistemas.

Ideas clave

- La agrupación hexagonal y los gráficos de contorno son herramientas útiles que permiten el examen gráfico de dos variables numéricas a la vez, sin que nos abrumen las grandes cantidades de datos.
- Las tablas de contingencia son la herramienta estándar para observar los recuentos de dos variables categóricas.
- Los diagramas de caja y los diagramas de violín nos permiten representar una variable numérica frente a una variable categórica.

Lecturas complementarias

- *Modern Data Science with R* de Benjamin Baumer, Daniel Kaplan y Nicholas Horton (Chapman & Hall/CRC Press, 2017) tiene una excelente presentación: "a grammar for graphics" (las "gg" de ggplot).
- *ggplot2: Elegant Graphics for Data Analysis* de Hadley Wickham (Springer, 2009) es un excelente recurso del creador de ggplot2.
- Josef Fruehwald tiene un tutorial web sobre ggplot2 (<https://www.ling.upenn.edu/~joseff/avml2012/>).

Resumen

El análisis exploratorio de datos (Exploratory Data Analysis [EDA]), iniciado por John Tukey, sentó las bases para desarrollar el campo de la ciencia de datos. La idea clave de EDA es que el primer paso y el más importante en cualquier proyecto basado en datos es la *observación de los datos* (*look at the data*). Al resumir y visualizar los datos, podemos desarrollar una valiosa intuición y conseguir comprender los proyectos.

Este capítulo ha revisado conceptos que van desde métricas sencillas, como son las estimaciones de localización y variabilidad, hasta completas presentaciones visuales que exploran las relaciones entre múltiples variables, como se muestra en la figura 1.12. El variado conjunto de herramientas y técnicas que está desarrollando la comunidad de código abierto, combinado con la expresividad de los lenguajes *R* y *Python*, han creado una gran cantidad de formas de explorar y analizar datos. El análisis exploratorio debería ser la piedra angular de cualquier proyecto de ciencia de datos.

CAPÍTULO 2

Distribuciones de datos y muestreo

Un error muy extendido sostiene que la era de los macrodatos (big data) va a suponer el fin de la necesidad de hacer muestreos. De hecho, la proliferación de datos de diferente calidad y relevancia refuerza la necesidad del muestreo como herramienta para trabajar de manera eficiente con una serie de datos y minimizar el sesgo. Incluso en un proyecto de big data, los modelos predictivos generalmente se desarrollan y se ponen a prueba mediante muestras. Las muestras también se utilizan en pruebas de diferentes tipos (por ejemplo, comparar el efecto de los diseños de páginas web en el número de clics).

La figura 2.1 muestra un esquema que fundamenta los conceptos que discutiremos en este capítulo: distribuciones de datos y muestreo. El lado izquierdo representa una población que, en estadística, se supone que sigue una distribución subyacente pero *desconocida (unknown)*. Lo único de lo que disponemos son los datos de la *muestra (sample)* y su distribución empírica, que aparecen en el lado derecho. Para pasar del lado izquierdo al lado derecho, se utiliza un procedimiento de *muestreo (sampling)* (representado por una flecha). La estadística tradicional se ha centrado principalmente en el lado izquierdo, utilizando teorías basadas en supuestos sólidos sobre la población. La estadística moderna se ha movido hacia el lado derecho, donde tales suposiciones no son necesarias.

En general, los científicos de datos no deben preocuparse por la naturaleza teórica del lado izquierdo y, en cambio, deben centrarse en los procedimientos de muestreo y los datos disponibles. Hay algunas excepciones notables. A veces, los datos se generan a partir de un proceso físico que se puede modelar. El ejemplo más sencillo es el de lanzar una moneda: sigue una distribución binomial. Se puede elaborar eficazmente un modelo de cualquier situación de la vida real que siga una distribución binomial (comprar o no comprar, que haya fraude o no, hacer clic o no hacer clic) con una moneda (con la probabilidad modificada para que salga cara, por supuesto). En estos casos, podemos obtener información adicional utilizando nuestro conocimiento de la población.

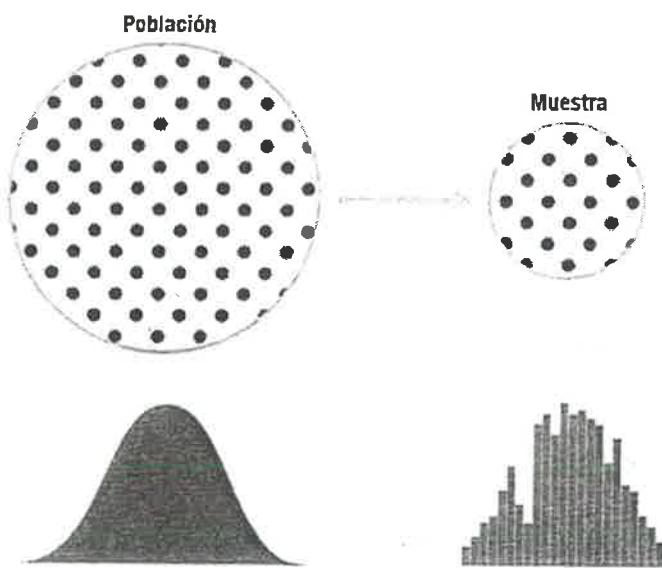


Figura 2.1 La población frente a la muestra.

Muestreo aleatorio y sesgo de la muestra

Una *muestra* (*sample*) es un subconjunto de datos de un conjunto más grande. Los estadísticos llaman *población* (*population*) a este conjunto de datos más grande. Una población en estadística no es lo mismo que en biología: es un conjunto de datos grande y definido (pero a veces teórico o imaginario).

El *muestreo aleatorio* (*random sampling*) es un proceso en el que cada miembro disponible de la población que se muestrea tiene la misma probabilidad de ser elegido para la muestra en cada extracción. La muestra resultante se denomina *muestra aleatoria simple* (*simple random sample*). El muestreo se puede hacer *con reposición* (*with replacement*), en el que las observaciones se vuelven a colocar en la población después de cada extracción para una posible futura nueva selección. O se puede hacer *sin reposición* (*without replacement*), en cuyo caso las observaciones, una vez seleccionadas, no están disponibles para futuras extracciones.

La calidad de los datos a menudo es más importante que la cantidad cuando se hace una estimación o se elabora un modelo basado en una muestra. La calidad de los datos en la ciencia de datos implica integridad, coherencia del formato, limpieza y precisión de los puntos de datos individuales. La estadística incorpora la noción de *representatividad* (*representativeness*).

Términos clave del muestreo aleatorio

Muestra

Subconjunto de un conjunto de datos más grande.

Población

El conjunto de datos más grande o la idea de un conjunto de datos.

$N(n)$

Tamaño de la población (muestra).

Muestreo aleatorio

Extracción de elementos de una muestra al azar.

Muestreo estratificado

División de la población en estratos y muestreo aleatorio de cada estrato.

Estrato (pl., estratos)

Subgrupo homogéneo de una población con características comunes.

Muestra aleatoria simple

Muestra que resulta de un muestreo aleatorio de una población sin estratificar.

Sesgo

Error sistemático.

Sesgo de la muestra

Muestra que describe erróneamente a la población.

El ejemplo clásico es la encuesta del *Literary Digest* de 1936 que predijo una victoria de Alf Landon sobre Franklin Roosevelt. El *Literary Digest*, uno de los principales diarios, encuestó a sus suscriptores además de hacerlo a los individuos que figuraban en otras listas, llegando a un total de más de 10 millones de personas, y predijo una victoria aplastante de Landon. George Gallup, fundador de Gallup Poll, realizó encuestas quincenales a solo 2000 personas y predijo acertadamente una victoria de Roosevelt. La diferencia radica en la selección de los encuestados.

Literary Digest apostó por la cantidad, prestando poca atención al método de selección. Terminaron encuestando a personas con un estatus socioeconómico relativamente alto (sus propios suscriptores, más aquellos que, en virtud de poseer lo que se consideraban lujos como eran los teléfonos y automóviles, aparecían en las listas de los especialistas en marketing). El resultado fue el sesgo de la muestra (*sample bias*), es decir, la muestra era, de manera significativa y no aleatoria, diferente de la población más grande a la que se pretendía representar. El término *no aleatorio (nonrandom)* es importante: casi ninguna muestra, incluidas las aleatorias, será exactamente representativa de la población. El sesgo ocurre cuando la diferencia es significativa y se puede esperar que continúe para otras muestras extraídas del mismo modo que la primera.



Sesgo del muestreo de autoselección

Las reseñas de restaurantes, hoteles, cafés, etc., que leemos en sitios de redes sociales, como Yelp, son propensas a estar sesgadas porque las personas que las envían no se seleccionan al azar, más bien, ellos mismos han tomado la iniciativa de escribir. Esto conduce a un sesgo de autoselección: las personas motivadas para escribir reseñas pueden haber tenido malas experiencias, pueden estar asociadas con el establecimiento o simplemente pueden ser un tipo de persona diferente de las que no escriben reseñas. Hay que tener en cuenta que, si bien las muestras de autoselección pueden ser indicadores poco confiables del verdadero estado de las cosas, pueden ser más confiables simplemente comparando un establecimiento con otro similar, el mismo sesgo de autoselección podría aplicarse a cada uno de ellos.

Sesgo

El sesgo estadístico se refiere a errores de medición o muestreo que son sistemáticos y se producen por el proceso de medición o muestreo. Debe hacerse una distinción importante entre los errores debidos al azar y los errores debidos al sesgo. Consideremos el proceso físico de un arma que disparamos a un objetivo. No golpeará en pleno centro del objetivo todo el tiempo, ni siquiera mucho tiempo. Un proceso no sesgado puede producir un error, pero es aleatorio y no tiende claramente a producirse en ninguna dirección (ver la figura 2.2). Los resultados que se muestran en la figura 2.3 muestran un proceso sesgado: a pesar de que hay un error aleatorio en las direcciones x e y, también hay un sesgo. Los disparos tienden a impactar en el cuadrante superior derecho.

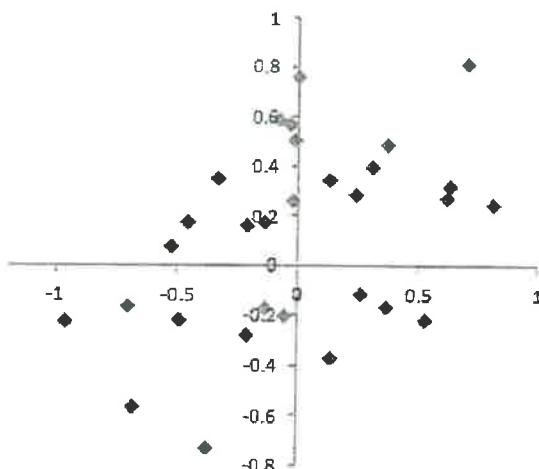


Figura 2.2 Diagrama de dispersión de los disparos de un arma con puntería no sesgada.

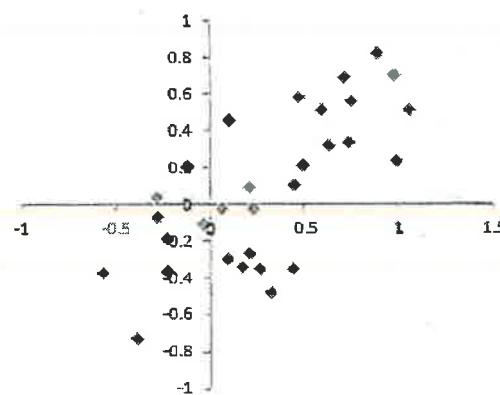


Figura 2.3 Diagrama de dispersión de los disparos de un arma con puntería sesgada.

El sesgo se presenta en diferentes formas y puede ser observable o no. Cuando un resultado sugiere sesgo (por ejemplo, en referencia a un parámetro o a valores reales), a menudo es un indicador de que se ha especificado incorrectamente un modelo estadístico o de aprendizaje automático, o que se ha omitido una variable importante.

Selección aleatoria

Para evitar el problema del sesgo de la muestra que llevó al *Literary Digest* a pronosticar la victoria de Landon sobre Roosevelt, George Gallup (que aparece en la figura 2.4) optó por elegir métodos con una base más científica para lograr una muestra que fuera representativa del electorado votante de EE. UU. En la actualidad existen diversos métodos para lograr la representatividad, pero en el centro de todos ellos se encuentra el *muestreo aleatorio* (*random sampling*).



Figura 2.4 George Gallup, catapultado a la fama por el fracaso de los "macrodatos (big data)" del *Literary Digest*.

El muestreo aleatorio no siempre es fácil. La definición adecuada de una población accesible es clave. Supongamos que queremos generar un perfil representativo de los clientes y necesitamos realizar una encuesta piloto a los mismos. La encuesta debe ser representativa, pero es muy laboriosa.

Estadística práctica para ciencia de datos con R y Python

En primer lugar, necesitamos definir quién es el cliente. Podríamos seleccionar todos los registros de clientes donde el importe de la compra > 0 . ¿Incluimos a los clientes antiguos? ¿Incluimos las devoluciones? ¿Las compras de prueba internas? ¿Revendedores? ¿Agentes de facturación que son clientes?

A continuación, necesitamos especificar un procedimiento de muestreo. Podría ser "seleccionar 100 clientes al azar". Cuando se trata de una muestra de flujo (por ejemplo, transacciones de clientes en tiempo real o visitantes de la web), las consideraciones de tiempo pueden ser importantes (por ejemplo, un visitante de la web a las 10 p.m. durante un fin de semana).

En el *muestreo estratificado (stratified sampling)*, la población se divide en *estratos (strata)* y se toman muestras aleatorias de cada estrato. Los encuestadores políticos pueden buscar conocer las preferencias electorales de blancos, negros e hispanos. Una muestra aleatoria simple tomada de la población seleccionaría a muy pocos negros e hispanos, por lo que esos estratos podrían sobreponerse en el muestreo estratificado para generar tamaños de muestra equivalentes.

Tamaño frente a calidad: ¿cuándo importa el tamaño?

En la era del big data, a veces resulta sorprendente que cuanto más pequeño, mejor. El tiempo y el esfuerzo dedicados al muestreo aleatorio no solo reducen el sesgo, sino que también permite una mayor atención a la exploración y calidad de los datos. Por ejemplo, los datos que faltan y los valores atípicos pueden contener información útil. Puede resultar prohibitivamente caro rastrear valores perdidos o evaluar valores atípicos en millones de registros, pero hacerlo en una muestra de varios miles de registros puede ser factible. El trazado de *datos* y la inspección manual se atascan si hay demasiados datos.

Entonces, ¿cuándo *son (are)* necesarias grandes cantidades de datos?

El escenario clásico para la estimación de big data es cuando la cantidad de datos no solo es grande sino que también los datos son escasos. Consideremos las consultas de búsqueda recibidas por Google, donde las columnas son términos, las filas son consultas de búsqueda individuales y los valores de las celdas son 0 o 1 dependiendo de si una consulta contiene un término. El objetivo es determinar el mejor destino de búsqueda previsto para una consulta determinada. Hay más de 150 000 palabras en inglés y Google procesa más de un billón de consultas al año. Esto produce una matriz enorme, la gran mayoría de cuyas entradas son "0".

Este es un verdadero problema de big data: solo cuando se acumulan cantidades tan enormes de datos, se pueden obtener resultados de búsqueda efectivos para la mayoría de las consultas. Y cuantos más datos se acumulen, mejores serán los resultados. Para los términos de búsqueda populares, esto no es un problema: se pueden encontrar datos efectivos con bastante rapidez para un puñado de temas extremadamente populares que son tendencia en un momento particular. El valor real de la moderna tecnología de búsqueda radica en la capacidad de responder con resultados detallados y útiles para una gran variedad de consultas de búsqueda, incluidas aquellas que ocurren con una frecuencia, digamos, de solo una en un millón.

Consideremos la búsqueda de la frase "Ricky Ricardo y Caperucita Roja". En los primeros días de Internet, esta consulta probablemente habría arrojado resultados sobre el líder de la banda Ricky Ricardo, el programa de televisión *I Love Lucy* en el que apareció ese personaje y el cuento infantil *Caperucita roja* (*Little Red Riding Hood*): Ambos elementos, de forma individual, habrían tenido muchos registros refiriéndose a ellos, pero la combinación habría tenido muy pocos. Posteriormente, ahora que se han acumulado billones de consultas de búsqueda, esta consulta devuelve el episodio exacto de *I Love Lucy* en el que Ricky narra, de manera dramática, la historia de Caperucita Roja a su hijo pequeño en una mezcla cómica de inglés y español.

Hay que tener en cuenta que la cantidad de registros *pertinentes* (*pertinent*) reales, aquellos en los que aparece esta consulta de búsqueda exacta, o algo muy similar, (junto con la información sobre el enlace en el que los usuarios finalmente hicieron clic), puede ser que solo necesiten ser miles para ser efectivos. Sin embargo, se necesitan muchos billones de puntos de datos para obtener los registros pertinentes (y el muestreo aleatorio, por supuesto, no ayudará). Consultar también "Distribuciones de cola larga" en la página 73.

Media muestral frente a media poblacional

El símbolo \bar{x} (pronunciado "barra x") se usa para representar la media de la muestra de una población, mientras que μ se usa para representar la media de una población. ¿Por qué hacer esta distinción? Se observa la información sobre las muestras y a menudo se infiere la información sobre poblaciones grandes a partir de muestras más pequeñas. A los estadísticos les gusta mantener las dos cosas separadas en lo que se refiere a la simbología.

Ideas clave

- Incluso en la era de los macrodatos (big data), el muestreo aleatorio sigue siendo una flecha importante en el carcaj del científico de datos.
- El sesgo ocurre cuando las mediciones u observaciones son erróneas de forma sistemática porque no son representativas de la población completa.
- La calidad de los datos suele ser más importante que la cantidad de los mismos, y el muestreo aleatorio puede reducir el sesgo y facilitar la mejora de la calidad que, de otro modo, sería prohibitivamente costoso.

Lecturas complementarias

- Se puede encontrar una revisión recomendable de los procedimientos de muestreo en el capítulo de Ronald Flicker "Sampling Methods for Online Surveys" en *The SAGE Handbook of Online Research Methods*, 2.^a ed., editado por Nigel G. Fielding, Raymond M. Lee y Grant Blank (Publicaciones SAGE, 2016). Este capítulo incluye una revisión de las modificaciones al muestreo aleatorio que se utilizan a menudo por razones prácticas de coste o viabilidad.

La historia del fracaso de la encuesta de *Literary Digest* se puede encontrar en el sitio web de Capital Century (<http://www.capitalcentury.com/1935.html>).

Sesgo de selección

Parafraseando a Yogi Berra: si no sabemos lo que estamos buscando, busquemos lo suficiente y lo encontraremos.

El sesgo de selección se refiere a la práctica de elegir datos de forma selectiva, consciente o inconscientemente, de manera que lleve a una conclusión engañosa o pasajera.

Términos clave del sesgo de selección

Sesgo de selección

Sesgo resultante de la forma en que se seleccionan las observaciones.

Búsqueda de datos

Amplia búsqueda de datos en busca de algo interesante.

Efecto de búsqueda masiva

Sesgo o no reproducibilidad resultante del modelado de datos repetidos, o del modelado de datos con un gran número de variables predictoras.

Si especificamos una hipótesis y realizamos un experimento bien diseñado para probarla, podemos tener una gran confianza en la conclusión. Sin embargo, esto no suele ser lo que ocurre. A menudo, uno mira los datos disponibles y trata de discernir patrones. Pero ¿son reales los patrones? ¿O son simplemente el producto de la *búsqueda de datos* (*data snooping*), es decir, una búsqueda exhaustiva de los datos hasta que surge algo interesante? Hay un dicho entre los estadísticos: "Si torturas los datos el tiempo suficiente, tarde o temprano confesarán".

La diferencia entre un fenómeno que verificamos cuando probamos una hipótesis mediante un *experimento* y un fenómeno que descubrimos al examinar los datos disponibles se puede aclarar con el siguiente experimento mental.

Imaginemos que alguien nos dice que podemos lanzar una moneda y hacer que salga cara en 10 lanzamientos consecutivos. Nosotros lo desafiamos (el equivalente a un experimento) y esta persona procede a lanzar la moneda 10 veces, y todos los lanzamientos salen cara. Claramente, a este individuo le atribuimos un talento especial: la probabilidad de que 10 lanzamientos de moneda arrojen cara por casualidad es de 1 entre 1000.

Ahora imaginemos que el locutor de un estadio deportivo pide a las 20 000 personas presentes que lancen una moneda 10 veces y que informen a un acomodador si obtienen 10 caras seguidas. La probabilidad de que *alguien* (*somebody*) en el estadio obtenga 10 caras es extremadamente alta (más del 99%; es 1 menos la probabilidad de que nadie obtenga 10 caras).

Claramente, seleccionar después del hecho a la persona (o personas) que obtienen 10 caras en el estadio no indica que tengan ningún talento especial, lo más probable es que sea suerte.

Dado que la revisión repetida de grandes conjuntos de datos es una propuesta de valor clave en la ciencia de datos, el sesgo de selección es algo de lo que hay que preocuparse. Una forma de sesgo de selección que preocupa especialmente a los científicos de datos es lo que John Elder (fundador de Elder Research, una respetada consultora de minería de datos) llama el *efecto de búsqueda masiva (vast search effect)*. Si ejecutamos repetidamente diferentes modelos y hacemos diferentes preguntas con un gran conjunto de datos, seguramente encontraremos algo interesante. Pero ¿el resultado que hallamos es realmente algo interesante, o es algo casual que se sale de lo normal?

Podemos protegernos de esta posibilidad mediante el uso de un conjunto de reserva y, a veces, de más de un conjunto de reserva, contra el cual validar el resultado. Elder también aboga por el uso de lo que él llama *mezcla de objetivos (target shuffling)* (esencialmente una prueba de permutación) para probar la validez de las asociaciones predictivas que sugiere un modelo de minería de datos.

Las formas típicas del sesgo de selección estadístico, además del efecto de búsqueda masiva, incluyen muestreo no aleatorio (consultar “Muestreo aleatorio y sesgo de la muestra” en la página 48), selección de datos, selección de intervalos de tiempo que acentúan un efecto estadístico particular y la detención de un experimento cuando los resultados parecen “interesantes”.

Regresión a la media

La *regresión a la media (regression to the mean)* se refiere a un fenómeno que involucra mediciones sucesivas de una variable dada: las observaciones extremas tienden a ser seguidas por otras más centrales. Dar un enfoque y un significado especiales a los valores extremos puede conducir a una forma de sesgo de selección.

Los fanáticos de los deportes están familiarizados con el fenómeno del “novato del año, caída en el segundo año”. Entre los atletas que comienzan su carrera en una temporada determinada (la clase de los novatos), siempre hay uno que se desenvuelve mejor que el resto. Generalmente, este “novato del año” no lo hace tan bien en su segundo año. ¿Por qué no?

En casi todos los deportes importantes, al menos en los que se juegan con pelota o disco, hay dos elementos que influyen en el rendimiento general:

- Habilidad
- Suerte

La regresión a la media es consecuencia de una forma particular de sesgo de selección. Cuando seleccionamos al novato con la mejor actuación, es probable que la habilidad y la

buenas suertes contribuyan. En su siguiente temporada, la habilidad seguirá ahí, pero muy a menudo la suerte no estará, por lo que su rendimiento disminuirá, retrocederá. El fenómeno lo identificó por primera vez Francis Galton en 1886 [Galton, 1886], quien escribió sobre él en relación con las tendencias genéticas. Por ejemplo, los hijos de hombres extremadamente altos tienden a no ser tan altos como sus padres (ver figura 2.5).

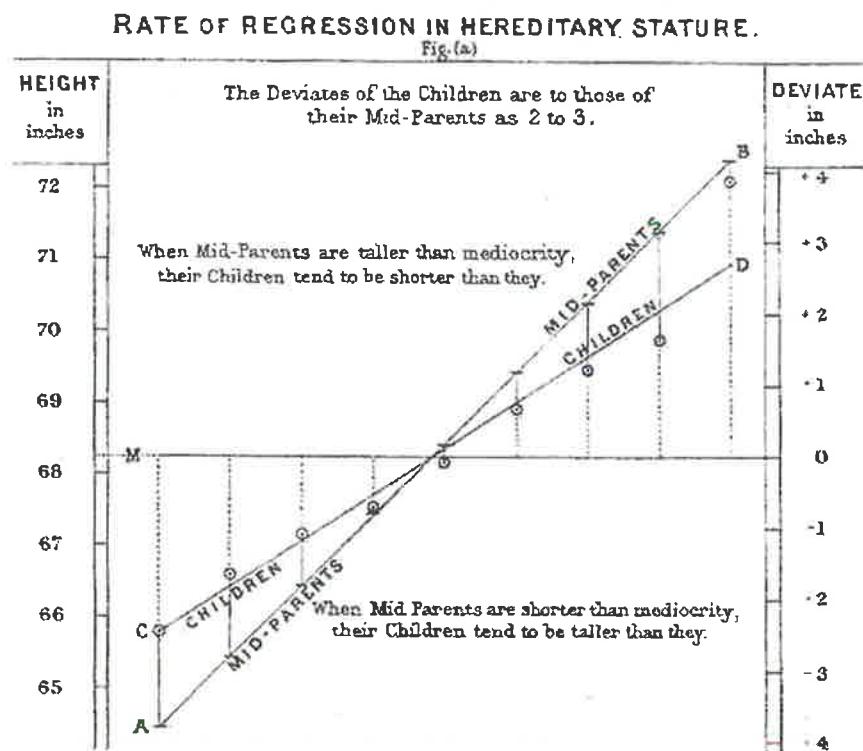


Figura 2.5 Estudio de Galton que identificó el fenómeno de regresión a la media.



La regresión a la media, que significa “volver”, es distinta del método de modelado estadístico de regresión lineal, en el que se estima una relación lineal entre las variables predictoras y la variable de resultado.

Ideas clave

- Especificar una hipótesis y luego recopilar datos siguiendo los principios de aleatorización y muestreo aleatorio asegura que no haya sesgos.
- Todas las demás formas de análisis de datos corren el riesgo de introducir sesgo resultante del proceso de recopilación/análisis de datos (ejecución repetida de modelos en la minería de datos, espionaje de datos en la investigación y selección posterior a los hechos de eventos interesantes).

Lecturas complementarias

- El artículo de Christopher J. Pannucci y Edwin G. Wilkins ("Identifying and Avoiding Bias in Research") en (sorprendentemente no es una revista de estadística) *Plastic and Reconstructive Surgery* (agosto de 2010) contiene una excelente revisión de varios tipos de sesgos que pueden producirse en la investigación, incluido el sesgo de selección.
- El artículo de Michael Harris ("Fooled by Randomness Through Selection Bias") proporciona una revisión interesante de las consideraciones del sesgo de selección en los esquemas de negociación del mercado de valores, desde la perspectiva de los operadores.

Distribución muestral del estadístico

El término *distribución muestral* (*sampling distribution*) de un estadístico se refiere a la distribución del estadístico de una muestra sobre muchas muestras extraídas de la misma población. Gran parte de la estadística clásica se ocupa de hacer inferencias de muestras (pequeñas) a poblaciones (muy grandes).

Términos clave de la distribución muestral

Estadístico muestral

Métrica calculada para una muestra de datos extraída de una población más grande.

Distribución de datos

Distribución de la frecuencia de valores individuales en un conjunto de datos.

Distribución muestral

Distribución de la frecuencia de un *estadístico muestral* (*sample statistic*) en muchas muestras o muestras repetidas.

Teorema del límite central

Tendencia de la distribución muestral a adoptar una forma normal a medida que aumenta el tamaño de la muestra.

Error estándar

La variabilidad (desviación estándar) del *estadístico* (*statistic*) muestral de una muestra en muchas muestras (no debe confundirse con la *desviación estándar* (*standard deviation*), que por sí misma se refiere a la variabilidad de los valores de los datos individuales).

Normalmente, una muestra se extrae con el objetivo de medir algo (con el *estadístico muestral* [*sample statistic*]) o modelar algo (con un modelo estadístico o de aprendizaje automático). Dado que nuestra estimación o modelo se basa en una muestra, podría dar lugar a error. Sería diferente si tuviéramos que extraer una muestra diferente. Por lo tanto, estamos interesados

Estadística práctica para ciencia de datos con R y Python

en cuánto de diferente podría ser: una preocupación clave es la *variabilidad muestral* (*sampling variability*). Si tuviéramos muchos datos, podríamos extraer muestras adicionales y observar la distribución del estadístico muestral directamente.

Por lo general, calcularemos la estimación o el modelo utilizando todos los datos que estén fácilmente disponibles, por lo que la extracción de muestras adicionales de la población no será una opción fácilmente realizable.



Es importante distinguir entre la distribución de los puntos de datos individuales, conocida como *distribución de datos* (*data distribution*), y la distribución de un estadístico muestral, conocida como *distribución muestral* (*sampling distribution*).

Es probable que la distribución de un estadístico muestral, como la media, sea más regular y adopte la forma de campana que la distribución de los datos en sí. Cuanto mayor sea la muestra en la que se fundamenta el estadístico, más cierta es esta afirmación. Además, cuanto mayor sea la muestra, más estrecha será la distribución del estadístico muestral.

Esto se ilustra en un ejemplo en el que se utilizan los ingresos anuales de los solicitantes de préstamos para LendingClub (consultar "Un pequeño ejemplo: pronóstico del incumplimiento de préstamos" en la página 239 para obtener una descripción de los datos). Tomemos tres muestras de estos datos: una muestra de 1000 valores, una muestra de 1000 medias de 5 valores y una muestra de 1000 medias de 20 valores. A continuación, representamos un histograma de cada muestra para generar la figura 2.6.

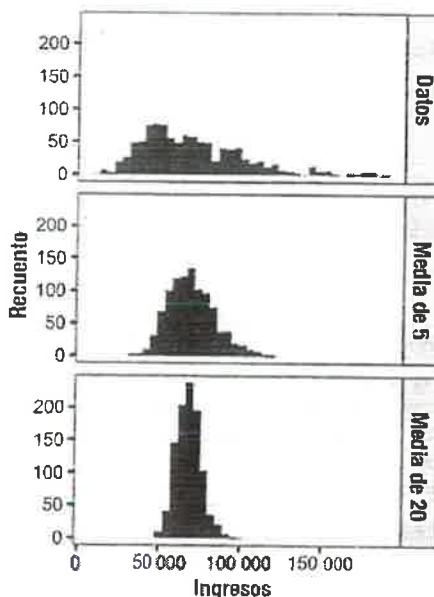


Figura 2.6 Histograma de ingresos anuales de 1000 solicitantes de préstamos (arriba); a continuación, 1000 medias para $n = 5$ solicitantes (centro), y, finalmente, 1000 medias para $n = 20$ solicitantes (abajo).

El histograma de los valores de los datos individuales está ampliamente distribuido y sesgado hacia valores más altos, como es de esperar con los datos de ingresos. Los histogramas de las medias de 5 y 20 son cada vez más compactos y tienen la forma de campana. A continuación, se detalla el código de *R* para generar estos histogramas, usando el paquete de visualización `ggplot2`:

```
library(ggplot2)
# take a simple random sample
samp_data <- data.frame(income=sample(loans_income, 1000),
                         type='data_dist')
# take a sample of means of 5 values
samp_mean_05 <- data.frame(
  income = tapply(sample(loans_income, 1000*5),
                  rep(1:1000, rep(5, 1000)), FUN=mean),
  type = 'mean_of_5')
# take a sample of means of 20 values
samp_mean_20 <- data.frame(
  income = tapply(sample(loans_income, 1000*20),
                  rep(1:1000, rep(20, 1000)), FUN=mean),
  type = 'mean_of_20')
# bind the data.frames and convert type to a factor
income <- rbind(samp_data, samp_mean_05, samp_mean_20)
income$type = factor(income$type,
                      levels=c('data_dist', 'mean_of_5', 'mean_of_20'),
                      labels=c('Data', 'Mean of 5', 'Mean of 20'))
# plot the histograms
ggplot(income, aes(x=income)) +
  geom_histogram(bins=40) +
  facet_grid(type ~ .)
```

El código de *Python* utiliza `FacetGrid` de `seaborn` para mostrar los tres histogramas:

```
import pandas as pd
import seaborn as sns

sample_data = pd.DataFrame({
    'income': loans_income.sample(1000),
    'type': 'Data',
})
sample_mean_05 = pd.DataFrame([
    'income': [loans_income.sample(5).mean() for _ in range(1000)],
    'type': 'Mean of 5',
])
sample_mean_20 = pd.DataFrame([
    'income': [loans_income.sample(20).mean() for _ in range(1000)],
    'type': 'Mean of 20',
])
results = pd.concat([sample_data, sample_mean_05, sample_mean_20])

g = sns.FacetGrid(results, col='type', col_wrap=1, height=2, aspect=2)
g.map(plt.hist, 'income', range=[0, 200000], bins=40)

g.set_axis_labels('Income', 'Count')
g.set_titles('{col_name}')
```

Teorema del límite central

El fenómeno que acabamos de describir se denomina *teorema del límite central (central limit theorem)*. Asegura que las medias extraídas de varias muestras se asemejarán a la conocida curva normal en forma de campana (consultar "Distribución normal" en la página 69), incluso si la población de origen no se distribuye normalmente, siempre que el tamaño de la muestra sea lo bastante grande y la desviación de los datos de lo habitual no sea demasiado grande. El teorema del límite central permite utilizar fórmulas de aproximación normal, como la distribución t, para calcular distribuciones muestrales con las que realizar inferencias, es decir, intervalos de confianza y pruebas de hipótesis.

Al teorema del límite central se le presta mucha atención en los textos de estadística tradicionales porque subyace en la maquinaria de las pruebas de hipótesis y los intervalos de confianza, que en sí mismos ocupan la mitad del espacio en dichos textos. Los científicos de datos deben ser conscientes de este papel. Sin embargo, dado que las pruebas de hipótesis formales y los intervalos de confianza juegan un papel marginal en la ciencia de datos y está disponible en cualquier caso el bootstrap (ver "Bootstrap" en la página 61), el teorema del límite central no es tan central en la práctica de la ciencia de datos.

Error estándar

El *error estándar (standard error)* es una métrica única que resume la variabilidad de la distribución muestral de un estadístico. El error estándar se puede estimar utilizando un estadístico basado en la desviación estándar s de los valores de la muestra y el tamaño n de la muestra:

$$\text{Error estándar} = SE = \frac{s}{\sqrt{n}}$$

A medida que aumenta el tamaño de la muestra, el error estándar disminuye, lo que corresponde a lo observado en la figura 2.6. La relación entre el error estándar y el tamaño de la muestra a veces se denomina regla de la *raíz cuadrada de n (square root of n)*: para reducir el error estándar en un factor de 2, el tamaño de la muestra debe aumentarse en un factor de 4.

La validez de la fórmula del error estándar surge del teorema del límite central. En realidad, no es necesario contar con el teorema del límite central para comprender el error estándar. Consideremos el siguiente enfoque para medir el error estándar:

1. Recolectamos varias muestras nuevas de la población.
2. Para cada nueva muestra, calculamos el estadístico (por ejemplo, la media).
3. Calculamos la desviación estándar de las medidas calculadas en el paso 2.
Utilizamos este valor como la estimación del error estándar.

En la práctica, este enfoque de recolectar nuevas muestras para estimar el error estándar generalmente no es factible (y estadísticamente es un desperdicio). Afortunadamente, resulta que no es necesario extraer muestras nuevas. En su lugar, podemos usar muestras repetidas de bootstrap. En la estadística moderna, el método bootstrap se ha convertido en la forma estándar de estimar el error estándar. Se puede utilizar para prácticamente cualquier estadístico y no se basa en el teorema del límite central ni en otros supuestos distributivos.



Desviación estándar versus error estándar

No hay que confundir la desviación estándar (que mide la variabilidad de puntos de datos individuales) con el error estándar (que mide la variabilidad de una métrica de la muestra).

Ideas clave

- La distribución de frecuencia de un estadístico muestral nos dice cómo esa métrica resultaría diferente de una muestra a otra.
- Esta distribución muestral se puede estimar mediante bootstrap o mediante fórmulas que se basan en el teorema del límite central.
- Una métrica clave que resume la variabilidad de un estadístico muestral es su error estándar.

Lecturas complementarias

El recurso multimedia en línea sobre estadística de David Lane (https://onlinestatbook.com/stat_sim/sampling_dist/) presenta una simulación muy útil que nos permite seleccionar una medida de la muestra, el tamaño de la muestra y el número de iteraciones y visualizar un histograma de la distribución de frecuencia resultante.

Bootstrap

Una manera sencilla y efectiva de estimar la distribución muestral de un estadístico, o de los parámetros del modelo, es extraer muestras adicionales, con reposición, de la misma muestra y volver a calcular el estadístico o modelo para cada muestra repetida. Este procedimiento se denomina *bootstrap* y no implica necesariamente ninguna suposición de que los datos o el estadístico muestral sigan una distribución normal.

Términos clave de bootstrap

Muestra de bootstrap

Muestra tomada con reposición de un conjunto de datos observados.

Remuestreo

Proceso de extraer muestras repetidas de datos observados. Incluye procedimientos tanto de bootstrap como de permutación (mezcla).

Conceptualmente, podemos imaginar que el método bootstrap replica la muestra original miles o millones de veces, de modo que tengamos una hipotética población que incorpore todo el conocimiento de nuestra muestra original (solo es más grande). A continuación, podemos extraer muestras de esta población hipotética con el fin de estimar una distribución muestral. Ver la figura 2.7.



Figura 2.7 Idea del método bootstrap.

En la práctica, no es necesario replicar la muestra una gran cantidad de veces. Simplemente reemplazamos cada observación después de cada extracción, es decir, extraemos *muestras con reposición* (*sample with replacement*). De esta manera, creamos efectivamente una población infinita en la que la probabilidad de que se extraiga un elemento permanece sin cambios de una extracción a otra. El algoritmo para el remuestreo bootstrap de la media, para una muestra de tamaño n , es el siguiente:

1. Extraemos un valor de la muestra, lo registramos y luego lo devolvemos a la muestra.
2. Lo repetimos n veces.
3. Registraremos la media de los n valores muestreados repetidamente.
4. Repetimos los pasos del 1 al 3 R veces.
5. Utilizamos los resultados de R para:

- a. Calcular su desviación estándar (esta estimación evalúa el error estándar medio de la muestra).
- b. Generar un histograma o un diagrama de caja.
- c. Encontrar el intervalo de confianza.

R, el número de iteraciones de bootstrap, se establece de forma un tanto arbitraria. Cuantas más iteraciones hagamos, más precisa será la estimación del error estándar o del intervalo de confianza. El resultado de este procedimiento es un conjunto inicial de medidas de la muestra o de parámetros estimados del modelo, que luego podemos examinar para ver cuál es su variabilidad.

El paquete `boot` de R combina estos pasos en una función. Por ejemplo, lo que sigue aplica el método bootstrap a los ingresos de los clientes que solicitan préstamos:

```
library(boot)
stat_fun <- function(x, idx) median(x[idx])
boot_obj <- boot(loans_income, R=1000, statistic=stat_fun)
```

La función `stat_fun` calcula la mediana para una muestra dada especificada por el índice `idx`. El resultado es el siguiente:

Bootstrap Statistics:		
	original	bias
t1*	62000	-70.5595
		std. Error
		209.1515

La estimación original de la mediana es de 62 000 \$. La distribución bootstrap indica que la estimación tiene un sesgo de aproximadamente -70 \$ y un error estándar de 209 \$. Los resultados variarán ligeramente entre ejecuciones consecutivas del algoritmo.

Los paquetes de *Python* más importantes no proporcionan implementaciones de la opción `bootstrap`. Se puede implementar usando el método `resample` de `scikit-learn`:

```
results = []
for nrepeat in range(1000):
    sample = resample(loans_income)
    results.append(sample.median())
results = pd.Series(results)
print('Bootstrap Statistics:')
print(f'original: {loans_income.median()}')
print(f'bias: {results.mean() - loans_income.median()}')
print(f'std. error: {results.std()}')
```

El método `bootstrap` se puede utilizar con datos multivariados donde las filas se muestrean como unidades (ver la figura 2.8). A continuación, se podría ejecutar un modelo sobre los datos después del bootstrap, por ejemplo, para hacer una estimación de la estabilidad (variabilidad) de los parámetros del modelo o para mejorar la capacidad de pronóstico. Con los árboles de clasificación y regresión (también llamados *árboles de decisión [decision trees]*), ejecutar varios árboles en muestras bootstrap y luego promediar sus pronósticos (o, con la clasificación, tomar un voto mayoritario) generalmente funciona mejor que usar un solo

árbol. A este proceso se le denomina *bagging* (acrónimo de "bootstrap aggregating"). (Consultar "Métodos de bagging y bosque aleatorio" en la página 259.)

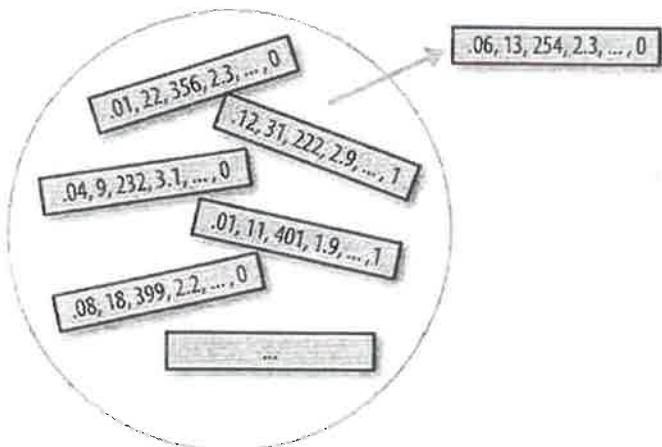


Figura 2.8 Muestreo bootstrap multivariable.

El nuevo remuestreo de bootstrap es conceptualmente simple, y Julian Simon, economista y demógrafo, publicó un compendio de ejemplos de remuestreo, incluido el método bootstrap, en su texto de 1969 *Basic Research Methods in Social Science* (Random House). Sin embargo, también requiere muchos recursos de cálculo y no era una opción factible antes de que la disponibilidad de la potencia de cálculo necesaria fuera una realidad. La técnica se ganó su nombre y despegó tras la publicación de varios artículos en revistas y de un libro del estadístico de Stanford Bradley Efron a finales de la década de 1970 y principios de la de 1980. Fue particularmente popular entre los investigadores que utilizan la estadística, pero no son estadísticos, así como para su utilización con métricas o modelos en los que no se dispone fácilmente de aproximaciones matemáticas. La distribución muestral de la media está sólidamente establecida desde 1908. No ocurre lo mismo con la distribución muestral de muchas otras métricas. El método bootstrap se puede utilizar para determinar el tamaño de la muestra. Podemos experimentar con diferentes valores de n para ver cómo se ve afectada la distribución muestral.

El método bootstrap fue recibido con considerable escepticismo cuando se introdujo por primera vez. Tenía el aura para muchos de convertir la paja en oro. Este escepticismo se debió a no haber entendido bien el propósito de bootstrap.



La utilización del método bootstrap no compensa para muestras de pequeño tamaño, no crea nuevos datos ni rellena huecos en un conjunto de datos existente. Simplemente nos informa sobre cómo se comportarían muchas muestras adicionales cuando se extrajeran de una población como nuestra muestra original.

Remuestreo frente a bootstrapping

A veces, el término *remuestreo* (*resampling*) se usa como sinónimo del término *bootstrapping*, como acabamos de describir. Más a menudo, el término *remuestreo* (*resampling*) también incluye procedimientos de permutación (consultar "Prueba de permutación" en la página 97), donde se combinan varias muestras y el muestreo se puede realizar sin reposición. En cualquier caso, el término *bootstrap* siempre implica un muestreo con reposición de un conjunto de datos observados.

Ideas clave

- El método bootstrap (muestreo con reposición de un conjunto de datos) es una potente herramienta para evaluar la variabilidad del estadístico de una muestra.
- Bootstrap se puede aplicar de manera similar en una amplia variedad de circunstancias, sin tener que realizar un estudio exhaustivo de aproximaciones matemáticas a las distribuciones muestrales.
- También nos permite estimar distribuciones muestrales para estadísticos donde no se ha desarrollado una aproximación matemática.
- Cuando se aplica a modelos predictivos, la combinación de varios pronósticos de la muestra bootstrap (bagging) supera el uso de un solo modelo.

Lecturas complementarias

- *An Introduction to the Bootstrap* de Bradley Efron y Robert Tibshirani (Chapman & Hall, 1993) fue donde se trató el método bootstrap por primera vez con la extensión de un libro. Todavía interesa mucho.
- La retrospectiva sobre bootstrap en la edición de mayo de 2003 de *Statistical Science* (vol. 18, n.º 2) analiza (entre otros antecedentes, en "A Short Prehistory of the Bootstrap" de Peter Hall) la publicación inicial de Julian Simon sobre bootstrap de 1969.
- Consultar *An Introduction to Statistical Learning* de Gareth James, Daniela Witten, Trevor Hastie y Robert Tibshirani (Springer, 2013) donde aparecen secciones sobre bootstrap y, en particular, bagging.

Intervalos de confianza

Las tablas de frecuencias, los histogramas, los diagramas de caja y los errores estándar son formas de comprender el error potencial en la estimación de una muestra. Los intervalos de confianza son otra forma de hacerlo.

Términos clave de los intervalos de confianza

Nivel de confianza

Porcentaje de intervalos de confianza, creados de la misma manera a partir de la misma población, que se espera que contengan el estadístico de interés.

Puntos finales del intervalo

Partes superior e inferior del intervalo de confianza.

Existe una aversión natural a la incertidumbre. En general (y especialmente los expertos) dicen "no sé" con demasiada frecuencia. Los analistas y gerentes, aunque reconocen la incertidumbre, depositan una confianza excesiva en una estimación cuando se presenta en forma de un único número (una *estimación puntual [point estimate]*). Presentar una estimación no como un único número sino en forma de rango es una forma de contrarrestar esta tendencia. Los intervalos de confianza llevan a cabo esta labor siguiendo los principios de muestreo estadístico.

Los intervalos de confianza siempre tienen un nivel de cobertura, expresado en porcentaje (alto), digamos 90% o 95%. Una forma de pensar en un intervalo de confianza del 90% es la siguiente: es el intervalo que encierra el 90% de la parte central de la distribución de muestreo bootstrap del estadístico de una muestra (consultar "Bootstrap" en la página 61). De manera más general, un intervalo de confianza de $x\%$ en torno a la estimación de una muestra debería, en promedio, contener estimaciones de muestras similares el $x\%$ del tiempo (cuando se sigue un procedimiento de muestreo similar).

Dada una muestra de tamaño n , y el estadístico que nos interesa de la muestra, el algoritmo para un intervalo de confianza bootstrap es el siguiente:

1. Extraemos una muestra aleatoria de tamaño n con reposición de los datos (muestra repetida).
2. Registramos el estadístico que nos interesa para la muestra repetida.
3. Repetimos los pasos 1 y 2 muchas (R) veces.
4. Para un intervalo de confianza del $x\%$, recortamos el $[(100-x) / 2]\%$ de los resultados de la muestra repetida R de cualquier extremo de la distribución.
5. Los puntos de corte son los puntos finales del intervalo de confianza bootstrap del $x\%$.

La figura 2.9 muestra un intervalo de confianza del 90% para el ingreso anual medio de los solicitantes de préstamos, basado en una muestra de 20 para la cual la media ha sido de 55 734 \$.

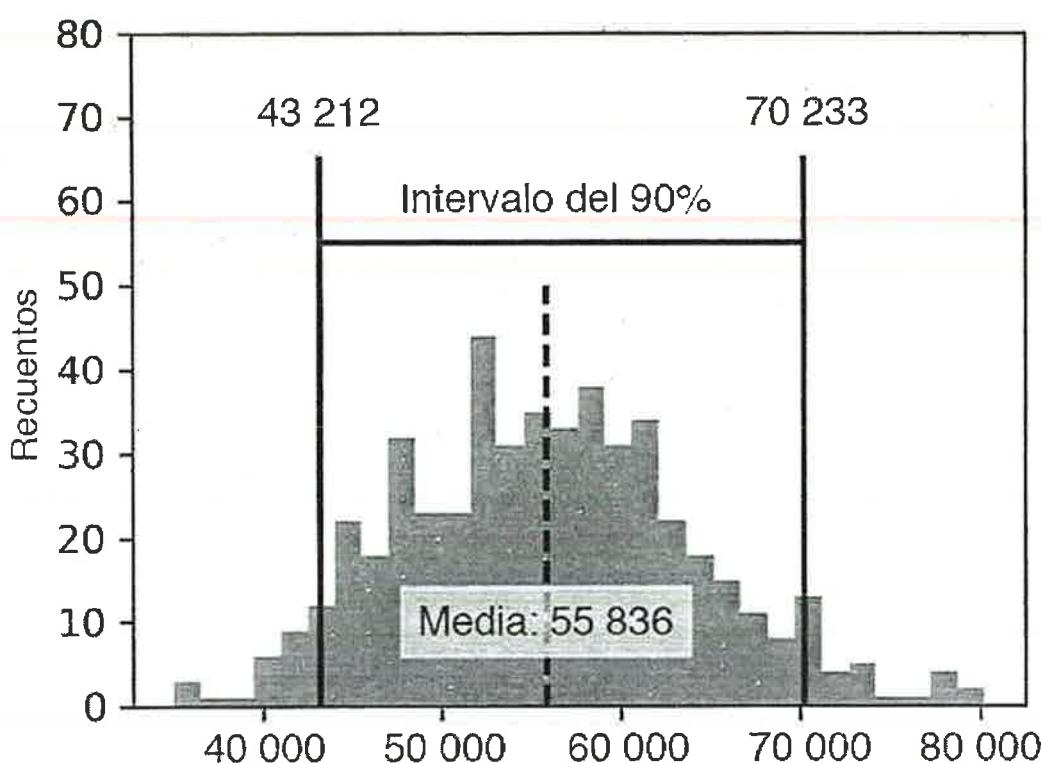


Figura 2.9 Intervalo de confianza bootstrap para los ingresos anuales de los solicitantes de préstamos, basado en una muestra de 20 solicitantes.

Bootstrap es una herramienta de carácter general que se puede utilizar para generar intervalos de confianza para la mayoría de los estadísticos o parámetros del modelo. Tanto los libros de texto como el software dedicados a la estadística, que hunde sus raíces en más de medio siglo de análisis estadístico en ausencia de ordenadores, también hacen referencia a los intervalos de confianza generados por fórmulas, especialmente la distribución t (consultar "Distribución t de Student" en la página 75).



Por supuesto, lo que realmente nos interesa cuando tenemos el resultado de una muestra es: "¿Cuál es la probabilidad de que el verdadero valor se encuentre dentro de un cierto intervalo?". Esta no es realmente la pregunta que responde el intervalo de confianza, pero termina siendo la forma en que la mayoría interpreta la respuesta.

La pregunta sobre probabilidad asociada con el intervalo de confianza comienza con la frase "Dado un procedimiento de muestreo y una población, ¿cuál es la probabilidad de que...?". Ir en la dirección opuesta, "Dado un resultado de la muestra, ¿cuál es la probabilidad de que (algo sea cierto sobre la población)??" implica cálculos más complejos e imponderables más profundos.

El porcentaje asociado con el intervalo de confianza se denomina *nivel de confianza (level of confidence)*. Cuanto mayor sea el nivel de confianza, más amplio será el intervalo. Además, cuanto más pequeña es la muestra, más amplio es el intervalo (es decir, mayor es la incertidumbre). Ambos casos tienen sentido: cuanta más confianza deseemos tener y cuantos menos datos tengamos, más amplio debemos hacer el intervalo de confianza para estar lo suficientemente seguros de capturar el valor real.



Para un científico de datos, el intervalo de confianza es una herramienta que puede usarse para tener una idea de cómo podría variar el resultado de una muestra. Los científicos de datos no usarían esta información para publicar un artículo académico o enviar un resultado a una agencia reguladora (como lo haría un investigador), sino que lo más probable es que comuniquen el error potencial mediante una estimación y tal vez para saber si se necesita una muestra más grande.

Ideas clave

- Los intervalos de confianza son la forma típica de presentar la estimación del rango de un intervalo.
- Cuantos más datos tengamos, menos variable será la estimación de muestra.
- Cuanto menor sea el nivel de confianza que podamos tolerar, más estrecho será el intervalo de confianza.
- Bootstrap es una forma eficaz de crear intervalos de confianza.

Lecturas complementarias

- Para abordar un enfoque bootstrap de los intervalos de confianza, consultar *Introductory Statistics and Analytics: A Resampling* de Peter Bruce (Wiley, 2014) o *Statistics: Unlocking the Power of Data*, 2.^a ed., de Robin Lock y otros cuatro miembros de la familia Lock (Wiley, 2016).
- Los ingenieros, que tienen la necesidad de comprender la precisión de sus mediciones, usan intervalos de confianza más que la mayoría de las otras especialidades, y *Modern Engineering Statistics* de Thomas Ryan (Wiley, 2007) analiza los intervalos de confianza. También revisa una herramienta que es igualmente útil y recibe menos atención: los *intervalos de pronóstico (prediction intervals)* (intervalos en torno a un valor único, en contraposición a la media u otro estadístico resumen).

Distribución normal

La distribución normal en forma de campana es icónica en la estadística tradicional¹. El hecho de que las distribuciones de los estadísticos muestrales tengan a menudo una forma normal la ha convertido en una potente herramienta en el desarrollo de fórmulas matemáticas que se aproximen a esas distribuciones.

Términos clave de la distribución normal

Error

Diferencia entre un punto de datos y un valor pronosticado o promedio.

Estandarizar

Restamos la media y dividimos por la desviación estándar.

Puntuación z

Resultado de estandarizar un punto de datos individual.

Estándar normal

Distribución normal con la media = 0 y la desviación estándar = 1.

Diagrama QQ

Diagrama para visualizar lo cerca que está una distribución de una muestra de una distribución específica, por ejemplo, la distribución normal.

En una distribución normal (figura 2.10), el 68% de los datos se encuentra en una desviación estándar de la media y el 95% se halla dentro de dos desviaciones estándar.



Es un error frecuente pensar que la distribución normal se llama así porque la mayoría de los datos siguen una distribución normal, es decir, es lo normal. La mayoría de las variables utilizadas en un proyecto típico de ciencia de datos (de hecho, la mayoría de los datos sin procesar en su conjunto) *no* se distribuyen normalmente: consultar "Distribuciones de cola larga" en la página 73. La utilidad de la distribución normal se deriva del hecho que muchos estadísticos *están* distribuidos normalmente en su distribución muestral. Aun así, los supuestos de normalidad son generalmente el último recurso y se utilizan cuando no se dispone de distribuciones de probabilidad empírica o distribuciones bootstrap.

¹ La curva de la campana es icónica pero quizás está sobrevalorada. George W. Cobb, el estadístico de Mount Holyoke, conocido por su contribución a la filosofía de la enseñanza de introducción a la estadística, argumentó en un editorial de noviembre de 2015 en el *American Statistician* que el "curso introductorio estándar, que coloca la distribución normal en su centro, había sobrevivido a la utilidad de su centralidad".

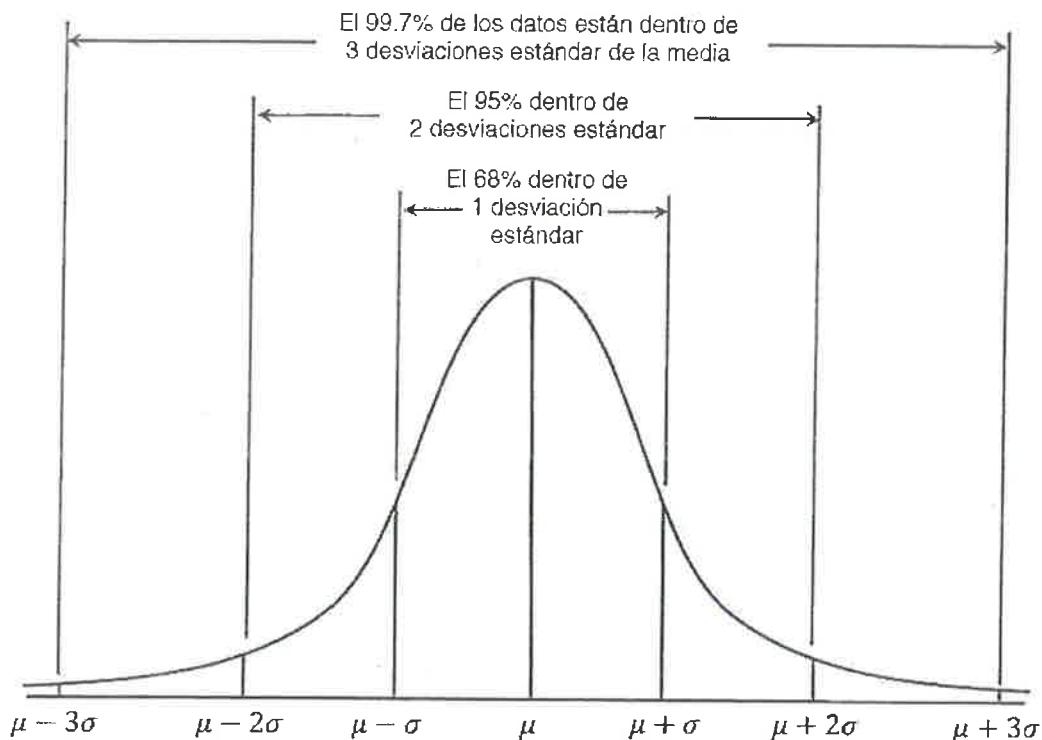


Figura 2.10 Curva normal.



La distribución normal también se conoce como distribución gaussiana en honor a Carl Friedrich Gauss, el prodigioso matemático alemán de finales del siglo XVIII y principios del XIX. Otro nombre que se utilizó anteriormente para la distribución normal fue la distribución de "error". Hablando en sentido estadístico, un *error* es la diferencia entre un valor real y una estimación estadística como la media de una muestra. Por ejemplo, la desviación estándar (consultar "Estimación de la variabilidad" en la página 13) se basa en los errores de la media de los datos. El desarrollo de Gauss de la distribución normal provino de su estudio de los errores de las mediciones astronómicas que se encontró que estaban distribuidas normalmente.

Normal estándar y diagramas QQ

Una distribución *normal estándar* (*standard normal*) es aquella en la que las unidades del eje x se expresan en términos de desviaciones estándar de la media. Para comparar los datos con una distribución normal estándar, restamos la media y luego la dividimos por la desviación estándar. A esto también se le denomina normalización o estandarización (consultar "Estandarización [normalización, puntuación z]" en la página 243). Hay que tener en cuenta que la "estandarización" en este sentido no está relacionada con la estandarización de registros de bases de datos (conversión a un formato común). El valor transformado se denomina puntuación z, y a veces a la distribución normal, *distribución z* (*z-distribution*).

El *diagrama QQ* (*QQ-Plot*) se utiliza para determinar visualmente lo cerca que está una muestra de una distribución especificada, en este caso, la distribución normal. El diagrama QQ ordena la puntuación z de menor a mayor y registra la puntuación z de cada valor en el eje y. El eje x es el cuantil correspondiente a una distribución normal para el rango de ese valor. Dado que los datos están normalizados, las unidades corresponden al número de desviaciones estándar de la media. Si los puntos caen aproximadamente en la línea diagonal, entonces la distribución de la muestra se puede considerar cercana a la normal. La figura 2.11 muestra un diagrama QQ para una muestra de 100 valores generados aleatoriamente a partir de una distribución normal. Como se esperaba, los puntos siguen de cerca la línea. Esta figura se puede generar en R con la función `qqnorm`:

```
norm_samp <- rnorm(100)
qqnorm(norm_samp)
abline(a=0, b=1, col='grey')
```

En *Python*, utilizamos el método `scipy.stats.probplot` para crear el diagrama QQ:

```
fig, ax = plt.subplots(figsize=(4, 4))
norm_sample = stats.norm.rvs(size=100)
stats.probplot(norm_sample, plot=ax)
```

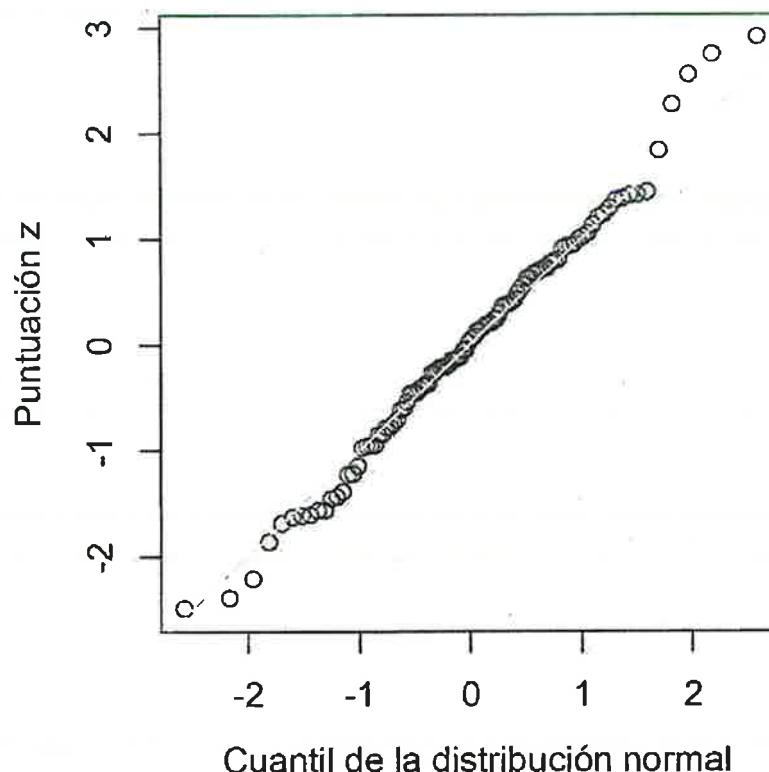


Figura 2.11 Diagrama QQ de una muestra de 100 valores extraídos de una distribución normal estándar.



Convertir datos a puntuación z (es decir, estandarizar o normalizar los datos) no hace que los datos se distribuyan normalmente. Coloca los datos en la misma escala que la distribución normal estándar, a menudo con fines de comparación.

Ideas clave

- La distribución normal fue fundamental para el desarrollo histórico de la estadística, ya que permitió una aproximación matemática a la incertidumbre y la variabilidad.
- Si bien los datos en bruto generalmente no se distribuyen normalmente, los errores a menudo lo hacen, al igual que los promedios y totales en muestras grandes.
- Para convertir datos en puntuación z, restamos la media de los datos y dividimos por la desviación estándar. Luego podemos comparar los datos con la distribución normal.

Distribuciones de cola larga

A pesar de la importancia histórica de la distribución normal en estadística, y en contraste con lo que sugiere el nombre, los datos no suelen distribuirse normalmente.

Términos clave de las distribuciones de cola larga

Cola

Porción larga y estrecha de una distribución de frecuencias, donde se presentan valores relativamente extremos a bajas frecuencias.

Colas asimétricas

Donde una de las colas de una distribución es más larga que la otra.

Si bien la distribución normal suele ser apropiada y resulta útil en lo que respecta a la distribución de errores y los estadísticos muestrales, por lo general no caracteriza a la distribución de los datos en bruto. A veces, la distribución está muy *sesgada* (*skewed*) (asimétrica), como ocurre con los datos de ingresos, o puede ser discreta, como ocurre con los datos binomiales. Tanto las distribuciones simétricas como asimétricas pueden tener *colas largas* (*long tails*). Las colas de una distribución corresponden a los valores extremos (pequeños y grandes). Las colas largas y la protección contra ellas están ampliamente reconocidas en la práctica. Nassim Taleb ha propuesto la teoría del *cisne negro* (*black swan*), que pronostica que es mucho más probable que ocurran eventos anómalos, como una caída del mercado de valores, de lo que pronosticaría la distribución normal.

Un buen ejemplo para ilustrar la naturaleza de la cola larga de los datos son las rentabilidades de las acciones. La figura 2.12 muestra el diagrama QQ correspondiente a la rentabilidad diaria de las acciones de Netflix (NFLX). El diagrama se genera en R con:

```
nflx <- sp500_px['NFLX']
nflx <- diff(log(nflx[nflx>0]))
qqnorm(nflx)
abline(a=0, b=1, col='grey')
```

El correspondiente código de *Python* es:

```
nflx = sp500_px.NFLX
nflx = np.diff(np.log(nflx[nflx>0]))
fig, ax = plt.subplots(figsize=(4, 4))
stats.probplot(nflx, plot=ax)
```

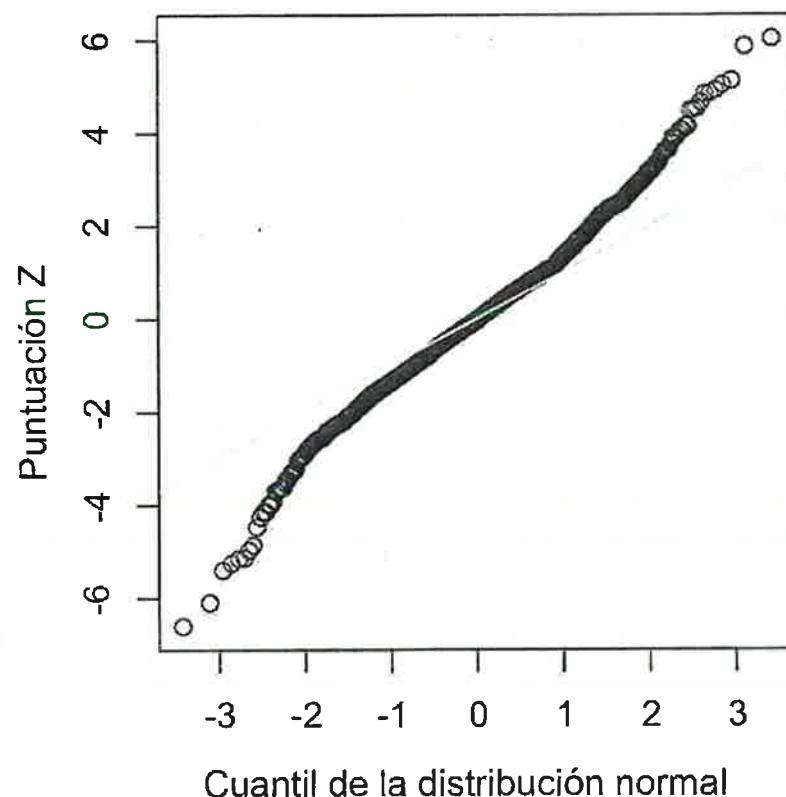


Figura 2.12 Diagrama QQ de las rentabilidades de Netflix (NFLX).

En contraste con la figura 2.11, los puntos están muy por debajo de la línea para valores bajos y muy por encima de la línea para valores altos, lo que indica que los datos no están distribuidos normalmente. Esto significa que es mucho más probable que observemos valores extremos de lo que cabría esperar si los datos tuvieran una distribución normal. La figura 2.12 muestra otro fenómeno frecuente: los puntos están cerca de la línea de los datos dentro de la desviación estándar de la media. Tukey se refiere a este fenómeno como datos "normales en el medio", pero con colas mucho más largas (ver [Tukey, 1987]).



Existe una gran cantidad de literatura estadística en relación con la tarea de ajustar las distribuciones estadísticas a los datos observados. Hay que tener cuidado con adoptar un enfoque excesivamente centrado en los datos para este trabajo, que es tanto arte como ciencia. A primera vista, los datos son variados y a menudo coherentes, con más de una forma y tipo de distribución. Por lo general, es necesario aplicar la competencia y los conocimientos estadísticos para determinar qué tipo de distribución es apropiada para modelar una situación determinada. Por ejemplo, podríamos tener datos sobre el nivel de tráfico de Internet en un servidor durante muchos períodos consecutivos de cinco segundos. Es útil saber que la mejor distribución para modelar "eventos por períodos de tiempo" es la de Poisson (consultar "Distribución de Poisson" en la página 83).

Ideas clave

- La mayoría de los datos no se distribuyen normalmente.
- Asumir una distribución normal puede llevar a subestimar los eventos extremos ("cisnes negros").

Lecturas complementarias

- *The Black Swan*, 2.^a ed., de Nassim Nicholas Taleb (Random House, 2010).
- *Handbook of Statistical Distributions with Applications*, 2.^a ed., de K. Krishnamoorthy (Chapman & Hall/CRC Press, 2016).

Distribución t de Student

La *distribución t* (*t-distribution*) es una distribución de forma normal, excepto que es un poco más gruesa y más larga en las colas. Se utiliza muy a menudo para representar distribuciones de estadísticos muestrales. Las distribuciones de las medias muestrales suelen tener la forma de la distribución t, y hay una familia de distribuciones t que difieren dependiendo del tamaño de la muestra. Cuanto más grande es la muestra, más normal se vuelve la distribución t.

Términos clave de la distribución t de Student

n

Tamaño de la muestra.

Grados de libertad

Parámetro que permite que la distribución t se ajuste a diferentes tamaños de la muestra, estadísticos y números de grupos.

La distribución t a menudo se llama *t de Student (Student's t)*; fue publicada en 1908 en *Biometrika* por W. S. Gosset con el nombre de "Student". La empresa para la que trabajaba Gosset, la fábrica de cerveza Guinness, no quería que los competidores supieran que estaba utilizando métodos estadísticos, por lo que insistió en que Gosset no usara su nombre en el artículo.

Gosset quería responder a la pregunta: "¿Cuál es la distribución muestral de la media de una muestra, extraída de una población más grande?". Comenzó con un experimento de remuestreo: extrajo muestras aleatorias de 4 de un conjunto de datos de 3000 medidas, realizadas a delincuentes, de la estatura y longitud del dedo medio izquierdo. (Siendo esta la era de la eugeniosidad, había mucho interés en los datos sobre delincuentes y en descubrir correlaciones entre las tendencias delictivas y los atributos físicos o psicológicos.) Gosset presentó en un diagrama los resultados estandarizados (la puntuación z) en el eje x y la frecuencia en el eje y. Por separado, había derivado una función, ahora conocida como *Student's t*, y ajustó esta función a los resultados de la muestra, trazando la comparación (consultar la figura 2.13).

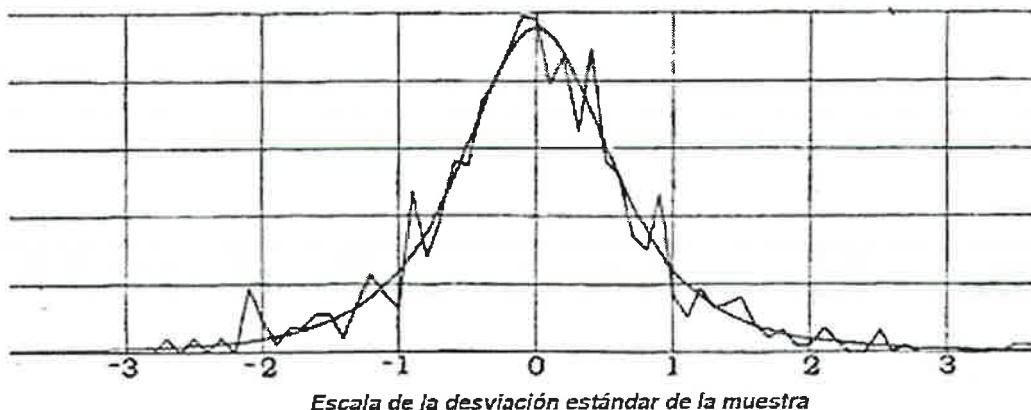


Figura 2.13 Resultados del experimento de remuestreo de Gosset y la curva t ajustada (de su artículo de 1908 en *Biometrika*).

Después de la estandarización, se pueden comparar varios estadísticos diferentes con la distribución t, para estimar los intervalos de confianza a la luz de la variación muestral. Consideremos una muestra de tamaño n para la cual se ha calculado la media muestral \bar{x} . Si s es la desviación estándar de la muestra, un intervalo de confianza del 90% en torno a la media de la muestra viene dado por:

$$\bar{x} \pm t_{n-1}(0.05) \cdot \frac{s}{\sqrt{n}}$$

donde $t_{n-1}(0.05)$ es el valor del estadístico t, con $(n - 1)$ grados de libertad (consultar “Grados de libertad” en la página 116), que “recorta” el 5% de la distribución t en cada extremo. La distribución t se ha utilizado como referencia para la distribución de la media muestral, la diferencia entre dos medias muestrales, parámetros de regresión y otros estadísticos.

Si la potencia de cálculo hubiera estado disponible de forma generalizada en 1908, la estadística sin duda se habría basado mucho más en métodos de remuestreo, ya que requieren muchos recursos de cálculo desde el principio. Al carecer de ordenadores, los estadísticos recurrieron a las matemáticas y a funciones como la distribución t para aproximar las distribuciones muestrales. El poder de los ordenadores permitió experimentos prácticos de remuestreo en la década de 1980, pero para entonces, el uso de la distribución t y distribuciones similares había arraigado profundamente en los libros de texto y el software.

La precisión de la distribución t al representar el comportamiento del estadístico de una muestra requiere que la distribución de ese estadístico para esa muestra tenga la forma de una distribución normal. Resulta que los estadísticos de las muestras a menudo *están* distribuidos de forma normal, incluso cuando los datos subyacentes de la población no lo estén (un hecho que llevó a una aplicación generalizada de la distribución t). Este hecho nos lleva de regreso al fenómeno conocido como el *teorema del límite central* (*central limit theorem*) (ver “Teorema del límite central” en la página 60).



¿Qué necesitan saber los científicos de datos sobre la distribución t y el teorema del límite central? No mucho. La distribución t se utiliza en la inferencia estadística clásica, pero no es tan esencial para los propósitos de la ciencia de datos. Comprender y cuantificar la incertidumbre y la variación es importante para los científicos de datos, pero el muestreo empírico de bootstrap puede responder a la mayoría de las preguntas sobre el error de muestreo. Sin embargo, los científicos de datos encontrarán en R de forma rutinaria estadísticos t en el resultado del software estadístico y en procedimientos estadísticos, por ejemplo, en pruebas A/B y regresiones, por lo que es útil familiarizarse con su propósito.

Ideas clave

- La distribución t es en realidad una familia de distribuciones que se asemeja a la distribución normal, pero con colas más gruesas.
- La distribución t se utiliza ampliamente como base de referencia para la distribución de medias muestrales, diferencias entre dos medias muestrales, parámetros de regresión, etc.

Lecturas complementarias

- El artículo original de W. S. Gosset publicado en *Biometrika* en 1908 está disponible en PDF (http://seismo.berkeley.edu/~kirchner/eps_120/Odds_n_ends/Students_original_paper.pdf).
- Podemos encontrar un tratamiento de la distribución t en el recurso en línea de David Lane (https://onlinestatbook.com/2/estimation/t_distribution.html).

Distribución binomial

Los resultados sí/no (binomiales) se encuentran en el corazón de la analítica, ya que a menudo son la culminación de una decisión o de otro proceso: comprar/no comprar, hacer clic/no hacer clic, sobrevivir/morir, etc. Para comprender la distribución binomial es fundamental la idea de un conjunto de *ensayos (trials)*, cada uno de los cuales tiene dos resultados posibles con probabilidades definidas.

Por ejemplo, lanzar una moneda 10 veces es un experimento binomial con 10 intentos, cada intento tiene dos resultados posibles (cara o cruz). (Ver la figura 2.14.) Estos resultados sí/no o 0/1 se denominan resultados *binarios (binary)* y no necesitan tener probabilidades de 50/50. Cualquier probabilidad que sume 1.0 es posible. Es convencional en estadística denominar el resultado “1” como resultado de *éxito (success)*. También es una práctica común asignar “1” al resultado más raro. El uso del término *éxito (success)* no implica que el resultado sea deseable o beneficioso, pero tiende a indicar el resultado que nos interesa. Por ejemplo, los incumplimientos de préstamos o las transacciones fraudulentas son eventos relativamente poco comunes que podríamos estar interesados en pronosticar, por lo que se denominan “1” o “éxitos”.



Figura 2.14 El lado cruz de una moneda de níquel de cinco centavos.

Términos clave de la distribución binomial

Ensayo

Evento con un resultado discreto (por ejemplo, el lanzamiento de una moneda).

Éxito

Resultado de interés para un ensayo.

Sinónimo

“1” (como opuesto a “0”)

Binomial

Que tiene dos resultados.

Sinónimos

sí/no, 0/1, binario

Ensayo binomial

Ensayo con dos resultados.

Sinónimo

Ensayo de Bernoulli

Distribución binomial

Distribución del número de éxitos en n ensayos.

Sinónimo

distribución de Bernoulli

La distribución binomial es la distribución de frecuencias del número de éxitos (x) en un número dado de ensayos (n) con una probabilidad específica (p) de éxito en cada ensayo. Existe una familia de distribuciones binomiales, dependiendo de los valores de n y p . La distribución binomial respondería a una pregunta como:

Si la probabilidad de que un clic se convierta en una venta es 0.02, ¿cuál es la probabilidad de observar 0 ventas en 200 clics?

La función `dbinom` de R calcula probabilidades binomiales. Por ejemplo:

```
dbinom(x=2, size=5, p=0.1)
```

devolvería 0.0729, la probabilidad de observar exactamente $x = 2$ éxitos con un tamaño (`size`) = 5 ensayos, donde la probabilidad de éxito para cada ensayo es $p = 0.1$. Para el ejemplo anterior, utilizamos $x = 0$, `size = 200` y $p = 0.02$. Con estos argumentos, `dbinom` devuelve una probabilidad de 0.0176.

A menudo nos interesa determinar la probabilidad de x o menos éxitos en n ensayos. En este caso, usamos la función `pbinom`:

```
pbinom(2, 5, 0.1)
```

Esta función devolvería 0.9914, la probabilidad de observar dos o menos éxitos en cinco ensayos, donde la probabilidad de éxito para cada ensayo es 0.1.

El módulo `scipy.stats` implementa una gran variedad de distribuciones estadísticas. Para la distribución binomial, usamos las funciones `stats.binom.pmf` y `stats.binom.cdf`:

```
stats.binom.pmf(2, n=5, p=0.1)
stats.binom.cdf(2, n=5, p=0.1)
```

La media de una distribución binomial es $n \times p$. También podemos pensar en este valor como el número esperado de éxitos en n ensayos, para una probabilidad de éxito = p .

La varianza es $n \times p (1 - p)$. Con un número suficientemente grande de ensayos (particularmente cuando p está cerca de 0.50), la distribución binomial es virtualmente indistinguible de la distribución normal. De hecho, calcular probabilidades binomiales con tamaños de muestra grandes exige un gran trabajo de cálculo, y la mayoría de los procedimientos estadísticos utilizan la distribución normal, con media y varianza, como aproximación.

Ideas clave

- Los resultados binomiales son importantes para modelar, ya que representan, entre otras cosas, decisiones fundamentales (comprar o no comprar, hacer clic o no hacer clic, sobrevivir o morir, etc.).
- Un ensayo binomial es un experimento con dos posibles resultados: uno con probabilidad p y otro con probabilidad $1 - p$.
- Con un valor de n grande, y siempre que p no esté demasiado cerca de 0 o 1, la distribución binomial se puede aproximar a la distribución normal.

Lecturas complementarias

- Información acerca de “quincunx” (<https://www.mathsisfun.com/data/binomial-distribution.html>), un dispositivo de simulación similar a un pinball para ilustrar la distribución binomial.
- La distribución binomial es un elemento básico en la introducción a la estadística, y todos los textos introductorios de estadística tendrán uno o dos capítulos sobre ella.

Distribución chi cuadrado

Una idea importante en estadística es *alejarse de las expectativas* (*departure from expectation*), especialmente con respecto a los recuentos de categorías. La expectativa se define vagamente como “nada inusual o notable en los datos” (por ejemplo, que no exista

correlación entre variables o patrones predecibles). Esto también se denomina "hipótesis nula" o "modelo nulo" (consultar "La hipótesis nula" en la página 94). Por ejemplo, es posible que deseemos probar si una variable (por ejemplo, una variable de fila que representa el género) es independiente de otra (por ejemplo, una variable de columna que representa "lo ascendieron en el trabajo"), y tenemos recuentos de la cantidad en cada una de las celdas de la tabla de datos. El estadístico que mide hasta qué punto los resultados se alejan de la expectativa nula de independencia es el estadístico chi cuadrado. Es la diferencia entre los valores observados y esperados, dividida por la raíz cuadrada del valor esperado, todo ello al cuadrado y a continuación se hace la suma de todas las categorías. Este proceso estandariza el estadístico para que pueda compararse con una distribución de referencia. Una forma más general de expresarlo es señalar que el estadístico chi cuadrado es una medida del grado en que un conjunto de valores observados "se ajusta" a una distribución específica (una prueba de "bondad del ajuste"). Es útil para determinar si varios tratamientos (una "prueba A/B/C...") difieren entre sí en sus efectos.

La distribución chi cuadrado es la distribución de este estadístico bajo repetidas extracciones de muestras repetidas del modelo nulo. Consultar "Prueba de chi cuadrado" en la página 124 para obtener un algoritmo detallado y la fórmula de chi cuadrado para una tabla de datos. Un valor bajo de chi cuadrado para un conjunto de recuentos indica que estos siguen de cerca la distribución esperada. Un valor alto de chi cuadrado indica que difieren notablemente de lo esperado. Hay una variedad de distribuciones de chi cuadrado asociadas con diferentes grados de libertad (por ejemplo, el número de observaciones). (Consultar "Grados de libertad" en la página 116.)

Ideas clave

- La distribución chi cuadrado se refiere típicamente a los recuentos de sujetos o elementos que se dividen en categorías.
- El estadístico chi cuadrado mide el grado de desviación que cabría esperar en un modelo nulo.

Lecturas complementarias

- La distribución chi cuadrado debe su lugar en la estadística moderna al gran estadístico Karl Pearson y al nacimiento de la prueba de hipótesis. Podemos leer sobre este tema y sobre más cosas en *The Lady Tasting Tea: How Statistics Revolutionized Science in the Twentieth Century*, de David Salsburg (W. H. Freeman, 2001).
- Para disponer de una exposición más detallada, consultar la sección de este libro sobre la prueba de chi cuadrado ("Prueba de chi cuadrado" en la página 124).

Distribución F

Un procedimiento habitual en la experimentación científica es probar varios tratamientos en todos los grupos, por ejemplo, diferentes fertilizantes en distintas parcelas de un campo de cultivo. Este procedimiento es similar a la prueba A/B/C a la que se hace referencia en la distribución chi cuadrado (consultar “Distribución chi cuadrado” en la página 80), a diferencia de que estamos tratando con medidas de valores continuos en lugar de tratar con recuentos. En este caso, nos interesa saber hasta qué punto las diferencias entre las medias de los grupos son mayores de lo que cabría esperar en una variación aleatoria normal. El estadístico F realiza esta medida y es la razón entre la variación entre las medias de los grupos y la variación dentro de cada grupo (también llamada variación residual). Esta comparación se denomina *análisis de varianza (analysis of variance)* (consultar “ANOVA” en la página 118). La distribución del estadístico F es la distribución de frecuencias de todos los valores que se produciría permutando aleatoriamente los datos en los que todas las medias de los grupos son iguales (es decir, un modelo nulo). Hay una serie de distribuciones F asociadas con diferentes grados de libertad (por ejemplo, el número de grupos. Consultar “Grados de libertad” en la página 116). El cálculo de F se ilustra en la sección de ANOVA. El estadístico F también se utiliza en regresión lineal para comparar la variación explicada por el modelo de regresión con la variación general de los datos. R y Python generan automáticamente los estadísticos F como parte de las rutinas de regresión y ANOVA.

Ideas clave

- La distribución F se utiliza con experimentos y modelos lineales que involucran datos medidos.
- El estadístico F compara la variación debida a factores de interés con la variación general.

Lecturas complementarias

Introduction to Design and Analysis of Experiments de George Cobb (Wiley, 2008) contiene una excelente exposición de la descomposición de los componentes de la varianza, lo que ayuda a comprender ANOVA y el estadístico F.

La distribución de Poisson y distribuciones relacionadas

Muchos procesos producen eventos aleatoriamente a una velocidad de conjunto determinada: visitantes que llegan a un sitio web o automóviles que llegan a una plaza de peaje (eventos que se extienden a lo largo del tiempo), imperfecciones en un metro cuadrado de tela o errores tipográficos por cada 100 líneas de código (eventos espaciados por el espacio).

Términos clave de la distribución de Poisson y distribuciones relacionadas

Lambda

La tasa (por unidad de tiempo o espacio) a la que ocurren los eventos.

Distribución de Poisson

Distribución de frecuencias del número de eventos en unidades muestreadas de tiempo o espacio.

Distribución exponencial

Distribución de frecuencias del tiempo o la distancia entre un evento y el siguiente.

Distribución de Weibull

Versión generalizada de la distribución exponencial en la que se permite que la tasa de eventos cambie con el tiempo.

Distribución de Poisson

A partir de datos agregados con anterioridad (por ejemplo, la cantidad de infecciones de la gripe al año), podemos estimar la cantidad promedio de eventos por unidad de tiempo o espacio (por ejemplo, infecciones al día o por unidad de censo). También podríamos querer saber lo diferente que puede ser este proceso de una unidad de espacio/tiempo a otra. La distribución de Poisson nos proporciona la distribución de eventos por unidad de tiempo o espacio cuando tomamos muestras de muchas de esas unidades. Es útil cuando se abordan preguntas sobre colas de espera como "¿Qué capacidad necesitamos para estar seguros al 95% de procesar completamente el tráfico de Internet que llega a un servidor en un periodo de cinco segundos?".

El parámetro clave en una distribución de Poisson es λ o lambda. Este es el número medio de eventos que ocurren en un intervalo de tiempo o espacio específico. La varianza para una distribución de Poisson también es λ .

Una técnica habitual es generar números aleatorios a partir de una distribución de Poisson como parte de una simulación de colas de espera. La función `rpois` en R realiza esta función, utilizando solamente dos argumentos: la cantidad de números aleatorios buscados y `lambda`:

```
rpois(100, lambda=2)
```

La correspondiente función `scipy` es `stats.poisson.rvs`:

```
stats.poisson.rvs(2, size=100)
```

Este código generará 100 números aleatorios a partir de la distribución de Poisson con $\lambda = 2$. Por ejemplo, si el promedio de llamadas entrantes de servicio al cliente es de dos por minuto, este código simulará 100 minutos, proporcionando el resultado de la cantidad de llamadas en cada uno de esos 100 minutos.

Distribución exponencial

Utilizando el mismo parámetro λ que empleamos en la distribución de Poisson, también podemos modelar la distribución del tiempo entre eventos: el tiempo entre visitas a un sitio web o entre automóviles que llegan a un aparcamiento. También se utiliza en ingeniería para modelar el tiempo hasta que se produce el fallo de un componente y en la gestión de procesos para modelar, por ejemplo, el tiempo requerido por llamada de servicio. El código R para generar números aleatorios a partir de una distribución exponencial utiliza dos argumentos: `n` (la cantidad de números que se generarán) y `rate` (número de eventos por periodo de tiempo). Por ejemplo:

```
rexp(n=100, rate=0.2)
```

En la función `stats.expon.rvs`, se invierte el orden de los argumentos:

```
stats.expon.rvs(scale=5, size=100)
```

Este código generaría 100 números aleatorios a partir de una distribución exponencial donde el número medio de eventos por periodo de tiempo es de 0.2. Por lo tanto, podríamos usarlo para simular 100 intervalos, en minutos, entre las llamadas de servicio, donde la tasa promedio de llamadas entrantes es 0.2 por minuto.

Un supuesto clave en cualquier estudio de simulación para la distribución de Poisson o exponencial es que la tasa, λ , permanece constante durante el periodo considerado. Esto rara vez es razonable desde el punto de vista global. Por ejemplo, el tráfico en carreteras o a través de redes de datos varía según la hora del día y el día de la semana. Sin embargo, los periodos de tiempo, o áreas del espacio, generalmente se pueden dividir en segmentos que sean lo suficientemente homogéneos para que el análisis o la simulación dentro de esos periodos sean válidos.

Estimación de la tasa de fallos

En muchas aplicaciones, la tasa de eventos, λ , se conoce o se puede estimar a partir de datos anteriores. Sin embargo, para eventos raros, esto no es necesariamente así. El fallo del motor de una aeronave, por ejemplo, es lo suficientemente raro (afortunadamente) que, para un tipo de motor dado, puede haber pocos datos sobre los que fundamentar una estimación del tiempo entre fallos. Si no disponemos de ningún dato, hay pocas bases para estimar la tasa de eventos. Sin embargo, podemos hacer algunas conjeturas: si no se han visto eventos después de 20 horas, podemos estar bastante seguros de que la tasa no es de 1 por hora. Mediante la simulación o el cálculo directo de probabilidades, podemos evaluar diferentes tasas de eventos hipotéticos y estimar valores de umbral por debajo de los cuales es muy poco probable que la tasa disminuya. Si hay algunos datos, pero no los suficientes para proporcionar una estimación precisa y confiable de la tasa, se puede aplicar la prueba de bondad de ajuste (consultar "Prueba de chi cuadrado" en la página 124) a varias tasas para determinar lo bien que se ajustan a los datos observados.

Distribución de Weibull

En muchos casos, la tasa de eventos no permanece constante a lo largo del tiempo. Si el periodo durante el cual cambia es mucho más largo que el intervalo típico entre eventos, no hay problema, simplemente subdividimos el análisis en los segmentos donde las tasas sean relativamente constantes, como se mencionó con anterioridad. Sin embargo, si la tasa de eventos cambia durante el tiempo del intervalo entre eventos, las distribuciones exponenciales (o de Poisson) ya no son útiles. Es probable que este sea el caso de los fallos mecánicos: el riesgo de fallos aumenta a medida que pasa el tiempo. La distribución de Weibull es una extensión de la distribución exponencial en la que se permite que la tasa de eventos cambie, según lo especificado por un *parámetro de forma (shape parameter)*, β . Si $\beta > 1$, la probabilidad de un evento aumenta con el tiempo. Si $\beta < 1$, la probabilidad disminuye. Debido a que la distribución de Weibull se utiliza con el análisis del tiempo hasta que se produce el fallo en lugar de la tasa de eventos, el segundo parámetro se expresa en términos de vida característica, en lugar de hacerlo en términos de la tasa de eventos por intervalo. El símbolo utilizado es η , la letra griega eta. También se le llama parámetro de escala.

Con Weibull, la tarea de valoración ahora incluye la valoración de ambos parámetros, β y η . Se emplea software para modelar los datos y obtener la estimación de la distribución de Weibull que mejor se ajusta.

El código *R* para generar números aleatorios a partir de una distribución de Weibull utiliza tres argumentos: *n* (cantidad de números que se generarán), *shape* y *scale*. Por ejemplo, el siguiente código generaría 100 números aleatorios (vidas útiles) a partir de una distribución de Weibull con *shape* de 1.5 y *vida característica* de 5000:

```
rweibull(100, 1.5, 5000)
```

Para conseguir lo mismo con *Python*, utilizamos la función `stats.weibull_min.rvs`:

```
stats.weibull_min.rvs(1.5, scale=5000, size=100)
```

Ideas clave

- Para eventos que ocurren con una tasa constante, el número de eventos por unidad de tiempo o espacio se puede modelar como una distribución de Poisson.
- También podemos modelar el tiempo o la distancia entre un evento y el siguiente como una distribución exponencial.
- Se puede modelar una tasa de eventos cambiante a lo largo del tiempo (por ejemplo, una probabilidad creciente de fallo del dispositivo) con la distribución de Weibull.

Lecturas complementarias

- *Modern Engineering Statistics* de Thomas Ryan (Wiley, 2007) tiene un capítulo dedicado a las distribuciones de probabilidad utilizadas en aplicaciones de ingeniería.
- Se puede leer una perspectiva basada en la ingeniería sobre el uso de la distribución de Weibull aquí (http://www.ipedr.com/vol75/29_ICQM2014-051.pdf).

Resumen

En la era del big data, los principios del muestreo aleatorio siguen siendo importantes cuando se necesitan estimaciones precisas. La selección aleatoria de datos puede reducir el sesgo y producir un conjunto de datos de mayor calidad que solo utilizando convenientemente los datos disponibles. El conocimiento de varias distribuciones de muestreo y generación de datos nos permite cuantificar errores potenciales en una estimación que podría deberse a variaciones aleatorias. Al mismo tiempo, el método bootstrap (muestreo con reposición de un conjunto de datos observados) es un método atractivo de “talla única” para determinar el posible error en las estimaciones de la muestra.

CAPÍTULO 3

Experimentos estadísticos y pruebas significativas

El diseño de experimentos es la piedra angular de la práctica de la estadística, encontrando aplicaciones en prácticamente todas las áreas de investigación. El objetivo es diseñar un experimento para confirmar o rechazar una hipótesis. Los científicos de datos necesitan a menudo realizar experimentos sin interrupción, particularmente en lo que respecta a la interfaz de usuario y al marketing de productos. Este capítulo revisa el diseño experimental tradicional y analiza algunos desafíos clásicos en la ciencia de datos. También trata algunos conceptos frecuentemente citados en inferencia estadística y explica su significado y relevancia (o falta de relevancia) para la ciencia de datos.

Siempre que veamos referencias de significación estadística, pruebas t o valores p, ocurren generalmente en el contexto del “proceso” de inferencia estadística clásica (ver figura 3.1). Este proceso comienza con una hipótesis (“el medicamento A es mejor que el medicamento estándar existente” o “el precio A es más rentable que el precio B existente”). Un experimento (podría ser una prueba A/B) está diseñado para probar la hipótesis, diseñado de tal manera que, con suerte, arrojará resultados concluyentes. Los datos se recopilan y analizan y luego se extrae una conclusión. El término *inferencia* (*inference*) refleja la intención de aplicar los resultados del experimento, que involucran un conjunto limitado de datos, a un proceso o población más grande.

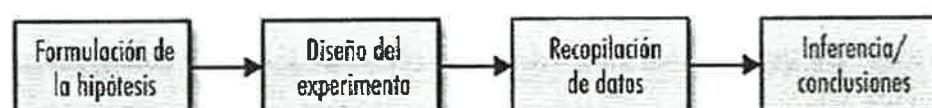


Figura 3.1 Proceso clásico de inferencia estadística.

Prueba A/B

Una prueba A/B es un experimento con dos grupos de sujetos para establecer cuál de los dos tratamientos, productos, procedimientos o similares es superior. A menudo, uno de los dos tratamientos es el tratamiento estándar existente que puede aplicarse o no. Si se utiliza un tratamiento estándar (o no), se le denomina *control* (*control*). Una hipótesis típica es que el nuevo tratamiento será mejor que el control.

Términos clave de la prueba A/B

Tratamiento

Algo (medicamento, precio, titular de la web) a lo que está expuesto un sujeto.

Grupo de tratamiento

Grupo de sujetos expuestos a un tratamiento específico.

Grupo de control

Grupo de sujetos no expuestos a ningún tratamiento (o al estándar).

Aleatorización

Proceso de asignación aleatoria de sujetos a tratamientos.

Sujetos

Elementos (visitantes de la web, pacientes, etc.) que están expuestos a tratamientos.

Estadística de la prueba

Métrica utilizada para medir el efecto del tratamiento.

Las pruebas A/B son habituales tanto en el diseño web como en marketing, ya que los resultados se evalúan muy fácilmente. Entre los ejemplos de pruebas A/B se pueden citar:

- Probar dos tratamientos de suelo para determinar cuál de ellos produce la mejor germinación de semillas.
- Probar dos terapias para determinar cuál de ellas elimina el cáncer más eficazmente.
- Probar dos precios para determinar cuál de ellos produce mayores ganancias netas.
- Probar dos titulares web para determinar con cuál de ellos se consiguen más clics (figura 3.2).
- Probar dos anuncios web para determinar cuál de ellos genera más conversiones.



Figura 3.2 Los especialistas en marketing prueban constantemente una presentación web frente a otra.

Una prueba A/B adecuada tiene *sujetos* (*subjects*) que se pueden asignar a un tratamiento u otro. El sujeto puede ser una persona, la semilla de una planta o un visitante de la web. La clave es que el sujeto esté expuesto al tratamiento. Idealmente, los sujetos se *asignan aleatoriamente* (*randomized*) (se asignan al azar) a los tratamientos. De esta manera, sabremos que cualquier diferencia entre los grupos de tratamiento se debe a una de dos cosas:

- El efecto de los diferentes tratamientos.
- Suerte en la extracción en la que los sujetos se asignan a un tratamiento (es decir, la asignación aleatoria puede haber dado por resultado que los sujetos con mejores habilidades se concentren en A o B).

También debemos prestar atención al *estadístico de prueba* (*test statistic*) o métrica que utilizamos para comparar el grupo A con el grupo B. Quizá la métrica más frecuente en la ciencia de datos es una variable binaria: hacer clic o no, comprar o no, si hay fraude o

no, etc. Esos resultados se resumirían en una tabla de 2×2 . La tabla 3.1 es una tabla de 2×2 para una prueba de precio real (consultar "Significación estadística y valores p" en la página 103 para ampliar la información sobre estos resultados).

Tabla 3.1 Tabla 2×2 de los resultados de experimentos de comercio electrónico

Resultado	Precio A	Precio B
Conversión	200	182
Sin conversión	23 539	22 406

Dependiendo de si la métrica es una variable continua (importe de la compra, ganancias, etc.) o un recuento (por ejemplo, días de ingreso en el hospital, páginas visitadas), el resultado se puede mostrar de manera diferente. Si uno no estuviera interesado en la conversión, sino en los ingresos por página vista, la salida típica de los resultados de la prueba de precios de la tabla 3.1 que proporciona por defecto el software se vería de la siguiente forma:

Revenue/page view with price A: mean = 3.87, SD = 51.10

Revenue/page view with price B: mean = 4.11, SD = 62.98

"SD" se refiere a la desviación estándar de los valores en cada grupo.



El hecho de que el software estadístico, incluidos *R* y *Python*, genere resultados por defecto no significa que todos los resultados sean útiles o relevantes. Podemos ver que las desviaciones estándar anteriores no son tan útiles. A primera vista, sugieren que numerosos valores pueden ser negativos, cuando los ingresos negativos no son viables. Estos datos constan de un pequeño conjunto de valores relativamente altos (páginas vistas con conversiones) y una gran cantidad de valores 0 (páginas vistas sin conversión). Es difícil resumir la variabilidad de tales datos con un simple número, aunque la desviación absoluta media de la media (7.68 para A y 8.15 para B) es más razonable que la desviación estándar.

¿Por qué tener un grupo de control?

¿Por qué no omitir el grupo de control y simplemente ejecutar un experimento aplicando el tratamiento que nos interesa a un solo grupo y comparar el resultado con la experiencia anterior?

Sin un grupo de control, no hay garantía de que "todas las demás cosas sean iguales" y que cualquier diferencia se deba realmente al tratamiento (o al azar). Cuando tenemos un grupo de control, está sujeto a las mismas condiciones (excepto al tratamiento que nos interesa) como grupo de tratamiento. Si simplemente hacemos una comparación con la experiencia previa o la "de partida", además del tratamiento, pueden diferir otros factores.



Estudios ciegos

El *estudio ciego* (*blind study*) es aquel en el que los sujetos no saben si están recibiendo el tratamiento A o el tratamiento B. El conocimiento de recibir un tratamiento en particular puede afectar la respuesta. El estudio *doble ciego* (*double-blind*) es aquel en el que los investigadores y facilitadores (por ejemplo, médicos y enfermeras en un estudio médico) tampoco saben qué sujetos están recibiendo qué tratamiento. El cegamiento no es posible cuando la naturaleza del tratamiento es transparente, por ejemplo, la terapia cognitiva de un ordenador frente a la de un psicólogo.

En ciencia de datos, las pruebas A/B se utilizan normalmente en un contexto web. Los tratamientos pueden ser el diseño de una página web, el precio de un producto, la redacción de un titular o algún otro artículo. Es necesario reflexionar un poco para preservar los principios de aleatorización. Normalmente, el sujeto del experimento es el visitante de la web, y los resultados que nos interesa medir son los clics, las compras, la duración de la visita, el número de páginas visitadas, si se visita una página en particular y demás. En un experimento A/B estándar, debemos decidir una métrica con anticipación. Se pueden obtener varias métricas de comportamiento y pueden ser de interés, pero si se espera que el experimento conduzca a una decisión entre el tratamiento A y el tratamiento B, es necesario establecer de antemano una métrica, o un *estadístico de prueba* (*test statistic*). La selección de un estadístico de prueba *después* (*after*) de realizar el experimento abre la puerta al sesgo por parte del investigador.

¿Por qué solo A/B? ¿Por qué no C, D, ...?

Las pruebas A/B son conocidas en el mundo del marketing y del comercio electrónico, pero están lejos de ser el único tipo de experimento estadístico. Se pueden incluir tratamientos adicionales. Se podrían tomar medidas repetidas a los sujetos. Los ensayos farmacéuticos, para los que no es fácil encontrar sujetos, además del gasto que supone su participación, se incorporan a lo largo del tiempo, por lo que a veces los ensayos se diseñan con varias posibilidades de detener el experimento y llegar a una conclusión.

Los diseños experimentales estadísticos tradicionales se centran en responder a la pregunta aclaratoria en relación con la eficacia de tratamientos específicos. Los científicos de datos están menos interesados en la pregunta:

¿Es la diferencia entre el precio A y el precio B estadísticamente significativa?

que en la pregunta:

¿Cuál de los diferentes precios posibles es el mejor?

Para ello, se utiliza un tipo de diseño experimental relativamente nuevo: *multi-arm bandit* (consultar "Algoritmo Multi-Arm Bandit" en la página 131).



Obtención de permisos

En la investigación científica y médica con personas, normalmente es necesario que estas otorguen su permiso, así como la aprobación por parte de una junta de revisión institucional. En los experimentos que se realizan en el mundo de los negocios como parte de las operaciones en curso, casi nunca se cumplen estos requisitos. En la mayoría de los casos (por ejemplo, experimentos de precios o experimentos sobre qué título mostrar o qué oferta debe hacerse), generalmente se acepta esta práctica. Facebook, sin embargo, entró en conflicto con esta aceptación general en 2014 cuando experimentó con el tono emocional en las fuentes de noticias de los usuarios. Facebook usó el análisis del sentimiento para clasificar las publicaciones de noticias como positivas o negativas, y luego alteró el balance positivo/negativo en lo que mostraba a los usuarios. Algunos usuarios seleccionados al azar experimentaron con noticias más positivas, mientras que otros experimentaron con noticias más negativas. Facebook descubrió que los usuarios a los que se les presentaron noticias más positivas tenían más probabilidades de publicar positivamente ellos mismos, y viceversa. Aunque la magnitud del efecto fue pequeña, Facebook tuvo que enfrentarse a muchas críticas por realizar el experimento sin el conocimiento de los usuarios. Algunos usuarios especularon con que Facebook podría haber empujado al límite a los usuarios que habiendo recibido la versión negativa de sus feed pudieran haber experimentado profundas depresiones.

Ideas clave

- Los sujetos se asignan a dos (o más) grupos que se tratan exactamente igual, excepto en que el tratamiento en estudio difiere de un grupo a otro.
- Idealmente, los sujetos se asignan al azar a los grupos.

Lecturas complementarias

- Las comparaciones de dos grupos (pruebas A/B) son un elemento básico de la estadística tradicional, y casi cualquier texto introductorio de estadística tratará ampliamente los principios de diseño y los procedimientos de inferencia. Para leer una discusión que coloca las pruebas A/B en un contexto más de ciencia de datos y usa remuestreo, consultar *Introductory Statistics and Analytics: A Resampling Perspective* de Peter Bruce (Wiley, 2014).
- Para las pruebas web, los aspectos logísticos de las pruebas pueden ser tan desafiantes como los estadísticos. Un buen lugar para comenzar es la sección de ayuda de Google Analytics sobre experimentos (<https://marketingplatform.google.com/about/optimize/>).

- Hay que tener cuidado con los consejos que se encuentran en las guías omnipresentes para las pruebas A/B que vemos en la web, como el siguiente contenido extraído de estas guías: "Espere a tener aproximadamente 1000 visitantes en total y asegúrese de ejecutar la prueba durante una semana". Estas reglas generales no son útiles. Consultar "Potencia y tamaño de la muestra" en la página 135 para ampliar detalles.

Pruebas de hipótesis

Las pruebas de hipótesis, también llamadas *pruebas significativas* (*significance tests*), están omnipresentes en el análisis estadístico tradicional de las publicaciones de investigación. Su propósito es ayudarnos a saber si el azar puede ser responsable de un efecto que hayamos observado.

Términos clave de las pruebas de hipótesis

Hipótesis nula

Hipótesis de que el azar es el responsable.

Hipótesis alternativa

Contrapunto a la hipótesis nula (lo que esperamos probar).

Prueba unidireccional

Prueba de hipótesis que cuenta los resultados aleatorios solo en un sentido.

Prueba bidireccional

Prueba de hipótesis que cuenta los resultados aleatorios en los dos sentidos.

Una prueba A/B (consultar "Prueba A/B" en la página 88) generalmente se elabora con una hipótesis en mente. Por ejemplo, la hipótesis podría ser que el precio B produce un mayor beneficio. ¿Por qué necesitamos una hipótesis? ¿Por qué no simplemente observar el resultado del experimento y seguir con el tratamiento que funcione mejor?

La respuesta radica en la tendencia de la mente a subestimar el alcance del comportamiento aleatorio natural. Una manifestación de esta circunstancia es la incapacidad de anticipar eventos extremos, o los también llamados "cisnes negros" (consultar "Distribuciones de cola larga" en la página 73). Otra manifestación es la tendencia a malinterpretar eventos aleatorios como si tuvieran patrones con algún significado. La prueba de hipótesis estadística se ideó como un medio de proteger a los investigadores frente a los engaños del azar.

Interpretación errónea de la aleatoriedad

El siguiente experimento nos permite observar la tendencia que tenemos a subestimar el fenómeno de la aleatoriedad. Pedimos a varios amigos que piensen en una serie de 50 lanzamientos de una moneda. Les pedimos que escriban una serie de resultados cara y cruz al azar. Luego les pedimos que lancen realmente la moneda 50 veces y anoten los resultados. Les pedimos que escriban los resultados reales del lanzamiento de la moneda en una columna y los resultados inventados en otra. Es fácil saber qué resultados son reales: las reales tendrán series más largas de cara y cruz. En un conjunto de 50 lanzamientos *reales* (*real*), no es nada inusual ver cinco o seis caras o cruces seguidas. Sin embargo, cuando la mayoría de nosotros ideamos lanzamientos de monedas al azar y hemos anotado tres o cuatro caras seguidas, nos decimos a nosotros mismos que, para que la serie parezca aleatoria, es mejor cambiar a cruces.

La otra cara de esta moneda, por así decirlo, es que cuando vemos el equivalente en el mundo real de seis caras seguidas (por ejemplo, cuando un titular supera a otro en un 10 %), nos inclinamos a atribuirlo a algo real, no solo al azar.

En una prueba A/B correctamente diseñada, recopilamos datos sobre los tratamientos A y B de tal manera que cualquier diferencia observada entre A y B debe tener su origen en:

- El azar en la asignación de los sujetos
- Una verdadera diferencia entre A y B

Una prueba de hipótesis estadística es un análisis adicional de una prueba A/B, o de cualquier experimento aleatorio, para evaluar si el azar es una explicación razonable de la diferencia observada entre los grupos A y B.

La hipótesis nula

Las pruebas de hipótesis utilizan la siguiente lógica: "Dada la tendencia que tenemos a reaccionar ante un comportamiento inusual pero aleatorio e interpretarlo como algo significativo y real, en los experimentos necesitaremos pruebas de que la diferencia entre grupos es más extrema de lo que razonablemente podría ocasionar el azar". Esto implica una suposición inicial de que los tratamientos son equivalentes y que cualquier diferencia entre los grupos se debe al azar. Este supuesto de referencia se denomina *hipótesis nula* (*null hypothesis*). Nuestra esperanza, entonces, es que de hecho podamos probar que la hipótesis nula es *incorrecta* (*wrong*) y mostrar que las diferencias entre los resultados para los grupos A y B son mayores de las que podría ocasionar el azar.

Una forma de hacerlo es a través de un procedimiento de permutación de remuestreo. En este procedimiento mezclamos los resultados de los grupos A y B y luego distribuimos

repetidamente los datos en grupos de tamaños similares, y a continuación observamos con qué frecuencia obtenemos una diferencia tan extrema como la diferencia observada. Los resultados mezclados y combinados de los grupos A y B, y el procedimiento de remuestreo de los mismos, encarnan la hipótesis nula de que los grupos A y B son equivalentes e intercambiables y se denomina modelo nulo. Consultar "Remuestreo" en la página 96 para ampliar detalles.

Hipótesis alternativa

Las pruebas de hipótesis, por su naturaleza, implican no solo una hipótesis nula, sino también una hipótesis alternativa de compensación. Estos son algunos ejemplos:

- Nula = "sin diferencia entre las medias del grupo A y del grupo B". Alternativa = "A es diferente de B" (podría ser más grande o más pequeño).
- Nula = " $A \leq B$ ". Alternativa = " $A > B$ ".
- Nula = " B no es un X % mayor que A ". Alternativa = " B es X % mayor que A ".

En conjunto, la hipótesis nula y la alternativa deben tener en cuenta todas las posibilidades. La naturaleza de la hipótesis nula determina la estructura de la prueba de la hipótesis.

Pruebas de hipótesis unidireccionales o bidireccionales

A menudo, en una prueba A/B, probamos una nueva opción (por ejemplo, B) con una opción predeterminada establecida (A), y se presume que nos quedaremos con la opción predeterminada a menos que la nueva opción demuestre ser definitivamente mejor. En tal caso, desearemos una prueba de hipótesis que nos proteja frente a los engaños del azar en la dirección que favorece a B. No nos importa que nos engañe el azar en la otra dirección, porque nos quedaríamos con A a menos que B demuestre definitivamente ser mejor. Entonces queremos una hipótesis direccional (*directional*) alternativa (B es mejor que A). En tal caso, utilizamos una prueba de hipótesis *unidireccional* (*one-way*) (o de una cola). Esto significa que los resultados de probabilidad extrema en solo una dirección cuentan para el valor p.

Si necesitamos que una prueba de hipótesis nos proteja de los engaños del azar en cualquier dirección, la hipótesis alternativa es *bidireccional* (*bidirectional*) (A es diferente de B, podría ser más grande o más pequeño). En tal caso, utilizamos una hipótesis *bidireccional* (*two-way*) (o de dos colas). Esto significa que los resultados de probabilidad extrema en cualquier dirección cuentan para el valor p.

La prueba de hipótesis de una cola a menudo se ajusta a la naturaleza de la toma de decisiones A/B, en la que se requiere una decisión. A una opción generalmente se le asigna el estado "predeterminado" a menos que la otra resulte mejor. Sin embargo, el software, incluidos R y scipy en Python, generalmente proporciona una prueba de dos

colas como resultado por defecto, y muchos estadísticos optan por la prueba de dos colas, más conservadora, solo para evitar el debate. Elegir entre una cola o dos colas es un tema confuso, y no es tan relevante para la ciencia de datos, donde la precisión de los cálculos del valor p no es muy importante.

Ideas clave

- La hipótesis nula es una construcción lógica que incorpora la noción de que no ha sucedido nada especial y que cualquier efecto que observemos se debe al azar.
- La prueba de hipótesis asume que la hipótesis nula es verdadera, crea un "modelo nulo" (un modelo de probabilidad) y prueba si el efecto que observamos es un resultado razonable de ese modelo.

Lecturas complementarias

- *The Drunkard's Walk* de Leonard Mlodinow (Pantheon, 2008) es un estudio interesante de las formas en que "la aleatoriedad gobierna nuestras vidas".
- El texto clásico sobre estadística de David Freedman, Robert Pisani y Roger Purves *Statistics*, 4.^a ed. (W. W. Norton, 2007), tiene excelentes desarrollos no matemáticos de la mayoría de los temas de estadística, incluida la prueba de hipótesis.
- *Introductory Statistics and Analytics: A Resampling Perspective* de Peter Bruce (Wiley, 2014) desarrolla conceptos de prueba de hipótesis utilizando remuestreo.

Remuestreo

El *remuestreo (resampling)* en estadística significa muestrear valores de forma repetida a partir de los datos observados, con el objetivo general de evaluar la variabilidad aleatoria en un estadístico. También se puede utilizar para evaluar y mejorar la precisión de algunos modelos de aprendizaje automático (por ejemplo, los pronósticos de los modelos de árboles de decisión construidos en diferentes conjuntos de datos sometidos a bootstrap se pueden promediar en un proceso conocido como *bagging*. Consultar "Métodos de bagging y bosque aleatorio" en la página 259).

Hay dos tipos principales de procedimientos de remuestreo: las pruebas de *bootstrap* (*bootstrap*) y de *permutación* (*permutation*). Bootstrap se utiliza para evaluar la confiabilidad de una estimación. Se analizó en el capítulo anterior (consultar "Bootstrap" en la página 61). Las pruebas de permutación, que tratamos en esta sección, se utilizan para probar hipótesis, que generalmente involucran a dos o más grupos.

Términos clave del remuestreo

Prueba de permutación

Procedimiento de combinar dos o más muestras y reasignar aleatoriamente (o con exhaustividad) las observaciones a nuevas muestras.

Sinónimos

prueba de aleatorización, prueba de permutación aleatoria, prueba exacta.

Remuestreo

Extracción de muestras adicionales ("muestras repetidas") de un conjunto de datos observados.

Con y sin reposición

En el muestreo, si un artículo se devuelve o no a la muestra antes de la siguiente extracción.

Prueba de permutación

En un procedimiento de *permutación* (*permutation*), se involucran dos o más muestras, generalmente los grupos en una prueba A/B u otra prueba de hipótesis. *Permutar* (*permute*) significa cambiar el orden de un conjunto de valores. El primer paso en una *prueba de permutación* (*permutation test*) de una hipótesis es combinar los resultados de los grupos A y B (y, si se usa, C, D, ...). Esta es la materialización lógica de la hipótesis nula de que los tratamientos a los que han estado expuestos los grupos no difieren. Luego probamos esa hipótesis extrayendo grupos al azar de este conjunto agrupado y observamos cuánto se diferencian entre sí. El procedimiento de permutación es el siguiente:

1. Agrupamos en un solo conjunto de datos los resultados de los diferentes grupos.
2. Mezclamos los datos agrupados y luego extraemos aleatoriamente (sin reposición) una nueva muestra del mismo tamaño que el grupo A (claramente contendrá algunos datos de los otros grupos).
3. De los datos restantes, extraemos al azar (sin reposición) una nueva muestra del mismo tamaño que el grupo B.
4. Hacemos lo mismo para los grupos C, D, etc. Ahora hemos recopilado un conjunto de muestras repetidas que reflejan los tamaños de las muestras originales.
5. Cualquiera que sea el estadístico o la estimación que se ha calculado para las muestras originales (por ejemplo, diferencia en las proporciones de los grupos), lo calculamos ahora para las nuevas muestras y lo registramos. Este proceso constituye una de las iteraciones de la permutación.
6. Repetimos los pasos anteriores R veces para obtener la distribución de permutación del estadístico de prueba.

Ahora regresamos a la diferencia observada entre grupos y la comparamos con el conjunto de diferencias obtenidas de la permutación. Si la diferencia observada se encuentra dentro del conjunto de diferencias obtenidas de la permutación, entonces no hemos probado nada. La diferencia observada está dentro del rango de lo que podría producir el azar. Sin embargo, si la diferencia observada se encuentra fuera de la mayor parte de la distribución de permutación, entonces concluimos que el azar no es el responsable. En términos técnicos, la diferencia es *estadísticamente significativa (statistically significant)*. (Consultar “Significación estadística y valores p” en la página 103).

Ejemplo: adherencia de la web

Una empresa que vende un servicio de valor relativamente alto quiere probar cuál de las dos presentaciones web vende mejor. Debido al alto valor del servicio, las ventas son poco frecuentes y el ciclo de ventas es largo. Se emplearía demasiado tiempo en acumular suficientes ventas para saber qué presentación es superior. Entonces la empresa decide medir los resultados con una variable proxy, utilizando la página interior detallada que describe el servicio.



La variable proxy es aquella que representa la verdadera variable que nos interesa, que puede no estar disponible, ser demasiado cara o emplear demasiado tiempo en medirla. En la investigación del clima, por ejemplo, el contenido de oxígeno de núcleos de hielo de tiempos remotos se utiliza como un proxy de la temperatura. Es útil tener al menos *algunos (some)* datos sobre la verdadera variable que nos interesa, de modo que se pueda evaluar la intensidad de su asociación con el proxy.

Una posible variable proxy para nuestra empresa es el número de clics en la página de destino detallada. Otra mejor es cuánto tiempo pasan los visitantes en la página. Es razonable pensar que una presentación web (página) queatraiga la atención de los visitantes durante más tiempo generará más ventas. Por lo tanto, nuestra métrica es el tiempo promedio de la sesión, comparando la página A con la B.

Debido al hecho de que se trata de una página interior con un propósito especial, no recibe una gran cantidad de visitantes. También hay que tener en cuenta que Google Analytics, que es la forma en que medimos el tiempo promedio en la página, no puede medir el tiempo dedicado a la última página dentro de una sesión. Google Analytics establecerá como cero el tiempo dedicado a la última página de una sesión, a menos que el usuario interactúe con la página. Por ejemplo, que haga clics o se desplace. Este también es el caso de las sesiones de una sola página. Los datos requieren un procesamiento adicional para tener esto en cuenta. El resultado es un total de 36 sesiones para las dos presentaciones diferentes, 21 para la página A y 15 para la página B. Mediante `ggplot`, podemos comparar visualmente los tiempos de las sesiones usando diagramas de caja, uno al lado del otro:

```
ggplot(session_times, aes(x=Page, y=Time)) +  
  geom_boxplot()
```

El comando `boxplot` de pandas utiliza el argumento de palabra clave `by` para crear la figura:

```
ax = session_times.boxplot(by='Page', column='Time')
ax.set_xlabel("")
ax.set_ylabel('Time (in seconds)')
plt.suptitle("")
```

El diagrama de caja, que se muestra en la figura 3.3, indica que la página B tiene sesiones más largas que la página A. Las medias para cada grupo se pueden calcular en *R* de la siguiente manera:

```
mean_a <- mean(session_times[session_times['Page'] == 'Page A', 'Time'])
mean_b <- mean(session_times[session_times['Page'] == 'Page B', 'Time'])
mean_b - mean_a
[1] 35.66667
```

En *Python*, filtramos el marco de datos de pandas primero por página y luego determinamos la media de la columna `Time`:

```
mean_a = session_times[session_times.Page == 'Page A'].Time.mean()
mean_b = session_times[session_times.Page == 'Page B'].Time.mean()
mean_b - mean_a
```

La página B tiene tiempos de sesión superiores a los de la página A en 35.67 segundos, en promedio. La pregunta es si esta diferencia está dentro del rango de lo que podría producir el azar aleatorio, es decir, si es estadísticamente significativa. Una forma de responder a esta pregunta es aplicar una prueba de permutación: agrupamos todos los tiempos de sesión y luego los mezclamos repetidamente y los dividimos en grupos de 21 (recordemos que $n_A = 21$ para la página A) y 15 ($n_B = 15$ para la página B).

Para aplicar una prueba de permutación, necesitamos una función para asignar aleatoriamente los 36 tiempos de sesión a un grupo de 21 (página A) y un grupo de 15 (página B). La versión *R* de esta función es:

```
perm_fun <- function(x, nA, nB)
{
  n <- nA + nB
  idx_b <- sample(1:n, nB)
  idx_a <- setdiff(1:n, idx_b)
  mean_diff <- mean(x[idx_b]) - mean(x[idx_a])
  return(mean_diff)
}
```

La versión de *Python* de esta prueba de permutación es la siguiente:

```
def perm_fun(x, nA, nB):
    n = nA + nB
    idx_B = set(random.sample(range(n), nB))
    idx_A = set(range(n)) - idx_B
    return x.loc[idx_B].mean() - x.loc[idx_A].mean()
```

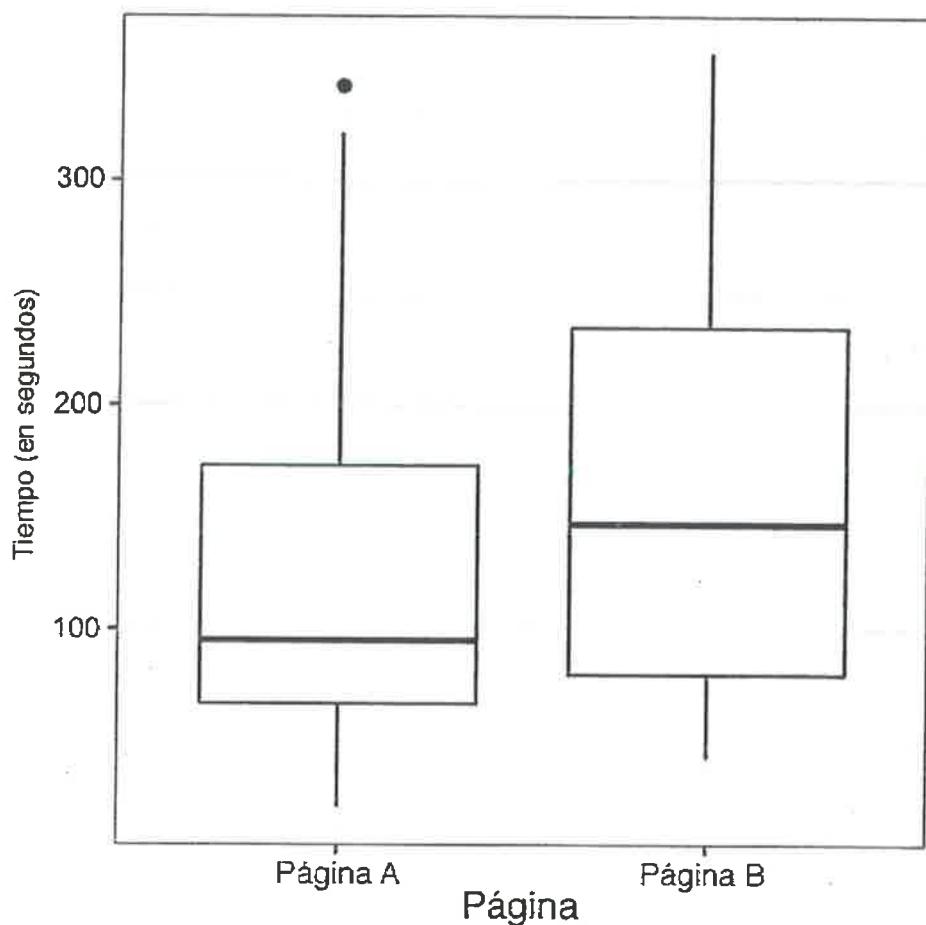


Figura 3.3 Tiempos de sesión para las páginas web A y B.

Esta función actúa muestreando (sin reposición) los índices n_B y los asigna al grupo B. Los índices n_A restantes se asignan al grupo A. Se devuelve la diferencia entre las dos medias. Llamamos a esta función $R = 1000$ veces y especificamos $n_A = 21$ y $n_B = 15$, lo que lleva a una distribución de diferencias en los tiempos de sesión que se puede trazar como un histograma. En R, esto se hace de la siguiente manera usando la función `hist`:

```
perm_diffs <- rep(0, 1000)
for (i in 1:1000) {
  perm_diffs[i] = perm_fun(session_times[, 'Time'], 21, 15)
}
hist(perm_diffs, xlab='Session time differences (in seconds)')
abline(v=mean_b - mean_a)
```

En Python, podemos crear un gráfico similar usando `matplotlib`:

```
perm_diffs = [perm_fun(session_times.Time, nA, nB) for _ in range(1000)]

fig, ax = plt.subplots(figsize=(5, 5))
ax.hist(perm_diffs, bins=11, rwidth=0.9)
ax.axvline(x = mean_b - mean_a, color='black', lw=2)
```

Experimentos estadísticos y pruebas significativas

```
ax.text(50, 190, 'Observed\ndifference', bbox={'facecolor':'white'})  
ax.set_xlabel('Session time differences (in seconds)')  
ax.set_ylabel('Frequency')
```

El histograma de la figura 3.4 muestra que la diferencia media de las permutaciones aleatorias a menudo excede la diferencia observada en los tiempos de sesión (la línea vertical). Para nuestros resultados, esto sucede en el 12.6 % de los casos:

```
mean(perm_diffs > (mean_b - mean_a))  
0.126
```

Como la simulación utiliza números aleatorios, el porcentaje variará. Por ejemplo, en la versión de *Python*, obtuvimos el 12.1 %:

```
np.mean(perm_diffs > mean_b - mean_a)  
0.121
```

Esto sugiere que la diferencia observada en el tiempo de sesión entre la página A y la página B está dentro del rango de variación debida al azar y, por lo tanto, no es estadísticamente significativa.

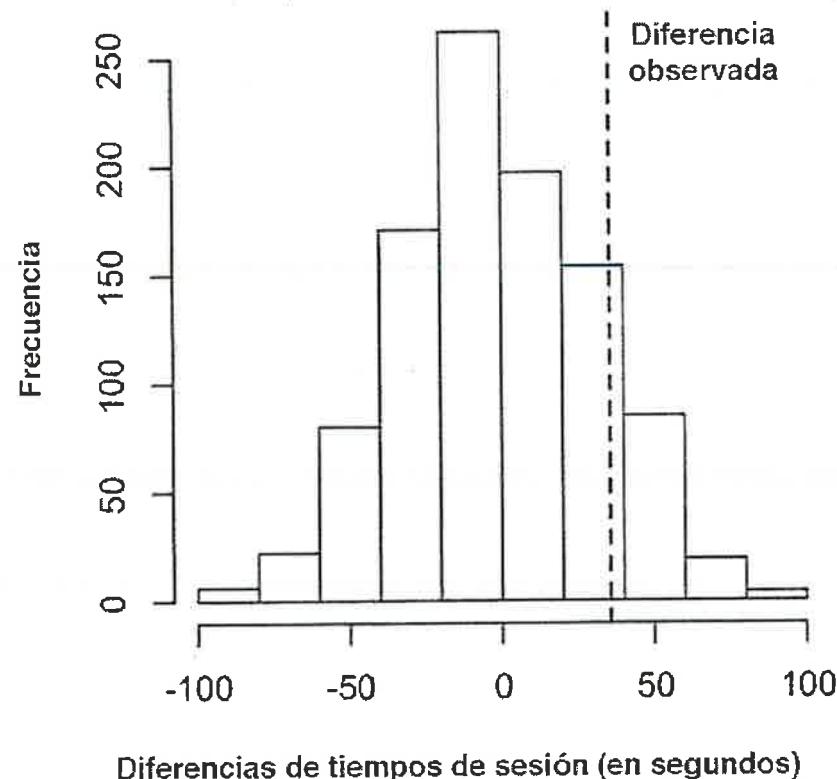


Figura 3.4 Distribución de frecuencias para las diferencias de tiempos de sesión entre las páginas A y B. La línea vertical muestra la diferencia observada.

Pruebas de permutación exhaustiva y de bootstrap

Además del procedimiento de mezcla aleatorio anterior, también llamado *prueba de permutación aleatoria (random permutation test)* o prueba de aleatorización, existen dos variantes de la prueba de permutación:

- *Prueba de permutación exhaustiva (exhaustive permutation test)*
- *Prueba de permutación bootstrap (bootstrap permutation test)*

En una prueba de permutación exhaustiva, en lugar de simplemente mezclar y dividir los datos al azar, en realidad descubrimos todas las formas posibles en que podrían dividirse. Esto resulta práctico solo para tamaños de la muestra relativamente pequeños. Con un gran número de repetición de mezclas, los resultados de la prueba de permutación aleatoria se aproximan a los de la prueba de permutación exhaustiva y se acercan a ellos en el límite. Las pruebas de permutación exhaustivas también se denominan a veces *pruebas exactas (exact tests)*, debido a su propiedad estadística de garantizar que el modelo nulo solo se probará como "significativo" para valores del nivel alfa de la prueba (consultar "Significación estadística y valores p" en la página 103).

En una prueba de permutación bootstrap, las extracciones descritas en los pasos 2 y 3 de la prueba de permutación aleatoria se realizan *con reposición (with replacement)* en lugar de hacerlo sin ella. De esta manera, el procedimiento de remuestreo modela no solo el elemento aleatorio en la asignación del tratamiento a un sujeto, sino también al elemento aleatorio en la selección de sujetos de una población. Ambos procedimientos se encuentran en los estadísticos, y la distinción entre ellos es algo complicada, pero lo anterior no tiene consecuencias en la práctica de la ciencia de datos.

Pruebas de permutación: el resultado final de la ciencia de datos

Las pruebas de permutación son procedimientos heurísticos útiles para explorar el papel de la variación aleatoria. Es relativamente fácil codificarlas, interpretarlas y explicarlas, y ofrecen un útil rodeo con el que se evita el formalismo y el "falso determinismo" de las estadísticas basadas en fórmulas, en las que la precisión de las "respuestas" de las fórmulas tiende a implicar una certeza injustificada.

Una virtud del remuestreo, en contraste con los enfoques de fórmulas, es que se acerca mucho más a un enfoque de inferencia de talla única. Los datos pueden ser numéricos o binarios. Los tamaños de las muestras pueden ser iguales o diferentes. No se necesitan suposiciones sobre datos distribuidos normalmente.

Ideas clave

- En una prueba de permutación, se agrupan varias muestras y luego se mezclan.
- Los valores mezclados se dividen en nuevas muestras y se calcula el estadístico que nos interesa.
- Luego se repite este proceso y se tabula el estadístico que se ha muestreado de forma repetida.
- La comparación del valor observado del estadístico con la distribución muestreada de forma repetida nos permite juzgar si podría ocurrir por azar la diferencia observada entre las muestras.

Lecturas complementarias

- *Randomization Tests*, 4.^a ed., de Eugene Edgington y Patrick Onghena (Chapman & Hall/CRC Press, 2007), pero no nos dejemos llevar por la maraña del muestreo no aleatorio.
- *Introductory Statistics and Analytics: A Resampling Perspective* de Peter Bruce (Wiley, 2014).

Significación estadística y valores p

La significación estadística es la forma en que los estadísticos miden si un experimento (o incluso el estudio de datos existentes) produce un resultado más extremo que el que podría producir el azar. Si el resultado está más allá del ámbito de la variación debida al azar, se dice que es estadísticamente significativo.

Términos clave de significación estadística y valores p

Valor p

Dado un modelo probabilístico que incorpora la hipótesis nula, el valor p es la probabilidad de obtener resultados tan inusuales o extremos como los resultados observados.

Alfa

Umbral de probabilidad de "rareza" que los resultados aleatorios deben superar para que los resultados reales se consideren estadísticamente significativos.

Error de tipo 1

Conclusión errónea de que un efecto es real (cuando se debe al azar).

Error de tipo 2

Conclusión errónea de que un efecto se debe al azar (cuando es real).

Estadística práctica para ciencia de datos con R y Python

Presentamos en la tabla 3.2 los resultados de la prueba web mostrada anteriormente.

Tabla 3.2 Tabla 2 × 2 con los resultados de experimentos de comercio electrónico

Resultado	Precio A	Precio B
Conversión	200	182
Sin conversión	23 539	22 406

El precio A convierte casi un 5 % mejor que el precio B ($0.8425\% = 200 / [23\,539 + 200] * 100$, frente a $0.8057\% = 182 / [22\,406 + 182] * 100$; una diferencia de 0.0368 puntos porcentuales), lo suficientemente grande como para ser significativa en un negocio de alto volumen. Tenemos en este caso más de 45 000 puntos de datos, y es tentador considerar esta cifra como "grandes datos", que no requieren pruebas significativas estadísticas (necesarias principalmente para tener en cuenta la variabilidad del muestreo en muestras pequeñas). Sin embargo, los porcentajes de conversión son tan bajos (menos del 1 %) que los valores significativos reales (las conversiones) están solo en el orden de las centenas, y el tamaño que se necesita de la muestra está realmente determinado por estas conversiones. Podemos probar si la diferencia en las conversiones entre los precios A y B está dentro del rango de *variación debida al azar* (*chance variation*), utilizando un procedimiento de remuestreo. Por variación debida al azar, nos referimos a la variación aleatoria producida por un modelo probabilístico que incorpora la hipótesis nula de que no hay diferencia entre los porcentajes de conversión (consultar "La hipótesis nula" en la página 94). En el siguiente procedimiento de permutación preguntamos: "Si los dos precios comparten el mismo porcentaje de conversión, ¿podría la variación debida al azar producir una diferencia tan grande como el 5 %?".

1. Colocamos las tarjetas etiquetadas con 1 y 0 en una casilla: esto representa el supuesto porcentaje de conversión compartido de $382 / 45\,945 = 0.008246 = 0.8246\%$.
2. Mezclamos y extraemos una muestra repetida del tamaño de 23 739 (el mismo n que el precio A) y registramos el número de unos.
3. Registraremos el número de unos en las 22 588 restantes (el mismo n que el precio B).
4. Registraremos la diferencia en proporción de unos.
5. Repetimos los pasos del 2 al 4.
6. ¿Con qué frecuencia la diferencia ha sido ≥ 0.0368 ?

Si reutilizamos la función `perm_fun` definida en "Ejemplo: adherencia de la web" en la página 98, podemos crear en *R* un histograma de las diferencias permutadas aleatoriamente en el porcentaje de conversión:

```
obs_pct_diff <- 100 * (200 / 23739 - 182 / 22588)
conversion <- c(rep(0, 45945), rep(1, 382))
perm_diffs <- rep(0, 1000)
```

Experimentos estadísticos y pruebas significativas

```
for (i in 1:1000) {  
  perm_diffs[i] = 100 * perm_fun(conversion, 23739, 22588)  
}  
hist(perm_diffs, xlab='Conversion rate (percent)', main="")  
abline(v=obs_pct_diff)
```

El correspondiente código Python es:

```
obs_pct_diff = 100 * (200 / 23739 - 182 / 22588)  
print(f'Observed difference: {obs_pct_diff:.4f}%')  
conversion = [0] * 45945  
conversion.extend([1] * 382)  
conversion = pd.Series(conversion)  
  
perm_diffs = [100 * perm_fun(conversion, 23739, 22588)  
             for _ in range(1000)]  
  
fig, ax = plt.subplots(figsize=(5, 5))  
ax.hist(perm_diffs, bins=11, rwidth=0.9)  
ax.axvline(x=obs_pct_diff, color='black', lw=2)  
ax.text(0.06, 200, 'Observed\\ndifference', bbox={'facecolor':'white'})  
ax.set_xlabel('Conversion rate (percent)')  
ax.set_ylabel('Frequency')
```

Veamos el histograma de 1000 resultados con remuestreo en la figura 3.5: como sucede en este caso, la diferencia observada de 0.0368 % está dentro del rango de variación debida al azar.

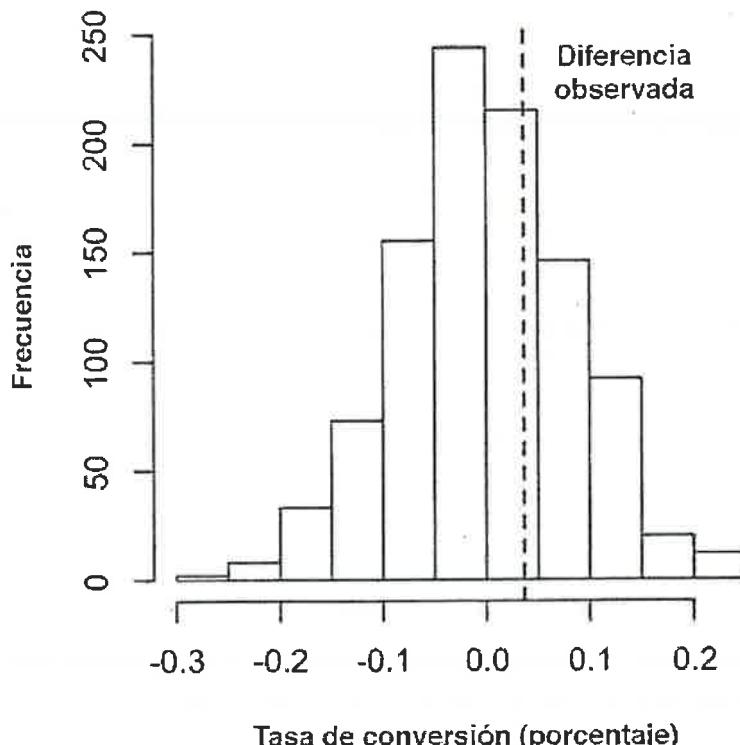


Figura 3.5 Distribución de frecuencias para la diferencia en los porcentajes de conversión entre los precios A y B.

Valor p

El simple hecho de observar el gráfico no es una forma muy precisa de medir la significación estadística, por lo que resulta más interesante utilizar el *valor p (p-value)*. Es la frecuencia con la que el modelo probabilístico genera un resultado más extremo que el resultado observado. Podemos estimar el valor p a partir de la prueba de permutación observando la proporción de veces que el resultado de dicha prueba es una diferencia igual o mayor que la diferencia observada:

```
mean(perm_diffs > obs_pct_diff)
[1] 0.308

np.mean([diff > obs_pct_diff for diff in perm_diffs])
```

Aquí, tanto *R* como *Python* usan el hecho de que verdadero se interpreta como 1 y falso como 0.

El valor p es 0.308, lo que significa que esperaríamos lograr un resultado tan extremo como este, o un resultado más extremo, como consecuencia del azar durante el 30 % de las veces.

En este caso, no necesitamos utilizar la prueba de permutación para obtener el valor p. Dado que tenemos una distribución binomial, podemos aproximar el valor p. En el código *R*, lo hacemos usando la función *prop.test*:

```
> prop.test(x=c(200, 182), n=c(23739, 22588), alternative='greater')
 2-sample test for equality of proportions with continuity correction
data: c(200, 182) out of c(23739, 22588)
X-squared = 0.14893, df = 1, p-value = 0.3498
alternative hypothesis: greater
95 percent confidence interval:
 -0.001057439 1.000000000
sample estimates:
    prop 1     prop 2 
0.008424955 0.008057376
```

El argumento *x* es el número de éxitos de cada grupo y el argumento *n* es el número de intentos.

El método *scipy.stats.chi2_contingency* proporciona los valores que se muestran en la tabla 3.2:

```
survivors = np.array([[200, 23739 - 200], [182, 22588 - 182]])
chi2, p_value, df, _ = stats.chi2_contingency(survivors)

print(f'p-value for single sided test: {p_value / 2:.4f}')
```

La aproximación normal arroja un valor p de 0.3498, que está cerca del valor p obtenido en la prueba de permutación.

Alfa

Los estadísticos desaprueban la práctica de dejar a discreción del investigador determinar si un resultado es "demasiado inusual" para que ocurra por casualidad. Más bien, se especifica un umbral de antemano, como "más extremo que el 5 % de los resultados consecuencia del azar (hipótesis nula)". Este umbral se conoce como *alfa* (*alpha*). Los niveles alfa típicos son el 5 % y el 1 %. Cualquier nivel elegido es una decisión arbitraria. No hay nada en el proceso que garantice decisiones correctas el x % del tiempo. Esto se debe a que la pregunta sobre probabilidad que se responde *no* es: "¿Cuál es la probabilidad de que esto sucediera por casualidad?", sino más bien: "Dado un modelo de probabilidad, ¿cuál es la probabilidad de que ocurra un resultado tan extremo?". Luego deducimos hacia atrás sobre la idoneidad del modelo probabilístico, pero ese juicio no conlleva una probabilidad. Este punto ha sido objeto de mucha confusión.

Controversia sobre el valor p

Una controversia considerable ha rodeado el uso del valor p en los últimos años. Una revista de psicología ha llegado a "prohibir" el uso del valor p en los artículos presentados con el argumento de que las decisiones de publicación basadas únicamente en el valor p estaban dando como resultado la publicación de una investigación deficiente. Demasiados investigadores, solo vagamente conscientes de lo que realmente significa el valor p, buscan en los datos y entre las diferentes hipótesis posibles a comprobar, hasta que encuentran una combinación que genera un valor p significativo y, por lo tanto, un artículo apto para su publicación.

El problema real es que el usuario quiere que el valor p tenga un mayor significado del que tiene. Esto es lo que nos *gustaría* (*like*) que expresara el valor p:

La probabilidad de que el resultado se deba al azar.

Esperamos un valor bajo, por lo que podemos concluir que hemos demostrado algo. Así es como muchos editores de revistas estaban interpretando el valor p. Pero esto es lo que *realmente* (*actually*) representa el valor p:

La probabilidad de que, dado un modelo probabilístico (*given a chance model*), se produzcan resultados tan extremos como los resultados observados.

La diferencia es sutil, pero real. Un valor p significativo no nos lleva tan lejos en el camino hacia la "prueba" como parece prometer. El fundamento lógico para la conclusión "estadísticamente significativo" es algo más débil cuando se comprende el significado real del valor p.

En marzo de 2016, la American Statistical Association, después de mucha deliberación interna, mostró el grado de malentendido acerca de los valores p emitiendo una declaración de advertencia sobre su uso. La declaración de la ASA (<https://amstat.tandfonline.com/doi/full/10.1080/>) destacó seis principios para los investigadores y editores de revistas:

Estadística práctica para ciencia de datos con R y Python

1. Los valores p pueden indicar lo incompatibles que son los datos con un modelo estadístico específico.
2. Los valores p no miden la probabilidad de que la hipótesis estudiada sea cierta, o la probabilidad de que los datos se hayan producido únicamente al azar.
3. Las conclusiones científicas y las decisiones comerciales o políticas no deben basarse únicamente en si un valor p supera un umbral específico.
4. Una inferencia adecuada requiere informes y transparencia completos.
5. El valor p, o la significación estadística, no miden el tamaño de un efecto o la importancia de un resultado.
6. Por sí mismo, el valor p no proporciona una buena medida de la evidencia con respecto a un modelo o hipótesis.

Significación práctica

Que un resultado sea estadísticamente significativo, no quiere decir que tenga importancia práctica. Una pequeña diferencia que no tiene una significación práctica puede ser estadísticamente significativa si surge de muestras suficientemente grandes. Las muestras grandes aseguran que los efectos pequeños y no significativos puedan, no obstante, ser lo suficientemente grandes como para descartar el azar como explicación. Descartar el azar no hace que mágicamente sea importante un resultado que, en esencia, carece de importancia.

Errores de tipo 1 y 2

Al evaluar la significación estadística, son posibles dos tipos de errores:

- Un error de tipo 1, en el que se concluye erróneamente que un efecto es real, cuando en realidad se debe al azar.
- Un error de tipo 2, en el que se concluye erróneamente que un efecto no es real (es decir, debido al azar), cuando en realidad es real.

De hecho, un error de tipo 2 no es tanto un error como un juicio de que el tamaño de la muestra es demasiado pequeño para detectar el efecto. Cuando un valor p no alcanza la significación estadística (por ejemplo, supera el 5 %), lo que realmente estamos diciendo es "efecto no probado". Podría ser que una muestra más grande produjera un valor p menor.

La función básica de las pruebas significativas (también llamadas *pruebas de hipótesis [hypothesis tests]*) es proteger contra los engaños del azar. Por lo tanto, normalmente están estructurados para minimizar los errores de tipo 1.

Ciencia de datos y valores p

El trabajo que hacen los científicos de datos generalmente no está destinado a su publicación en revistas científicas, por lo que el debate sobre el valor del valor p es algo meramente académico. Para un científico de datos, un valor p es una métrica conveniente en situaciones en las que deseamos saber si el resultado de un modelo que parece interesante y útil está dentro del rango de variabilidad de probabilidad normal. Como herramienta de decisión en un experimento, un valor p no debe considerarse un control, sino simplemente otro punto de información que influye en la decisión. Por ejemplo, los valores p a veces se utilizan como entradas intermedias en algunos modelos estadísticos o de aprendizaje automático. Una característica puede incluirse o excluirse de un modelo en función de su valor p.

Ideas clave

- Las pruebas significativas se utilizan para determinar si un efecto observado se encuentra dentro del rango de variación debida al azar de un modelo de hipótesis nula.
- El valor p es la probabilidad de que se produzcan resultados tan extremos como los resultados observados, dado un modelo de hipótesis nula.
- El valor alfa es el umbral de "excepcionalidad" en un modelo de probabilidad de hipótesis nula.
- Las pruebas significativas han sido mucho más relevantes para la presentación de informes formales de investigación que para la ciencia de datos (pero recientemente han desaparecido, incluso para los primeros).

Lecturas complementarias

- Stephen Stigler, “Fisher and the 5 % Level,” *Chance* 21, n.º 4 (2008): 12. Este artículo es un breve comentario sobre el libro de Ronald Fisher de 1925 *Statistical Methods for Research Workers* (Oliver & Boyd), y sobre la importancia de Fisher en el nivel de significación del 5 %.
- Consultar también “Pruebas de hipótesis” en la página 93 y la lectura complementaria que se menciona allí.

Pruebas t

Existen numerosos tipos de pruebas significativas, dependiendo de si los datos son datos de recuento o datos medidos, de cuántas muestras hay y qué se está midiendo. Una muy extendida es la *prueba t* (*t-test*), que lleva el nombre de la distribución t de Student, desarrollada originalmente por W. S. Gosset para aproximar la distribución de una media muestral (consultar “Distribución t de Student” en la página 75).

Términos clave de pruebas t

Estadístico de prueba

Métrica de la diferencia o efecto que nos interesa.

Estadístico t

Una versión estandarizada de estadísticos de prueba corrientes, como pueden ser las medias.

Distribución t

Distribución de referencia (en este caso derivada de la hipótesis nula), con la que se puede comparar el estadístico t observado.

Las pruebas significativas requieren que especifiquemos un *estadístico de prueba (test statistic)* para medir el efecto que nos interesa y ayudarnos a determinar si ese efecto observado se encuentra dentro del rango de variación debida al azar. En una prueba de remuestreo (consultar la discusión sobre permutaciones en “Prueba de permutación” en la página 97), la escala de los datos no importa. Creamos la distribución de referencia (hipótesis nula) a partir de los datos en sí y usamos el estadístico de prueba tal cual.

En las décadas de 1920 y 1930, cuando se desarrollaba la prueba de hipótesis estadística, no era factible mezclar aleatoriamente los datos miles de veces para hacer una prueba de remuestreo. Los estadísticos encontraron que una buena aproximación a la distribución de permutación (mezclada) era la prueba t, basada en la distribución t de Gosset. Se utiliza para la comparación, muy frecuente por otra parte, de dos muestras (prueba A/B) en la que los datos son numéricos. Pero para que la distribución t se emplee sin tener en cuenta la escala, se debe usar una forma estandarizada del estadístico de prueba.

En esta etapa, un texto de estadística clásica mostraría varias fórmulas que incorporarían la distribución de Gosset y demostrarían cómo estandarizar nuestros datos para compararlos con la distribución t estándar. Estas fórmulas no se muestran aquí porque el software estadístico, así como R y Python, incluyen comandos que incorporan la fórmula. En R, la función es `t.test`:

```
> t.test(Time ~ Page, data=session_times, alternative='less')
Welch Two Sample t-test

data: Time by Page
t = -1.0983, df = 27.693, p-value = 0.1408
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
-Inf 19.59674
sample estimates:
mean in group Page A mean in group Page B
126.3333            162.0000
```

En Python, se puede utilizar la función `scipy.stats.ttest_ind`:

```
res = stats.ttest_ind(session_times[session_times.Page == 'Page A'].Time,
                      session_times[session_times.Page == 'Page B'].Time,
                      equal_var=False)
print(f'p-value for single sided test: {res.pvalue / 2:.4f}')
```

La hipótesis alternativa es que la media del tiempo de sesión para la página A es menor que para la página B. El valor p de 0.1408 está bastante cerca de los valores p de la prueba de permutación de 0.121 y 0.126 (consultar "Ejemplo: adherencia de la web" en la página 98).

En un modo de remuestreo, estructuramos la solución para reflejar los datos observados y la hipótesis que se va a probar, sin preocuparnos de que los datos sean numéricos o binarios, de que los tamaños de las muestras estén equilibrados, de las varianzas de las muestras o de una serie de otros factores. En el mundo de las fórmulas, se presentan muchas variaciones y pueden ser desconcertantes. Los estadísticos necesitan navegar por ese mundo y aprender su mapa, pero los científicos de datos, no. Por lo general, no están en el negocio de sudar los detalles de las pruebas de hipótesis y los intervalos de confianza como lo haría un investigador que prepara un artículo para una presentación.

Ideas clave

- Antes de la llegada de los ordenadores, las pruebas de remuestreo no eran factibles y los estadísticos usaban distribuciones de referencia estándar.
- A continuación, se podía estandarizar el estadístico de prueba y compararlo con la distribución de referencia.
- Uno de estos estadísticos estandarizados que se utiliza ampliamente es el estadístico t.

Lecturas complementarias

- Cualquier texto introductorio de estadística tendrá ilustraciones del estadístico t y de sus usos. Dos textos recomendables son *Statistics*, 4.^a ed., de David Freedman, Robert Pisani y Roger Purves (WW Norton, 2007), y *The Basic Practice of Statistics*, 8.^a ed., de David S. Moore, William I. Notz, y Michael A. Fligner (WH Freeman, 2017).
- Para un tratamiento tanto de la prueba t como de los procedimientos de remuestreo en paralelo, consultar *Introductory Statistics and Analytics: A Resampling Perspective* de Peter Bruce (Wiley, 2014) o *Statistics: Unlocking the Power of Data*, 2.^a ed., de Robin Lock y otros cuatro miembros de la familia Lock (Wiley, 2016).

Pruebas múltiples

Como mencionamos anteriormente, hay un dicho en estadística: "Tortura los datos el tiempo suficiente y confesarán". Esto significa que si observamos los datos a través de

bastantes perspectivas diferentes y hacemos las suficientes preguntas, casi invariablemente encontraremos un efecto estadísticamente significativo.

Por ejemplo, si tenemos 20 variables predictoras y una variable de resultado, todas generadas *aleatoriamente (randomly)*, las probabilidades son bastante buenas de que al menos una predictora resulte (falsamente) estadísticamente significativa si realizamos una serie de 20 pruebas significativas en el nivel alfa = 0.05. Como se mencionó con anterioridad, a esto se le denomina *error de tipo 1 (Type 1 error)*. Podemos calcular esta probabilidad encontrando primero la probabilidad de que todas den correctamente la prueba no significativa al nivel 0.05. La probabilidad de que *una (one)* prueba *correctamente (correctly)* no sea significativa es de 0.95, por lo que la probabilidad de que las 20 prueben correctamente que no son significativas es de $0.95 \times 0.95 \times 0.95\dots$, es decir, $0.95^{20} = 0.36$ ⁽²⁾. La probabilidad de que al menos una predictora resulte (falsamente) significativa es la otra cara de esta probabilidad, o $1 - (\text{probabilidad de que todas sean no significativas [probability that all will be nonsignificant]}) = 0.64$. A esto se le conoce como *inflación alfa (alpha inflation)*.

Este tema está relacionado con el problema del sobreajuste en la minería de datos o "ajuste del modelo al ruido". Cuantas más variables agreguemos, o más modelos ejecutemos, mayor será la probabilidad de que algo emerja como "significativo" por casualidad.

Términos clave de pruebas múltiples

Error de tipo 1

Concluir erróneamente que un efecto es estadísticamente significativo.

Tasa de falsos descubrimientos

Mediante múltiples pruebas, la tasa de cometer un error de tipo 1.

Inflación alfa

Fenómeno de pruebas múltiples, en el que *alfa (alpha)*, la probabilidad de cometer un error de tipo 1, aumenta a medida que se realizan más pruebas.

Ajuste de valores p

Contabilización de la realización de varias pruebas con los mismos datos.

Sobreajuste

Ajuste del ruido.

² La regla de la multiplicación establece que la probabilidad de que sucedan *n* eventos independientes es el producto de las probabilidades individuales. Por ejemplo, si tú y yo lanzamos una moneda una vez, la probabilidad de que tu moneda y la mía caigan cara es $0.5 \times 0.5 = 0.25$.

Experimentos estadísticos y pruebas significativas

En tareas de aprendizaje supervisado, un conjunto de prueba en el que los modelos se evalúan con datos que los modelos no han visto antes mitiga este riesgo. En las tareas de estadística y de aprendizaje automático que no involucran un conjunto de prueba etiquetado, persiste el riesgo de llegar a conclusiones basadas en el ruido estadístico.

En estadística, existen algunos procedimientos destinados a abordar este problema en circunstancias muy específicas. Por ejemplo, si estamos comparando resultados a través de varios grupos de tratamientos, podríamos formular algunas preguntas. Así, para los tratamientos A-C, podríamos preguntar:

- ¿Es A diferente de B?
- ¿Es B diferente de C?
- ¿Es A diferente de C?

O, en un ensayo clínico, es posible que deseemos ver los resultados de una terapia en varias etapas. En cada caso, hacemos varias preguntas, y con cada pregunta aumenta la posibilidad de que el azar nos engañe. Los procedimientos de ajuste en los estadísticos pueden compensar este riesgo estableciendo el listón de significación estadística de manera más estricta de lo que se establecería para una única prueba de hipótesis. Estos procedimientos de ajuste normalmente implican "dividir alfa" según el número de pruebas. Esto da como resultado un alfa más pequeño (es decir, un listón más estricto de significación estadística) para cada prueba. Uno de esos procedimientos, el ajuste de Bonferroni, simplemente divide alfa por el número de comparaciones. Otro, utilizado para comparar las medias de varios grupos, es la "diferencia significativa honesta" de Tukey, o la *HSD de Tukey (Tukey's HSD)*. Esta prueba se aplica a la diferencia máxima entre las medias de los grupos, comparándola con un punto de referencia basado en la *distribución t* (*t-distribution*) (aproximadamente equivalente a mezclar todos los valores, tratar grupos muestreados de forma repetida del mismo tamaño que los grupos originales y encontrar la diferencia máxima entre las medias del grupo remuestreado).

Sin embargo, el problema de las comparaciones múltiples va más allá de estos casos altamente estructurados y está relacionado con el fenómeno del "dragado" repetido de datos que da lugar al dicho sobre la tortura de los datos. Expresado de otra manera, dados unos datos bastante complejos, si no hemos encontrado algo interesante, simplemente es porque no hemos buscado lo suficiente. Ahora hay más datos disponibles que nunca, y el número de artículos científicos publicados casi se duplicó entre 2002 y 2010. Esto da lugar a muchas oportunidades para encontrar algo interesante en los datos, incluidos problemas de multiplicidad como:

- Comprobación de múltiples diferencias por pares entre grupos.
- Observación de los resultados de múltiples subgrupos ("no encontramos ningún efecto de tratamiento significativo en general, pero sí encontramos un efecto para las mujeres solteras menores de 30 años").
- Probar muchos modelos estadísticos.
- Incluir muchas variables en los modelos.
- Hacer una serie de preguntas diferentes (es decir, diferentes resultados posibles).



Tasa de descubrimientos falsos

El término tasa de descubrimientos falsos (*false discovery rate*) se usó originalmente para describir la tasa a la que un conjunto dado de pruebas de hipótesis identificaría falsamente un efecto significativo. Se convirtió en un concepto particularmente útil con la llegada de la investigación genómica, en la que se podían realizar un gran número de pruebas estadísticas como parte de un proyecto de secuenciación de genes. En estos casos, el término se aplica al protocolo de pruebas, y un "descubrimiento" falso se refiere al resultado de una prueba de hipótesis (por ejemplo, entre dos muestras). Los investigadores trataron de establecer los parámetros del proceso de prueba para controlar la tasa de falsos descubrimientos a un nivel específico. El término también se ha utilizado para la clasificación en minería de datos. Es la tasa de clasificación errónea en los pronósticos de clase 1. O, dicho de otra manera, es la probabilidad de que un "descubrimiento" (etiquetar un registro como "1") sea falso. Aquí normalmente nos enfrentamos al caso en el que los 0 son abundantes y los 1 son interesantes y raros (consultar el capítulo 5 y "El problema de las clases raras" en la página 223).

Por diversas razones, incluyendo especialmente este problema general de "multiplicidad", más investigación no significa necesariamente mejor investigación. Por ejemplo, la compañía farmacéutica Bayer descubrió en 2011 que cuando intentaba replicar 67 estudios científicos, solo pudo replicar completamente 14 de ellos. Casi dos tercios no pudieron reproducirse en absoluto.

En cualquier caso, los procedimientos de ajuste para pruebas estadísticas muy definidas y estructuradas son demasiado específicos e inflexibles para su uso general por parte de los científicos de datos. Las conclusiones para los científicos de datos sobre multiplicidad son:

- Para el modelado predictivo, el riesgo de obtener un modelo ilusorio cuya eficacia aparente es en gran parte producto del azar disminuye haciendo uso de la validación cruzada (consultar "Validación cruzada" en la página 155) y utilizando una muestra reservada.

- Para otros procedimientos sin un conjunto de prueba etiquetado para verificar el modelo, debemos confiar en:
 - Ser conscientes de que cuanto más consultamos y manipulamos los datos, mayor es el papel que puede desempeñar el azar.
 - Heurísticas de remuestreo y simulación para proporcionar puntos de referencia del azar con los que comparar los resultados observados.

Ideas clave

- La multiplicidad en un estudio de investigación o proyecto de minería de datos (comparaciones múltiples, muchas variables, muchos modelos, etc.) aumenta el riesgo de concluir que algo es significativo por casualidad.
- Para situaciones que involucran múltiples comparaciones estadísticas (es decir, múltiples pruebas significativas) existen procedimientos de ajuste estadístico.
- En una situación de minería de datos, el uso de una muestra de prueba con variables de resultado etiquetadas puede ayudar a evitar resultados engañosos.

Lecturas complementarias

- Para una breve exposición del procedimiento (la prueba de Dunnett) para ajustar las comparaciones múltiples, consultar el texto de estadística en línea de David Lane (<https://davidmlane.com/hyperstat/B112114.html>).
- Megan Goldman ofrece un tratamiento un poco más largo del procedimiento de ajuste de Bonferroni (<https://www.stat.berkeley.edu/~mgoldman/Section0402.pdf>).
- Para un tratamiento en profundidad de los procedimientos estadísticos más flexibles para ajustar los valores p, consultar *Resampling-Based Multiple Testing* de Peter Westfall y Stanley Young (Wiley, 1993).
- Para leer sobre un debate sobre la partición de datos y el uso de muestras de prueba en el modelado predictivo, consultar el capítulo 2 de *Data Mining for Business Analytics*, de Galit Shmueli, Peter Bruce, Nitin Patel, Peter Gedeck, Inbal Yahav y Kenneth Lichtendahl (Wiley, 2007-2020, con ediciones para R, Python, Excel y JMP).

Grados de libertad

En la documentación y la configuración de muchas pruebas estadísticas y distribuciones de probabilidad, veremos la referencia a los "grados de libertad". El concepto se aplica a estadísticos calculados a partir de los datos de una muestra y se refiere al número de valores que pueden variar libremente. Por ejemplo, si conocemos la media de una muestra de 10 valores, hay 9 grados de libertad (una vez que conozcamos 9 de los valores de la muestra, se

puede calcular el décimo y no es libre de variar). El parámetro de grados de libertad, aplicado a muchas distribuciones de probabilidad, afecta a la forma de la distribución.

El número de grados de libertad es una entrada para muchas pruebas estadísticas. Por ejemplo, grados de libertad es el nombre que se le da al denominador $n - 1$ que hemos visto en los cálculos de la varianza y la desviación estándar. ¿Por qué importa tenerlo en cuenta? Cuando usamos una muestra para estimar la varianza de una población, terminaremos con una estimación que está ligeramente sesgada a la baja si usamos n en el denominador. Si usamos $n - 1$ en el denominador, la estimación estará libre de ese sesgo.

Términos clave de los grados de libertad

n o tamaño de la muestra

Número de observaciones (también llamadas *filas [rows]* o *registros [records]*) de los datos.

d.f.

Grados de libertad.

Una gran parte de un curso o texto de estadística tradicional la ocupan diferentes pruebas estándar de hipótesis (prueba t, prueba F, etc.). Cuando los estadísticos de la muestra están estandarizados para usarlos en fórmulas estadísticas tradicionales, los grados de libertad son parte del cálculo de estandarización para garantizar que nuestros datos estandarizados coincidan con la adecuada distribución de referencia (distribución t, distribución F, etc.).

¿Es importante para la ciencia de datos? En realidad, no, al menos en el contexto de las pruebas significativas. Por un lado, las pruebas estadísticas formales se utilizan con moderación en la ciencia de datos. Por otro lado, el tamaño de los datos suele ser lo suficientemente grande como para que rara vez suponga una diferencia real para un científico de datos si, por ejemplo, el denominador es n o $n - 1$. (A medida que n se hace grande, el sesgo que se produciría al usar n en el denominador desaparece.)

Sin embargo, hay un contexto en el que es relevante: el uso de variables factorizadas en la regresión (incluida la regresión logística). Algunos algoritmos de regresión se bloquean si están presentes variables predictoras exactamente redundantes. Esto ocurre con mayor frecuencia al factorizar variables categóricas en indicadores binarios (variables ficticias). Consideremos la variable "día de la semana". Aunque hay siete días de la semana, solo hay seis grados de libertad para especificar el día de la semana. Por ejemplo, una vez que sepamos que el día de la semana no es de lunes a sábado, sabremos que debe ser domingo. La inclusión de los indicadores de lunes a sábado significa que también incluir el domingo provocaría que la regresión fallara debido a un error de *multicolinealidad (multicollinearity)*.

Ideas clave

- El número de grados de libertad (d.f. [degrees of freedom]) forma parte del cálculo para estandarizar los estadísticos de prueba para que puedan compararse con distribuciones de referencia (distribución t, distribución F, etc.).
- El concepto de grados de libertad se encuentra detrás de la factorización de variables categóricas en el indicador $n - 1$ o en variables ficticias al hacer una regresión (para evitar la multicolinealidad).

Lecturas complementarias

Hay varios tutoriales web sobre grados de libertad (<https://blog.minitab.com/en/statistics-and-quality-data-analysis/what-are-degrees-of-freedom-in-statistics>).

ANOVA

Supongamos que, en lugar de una prueba A/B, tuviéramos una comparación de varios grupos, digamos A/B/C/D, cada uno con datos numéricos. El procedimiento estadístico que prueba una diferencia estadísticamente significativa entre los grupos se llama *análisis de varianza* (*analysis of variance*) o ANOVA.

Términos clave de ANOVA

Comparación por pares

Prueba de hipótesis (por ejemplo, de medias) entre dos grupos de entre varios grupos.

Prueba ómnibus

Prueba de hipótesis única de la varianza general entre las medias de varios grupos.

Descomposición de la varianza

Separación de los componentes que contribuyen a un valor individual (por ejemplo, del promedio general, de la media del tratamiento y del error residual).

Estadístico F

Estadístico estandarizado que mide hasta qué punto las diferencias entre las medias de los grupos superan lo que podría esperarse en un modelo probabilístico.

SS

“Suma de cuadrados”, que se refiere a las desviaciones de algún valor promedio.

Estadística práctica para ciencia de datos con R y Python

La tabla 3.3 muestra la adherencia de cuatro páginas web, definida como la cantidad de segundos que un visitante ha estado en la página. Las cuatro páginas se cambian para que cada visitante de la web reciba una al azar. Hay un total de cinco visitantes para cada página y, en la tabla 3.3, cada columna es un conjunto de datos independiente. El primer espectador de la página 1 no tiene conexión con el primer espectador de la página 2. Tengamos en cuenta que en una prueba web como esta no podemos implementar completamente el diseño de muestreo aleatorio clásico en el que cada visitante se selecciona al azar de una gran población. Debemos tomar a los visitantes como vienen. Los visitantes pueden diferir sistemáticamente según la hora del día, la hora de la semana, la temporada del año, las condiciones de Internet, el dispositivo que están usando, etc. Estos factores deben considerarse como sesgos potenciales cuando se revisan los resultados del experimento.

Tabla 3.3 Adherencia (en segundos) de cuatro páginas web

	Página 1	Página 2	Página 3	Página 4
	164	178	175	155
	172	191	193	166
	177	182	171	164
	156	185	163	170
	195	177	176	168
Promedio	172	185	176	162
Promedio general				173.75

Ahora tenemos un acertijo (ver la figura 3.6). Cuando comparábamos solo dos grupos, era una cuestión sencilla, simplemente veímos la diferencia entre las medias de cada grupo. Con cuatro medias, hay seis posibles comparaciones entre grupos:

- Página 1 comparada con la página 2
- Página 1 comparada con la página 3
- Página 1 comparada con la página 4
- Página 2 comparada con la página 3
- Página 2 comparada con la página 4
- Página 3 comparada con la página 4

Cuanta más comparaciones por *pares* (*pairwise*) hagamos, mayor será la posibilidad de que nos engañe el azar (consultar “Pruebas múltiples” en la página 112). En lugar de preocuparnos por todas las diferentes comparaciones entre páginas individuales que podríamos hacer, podemos efectuar una única prueba general que aborde la pregunta: “¿Podrían todas las páginas tener la misma adherencia subyacente y que las diferencias entre ellas se debieran a la forma aleatoria en la que se asignó un conjunto de tiempos de sesión normales entre las cuatro páginas?”.

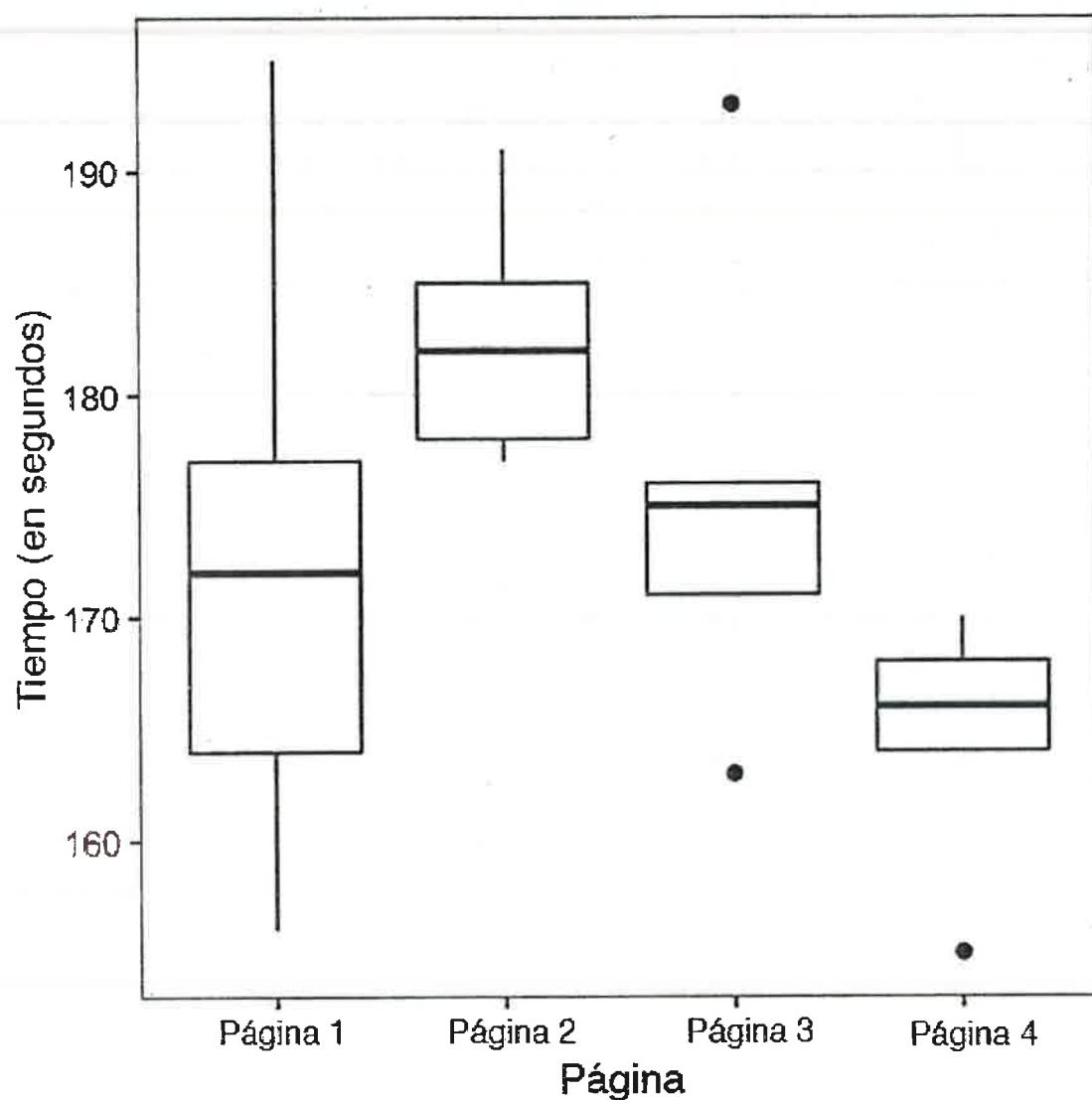


Figura 3.6 Los diagramas de caja de los cuatro grupos muestran diferencias considerables entre ellos.

El procedimiento utilizado para comprobar lo anterior es ANOVA. El fundamento para ello se puede ver en el siguiente procedimiento de remuestreo (especificado a continuación para la prueba A/B/C/D de adherencia de la página web):

1. Agrupamos todos los datos en una sola caja.
2. Mezclamos y extraemos cuatro remuestreos de cinco valores cada uno.
3. Registramos la media de cada uno de los cuatro grupos.
4. Registramos la varianza entre las medias de los cuatro grupos.
5. Repetimos los pasos del 2 al 4 muchas (digamos, 1000) veces.

Estadística práctica para ciencia de datos con R y Python

¿En qué proporción de tiempo la varianza muestreada de forma repetida excedió a la varianza observada? Este es el valor p.

Este tipo de prueba de permutación es un poco más complicado que el tipo utilizado en “Prueba de permutación” en la página 97. Afortunadamente, la función `aovp` del paquete `lmPerm` calcula la prueba de permutación para este caso:

```
> library(lmPerm)
> summary(aovp(Time ~ Page, data=four_sessions))
[1] "Settings: unique SS"
Component 1 :
      Df R Sum Sq R Mean Sq Iter Pr(Prob)
Page      3     831.4    277.13 3104  0.09278 .
Residuals 16    1618.4    101.15
---
Signif. codes: 0 '****' 0.001 '***' 0.01 '**' 0.05 '*' 0.1 '.' 1
```

El valor p, dado por `Pr(Prob)`, es 0.09278. En otras palabras, dada la misma adherencia subyacente, el 93 % de las veces la tasa de respuesta entre cuatro páginas puede diferir tanto como se observó realmente, solo por casualidad. Este grado de improbabilidad no alcanza el umbral estadístico tradicional del 5 %, por lo que concluimos que la diferencia entre las cuatro páginas podría haber surgido por casualidad.

La columna `Iter` enumera el número de iteraciones llevadas a cabo en la prueba de permutación. Las otras columnas corresponden a una tabla ANOVA tradicional y se describen más adelante.

En *Python*, podemos calcular la prueba de permutación usando el siguiente código:

```
observed_variance = four_sessions.groupby('Page').mean().var()[0]
print('Observed means:', four_sessions.groupby('Page').mean().values.ravel())
print('Variance:', observed_variance)

def perm_test(df):
    df = df.copy()
    df['Time'] = np.random.permutation(df['Time'].values)
    return df.groupby('Page').mean().var()[0]

perm_variance = [perm_test(four_sessions) for _ in range(3000)]
print('Pr(Prob)', np.mean([var > observed_variance for var in perm_variance]))
```

Estadístico F

Al igual que se puede utilizar la prueba t en lugar de la prueba de permutación para comparar la media de dos grupos, existe la prueba estadística para ANOVA basada en el *estadístico F* (*F-statistic*). El estadístico F se basa en la razón entre la varianza de las medias del grupo (es decir, el efecto del tratamiento) y la varianza debida al error

residual. Cuanto mayor sea esta relación, más significativo estadísticamente será el resultado. Si los datos siguen una distribución normal, entonces la teoría estadística dicta que el estadístico debe tener una determinada distribución. En base a esto, es posible calcular el valor p.

En R, podemos calcular una *tabla ANOVA* usando la función `aov`:

```
> summary(aov(Time ~ Page, data=four_sessions))
   Df Sum Sq Mean Sq F value Pr(>F)
Page        3  831.4   277.1   2.74 0.0776 .
Residuals   16 1618.4    101.2
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El paquete `statsmodels` proporciona una implementación de ANOVA en Python:

```
model = smf.ols('Time ~ Page', data=four_sessions).fit()

aov_table = sm.stats.anova_lm(model)
aov_table
```

El resultado del código Python es casi idéntico al de R.

Df es "grados de libertad", Sum Sq es "suma de cuadrados", Mean Sq es "cuadrados medios" (abreviatura de desviaciones cuadráticas medias) y F value es el estadístico F. Para el promedio general, la suma de cuadrados es la desviación del promedio general de 0, al cuadrado, multiplicado por 20 (el número de observaciones). El valor de los grados de libertad para el promedio general es 1, por definición.

Para las medias del tratamiento, los grados de libertad son 3 (una vez que se establecen tres valores, y luego se establece el promedio general, la otra media del tratamiento no puede variar). La suma de los cuadrados de las medias del tratamiento es la suma de las desviaciones al cuadrado entre las medias del tratamiento y el promedio general.

Para los residuos, los grados de libertad son 16 (20 observaciones, 16 de las cuales pueden variar después de que se establezcan la media general y las medias del tratamiento), y SS es la suma de la diferencia al cuadrado entre las observaciones individuales y las medias del tratamiento. Los cuadrados medios (MS) es la suma de los cuadrados dividida por los grados de libertad.

El estadístico F es MS (tratamiento)/MS (error). Por lo tanto, el valor de F depende solo de esta razón y se puede comparar con una distribución F estándar para determinar si las diferencias entre las medias del tratamiento son mayores de lo que cabría esperar en la variación debida al azar.



Descomposición de la varianza

Los valores observados en un conjunto de datos pueden considerarse sumas de diferentes componentes. Cualquier valor de datos observado en un conjunto de datos podemos desglosarlo en el promedio general, el efecto del tratamiento y el error residual. A esto lo llamamos "descomposición de la varianza":

1. Comenzamos con el promedio general (173.75 para los datos de adherencia de la página web).
2. Añadimos el efecto del tratamiento, que puede ser negativo (variable independiente = página web).
3. Añadimos el error residual, que puede ser negativo.

Por lo tanto, la descomposición de la varianza para el valor superior izquierdo en la tabla de prueba A/B/C/D es la siguiente:

1. Comenzamos con el promedio general: 173.75.
2. Añadimos el efecto de tratamiento (grupo): -1.75 ($172 - 173.75$).
3. Añadimos el residuo: -8 ($164 - 172$).
4. Igual a: 164.

ANOVA bidireccional

La prueba A/B/C/D que se acaba de describir es ANOVA "unidireccional", en la que tenemos un factor (grupo) que varía. Podríamos tener un segundo factor involucrado, digamos "fin de semana frente a día de la semana", con datos recolectados en cada combinación (grupo A fin de semana, grupo A día de la semana, grupo B fin de semana, etc.). Este sería un "ANOVA bidireccional" y lo manejaríamos de manera similar a ANOVA unidireccional identificando el "efecto de interacción". Después de identificar el efecto del promedio general y el efecto del tratamiento, separamos las observaciones del fin de semana y del día de la semana para cada grupo y encontramos la diferencia entre los promedios para esos subconjuntos y el promedio del tratamiento.

Podemos ver que ANOVA seguido por ANOVA bidireccional son los primeros pasos en el camino hacia un modelo estadístico completo, como la regresión y la regresión logística, en el que se pueden modelar múltiples factores y sus efectos (consultar el capítulo 4).

Ideas clave

- ANOVA es un procedimiento estadístico para analizar los resultados de un experimento con varios grupos.
- Es la extensión de procedimientos similares a la prueba A/B, que se utiliza para evaluar si la variación general entre los grupos está dentro del rango de variación debida al azar.
- Un resultado importante de ANOVA es la identificación de los componentes de la varianza asociados con los tratamientos de grupo, los efectos de interacción y los errores.

Lecturas complementarias

- *Introductory Statistics and Analytics: A Resampling Perspective* de Peter Bruce (Wiley, 2014) tiene un capítulo sobre ANOVA.
- *Introduction to Design and Analysis of Experiments* de George Cobb (Wiley, 2008) expone un tratamiento comprensible e interesante del tema.

Prueba de chi cuadrado

Las pruebas web a menudo van más allá de las pruebas A/B y ensayan múltiples tratamientos a la vez. La prueba de chi cuadrado se utiliza con datos de recuento para probar lo bien que se ajustan a alguna distribución esperada. El uso más habitual del estadístico *chi cuadrado* (*chi-square*) en la práctica estadística es con tablas de contingencia $r \times c$, para evaluar si la hipótesis nula de independencia entre variables es razonable (consultar también “Distribución chi cuadrado” en la página 80).

La prueba de chi cuadrado la desarrolló originalmente Karl Pearson en el año 1900 (<http://www.economics.soton.ac.uk/staff/aldrich/1900.pdf>). El término *chi* proviene de la letra griega X que utilizó Pearson en el artículo.

Términos clave de la prueba de chi cuadrado

Estadística chi cuadrado

Medida del grado en el que algunos datos observados se apartan de las expectativas.

Expectativas o resultados esperados

Cómo esperaríamos que resultaran los datos bajo algún supuesto, típicamente la hipótesis nula.



$r \times c$ es “filas (rows) y columnas (columns)”, una tabla 2×3 tiene dos filas y tres columnas.

Prueba de chi cuadrado: enfoque de remuestreo

Supongamos que estamos probando tres titulares diferentes (A, B y C) y los ejecutamos cada uno en 1000 visitantes, con los resultados mostrados en la tabla 3.4.

Tabla 3.4 Resultados de pruebas web para tres titulares diferentes

	Titular A	Titular B	Titular C
Clic	14	8	12
Sin clic	986	992	988

Los titulares ciertamente parecen diferir. El título A obtiene casi el doble de la tasa de clics que el B. Sin embargo, las cifras reales son pequeñas. Un procedimiento de remuestreo puede probar si las tasas de clics difieren en mayor medida de lo que podría ser fruto de la casualidad. Para esta prueba, necesitamos tener la distribución "esperada" de clics y, en este caso, eso sería bajo el supuesto de hipótesis nula de que los tres titulares comparten la misma tasa de clics, para una tasa de clics general de 34/3000. Bajo este supuesto, nuestra tabla de contingencias se parecería a la tabla 3.5.

Tabla 3.5 Valores esperados si los tres titulares tienen la misma tasa de clics (hipótesis nula)

	Titular A	Titular B	Titular C
Clic	11.33	11.33	11.33
Sin clic	988.67	988.67	988.67

El *residuo de Pearson* (*Pearson residual*) se define como:

$$R = \frac{\text{Observado} - \text{Esperado}}{\sqrt{\text{Esperado}}}$$

R mide el grado en que los recuentos reales difieren de los recuentos esperados (consultar la tabla 3.6).

Tabla 3.6 Residuos de Pearson

	Titular A	Titular B	Titular C
Clic	0.792	-0.990	0.198
Sin clic	-0.085	0.106	-0.021

El estadístico chi cuadrado se define como la suma de los residuos de Pearson al cuadrado:

$$\chi^2 = \sum_{i=1}^r \sum_{j=1}^c R_{ij}^2$$

donde r y c son el número de filas y columnas, respectivamente.

Experimentos estadísticos y pruebas significativas

El estadístico chi cuadrado para este ejemplo es 1.666. ¿Es eso más de lo que razonablemente podría ocurrir en un modelo probabilístico?

Podemos probar con este algoritmo de remuestreo:

1. Constituimos una caja con 34 1 (con clics) y 2966 0 (sin clics).
2. Las mezclamos, tomando tres muestras separadas de 1000 registros cada una y contamos los clics en cada registro.
3. Hallamos las diferencias al cuadrado entre los recuentos mezclados y los esperados y las sumamos.
4. Repetimos los pasos 2 y 3, digamos, 1000 veces.
5. ¿Con qué frecuencia la suma del remuestreo de las desviaciones al cuadrado supera a la observada? Ese es el valor p.

La función `chisq.test` se puede utilizar para calcular el estadístico chi cuadrado remuestreado en R. Para los datos de clics, la prueba de chi cuadrado es:

```
> chisq.test(clicks, simulate.p.value=TRUE)
Prueba de chi cuadrado de Pearson con el valor p simulado (basado en 2000 replicaciones)

data: clicks
X-squared = 1.6659, df = NA, p-value = 0.4853
```

La prueba muestra que este resultado podría haberse obtenido fácilmente por azar.

Para ejecutar una prueba de permutación en *Python*, utilizamos la siguiente implementación:

```
box = [1] * 34
box.extend([0] * 2966)
random.shuffle(box)

def chi2(observed, expected):
    pearson_residuals = []
    for row, expect in zip(observed, expected):
        pearson_residuals.append([(observe - expect) ** 2 / expect
                                   for observe in row])
    # return sum of squares
    return np.sum(pearson_residuals)

expected_clicks = 34 / 3
expected_noclicks = 1000 - expected_clicks
expected = [34 / 3, 1000 - 34 / 3]
chi2observed = chi2(clicks.values, expected)

def perm_fun(box):
    sample_clicks = [sum(random.sample(box, 1000)),
                    sum(random.sample(box, 1000)),
                    sum(random.sample(box, 1000))]
    sample_noclicks = [1000 - n for n in sample_clicks]
    return chi2([sample_clicks, sample_noclicks], expected)
```

Estadística práctica para ciencia de datos con R y Python

```
perm_chi2 = [perm_fun(box) for _ in range(2000)]  
  
resampled_p_value = sum(perm_chi2 > chi2observed) / len(perm_chi2)  
print(f'Observed chi2: {chi2observed:.4f}')  
print(f'Resampled p-value: {resampled_p_value:.4f}')
```

Prueba de chi cuadrado: teoría estadística

La teoría estadística asintótica muestra que la distribución del estadístico de chi cuadrado se puede aproximar mediante la *distribución chi cuadrado* (*chi-square distribution*) (consultar “Distribución chi cuadrado” en la página 80). La correspondiente distribución estándar chi cuadrado está determinada por los *grados de libertad* (*degrees of freedom*) (consultar “Grados de libertad” en la página 116). Para una tabla de contingencias, los grados de libertad están relacionados con el número de filas (r) y columnas (c) de la siguiente manera:

$$\text{grados de libertad} = (r - 1) \times (c - 1)$$

La distribución chi cuadrado está normalmente sesgada, con una cola larga a la derecha. Consultar la figura 3.7 para ver la distribución con 1, 2, 5 y 20 grados de libertad. Cuanto más alejado esté el estadístico observado en la distribución chi cuadrado, menor será el valor p .

La función `chisq.test` se puede utilizar para calcular el valor p utilizando la distribución chi cuadrado como referencia:

```
> chisq.test(clicks, simulate.p.value=FALSE)
```

Prueba de chi cuadrado de Pearson Chi-squared test

```
data: clicks  
X-squared = 1.6659, df = 2, p-value = 0.4348
```

En *Python*, utilizamos la función `scipy.stats.chi2_contingency`:

```
chisq, pvalue, df, expected = stats.chi2_contingency(clicks)  
print(f'Observed chi2: {chi2observed:.4f}')  
print(f'p-value: {pvalue:.4f}')
```

El valor p es un poco menor que el valor p de remuestreo. Esto se debe a que la distribución chi cuadrado es solo una aproximación de la distribución real del estadístico.

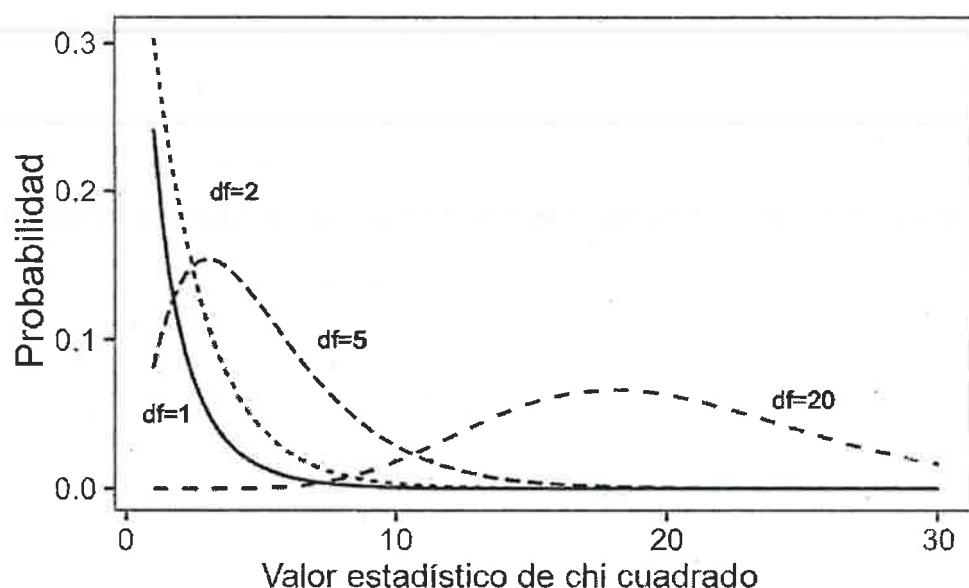


Figura 3.7 Distribución chi cuadrado con varios grados de libertad.

Prueba exacta de Fisher

La distribución chi cuadrado es una buena aproximación de la prueba de remuestreo aleatorio que se acaba de describir, excepto cuando los recuentos son extremadamente bajos (de un solo dígito, especialmente menor o igual a cinco). En tales casos, el procedimiento de remuestreo producirá valores p más precisos. De hecho, la mayoría del software estadístico tiene un procedimiento para enumerar *todas (all)* las posibles reordenaciones (permutaciones) que pueden ocurrir, tabular sus frecuencias y determinar exactamente lo extremo que es el resultado observado. A este procedimiento se le llama prueba exacta de Fisher (*Fisher's exact test*) en honor al gran estadístico R. A. Fisher. El código R para la prueba exacta de Fisher es simple en su forma básica:

```
> fisher.test(clicks)
Prueba exacta de Fisher para datos de recuento DataFisher.
data: clicks
p-value = 0.4824
alternative hypothesis: two.sided
```

El valor p está muy cerca del valor p de 0.4853 obtenido mediante el método de remuestreo.

Cuando algunos recuentos son muy bajos, pero otros son bastante altos (por ejemplo, el denominador en una tasa de conversión), puede ser necesario realizar una prueba de permutación aleatoria en lugar de una prueba exacta completa, debido a la dificultad de calcular todas las permutaciones posibles. La función R anterior tiene varios argumentos que controlan si se debe usar esta aproximación (`simulate.p.value=TRUE or FALSE`),

cuántas iteraciones se deben usar ($B = \dots$) y una restricción de cálculo (`workspace=...`) que limita hasta dónde deben llegar los cálculos para obtener el resultado *exacto* (`exact`).

No existe una implementación de la prueba exacta de Fisher disponible con facilidad en *Python*.

Detección del fraude científico

Un ejemplo interesante lo proporciona el caso de la investigadora de la Universidad de Tufts, Thereza Imanishi-Kari, acusada en 1991 de falsificar datos en su investigación. El congresista John Dingell se involucró en el caso, lo que finalmente llevó a que su colega, David Baltimore, renunciara a la presidencia de la Universidad Rockefeller.

Un elemento del caso descansaba en la evidencia estadística con respecto a la distribución esperada de dígitos en sus datos de laboratorio, en los que cada observación tenía muchos dígitos. Los investigadores se centraron en los dígitos *interiores* (*interior*) (ignorando el primer y último dígito de cada número), de los que se esperaría que siguieran una *distribución aleatoria uniforme* (*uniform random*). Es decir, que se producirían de forma aleatoria, por lo que cada dígito tendría la misma probabilidad de ocurrir (el dígito principal podría ser predominantemente un valor y los dígitos finales podrían verse afectados por el redondeo). La tabla 3.7 enumera las frecuencias de los dígitos interiores de los datos reales del caso.

Tabla 3.7 Frecuencia de dígitos interiores en datos de laboratorio

Dígito	Frecuencia
0	14
1	71
2	7
3	65
4	23
5	19
6	12
7	45
8	53
9	6

La distribución de los 315 dígitos, que se muestra en la figura 3.8, ciertamente no parece aleatoria.

Los investigadores calcularon la desviación de la expectativa (31.5 es la frecuencia con la que cada dígito ocurriría en una distribución estrictamente uniforme) y usaron una prueba de chi cuadrado (se podría haber utilizado igualmente un procedimiento de remuestreo) para mostrar que la distribución real estaba mucho más allá del rango de variación debida

al azar normal, lo que indica que los datos podrían haber sido inventados. (Nota: Imanishi-Kari fue finalmente exonerada después de un largo proceso.)

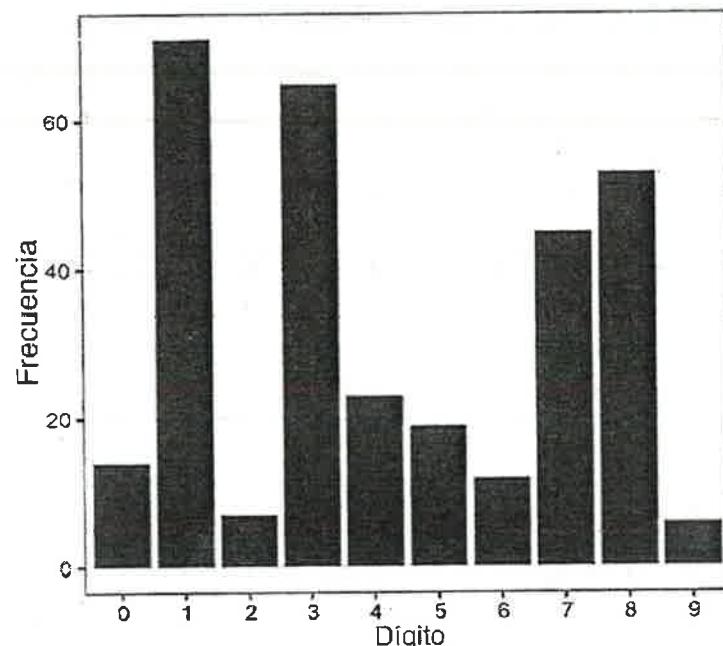


Figura 3.8 Histograma de frecuencias para los datos de laboratorio de Imanishi-Kari.

Relevancia para la ciencia de datos

La prueba de chi cuadrado o la prueba exacta de Fisher se utilizan cuando deseamos saber si un efecto es real o podría ser producto del azar. En la mayoría de las aplicaciones estadísticas clásicas de la prueba de chi cuadrado, su función es establecer la significación estadística, que normalmente se necesita antes de que se pueda publicar un estudio o un experimento. Esta circunstancia no es tan importante para los científicos de datos. En la mayoría de los experimentos de ciencia de datos, ya sea A/B o A/B/C..., el objetivo no es simplemente establecer la significación estadística, sino más bien llegar al mejor tratamiento. Para este propósito, Multi-Arm Bandit (consultar "Algoritmo Multi-Arm Bandit" en la página 131) ofrece una solución más completa.

Una aplicación de ciencia de datos de la prueba de chi cuadrado, especialmente la versión exacta de Fisher, consiste en determinar tamaños de muestra apropiados para experimentos web. Estos experimentos suelen tener tasas de clics muy bajas y, a pesar de miles de exposiciones, las tasas de recuentos pueden ser demasiado pequeñas para dar lugar a conclusiones definitivas en un experimento. En tales casos, la prueba exacta de Fisher, la prueba de chi cuadrado y otros tipos de pruebas pueden ser útiles como componente de los cálculos de la potencia y tamaño de la muestra (consultar "Potencia y tamaño de la muestra" en la página 135).

Estadística práctica para ciencia de datos con R y Python

Los investigadores utilizan generalmente las pruebas de chi cuadrado en la investigación en busca del elusivo valor p estadísticamente significativo que permitirá la publicación de la misma. Las pruebas de chi cuadrado, o simulaciones de remuestreo similares, se utilizan en aplicaciones de ciencia de datos más como un filtro para determinar si un efecto o una característica es digna de mayor consideración que como una prueba formal de significación. Por ejemplo, se utilizan en estadísticas espaciales y cartografía para determinar si los datos espaciales se ajustan a una distribución nula especificada (por ejemplo, ¿se concentran los delitos en un área determinada en mayor grado de lo que permitiría el azar?). También se pueden emplear en la selección automatizada de características en el aprendizaje automático, para evaluar la prevalencia de clases en todas las características e identificar características donde la prevalencia de una determinada clase es inusualmente alta o baja, de una manera que no es compatible con la variación aleatoria.

Ideas clave

- Un procedimiento habitual en estadística es probar si los recuentos de datos observados son consistentes con un supuesto de independencia (por ejemplo, si la propensión a comprar un artículo en particular es independiente del género).
- La distribución chi cuadrado es la distribución de referencia (que incorpora el supuesto de independencia) con la que se debe comparar el estadístico chi cuadrado calculado y observado.

Lecturas complementarias

- El famoso ejemplo de R. A. Fisher "Lady Tasting Tea" de principios del siglo XX sigue siendo una ilustración sencilla y eficaz de su prueba exacta. Si buscamos en Google "Lady Tasting Tea" encontraremos una serie de buenas reseñas.
- Stat Trek ofrece un buen tutorial sobre la prueba de chi cuadrado (<https://stattrek.com/chisquare-test/independence.aspx?Tutorial=AP>).

Algoritmo Multi-Arm Bandit

Multi-Arm Bandit ofrece un enfoque para las pruebas, especialmente en las pruebas web, que permite una optimización explícita y una toma de decisiones más rápida para el diseño de experimentos que el enfoque estadístico tradicional.

Términos clave de Multi-Arm Bandit

Multi-Arm Bandit

Máquina tragaperras imaginaria con múltiples brazos para que el cliente elija, cada uno con diferentes recompensas, utilizada aquí como analogía para un experimento de múltiples tratamientos.

Arm

Es el tratamiento en un experimento (por ejemplo, "Título A en una prueba web").

Acierto

Analogía experimental de un acierto en la máquina tragaperras (por ejemplo, "el cliente hace clic en el enlace").

La prueba A/B tradicional involucra datos recopilados en un experimento, de acuerdo con un diseño específico, para responder a una pregunta específica como: "¿Cuál es mejor, el tratamiento A o el tratamiento B?". La presunción es que una vez que obtenemos una respuesta a esa pregunta, la experimentación termina y procedemos a actuar sobre los resultados.

Probablemente podamos percibir algunas dificultades con ese enfoque. Primero, nuestra respuesta puede no ser concluyente: "efecto no probado". En otras palabras, los resultados del experimento pueden sugerir un efecto, pero si hay un efecto, no tenemos una muestra lo suficientemente grande para probarlo (a satisfacción de los estándares estadísticos tradicionales). ¿Qué decisión tomamos? En segundo lugar, es posible que deseemos comenzar a aprovechar los resultados que se obtienen antes de la conclusión del experimento. En tercer lugar, es posible que deseemos tener la posibilidad de cambiar de opinión o probar algo diferente en función de los datos adicionales que se reciben después de que termina el experimento. El enfoque tradicional de los experimentos y las pruebas de hipótesis data de la década de 1920 y es bastante inflexible. La aparición de la potencia de cálculo y el software de los ordenadores ha permitido enfoques flexibles más capaces. Además, la ciencia de datos y las empresas en general no están tan preocupadas por la significación estadística, sino más bien por optimizar el esfuerzo y los resultados generales.

Los algoritmos Bandit, que son muy populares en las pruebas web, nos permiten probar múltiples tratamientos a la vez y llegar a conclusiones más rápido que los diseños estadísticos tradicionales. Adopta su nombre de las máquinas tragaperras utilizadas en el juego, también llamadas bandidos de un solo brazo (ya que están hechas de tal manera que extraen dinero del jugador en un flujo constante). Si imaginamos una máquina tragamonedas con más de un brazo, cada brazo pagando a un ritmo diferente, tendríamos un bandido de múltiples brazos, que es el nombre completo de este algoritmo.

Estadística práctica para ciencia de datos con R y Python

Nuestro objetivo es ganar la mayor cantidad de dinero posible y, más concretamente, identificar y decidir sobre el brazo ganador más pronto que tarde. El desafío es que no sabemos a qué tasa general realizan el pago los brazos. Solo conocemos los resultados de las extracciones individuales de los brazos. Suponemos que cada "acerto" es por la misma cantidad, sin importar en qué brazo.

Lo que difiere es la probabilidad de que ocurra un acerto. Supongamos además que inicialmente probamos cada brazo 50 veces y obtenemos los siguientes resultados:

Brazo A: 10 aciertos de 50

Brazo B: 2 aciertos de 50

Brazo C: 4 aciertos de 50

Un planteamiento radical es decir: "Parece que el brazo A es el ganador. Dejemos de probar los otros brazos y sigamos con A". Este enfoque aprovecha al máximo la información de la prueba inicial. Si A es realmente superior, obtenemos el beneficio de ese hecho desde el principio. Por otro lado, si B o C son realmente mejores, perdemos cualquier oportunidad de descubrirlo. Otro planteamiento tajante es decir: "Todo esto parece estar dentro de las posibilidades del azar. Vamos a seguir tirando de todos por igual". Este último planteamiento brinda la máxima oportunidad para que se manifiesten los posibles suplentes de A. Sin embargo, en el proceso, estamos implementando lo que parecen ser tratamientos deficientes. ¿Cuánto tiempo podemos permitirlo? Los algoritmos Bandit adoptan un enfoque híbrido: comenzamos a extraer A más a menudo, para aprovechar su aparente superioridad, pero no abandonamos B y C. Simplemente los extraemos con menos frecuencia. Si A continúa obteniendo mejores resultados, continuamos desplazando recursos (extrayendo) de B y C y extrayendo de A con más frecuencia. Si, por otro lado, C comienza a mejorar y A empieza a empeorar, podemos cambiar las extracciones de A de nuevo a C. Si uno de ellos resulta ser superior a A y esto estaba oculto en la prueba inicial debido al azar, ahora tiene la oportunidad de aparecer al realizar más pruebas.

Ahora pensemos en aplicar esto a las pruebas web. En lugar de varios brazos de máquinas tragamonedas, es posible que tengamos varias ofertas, titulares, colores, etc., que se están probando en un sitio web. Los clientes hacen clic (una "ganancia" para el vendedor) o no hacen clic. Inicialmente, las ofertas se muestran de forma aleatoria y equitativa. Sin embargo, si una oferta comienza a superar a las demás, puede mostrarse ("extraerse") con más frecuencia. Pero ¿cuáles deberían ser los parámetros del algoritmo que modifica las tasas de extracción? ¿A qué "tasas de extracción" deberíamos cambiar y cuándo deberíamos cambiar?

El siguiente es un sencillo algoritmo, el algoritmo épsilon-greedy para una prueba A/B:

- I. Generamos un número aleatorio distribuido uniformemente entre 0 y 1.

2. Si el número se encuentra entre 0 y épsilon (donde épsilon es un número entre 0 y 1, generalmente bastante pequeño), lanzamos una moneda de curso legal (probabilidad de 50/50) y:
 - a. Si la moneda sale cara, mostramos la oferta A.
 - b. Si la moneda es cruz, mostramos la oferta B.
3. Si el número es $\geq \epsilon$, mostramos la oferta que haya tenido la tasa de respuesta más alta hasta el momento.

Épsilon es el único parámetro que gobierna este algoritmo. Si épsilon es 1, concluimos con un sencillo experimento A/B estándar (asignación aleatoria entre A y B para cada sujeto). Si épsilon es 0, concluimos con un algoritmo completamente *greedy* (voraz), uno que elige la mejor opción inmediata disponible (un óptimo local). No busca hacer más experimentos, simplemente asigna a los sujetos (visitantes de la web) al tratamiento de mejor rendimiento.

Un algoritmo más sofisticado utiliza el "muestreo de Thompson". Este procedimiento "toma muestras" (hace la extracción de un brazo del bandido) en cada etapa para maximizar la probabilidad de elegir el mejor brazo. Por supuesto, no sabe cuál es el mejor brazo, ¡ese es todo el problema! Pero a medida que observa la recompensa con cada sucesiva extracción, obtiene más información. El muestreo de Thompson emplea un enfoque bayesiano: inicialmente se asume alguna distribución previa de recompensas, usando lo que se llama una *distribución beta* (*beta distribution*) (es un mecanismo frecuente para especificar información previa en un problema bayesiano). A medida que se acumula la información de cada extracción, esta información se puede actualizar, lo que permite optimizar mejor la selección de la siguiente extracción en cuanto a elegir el brazo adecuado.

Los algoritmos Bandit pueden manejar de manera eficiente más de 3 tratamientos y avanzar hacia la selección óptima de los "mejores". Para los procedimientos estadísticos tradicionales de pruebas, la complejidad de la toma de decisiones para más de 3 tratamientos supera con creces la de la prueba A/B tradicional, y la ventaja de los algoritmos Bandit es mucho mayor.

Ideas clave

- Las pruebas A/B tradicionales prevén un proceso de muestreo aleatorio, que puede conducir a una exposición excesiva a un tratamiento deficiente.
- Multi-Arm Bandits, por el contrario, altera el proceso de muestreo para incorporar información aprendida durante el experimento y reducir la frecuencia del tratamiento deficiente.
- También facilitan el tratamiento eficaz de más de dos tratamientos.
- Existen diferentes algoritmos para desviar la probabilidad de muestreo del tratamiento o tratamientos deficientes al (presunto) tratamiento óptimo.

Lecturas complementarias

- Un excelente y breve tratamiento de los algoritmos Multi-Arm Bandit se encuentra en *Bandit Algorithms for Website Optimization*, de John Myles White (O'Reilly, 2012). White incluye código *Python*, así como los resultados de las simulaciones para evaluar el rendimiento de Bandit.
- Para más información (algo técnica) sobre el muestreo de Thompson, consultar “Analysis of Thompson Sampling for the Multi-armed Bandit Problem” (<http://proceedings.mlr.press/v23/agrawal12/agrawal12.pdf>) de Shipra Agrawal y Navin Goyal.

Potencia y tamaño de la muestra

Si ejecutamos una prueba web, ¿cómo decidimos cuánto tiempo debe ejecutarse (es decir, cuántas impresiones por tratamiento se necesitan)? A pesar de lo que podemos leer en muchas guías de pruebas web, no existe una buena guía general. Depende, principalmente, de la frecuencia con la que se logre el objetivo deseado.

Términos clave de potencia y tamaño de la muestra

Tamaño del efecto

Tamaño mínimo del efecto que esperamos poder detectar en una prueba estadística, como “una mejora del 20 % en las tasas de clics”.

Potencia

Probabilidad de detectar el tamaño de un efecto dado con un tamaño de muestra dado.

Nivel de significación

Nivel de significación estadística en el que se realizará la prueba.

Un paso en los cálculos estadísticos para el tamaño de la muestra es preguntar: “¿Una prueba de hipótesis realmente revelará una diferencia entre los tratamientos A y B?”. El resultado de la prueba de hipótesis, el valor *p*, depende de cuál sea la diferencia real entre el tratamiento A y el tratamiento B. También depende de la suerte de la extracción, quién es seleccionado para los grupos en el experimento. Pero tiene sentido que cuanto mayor sea la diferencia real entre los tratamientos A y B, mayor será la probabilidad de que nuestro experimento la revele, y cuanto menor sea la diferencia, se necesitarán más datos para detectarla. Para distinguir entre un bateador de .350 y un bateador de .200 en béisbol, no se necesitan tantos turnos al bate. Para distinguir entre un bateador de .300 y un bateador de .280, se necesitarán muchos más turnos al bate.

Potencia (power) es la probabilidad de detectar un *tamaño del efecto (effect size)* específico con características de la muestra específicas (tamaño y variabilidad). Por ejemplo, podríamos decir (hipotéticamente) que la probabilidad de distinguir entre un bateador de .330 y un bateador de .200 en 25 turnos al bate es de 0.75. El tamaño del efecto aquí es una diferencia de .130. Y "detectar" significa que una prueba de hipótesis rechazará la hipótesis nula de "ninguna diferencia" y concluirá que hay un efecto real. Entonces, el experimento de 25 turnos al bate ($n = 25$) para dos bateadores, con un tamaño de efecto de 0.130, tiene una potencia (hipotética) de 0.75, o 75 %.

Podemos ver que aquí hay varias partes móviles, y es fácil enredarse en las numerosas suposiciones y fórmulas estadísticas que serán necesarias (para especificar la variabilidad muestral, el tamaño del efecto, el tamaño de la muestra, el nivel alfa para la prueba de hipótesis, etc., y para calcular la potencia). De hecho, existe un software estadístico de propósito especial para calcular la potencia. La mayoría de los científicos de datos no necesitarán seguir todos los pasos formales necesarios para informar sobre la potencia, como ocurre, por ejemplo, en la publicación de un artículo. Sin embargo, pueden enfrentarse a ocasiones en las que deseen recopilar algunos datos para una prueba A/B, y recopilar o procesar los datos implica cierto coste. En ese caso, saber aproximadamente cuántos datos recopilar puede ayudar a evitar la situación en la que recopilamos datos con cierto esfuerzo y el resultado no es concluyente. Este es un enfoque alternativo bastante intuitivo:

1. Comenzamos con algunos datos hipotéticos que representen nuestra mejor suposición sobre los datos que se obtendrán (tal vez basándonos en datos anteriores), por ejemplo, una casilla con 20 unos y 80 ceros para representar un bateador de .200, o una casilla con algunas observaciones sobre el "tiempo dedicado al sitio web".
2. Creamos una segunda muestra simplemente agregando el tamaño del efecto deseado a la primera muestra, por ejemplo, un segundo cuadro con 33 unos y 67 ceros, o un segundo cuadro con 25 segundos agregados a cada "tiempo inicial dedicado a la página web".
3. Extraemos una muestra bootstrap de tamaño n de cada cuadro.
4. Realizamos una prueba de hipótesis de permutación (o basada en fórmulas) en las dos muestras bootstrap y registramos si la diferencia entre ellas es estadísticamente significativa.
5. Repetimos los dos pasos anteriores muchas veces y determinamos con qué frecuencia la diferencia ha sido significativa. Esa es la potencia estimada.

Tamaño de la muestra

El uso más habitual de los cálculos de potencia es la estimación del tamaño de la muestra que necesitaremos.

Por ejemplo, supongamos que estamos observando las tasas de clics (clics como porcentaje de las exposiciones) y probando un anuncio nuevo con un anuncio existente. ¿Cuántos clics necesitamos acumular en el estudio? Si solo nos interesan los resultados que muestran una gran diferencia (por ejemplo, una diferencia del 50 %), la solución podría ser una muestra relativamente pequeña. Si, por el contrario, fuera de interés sin embargo una pequeña diferencia, entonces se necesita una muestra mucho mayor. Un planteamiento normal es establecer la política de que un anuncio nuevo debe funcionar mejor que un anuncio existente en un porcentaje, digamos, del 10 %; de lo contrario, el anuncio existente permanecerá en su lugar. Este objetivo, el "tamaño del efecto", determina el tamaño de la muestra.

Por ejemplo, supongamos que las tasas de clics actuales son aproximadamente del 1.1 % y estamos buscando un aumento del 10 % al 1.21 %. Así que tenemos dos casillas: la casilla A con 1.1 % de unos (digamos, 110 unos y 9890 ceros), y la casilla B con 1.21 % de unos (digamos, 121 unos y 9879 ceros). Para empezar, intentemos 300 extracciones de cada casilla (esto sería como 300 "impresiones" para cada anuncio). Supongamos que en nuestra primera extracción se produce lo siguiente:

Caja A: 3 unos

Caja B: 5 unos

Inmediatamente podemos ver que cualquier prueba de hipótesis revelaría que esta diferencia (5 frente a 3) está dentro del rango de la variación debida al azar. Esta combinación de tamaño de la muestra ($n = 300$ en cada grupo) y tamaño del efecto (diferencia del 10 %) es demasiado pequeña para que cualquier prueba de hipótesis muestre una diferencia de manera confiable.

Por lo tanto, podemos intentar aumentar el tamaño de la muestra (intentemos 2000 impresiones) y exigir una mejora mayor (50 % en lugar de 10 %).

Por ejemplo, supongamos que las tasas de clics actuales siguen siendo del 1.1 %, pero ahora buscamos un aumento del 50 % hasta el 1.65 %. Así que tenemos dos casillas: la casilla A todavía con 1.1 % de unos (digamos, 110 unos y 9890 ceros), y la casilla B con 1.65 % de unos (digamos, 165 unos y 9868 ceros). Ahora intentaremos 2000 extracciones de cada caja. Supongamos que nuestra primera extracción tiene como resultado lo siguiente:

Caja A: 19 unos

Caja B: 34 unos

Una prueba significativa de esta diferencia (34-19) muestra que todavía se registra como "no significativa" (aunque está mucho más cercana a la significación que la diferencia anterior de 5-3). Para calcular la potencia, tendríamos que repetir el procedimiento anterior muchas veces o usar un software estadístico que pueda calcular la potencia, pero nuestra extracción inicial nos sugiere que incluso detectar una mejora del 50 % requerirá varios miles de impresiones de anuncios.

En resumen, para calcular la potencia o el tamaño de muestra requerido, hay cuatro partes móviles:

- tamaño de la muestra,
- tamaño del efecto que deseamos detectar,
- nivel de significación (alfa) en el que se realizará la prueba,
- potencia.

Especificamos tres de ellos y se podrá calcular el cuarto. Por lo general, desearemos calcular el tamaño de la muestra, por lo que debemos especificar los otros tres. Con *R* y *Python*, también debemos especificar la hipótesis alternativa como "mayor que" o "más grande que" para obtener una prueba unidireccional. Consultar "Pruebas de hipótesis unidireccionales o bidireccionales" en la página 95 para obtener más información sobre las pruebas unidireccionales o bidireccionales. Aquí está el código *R* para una prueba que involucra dos porcentajes, donde ambas muestras son del mismo tamaño (en este caso se usa el paquete *pwr*):

```
effect_size = ES.h(p1=0.0121, p2=0.011)
pwr.2p.test(h=effect_size, sig.level=0.05, power=0.8, alternative='greater')

Diferencia en el porcentaje de cálculo de la potencia para una distribución binomial
(transformación arco seno)

h = 0.01029785
n = 116601.7
sig.level = 0.05
power = 0.8
alternative = greater
```

NOTE: same sample sizes

La función *ES.h* calcula el tamaño del efecto. Vemos que si queremos una potencia del 80 %, necesitamos que la muestra tenga un tamaño de casi 120 000 impresiones. Si buscamos un aumento del 50 % (*p1* = 0.0165), el tamaño de la muestra se reduce a 5500 impresiones.

El paquete *statsmodels* contiene varios métodos para calcular la potencia. En el siguiente código, usamos *proportion_effectsize* para calcular el tamaño del efecto y *TTestIndPower* para resolver el tamaño de la muestra:

```
effect_size = sm.stats.proportion_effectsize(0.0121, 0.011)
analysis = sm.stats.TTestIndPower()
result = analysis.solve_power(effect_size=effect_size,
                               alpha=0.05, power=0.8, alternative='larger')
print('Sample Size: %.3f' % result)

Sample Size: 116602.393
```

Ideas clave

- Para saber qué tamaño de muestra necesitamos, es necesario pensar con anticipación en la prueba estadística que planeamos realizar.
- Debemos especificar el tamaño mínimo del efecto que deseamos detectar.
- También debemos especificar la probabilidad requerida de detectar ese tamaño del efecto (potencia).
- Por último, debemos especificar el nivel de significación (α) con el que se realizará la prueba.

Lecturas complementarias

- *Sample Size Determination and Power* de Thomas Ryan (Wiley, 2013) es una revisión completa y fácil de leer sobre este tema.
- Steve Simon, consultor estadístico, ha escrito una publicación de estilo narrativo muy interesante sobre el tema (<http://www.pmean.com/09/AppropriateSampleSize.html>).

Resumen

Los principios del diseño experimental (la aleatorización de sujetos en dos o más grupos que reciben diferentes tratamientos) nos permiten sacar conclusiones válidas sobre lo bien que funcionan los tratamientos. Es mejor incluir un tratamiento de control de "no hacer cambios". El tema de la inferencia estadística formal (pruebas de hipótesis, valores p , pruebas t y muchas otras pruebas de este tipo) ocupa mucho tiempo y espacio en un curso o texto de estadística tradicional, y el formalismo es en su mayoría innecesario desde el punto de vista de la ciencia de datos. Sin embargo, sigue siendo importante reconocer el papel que puede desempeñar la variación aleatoria para engañar al cerebro humano. Los procedimientos intuitivos de remuestreo (permutación y bootstrap) permiten a los científicos de datos medir hasta qué punto la variación debida al azar puede desempeñar un papel en el análisis de los datos.

CAPÍTULO 4

Regresión y pronóstico

Quizás el objetivo más frecuente en estadística es responder a la pregunta “¿La variable X (o más probablemente, X_1, \dots, X_p) está asociada con una variable Y , y si es así, ¿cuál es la relación? ¿Es posible utilizarla para pronosticar Y ?“.

En ninguna disciplina es más fuerte el nexo entre la estadística y la ciencia de datos que en el ámbito del pronóstico, específicamente, el pronóstico de una variable de resultado (objetivo) basado en los valores de otras variables "predictoras". Este proceso de entrenar un modelo con datos en el que se conoce el resultado, para su posterior aplicación a datos donde se desconoce el resultado, se denomina *aprendizaje supervisado (supervised learning)*. Otra conexión importante entre la ciencia de datos y la estadística se encuentra en el área de *detección de anomalías (anomaly detection)*, donde los diagnósticos de regresión originalmente destinados al análisis de datos y la mejora del modelo de regresión se pueden utilizar para detectar registros inusuales.

Regresión lineal simple

La regresión lineal simple proporciona un modelo de la relación entre la magnitud de una variable y la de una segunda variable. Por ejemplo, a medida que aumenta X , Y también aumenta. O a medida que X aumenta, Y disminuye³. La correlación es otra forma de medir cómo se relacionan dos variables. Consultar la sección "Correlación" en la página 30. La diferencia es que, si bien la correlación mide la fuerza (*strength*) de una asociación entre dos variables, la regresión cuantifica la naturaleza (*nature*) de la relación.

³ Esta sección y las siguientes de este capítulo © 2020 Datastats, LLC, Peter Bruce, Andrew Bruce y Peter Gedeck. Utilizado con el correspondiente permiso.

Términos clave de la regresión lineal simple

Respuesta

La variable que estamos tratando de pronosticar.

Sinónimos

variable dependiente, variable Y , objetivo, resultado

Variable independiente

Variable utilizada para pronosticar la respuesta.

Sinónimos

variable X , característica, atributo, predictora

Registro

Vector de valores predictivos y de resultado para un caso o individuo específico.

Sinónimos

fila, caso, instancia, ejemplo

Intersección

Intersección de la línea de regresión, es decir, el valor pronosticado cuando $X = 0$.

Sinónimos

b_0, β_0

Coeficiente de regresión

Pendiente de la recta de regresión.

Sinónimos

pendiente, b_1, β_1 , estimaciones de parámetros, ponderaciones

Valores ajustados

Estimaciones \hat{Y}_i obtenidas de la línea de regresión.

Sinónimo

valores pronosticados

Residuos

Diferencias entre los valores observados y los valores ajustados.

Sinónimo

errores

Mínimos cuadrados

Método para ajustar una regresión minimizando la suma de los residuos al cuadrado.

Sinónimos

mínimos cuadrados ordinarios, MCO

La ecuación de regresión

La regresión lineal simple estima cuánto cambiará Y cuando X cambie en una cierta cantidad. Con el coeficiente de correlación, las variables X e Y son intercambiables. Con la regresión, estamos tratando de pronosticar la variable Y a partir de X usando una relación lineal (es decir, una línea):

$$Y = b_0 + b_1 X$$

Leemos esta expresión como "Y es igual a b_1 por X, más una constante b_0 ". El símbolo b_0 se conoce como intersección (*intercept*) (o constante) y el símbolo b_1 como la *pendiente* (*slope*) de X. Ambos aparecen en el resultado de R como *coeficientes* (*coefficients*), aunque en sentido general el término *coeficiente* (*coefficient*) se reserva a menudo para b_1 . La variable Y se conoce como *respuesta* (*response*) o variable *dependiente* (*dependent*), ya que depende de X. La variable X se conoce como *predictora* (*predictor*) o variable *independiente* (*independent*). El colectivo de aprendizaje automático tiende a usar otros términos, llamando a Y el *objetivo* (*target*) y X el vector de *características* (*feature*). A lo largo de este libro, utilizaremos los términos *predictora* (*predictor*) y *características* (*feature*) indistintamente.

Consideremos el diagrama de dispersión de la figura 4.1 que muestra el número de años que un trabajador ha estado expuesto al polvo de algodón (Exposure) frente a una medida de la capacidad pulmonar (PEFR o ["peak expiratory flow rate"] "índice de flujo espiratorio máximo"). ¿Cómo se relaciona PEFR con Exposure? Es difícil de decir basándonos solo en la imagen.

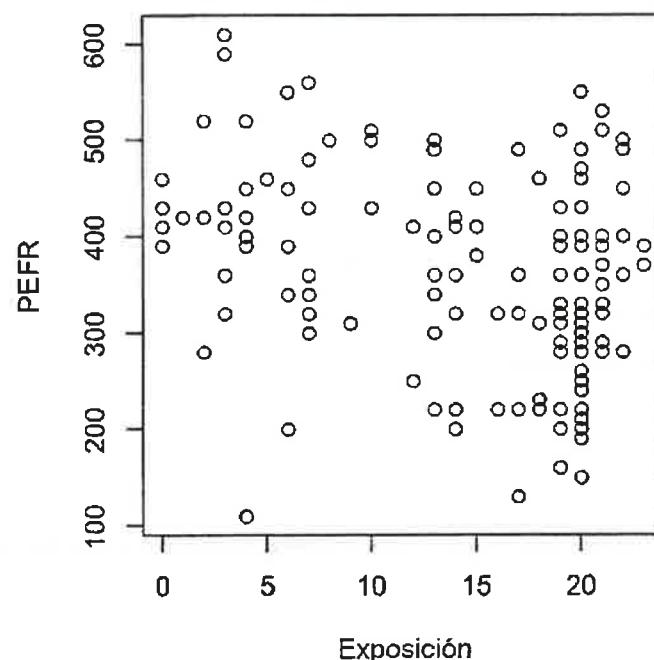


Figura 4.1 Exposición al algodón en relación con la capacidad pulmonar.

Estadística práctica para ciencia de datos con R y Python

La regresión lineal simple intenta encontrar la "mejor" recta para pronosticar la PEFR de respuesta en función de la variable predictora Exposure:

$$\text{PEFR} = b_0 + b_1 \text{Exposure}$$

La función `lm` en R se puede usar para ajustar la regresión lineal:

```
model <- lm(PEFR ~ Exposure, data=lung)
```

`lm` significa modelo *lineal* (*linear model*), y el símbolo `~` indica que a PEFR la pronostica Exposure. Con esta definición del modelo, la intersección automáticamente se incluye y se ajusta. Si deseamos excluir la intersección del modelo, debemos escribir la definición del modelo de la siguiente manera:

```
PEFR ~ Exposure - 1
```

Si imprimimos el objeto `model` obtenemos el siguiente resultado:

```
Call:  
lm(formula = PEFR ~ Exposure, data = lung)
```

```
Coefficients:  
(Intercept) Exposure  
424.583     -4.185
```

La intersección, o b_0 , es 424.583 y se puede interpretar como la PEFR pronosticada para un trabajador con cero años de exposición. El coeficiente de regresión, o b_1 , se puede interpretar de la siguiente manera: por cada año adicional que un trabajador está expuesto al polvo de algodón, la medición de la PEFR del trabajador se reduce en -4.185.

En Python, podemos usar `LinearRegression` del paquete `scikit-learn` (el paquete `statsmodels` tiene una implementación de regresión lineal más parecida a la de R [`sm.OLS`]. Lo utilizaremos más adelante en este capítulo):

```
predictors = ['Exposure']
outcome = 'PEFR'

model = LinearRegression()
model.fit(lung[predictors], lung[outcome])

print(f'Intercept: {model.intercept_.3f}')
print(f'Coefficient Exposure: {model.coef_[0].3f}'
```

La recta de regresión de este modelo se muestra en la figura 4.2.

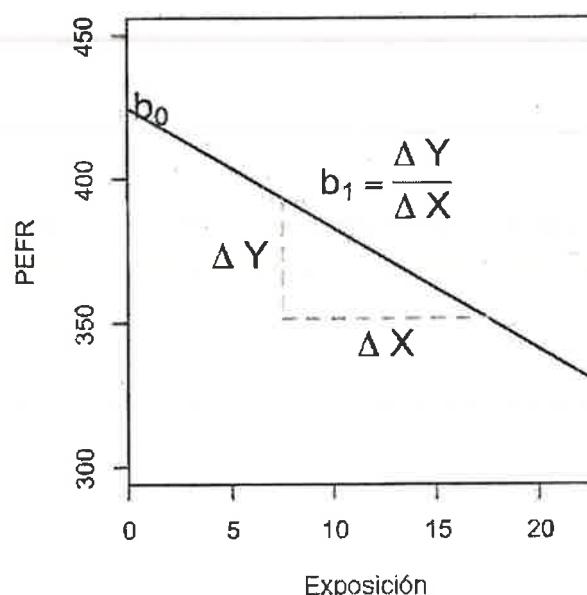


Figura 4.2 Pendiente e intersección para el ajuste de regresión de los datos pulmonares.

Valores ajustados y residuos

Los conceptos importantes en el análisis de regresión son los *valores ajustados* (*fitted values*) (los pronósticos) y los *residuos* (*residuals*) (errores de pronóstico). En general, los datos no se distribuyen formando exactamente una línea, por lo que la ecuación de regresión debe incluir explícitamente un término de error, e :

$$Y_i = b_0 + b_1 X_i + e_i$$

Los valores ajustados, también denominados *valores pronosticados* (*predicted values*), generalmente se denotan por \hat{Y}_i . Estos vienen dados por:

$$\hat{Y}_i = \hat{b}_0 + \hat{b}_1 X_i$$

La notación \hat{b}_0 y \hat{b}_1 indica que los coeficientes son los estimados frente a los conocidos.



Notación hat: valores estimados frente a valores conocidos

La notación “hat” se utiliza para diferenciar entre estimaciones y valores conocidos. Entonces, el símbolo \hat{b} (“b-hat”) es una estimación del parámetro desconocido b . ¿Por qué los estadísticos diferencian entre la estimación y el valor real? La estimación tiene incertidumbre, mientras que el valor real es fijo⁴.

⁴ En la estadística bayesiana, se supone que el valor real es una variable aleatoria con una distribución específica. En el contexto bayesiano, en lugar de estimaciones de parámetros desconocidos, existen distribuciones anteriores y posteriores.

Estadística práctica para ciencia de datos con R y Python

Calculamos los residuos \hat{e}_i restando los valores pronosticados de los datos originales:

$$\hat{e}_i = Y_i - \hat{Y}_i$$

En *R*, podemos obtener los valores ajustados y los residuos usando las funciones `predict` y `residuals`:

```
fitted <- predict(model)  
resid <- residuals(model)
```

Con el modelo `LinearRegression` de `scikit-learn`, utilizamos el método `predict` con los datos de entrenamiento para obtener los valores `fitted` y posteriormente los `residuals`. Como veremos, este es un patrón general que siguen todos los modelos en `scikit-learn`:

```
fitted = model.predict(lung[predictors])  
residuals = lung[outcome] - fitted
```

La figura 4.3 ilustra los residuos de la recta de regresión ajustada a los datos pulmonares. Los residuos son la longitud de las líneas punteadas verticales desde los datos hasta la línea.

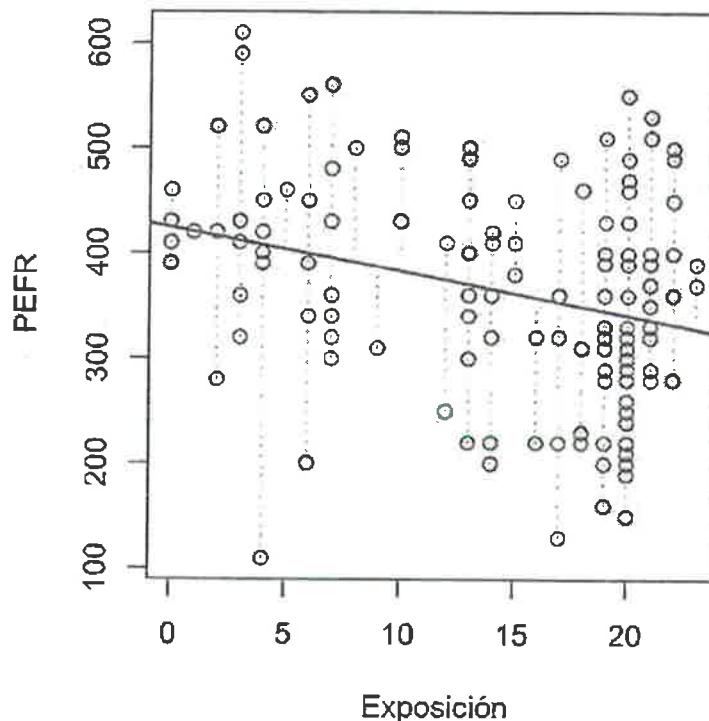


Figura 4.3 Residuos de la recta de regresión (para acomodar todos los datos, la escala del eje y difiere de la figura 4.2, de ahí la pendiente aparentemente diferente).

Mínimos cuadrados

¿Cómo se ajusta el modelo a los datos? Cuando existe una relación clara, podemos imaginar el ajuste de la recta de forma manual. En la práctica, la recta de regresión es la estimación que minimiza la suma de los residuos al cuadrado, también llamada *suma de los cuadrados de los residuos (residual sum of squares)* o *SCR (RSS)*:

$$\begin{aligned} RSS &= \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \\ &= \sum_{i=1}^n (Y_i - \hat{b}_0 - \hat{b}_1 X_i)^2 \end{aligned}$$

Las estimaciones \hat{b}_0 y \hat{b}_1 son los valores que minimizan RSS.

Al método para minimizar la suma de los cuadrados de los residuos se denomina regresión de *mínimos cuadrados (least squares)* o regresión de *mínimos cuadrados ordinarios (ordinary least squares)* (MCO). A menudo se atribuye a Carl Friedrich Gauss, el matemático alemán, pero fue publicado por primera vez por el matemático francés Adrien-Marie Legendre en 1805. La regresión por mínimos cuadrados se puede calcular rápida y fácilmente con cualquier software estadístico estándar.

Históricamente, la conveniencia de cálculo es una de las razones del uso generalizado de mínimos cuadrados en la regresión. Con la llegada del big data, la velocidad de cálculo sigue siendo un factor importante. Los mínimos cuadrados, como la media (consultar “Estimación de medianas robustas” en la página 10), son sensibles a los valores atípicos, aunque esto tiende a ser un problema importante solo en conjuntos de datos de tamaño pequeño o moderado.



Terminología de la regresión

Cuando los analistas e investigadores utilizan el término regresión (*regression*) por sí solo, normalmente se refieren a la regresión lineal. El planteamiento suele centrarse en el desarrollo de un modelo lineal para explicar la relación entre las variables predictoras y una variable numérica como resultado. En su sentido estadístico formal, la regresión también incluye modelos no lineales que producen una relación funcional entre predictoras y variables de resultado. En el colectivo de aprendizaje automático, el término también se usa ocasionalmente para referirse al uso de cualquier modelo predictivo que proporcione un resultado numérico pronosticado (a diferencia de los métodos de clasificación que pronostican un resultado binario o categórico).

Pronóstico frente a explicación (elaboración de perfiles)

Históricamente, uno de los principales usos de la regresión ha sido tratar de aclarar una supuesta relación lineal entre las variables predictoras y una variable de resultado. El objetivo ha sido comprender la relación y explicarla utilizando los datos a los que se ha ajustado la regresión. En este caso, el enfoque principal está en la pendiente estimada de la ecuación de regresión, \hat{b} . Los economistas, por ejemplo, quieren conocer la relación entre el gasto del consumidor y el crecimiento del PIB. Es posible que los funcionarios de salud pública quieran saber si una campaña de información pública es eficaz para promover prácticas sexuales seguras. En tales casos, la atención no se centra en pronosticar casos individuales, sino en comprender la relación general entre las variables.

Con la llegada del big data, la regresión se utiliza muy a menudo para dar forma a un modelo con el que pronosticar resultados individuales para nuevos datos (es decir, un modelo predictivo) y no para explicar los datos disponibles. En este caso, los principales elementos que nos interesan son los valores ajustados \hat{Y} . En marketing, la regresión se puede utilizar para pronosticar el cambio en los ingresos en respuesta al tamaño de una campaña publicitaria. Las universidades utilizan la regresión para pronosticar la nota media final de los estudiantes en función de sus puntuaciones en la prueba de aptitud escolar.

Se establece un modelo de regresión que se ajusta bien a los datos, de manera que los cambios en X conducen a cambios en Y . Sin embargo, por sí misma, la ecuación de regresión no prueba la dirección de la causalidad. Las conclusiones sobre la causalidad deben provenir de una comprensión más amplia de la relación. Por ejemplo, una ecuación de regresión puede mostrar una relación definida entre el número de clics en un anuncio web y el número de conversiones. Es nuestro conocimiento del proceso de marketing, no la ecuación de regresión, lo que nos lleva a la conclusión de que los clics en el anuncio generan ventas, y no al revés.

Ideas clave

- La ecuación de regresión modela la relación entre una variable de respuesta Y y una variable predictora X como una línea recta.
- Un modelo de regresión produce valores ajustados y residuos (los pronósticos de la respuesta y los errores de los pronósticos).
- Los modelos de regresión suelen ajustarse mediante el método de mínimos cuadrados.
- La regresión se utiliza tanto para formular un pronóstico como para la ofrecer una explicación.

Lecturas complementarias

Para un tratamiento en profundidad del pronóstico frente a la explicación, consultar el artículo de Galit Shmueli “To Explain or to Predict?” (<https://projecteuclid.org/journals/statistical-science/volume-25/issue-3/To-Explain-or-to-Predict/10.1214/10-STS330.full>).

Regresión lineal múltiple

Cuando hay varias variables predictoras, la ecuación simplemente se amplía incluyéndolas:

$$Y = b_0 + b_1X_1 + b_2X_2 + \dots + b_pX_p + e$$

En lugar de una línea, ahora tenemos un modelo lineal: la relación entre cada coeficiente y su variable (característica) es lineal.

Términos clave de la regresión lineal múltiple

Error cuadrático medio

Es la raíz cuadrada del error cuadrático medio de la regresión (esta es la métrica más utilizada para comparar modelos de regresión).

Sinónimo

ECM (RMSE en inglés)

Error estándar residual

Lo mismo que la raíz del error cuadrático medio, pero ajustado por grados de libertad.

Sinónimo

ERE (RSE en inglés)

R cuadrado

Proporción de la varianza explicada por el modelo, de 0 a 1.

Sinónimos

coeficiente de determinación, R^2

Estadístico t

Coeficiente de una predictora, dividido por el error estándar del coeficiente. Proporciona una métrica para comparar la importancia de las variables en el modelo. Consultar “Pruebas t” en la página 110.

Regresión ponderada

Regresión con los registros que tienen diferentes ponderaciones.

Estadística práctica para ciencia de datos con R y Python

Todos los demás conceptos de la regresión lineal simple, como el ajuste por mínimos cuadrados y la definición de valores ajustados y residuos, se extienden a la configuración de regresión lineal múltiple. Por ejemplo, los valores ajustados vienen dados por:

$$\hat{Y}_i = \hat{b}_0 + \hat{b}_1 X_{1,i} + \hat{b}_2 X_{2,i} + \dots + \hat{b}_p X_{p,i}$$

Ejemplo: datos de las viviendas del condado de King

Un ejemplo de utilización de la regresión lineal múltiple es la estimación del valor de las viviendas. Los tasadores deben estimar el valor de una vivienda a efectos de evaluar los impuestos. Los profesionales inmobiliarios y los compradores de viviendas consultan sitios web populares, como Zillow (<https://zillow.com>), para determinar un precio justo. A continuación se muestran algunas filas de datos de las viviendas del condado de King (Seattle), Washington, de `house` `data.frame`:

```
head(house[, c('AdjSalePrice', 'SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms',
             'BldgGrade')])  
Source: local data frame [6 x 6]
```

	AdjSalePrice (dbl)	SqFtTotLiving (int)	SqFtLot (int)	Bathrooms (dbl)	Bedrooms (int)	BldgGrade (int)
1	300805	2400	9373	3.00	6	7
2	1076162	3764	20156	3.75	4	10
3	761805	2060	26036	1.75	4	8
4	442065	3200	8618	3.75	5	7
5	297065	1720	8620	1.75	4	7
6	411781	930	1012	1.50	2	8

El método `head` del marco de datos de pandas lista las primeras filas:

```
subset = ['AdjSalePrice', 'SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade']  
house[subset].head()
```

El objetivo es pronosticar el precio de venta a partir de las otras variables. La función `lm` maneja el caso de regresión múltiple simplemente incluyendo más términos en el lado derecho de la ecuación. El argumento `na.action=na.omit` hace que el modelo descarte los registros que no tienen valores registrados:

```
house_lm <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms + Bedrooms +  
BldgGrade,  
data=house, na.action=na.omit)
```

`LinearRegression` de `scikit-learn` también se puede utilizar para regresiones lineales múltiples:

```
predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade']  
outcome = 'AdjSalePrice'  
  
house_lm = LinearRegression()  
house_lm.fit(house[predictors], house[outcome])
```

Si imprimimos el objeto `house_lm` obtenemos el siguiente resultado:

```
house_lm
Call:
lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
    Bedrooms + BldgGrade, data = house, na.action = na.omit)

Coefficients:
            (Intercept)   SqFtTotLiving      SqFtLot      Bathrooms
              -5.219e+05       2.288e+02      -6.047e-02     -1.944e+04
             Bedrooms        BldgGrade
              -4.777e+04       1.061e+05
```

Para el modelo `LinearRegression`, la intersección y los coeficientes del modelo ajustado son los campos `intercept_` y `coef_`:

```
print(f'Intercept: {house_lm.intercept_.3f}')
print('Coefficients:')
for name, coef in zip(predictors, house_lm.coef_):
    print(f'{name}: {coef}')
```

La interpretación de los coeficientes es la misma que en la regresión lineal simple: el valor de pronóstico \hat{Y} cambia según el coeficiente b_j para cada cambio unitario en X_j , asumiendo que todas las demás variables, X_k para $k \neq j$, permanecen iguales. Por ejemplo, agregar un pie cuadrado terminado adicional a una vivienda aumenta el valor estimado en aproximadamente 229 \$. Agregar 1000 pies cuadrados terminados implica que el valor aumentará en 228 800 \$.

Evaluación del modelo

La métrica de rendimiento más importante desde la perspectiva de la ciencia de datos es la *raíz del error cuadrático medio* (*root meansquared error*) o RECM (RMSE en inglés). RECM es la raíz cuadrada del error cuadrático medio de los valores de pronóstico \hat{y}_i :

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{n}}$$

Este valor mide la precisión general del modelo y es una base para compararlo con otros modelos (incluidos los modelos que se ajustan mediante técnicas de aprendizaje automático). Similar a RECM es el *error estándar residual* (*residual standard error*), o ERS (RSE). En este caso tenemos p predictoras, y ERS viene dado por:

$$RSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n - p - 1)}}$$

La única diferencia es que en el denominador figuran los grados de libertad, en lugar del número de registros (consultar “Grados de libertad” en la página 116). En la práctica,

Estadística práctica para ciencia de datos con R y Python

para la regresión lineal, la diferencia entre RECM y ERS es muy pequeña, particularmente para aplicaciones de big data.

Para el modelo de regresión, la función `summary` en R calcula ERS y otras métricas:

```
summary(house_lm)
```

Call:

```
lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +  
    Bedrooms + BldgGrade, data = house, na.action = na.omit)
```

Residuals:

Min	1Q	Median	3Q	Max
-1199479	-118908	-20977	87435	9473035

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-5.219e+05	1.565e+04	-33.342	< 2e-16 ***
SqFtTotLiving	2.288e+02	3.899e+00	58.694	< 2e-16 ***
SqFtLot	-6.047e-02	6.118e-02	-0.988	0.323
Bathrooms	-1.944e+04	3.625e+03	-5.363	8.27e-08 ***
Bedrooms	-4.777e+04	2.490e+03	-19.187	< 2e-16 ***
BldgGrade	1.061e+05	2.396e+03	44.277	< 2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '' 1

Residual standard error: 261300 on 22681 degrees of freedom

Multiple R-squared: 0.5406, Adjusted R-squared: 0.5405

F-statistic: 5338 on 5 and 22681 DF, p-value: < 2.2e-16

scikit-learn proporciona una serie de métricas de regresión y clasificación. Aquí, utilizamos `mean_squared_error` para obtener RECM (RMSE en inglés) y `r2_score` para el coeficiente de determinación:

```
fitted = house_lm.predict(house[predictors])  
RMSE = np.sqrt(mean_squared_error(house[outcome], fitted))  
r2 = r2_score(house[outcome], fitted)  
print(f'RMSE: {RMSE:.0f}')  
print(f'r2: {r2:.4f}'")
```

Para obtener en *Python* un análisis más detallado del modelo de regresión usamos `statsmodels`:

```
model = sm.OLS(house[outcome], house[predictors].assign(const=1))  
results = model.fit()  
results.summary()
```

El método `assign` de pandas, como se utiliza aquí, añade una columna con el valor constante 1 a las predictoras. Esta adición es necesaria para modelar la intersección.

Otra métrica útil que veremos en el resultado del software es el *coeficiente de determinación* (*coefficient of determination*) también llamado estadístico *R cuadrado* (*R-squared*) o *R²*. *R cuadrado* varía de 0 a 1 y mide la proporción de la variación en los datos contabilizados en el

modelo. Es útil principalmente en aspectos aclaratorios de la regresión en los que deseamos evaluar lo bien que se ajusta el modelo a los datos. La fórmula para R^2 es:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

El denominador es proporcional a la varianza de Y . El resultado de R también informa de un R cuadrado ajustado (*adjusted R-squared*), que se ajusta a los grados de libertad, penalizando efectivamente la adición de más predictoras a un modelo. Rara vez este valor es significativamente diferente del de R cuadrado (*R-squared*) en regresión múltiple con grandes conjuntos de datos.

Junto con los coeficientes estimados, R y statsmodels informan del error estándar de los coeficientes SE (Standard Error) y del estadístico t (*t-statistic*):

$$t_b = \frac{b}{SE(b)}$$

El estadístico t (y su imagen espectral, el valor p) mide el grado en el que un coeficiente es "estadísticamente significativo", es decir, está fuera del rango del que podría producir una disposición aleatoria de la variable predictora y la variable objetivo. Cuanto mayor sea el estadístico t (y menor el valor p), más significativa será la predictora. Dado que la parsimonia es una característica valiosa del modelo, es útil tener una herramienta como esta para orientar la elección de variables a incluir como predictoras (consultar "Selección del modelo y regresión escalonada" en la página 156).



Además del estadístico t , R y otros paquetes reportarán a menudo el *valor p* (*p-value*) ($\Pr(>|t|)$) en el resultado de R y el *estadístico F* (*F-statistic*). Los científicos de datos generalmente no se involucran demasiado en la interpretación de estos estadísticos, ni tampoco con el tema de la significación estadística. Los científicos de datos se centran principalmente en el estadístico t como una guía útil para incluir o no una predictora en un modelo. Valores altos de los estadísticos t (que acompañan a valores p próximos a 0) indican que una predictora debe retenerse en un modelo, mientras que valores muy bajos de los estadísticos t indican que la predictora podría descartarse. Consultar "Valor p" en la página 106 para obtener más información.

Validación cruzada

Las métricas clásicas de regresión estadística (R^2 , estadístico F y valores p) son todas métricas "dentro de la muestra": se aplican a los mismos datos que se utilizaron para ajustar el modelo. Intuitivamente, podemos ver que tendría mucho sentido dejar de lado algunos de los datos

originales, no usarlos para ajustar el modelo, y luego aplicar el modelo a los datos reservados (retenidos) para ver lo bien que funciona. Normalmente, usaríamos la mayoría de los datos para ajustar el modelo y utilizariamos una porción más pequeña para probarlo.

La idea de validación “fuera de la muestra” no es nueva, pero en realidad no se afianzó hasta que fueron más frecuentes los conjuntos de datos más grandes. Con un conjunto de datos pequeño, los analistas normalmente quieren utilizar todos los datos y ajustar lo mejor posible el modelo.

Sin embargo, el uso de una muestra reservada nos somete a una cierta incertidumbre que surge sencillamente de la variabilidad de la pequeña muestra reservada. ¿Sería muy distinta la evaluación si seleccionáramos una muestra reservada diferente?

La validación cruzada extiende la idea de una muestra reservada a múltiples muestras reservadas secuenciales. El algoritmo para la *validación cruzada básica de k-fold (o k iteraciones)* (*k-fold cross-validation*) es el siguiente:

1. Apartamos $1 / k$ de los datos como muestra reservada.
2. Entrenamos el modelo con los datos restantes.
3. Aplicamos (puntuamos) el modelo a la reserva de $1 / k$ y registramos las métricas necesarias de evaluación del modelo.
4. Restituimos el primer $1 / k$ de los datos y reservamos el siguiente $1 / k$ (excluyendo los registros que se seleccionaron la primera vez).
5. Repetimos los pasos 2 y 3.
6. Repetimos hasta que hayamos utilizado cada registro en la parte reservada.
7. Promediamos o combinamos las métricas de evaluación del modelo.

La división de los datos en dos, la muestra de entrenamiento y la muestra reservada también se denomina *pliegue (fold)*.

Selección del modelo y regresión escalonada

En algunos problemas se podrían utilizar muchas variables como predictoras en una regresión. Por ejemplo, para pronosticar el valor de las viviendas, se podrían emplear variables adicionales como el tamaño del sótano o el año de construcción. En R, estas variables son fáciles de agregar a la ecuación de regresión:

```
house_full <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
  Bedrooms + BldgGrade + PropertyType + NbrLivingUnits +
  SqFtFinBasement + YrBuilt + YrRenovated +
  NewConstruction,
  data=house, na.action=na.omit)
```

En *Python* necesitamos convertir las variables categóricas y booleanas en números:

```

predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade',
              'PropertyType', 'NbrLivingUnits', 'SqFtFinBasement', 'YrBuilt',
              'YrRenovated', 'NewConstruction']

X = pd.get_dummies(house[predictors], drop_first=True)
X['NewConstruction'] = [1 if nc else 0 for nc in X['NewConstruction']]

house_full = sm.OLS(house[outcome], X.assign(const=1))
results = house_full.fit()
results.summary()

```

Sin embargo, añadir más variables no significa necesariamente que tengamos un modelo mejor. Los estadísticos utilizan el principio de la navaja de Occam (*Occam's razor*) para orientarse en la elección de un modelo: en igualdad de condiciones, se debe utilizar el modelo más simple en lugar del modelo más complicado.

La inclusión de variables adicionales siempre reduce el RECM (RMSE en inglés) y aumenta R^2 para los datos de entrenamiento. Por lo tanto, estas no son apropiadas para ayudar a orientarse en la elección del modelo. Un enfoque para incluir la complejidad del modelo es utilizar R^2 ajustado:

$$R^2_{adj} = 1 - (1 - R^2) \frac{n - 1}{n - p - 1}$$

donde n es el número de registros y P es el número de variables en el modelo.

En la década de 1970, Hirotugu Akaike, el eminente estadístico japonés, desarrolló una métrica llamada AIC (Criterios de información de Akaike [Akaike's Information Criteria]) que penaliza la adición de términos a un modelo. En el caso de regresión, AIC tiene la expresión:

$$AIC = 2P + n \log(RSS/n)$$

donde P es el número de variables y n es el número de registros. El objetivo es encontrar el modelo que minimice AIC. Los modelos con k más variables extra se penalizan con $2k$.



AIC, BIC, y Mallows Cp

La fórmula de AIC puede parecer un poco misteriosa, pero de hecho se basa en resultados asintóticos de la teoría de la información. Hay varias variantes de AIC.

AICc

Es una versión de AIC corregida para tamaños de muestra pequeños.

BIC o criterio de información bayesiano

Similar a AIC, con una penalización más fuerte por incluir variables adicionales al modelo.

Mallows Cp

Una variante de AIC desarrollada por Colin Mallows.

Por lo general, estas se identifican como métricas dentro de la muestra (es decir, sobre los datos de entrenamiento), y los científicos de datos, que usan los datos retenidos para la evaluación del modelo, no necesitan preocuparse por las diferencias entre ellas o la teoría que subyace tras las mismas.

¿Cómo encontramos el modelo que minimiza AIC o maximiza R^2 ajustado? Una forma es buscar en todos los modelos posibles un enfoque llamado *regresión de todos los subconjuntos* (*all subset regression*). Esto es costoso desde el punto de vista del cálculo y no es factible para problemas con grandes cantidades de datos y muchas variables. Una alternativa atractiva es utilizar la *regresión por pasos* (*stepwise regression*). Se podría comenzar con un modelo completo y eliminar sucesivamente las variables que no contribuyan de manera significativa. A esto se le llama eliminación hacia atrás (*backward elimination*). Alternativamente, se podría comenzar con un modelo constante y agregar variables sucesivamente (selección hacia delante [*forward selection*]). Como tercera opción, también podemos agregar y eliminar predictoras sucesivamente para encontrar un modelo que reduzca el AIC o R^2 ajustado. El paquete MASS en R de Venables y Ripley ofrece una función de regresión por pasos llamada *stepAIC*:

```
library(MASS)
step <- stepAIC(house_full, direction="both")
step

Call:
lm(formula = AdjSalePrice ~ SqFtTotLiving + Bathrooms + Bedrooms +
    BldgGrade + PropertyType + SqFtFinBasement + YrBuilt, data = house,
    na.action = na.omit)
```

Coefficients:

	(Intercept)	SqFtTotLiving
	6.179e+06	1.993e+02
Bathrooms		Bedrooms
	4.240e+04	-5.195e+04
BldgGrade		PropertyTypeSingle Family
	1.372e+05	2.291e+04
PropertyTypeTownhouse		SqFtFinBasement
	8.448e+04	7.047e+00
YrBuilt		
	-3.565e+03	

scikit-learn no tiene implementación para la regresión paso a paso. Implementamos las funciones *stepwise_selection*, *forward_selection*, y *backward_elimination* en nuestro paquete dmba:

```
y = house[outcome]

def train_model(variables): ❶
    if len(variables) == 0:
```

```

    return None
model = LinearRegression()
model.fit(X[variables], y)
return model

def score_model(model, variables): ②
    if len(variables) == 0:
        return AIC_score(y, [y.mean()] * len(y), model, df=1)
    return AIC_score(y, model.predict(X[variables]), model)

best_model, best_variables = stepwise_selection(X.columns, train_model,
                                                score_model, verbose=True)

print(f'Intercept: {best_model.intercept_:.3f}')
print('Coefficients:')
for name, coef in zip(best_variables, best_model.coef_):
    print(f' {name}: {coef}')

```

- ① Define una función que proporciona como resultado un modelo ajustado para un conjunto de variables dado.
- ② Define una función que proporciona como resultado una puntuación para un modelo y un conjunto de variables dados. En este caso, usamos el `AIC_score` implementado en el paquete `dmbe`.

La función ha elegido un modelo en el que se han eliminado varias variables de `house_full: SqFtLot, NbrLivingUnits, YrRenovated y NewConstruction`.

Más simples aún son la *selección hacia delante* y la *selección hacia atrás* (*forward selection* y *backward selection*). En la selección hacia delante, comenzamos sin predictoras y se añaden una por una, incorporando en cada paso la predictorá que aporta la mayor contribución a R^2 y se detiene cuando la contribución ya no es estadísticamente significativa. En la selección hacia atrás, o eliminación hacia atrás (*backward elimination*), comenzamos con el modelo completo y eliminamos las predictoras que no son estadísticamente significativas hasta que nos quedamos con un modelo en el que todas las predictoras son estadísticamente significativas.

La *regresión penalizada* (*penalized regression*) es similar en espíritu a AIC. En lugar de buscar directamente a través de un conjunto discreto de modelos, la ecuación de ajuste del modelo incorpora una restricción que lo penaliza en caso de utilizar demasiadas variables (parámetros). En lugar de eliminar por completo las variables predictoras, como ocurre con la selección por pasos, hacia delante y hacia atrás, la regresión penalizada aplica la penalización reduciendo los coeficientes, en algunos casos hasta casi cero. Los métodos de regresión penalizados habituales son la *regresión de crestas* (*ridge regression*) y la *regresión de lazo* (*lasso regression*).

La regresión escalonada y la regresión de todos los subconjuntos son métodos *dentro de la muestra* (*in-sample*) para evaluar y ajustar modelos. Esto significa que la selección del modelo posiblemente esté sujeta a sobreajuste (ajustando el ruido en los datos) y puede no funcionar tan bien cuando se aplica a nuevos datos. Un enfoque frecuente para evitar lo anterior es

utilizar la validación cruzada para validar los modelos. En regresión lineal, el sobreajuste no suele ser un problema importante, debido a la estructura global simple (lineal) impuesta a los datos. Para los tipos de modelos más sofisticados, en particular los procedimientos iterativos que responden a una estructura de datos local, la validación cruzada es una herramienta muy importante. Consultar “Validación cruzada” en la página 155 para ampliar detalles.

Regresión ponderada

Los estadísticos utilizan la regresión ponderada para una serie de propósitos. En particular, es importante para el análisis de encuestas complejas. Los científicos de datos pueden encontrar útil la regresión ponderada en dos casos:

- En la ponderación de la varianza inversa, cuando se han medido distintas observaciones con diferente precisión. Las de mayor varianza reciben ponderaciones más bajas.
- En el análisis de datos, donde las filas representan múltiples casos. La variable de ponderación codifica cuántas observaciones originales representa cada fila.

Por ejemplo, con los datos de las viviendas, las ventas más antiguas son menos confiables que las ventas más recientes. Usando DocumentDate para determinar el año de la venta, podemos calcular Weight como el número de años desde 2005 (el comienzo de los datos):

R

```
library(lubridate)
house$Year = year(house$DocumentDate)house$Weight = house$Year - 2005
```

Python

```
house['Year'] = [int(date.split('-')[0]) for date in house.DocumentDate]
house['Weight'] = house.Year - 2005
```

Podemos calcular la regresión ponderada con la función lm utilizando el argumento weight:

```
house_wt <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
  Bedrooms + BldgGrade,
  data=house, weight=Weight)
round(cbind(house_lm=house_lm$coefficients,
  house_wt=house_wt$coefficients), digits=3)
```

	house_lm	house_wt
(Intercept)	-521871.368	-584189.329
SqFtTotLiving	228.831	245.024
SqFtLot	-0.060	-0.292
Bathrooms	-19442.840	-26085.970
Bedrooms	-47769.955	-53608.876
BldgGrade	106106.963	115242.435

Los coeficientes de la regresión ponderada son ligeramente diferentes de los de la regresión original.

La mayoría de los modelos en `scikit-learn` aceptan ponderaciones como el argumento de palabra clave `sample_weight` en la llamada del método `fit`:

```
predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade']
outcome = 'AdjSalePrice'

house_wt = LinearRegression()
house_wt.fit(house[predictors], house[outcome], sample_weight=house.Weight)
```

Ideas clave

- La regresión lineal múltiple modela la relación entre una variable de respuesta Y y múltiples variables predictoras X_1, \dots, X_p .
- Las métricas más importantes para evaluar un modelo son el error cuadrático medio RECM (RMSE en inglés) y R cuadrado (R^2).
- El error estándar de los coeficientes se puede utilizar para medir la confiabilidad de la contribución de una variable a un modelo.
- La regresión escalonada es una forma de determinar automáticamente qué variables deben incluirse en el modelo.
- La regresión ponderada se usa para dar a ciertos registros más o menos ponderación al ajustar la ecuación.

Lecturas complementarias

Se puede encontrar un excelente tratamiento de la validación cruzada y el remuestreo en *An Introduction to Statistical Learning* de Gareth James, Daniela Witten, Trevor Hastie, y Robert Tibshirani (Springer, 2013).

Pronóstico mediante la regresión

El propósito principal de la regresión en la ciencia de datos es el pronóstico. Es importante tenerlo en cuenta, ya que la regresión, al ser un método estadístico usado desde hace mucho tiempo y consolidado, viene con un bagaje que es más apropiado para su función tradicional como herramienta para el modelado descriptivo que para el pronóstico.

Términos clave del pronóstico mediante regresión

Intervalo de pronóstico

Intervalo de incertidumbre en torno a un valor individual pronosticado.

Extrapolación

Extensión de un modelo fuera del rango de los datos utilizados para su ajuste.

Los peligros de la extrapolación

Los modelos de regresión no deben emplearse para hacer una extrapolación fuera del rango de los datos (dejando de lado la utilización de la regresión para el pronóstico de series de tiempo). El modelo es válido solo para valores predictores para los que los datos tienen valores suficientes (incluso en el caso de que haya suficientes datos disponibles, podrían existir otros problemas. Consultar “Diagnósticos de regresión” en la página 176). Como caso extremo, supongamos que `model_1m` se usa para pronosticar el valor de un solar vacío de 5000 pies cuadrados. En tal caso, todos los valores predictores relacionados con el edificio tendrían un valor de 0, y la ecuación de regresión arrojaría un pronóstico absurdo de $-521\,900 + 5000 \times -.0605 = -522\,202$ \$. ¿Por qué ha ocurrido esto? Los datos contienen solo parcelas construidas, no hay registros correspondientes a solares vacíos. En consecuencia, el modelo no tiene información que le diga cómo pronosticar el precio de venta de los solares vacíos.

Intervalos de confianza y de pronóstico

Gran parte de la estadística consiste en comprender y medir la variabilidad (incertidumbre). Los estadísticos t y los valores p que aparecen en el resultado de la regresión la tratan de una manera formal, lo que a veces es útil para la selección de variables (consultar “Evaluación del modelo” en la página 153). Las métricas más útiles son los intervalos de confianza, que son intervalos de incertidumbre situados en torno a los coeficientes de regresión y los pronósticos. Una manera sencilla de entenderlo es mediante bootstrap (consultar “Bootstrap” en la página 61 para ampliar detalles sobre el procedimiento general de bootstrap). Los intervalos de confianza de regresión más habituales que se encuentran en el resultado que proporciona el software son los de los parámetros de regresión (coeficientes). A continuación se muestra un algoritmo bootstrap para generar intervalos de confianza para parámetros de regresión (coeficientes) de un conjunto de datos con P predictoras y n registros (filas):

1. Consideraremos cada fila (incluida la variable de resultado) como un solo "ticket" y colocamos todos los tickets n en una caja.
2. Extraemos un ticket al azar, registramos los valores y lo volvemos a colocar en la caja.
3. Repetimos n veces el paso 2. Ahora tenemos un remuestreo de bootstrap.
4. Ajustamos la regresión a la muestra bootstrap y registramos los coeficientes estimados.
5. Repetimos los pasos del 2 al 4, por ejemplo, 1000 veces.
6. Ahora tenemos 1000 valores de bootstrap para cada coeficiente. Encontramos los percentiles apropiados para cada uno (por ejemplo, 5° y 95° para un intervalo de confianza del 90 %).

Podemos usar la función `Boot` de *R* para generar intervalos de confianza bootstrap reales para los coeficientes, o simplemente podemos usar los intervalos basados en fórmulas que son un resultado rutinario de *R*. El significado desde el punto de vista conceptual y de interpretación es el mismo, y no tiene una importancia fundamental para los científicos de datos, porque se refieren a los coeficientes de regresión. De mayor interés para ellos son los intervalos en torno a los valores y pronosticados (\hat{Y}_i). La incertidumbre en torno a \hat{Y}_i proviene de dos fuentes:

- La incertidumbre sobre cuáles son las variables predictoras relevantes y sus coeficientes (consultar el algoritmo bootstrap anterior).
- El error adicional inherente a puntos de datos individuales.

El error del punto de datos individual puede pensarse de la siguiente manera: incluso si supiéramos con certeza cuál era la ecuación de regresión (por ejemplo, si tuviéramos una gran cantidad de registros para ajustarla), los valores de resultado *reales (actual)* para un conjunto dado de valores de las predictoras variarán. Por ejemplo, varias viviendas, cada una con 8 habitaciones, un solar de 6500 pies cuadrados, 3 baños y un sótano, pueden tener valores diferentes. Podemos modelar este error individual con los residuos de los valores ajustados. El algoritmo bootstrap para modelar tanto el error del modelo de regresión como el error del punto de datos individual se vería de la siguiente manera:

1. Extraemos una muestra bootstrap de los datos (explicada con mayor detalle anteriormente).
2. Ajustamos la regresión y pronosticamos el nuevo valor.
3. Extraemos un único residuo al azar del ajuste de regresión original, lo añadimos al valor pronosticado y registramos el resultado.
4. Repetimos los pasos del 1 al 3, por ejemplo, 1000 veces.
5. Encontramos los percentiles 2.5 y 9.5 de los resultados.

Ideas clave

- La extrapolación fuera del rango de los datos puede provocar errores.
- Los intervalos de confianza cuantifican la incertidumbre en torno a los coeficientes de regresión.
- Los intervalos de pronóstico cuantifican la incertidumbre en los pronósticos individuales.
- La mayoría del software, incluido *R*, elaborará pronósticos e intervalos de confianza que presentará en la salida predeterminada o especificada, utilizando fórmulas.
- Bootstrap también se puede utilizar para elaborar pronósticos e intervalos de confianza. La interpretación y la idea son las mismas.



¿Intervalo de pronóstico o intervalo de confianza?

El intervalo de pronóstico se refiere a la incertidumbre en torno a un solo valor, mientras que el intervalo de confianza se refiere a una media u otro estadístico calculado a partir de múltiples valores. Por tanto, para un mismo valor, el intervalo de pronóstico suele ser mucho más amplio que el intervalo de confianza. Modelamos el error del valor individual en el modelo bootstrap seleccionando un residuo individual para agregar al valor pronosticado. ¿Cuál de ellos deberíamos utilizar? Depende del contexto y del propósito del análisis, pero, en general, los científicos de datos están interesados en pronósticos individuales específicos, por lo que sería más apropiado un intervalo de pronóstico. Usar un intervalo de confianza cuando deberíamos usar un intervalo de pronóstico subestimaría en gran medida la incertidumbre en el valor pronosticado dado.

VARIABLES DE TIPO FACTOR EN LA REGRESIÓN

Las variables de tipo *factor* (*factor*), también denominadas variables *categóricas* (*categorical*), adoptan un número limitado de valores discretos. Por ejemplo, el propósito de un préstamo puede ser “consolidación de deuda”, “boda”, “automóvil”, etc. La variable binaria (sí/no), también llamada variable *indicadora* (*indicator*), es un caso especial de una variable de tipo factor. La regresión requiere entradas numéricas, por lo que las variables de tipo factor deben codificarse de nuevo para utilizarlas en el modelo. El planteamiento más habitual es convertir una variable en un conjunto de variables *ficticias* (*dummy*) binarias.

Términos clave de las variables de tipo factor

Variables ficticias

Variables binarias 0-1 derivadas de una nueva codificación de los datos de los factores para su utilización en la regresión y otros modelos.

Codificación de referencia

Es el tipo de codificación más habitual que utilizan los estadísticos, en el que el nivel de un factor se usa como referencia y otros factores se comparan con ese nivel.

Sinónimo

codificación de tratamiento

Codificador One-Hot

Tipo frecuente de codificación utilizado en el colectivo de aprendizaje automático en el que se conservan todos los niveles del factor. Si bien es útil para ciertos algoritmos de aprendizaje automático, este enfoque no es apropiado para la regresión lineal múltiple.

Codificación de desviación

Tipo de codificación que compara cada nivel con la media general en lugar de hacerlo con el nivel de referencia.

Sinónimo

contrastos de suma

Representación de variables ficticias

En los datos de las viviendas del condado de King, hay una variable de tipo factor para el tipo de propiedad. A continuación se presenta un pequeño subconjunto de seis registros:

R:

```
head(house[, 'PropertyType']) Source: local data frame [6 x 1]
```

	PropertyParams (fctr)
1	Multiplex
2	Single Family
3	Single Family
4	Single Family
5	Single Family
6	Townhouse

Python:

```
house.PropertyType.head()
```

Hay tres valores posibles: Multiplex, Single Family y Townhouse. Para usar esta variable de tipo factor, necesitamos convertirla en un conjunto de variables binarias. Lo hacemos creando una variable binaria para cada valor posible de la variable de tipo factor. Para hacer esto con R, utilizamos la función⁵ `model.matrix`:

```
prop_type_dummies <- model.matrix(~.PropertyType -1, data=house)
head(prop_type_dummies)
  PropertyTypeMultiplex PropertyTypeSingle Family PropertyTypeTownhouse
1                      1                     0                 0
2                      0                     1                 0
3                      0                     1                 0
4                      0                     1                 0
5                      0                     1                 0
6                      0                     0                 1
```

La función `model.matrix` convierte un marco de datos en una matriz compatible con un modelo lineal. La variable de tipo factor `.PropertyType`, que tiene tres niveles distintos, se representa como una matriz con tres columnas. En el colectivo de aprendizaje automático, esta representación se conoce como *codificación One-Hot (One-Hot encoding)* (consultar “Codificador One-Hot” en la página 242).

En *Python*, podemos convertir las variables categóricas en ficticias utilizando el método `get_dummies` de `pandas`:

```
pd.get_dummies(house['.PropertyType']).head() ❶
pd.get_dummies(house['.PropertyType'], drop_first=True).head() ❷
```

- ❶ Devuelve por defecto la codificación One-Hot de la variable categórica.

⁵ El argumento `-1` en `model.matrix` produce una representación de codificación One-Hot (eliminando la intersección, de ahí el `"-"`). De lo contrario, el valor predeterminado en R es generar una matriz con $P - 1$ columnas con el primer nivel del factor como referencia.

Estadística práctica para ciencia de datos con R y Python

- ② El argumento de palabra clave `drop_first` devolverá $P - 1$ columnas. Lo utilizamos para evitar el problema de la multicolinealidad.

En ciertos algoritmos de aprendizaje automático, como los modelos de árbol y los de vecinos más cercanos, la codificación One-Hot es la forma estándar de representar las variables de tipo factor (por ejemplo, consultar “Modelos de árbol” en la página 249).

En la configuración de regresión, una variable de tipo factor con P niveles distintos generalmente se representa mediante una matriz con solo $P - 1$ columnas. Esto se debe a que un modelo de regresión generalmente incluye un término de intersección. Con una intersección, una vez que hayamos definido los valores para los binarios $P - 1$, se conoce el valor de P ésimo y podría considerarse redundante. Agregar la columna P ésimo provocará un error de multicolinealidad (consultar “Multicolinealidad” en la página 172).

La representación por defecto en *R* es utilizar el primer nivel del factor como *referencia* (*reference*) e interpretar los niveles restantes en relación con ese factor:

```
lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
   Bedrooms + BldgGrade + PropertyType, data=house)
```

Call:

```
lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
   Bedrooms + BldgGrade + PropertyType, data = house)
```

Coefficients:

	(Intercept)	SqFtTotLiving
	-4.468e+05	2.234e+02
	SqFtLot	Bathrooms
	-7.037e-02	-1.598e+04
	Bedrooms	BldgGrade
	-5.089e+04	1.094e+05
PropertyTypeSingle Family		PropertyTypeTownhouse
	-8.468e+04	-1.151e+05

El método `get_dummies` toma el argumento de palabra clave opcional `drop_first` para excluir el primer factor como *referencia* (*reference*):

```
predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms',
              'BldgGrade', 'PropertyType']

X = pd.get_dummies(house[predictors], drop_first=True)

house_lm_factor = LinearRegression()
house_lm_factor.fit(X, house[outcome])

print(f'Intercept: {house_lm_factor.intercept_:.3f}')
print('Coefficients:')
for name, coef in zip(X.columns, house_lm_factor.coef_):
    print(f'{name}: {coef}'')
```

El resultado de la regresión *R* muestra dos coeficientes correspondientes al tipo de propiedad: `PropertyTypeSingle Family` y `PropertyTypeTownhouse`. No hay coeficiente de

Multiplex ya que se define implícitamente cuando `PropertyTypeSingle Family == 0` y `PropertyTypeTownhouse == 0`. Los coeficientes se interpretan como relativos a Multiplex, por lo que una vivienda Single Family vale casi 85 000 \$ menos, y una vivienda Townhouse vale 150 000 \$ menos⁶.



Diferentes codificaciones de los factores

Hay varias formas diferentes de codificar variables de tipo factor, a las que se conoce como sistemas de *codificación de contraste* (*contrast coding*). Por ejemplo, la codificación de desviación, también conocida como *contrastes de suma* (*sum contrasts*), compara cada nivel con la media general. Otro contraste es la *codificación polinomial* (*polynomial coding*), que es apropiada para factores ordenados. Consultar la sección “Variables de tipo factor ordenadas” en la página 169. Con la excepción de los factores ordenados, los científicos de datos generalmente no encontrarán ningún tipo de codificación además de la codificación de referencia o el codificador One-Hot.

Variables de tipo factor con muchos niveles

Algunas variables de tipo factor pueden producir una gran cantidad de variables ficticias binarias: los códigos postales son una variable de tipo factor y hay 43 000 códigos postales en EE. UU. En tales casos, es conveniente explorar los datos y las relaciones entre las variables predictoras y el resultado para determinar si las categorías contienen información útil. Si es así, debemos decidir además si es conveniente retener todos los factores o si los niveles deben consolidarse.

En el condado de King, hay 80 códigos postales en los que se han producido ventas de viviendas:

```
table(house$ZipCode)
```

98001	98002	98003	98004	98005	98006	98007	98008	98010	98011	98014	98019
358	180	241	293	133	460	112	291	56	163	85	242
98022	98023	98024	98027	98028	98029	98030	98031	98032	98033	98034	98038
188	455	31	366	252	475	263	308	121	517	575	788
98039	98040	98042	98043	98045	98047	98050	98051	98052	98053	98055	98056
47	244	641	1	222	48	7	32	614	499	332	402
98057	98058	98059	98065	98068	98070	98072	98074	98075	98077	98092	98102
4	420	513	430	1	89	245	502	388	204	289	106
98103	98105	98106	98107	98108	98109	98112	98113	98115	98116	98117	98118
671	313	361	296	155	149	357	1	620	364	619	492
98119	98122	98125	98126	98133	98136	98144	98146	98148	98155	98166	98168
260	380	409	473	465	310	332	287	40	358	193	332
98177	98178	98188	98198	98199	98224	98288	98354				
216	266	101	225	393	3	4	9				

⁶ Esto no es intuitivo, pero puede explicarse por el impacto de la localización como una variable de confusión. Consultar “Variables de confusión” en la página 172.

Estadística práctica para ciencia de datos con R y Python

El método `value_counts` de los marcos de datos de pandas devuelve la misma información:

```
pd.DataFrame(house['ZipCode'].value_counts()).transpose()
```

`ZipCode` es una variable importante, ya que es un proxy del efecto de la localización en el valor de la vivienda. Incluir todos los niveles requiere 79 coeficientes correspondientes a 79 grados de libertad. El modelo original `house_lm` tiene solo 5 grados de libertad; consultar “Evaluación del modelo” en la página 153. Además, varios códigos postales tienen una sola venta. En algunos problemas, podemos consolidar un código postal utilizando los primeros dos o tres dígitos, correspondientes a una región geográfica submetropolitana. Para el condado de King, casi todas las ventas ocurren en 980xx o 981xx, por lo que esta circunstancia no ayuda.

Un enfoque alternativo es agrupar los códigos postales de acuerdo con otra variable, como el precio de venta. Otra opción aún mejor es formar grupos de códigos postales utilizando los residuos del modelo inicial. El siguiente código `dplyr` de R consolida los 80 códigos postales en cinco grupos según la mediana de los residuos de la regresión `house_lm`:

```
zip_groups <- house %>%
  mutate(resid = residuals(house_lm)) %>%
  group_by(ZipCode) %>%
  summarize(med_resid = median(resid),
            cnt = n()) %>%
  arrange(med_resid) %>%
  mutate(cum_cnt = cumsum(cnt),
        ZipGroup = ntile(cum_cnt, 5))
house <- house %>%
  left_join(select(zip_groups, ZipCode, ZipGroup), by='ZipCode')
```

La mediana de los residuos se calcula para cada código postal, y la función `ntile` se utiliza para dividir los códigos postales, ordenados por la mediana, en cinco grupos. Consultar “Variables de confusión” en la página 172 para ver un ejemplo de cómo se utiliza esta opción como un término en la regresión que mejora el ajuste original.

En *Python* podemos determinar esta información de la siguiente manera:

```
zip_groups = pd.DataFrame([
    *pd.DataFrame({
        'ZipCode': house['ZipCode'],
        'residual' : house[outcome] - house_lm.predict(house[predictors]),
    })
    .groupby(['ZipCode'])
    .apply(lambda x: {
        'ZipCode': x.iloc[0,0],
        'count': len(x),
        'median_residual': x.residual.median()
    })
])
.sort_values('median_residual')
zip_groups['cum_count'] = np.cumsum(zip_groups['count'])
zip_groups['ZipGroup'] = pd.qcut(zip_groups['cum_count'], 5, labels=False,
                                 retbins=False)
to_join = zip_groups[['ZipCode', 'ZipGroup']].set_index('ZipCode')
```

```
house = house.join(to_join, on='ZipCode')
house['ZipGroup'] = house['ZipGroup'].astype('category')
```

El concepto de utilizar los residuos para ayudar a orientar el ajuste de regresión es un paso fundamental en el proceso de modelado. Consultar “Diagnósticos de regresión” en la página 176.

Variables de tipo factor ordenadas

Algunas variables de tipo factor reflejan los niveles de un factor. Estas se denominan variables de tipo factor ordenadas o variables categóricas ordenadas. Por ejemplo, la calificación del préstamo podría ser A, B, C, etc., cada calificación conlleva más riesgo que la calificación anterior. A menudo, las variables de tipo factor ordenadas se pueden convertir en valores numéricos y usarse tal cual. Por ejemplo, la variable `BldgGrade` es una variable de tipo factor ordenada. En la tabla 4.1 se muestran varios tipos de calificaciones. Si bien las calificaciones tienen un significado específico, el valor numérico está ordenado de menor a mayor, correspondiendo con la calificación ascendente de las viviendas. Con el modelo de regresión `house_lm`, que se ajusta al de “Regresión lineal múltiple” en la página 150, `BldgGrade` se ha tratado como una variable numérica.

Tabla 4.1 Calificaciones de los inmuebles y equivalentes numéricos

Valor	Descripción
1	Inmuble pequeño
2	Deficiente
5	Aceptable
10	Muy bueno
12	Lujoso
13	Mansión

Tratar los factores ordenados como una variable numérica conserva la información contenida en la ordenación, la cual se perdería si se convirtiera en un factor.

Ideas clave

- Las variables de tipo factor deben convertirse en variables numéricas para utilizarlas en la regresión.
- El método más extendido para codificar una variable de tipo factor con distintos valores de P es representarlos usando variables ficticias P – 1.
- Es posible que sea necesario consolidar una variable de tipo factor con muchos niveles, incluso en conjuntos de datos muy grandes, en una variable con menos niveles.
- Algunos factores tienen niveles que están ordenados y se pueden representar como una única variable numérica.

Interpretación de la ecuación de regresión

En ciencia de datos, el uso más importante de la regresión es pronosticar alguna variable dependiente (resultado). En algunos casos, sin embargo, puede resultar conveniente conocer la ecuación en sí para comprender la naturaleza de la relación entre las predictoras y el resultado. Esta sección proporciona orientación sobre cómo examinar la ecuación de regresión e interpretarla.

Términos clave de la interpretación de la ecuación de regresión

Variables correlacionadas

Variables que tienden a moverse en la misma dirección, cuando una sube también lo hace la otra, y viceversa (con correlación negativa, cuando una sube, la otra baja).

Cuando las variables predictoras están altamente correlacionadas, es difícil interpretar los coeficientes individuales.

Multicolinealidad

Cuando las variables predictoras tienen una correlación perfecta o casi perfecta, la regresión puede ser inestable o imposible de calcular.

Sinónimo

colinealidad

Variables de confusión

Es una predictor variable importante que, cuando se omite, conduce a relaciones falsas en una ecuación de regresión.

Efectos principales

Relación entre una predictor variable y la variable de resultado, independientemente de otras variables.

Interacciones

Relación interdependiente entre dos o más predictoras y la respuesta.

Predictoras correlacionadas

En regresión múltiple, las variables predictoras a menudo están correlacionadas. Como ejemplo, examinemos los coeficientes de regresión para el modelo `step_lm`, incluido en el apartado “Selección del modelo y regresión escalonada” en la página 156.

R:

step_lm\$coefficients		
(Intercept)		Bathrooms
6.178645e+06	1.992776e+02	4.239616e+04
Bedrooms	BldgGrade	PropertyTypeSingle Family
-5.194738e+04	1.371596e+05	2.291206e+04
PropertyTypeTownhouse	SqFtFinBasement	YrBuilt
8.447916e+04	7.046975e+00	-3.565425e+03

Python:

```
print(f'Intercept: {best_model.intercept_:.3f}')
print('Coefficients:')
for name, coef in zip(best_variables, best_model.coef_):
    print(f'{name}: {coef}')
```

El coeficiente para `Bedrooms` es negativo! Esto implica que agregar un dormitorio a una vivienda reducirá su valor. ¿Cómo puede ocurrir esto? Se debe a que las variables predictoras están correlacionadas: las viviendas más grandes tienden a tener más dormitorios y es el tamaño lo que determina el valor de la vivienda, no el número de dormitorios. Pensemos dos viviendas exactamente del mismo tamaño: es razonable esperar que una vivienda con más habitaciones pero más pequeñas se considere menos deseable.

Tener predictoras correlacionadas puede dificultar la interpretación del signo y el valor de los coeficientes de regresión (y puede inflar el error estándar de las estimaciones). Las variables para los dormitorios, el tamaño de la vivienda y el número de baños están correlacionadas. Esto se ilustra en el siguiente ejemplo en *R*, que se ajusta a otra regresión que elimina las variables `SqFtTotLiving`, `SqFtFinBasement`, y `Bathrooms` de la ecuación:

```
update(step_lm, . ~ . - SqFtTotLiving - SqFtFinBasement - Bathrooms)
```

Call:

```
lm(formula = AdjSalePrice ~ Bedrooms + BldgGrade + PropertyType +
YrBuilt, data = house, na.action = na.omit)
```

Coefficients:

	(Intercept)	Bedrooms
	4913973	27151
BldgGrade	248998	PropertyTypeSingleFamily
		-19898
PropertyTypeTownhouse	-47355	YrBuilt
		-3212

La función `update` se puede utilizar para agregar o eliminar variables de un modelo. Ahora el coeficiente para los dormitorios es positivo, en línea con lo que cabría esperar (aunque en realidad actúa como un sustituto del tamaño de la vivienda, ahora que se han eliminado esas variables).

En *Python*, no hay equivalente a la función de actualización de *R*. Necesitamos reajustar el modelo con la lista de predictoras modificada:

```
predictors = ['Bedrooms', 'BldgGrade', 'PropertyType', 'YrBuilt']
outcome = 'AdjSalePrice'

X = pd.get_dummies(house[predictors], drop_first=True)
reduced_lm = LinearRegression()
reduced_lm.fit(X, house[outcome])
```

Las variables correlacionadas solo son un problema al interpretar los coeficientes de regresión. En `house_lm`, no hay ninguna variable para dar cuenta de la localización de

la vivienda, y el modelo mezcla tipos de regiones muy diferentes. La localización puede ser una variable de confusión. Consultar “Variables de confusión” en la página 172 para obtener más información.

Multicolinealidad

Un caso extremo de variables correlacionadas da lugar a la multicolinealidad, una condición en la que hay redundancia entre las variables predictoras. La multicolinealidad perfecta ocurre cuando una variable predictora puede expresarse como una combinación lineal de otras. La multicolinealidad ocurre cuando:

- Se incluye una variable varias veces por error.
- Se crean P ficticias, en lugar de las $P - 1$, a partir de una variable de tipo factor (consultar “Variables de tipo factor en la regresión” en la página 163).
- Dos variables están casi perfectamente correlacionadas entre sí.

Se debe abordar la multicolinealidad en la regresión. Las variables deben eliminarse hasta que desaparezca la multicolinealidad. Una regresión no tiene una solución bien definida en presencia de una multicolinealidad perfecta. Muchos paquetes de software, incluidos *R* y *Python*, manejan automáticamente ciertos tipos de multicolinealidad. Por ejemplo, si `SqFtTotLiving` se incluye dos veces en la regresión de los datos de las viviendas, los resultados son los mismos que para el modelo `house_lm`. En el caso de multicolinealidad no perfecta, el software puede proporcionar una solución, pero los resultados pueden ser inestables.



La multicolinealidad no es un problema para los métodos de regresión no lineal como los métodos de árboles, de agrupamiento y de vecinos más cercanos, y en tales métodos puede ser aconsejable retener variables ficticias P (en lugar de $P - 1$). Dicho esto, incluso en esos métodos, la no redundancia en las variables predictoras sigue siendo una virtud.

Variabes de confusión

Con las variables correlacionadas, el problema es de acción: incluir diferentes variables que tienen una relación predictiva similar con la respuesta. Con las *variables de confusión* (*confounding variables*), el problema es de omisión: una variable importante no está incluida en la ecuación de regresión. La interpretación ingenua de los coeficientes de la ecuación puede llevar a conclusiones no válidas.

Tomemos, por ejemplo, la ecuación de regresión del condado de King `house_lm` del “Ejemplo: datos de las viviendas del condado de King” en la página 151. Los coeficientes de regresión de `SqFtLot`, `Bathrooms` y `Bedrooms` son todos negativos. El modelo de

Regresión y pronóstico

regresión original no contiene una variable para representar la localización, una predictor muy importante del precio de la vivienda.

Para modelar la localización, incluimos la variable ZipGroup para que clasifique el código postal en uno de cinco grupos, desde el más económico (1) al más caro (5)⁷:

```
lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms + Bedrooms +
  BldgGrade + PropertyType + ZipGroup, data = house, na.action = na.omit)
```

Coefficients:

	(Intercept)	SqFtTotLiving
	-6.666e+05	2.106e+02
	SqFtLot	Bathrooms
	4.550e-01	5.928e+03
	Bedrooms	BldgGrade
	-4.168e+04	9.854e+04
PropertyTypeSingle Family	1.932e+04	PropertyTypeTownhouse
	ZipGroup2	-7.820e+04
	5.332e+04	ZipGroup3
	ZipGroup4	1.163e+05
	1.784e+05	ZipGroup5
		3.384e+05

El mismo modelo en *Python*:

```
predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms',
  'BldgGrade', 'PropertyType', 'ZipGroup']
outcome = 'AdjSalePrice'

X = pd.get_dummies(house[predictors], drop_first=True)

confounding_lm = LinearRegression()
confounding_lm.fit(X, house[outcome])

print(f'Intercept: {confounding_lm.intercept_:.3f}')
print('Coefficients:')
for name, coef in zip(X.columns, confounding_lm.coef_):
  print(f' {name}: {coef}'')
```

ZipGroup es claramente una variable importante: se estima que una vivienda en el grupo del código postal en el que son más caras tiene un precio de venta más alto en casi 340 000 \$. Los coeficientes de SqFtLot y Bathrooms son ahora positivos, y agregar un baño aumenta el precio de venta en 5928 \$.

El coeficiente de Bedrooms sigue siendo negativo. Si bien este hecho no es intuitivo, es un fenómeno bien conocido en el sector inmobiliario. En el caso de viviendas de la misma superficie habitable y número de baños, tener más dormitorios y por tanto más pequeños, se asocia con viviendas de menor valor.

⁷ Hay 80 códigos postales en el condado de King, varios con solo un puñado de ventas. Una alternativa al uso directo del código postal como variable de tipo factor, ZipGroup agrupa códigos postales similares en un solo grupo. Consultar "Variables de tipo factor con muchos niveles" en la página 167 para ampliar detalles.

Interacciones y efectos principales

A los estadísticos les gusta distinguir entre los efectos principales (*main effects*), o variables independientes, y las *interacciones* (*interactions*) entre los efectos principales. Los efectos principales son lo que a menudo se denominan variables predictoras (*predictor variables*) en la ecuación de regresión. Una suposición implícita cuando solo se utilizan efectos principales en un modelo es que la relación entre una variable predictora y la respuesta es independiente de las otras variables predictoras. Con frecuencia este no es el caso.

Por ejemplo, el modelo que se ajusta a los datos de las viviendas del condado de King en “Variables de confusión” en la página 172 contiene varias variables como efectos principales, incluido `ZipCode`. La ubicación en el sector inmobiliario lo es todo, y es natural suponer que la relación entre, digamos, el tamaño de la vivienda y el precio de venta depende de la ubicación. Una vivienda grande construida en un distrito de renta baja no va a conservar el mismo valor que una vivienda grande construida en un área cara. En *R* podemos incluir interacciones entre variables usando el operador `*`. Para los datos del condado de King, lo siguiente se ajusta a una interacción entre `SqFtTotLiving` y `ZipGroup`:

```
lm(formula = AdjSalePrice ~ SqFtTotLiving * ZipGroup + SqFtLot +
  Bathrooms + Bedrooms + BldgGrade + PropertyType, data = house,
  na.action = na.omit)
```

Coefficients:

	(Intercept)	SqFtTotLiving
	-4.853e+05	1.148e+02
	ZipGroup2	ZipGroup3
	-1.113e+04	2.032e+04
	ZipGroup4	ZipGroup5
	2.050e+04	-1.499e+05
	SqFtLot	Bathrooms
	6.869e-01	-3.619e+03
	Bedrooms	BldgGrade
	-4.180e+04	1.047e+05
PropertyTypeSingle Family	1.357e+04	PropertyTypeTownhouse
	3.260e+01	-5.884e+04
SqFtTotLiving:ZipGroup2	3.260e+01	SqFtTotLiving:ZipGroup3
SqFtTotLiving:ZipGroup4	6.934e+01	4.178e+01
		SqFtTotLiving:ZipGroup5
		2.267e+02

El modelo resultante tiene cuatro términos nuevos: `SqFtTotLiving:ZipGroup2`, `SqFtTotLiving:ZipGroup3`, etc.

En *Python*, necesitamos utilizar el paquete `statsmodels` para entrenar modelos de regresión lineal con interacciones. Este paquete se diseñó de manera similar a *R* y permite definir modelos usando la interfaz `formula`:

```
model = smf.ols(formula='AdjSalePrice ~ SqFtTotLiving*ZipGroup + SqFtLot + '
  'Bathrooms + Bedrooms + BldgGrade + PropertyType', data=house)
results = model.fit()
results.summary()
```

El paquete `statsmodels` se ocupa de las variables categóricas (por ejemplo, `ZipGroup[T.1]`, `.PropertyType[T.Single Family]`) y los términos de interacción (por ejemplo, `SqFtTotLiving:ZipGroup[T.1]`).

La ubicación y el tamaño de la vivienda parecen tener una fuerte interacción. Para una vivienda en el `zipGroup` más bajo, la pendiente es la misma que la pendiente para el efecto principal `SqFtTotLiving`, que es de 115 \$ por pie cuadrado (esto se debe a que *R* utiliza una codificación de referencia para variables de tipo factor. Consultar “Variables de tipo factor en la regresión” en la página 163). Para una vivienda con el `zipGroup` más alto, la pendiente es la suma del efecto principal más `SqFtTotLiving:ZipGroup5`, o de 115 \$ + 227 \$ = 342 \$ por pie cuadrado. En otras palabras, añadir un pie cuadrado en el grupo de código postal más caro aumenta el precio de venta previsto en un factor de casi tres, en comparación con el aumento promedio de añadir un pie cuadrado en el grupo de código postal más bajo.



Selección de modelos con los términos de interacción

En problemas en los que están involucradas muchas variables, puede resultar difícil decidir qué términos de interacción deben incluirse en el modelo. Por lo general, se adoptan enfoques diferentes:

- En algunos problemas, el conocimiento previo y la intuición pueden orientar en la elección de qué términos de interacción incluir en el modelo.
- Se puede utilizar la selección escalonada (consultar “Selección del modelo y regresión escalonada” en la página 156) para examinar los distintos modelos.
- La regresión penalizada puede ajustarse automáticamente a un gran conjunto de posibles términos de interacción.
- Quizás el criterio más extendido es utilizar *modelos de árbol* (*tree models*), así como sus descendientes, *bosques aleatorios* (*random forest*) y *árboles potenciados por gradientes* (*gradient boosted trees*). Esta clase de modelos busca automáticamente términos de interacción óptimos. Consultar “Modelos de árbol” en la página 249.

Ideas clave

- Debido a la correlación entre predictoras, se debe tener cuidado a la hora de interpretar los coeficientes en la regresión lineal múltiple.
- La multicolinealidad puede causar inestabilidad numérica al ajustar la ecuación de regresión.
- Una variable de confusión es una predictora importante que se omite en un modelo y puede llevar a una ecuación de regresión con relaciones espurias.
- Se necesita un término de interacción entre dos variables si la relación entre las variables y la respuesta es interdependiente.

Diagnósticos de regresión

En el modelado descriptivo (es decir, en un contexto de investigación), se ejecutan varios pasos, además de las métricas mencionadas anteriormente (consultar “Evaluación del modelo” en la página 153), para evaluar lo bien que se ajusta el modelo a los datos. La mayoría de ellos se basan en el análisis de los residuos. Estos pasos no abordan directamente la precisión del pronóstico, pero pueden proporcionar información útil en un entorno predictivo.

Términos clave de los diagnósticos de regresión

Residuos estandarizados

Residuos divididos por el error estándar de los residuos.

Valores atípicos

Registros (o valores de resultado) que están distantes del resto de los datos (o del resultado previsto).

Valor influyente

Valor o registro cuya presencia o ausencia marca una gran diferencia en la ecuación de regresión.

Apalancamiento

Grado de influencia que tiene un solo registro en una ecuación de regresión.

Sinónimo

valor con circunflejo

Residuos anormales

Los residuos no distribuidos normalmente pueden invalidar algunos requisitos técnicos de la regresión, pero generalmente no son una preocupación en la ciencia de datos.

Heterocedasticidad

Cuando algunos rangos del resultado experimentan residuos con mayor varianza (puede indicar que falta un predictor en la ecuación).

Diagrama de residuos parciales

Gráfico de diagnóstico para ilustrar la relación entre la variable de resultado y una sola predictora.

Sinónimo

diagrama de variables agregadas

Valores atípicos

En términos generales, un valor extremo, también llamado *valor atípico* (*outlier*), es aquel que está alejado de la mayoría de las otras observaciones. Así como los valores atípicos deben manejarse para las estimaciones de la localización y la variabilidad (consultar “Estimación de la localización” en la página 7 y “Estimación de la variabilidad” en la página 13), los valores atípicos pueden causar problemas en los modelos de regresión. En regresión, un valor atípico es un registro cuyo valor real de y está distante del valor pronosticado. Podemos detectar valores atípicos examinando el *residuo estandarizado* (*standardized residual*), que es el residuo dividido por el error estándar de los residuos.

No existe una teoría estadística que separe los valores atípicos de los típicos. Más bien, existen reglas prácticas (arbitrarias) sobre lo alejada que debe estar una observación del grueso de los datos para que se considere un valor atípico. Por ejemplo, con el diagrama de caja, los valores atípicos son aquellos puntos de datos que están demasiado por encima o por debajo de los límites de la caja (consultar “Percentiles y diagramas de caja” en la página 20), donde “demasiado lejos” = “más de 1.5 veces el rango intercuartil”. El residuo estandarizado es la métrica que se usa generalmente en la regresión para determinar si un registro se clasifica como valor atípico. Los residuos estandarizados se pueden interpretar como “el número de errores estándar fuera de la línea de regresión”.

Vamos a ajustar en *R* una regresión a los datos de ventas de las viviendas del condado de King para todas las ventas que figuran en el código postal 98105:

```
house_98105 <- house[house$ZipCode == 98105,]
lm_98105 <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
    Bedrooms + BldgGrade, data=house_98105)
```

En *Python*:

```
house_98105 = house.loc[house['ZipCode'] == 98105,]

predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade']
outcome = 'AdjSalePrice'

house_outlier = sm.OLS(house_98105[outcome],
                      house_98105[predictors].assign(const=1))
result_98105 = house_outlier.fit()
```

Extraemos los residuos estandarizados con *R* utilizando la función `rstandard` y obtenemos el índice del residuo más pequeño utilizando la función `order`:

```
sresid <- rstandard(lm_98105)
idx <- order(sresid)
sresid[idx[1]]
20429
-4.326732
```

Estadística práctica para ciencia de datos con R y Python

En statsmodels, utilizamos OLSInfluence para analizar los residuos:

```
influence = OLSInfluence(result_98105)
sresiduals = influence.resid_studentized_internal
sresiduals.idxmin(), sresiduals.min()
```

La mayor sobreestimación del modelo es de más de cuatro errores estándar por encima de la línea de regresión, lo que corresponde a una sobreestimación de 757 754 \$. En R el registro de datos original correspondiente a este valor atípico es el siguiente:

```
house_98105[idx[1], c('AdjSalePrice', 'SqFtTotLiving', 'SqFtLot',
'Bathrooms', 'Bedrooms', 'BldgGrade')]
```

AdjSalePrice (dbl)	SqFtTotLiving (int)	SqFtLot (int)	Bathrooms (dbl)	Bedrooms (int)	BldgGrade (int)
20429	19748	2900	7276	3	6

En Python:

```
outlier = house_98105.loc[sresiduals.idxmin(), :]
print('AdjSalePrice', outlier[outcome])
print(outlier[predictors])
```

En este caso, parece que algo no está bien en el registro: una vivienda de ese tamaño normalmente se vende por mucho más de 119 748 \$ en ese código postal. La figura 4.4 muestra un extracto de la escritura legal de esta venta: está claro que la venta consistió solo en una participación parcial en la propiedad. En este caso, el valor atípico corresponde a una venta que es anómala y no debe incluirse en la regresión. Los valores atípicos también podrían ser el resultado de otros problemas, como una introducción de datos errónea de compraventa o un desajuste de unidades (por ejemplo, informar una venta en miles de dólares en lugar de simplemente en dólares).

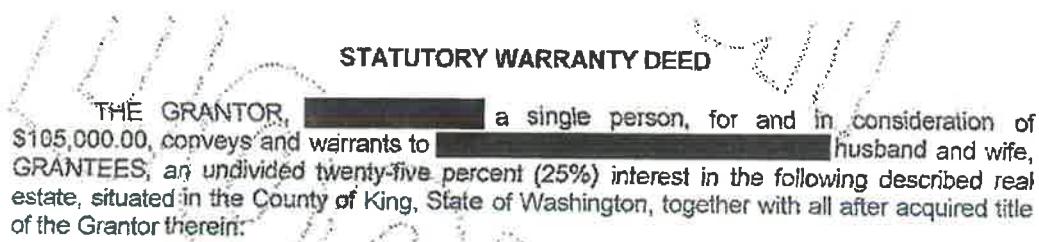


Figura 4.4 Escritura de garantía legal del mayor residuo negativo.

Para los problemas de big data, los valores atípicos generalmente no son un inconveniente a la hora de ajustar la regresión que se utilizará para pronosticar nuevos datos. Sin embargo, los valores atípicos son fundamentales para la detección de anomalías, labor en la que encontrar valores atípicos es la clave. El valor atípico también podría corresponder a un caso de fraude o una actuación accidental. En cualquier caso, la detección de valores atípicos puede ser una necesidad empresarial crítica.

Valores influyentes

A un valor cuya ausencia cambiaría de forma importante la ecuación de regresión se denomina observación *influyente* (*influential*). En regresión, no es necesario que dicho valor esté asociado con un residuo grande. Como ejemplo, consideremos las líneas de regresión de la figura 4.5. La línea continua corresponde a la regresión con todos los datos, mientras que la línea discontinua corresponde a la regresión obtenida al eliminar el punto de la esquina superior derecha. Evidentemente, ese valor de los datos tiene una gran influencia en la regresión, aunque no esté asociado con un valor atípico grande (el de la regresión completa). Se considera que este valor de los datos tiene un gran *apalancamiento* (*leverage*) sobre la regresión.

Además de los residuos estandarizados (consultar “Valores atípicos” en la página 177), los estadísticos han desarrollado varias métricas para determinar la influencia de un solo registro en la regresión. Una medida habitual del apalancamiento es el valor *con circunflejo* (*hat-value*). Valores por encima de $(2P + 1 / n)$ indican un valor de los datos de alto apalancamiento⁸.

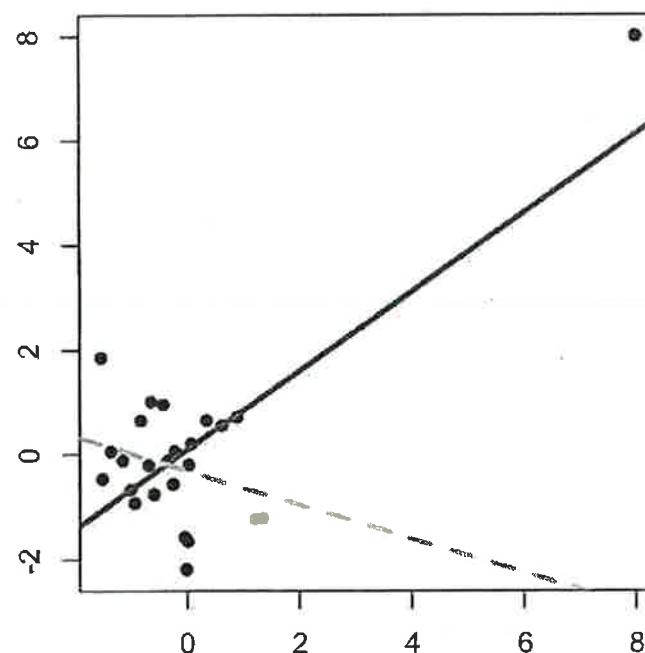


Figura 4.5 Ejemplo de un punto de datos que influye en la regresión.

Otra métrica es la distancia de Cook (*Cook's distance*), que define la influencia como una combinación del apalancamiento y del tamaño del residuo. Una regla general es que una observación tiene gran influencia si la distancia de Cook es superior a $4 / (n - P - 1)$.

⁸ El término *valor con circunflejo* (*hat-value*) proviene de la noción de matriz con circunflejo en regresión. La regresión lineal múltiple se puede expresar mediante la fórmula $\hat{Y} = HY$ donde H es la matriz con circunflejo. Los valores con circunflejo corresponden a la diagonal de H .

El *diagrama de influencia* (*influence plot*) o *diagrama de burbujas* (*bubble plot*) combina los residuos estandarizados, los valores con circunflejo y la distancia de Cook en un solo diagrama. La figura 4.6 muestra la gráfica de influencia para los datos de las viviendas del condado de King, que se puede crear mediante el siguiente código R:

```
std_resid <- rstandard(lm_98105)
cooks_D <- cooks.distance(lm_98105)
hat_values <- hatvalues(lm_98105)
plot(subset(hat_values, cooks_D > 0.08), subset(std_resid, cooks_D > 0.08),
     xlab='hat_values', ylab='std_resid',
     cex=10*sqrt(subset(cooks_D, cooks_D > 0.08)), pch=16, col='lightgrey')
points(hat_values, std_resid, cex=10*sqrt(cooks_D))
abline(h=c(-2.5, 2.5), lty=2)
```

A continuación se detalla el código de *Python* para crear una figura similar:

```
influence = OLSInfluence(result_98105)
fig, ax = plt.subplots(figsize=(5, 5))
ax.axhline(-2.5, linestyle='--', color='C1')
ax.axhline(2.5, linestyle='--', color='C1')
ax.scatter(influence.hat_matrix_diag, influence.resid_studentized_internal,
           s=1000 * np.sqrt(influence.cooks_distance[0]),
           alpha=0.5)
ax.set_xlabel('hat values')
ax.set_ylabel('studentized residuals')
```

Aparentemente, hay varios puntos de datos que exhiben una gran influencia en la regresión. La distancia de Cook se puede calcular usando la función `cooks.distance`, y podemos utilizar `hatvalues` para calcular los diagnósticos. Los valores con circunflejo se reflejan en el eje x, los residuos en el eje y; el tamaño de los puntos está relacionado con el valor de la distancia de Cook.

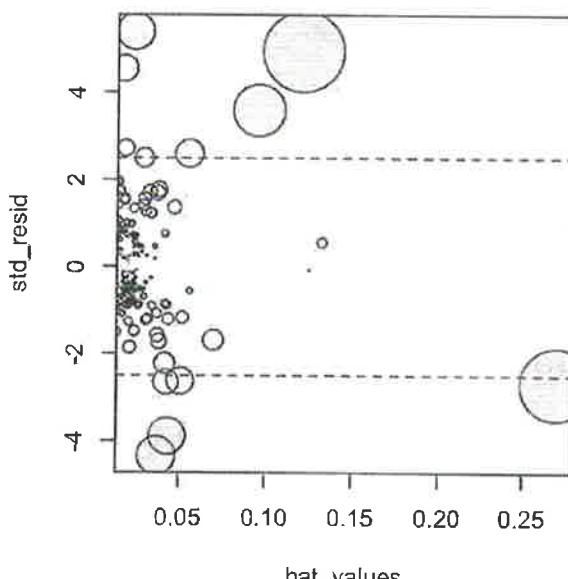


Figura 4.6 Diagrama para determinar las observaciones que tienen una gran influencia. Los puntos con una distancia de Cook superior a 0.08 están resaltados en gris.

La tabla 4.2 compara la regresión con el conjunto de datos completo y con los puntos de datos de gran influencia eliminados (distancia de Cook > 0.08). El coeficiente de regresión para Bathrooms cambia drásticamente⁹.

Tabla 4.2 Comparación de los coeficientes de regresión con todos los datos y con los datos influyentes eliminados

	Datos originales	Datos influyentes eliminados
(Intercept)	-772 550	-647 137
SqFtTotLiving	210	230
SqFtLot	39	33
Bathrooms	2 282	-16 132
Bedrooms	-26 320	-22 888
BldgGrade	130 000	114 871

Para el propósito de ajustar una regresión que pronostica de manera confiable datos futuros, la identificación de observaciones influyentes es útil solo para pequeños conjuntos de datos. Para las regresiones que involucran muchos registros, es poco probable que una sola observación tenga el peso suficiente para causar una influencia extrema en la ecuación ajustada (aunque la regresión aún puede tener valores atípicos grandes). Sin embargo, para fines de detección de anomalías, la identificación de observaciones influyentes puede ser muy útil.

Heterocedasticidad, anormalidad y errores correlacionados

Los estadísticos prestan mucha atención a la distribución de los residuos. Resulta que los mínimos cuadrados ordinarios (consultar “Mínimos cuadrados” en la página 148) no son sesgados y, en algunos casos, son el estimador “óptimo”, bajo una amplia gama de supuestos distributivos. Esto significa que, en la mayoría de los problemas, los científicos de datos no necesitan preocuparse demasiado por la distribución de los residuos.

La distribución de los residuos es relevante principalmente para la validez de la inferencia estadística formal (pruebas de hipótesis y valores p), pero es de mínima importancia para los científicos de datos interesados principalmente en la precisión predictiva. Los errores normalmente distribuidos son una señal de que el modelo está completo. Los errores que no se distribuyen normalmente indican que al modelo le puede faltar algo. Para que la inferencia formal sea completamente válida, se supone que los residuos están distribuidos normalmente, tienen la misma varianza y son independientes. Un área en la que la distribución de los residuos puede ser de interés para los científicos de datos es el cálculo estándar de los intervalos de confianza para los valores pronosticados, que se basan en las suposiciones sobre los residuos.

⁹ El coeficiente para Bathrooms se convierte en negativo, lo que no es intuitivo. No se ha tenido en cuenta la ubicación. El código postal 98105 contiene áreas con viviendas de distintos tipos. Consultar “Variables de confusión” en la página 172 para obtener más información sobre las variables de confusión.

Estadística práctica para ciencia de datos con R y Python

La heterocedasticidad (*heteroskedasticity*) es la falta de varianza residual constante en el rango de los valores pronosticados. En otras palabras, los errores son mayores en algunas partes del rango que en otras. La visualización de los datos es una forma conveniente de analizar los residuos.

El siguiente código de R representa los residuos absolutos en función de los valores pronosticados para el ajuste de la regresión lm_98105 en “Valores atípicos” en la página 177:

```
df <- data.frame(resid = residuals(lm_98105), pred = predict(lm_98105))
ggplot(df, aes(pred, abs(resid))) + geom_point() + geom_smooth()
```

La figura 4.7 muestra el diagrama resultante. Con `geom_smooth`, es fácil aplicar un suavizado a los residuos absolutos. La función llama al método loess (suavizado de diagramas de dispersión estimado localmente) para obtener una estimación suavizada de la relación entre las variables en el eje x y el eje y en el diagrama de dispersión (consultar “Suavizadores de diagramas de dispersión” en la página 185).

En *Python*, el paquete `seaborn` tiene la función `regplot` para crear una figura similar:

```
fig, ax = plt.subplots(figsize=(5, 5))
sns.regplot(result_98105.fittedvalues, np.abs(result_98105.resid),
            scatter_kws={'alpha': 0.25}, line_kws={'color': 'C1'},
            lowess=True, ax=ax)
ax.set_xlabel('predicted')
ax.set_ylabel('abs(residual)')
```

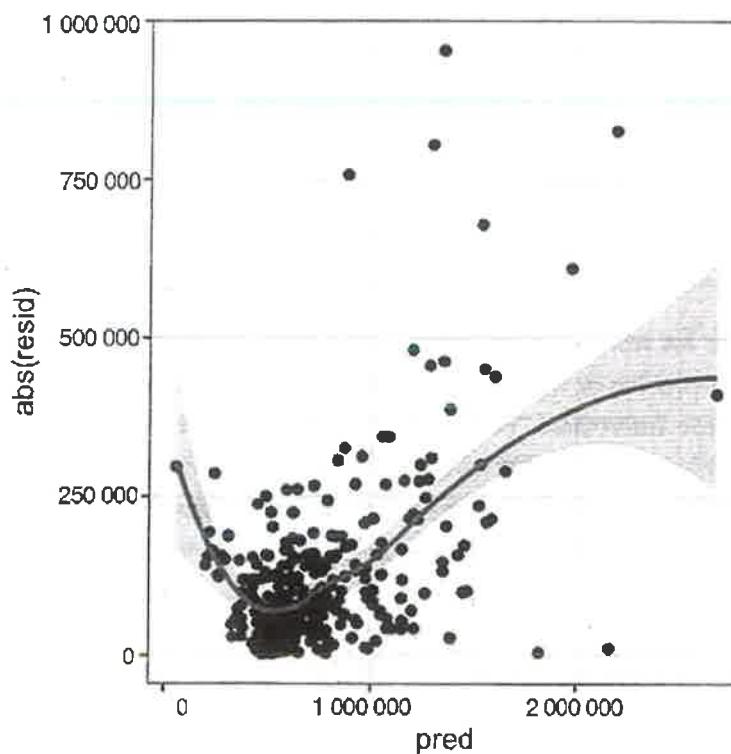


Figura 4.7 Diagrama del valor absoluto de los residuos en relación con los valores pronosticados.

Evidentemente, la varianza de los residuos tiende a aumentar para las viviendas de mayor valor, pero también es grande para las de menor valor. Este diagrama indica que `lm_98105` tiene errores heteroscedásticos.



¿Por qué a un científico de datos le importaría la heterocedasticidad?

La heterocedasticidad indica que los errores de pronóstico difieren para diferentes rangos del valor pronosticado y puede sugerir un modelo incompleto. Por ejemplo, la heterocedasticidad en `lm_98105` puede indicar que la regresión ha dejado algo sin explicar en las viviendas de rango alto y bajo.

La figura 4.8 es un histograma de los residuos estandarizados para la regresión `lm_98105`. La distribución tiene colas claramente más largas que la distribución normal y exhibe una leve asimetría hacia residuos más grandes.

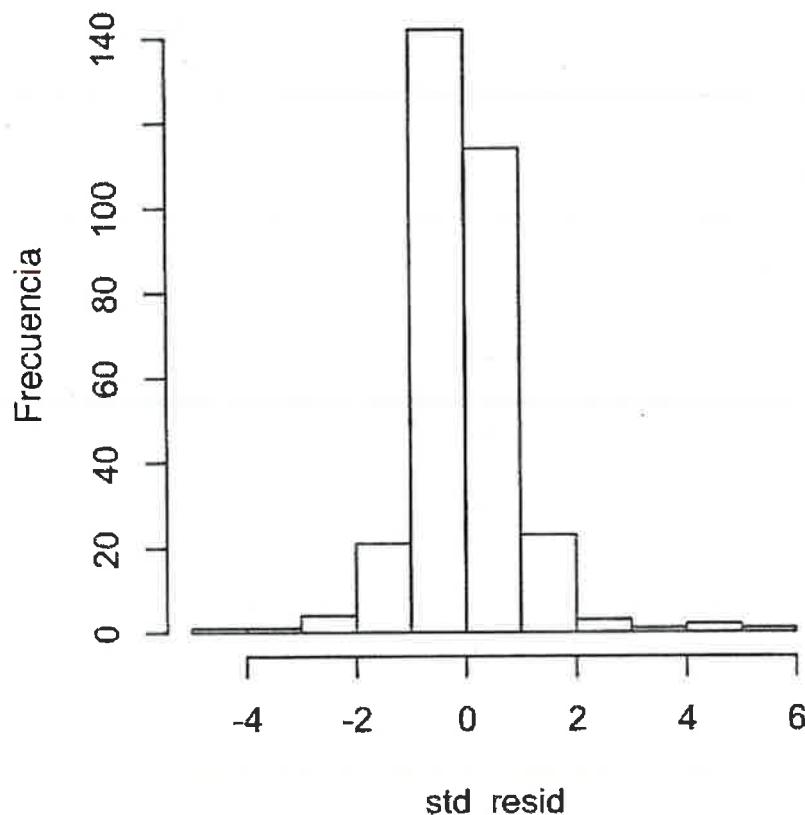


Figura 4.8 Histograma de los residuos de la regresión de los datos de la vivienda.

Los estadísticos también pueden verificar el supuesto de que los errores son independientes. Esto es particularmente cierto para los datos que se recopilan a lo largo del tiempo o el espacio. La estadística de Durbin-Watson se puede utilizar para detectar si existe una autocorrelación significativa en una regresión que involucra datos de series de tiempo. Si los errores de un modelo de regresión están correlacionados, esta

información puede ser útil para hacer pronósticos a corto plazo y debe incorporarse al modelo. Consultar *Practical Time Series Forecasting with R*, 2.^a ed., de Galit Shmueliy Kenneth Lichtendahl (Axelrod Schnall, 2018) para obtener más información sobre cómo elaborar información de autocorrelación en modelos de regresión para datos de series de tiempo. Si el objetivo son los pronósticos a más largo plazo o los modelos descriptivos, el exceso de datos autocorrelacionados a nivel micro puede distraernos. En ese caso, el suavizado o la recopilación de datos menos granular en primer lugar puede estar bien.

Aunque una regresión pudiera violar uno de los supuestos distributivos, ¿debería importarnos? La mayoría de las veces, en la ciencia de datos, el interés se centra principalmente en la precisión del pronóstico, por lo que puede ser necesario realizar una revisión de la heterocedasticidad. Podemos descubrir que hay alguna señal en los datos que nuestro modelo no ha capturado. Sin embargo, satisfacer los supuestos distributivos simplemente por el hecho de validar la inferencia estadística formal (valores p, estadísticas F, etc.) no es tan importante para el científico de datos.



Suavizadores de diagramas de dispersión

La regresión consiste en modelar la relación entre la respuesta y las variables predictoras. Al evaluar un modelo de regresión, es conveniente utilizar un *diagrama de dispersión más suave* (*scatterplot smoother*) para resaltar visualmente las relaciones entre dos variables.

Por ejemplo, en la figura 4.7, el suavizado muestra que la varianza de los residuos depende del valor del residuo. En este caso, se ha utilizado la función loess. loess funciona ajustando repetidamente una serie de regresiones locales a subconjuntos contiguos para conseguir el suavizado. Si bien loess es probablemente el suavizador más utilizado, en R están disponibles otros suavizadores de diagramas de dispersión, como super smooth (`supsmu`) y kernel smoothing (`ksmooth`). En Python, podemos encontrar suavizadores adicionales en `scipy` (`wiener` o `sav`) y `statsmodels` (`kernel_regression`). Si el fin es evaluar un modelo de regresión, normalmente no hay necesidad de preocuparse por los detalles de estos diagramas de dispersión.

Diagramas de residuos parciales y falta de linealidad

Los *diagramas de residuos parciales* (*partial residual plots*) son una forma de visualizar lo bien que el ajuste estimado explica la relación entre una predictora y el resultado. La idea básica de un diagrama de residuos parciales es aislar la relación entre una variable predictora y la respuesta, *teniendo en cuenta todas las demás variables predictoras* (*taking into account all of the other predictor variables*). Un residuo parcial podría considerarse como un valor de "resultado sintético", que combina el pronóstico basado en una única

predictora con el residuo real de la ecuación de regresión completa. Un residuo parcial para la predictora X_i es el residuo normal más el término de regresión asociado con X_i :

$$\text{Residuo parcial} = \text{Residuo} + \hat{b}_i X_i$$

donde \hat{b}_i es el coeficiente de regresión estimado. La función `predict` de R tiene una opción para obtener los términos de regresión individuales $\hat{b}_i X_i$:

```
terms <- predict(lm_98105, type='terms')
partial_resid <- resid(lm_98105) + terms
```

El diagrama de residuos parciales muestra la predictora X_i en el eje x y los residuos parciales en el eje y. El uso de `ggplot2` facilita la superposición del suavizado de los residuos parciales:

```
df <- data.frame(SqFtTotLiving = house_98105[, 'SqFtTotLiving'],
                  Terms = terms[, 'SqFtTotLiving'],
                  PartialResid = partial_resid[, 'SqFtTotLiving'])
ggplot(df, aes(SqFtTotLiving, PartialResid)) +
  geom_point(shape=1) + scale_shape(solid = FALSE) +
  geom_smooth(linetype=2) +
  geom_line(aes(SqFtTotLiving, Terms))
```

El paquete `statsmodels` tiene el método `sm.graphics.plot_ccpr`, que crea un diagrama de residuos parciales similar:

```
sm.graphics.plot_ccpr(result_98105, 'SqFtTotLiving')
```

Los gráficos R y Python se diferencian en un desplazamiento constante. En R, se suma una constante para que la media de los términos sea cero.

El diagrama resultante se muestra en la figura 4.9. El residuo parcial es una estimación de la contribución que `SqFtTotLiving` agrega al precio de venta. La relación entre `SqFtTotLiving` y el precio de venta es evidentemente no lineal (línea discontinua). La línea de regresión (línea continua) subestima el precio de venta de las viviendas de menos de 1000 pies cuadrados y sobreestima el precio de las viviendas entre 2000 y 3000 pies cuadrados. Hay muy pocos puntos de datos por encima de los 4000 pies cuadrados para poder sacar conclusiones en relación con esas viviendas.

Esta falta de linealidad tiene sentido en este caso: agregar 500 pies a una vivienda pequeña hace que exista una diferencia mucho mayor que agregar 500 pies a una vivienda grande. Esto sugiere que, en lugar de un término lineal simple para `SqFtTotLiving`, se debe considerar un término no lineal (consultar “Regresión polinomial y por spline” en la página 187).

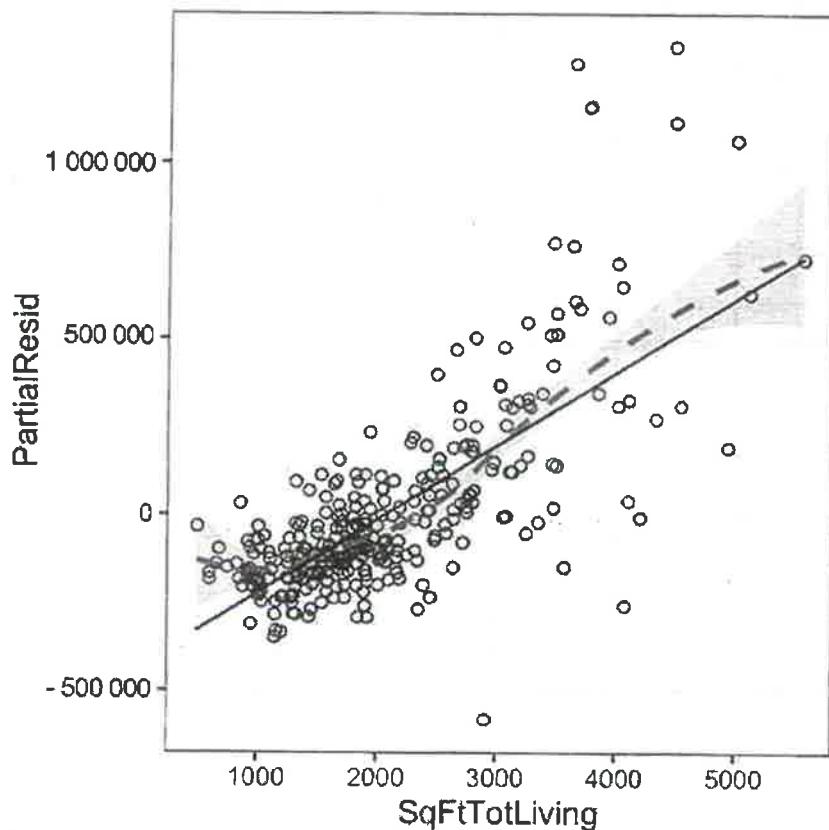


Figura 4.9 Diagrama de residuos parciales para la variable `SqFtTotLiving`.

Ideas clave

- Si bien los valores atípicos pueden causar problemas para conjuntos de datos pequeños, el interés principal de los valores atípicos es identificar problemas con los datos o localizar anomalías.
- Los registros individuales (incluidos los valores atípicos de la regresión) pueden tener una gran influencia en una ecuación de regresión con cantidades pequeñas de datos, pero este efecto se desvanece con cantidades grandes de datos.
- Si el modelo de regresión se usa para inferencias formales (valores p y similares), entonces deben verificarse ciertos supuestos sobre la distribución de los residuos. Sin embargo, en general, la distribución de los residuos no es crítica en la ciencia de datos.
- La gráfica de residuos parciales se puede utilizar para evaluar cualitativamente el ajuste para cada término de la regresión, lo que posiblemente conduzca a una especificación alternativa del modelo.

Regresión polinomial y por spline

La relación entre la respuesta y una variable predictora no es necesariamente lineal. La respuesta a la dosis de un medicamento a menudo no es lineal: duplicar la dosis generalmente no conduce a que se duplique la respuesta. La demanda de un producto no es una función lineal del dinero destinado al marketing, ya que en algún momento es probable que la demanda se sature. Hay muchas formas de ampliar la regresión para capturar estos efectos no lineales.

Términos clave de la regresión no lineal

Regresión polinomial

Agrega términos polinomiales (cuadrados, cubos, etc.) a la regresión.

Regresión por spline

Ajuste suave de una curva con una serie de segmentos polinomiales.

Nudos

Valores que separan segmentos de una spline.

Modelos aditivos generalizados

Modelos de splines con selección automática de nudos.

Sinónimo

GAM



Regresión no lineal

Cuando los estadísticos hablan de *regresión no lineal* (*nonlinear regression*), se refieren a modelos que no se pueden ajustar mediante mínimos cuadrados. ¿Qué tipo de modelos son no lineales? Esencialmente todos los modelos en los que la respuesta no se puede expresar como una combinación lineal de las predictoras o alguna transformación de las mismas. El ajuste de los modelos de regresión no lineal es más difícil y demanda recursos de cálculo intensivos, ya que requieren optimización numérica. Por esta razón, generalmente se prefiere utilizar un modelo lineal si es posible.

Polinomial

La regresión polinomial (*polynomial regression*) implica incluir términos polinomiales en la ecuación de regresión. El uso de la regresión polinomial se remonta casi al desarrollo de la regresión en sí con un artículo de Gergonne en 1815. Por ejemplo, una regresión cuadrática entre la respuesta Y y la predictora X adoptaría la forma:

Estadística práctica para ciencia de datos con R y Python

$$Y = b_0 + b_1X + b_2X^2 + e$$

La regresión polinomial se puede ajustar en R a través de la función `poly`. Por ejemplo, lo siguiente ajusta un polinomio cuadrático para `SqFtTotLiving` con los datos de las viviendas del condado de King:

```
lm(AdjSalePrice ~ poly(SqFtTotLiving, 2) + SqFtLot +
  BldgGrade + Bathrooms + Bedrooms,
  data=house_98105)
```

Call:

```
lm(formula = AdjSalePrice ~ poly(SqFtTotLiving, 2) + SqFtLot +
  BldgGrade + Bathrooms + Bedrooms, data = house_98105)
```

Coefficients:

	(Intercept)	poly(SqFtTotLiving, 2)1	poly(SqFtTotLiving, 2)2
	-402530.47	3271519.49	776934.02
SqFtLot		BldgGrade	Bathrooms
	32.56	135717.06	-1435.12
Bedrooms			
	-9191.94		

En `statsmodels` agregamos el término al cuadrado a la definición del modelo usando `I(SqFtTotLiving**2)`:

```
model_poly = smf.ols(formula='AdjSalePrice ~ SqFtTotLiving + ' +
  '+ I(SqFtTotLiving**2) + ' +
  'SqFtLot + Bathrooms + Bedrooms + BldgGrade', data=house_98105)
result_poly = model_poly.fit()
result_poly.summary()
```

La intersección y los coeficientes del polinomio son diferentes en comparación

- ❶ con R. Esto se debe a diferentes implementaciones. Los coeficientes restantes y los pronósticos son equivalentes.

Ahora hay dos coeficientes asociados con `SqFtTotLiving`: uno para el término lineal y otro para el término cuadrático.

El diagrama de residuos parciales (consultar “Diagramas de residuos parciales y falta de linealidad” en la página 185) indica cierta curvatura en la ecuación de regresión asociada con `SqFtTotLiving`. La línea ajustada se asemeja más a la suavidad de los residuos parciales en comparación con el ajuste lineal (consultar la figura 4.10).

La implementación de `statsmodels` funciona solo para términos lineales. El código fuente adjunto proporciona una implementación que también funcionará para la regresión polinomial.

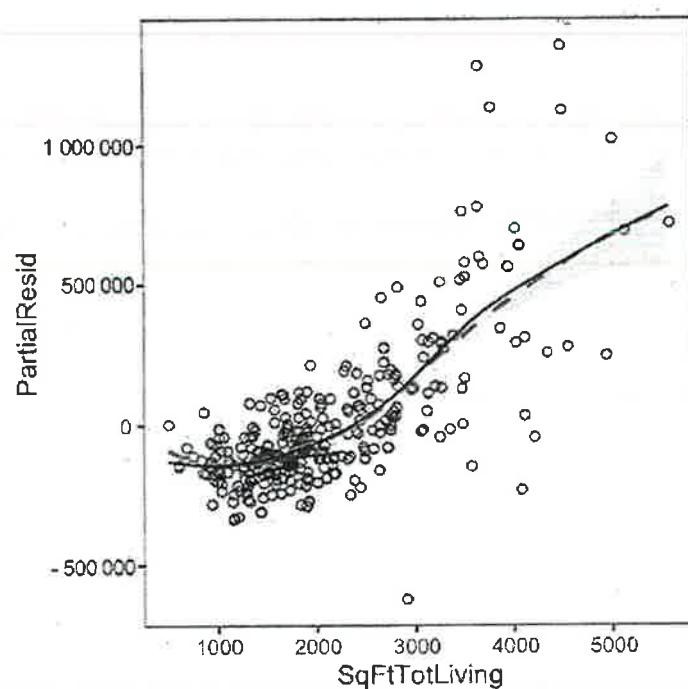


Figura 4.10 Ajuste de regresión polinomial para la variable *SqFtTotLiving* (línea continua) frente a un ajuste suave (línea discontinua, consultar la siguiente sección sobre *splines*).

Splines

En una relación no lineal, la regresión polinomial captura solo una cierta parte de la curvatura. Añadir términos de orden superior, como un polinomio de tercer o cuarto grado conduce a menudo a una "ondulación" indeseable en la ecuación de regresión. Un enfoque alternativo, a menudo mejor que el anterior, para modelar relaciones no lineales es utilizar *splines*. Las splines proporcionan una forma de interpolación suave entre puntos fijos. Los dibujantes usaban originalmente splines para trazar curvas suaves, particularmente en la construcción de barcos y aviones.

Las splines se creaban flexionando una delgada pieza de madera mediante el uso de pesos, a los que se conoce como "patos". Ver la figura 4.11.



Figura 4.11 Las splines se crearon originalmente con madera flexible y "patos". Los dibujantes las utilizaron como herramienta para reproducir curvas (foto cortesía de Bob Perry).

La definición técnica de una spline es la de una serie de polinomios continuos por partes. Los desarrolló el matemático rumano I. J. Schoenberg durante la Segunda Guerra Mundial en los US Aberdeen Proving Grounds. Las partes del polinomio se conectan prolongándose en una serie de puntos fijos de una variable predictora, denominados *nudos (knots)*. La formulación de splines es mucho más complicada que la de la regresión polinomial. El software estadístico generalmente controla los detalles del ajuste de una spline. El paquete splines de *R* incluye la función *bs* para crear el término *b-spline* (*b-spline*) en el modelo de regresión. Por ejemplo, lo siguiente agrega un término *b-spline* al modelo de regresión de las viviendas:

```
library(splines)
knots <- quantile(house_98105$SqFtTotLiving, p=c(.25, .5, .75))
lm_spline <- lm(AdjSalePrice ~ bs(SqFtTotLiving, knots=knots, degree=3) +
  SqFtLot + Bathrooms + Bedrooms + BldgGrade, data=house_98105)
```

Es necesario especificar dos parámetros: el grado del polinomio y la ubicación de los nudos. En este caso, se incluye en el modelo la predictora *SqFtTotLiving* usando una spline de tercer orden (*degree=3*). Por defecto, *bs* coloca nudos en los límites. Además, los nudos también se han colocado en el cuartil inferior, el cuartil mediano y el cuartil superior.

La interfaz de fórmula *statsmodels* admite el uso de splines de manera similar a *R*. Aquí, especificamos *b-spline* usando *df*, los grados de libertad. Esto creará *df* – *degree* = 6 – 3 = 3 nudos internos con posiciones calculadas de la misma manera que en el código *R* anterior:

```
formula = 'AdjSalePrice ~ bs(SqFtTotLiving, df=6, degree=3) + '+SqFtLot +
  Bathrooms + Bedrooms + BldgGrade'
model_spline = smf.ols(formula=formula, data=house_98105)
result_spline = model_spline.fit()
```

A diferencia del término lineal, para el cual el coeficiente tiene un significado claro, los coeficientes de un término spline no son interpretables. En cambio, es más conveniente usar una representación visual para revelar la naturaleza del ajuste de spline. La figura 4.12 muestra el diagrama de residuos parciales de la regresión. En contraste con el modelo polinomial, el modelo de splines se ajusta más a transiciones suaves, lo que demuestra la mayor flexibilidad de las splines. En este caso, la línea se ajusta más a los datos. ¿Significa esto que la regresión spline es un modelo mejor? No necesariamente: no tiene sentido en términos económicos que las viviendas muy pequeñas (menos de 1000 pies cuadrados) tengan un valor más alto que las viviendas un poco más grandes. Esto es posiblemente el artificio de una variable de confusión. Consultar “Variables de confusión” en la página 172.

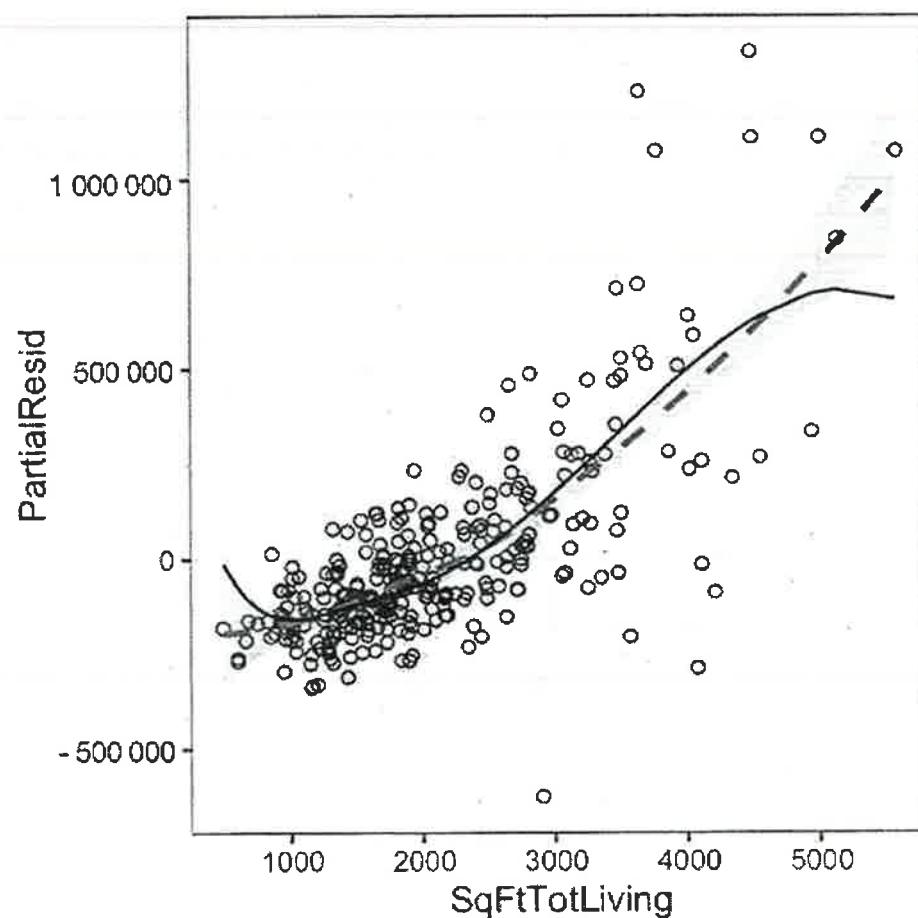


Figura 4.12 Ajuste de la regresión spline para la variable *SqFtTotLiving* (línea continua) en comparación con la línea suave (línea discontinua).

Modelos aditivos generalizados

Supongamos que tenemos la sospecha de que existe una relación no lineal entre la respuesta y una variable predictoría, ya sea por conocimiento *a priori* o examinando los diagnósticos de regresión. Los términos polinomiales pueden no ser lo suficientemente flexibles para reflejar la relación, y los términos spline requieren especificar los nodos. Los *modelos aditivos generalizados* (*generalized additive models*), o GAM, son una técnica de modelado flexible que se puede utilizar para ajustar automáticamente una regresión spline. Se puede emplear el paquete mgcv en R para ajustar un modelo GAM a los datos de las viviendas:

```
library(mgcv)
lm_gam <- gam(AdjSalePrice ~ s(SqFtTotLiving) + SqFtLot +
  Bathrooms + Bedrooms + BldgGrade,
  data=house_98105)
```

El término `s(SqFtTotLiving)` le dice a la función `gam` que encuentre los “mejores” nodos para un término spline (ver figura 4.13).

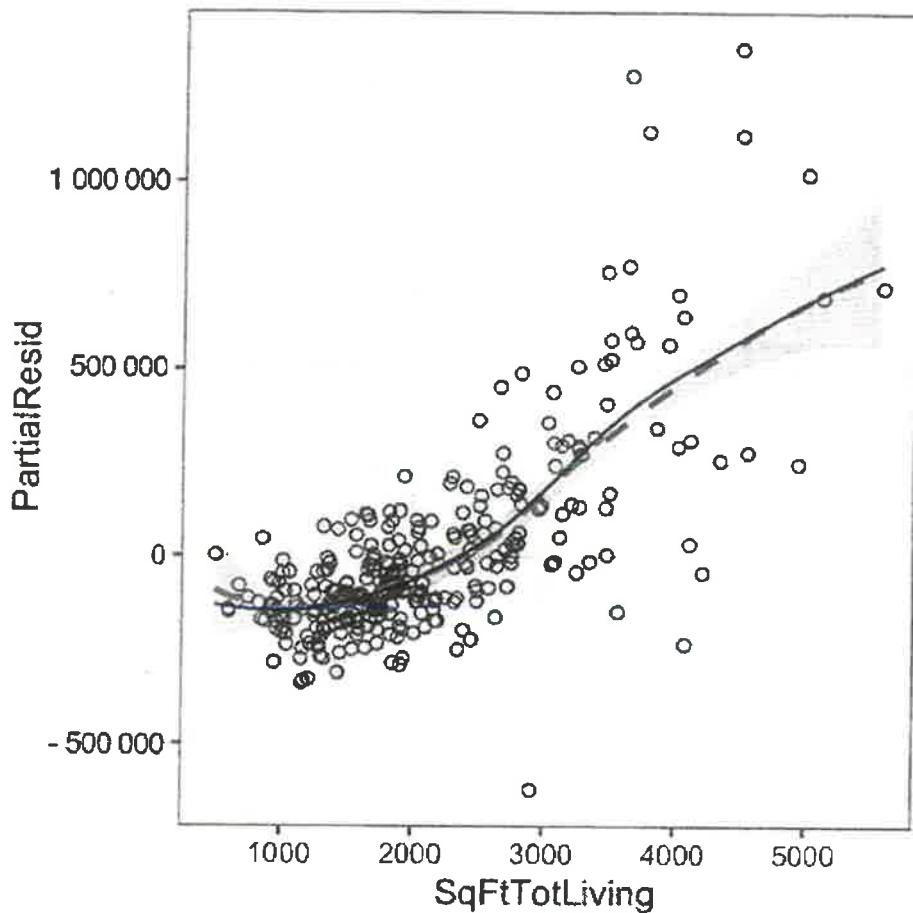


Figura 4.13 Ajuste de regresión GAM para la variable *SqFtTotLiving* (línea continua) en comparación con la línea (línea discontinua).

En *Python*, podemos usar el paquete *pyGAM*. Proporciona métodos de regresión y clasificación. A continuación, usamos *LinearGAM* para crear el modelo de regresión:

```

predictors = ['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade']
outcome = 'AdjSalePrice'
X = house_98105[predictors].values
y = house_98105[outcome]

gam = LinearGAM(s(0, n_splines=12) + l(1) + l(2) + l(3) + l(4)) ❶
gam.gridsearch(X, y)

```

- ❶ El valor por defecto para *n_splines* es 20. Esto conduce a un sobreajuste para valores *SqFtTotLiving* más grandes. Un valor de 12 permite un ajuste más razonable.

Ideas clave

- Los valores atípicos en una regresión son registros con un residuo grande.
- La multicolinealidad puede causar inestabilidad numérica al ajustar la ecuación de regresión.
- Una variable de confusión es una predictora importante que se omite en un modelo y puede llevar a que la ecuación de regresión tenga relaciones espurias.
- Se necesita un término de interacción entre dos variables si el efecto de una variable depende del nivel o la magnitud de la otra.
- La regresión polinomial puede ajustar relaciones no lineales entre las variables predictoras y la variable de resultado.
- Las splines son series de segmentos polinomiales encadenados, que se unen en nudos.
- Podemos automatizar el proceso de especificación de los nudos en splines utilizando modelos aditivos generalizados (GAM).

Lecturas complementarias

- Para disponer de una información más amplia sobre modelos spline y GAM, consultar *The Elements of Statistical Learning*, 2.^a ed., de Trevor Hastie, Robert Tibshirani y Jerome Friedman (2009), y su primo menor basado en R, *An Introduction to Statistical Learning* de Gareth James, Daniela Witten, Trevor Hastie y Robert Tibshirani (2013). Ambos son libros de Springer.
- Para disponer de una información más amplia sobre el uso de modelos de regresión para el pronóstico de series de tiempo, consultar *Practical Time Series Forecasting with R* de Galit Shmueli y Kenneth Lichtendahl (Axelrod Schnall, 2018).

Resumen

Quizá ningún otro método estadístico se ha utilizado más a lo largo de los años que la regresión, el proceso de establecer una relación entre múltiples variables predictoras y una variable de resultado. La forma fundamental es lineal: cada variable predictora tiene un coeficiente que describe una relación lineal entre ella y el resultado. Las formas más avanzadas de regresión, como la regresión polinomial y spline, permiten que la relación no sea lineal. En la estadística clásica, lo importante es encontrar un buen ajuste a los datos observados para explicar o describir algún fenómeno, y la fortaleza de este ajuste es la forma en que se utilizan las métricas tradicionales *dentro de la muestra (in-sample)* para evaluar el modelo. En la ciencia de datos, por el contrario, el objetivo suele ser

Estadística práctica para ciencia de datos con R y Python

pronosticar valores para nuevos datos, por lo que se utilizan métricas basadas en la precisión del pronóstico para datos fuera de la muestra. Los métodos de selección variable se utilizan para reducir la dimensionalidad y crear modelos más compactos.

CAPÍTULO 5

Clasificación

Los científicos de datos tienen a menudo la tarea de automatizar decisiones para resolver problemas comerciales. ¿Es un correo electrónico un intento de phishing? ¿Es probable que perdamos un cliente? ¿Es probable que el usuario de la web haga clic en un anuncio? Todos estos son problemas de *clasificación (classification)*, una forma de *aprendizaje supervisado (supervised learning)* en el que primero entrenamos un modelo con datos donde se conoce el resultado y luego aplicamos el modelo a los datos donde se desconoce el resultado. La clasificación es quizás la forma más importante de pronóstico: el objetivo es pronosticar si un registro es un 1 o un 0 (phishing/no phishing, hacer clic/no hacer clic, perder/no perder) o, en algunos casos, una de varias categorías (por ejemplo, el filtrado de Gmail de su bandeja de entrada en "principal", "social", "promocional" o "foros").

A menudo, necesitamos algo más que una simple clasificación binaria: queremos conocer la probabilidad del pronóstico de que un caso pertenezca a una clase. En lugar de que un modelo simplemente asigne una clasificación binaria, la mayoría de los algoritmos pueden obtener una puntuación de la probabilidad (propensión) de pertenecer a la clase que nos interesa. De hecho, con la regresión logística, el resultado por defecto de R está en la escala logarítmica de probabilidades, y esto debe transformarse en propensión. En `scikit-learn` de *Python*, la regresión logística, como la mayoría de los métodos de clasificación, proporciona dos métodos de pronóstico: `predict` (que obtiene la clase) y `predict_proba` (que obtiene las probabilidades para cada clase). A continuación, se puede usar un límite deslizante para convertir la puntuación de propensión en una decisión. El enfoque general es el siguiente:

1. Establecemos una probabilidad de corte para la clase que nos interesa, por encima de la cual consideramos que un registro pertenece a esa clase.
2. Hacemos una estimación (con cualquier modelo) de la probabilidad de que un registro pertenezca a la clase que nos interesa.
3. Si esa probabilidad está por encima de la probabilidad de corte, asignamos el nuevo registro a la clase que nos interesa.

Cuanto más alto sea el límite, habrá menos registros pronosticados como 1, es decir, pertenecientes a la clase que nos interesa. Cuanto más bajo sea el límite, existirán más registros pronosticados como 1.

En este capítulo se tratan varias técnicas clave para clasificar y estimar propensiones. En el capítulo siguiente se describen métodos adicionales que se pueden utilizar tanto para la clasificación como para el pronóstico numérico.

¿Más de dos categorías?

La gran mayoría de los problemas implican una respuesta binaria. Sin embargo, algunos problemas de clasificación requieren una respuesta con más de dos resultados posibles. Por ejemplo, en el aniversario del contrato de suscripción de un cliente, puede haber tres resultados: el cliente se va o nos "abandona" ($Y = 2$), tiene un contrato mensual renovable ($Y = 1$), o firma un nuevo contrato a largo plazo ($Y = 0$). El objetivo es pronosticar $Y = j$ para $j = 0, 1$ o 2 . La mayoría de los métodos de clasificación de este capítulo se pueden aplicar, ya sea directamente o con adaptaciones modestas, a respuestas que tienen más de dos resultados. Incluso en el caso de más de dos resultados, el problema a menudo se puede reformular en una serie de problemas binarios utilizando probabilidades condicionales. Por ejemplo, para pronosticar el resultado del contrato, podemos resolver dos problemas de pronóstico binario:

- Pronosticar si $Y = 0$ o $Y > 0$.
- Dado que $Y > 0$, pronosticar si $Y = 1$ o $Y = 2$.

En este caso, tiene sentido dividir el problema en dos casos: (1) si el cliente abandona, y (2) si no abandona, qué tipo de contrato elegirá. Desde el punto de vista del ajuste del modelo, a menudo es ventajoso convertir el problema multiclas en una serie de problemas binarios. Esto es particularmente cierto cuando una categoría es mucho más corriente que las otras.

Bayes ingenuo

El algoritmo Bayes ingenuo utiliza la probabilidad de observar valores predictores, dado un resultado, para estimar lo que realmente interesa: la probabilidad de observar el resultado $Y = i$, dado un conjunto de valores predictores¹⁰.

¹⁰ Esta sección y las siguientes de este capítulo © 2020 Datastats, LLC, Peter Bruce, Andrew Bruce y Peter Gedeck; utilizado con el correspondiente permiso.

Términos clave de Bayes ingenuo

Probabilidad condicional

Probabilidad de observar algún evento (digamos, $X = i$) dado algún otro evento (digamos, $Y = j$). Se escribe $P(X_i | Y_j)$.

Probabilidad a posteriori

Probabilidad de un resultado después de que se haya incorporado la información de la predictora (en contraste con la probabilidad previa de los resultados, sin tener en cuenta la información de la predictora).

Para comprender la clasificación de Bayes ingenuo, podemos comenzar imaginando una clasificación bayesiana completa o exacta. Para cada registro a clasificar:

1. Buscamos los demás registros con el mismo perfil de predictora (es decir, donde los valores de las predictoras son los mismos).
2. Determinamos a qué clases pertenecen esos registros y cuál es la más prevalente (es decir, probable).
3. Asignamos esa clase al nuevo registro.

El enfoque anterior equivale a encontrar todos los registros de la muestra que son exactamente iguales al nuevo registro que se va a clasificar, en el sentido de que todos los valores de la predictora son idénticos.



Las variables predictoras deben ser variables categóricas (factores) en el algoritmo estándar de Bayes ingenuo. Consultar “Variables predictoras numéricas” en la página 200 para ver dos soluciones para el uso de variables continuas.

Por qué la clasificación bayesiana exacta no es práctica

Cuando el número de variables predictoras excede a unas cuantas, muchos de los registros que se clasificarán no tendrán coincidencias exactas. Consideremos un modelo para pronosticar la votación sobre la base de variables demográficas. Incluso una muestra considerable puede que no contenga ni una sola coincidencia para un nuevo registro que sea un hombre hispano con altos ingresos, del Medio Oeste de EE. UU., que votó en las últimas elecciones, que no votó en las elecciones anteriores, tiene tres hijas y un hijo y está divorciado. Y esto es con solo ocho variables, un número pequeño para la mayoría de los problemas de clasificación. La adición de una sola variable nueva con cinco categorías igualmente frecuentes reduce la probabilidad de una coincidencia en un factor de 5.

La solución ingenua

En la solución de Bayes ingenuo, ya no restringimos el cálculo de probabilidad a aquellos registros que coinciden con el registro a clasificar. En su lugar, usamos el conjunto de datos completo. La modificación de Bayes ingenuo es la siguiente:

1. Para una respuesta binaria $Y = i$ ($i = 0$ o 1), estimamos las probabilidades condicionales individuales para cada predictora $P(X_j | Y = i)$. Estas son las probabilidades de que el valor de la predictora esté en el registro cuando observamos $Y = i$. Esta probabilidad se estima mediante la proporción de valores de X_j entre los registros de $Y = i$ en el conjunto de entrenamiento.
2. Multiplicamos estas probabilidades entre sí y luego por la proporción de registros pertenecientes a $Y = i$.
3. Repetimos los pasos 1 y 2 para todas las clases.
4. Hacemos una estimación de la probabilidad del resultado i tomando el valor calculado en el paso 2 para la clase i y dividiéndolo por la suma de dichos valores para todas las clases.
5. Asignamos el registro a la clase con la probabilidad más alta para este conjunto de valores de las predictoras.

Este algoritmo de Bayes ingenuo también se puede especificar como una ecuación para la probabilidad de observar el resultado $Y = i$, dado un conjunto de valores predictores X_1, \dots, X_p :

$$P(Y = i | X_1, X_2, \dots, X_p)$$

A continuación se presenta la fórmula completa para calcular las probabilidades de clase usando la clasificación exacta de Bayes:

$$P(Y = i | X_1, X_2, \dots, X_p) = \frac{P(Y = i) P(X_1, \dots, X_p | Y = i)}{P(Y = 0) P(X_1, \dots, X_p | Y = 0) + P(Y = 1) P(X_1, \dots, X_p | Y = 1)}$$

Teniendo en cuenta la suposición de independencia condicional de Bayes ingenuo, esta ecuación cambia a:

$$\begin{aligned} & P(Y = i | X_1, X_2, \dots, X_p) \\ &= \frac{P(Y = i) P(X_1 | Y = i) \dots P(X_p | Y = i)}{P(Y = 0) P(X_1 | Y = 0) \dots P(X_p | Y = 0) + P(Y = 1) P(X_1 | Y = 1) \dots P(X_p | Y = 1)} \end{aligned}$$

¿Por qué esta fórmula se llama "ingenua"? Hemos hecho una suposición simplificadora de que la *probabilidad condicional exacta* (*exact conditional probability*) de un vector

de valores de las predictoras, dada la observación de un resultado, está suficientemente bien estimada por el producto de las probabilidades condicionales individuales $P(X_j | Y = i)$. En otras palabras, al estimar $P(X_j | Y = i)$ en lugar de $P(X_1, X_2, \dots, X_p | Y = i)$, asumimos que X_j es independiente de todas las demás variables predictoras X_k para $k \neq j$.

Se pueden usar varios paquetes de R para estimar un modelo de Bayes ingenuo. El siguiente código ajusta un modelo a los datos de pago del préstamo utilizando el paquete klaR:

```
library(klaR)
naive_model <- NaiveBayes(outcome ~ purpose_ + home_ + emp_len_,
                           data = na.omit(loan_data))
naive_model$table
$purpose_
  var
grouping credit_card debt_consolidation home_improvement major_purchase
paid off 0.18759649 0.55215915 0.07150104 0.05359270
default 0.15151515 0.57571347 0.05981209 0.03727229
  var
grouping medical other small_business
paid off 0.01424728 0.09990737 0.02099599
default 0.01433549 0.11561025 0.04574126

$home_
  var
grouping MORTGAGE OWN RENT
paid off 0.4894800 0.0808963 0.4296237
default 0.4313440 0.0832782 0.4853778

$emp_len_
  var
grouping < 1 Year > 1 Year
paid off 0.03105289 0.96894711
default 0.04728508 0.95271492
```

El resultado del modelo son las probabilidades condicionales $P(X_j | Y = i)$.

En Python podemos usar sklearn.naive_bayes.MultinomialNB de scikit-learn. Necesitamos convertir las características categóricas en variables ficticias antes de ajustar el modelo:

```
predictors = ['purpose_', 'home_', 'emp_len_']
outcome = 'outcome'
X = pd.get_dummies(loan_data[predictors], prefix="", prefix_sep="")
y = loan_data[outcome]

naive_model = MultinomialNB(alpha=0.01, fit_prior=True)
naive_model.fit(X, y)
```

Es posible obtener las probabilidades condicionales del modelo ajustado usando la propiedad feature_log_prob_.

Estadística práctica para ciencia de datos con R y Python

El modelo se puede utilizar para pronosticar el resultado de un nuevo préstamo. Utilizamos el último valor del conjunto de datos para la prueba:

```
new_loan <- loan_data[147, c('purpose_', 'home_', 'emp_len_')]
row.names(new_loan) <- NULL
new_loan
  purpose_   home_ emp_len_
  1 small_business MORTGAGE > 1 Year
```

En *Python* obtenemos este valor como sigue:

```
new_loan = X.loc[146:146, :]
```

En este caso el modelo pronostica que habrá incumplimiento de pago (*R*):

```
predict(naive_model, new_loan)
$class
[1] default
Levels: paid off default

$posterior
      paid off      default
[1,] 0.3463013  0.6536987
```

Como hemos comentado, los modelos de clasificación de `scikit-learn` tienen dos métodos: `predict`, que obtiene la clase pronosticada, y `predict_proba`, que obtiene las probabilidades de la clase:

```
print('predicted class: ', naive_model.predict(new_loan)[0])

probabilities = pd.DataFrame(naive_model.predict_proba(new_loan),
                               columns=loan_data[outcome].cat.categories)
print('predicted probabilities', probabilities)

predicted class: default
predicted probabilities
      default      paid off
0    0.653696    0.346304
```

El pronóstico también arroja la estimación posterior de la probabilidad de incumplimiento. Se sabe que el clasificador bayesiano ingenuo produce estimaciones *sesgadas* (*biased*). Sin embargo, cuando el objetivo es *clasificar (rank)* los registros de acuerdo con la probabilidad de que $Y = 1$, no se necesitan estimaciones de probabilidad no sesgadas y Bayes ingenuo produce buenos resultados.

Variables predictoras numéricas

El clasificador bayesiano solo funciona con predictoras categóricas (por ejemplo, con clasificación de spam, donde la presencia o ausencia de palabras, frases, caracteres, etc., se encuentra en el centro de la tarea predictiva). Para aplicar Bayes ingenuo a los valores predictores numéricos, se debe adoptar uno de los dos enfoques:

- Agrupamos las predictoras numéricas, las convertimos en predictoras categóricas y aplicamos el algoritmo de la sección anterior.
- Utilizamos un modelo de probabilidad, por ejemplo, la distribución normal (consultar “Distribución normal” en la página 69) para estimar la probabilidad condicional $P(X_j | Y = i)$.



Cuando la categoría de una predictora está ausente en los datos de entrenamiento, el algoritmo asigna *probabilidad cero (zero probability)* a la variable de resultado en los datos nuevos, en lugar de simplemente ignorar esta variable y usar la información de otras variables, como lo harían otros métodos. Para evitar que ocurra esto, la mayoría de las implementaciones de Bayes ingenuo utilizan un parámetro de suavizado (suavizado de Laplace).

Ideas clave

- Bayes ingenuo trabaja con predictoras y resultados categóricos (factores).
- Se pregunta: “Dentro de cada categoría del resultado, ¿qué categorías de predictoras son las más probables?”.
- A continuación, esa información se invierte para estimar las probabilidades de las categorías de resultados, dados los valores de las predictoras.

Lecturas complementarias

- *The Elements of Statistical Learning*, 2.^a ed., de Trevor Hastie, Robert Tibshirani y Jerome Friedman (Springer, 2009).
- Hay un capítulo dedicado exclusivamente a Bayes ingenuo en *Data Mining for Business Analytics* by Galit Shmueli, Peter Bruce, Nitin Patel, Peter Gedeck, Inbal Yahav y Kenneth Lichtendahl (Wiley, 2007-2020, con ediciones para R, Python, Excel y JMP).

Análisis discriminante

El análisis discriminante (*discriminant analysis*) es el clasificador estadístico más antiguo. Lo introdujo R. A. Fisher en 1936 en un artículo publicado en la revista *Annals of Eugenics*¹¹.

¹¹ Sin duda, es sorprendente que el primer artículo sobre clasificación estadística se haya publicado en una revista dedicada a la eugenesia. De hecho, existe una conexión desconcertante entre el desarrollo temprano de la estadística y la eugenesia (<https://www.statistics.com/history-eugenics-journey-to-the-dark-side-at-the-dawn-of-statistics/>).

Términos clave de análisis discriminante

Covarianza

Medida del grado en que una variable varía de acuerdo con otra (es decir, magnitud y dirección similares).

Función discriminatoria

Función que, aplicada a las variables predictoras, maximiza la separación de las clases.

Ponderaciones discriminatorias

Puntuaciones que resultan de la aplicación de la función discriminatoria y se utilizan para estimar probabilidades de pertenecer a una clase u otra.

Si bien el análisis discriminante comprende varias técnicas, la más utilizada es el *análisis discriminante lineal* (*linear discriminant analysis*), o *LDA*. El método original propuesto por Fisher era en realidad ligeramente diferente del LDA, pero la mecánica es esencialmente la misma. Actualmente, con la aparición de técnicas más sofisticadas como los modelos de árboles y la regresión logística, LDA se utiliza en menor medida.

Sin embargo, es posible que aún encontremos LDA en algunas aplicaciones y con vínculos a otros métodos que se utilizan con mayor frecuencia (como el análisis de componentes principales. Consultar “Análisis de componentes principales” en la página 284).



El análisis discriminante lineal no debe confundirse con la asignación de Dirichlet latente, también conocida como LDA. La asignación de Dirichlet latente se utiliza en el procesamiento de texto y el lenguaje natural y no está relacionada con el análisis discriminante lineal.

Matriz de covarianza

Para comprender el análisis discriminante, primero es necesario introducir el concepto de covarianza (*covariance*) entre dos o más variables. La covarianza mide la relación entre dos variables x y z . Denotamos la media de cada variable con \bar{x} y \bar{z} (consultar “Media” en la página 9). La covarianza $s_{x,z}$ entre x y z viene dada por:

$$s_{x,z} = \frac{\sum_{i=1}^n (x_i - \bar{x})(z_i - \bar{z})}{n - 1}$$

donde n es el número de registros (hay que tener en cuenta que dividimos entre $n - 1$ en lugar de n . Consultar “Grados de libertad, y ¿ n o $n - 1$? ” en la página 15).

Al igual que con el coeficiente de correlación (consultar “Correlación” en la página 30), los valores positivos indican una relación positiva y los negativos una relación negativa.

Sin embargo, la correlación está limitada a estar entre -1 y 1 , mientras que la escala de covarianza depende de la escala de las variables x y z . La *matriz de covarianza* Σ (*covariance matrix* Σ) para x y z consta de las varianzas de las variables individuales, s_x^2 y s_z^2 , en la diagonal (donde la fila y la columna son la misma variable) y las covarianzas entre los pares de variables situadas fuera de la diagonal:

$$\hat{\Sigma} = \begin{bmatrix} s_x^2 & s_{x,z} \\ s_{z,x} & s_z^2 \end{bmatrix}$$



Recordemos que la desviación estándar se usa para normalizar una variable a la puntuación z . La matriz de covarianza se utiliza en una extensión multivariante de este proceso de estandarización. Esto se conoce como distancia de Mahalanobis (consultar “Otras métricas de distancias” en la página 242) y está relacionado con la función LDA.

Discriminante lineal de Fisher

Para simplificar, centrémonos en un problema de clasificación en el que queremos pronosticar un resultado binario y utilizando solo dos variables numéricas continuas (x , z). Técnicamente, el análisis discriminante asume que las variables predictoras son variables continuas distribuidas normalmente, pero en la práctica el método funciona bien incluso para desviaciones no extremas de la normalidad y para predictoras binarias. El discriminante lineal de Fisher distingue la variación *entre* (*between*) grupos, por un lado, de la variación *dentro* (*within*) de los grupos, por otro lado. Específicamente, buscando dividir los registros en dos grupos, el análisis discriminante lineal (LDA) se enfoca en maximizar la suma de cuadrados “entre” SS_{between} (midiendo la variación entre los dos grupos) en relación con la suma de cuadrados “dentro” SS_{within} (midiendo la variación dentro del grupo). En este caso, los dos grupos corresponden a los registros x_0 , z_0 para los cuales $y = 0$ y los registros (x_1, z_1) para los cuales $y = 1$. El método encuentra la combinación lineal $w_x x + w_z z$ que maximiza esa razón de suma de cuadrados:

$$\frac{SS_{\text{between}}}{SS_{\text{within}}}$$

La suma de cuadrados entre grupos es la distancia al cuadrado entre las medias de los dos grupos, y la suma de cuadrados dentro de los grupos es la distribución en torno a las medias dentro de cada grupo, ponderada por la matriz de covarianza. Intuitivamente, al maximizar la suma de cuadrados entre grupos y minimizar la suma de cuadrados dentro de los grupos, este método obtiene la mayor separación entre los dos grupos.

Un ejemplo sencillo

El paquete MASS, asociado con el libro *Modern Applied Statistics with S* de W. N. Venables y B. D. Ripley (Springer, 1994), proporciona una función para calcular LDA con R. Lo que viene a continuación aplica esta función a una muestra de datos de préstamos utilizando dos variables predictoras, `borrower_score` y `payment_inc_ratio`, e imprime las ponderaciones estimadas del discriminador lineal:

```
library(MASS)
loan_lda <- lda(outcome ~ borrower_score + payment_inc_ratio,
                 data=loan3000)
loan_lda$scaling
```

	LD1
borrower_score	7.17583880
payment_inc_ratio	-0.09967559

En Python, podemos usar `LinearDiscriminantAnalysis` de `sklearn.discriminant_analysis`. La propiedad `scalings_` proporciona las ponderaciones estimadas:

```
loan3000.outcome = loan3000.outcome.astype('category')

predictors = ['borrower_score', 'payment_inc_ratio']
outcome = 'outcome'

X = loan3000[predictors]
y = loan3000[outcome]

loan_lda = LinearDiscriminantAnalysis()
loan_lda.fit(X, y)
pd.DataFrame(loan_lda.scalings_, index=X.columns)
```



Uso del análisis discriminante para la selección de características

Si las variables predictoras se normalizan antes de ejecutar LDA, las ponderaciones del discriminador son medidas de importancia variable, lo que proporciona un método computacionalmente eficiente de selección de características.

La función `lda` puede pronosticar la probabilidad de "no pagado" frente a "pagado":

```
pred <- predict(loan_lda)
head(pred$posterior)

paid off      default
1 0.4464563  0.5535437
2 0.4410466  0.5589534
3 0.7273038  0.2726962
4 0.4937462  0.5062538
5 0.3900475  0.6099525
6 0.5892594  0.4107406
```

El método `predict_proba` del modelo ajustado proporciona las probabilidades de los resultados "no pagado" y "pagado":

```
pred = pd.DataFrame(loan_lda.predict_proba(loan3000[predictors]),
                     columns=loan_lda.classes_)
pred.head()
```

Un diagrama de los pronósticos ayuda a ilustrar cómo funciona LDA. Utilizando el resultado de la función `predict`, se genera un diagrama de la probabilidad estimada de incumplimiento de la siguiente manera:

```
center <- 0.5 * (loan_lda$mean[1, ] + loan_lda$mean[2, ])
slope <- -loan_lda$scaling[1] / loan_lda$scaling[2]
intercept <- center[2] - center[1] * slope

ggplot(data=lda_df, aes(x=borrower_score, y=payment_inc_ratio,
                         color=prob_default)) +
  geom_point(alpha=.6) +
  scale_color_gradientn(colors=c('#ca0020', '#f7f7f7', '#0571b0')) +
  scale_x_continuous(expand=c(0,0)) +
  scale_y_continuous(expand=c(0,0), lim=c(0, 20)) +
  geom_abline(slope=slope, intercept=intercept, color='darkgreen')
```

Se crea un gráfico similar en Python usando el siguiente código:

```
# Use scalings and center of means to determine decision boundary
center = np.mean(loan_lda.means_, axis=0)
slope = -loan_lda.scalings_[0] / loan_lda.scalings_[1]
intercept = center[1] - center[0] * slope

# payment_inc_ratio for borrower_score of 0 and 20
x_0 = (0 - intercept) / slope
x_20 = (20 - intercept) / slope

lda_df = pd.concat([loan3000, pred['default']], axis=1)
lda_df.head()

fig, ax = plt.subplots(figsize=(4, 4))
g = sns.scatterplot(x='borrower_score', y='payment_inc_ratio',
                     hue='default', data=lda_df,
                     palette=sns.diverging_palette(240, 10, n=9, as_cmap=True),
                     ax=ax, legend=False)

ax.set_xlim(0, 20)
ax.set_ylim(0, 0.8)
ax.plot((x_0, x_20), (0, 20), linewidth=3)
ax.plot(*loan_lda.means_.transpose())
```

El diagrama resultante se muestra en la figura 5.1. El pronóstico es que los puntos de datos a la izquierda de la línea diagonal son préstamos que no se pagarán (probabilidad superior a 0.5).

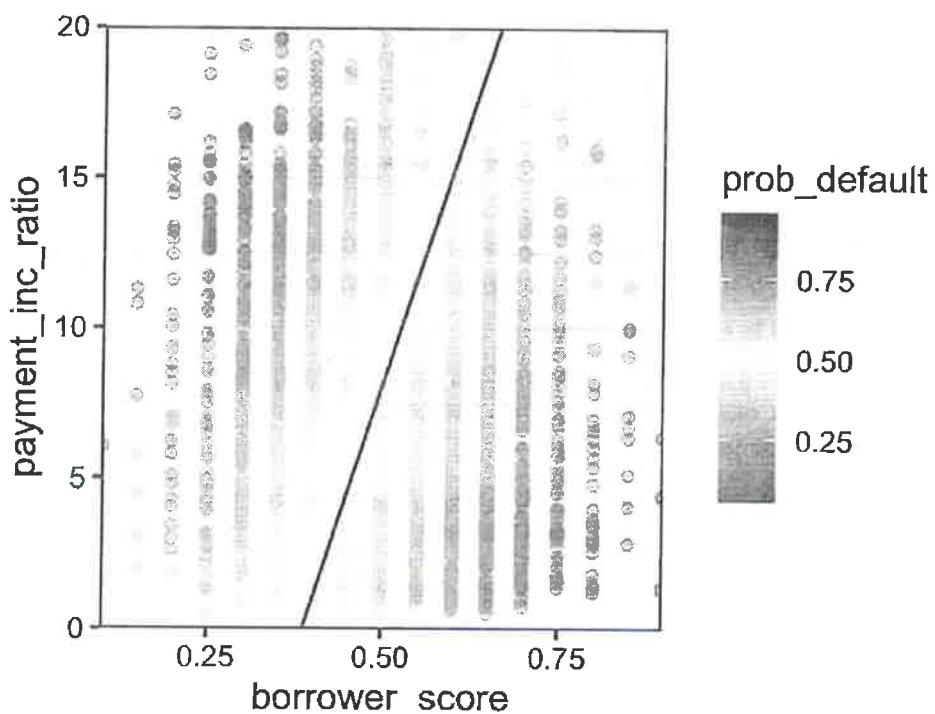


Figura 5.1 Pronóstico de LDA del incumplimiento de préstamos utilizando dos variables: la puntuación de la solvencia del prestatario y la relación pago-ingresos.

Usando las ponderaciones de la función discriminatoria, LDA divide el espacio de la predictora en dos regiones, como muestra la línea continua. Los pronósticos más alejados de la línea en ambas direcciones tienen un mayor nivel de confianza (es decir, una probabilidad más alejada de 0.5).



Generalización del análisis discriminante

Más variables predictoras: mientras que en el texto y el ejemplo de esta sección se han utilizado solo dos variables predictoras, LDA funciona igual de bien con más de dos variables predictoras. El único factor limitador es el número de registros (la estimación de la matriz de covarianza requiere un número suficiente de registros por variable, lo que normalmente no es un problema en las aplicaciones de ciencia de datos).

Hay otras variantes del análisis discriminante. El más conocido es el análisis discriminante cuadrático (QDA). A pesar de su nombre, QDA sigue siendo una función discriminatoria lineal. La principal diferencia es que, en LDA, se supone que la matriz de covarianza es la misma para los dos grupos correspondientes a $Y = 0$ e $Y = 1$. En QDA, se permite que la matriz de covarianza sea diferente para los dos grupos. En la práctica, la diferencia en la mayoría de las aplicaciones no es crítica.

Ideas clave

- El análisis discriminante funciona con predictoras continuas o categóricas, así como con resultados categóricos.
- Mediante la matriz de covarianza se calcula una *función discriminatoria lineal* (*linear discriminant function*), que se utiliza para distinguir los registros que pertenecen a una clase de los que pertenecen a otra.
- Esta función se aplica a los registros para obtener las ponderaciones, o puntuaciones, para cada registro (una ponderación para cada clase posible), lo que determina la clase estimada para el mismo.

Lecturas complementarias

- Tanto *The Elements of Statistical Learning*, 2.^a ed., de Trevor Hastie, Robert Tibshirani y Jerome Friedman (Springer, 2009), y su primo más breve, *An Introduction to Statistical Learning* de Gareth James, Daniela Witten, Trevor Hastie y Robert Tibshirani (Springer, 2013), tienen una sección sobre análisis discriminante.
- *Data Mining for Business Analytics* de Galit Shmueli, Peter Bruce, Nitin Patel, Peter Gedeck, Inbal Yahav y Kenneth Lichtendahl (Wiley, 2007-2020, con ediciones para *R*, *Python*, Excel, y JMP) tiene un capítulo dedicado exclusivamente a análisis discriminante.
- De interés histórico, el artículo original de Fisher sobre el tema, “The Use of Multiple Measurements in Taxonomic Problems”, publicado en 1936 en *Annals of Eugenics* (ahora llamado *Annals of Genetics*), se puede encontrar en línea (<https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1469-1809.1936.tb02137.x>).

Regresión logística

La regresión logística es análoga a la regresión lineal múltiple (ver capítulo 4), con la excepción de que el resultado es binario. Se emplean varias transformaciones para convertir el problema en uno en el que se pueda ajustar un modelo lineal. Al igual que el análisis discriminante, y a diferencia de K-vecinos más cercanos y Bayes ingenuo, la regresión logística es un enfoque de modelo estructurado en lugar de un enfoque centrado en los datos. Se ha convertido en un método popular debido a la rapidez con la que se efectúan sus cálculos, así como a la obtención de un modelo que se presta a una rápida puntuación de nuevos datos.

Términos clave de la regresión logística

Logit

Función que asigna la probabilidad de pertenencia a una clase en un rango de $\pm \infty$ (en lugar de 0 a 1).

Sinónimo

log odds (ver más abajo)

Odds

La proporción de "éxito" (1) a "no éxito" (0).

Log odds

Respuesta en el modelo transformado (ahora lineal), que se asigna de nuevo a una probabilidad.

Función de respuesta logística y logit

Los ingredientes clave para la regresión logística son la *función de respuesta logística* (*logistic response function*) y la función logit, en las que asignamos una probabilidad (que está en una escala de 0 a 1) a una escala más amplia adecuada para el modelado lineal.

El primer paso es pensar en la variable de resultado no como una etiqueta binaria sino como la probabilidad p de que la etiqueta sea un "1". Ingenuamente, podríamos tener la tentación de modelar p como una función lineal de las variables predictoras:

$$p = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q$$

Sin embargo, ajustar este modelo no asegura que p terminará entre 0 y 1, como debe hacerlo una medida de probabilidad.

En su lugar, modelamos p aplicando una función de *respuesta logística* (*logistic response*) o *logit inversa* (*inverse logit*) a las predictoras:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q)}}$$

Esta transformación asegura que p permanece entre 0 y 1.

Para que la expresión exponencial no figure en el denominador, consideramos *oportunidades* (*odds*) en lugar de probabilidades. Las oportunidades, que resultan familiares a los apostadores de cualquier parte del mundo, son la proporción de "éxitos" (1) y "no éxitos" (0). En términos de probabilidades, las oportunidades son la probabilidad de un evento dividida por la probabilidad de que el evento no ocurra. Por

Por ejemplo, si la probabilidad de que un caballo gane es 0.5, la probabilidad de que "no gane" es $(1 - 0.5) = 0.5$, y las oportunidades son 1.0:

$$\text{Oportunidad}(Y = 1) = \frac{p}{1 - p}$$

Podemos obtener la probabilidad a partir de las oportunidades usando la función de oportunidades inversa:

$$p = \frac{\text{Oportunidades}}{1 + \text{Oportunidades}}$$

Combinamos lo anterior con la función de respuesta logística, mostrada anteriormente, para obtener:

$$\text{Oportunidades}(Y = 1) = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_q x_q}$$

Finalmente, tomando el logaritmo en ambos lados, obtenemos una expresión que es una función lineal de las predictoras:

$$\log(\text{Oportunidades}(Y = 1)) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_q x_q$$

La función *log-odds*, también conocida como función *logit*, asigna la probabilidad p de $(0, 1)$ a cualquier valor $(-\infty, +\infty)$. Ver la figura 5.2. El círculo de la transformación se ha completado. Hemos utilizado un modelo lineal para pronosticar una probabilidad, que a su vez podemos asignar a una etiqueta de clase aplicando una regla de corte. Cualquier registro con una probabilidad mayor que el corte se clasifica como 1.

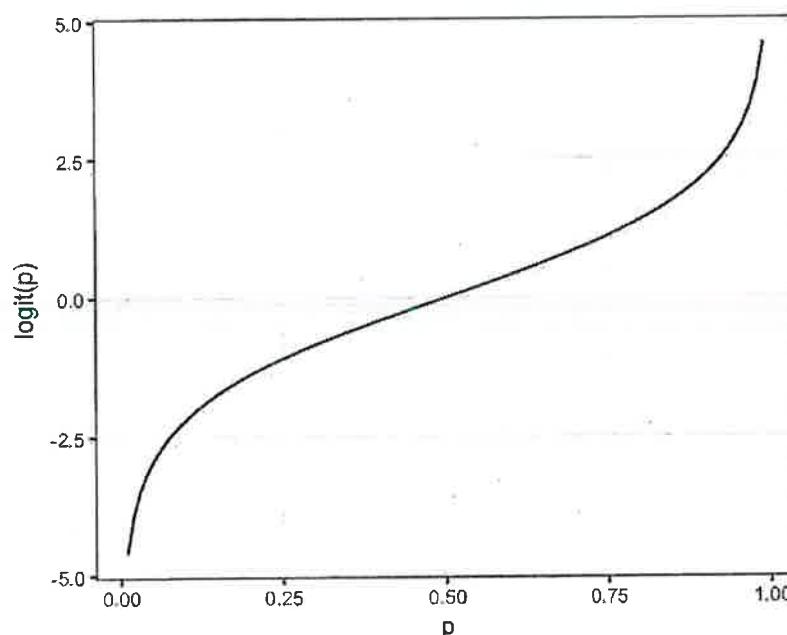


Figura 5.2 Gráfico de la función logit que asigna la probabilidad a una escala adecuada al modelo lineal.

Regresión logística y GLM

La respuesta de la fórmula de regresión logística es el logaritmo de oportunidades del resultado binario de 1. Vemos solo el resultado binario, no el logaritmo de oportunidades, por lo que se necesitan métodos estadísticos especiales para ajustar la ecuación. La regresión logística es una instancia especial del modelo lineal generalizado (GLM) desarrollado para extender la regresión lineal a otros entornos.

Para ajustar la regresión logística en *R*, la función `glm` se usa con el parámetro `family` establecido en binomial. El siguiente código ajusta la regresión logística a los datos de préstamos personales introducidos en "K-vecinos más cercanos" en la página 238:

```
logistic_model <- glm(outcome ~ payment_inc_ratio + purpose_ +
                      home_ + emp_len_ + borrower_score,
                      data=loan_data, family="binomial")
logistic_model

Call: glm(formula = outcome ~ payment_inc_ratio + purpose_ + home_ +
          emp_len_ + borrower_score, family = "binomial", data = loan_data)

Coefficients:
              (Intercept)           payment_inc_ratio
                           1.63809                   0.07974
              purpose_debt_consolidation purpose_home_improvement
                           0.24937                   0.40774
              purpose_major_purchase     purpose_medical
                           0.22963                   0.51048
              purpose_other             purpose_small_business
                           0.62066                   1.21526
              home_own                  home_RENT
                           0.04833                   0.15732
              emp_len_ > 1 Year         borrower_score
                           -0.35673                  -4.61264

Degrees of Freedom: 45341 Total (i.e. Null); 45330 Residual
Null Deviance: 62860
Residual Deviance: 57510    AIC: 57540
```

La respuesta es `outcome`, que toma el valor 0 si el préstamo se ha pagado y 1 si el préstamo no se ha pagado. `purpose_` y `home_` son variables de tipo factor que representan el propósito del préstamo y el estado de propiedad de la vivienda. Al igual que en la regresión lineal, una variable de tipo factor con P niveles se representa con columnas $P - 1$. Por defecto, en *R*, se utiliza la codificación de *referencia* (*reference*) y todos los niveles se comparan con el nivel de referencia (consultar "Variables de tipo factor en la regresión" en la página 163). Los niveles de referencia para estos factores son `credit_card` y `MORTGAGE`, respectivamente. La variable `borrower_score` es la puntuación de 0 a 1 que representa la solvencia del prestatario (de mala a excelente). Esta variable se ha creado a partir de otras variables utilizando K-vecinos más cercanos. Consultar "KNN como motor de características" en la página 247.

En *Python*, usamos la clase `LogisticRegression` de `scikit-learn` de `sklearn.linear_model`. Los argumentos `penalty` y `C` se utilizan para evitar el sobreajuste por la regularización L1 o L2. La regularización está activada por defecto. Para realizar el ajuste sin regularización, establecemos `C` en un valor muy grande. El argumento `solver` selecciona el minimizador utilizado. El método `liblinear` es el predeterminado:

```
predictors = ['payment_inc_ratio', 'purpose_', 'home_', 'emp_len_',
              'borrower_score']
outcome = 'outcome'
X = pd.get_dummies(loan_data[predictors], prefix="", prefix_sep="",
                    drop_first=True)
y = loan_data[outcome]

logit_reg = LogisticRegression(penalty='l2', C=1e42, solver='liblinear')
logit_reg.fit(X, y)
```

A diferencia de *R*, `scikit-learn` obtiene las clases de los valores singulares en `y` (*pagado [paid off]* y *no pagado [default]*). Internamente, las clases están ordenadas alfabéticamente. Como este es el orden inverso de los factores usados en *R*, veremos que los coeficientes están invertidos. El método `predict` proporciona la etiqueta de la clase y `predict_proba` proporciona las probabilidades en el orden disponible en el atributo `logit_reg.classes_`.

Modelos lineales generalizados

Los modelos lineales generalizados (GLM) se caracterizan por dos componentes principales:

- Una distribución de probabilidad o familia (binomial en el caso de regresión logística).
- Una función de enlace, es decir, una función de transformación que asigna la respuesta a las predictoras (logit en el caso de regresión logística).

La regresión logística es, con mucho, la forma más extendida de GLM. Un científico de datos se encontrará con otros tipos de GLM. A veces se utiliza una función de enlace de registro en lugar de logit. En la práctica, es poco probable que el uso de un enlace de registro dé lugar a resultados muy diferentes para la mayoría de las aplicaciones. La distribución de Poisson se usa normalmente para modelar datos de recuento (por ejemplo, la cantidad de veces que un usuario visita una página web en un cierto periodo de tiempo). Otras familias incluyen el binomio negativo y gamma, que a menudo se utilizan para modelar el tiempo transcurrido (por ejemplo, el tiempo hasta que se produce el fallo). A diferencia de la regresión logística, la aplicación de GLM con estos modelos requiere mayor cuidado. Es mejor evitarlos a menos que estemos familiarizados y comprendamos la utilidad y los peligros de estos métodos.

Valores pronosticados de regresión logística

El valor pronosticado de la regresión logística está en términos del logaritmo de oportunidades: $\hat{Y} = \log(\text{Oportunidades } [Y=1])$. La probabilidad pronosticada viene dada por la función de respuesta logística:

$$\hat{P} = \frac{1}{1 + e^{-\hat{Y}}}$$

Por ejemplo, observemos los pronósticos del modelo `logistic_model` en R:

```
pred <- predict(logistic_model)
summary(pred)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
	-2.704774	-0.518825	-0.008539	0.002564	0.505061	3.509606

En Python, podemos adaptar las probabilidades a un marco de datos y usar el método `describe` para obtener las características de la distribución:

```
pred = pd.DataFrame(logit_reg.predict_log_proba(X),
                     columns=loan_data[outcome].cat.categories)
pred.describe()
```

La conversión de estos valores a probabilidades es una transformación sencilla:

```
prob <- 1/(1 + exp(-pred))
> summary(prob)

Min.   1st Qu.    Median     Mean   3rd Qu.    Max.
0.06269  0.37313  0.49787  0.50000  0.62365  0.97096
```

Las probabilidades están disponibles directamente usando los métodos `predict_proba` de `scikit-learn`:

```
pred = pd.DataFrame(logit_reg.predict_proba(X),
                     columns=loan_data[outcome].cat.categories)
pred.describe()
```

Estas están en una escala de 0 a 1 y aún no informan de si el valor pronosticado es no pagado o pagado. Podríamos declarar cualquier valor superior a 0.5 como no pagado. En la práctica, suele ser apropiado un límite más bajo si el objetivo es identificar a los miembros de una clase rara (consultar “El problema de las clases raras” en la página 223).

Interpretación de los coeficientes y de la razón de oportunidades

Una ventaja de la regresión logística es que crea un modelo en el que se pueden anotar nuevos datos de forma inmediata, sin tener que volver a realizar los cálculos. Otra ventaja es la relativa facilidad de interpretación del modelo en comparación con otros métodos de clasificación. La idea conceptual clave es comprender la razón de oportunidades (*odds*)

ratio) o razón de momios. La razón de oportunidades es más fácil de entender para una variable de tipo factor binario X :

$$\text{razón oportunidad} = \frac{\text{Odds}(Y = 1 | X = 1)}{\text{Odds}(Y = 1 | X = 0)}$$

Esta fórmula se interpreta como las oportunidades de que $Y = 1$ cuando $X = 1$ frente a las oportunidades de que $Y = 1$ cuando $X = 0$. Si la razón de oportunidades es 2, entonces las oportunidades de que $Y = 1$ son dos veces mayores cuando $X = 1$ que cuando $X = 0$.

¿Por qué preocuparse por una razón de oportunidades en lugar de probabilidades? Trabajamos con oportunidades porque el coeficiente β_j en regresión logística es el logaritmo de la razón de oportunidades para X_j .

Un ejemplo lo explicará mejor. Para el ajuste del modelo en “Regresión logística y GLM” en la página 210, el coeficiente de regresión para `purpose_small_business` es 1.21526. Esto significa que un préstamo a una pequeña empresa en comparación con un préstamo para pagar la deuda de la tarjeta de crédito aumenta las posibilidades de incumplimiento frente a que se pague en $\exp(1.21526) \approx 3.4$. Evidentemente, los préstamos con el propósito de crear o expandir una pequeña empresa tiene considerablemente más riesgo que otros tipos de préstamos.

La figura 5.3 muestra la relación entre la razón de oportunidades y la razón logarítmica de oportunidades para las razones de oportunidades mayores que 1. Debido a que los coeficientes están en escala logarítmica, un aumento de 1 en el coeficiente da como resultado un aumento de $\exp(1) \approx 2.72$ en la razón de oportunidades.

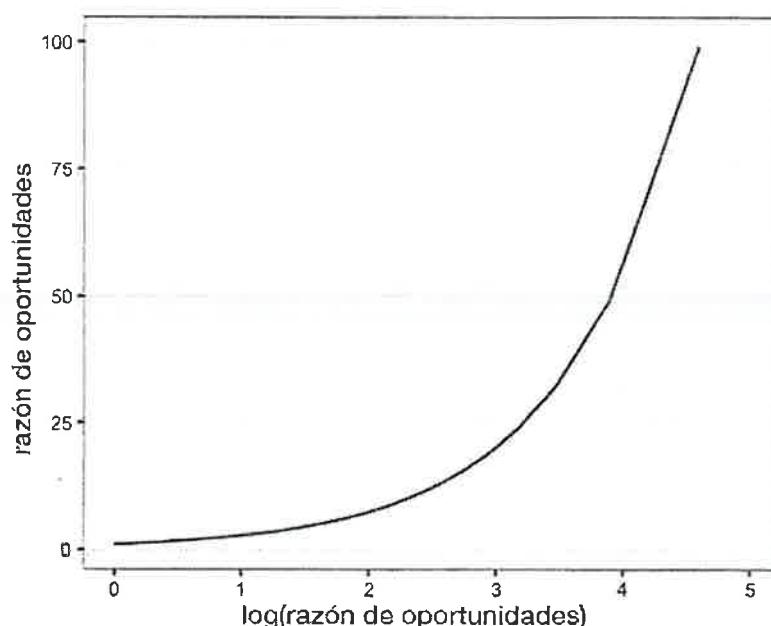


Figura 5.3 Relación entre la razón de oportunidades y la razón logarítmica de oportunidades.

Las razones de oportunidades para las variables numéricas X se pueden interpretar de manera similar: miden el cambio en la razón de oportunidades cuando X cambia en una unidad. Por ejemplo, el efecto de aumentar la relación pago-ingresos de, digamos, 5 a 6 aumenta las probabilidades del incumplimiento del préstamo por un factor de $\exp(0.08244) \approx 1.09$. La variable `borrower_score` es una calificación de la solvencia crediticia de los prestatarios y varía de 0 (bajo) a 1 (alto). Las posibilidades de que los mejores prestatarios en relación con los peores no cumplan con sus préstamos es menor en un factor de $\exp(-4.61264) \approx 0.01$. En otras palabras, el riesgo de incumplimiento de los prestatarios con peor solvencia crediticia es 100 veces mayor que el de los mejores prestatarios!

Regresión lineal y logística: similitudes y diferencias

La regresión lineal y la regresión logística comparten muchos puntos en común. Ambas adoptan una forma paramétrica lineal que relaciona las predictoras con la respuesta. La exploración y búsqueda del mejor modelo se realizan de formas muy similares. Las extensiones del modelo lineal, como el uso de una transformación spline de una predictor (consultar “Splines” en la página 189), son igualmente aplicables a la configuración de regresión logística. La regresión logística se diferencia en dos aspectos fundamentales:

- La forma en que se ajusta el modelo (los mínimos cuadrados no son aplicables).
- La naturaleza y análisis de los residuos del modelo.

Ajuste del modelo

La regresión lineal se ajusta mediante mínimos cuadrados y la calidad del ajuste se evalúa con los estadísticos RECM (RMSE en inglés) y R cuadrado. En regresión logística (a diferencia de la regresión lineal), no existe una solución cerrada, y el modelo debe ajustarse utilizando la *valoración de máxima verosimilitud* (*maximum likelihood estimation*) (MLE). La valoración de máxima verosimilitud es un proceso que intenta encontrar el modelo que es más probable que haya producido los datos que vemos. En la ecuación de regresión logística, la respuesta no es 0 o 1, sino una estimación del logaritmo de oportunidades de que la respuesta sea 1. MLE encuentra la solución de tal manera que el logaritmo de oportunidades estimadas describe mejor el resultado observado. La mecánica del algoritmo implica una optimización quasi-Newton que itera entre un paso de puntuación (*puntuación de Fisher* [*Fisher's scoring*]), basada en los parámetros actuales y la actualización de los parámetros para mejorar el ajuste.

Afortunadamente, la mayoría de los profesionales no necesitan preocuparse por los detalles del algoritmo de ajuste, ya que de ello se ocupa el software. La mayoría de los científicos de datos no necesitarán preocuparse por el método de ajuste, aparte de comprender que se trata de encontrar un buen modelo bajo ciertos supuestos.

Valoración de máxima verosimilitud

Si nos atraen los símbolos estadísticos podemos ver aquí más detalles: comenzamos con un conjunto de datos (X_1, X_2, \dots, X_n) y un modelo de probabilidad $P_\theta(X_1, X_2, \dots, X_n)$ que depende de un conjunto de parámetros θ . El objetivo de MLE es encontrar el conjunto de parámetros θ que maximice el valor de $P_\theta(X_1, X_2, \dots, X_n)$, es decir, que maximice la probabilidad de observación de (X_1, X_2, \dots, X_n) dado el modelo P . En el proceso de ajuste, el modelo se evalúa usando una métrica llamada *desviación* (*deviance*):

$$\text{deviance} = -2 \log \left(P_\theta(X_1, X_2, \dots, X_n) \right)$$

A una desviación menor le corresponde un mejor ajuste.



Gestión de las variables de tipo factor

En regresión logística, las variables de tipo factor deben codificarse como en el caso de la regresión lineal. Consultar “Variables de tipo factor en la regresión” en la página 163. En R y otros tipos de software, esta operación normalmente se produce de forma automática, y generalmente se usa una codificación de referencia. Todos los demás métodos de clasificación que se tratan en este capítulo suelen utilizar la representación del codificador One-Hot (consultar “Codificador One-Hot” en la página 242). En scikit-learn de Python, es más fácil usar la codificación One-Hot, lo que significa que solo se pueden usar en la regresión $n - 1$ de las variables ficticias resultantes.

Evaluación del modelo

Al igual que con otros métodos de clasificación, la regresión logística se evalúa dependiendo de la precisión con la que el modelo clasifica los nuevos datos (consultar “Evaluación de modelos de clasificación” en la página 219). Como en el caso de la regresión lineal, se encuentran disponibles algunas herramientas estadísticas estándar adicionales para examinar y mejorar el modelo. Junto con los coeficientes estimados, R proporciona el error estándar de los coeficientes (SE), el valor z y el valor p:

```
summary(logistic_model)

Call:
glm(formula = outcome ~ payment_inc_ratio + purpose_ + home_ +
    emp_len_ + borrower_score, family = "binomial", data = loan_data)
```

Deviance Residuals:

	Min	1Q	Median	3Q	Max
	-2.51951	-1.06908	-0.05853	1.07421	2.15528

Estadística práctica para ciencia de datos con R y Python

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	1.638092	0.073708	22.224	< 2e-16 ***
payment_inc_ratio	0.079737	0.002487	32.058	< 2e-16 ***
purpose_debt_consolidation	0.249373	0.027615	9.030	< 2e-16 ***
purpose_home_improvement	0.407743	0.046615	8.747	< 2e-16 ***
purpose_major_purchase	0.229628	0.053683	4.277	1.89e-05 ***
purpose_medical	0.510479	0.086780	5.882	4.04e-09 ***
purpose_other	0.620663	0.039436	15.738	< 2e-16 ***
purpose_small_business	1.215261	0.063320	19.192	< 2e-16 ***
home_OWN	0.048330	0.038036	1.271	0.204
home_RENT	0.157320	0.021203	7.420	1.17e-13 ***
emp_len_> 1 Year	-0.356731	0.052622	-6.779	1.21e-11 ***
borrower_score	-4.612638	0.083558	-55.203	< 2e-16 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(El parámetro de dispersión para la familia binomial se toma como 1)

Null deviance: 62857 on 45341 degrees of freedom

Residual deviance: 57515 on 45330 degrees of freedom

AIC: 57539

Número de iteraciones de la puntuación de Fisher: 4

El paquete `statsmodels` tiene una implementación para el modelo lineal generalizado (GLM) que proporciona una información detallada de manera similar:

```
y_numbers = [1 if yi == 'default' else 0 for yi in y]
logit_reg_sm = sm.GLM(y_numbers, X.assign(const=1),
                      family=sm.families.Binomial())
logit_result = logit_reg_sm.fit()
logit_result.summary()
```

Con la interpretación del valor p hay que hacer la misma salvedad que con la regresión, por lo que se debe ver más como un indicador relativo de importancia variable (consultar “Evaluación del modelo” en la página 153) que como una medida formal de significación estadística. El modelo de regresión logística, que tiene una respuesta binaria, no tiene un RECM (RMSE en inglés) o R cuadrado asociados. En cambio, este modelo generalmente se evalúa utilizando métricas más generales para la clasificación. Consultar “Evaluación de modelos de clasificación” en la página 219.

Muchos otros conceptos de regresión lineal se pueden trasladar a la configuración de regresión logística (y a otros GLM). Por ejemplo, podemos utilizar la regresión por pasos, ajustar términos de interacción o incluir términos de spline. Las mismas consideraciones con respecto a las variables de confusión y correlacionadas se aplican a la regresión logística (consultar “Interpretación de la ecuación de regresión” en la página 169). Podemos ajustar modelos aditivos generalizados (consultar “Modelos aditivos generalizados” en la página 192) utilizando el paquete `mgcv` de R:

```
logistic_gam <- gam(outcome ~ s(payment_inc_ratio) + purpose_ +
                     home_ + emp_len_ + s(borrower_score),
                     data=loan_data, family='binomial')
```

La interfaz formula de `statsmodels` también admite estas extensiones en *Python*:

```
import statsmodels.formula.api as smf
formula = ('outcome ~ bs(payment_inc_ratio, df=4) + purpose_ + '
           'home_ + emp_len_ + bs(borrower_score, df=4)')
model = smf.glm(formula=formula, data=loan_data, family=sm.families.Binomial())
results = model.fit()
```

Análisis de los residuos

Un área en la que la regresión logística difiere de la regresión lineal es en el análisis de los residuos. Como en la regresión lineal (ver la figura 4.9), es sencillo calcular en *R* los residuos parciales:

```
terms <- predict(logistic_gam, type='terms')
partial_resid <- resid(logistic_model) + terms
df <- data.frame(payment_inc_ratio = loan_data[, 'payment_inc_ratio'],
                  terms = terms[, 's(payment_inc_ratio)'],
                  partial_resid = partial_resid[, 's(payment_inc_ratio)'])
ggplot(df, aes(x=payment_inc_ratio, y=partial_resid, solid = FALSE)) +
  geom_point(shape=46, alpha=0.4) +
  geom_line(aes(x=payment_inc_ratio, y=terms),
            color='red', alpha=0.5, size=1.5) +
  labs(y='Partial Residual')
```

El diagrama resultante se muestra en la figura 5.4. El ajuste estimado, representado por la línea, se extiende entre dos conjuntos de nubes de puntos. La nube superior corresponde a la respuesta de 1 (préstamos impagados) y la nube inferior corresponde a la respuesta de 0 (préstamos pagados). Esto es muy típico de los residuos en regresión logística, ya que el resultado es binario. El pronóstico se obtiene a partir de logit (logaritmo de la razón de oportunidades), que siempre será un valor finito. El valor real, un 0 o 1 absolutos, corresponde a un logit infinito, ya sea positivo o negativo, por lo que los residuos (que se suman al valor ajustado) nunca serán iguales a 0. Por lo tanto, los puntos del diagrama se encuentran en las nubes de arriba o de abajo, debajo de la línea ajustada en el diagrama de residuos parciales. Los residuos parciales en regresión logística, aunque menos importantes que en la regresión, siguen siendo útiles para confirmar el comportamiento no lineal e identificar registros que tengan una influencia importante.

Actualmente no existen implementaciones de residuos parciales en ninguno de los principales paquetes de *Python*. Proporcionamos el código *Python* para crear el diagrama de residuos parciales en el repositorio de código fuente adjunto.



En la práctica, se pueden ignorar algunos de los resultados de la función `summary`. El parámetro de dispersión no se aplica a la regresión logística, pero está disponible para otros tipos de GLM. La desviación residual y el número de iteraciones de puntuación están relacionados con el método de ajuste de máxima verosimilitud. Consultar “Valoración de máxima verosimilitud” en la página 215.

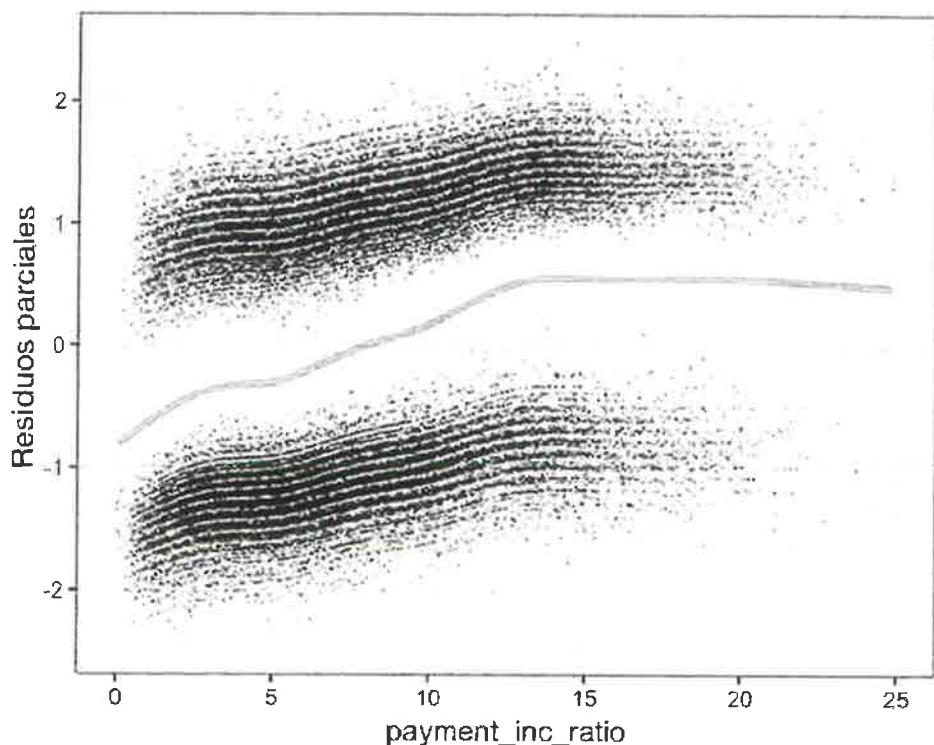


Figura 5.4 Residuos parciales de la regresión logística.

Ideas clave

- La regresión logística es como la regresión lineal, excepto que el resultado es una variable binaria.
- Se necesitan varias transformaciones para obtener el modelo en una forma que se pueda ajustar como modelo lineal, con el logaritmo de la razón de oportunidades como variable de respuesta.
- Una vez que se ajusta el modelo lineal (mediante un proceso iterativo), el logaritmo de oportunidades se vuelve a asignar a una probabilidad.
- La regresión logística es popular por su rapidez en el cálculo y crea un modelo que puede puntuar nuevos datos con solo unas pocas operaciones aritméticas.

Lecturas complementarias

- La referencia clásica sobre regresión logística es *Applied Logistic Regression*, 3.^a ed., de David Hosmer, Stanley Lemeshow y Rodney Sturdivant (Wiley, 2013).
- También son populares dos libros de Joseph Hilbe: *Logistic Regression Models* (muy completo, 2017) y *Practical Guide to Logistic Regression* (conciso, 2015), ambos de Chapman & Hall/CRC Press.

- Tanto *The Elements of Statistical Learning*, 2.^a ed., de Trevor Hastie, Robert Tibshirani y Jerome Friedman (Springer, 2009), y su primo más breve, *An Introduction to Statistical Learning* de Gareth James, Daniela Witten, Trevor Hastie y Robert Tibshirani (Springer, 2013), tienen una sección sobre regresión logística.
- *Data Mining for Business Analytics*, de Galit Shmueli, Peter Bruce, Nitin Patel, Peter Gedeck, Inbal Yahav y Kenneth Lichtendahl (Wiley, 2007-2020, con ediciones de R, Python, Excel, y JMP), dedica un capítulo completo a la regresión logística.

Evaluación de modelos de clasificación

Es corriente en el modelado predictivo entrenar varios modelos diferentes, aplicar cada uno a una muestra reservada y evaluar su rendimiento. A veces, después de que se hayan evaluado y ajustado varios modelos, y si hay suficientes datos, se utiliza una tercera muestra reservada, no utilizada anteriormente, para estimar cómo funcionará el modelo elegido con datos completamente nuevos. Diferentes disciplinas y profesionales también utilizarán los términos *validación* (*validation*) y *prueba* (*test*) para referirse a las muestras reservadas. Básicamente, el proceso de evaluación intenta aprender qué modelo genera los pronósticos más precisos y útiles.

Términos clave de la evaluación de modelos de clasificación

Precisión

Porcentaje (o proporción) de casos clasificados correctamente.

Matriz de confusión

Visualización tabular (2×2 en el caso binario) de los recuentos de registros según su estado de clasificación previsto y real.

Sensibilidad

Porcentaje (o proporción) de todos los 1 que se clasifican correctamente como 1.

Sinónimo

exhaustividad

Especificidad

Porcentaje (o proporción) de todos los 0 que se clasifican correctamente como 0.

Precisión

Porcentaje (o proporción) de 1 pronosticados que en realidad son 1.

Curva ROC

Diagrama de la sensibilidad frente a la especificidad.

Sustentación

Medida de la eficacia del modelo para identificar 1 (comparativamente raros) con diferentes límites de probabilidad

Una forma sencilla de medir el rendimiento de la clasificación es contar la proporción de pronósticos correctos, es decir, medir la *precisión* (*accuracy*). La precisión es simplemente una medida del error total:

$$\text{accuracy} = \frac{\sum \text{TruePositive} + \sum \text{TrueNegative}}{\text{SampleSize}}$$

En la mayoría de los algoritmos de clasificación, a cada caso se le asigna una “probabilidad estimada de ser un 1”. El punto de decisión predeterminado, o de corte, es típicamente 0.50 o 50 %. Si la probabilidad es superior a 0.5, la clasificación es “1”, de lo contrario, es “0”. Un punto de corte predeterminado alternativo es la probabilidad predominante de 1 en los datos.

Matriz de confusión

En el corazón de las métricas de clasificación se encuentra la *matriz de confusión* (*confusion matrix*). La matriz de confusión es una tabla que muestra el número de pronósticos correctos e incorrectos categorizados por tipo de respuesta. Hay varios paquetes disponibles en *R* y *Python* para calcular la matriz de confusión, pero en el caso binario, es sencillo calcularla a mano.

Para ilustrar la matriz de confusión, consideremos el modelo `logistic_gam` que se ha entrenado con un conjunto de datos equilibrados con un número igual de préstamos no pagados y pagados (ver figura 5.4). Siguiendo las convenciones habituales, $Y = 1$ corresponde al evento que nos interesa (por ejemplo, no pagado), e $Y = 0$ corresponde a un evento negativo (o habitual) (por ejemplo, pagado). A continuación, se calcula en *R* la matriz de confusión para el modelo `logistic_gam` aplicado a todo el conjunto de entrenamiento (desequilibrado):

```
pred <- predict(logistic_gam, newdata=train_set)
pred_y <- as.numeric(pred > 0)
true_y <- as.numeric(train_set$outcome=='default')
true_pos <- (true_y==1) & (pred_y==1)
true_neg <- (true_y==0) & (pred_y==0)
false_pos <- (true_y==0) & (pred_y==1)
false_neg <- (true_y==1) & (pred_y==0)
conf_mat <- matrix(c(sum(true_pos), sum(false_pos),
                      sum(false_neg), sum(true_neg)), 2, 2)
colnames(conf_mat) <- c('Yhat = 1', 'Yhat = 0')
rownames(conf_mat) <- c('Y = 1', 'Y = 0')
conf_mat
      Yhat = 1 Yhat = 0
Y = 1 14295    8376
Y = 0  8052    14619
```

En *Python*:

```
pred = logit_reg.predict(X)
pred_y = logit_reg.predict(X) == 'default'
true_y = y == 'default'
true_pos = true_y & pred_y
true_neg = ~true_y & ~pred_y
false_pos = ~true_y & pred_y
```

```

false_neg = true_y & ~pred_y
conf_mat = pd.DataFrame([[np.sum(true_pos), np.sum(false_neg)],
                         [np.sum(false_pos), np.sum(true_neg)]],
                        index=['Y = default', 'Y = paid off'],
                        columns=['Yhat = default', 'Yhat = paid off'])

conf_mat

```

Los resultados pronosticados son las columnas y los verdaderos resultados son las filas. Los elementos diagonales de la matriz muestran el número de pronósticos correctos y los elementos fuera de la diagonal muestran el número de pronósticos incorrectos. Por ejemplo, 14 295 préstamos no pagados se pronosticaron correctamente, pero 8376 préstamos no pagados se pronosticaron incorrectamente como pagados.

La figura 5.5 muestra la relación entre la matriz de confusión para una respuesta binaria Y y diferentes métricas (consultar “Precisión, exhaustividad y especificidad” en la página 223 para obtener más información sobre las métricas). Al igual que en el ejemplo de los datos de préstamos, la respuesta real se encuentra en las filas y la respuesta prevista se encuentra en las columnas. Los cuadros diagonales (arriba a la izquierda, abajo a la derecha) muestran cuándo los pronósticos \hat{Y} predicen correctamente la respuesta. Una métrica importante que no se menciona explícitamente es la *tasa (rate)* de falsos positivos (la imagen espejular de la precisión). Cuando los 1 son poco frecuentes, la proporción de falsos positivos entre todos los positivos previstos puede ser alta, lo que lleva a una situación poco intuitiva en la que un 1 pronosticado probablemente sea un 0. Este problema afecta a las pruebas de detección médica (por ejemplo, mamografías) que se aplican ampliamente: debido a la relativa rareza de la afección, lo más probable es que los resultados positivos de la prueba no signifiquen cáncer de mama. Esto genera mucha confusión en las personas que se someten a la prueba.

		Respuesta prevista			
		$\hat{y} = 1$	$\hat{y} = 0$		
Respuesta real	$y = 1$	Verdadero Positivo	Falso Negativo	Exhaustividad (Sensibilidad) $VP/(y=1)$	
	$y = 0$	Falso Positivo	Verdadero Negativo	Especificidad $TN/(y=0)$	
	Prevalencia $(y=1)/\text{total}$	Precisión $VP/(\hat{y}=1)$		Precisión $(TP+TN)/\text{total}$	

Figura 5.5 Matriz de confusión para una respuesta binaria y varias métricas.



Aquí, presentamos la respuesta real en las filas y la respuesta pronosticada en las columnas, pero no es raro ver esta disposición al revés. Un ejemplo destacable es el popular paquete caret de R.

El problema de las clases raras

En muchos casos, existe un desequilibrio en las clases a pronosticar, con una clase mucho más prevalente que la otra, por ejemplo, reclamaciones de seguros legítimas frente a fraudulentas, o personas que navegan por la red frente a compradores en un sitio web. La clase rara (por ejemplo, las reclamaciones fraudulentas) suele ser la clase de mayor interés y normalmente se designa como 1, en contraste con los 0 más frecuentes. En el escenario típico, los 1 son el caso más importante, en el sentido de que clasificarlos incorrectamente como 0 es más costoso que clasificarlos incorrectamente como 1. Por ejemplo, identificar correctamente una reclamación de seguro fraudulenta puede ahorrar miles de dólares. Por otro lado, identificar correctamente una reclamación no fraudulenta simplemente nos ahorra el coste y el esfuerzo de revisar la reclamación a mano con una revisión más cuidadosa (que es lo que haríamos si la reclamación fuera etiquetada como "fraudulenta").

En tales casos, a menos que las clases sean fácilmente separables, el modelo de clasificación más preciso puede ser uno que simplemente clasifique todo como 0. Por ejemplo, si solo se terminan comprando en una tienda web el 0.1 % de las personas que navegan, un modelo que pronostique que cada persona que navega se irá sin comprar tendrá una precisión del 99.9 %. Sin embargo, será inútil. En cambio, estaríamos contentos con un modelo que sea menos preciso en general, pero que sea bueno para seleccionar a los compradores, incluso aunque por el camino clasifique erróneamente a algunos que no compran.

Precisión, exhaustividad y especificidad

Las métricas que no se ocupan exclusivamente de la precisión (métricas que tienen más matices) se utilizan normalmente en la evaluación de modelos de clasificación. Algunas de estas métricas tienen una larga historia en estadística, especialmente en bioestadística, donde se utilizan para describir el rendimiento esperado de las pruebas de diagnóstico. La precisión (*precision*) mide la precisión de un resultado positivo previsto (consultar la figura 5.5):

$$\text{precision} = \frac{\Sigma \text{TruePositive}}{\Sigma \text{TruePositive} + \Sigma \text{FalsePositive}}$$

La *exhaustividad* (*recall*), también conocida como *sensibilidad* (*sensitivity*), mide la fuerza del modelo para pronosticar un resultado positivo, es decir, la proporción de 1 que identifica correctamente (consultar la figura 5.5). El término *sensibilidad* (*sensitivity*) se utiliza mucho en bioestadística y diagnósticos médicos, mientras que la *exhaustividad* (*recall*) la utiliza más el colectivo de aprendizaje automático. La definición de *exhaustividad* es:

$$\text{recall} = \frac{\Sigma \text{TruePositive}}{\Sigma \text{TruePositive} + \Sigma \text{FalseNegative}}$$

Otra métrica que se utiliza es la *especificidad (specificity)*, que mide la capacidad de un modelo para pronosticar un resultado negativo:

$$\text{specificity} = \frac{\sum \text{TrueNegative}}{\sum \text{TrueNegative} + \sum \text{FalsePositive}}$$

Podemos calcular las tres métricas mediante `conf_mat` de *R*:

```
# precision
conf_mat[1, 1] / sum(conf_mat[, 1])
# recall
conf_mat[1, 1] / sum(conf_mat[1, ])
# specificity
conf_mat[2, 2] / sum(conf_mat[2, ])
```

A continuación se muestra el código equivalente para calcular las métricas con *Python*:

```
conf_mat = confusion_matrix(y, logit_reg.predict(X))
print('Precision', conf_mat[0, 0] / sum(conf_mat[:, 0]))
print('Recall', conf_mat[0, 0] / sum(conf_mat[0, :]))
print('Specificity', conf_mat[1, 1] / sum(conf_mat[1, :]))

precision_recall_fscore_support(y, logit_reg.predict(X),
                                 labels=['default', 'paid off'])
```

`scikit-learn` tiene el método personalizado `precision_recall_fscore_support` que calcula la precisión y la exhaustividad/especificidad de una sola vez.

Curva ROC

Podemos ver que existe un equilibrio entre la exhaustividad y la especificidad. Capturar más 1 generalmente significa clasificar erróneamente más 0 como 1. El clasificador ideal haría un excelente trabajo al clasificar los 1, sin clasificar erróneamente más 0 como 1.

La métrica que captura este equilibrio es la curva "Características operativas del receptor" ("Receiver Operating Characteristics"), generalmente conocida como la curva (*curve*) ROC. La curva ROC presenta la exhaustividad (sensibilidad) en el eje y y la especificidad en el eje x¹². La curva muestra el equilibrio entre la exhaustividad y la especificidad a medida que cambia el corte para determinar cómo clasificar un registro. La sensibilidad (exhaustividad) se traza en el eje y, y podemos encontrar el eje x etiquetado de dos formas:

- La especificidad se refleja en el eje x, con 1 a la izquierda y 0 a la derecha.
- La especificidad 1 se refleja en el eje x, con 0 a la izquierda y 1 a la derecha.

¹² La curva ROC se utilizó por primera vez durante la Segunda Guerra Mundial para describir el rendimiento de las estaciones receptoras de radar, cuyo trabajo consistía en identificar (clasificar) correctamente las señales de radar reflejadas y alertar a las fuerzas de defensa sobre aviones enemigos.

Estadística práctica para ciencia de datos con R y Python

La curva se ve idéntica independientemente de la forma utilizada. El proceso para el cálculo es:

1. Ordenamos los registros por la probabilidad pronosticada de que sea un 1, comenzando con el más probable y terminando con el menos probable.
2. Calculamos la especificidad acumulada y la exhaustividad basándonos en los registros ordenados.

El cálculo en *R* de la curva ROC es sencillo. El siguiente código calcula la curva ROC para los datos de los préstamos:

```
idx <- order(-pred)
recall <- cumsum(true_y[idx] == 1) / sum(true_y == 1)
specificity <- (sum(true_y == 0) - cumsum(true_y[idx] == 0)) / sum(true_y == 0)
roc_df <- data.frame(recall = recall, specificity = specificity)
ggplot(roc_df, aes(x=specificity, y=recall)) +
  geom_line(color='blue') +
  scale_x_reverse(expand=c(0, 0)) +
  scale_y_continuous(expand=c(0, 0)) +
  geom_line(data=data.frame(x=(0:100) / 100), aes(x=x, y=1-x),
            linetype='dotted', color='red')
```

En *Python*, podemos usar la función `sklearn.metrics.roc_curve` de *scikit-learn* para calcular la información requerida para la curva ROC. En *R* podemos encontrar paquetes similares, por ejemplo, *ROCR*:

```
fpr, tpr, thresholds = roc_curve(y, logit_reg.predict_proba(X)[:,0],
                                   pos_label='default')
roc_df = pd.DataFrame({'recall': tpr, 'specificity': 1 - fpr})
ax = roc_df.plot(x='specificity', y='recall', figsize=(4, 4), legend=False)
ax.set_xlim(0, 1)
ax.set_ylim(0, 1)
ax.plot((1, 0), (0, 1))
ax.set_xlabel('specificity')
ax.set_ylabel('recall')
```

El resultado se muestra en la figura 5.6. La línea diagonal punteada corresponde a un clasificador que no es mejor que el azar. Un clasificador extremadamente eficaz (o, en el caso de valoraciones en medicina, una prueba de diagnóstico extremadamente eficaz) tendrá una curva ROC que abraza la esquina superior izquierda. Identificará correctamente lotes de 1, sin clasificar erróneamente lotes de 0 como 1. Para este modelo, si queremos un clasificador con una especificidad de al menos el 50 %, la exhaustividad es de aproximadamente el 75 %.



Curva de precisión-exhaustividad

Además de las curvas ROC, puede resultar esclarecedor examinar la curva de la precisión en relación con la exhaustividad (precision-recall) (PR) (<https://www.biostat.wisc.edu/~page/rocpr.pdf>). Las curvas PR se calculan de manera similar, excepto que los datos se ordenan de menor a mayor probabilidad y se calculan los estadísticos de precisión acumulada y exhaustividad. Las curvas de PR son especialmente útiles para evaluar datos con resultados muy desequilibrados.

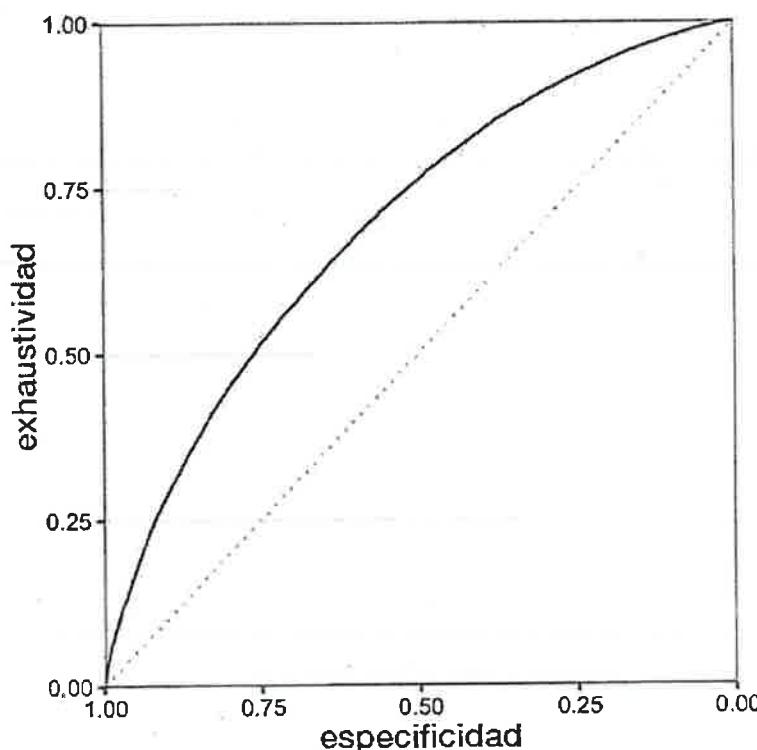


Figura 5.6 Curva ROC de los datos de préstamos.

AUC

La curva ROC es una valiosa herramienta gráfica, pero por sí misma no constituye una medida única para el rendimiento de un clasificador. Sin embargo, la curva ROC se puede utilizar para mostrar la métrica del área por debajo de la curva (area underneath the curve, (AUC)). AUC es simplemente el área total bajo la curva ROC. Cuanto mayor sea el valor de AUC, más eficaz será el clasificador. Un AUC de 1 indica un clasificador perfecto: obtiene todos los 1 clasificados correctamente y no clasifica erróneamente ningún 0 como 1.

Un clasificador completamente ineficaz, la línea diagonal, tendrá un AUC de 0.5.

La figura 5.7 muestra el área bajo la curva ROC para el modelo de préstamos. El valor de AUC se puede calcular en R mediante una integración numérica:

```
sum(roc_df$recall[-1] * diff(1 - roc_df$specificity))
[1] 0.6926172
```

En Python, podemos calcular o bien la precisión como se muestra para R o bien usar la función `sklearn.metrics.roc_auc_score` de `scikit-learn`. Deberemos proporcionar el valor esperado como 0 o 1:

```
print(np.sum(roc_df.recall[:-1] * np.diff(1 - roc_df.specificity)))
print(roc_auc_score([1 if yi == 'default' else 0 for yi in y],
                    logit_reg.predict_proba(X)[:, 0]))
```

El valor de AUC para el modelo es de aproximadamente 0.69, que corresponde a un clasificador relativamente débil.

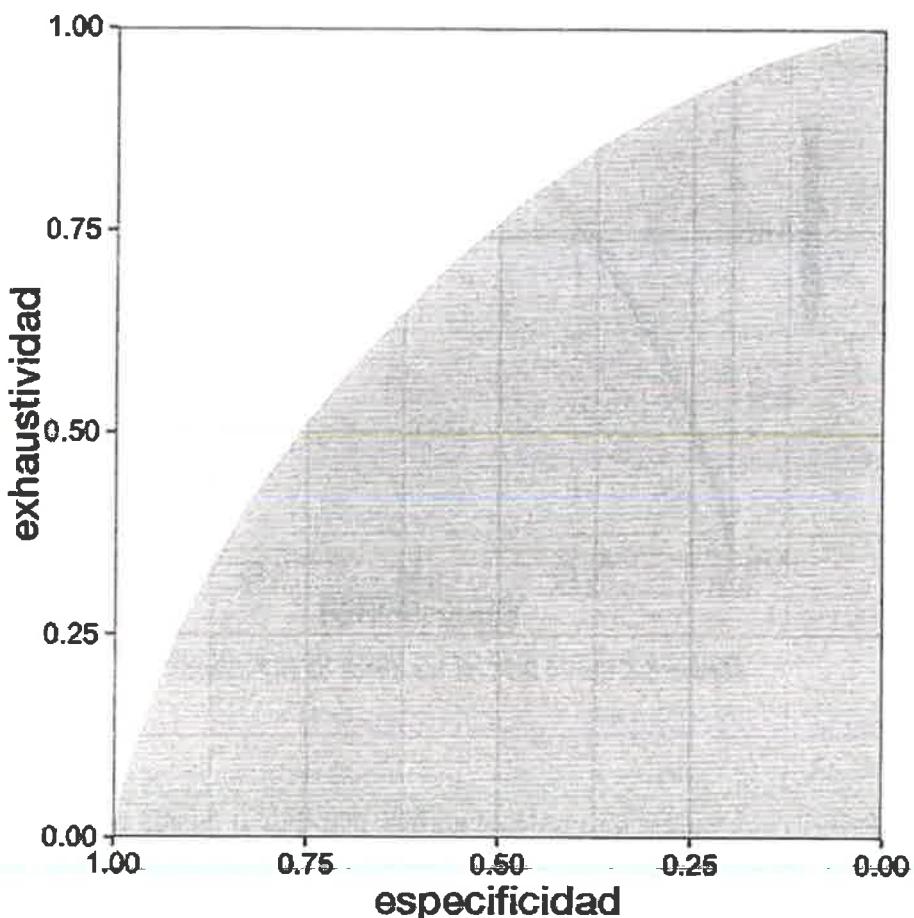


Figura 5.7 Área bajo la curva ROC para los datos de préstamos.



Confusión con las tasas de falsos positivos

Las tasas de falsos positivos/negativos a menudo se confunden o se vinculan a la especificidad o sensibilidad (¡incluso en publicaciones y en el software!). A veces, la tasa de falsos positivos se define como la proporción de verdaderos negativos que dan positivo en la prueba. En muchos casos (como la detección de intrusiones en la red), el término se utiliza para referirse a la proporción de señales positivas que son verdaderas negativas.

Sustentación

La utilización de AUC como métrica para evaluar un modelo constituye una mejora con respecto a la mera precisión, ya que puede evaluar lo bien que un clasificador gestiona el equilibrio entre la precisión general y la necesidad de identificar los 1 más importantes. Pero no aborda en su totalidad el problema de los casos raros, en los que es necesario

reducir el corte de probabilidad del modelo por debajo de 0.5 para evitar que todos los registros se clasifiquen como 0. En tales casos, para que un registro se clasifique como 1, podría ser suficiente tener una probabilidad de 0.4, 0.3 o menor. En efecto, terminamos sobreidentificando los 1, lo que refleja su mayor importancia.

Cambiar este corte mejorará nuestras posibilidades de atrapar los 1 (a costa de clasificar erróneamente más 0 como 1). Pero ¿cuál es el corte óptimo?

El concepto de sustentación nos permitirá responder seguidamente a esa pregunta. Para ello, consideremos los registros en el orden de la probabilidad pronosticada de que sean 1. Digamos, del 10 % superior clasificados como 1, ¿cuánto mejor lo ha hecho el algoritmo en comparación con la referencia de hacer una selección simplemente a ciegas? Si podemos obtener una respuesta del 0.3 % en este decil superior en lugar del 0.1 % que obtenemos en general al elegir al azar, se dice que el algoritmo tiene una *sustentación* (*lift*) (también llamado ganancias [*gains*]) de 3 en el decil superior. Un gráfico de sustentación (gráfico de ganancias) la cuantifica en el rango de los datos. Puede producirse decil a decil o de forma continua sobre el rango de los datos.

Para calcular el gráfico de sustentación, primero elaboramos un *gráfico de ganancias acumuladas* (*cumulative gains chart*) que muestra la exhaustividad en el eje y, y el número total de registros en el eje x. La *curva de sustentación* (*lift curve*) es la razón entre las ganancias acumuladas y la línea diagonal correspondiente a la selección aleatoria. Los gráficos de ganancias de deciles (*decile gains charts*) son una de las técnicas más antiguas en el modelado predictivo; data de tiempos anteriores al comercio por Internet. Fueron particularmente populares entre los profesionales del correo directo. El correo directo es un método de publicidad costoso si se aplica indiscriminadamente, y los anunciantes utilizaron modelos predictivos (bastante simples, en sus comienzos) para identificar a los clientes potenciales con la perspectiva más probable de rentabilidad.



Elevación

A veces, el término *elevación* (*uplift*) se usa para significar lo mismo que sustentación (*lift*). Se emplea un significado alternativo en un entorno más restrictivo, cuando se ha realizado una prueba A/B y el tratamiento (A o B) se usa luego como una variable predictiva en un modelo predictivo. La elevación (*uplift*) es la mejora en la respuesta pronosticada *para un caso individual* (*for an individual case*) con el tratamiento A frente al tratamiento B. Esto se determina puntuando el caso individual primero con la predictora establecida en A, y luego nuevamente con la predictora cambiada a B. Los especialistas en marketing y los consultores de campañas políticas utilizan este método para determinar cuál de los dos tratamientos de mensajería se debe utilizar con qué clientes o votantes.

La curva de sustentación nos permite ver las consecuencias de establecer diferentes cortes de probabilidad para clasificar registros como 1. Puede ser un paso intermedio para establecer un nivel de corte adecuado. Por ejemplo, una autoridad fiscal puede tener solo una cierta cantidad de recursos que puede gastar en auditorías fiscales y quiere gastarlos en investigar los fraudes fiscales más probables. Con la limitación de recursos en mente, la autoridad usaría un gráfico de sustentación para estimar dónde trazar la línea entre las declaraciones de impuestos seleccionadas para auditoría y las que no hay que comprobar.

Ideas clave

- La precisión (el porcentaje de clasificaciones previstas que son correctas) no es más que un primer paso en la evaluación de un modelo.
- Otras métricas (exhaustividad, especificidad, precisión) se centran en características de rendimiento más específicas (por ejemplo, la exhaustividad mide lo bueno que es un modelo para identificar correctamente los 1).
- AUC (área bajo la curva ROC) es una métrica corriente de la capacidad de un modelo para distinguir 1 de 0.
- De manera similar, la elevación mide la eficacia de un modelo para identificar los 1 y, a menudo, se calcula decil por decil, comenzando con los 1 más probables.

Lecturas complementarias

Por lo general, la evaluación y la valoración se tratan en el contexto de un modelo particular (por ejemplo, K-vecinos más cercanos o árboles de decisión). Tres libros que tratan el tema en sus correspondientes capítulos son:

- *Data Mining*, 3.^a ed., de Ian Whitten, Eibe Frank y Mark Hall (Morgan Kaufmann, 2011).
- *Modern Data Science with R* de Benjamin Baumer, Daniel Kaplan y Nicholas Horton (Chapman & Hall/CRC Press, 2017).
- *Data Mining for Business Analytics* de Galit Shmueli, Peter Bruce, Nitin Patel, Peter Gedeck, Inbal Yahav y Kenneth Lichtendahl (Wiley, 2007-2020, con ediciones para *R*, *Python*, Excel y JMP).

Estrategias para datos que no están equilibrados

La sección anterior trataba sobre la evaluación de modelos de clasificación utilizando métricas que van más allá de la mera precisión y son adecuadas para datos que no están equilibrados, datos en los que el resultado que nos interesa (compra en un sitio web, fraude de seguros, etc.) es poco corriente. En esta sección, analizamos estrategias

adicionales que pueden mejorar el rendimiento del modelado predictivo con datos que no están equilibrados.

Términos clave de datos que no están equilibrados

Submuestra

Utiliza menos registros de clases prevalentes en el modelo de clasificación.

Sinónimo

Muestra reducida

Sobremuestra

Utiliza más registros de clases raras en el modelo de clasificación, con bootstrapping si es necesario.

Sinónimo

muestra ampliada

Aumento/disminución de la ponderación

Aplicar más (o menos) ponderación a la clase rara (o prevalente) en el modelo.

Generación de datos

Como bootstrapping, excepto que cada nuevo registro bootstrap es ligeramente diferente de su fuente.

Puntuación z

Valor que resulta después de la estandarización.

K

Número de vecinos considerados en el cálculo del vecino más cercano.

Submuestreo

Si tenemos suficientes datos, como es el caso de los datos de préstamos, una solución es submuestrear (*undersample*) (o reducir) la clase predominante, de modo que los datos a modelar estén más equilibrados entre 0 y 1. La idea básica del submuestreo es que los datos de la clase dominante tienen muchos registros redundantes. Tratar con un conjunto de datos más pequeño y equilibrado mejora el rendimiento del modelo y facilita la preparación de los datos y la exploración y prueba de modelos.

¿Cuántos datos son suficientes? Depende de la aplicación, pero en general, basta con tener decenas de miles de registros para la clase menos dominante. Cuanto más fácilmente se distingan los 1 de los 0, menos datos se necesitan.

Los datos de préstamos analizados en “Regresión logística”, en la página 208, se basaron en un conjunto de capacitación equilibrado: la mitad de los préstamos fueron pagados y

Estadística práctica para ciencia de datos con R y Python

la otra mitad estaban sin pagar. Los valores pronosticados fueron similares: la mitad de las probabilidades fueron menores de 0.5 y la mitad fueron mayores de 0.5. En el conjunto de datos completo, solo alrededor del 19 % de los préstamos estaban sin pagar, como se muestra en *R*:

```
mean(full_train_set$outcome=='default')
[1] 0.1889455
```

En *Python*:

```
print('percentage of loans in default: ',
      100 * np.mean(full_train_set.outcome == 'default'))
```

¿Qué sucede si usamos el conjunto de datos completo para entrenar el modelo? Veamos cómo se hace en porcentaje *R*:

```
full_model <- glm(outcome ~ payment_inc_ratio + purpose_ + home_ +
                    emp_len_ + dti + revol_bal + revol_util,
                    data=full_train_set, family='binomial')
pred <- predict(full_model)
mean(pred > 0)
[1] 0.003942094
```

Y en *Python*:

```
predictors = ['payment_inc_ratio', 'purpose_', 'home_', 'emp_len_',
              'dti', 'revol_bal', 'revol_util']
outcome = 'outcome'
X = pd.get_dummies(full_train_set[predictors], prefix="", prefix_sep="",
                    drop_first=True)
y = full_train_set[outcome]

full_model = LogisticRegression(penalty='l2', C=1e42, solver='liblinear')
full_model.fit(X, y)
print('percentage of loans predicted to default: ',
      100 * np.mean(full_model.predict(X) == 'default'))
```

Se prevé que solo el 0.39 % de los préstamos estén sin pagar, o menos de 1/47 del número esperado¹³. Los préstamos pagados superan a los préstamos sin pagar porque el modelo se entrena utilizando todos los datos por igual. Pensándolo intuitivamente, la presencia de tantos préstamos sin pagar, junto con la inevitable variabilidad en los datos de pronóstico, significa que, incluso para un préstamo sin pagar, es probable que el modelo encuentre algunos préstamos sin pagar a los que es similar, por casualidad. Cuando se utilizó una muestra equilibrada, se pronosticó que aproximadamente el 50 % de los préstamos estaban sin pagar.

Sobremuestreo y aumento/disminución de la ponderación

Una de las críticas al método de submuestreo es que descarta datos y no utiliza toda la información disponible. Si tenemos un conjunto de datos relativamente pequeño, y la

¹³ Debido a las diferencias en la implementación, los resultados en *Python* difieren ligeramente: el 1 %, o aproximadamente 1/18 del número esperado.

clase más rara contiene algunos cientos o miles de registros, entonces el submuestreo de la clase dominante tiene el riesgo de desechar información útil. En este caso, en lugar de reducir la muestra del caso dominante, debemos sobremuestrear (aumentar la muestra) la clase más rara extrayendo filas adicionales con reposición (bootstrapping).

Podemos lograr un efecto similar ponderando los datos. Muchos algoritmos de clasificación adoptan un argumento de ponderación que nos permitirá aumentar o disminuir la ponderación de los datos. Por ejemplo, aplicamos un vector de ponderación a los datos de préstamos utilizando el argumento `weight` para `glm` en R:

```
wt <- ifelse(full_train_set$outcome=='default',
              1 / mean(full_train_set$outcome == 'default'), 1)
full_model <- glm(outcome ~ payment_inc_ratio + purpose_ + home_ +
                  emp_len_ + dti + revol_bal + revol_util,
                  data=full_train_set, weight=wt, family='quasibinomial')
pred <- predict(full_model)
mean(pred > 0)
[1] 0.5767208
```

La mayoría de los métodos de `scikit-learn` permiten especificar ponderaciones en la función `fit` usando el argumento de palabra clave `sample_weight`:

```
default_wt = 1 / np.mean(full_train_set.outcome == 'default')
wt = [default_wt if outcome == 'default' else 1
      for outcome in full_train_set.outcome]
full_model = LogisticRegression(penalty="l2", C=1e42, solver='liblinear')
full_model.fit(X, y, sample_weight=wt)
print('percentage of loans predicted to default (weighting): ',
     100 * np.mean(full_model.predict(X) == 'default'))
```

Las ponderaciones de los préstamos sin pagar se establecen en $1/p$, donde p es la probabilidad de incumplimiento. Los préstamos sin pagar tienen una ponderación de 1. Las sumas de las ponderaciones para los préstamos no pagados y los préstamos pagados son aproximadamente iguales. La media de los valores pronosticados es ahora de aproximadamente un 58 % en lugar de un 0.39 %.

Hay que tener en cuenta que la ponderación proporciona una alternativa tanto al aumento de la clase más rara como a la disminución de la clase dominante.



Adaptación de la función de pérdida

Muchos algoritmos de clasificación y regresión optimizan un determinado criterio o *función de pérdida (loss function)*. Por ejemplo, la regresión logística intenta minimizar la desviación. Desde el punto de vista teórico, algunos proponen modificar la función de pérdida para evitar los problemas provocados por una clase rara. En la práctica, esto es difícil de conseguir: los algoritmos de clasificación pueden ser complejos y difíciles de modificar. La ponderación es una forma sencilla de cambiar la función de pérdida, descontando los errores de los registros con ponderaciones bajas en favor de registros con ponderaciones más altas.

Generación de datos

Una variante del sobremuestreo mediante bootstrapping es la *generación de datos (data generation)* mediante la alteración de los registros existentes para crear nuevos registros. La intuición detrás de esta idea es que, dado que observamos solo un conjunto limitado de instancias, el algoritmo no tiene un conjunto rico de información para construir "reglas" de clasificación. Al crear nuevos registros que son similares, pero no idénticos a los registros existentes, el algoritmo tiene la oportunidad de aprender un conjunto de reglas más sólido. Esta noción es similar en espíritu al conjunto de modelos estadísticos, como son boosting y bagging.

La idea ganó fuerza con la publicación del algoritmo *SMOTE*, cuyas siglas corresponden a "Synthetic Minority Oversampling Technique" (técnica de sobremuestreo de minorías sintéticas). El algoritmo SMOTE encuentra un registro que es similar al registro que se está muestreando (consultar "K-vecinos más cercanos" en la página 238) y crea un registro sintético que es un promedio ponderado aleatoriamente del registro original y del registro vecino, donde la ponderación se genera por separado para cada predictora. El número de registros sintéticos sobremuestreados que se crean depende de la proporción de sobremuestreo necesaria para que el conjunto de datos esté aproximadamente equilibrado con respecto a las clases de resultados.

Hay varias implementaciones de SMOTE en *R*. El paquete más completo para manejar datos desbalanceados es *unbalanced*. Ofrece una variedad de técnicas, incluido el algoritmo "Racing", para seleccionar el mejor método. Sin embargo, el algoritmo SMOTE es lo bastante sencillo como para que se pueda implementar directamente en *R* usando el paquete *FNN*.

El paquete de *Python* *imbalanced-learn* implementa una serie de métodos con una API que es compatible con *scikit-learn*. Proporciona varios métodos de sobremuestreo y submuestreo, así como soporte para el uso de estas técnicas con clasificadores de boosting y bagging.

Clasificación basada en los costes

En la práctica, la precisión y AUC son una forma deficiente de elegir una regla de clasificación. A menudo, se puede asignar un coste estimado a los falsos positivos frente a los falsos negativos, y es más apropiado incorporar estos costes para determinar el mejor punto de corte al clasificar 1 y 0. Por ejemplo, supongamos que el coste esperado del incumplimiento de un nuevo préstamo es *C* y el rendimiento esperado de un préstamo pagado es *R*. Entonces el rendimiento esperado de ese préstamo es:

$$\text{rendimiento esperado} = P(Y = 0) \times R + P(Y = 1) \times C$$

En lugar de simplemente etiquetar un préstamo como no pagado o pagado, o determinar la probabilidad de incumplimiento, tiene más sentido determinar si el préstamo tiene un

rendimiento esperado positivo. La probabilidad de incumplimiento prevista es un paso intermedio y debe combinarse con el valor total del préstamo para determinar la ganancia esperada, que es la métrica de planificación final del negocio. Por ejemplo, un préstamo de menor importe podría denegarse en favor de uno de mayor importe que presente una probabilidad prevista de incumplimiento ligeramente mayor.

Exploración de pronósticos

Una sola métrica, como AUC, no puede evaluar todos los aspectos de la idoneidad de un modelo para una situación. La figura 5.8 muestra las reglas de decisión para cuatro modelos diferentes que se ajustan a los datos de préstamos utilizando solo dos variables predictoras: `borrower_score` y `payment_inc_ratio`. Los modelos son el análisis discriminante lineal (LDA), la regresión lineal logística, el ajuste de regresión logística que utiliza un modelo aditivo generalizado (GAM) y el modelo de árbol (consultar “Modelos de árbol” en la página 249). La zona de la parte superior izquierda de las líneas corresponde a un incumplimiento pronosticado. Los modelos LDA y de regresión lineal logística dan resultados casi idénticos en este caso. El modelo de árbol genera la regla menos regular, con dos pasos. Finalmente, el ajuste GAM de la regresión logística representa un compromiso entre el modelo de árbol y el modelo lineal.

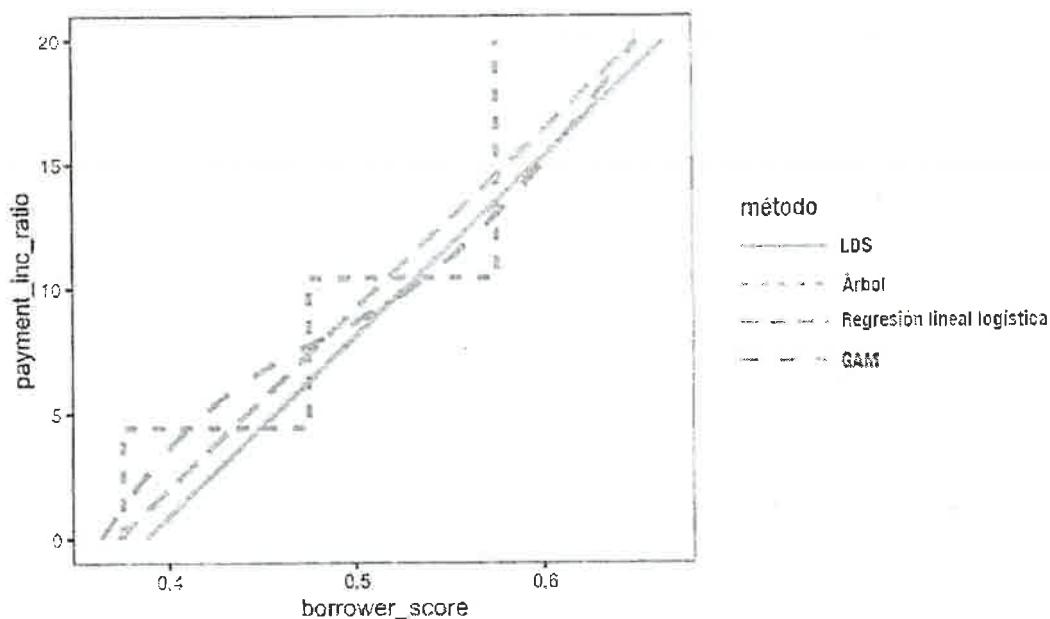


Figura 5.8 Comparación de las reglas de clasificación para cuatro métodos diferentes.

No es fácil visualizar las reglas de pronóstico para dimensiones superiores o, en el caso de GAM y el modelo de árbol, incluso generar las regiones para tales reglas.

En cualquier caso, siempre está justificado el análisis exploratorio de los valores pronosticados.

Ideas clave

- Los datos muy desequilibrados (es decir, donde los resultados interesantes, esto es, los 1, son raros) son problemáticos para los algoritmos de clasificación.
- Una estrategia para trabajar con datos que no están equilibrados es equilibrar los datos de entrenamiento submuestreando el caso mayoritario (o sobremuestreando el caso raro).
- Si el uso de todos los 1 todavía nos deja con muy pocos 1, podemos hacer bootstrap a los casos raros o usar SMOTE para crear datos sintéticos similares a los casos raros existentes.
- Los datos que no están equilibrados generalmente indican que la clasificación correcta de una clase (los 1) tiene mayor valor, y esa relación de valor debe incorporarse a la métrica de evaluación.

Lecturas complementarias

- Tom Fawcett, autor de *Data Science for Business*, tiene un buen artículo sobre clases desequilibradas (<https://www.svds.com/learning-imbalanced-classes/>).
- Para obtener más información sobre SMOTE, consultar Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall y W. Philip Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique" (<https://jair.org/index.php/jair/article/view/10302>), *Journal of Artificial Intelligence Research* 16 (2002):321-357.

Resumen

La clasificación, el proceso de pronosticar a cuál de dos o más categorías pertenece un registro, es una herramienta fundamental del análisis predictivo. ¿Habrá un préstamo que no se pagará (sí o no)? ¿Se pagará por adelantado? ¿Un visitante de la web hará clic en un enlace? ¿Comprará algo? ¿Es fraudulenta una reclamación al seguro? A menudo, en los problemas de clasificación, una clase es de primordial interés (por ejemplo, una reclamación fraudulenta al seguro), y en la clasificación binaria, esta clase se designa como 1, mientras que la otra clase, más prevalente, se designa con un 0. A menudo, una parte clave del proceso es la de estimar la *puntuación de propensión* (*propensity score*), la probabilidad de pertenecer a la clase de interés. Un escenario frecuente es aquel en el que la clase de interés es relativamente rara. Al evaluar un clasificador, hay una variedad de métricas de evaluación de modelos que van más allá de la mera precisión. Estas son importantes en la situación de las clases raras, cuando la clasificación de todos los registros como 0 puede proporcionar una alta precisión.

CAPÍTULO 6

Aprendizaje automático estadístico

Los avances recientes en estadística se han dedicado al desarrollo de técnicas automatizadas más potentes para el modelado predictivo, tanto de regresión como de clasificación. Estos métodos, como los que se discutieron en el capítulo anterior, son *métodos supervisados (supervised methods)*: se entrena con datos donde se conocen los resultados y aprenden a pronosticar los resultados con nuevos datos. Caen bajo el paraguas del aprendizaje automático estadístico y se distinguen de los métodos estadísticos clásicos en que se basan en datos y no buscan imponer una estructura lineal o de otro tipo a los datos. El método K-vecinos más cercanos, por ejemplo, es bastante simple: clasifica un registro en función de cómo se clasifican registros similares. Las técnicas de mayor éxito y más utilizadas se basan en el *aprendizaje por conjuntos (ensemble learning)* aplicado a árboles de decisión (*decision trees*). La idea básica del aprendizaje por conjuntos es utilizar muchos modelos para formular un pronóstico, en lugar de usar un solo modelo. Los árboles de decisión son una técnica flexible y automática para aprender reglas sobre las relaciones entre las variables predictoras y las variables de resultado. Afortunadamente, la combinación del aprendizaje por conjuntos con los árboles de decisión da lugar a algunas de las técnicas de modelado predictivo más eficaces.

El desarrollo de muchas de las técnicas del aprendizaje automático estadístico se remonta a los estadísticos Leo Breiman (ver figura 6.1) de la Universidad de California en Berkeley y Jerry Friedman de la Universidad de Stanford. Sus trabajos, junto con los de otros investigadores de Berkeley y Stanford, comenzaron con el desarrollo de modelos de árboles en 1984. El desarrollo posterior de métodos conjuntos de bagging y boosting en la década de 1990 establecieron las bases del aprendizaje automático estadístico.



Figura 6.1 Leo Breiman, e profesor de estadística en UC Berkeley, estuvo a la vanguardia del desarrollo de muchas técnicas aplicadas en el conjunto de herramientas que utilizan los científicos de datos en la actualidad.



El aprendizaje automático frente a la estadística

En el contexto del modelado predictivo, ¿cuál es la diferencia entre el aprendizaje automático y la estadística? No hay una línea clara que separe ambas disciplinas. El aprendizaje automático tiende a centrarse más en desarrollar algoritmos eficientes que escalen a grandes datos para optimizar el modelo predictivo. La estadística generalmente presta más atención a la teoría probabilística y la estructura subyacente del modelo. Los métodos de bagging y bosque aleatorio (consultar "Métodos de bagging y bosque aleatorio" en la página 259) tuvieron un sólido crecimiento en el campo de la estadística. El método de boosting, por otro lado, se ha desarrollado en ambas disciplinas, pero recibe más atención en el lado de la división del aprendizaje automático. Independientemente de la historia, el compromiso de boosting asegura que prosperará como técnica tanto en estadística como en aprendizaje automático.

K-vecinos más cercanos

La idea que está detrás de K-vecinos más cercanos (K-Nearest Neighbours [KNN]) es muy sencilla¹⁴. Para clasificar o pronosticar cada registro:

1. Buscamos K registros que tengan características similares (es decir, valores de pronóstico similares).
2. Para la clasificación, averiguamos cuál es la clase mayoritaria entre esos registros similares y asignamos esa clase al nuevo registro.
3. Para el pronóstico (también llamado *regresión KNN* [*KNN regression*]), encontramos el promedio entre esos registros similares y pronosticamos ese promedio para el nuevo registro.

¹⁴ Esta sección y las siguientes de este capítulo © 2020 Datastats, LLC, Peter Bruce, Andrew Bruce y Peter Gedeck; utilizado con el correspondiente permiso.

Términos clave de K-vecinos más cercanos

Vecino

- Registro que tiene valores de pronóstico similares a otro registro.

Métricas de distancia

- Medidas que suman en una sola cifra la distancia entre un registro y otro.

Estandarización

- Operación de restar la media y dividirla por la desviación estándar.

Sinónimo

normalización

Puntuación z

Valor que resulta después realizar la estandarización.

K

Número de vecinos considerados en el cálculo del vecino más cercano.

KNN es una de las técnicas de pronóstico/clasificación más sencillas: no hay ningún modelo que ajustar (como en la regresión). Esto no significa que la utilización de KNN sea un procedimiento automático. Los resultados del pronóstico dependen de cómo se escalen las características, cómo se mida la similitud y de lo grande que se establezca K . Además, todos los valores de las predictoras deben estar en formato numérico. Ilustraremos cómo utilizar el método KNN con un ejemplo de clasificación.

Un pequeño ejemplo: pronóstico del incumplimiento de préstamos

La tabla 6.1 muestra algunos registros de datos de préstamos personales de LendingClub. LendingClub es líder en préstamos entre particulares en el que los grupos de inversores otorgan préstamos personales a particulares. El objetivo del análisis sería pronosticar el resultado de un potencial nuevo préstamo: pagado frente a no pagado.

Tabla 6.1 Algunos registros y columnas de los datos de préstamos de LendingClub

Resultado	Total préstamo	Ingresos	Propósito	Años de empleo	Prop. vivienda	Estado
Pagado	10 000	79 100	consolidación de deuda	11	HIPOTECA	NV
Pagado	9600	48 000	traslado	5	HIPOTECA	TN
Pagado	18 800	120 036	consolidación de deuda	11	HIPOTECA	MD
No pagado	15 250	232 000	pequeño negocio	9	HIPOTECA	CA
Pagado	17 050	35 000	consolidación de deuda	4	ALQUILER	MD
Pagado	5500	43 000	consolidación de deuda	4	ALQUILER	KS

Consideremos un modelo muy simple con solo dos variables predictoras: `dti`, que es la razón entre el pago de las deudas (excluyendo la hipoteca) y los ingresos, y `payment_inc_ratio`, que es la razón entre el pago del préstamo y los ingresos. Ambas razones se multiplican por 100. Utilizando un pequeño conjunto de 200 préstamos, `loan200`, con resultados binarios conocidos (no pagado o pagado, especificados en la predictora `outcome200`), y con K establecido en 20, la estimación de KNN para un nuevo préstamo a pronosticar, `newloan`, con `dti=22.5` y `payment_inc_ratio=9` se puede calcular en *R* de la siguiente manera¹⁵:

```
newloan <- loan200[1, 2:3, drop=FALSE]
knn_pred <- knn(train=loan200[-1, 2:3], test=newloan, cl=loan200[-1, 1], k=20)
knn_pred == 'paid off'
[1] TRUE
```

El pronóstico de KNN es que el préstamo se pagará.

Si bien *R* tiene una función `knn` nativa, el paquete FNN que aporta *R*, para el método de vecino más cercano rápido (<https://cran.r-project.org/web/packages/FNN/FNN.pdf>), escala de manera más efectiva a grandes cantidades de datos y brinda más flexibilidad.

El paquete scikit-learn proporciona una implementación rápida y eficiente de KNN en *Python*:

```
predictors = ['payment_inc_ratio', 'dti']
outcome = 'outcome'

newloan = loan200.loc[0:0, predictors]
X = loan200.loc[1:, predictors]
y = loan200.loc[1:, outcome]

knn = KNeighborsClassifier(n_neighbors=20)
knn.fit(X, y)
knn.predict(newloan)
```

La figura 6.2 ofrece una representación visual de este ejemplo. El nuevo préstamo que se pronostica está identificado con una cruz. Los cuadrados (pagados) y los círculos (no pagados) son los datos de entrenamiento. El círculo grande definido por una circunferencia de color negro muestra el límite de los 20 puntos más cercanos. En este caso, dentro del círculo se encuentran 9 préstamos no pagados, en comparación con 11 préstamos pagados. Por tanto, el resultado previsto del préstamo es que se paga. Hay que tener en cuenta que si consideramos solamente a los tres vecinos más cercanos, el pronóstico sería que el préstamo incumple.

¹⁵ Para este ejemplo, extraemos la primera fila del conjunto de datos `loan200` como `newloan` y la excluimos del conjunto de datos para el entrenamiento.

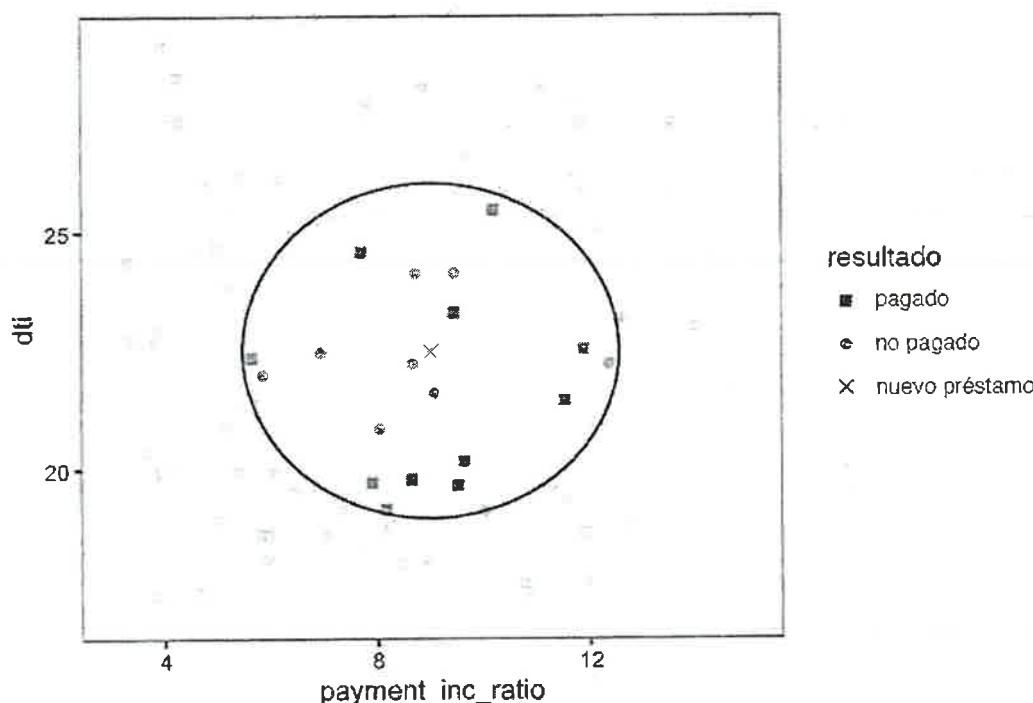


Figura 6.2 Pronóstico KNN del incumplimiento de los préstamos utilizando dos variables: la relación deuda-ingresos (*debt-to-income*) y la relación pago del préstamo-ingresos (*loan-payment-to-income*).



Si bien el resultado KNN para la clasificación suele ser una decisión binaria, como son el incumplimiento o el pago como dato del préstamo, las rutinas de KNN suelen ofrecer la oportunidad de generar una probabilidad (propensión) entre 0 y 1. La probabilidad se basa en la fracción de una clase en los K-vecinos más cercanos. En el ejemplo anterior, esta probabilidad de incumplimiento se habría estimado en 9/20 o 0.45.

El uso de una puntuación de probabilidad nos permite utilizar reglas de clasificación distintas de los votos por mayoría simple (probabilidad de 0.5). Esto es especialmente importante en problemas con clases desequilibradas. Consultar “Estrategias para datos que no están equilibrados” en la página 230. Por ejemplo, si el objetivo es identificar miembros de una clase poco común, el corte normalmente se establecería por debajo del 50 %. Un planteamiento habitual es establecer el corte en la probabilidad del evento raro.

Métricas de distancia

La similitud (cercanía) se determina usando una métrica de distancia, que es una función que mide lo separados que están dos registros (x_1, x_2, \dots, x_p) y (u_1, u_2, \dots, u_p) entre sí. La métrica de distancia más popular entre dos vectores es la distancia euclídea (*Euclidean*

distance). Para medir la distancia euclíadiana entre dos vectores, restamos uno del otro, elevamos al cuadrado las diferencias, las sumamos y extraemos la raíz cuadrada:

$$\sqrt{(x_1 - u_1)^2 + (x_2 - u_2)^2 + \cdots + (x_p - u_p)^2}.$$

Otra métrica de distancia habitual para datos numéricos es la distancia Manhattan (*Manhattan distance*):

$$|x_1 - u_1| + |x_2 - u_2| + \cdots + |x_p - u_p|$$

La distancia euclíadiana corresponde a la distancia en línea recta entre dos puntos (por ejemplo, a vuelo de pájaro). La distancia Manhattan es la distancia entre dos puntos recorrida de una vez en una sola dirección (por ejemplo, recorriendo el trayecto rectangular que determinan las manzanas de la ciudad). Por esta razón, la distancia Manhattan es una aproximación conveniente si la similitud se define como el tiempo de trayecto de un punto a otro.

Al medir la distancia entre dos vectores, las variables (características) que se miden con una escala comparativamente grande dominarán la medida. Por ejemplo, para los datos de préstamos, la distancia sería casi exclusivamente una función de las variables de ingresos e importe del préstamo, que se miden por decenas o cientos de miles. Comparativamente, las variables de razón no contarían prácticamente nada. Abordamos este problema estandarizando los datos. Consultar “Estandarización (normalización, puntuación z)” en la página 243.



Otras métricas de distancias

Existen muchas otras métricas para medir la distancia entre vectores. Para datos numéricos, la *distancia de Mahalanobis* (*Mahalanobis distance*) es atractiva ya que informa de la correlación entre dos variables. Este hecho es conveniente ya que si dos variables están altamente correlacionadas, Mahalanobis las tratará esencialmente como una sola variable en términos de distancia. La distancia euclíadiana y de Manhattan no tienen en cuenta la correlación, lo que otorga mayor ponderación al atributo que subyace en esas características. La distancia de Mahalanobis es la distancia euclíadiana entre los componentes principales (consultar “Análisis de componentes principales” en la página 284). La desventaja de usar la distancia de Mahalanobis es un mayor esfuerzo y complejidad de cálculo. Se calcula utilizando la matriz de covarianza (*covariance matrix*) (consultar “Matriz de covarianza” en la página 202).

Codificador One-Hot

Los datos de préstamos en la tabla 6.1 incluyen varias variables de tipo factor (cadena). La mayoría de los modelos estadísticos y de aprendizaje automático requieren que este tipo de variable se convierta en una serie de variables ficticias binarias que contienen la misma

información, como en la tabla 6.2. En lugar de una sola variable que denote el estado del ocupante de la vivienda como "propietario con una hipoteca", "propietario sin hipoteca", "alquila" u "otro", acabamos con cuatro variables binarias. La primera sería "propietario con hipoteca, S/N", la segunda sería "propietario sin hipoteca, S/N", y así sucesivamente. Esta predictor, el estado del ocupante de la vivienda, crea un vector con un 1 y tres 0 que se puede utilizar en algoritmos estadísticos y de aprendizaje automático. La frase *One-Hot encoding* proviene de la terminología de circuitos digitales, en la que describe configuraciones de circuitos en las que solo se permite que un bit sea positivo (Hot).

Tabla 6.2 Representación de los datos del tipo factor de la propiedad de la vivienda de la tabla 6.1 como una variable ficticia numérica

PROPIETARIO CON HIPOTECA	PROPIETARIO SIN HIPOTECA	OTRO	ALQUILER
1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	0
0	0	0	1
0	0	0	1



En regresión lineal y logística, la codificación One-Hot causa problemas con la multicolinealidad. Consultar "Multicolinealidad" en la página 172. En tales casos, se omite una variable ficticia (su valor puede inferirse de los otros valores). KNN y otros métodos que se tratan en este libro no tienen este problema.

Estandarización (normalización, puntuación z)

En la medición, a menudo no nos interesa tanto "cuánto" sino "lo diferente que es del promedio". La estandarización, también llamada *normalización (normalization)*, coloca todas las variables en escalas similares restando la media y dividiendo entre la desviación estándar. De esta forma, aseguramos que una variable no influya demasiado en un modelo simplemente por la escala de su medida original:

$$z = \frac{x - \bar{x}}{s}$$

Al resultado de esta transformación se le denomina normalmente puntuación z (*z-score*). Las mediciones entonces se expresan en términos de "desviaciones estándar de la media".



La normalización (*normalization*) en este contexto estadístico no debe confundirse con la normalización de la base de datos (*database normalization*), que es la eliminación de datos redundantes y la verificación de las dependencias de los datos.

Estadística práctica para ciencia de datos con R y Python

Para KNN y para algunos otros procedimientos (por ejemplo, análisis de componentes principales y agrupamiento), es esencial considerar la estandarización de los datos antes de aplicar el procedimiento. Para ilustrar esta idea, KNN se aplica a los datos de préstamos usando `dti` y `payment_inc_ratio` (consultar “Un pequeño ejemplo: pronóstico del incumplimiento del préstamo” en la página 239) más otras dos variables: `revol_bal`, el crédito renovable total disponible para el solicitante en dólares y `revol_util`, el porcentaje del crédito que se utiliza. El nuevo registro a pronosticar se muestra a continuación:

```
newloan
  payment_inc_ratio    dti    revol_bal    revol_util
1          2.3932      1       1687         9.4
```

La magnitud de `revol_bal`, que está en dólares, es mucho mayor que la de las otras variables. La función `knn` devuelve el índice de los vecinos más cercanos como un atributo `nn.index`, y esto se puede usar para mostrar las primeras cinco filas más cercanas en `loan_df`:

```
loan_df <- model.matrix(~ -1 + payment_inc_ratio + dti + revol_bal +
                         revol_util, data=loan_data)
newloan <- loan_df[1, , drop=FALSE]
loan_df <- loan_df[-1,]
outcome <- loan_data[-1, 1]
knn_pred <- knn(train=loan_df, test=newloan, cl=outcome, k=5)
loan_df[attr(knn_pred, "nn.index"),]

            payment_inc_ratio    dti    revol_bal    revol_util
35537          1.47212   1.46     1686        10.0
33652          3.38178   6.37     1688        8.4
25864          2.36303   1.39     1691        3.5
42954          1.28160   7.14     1684        3.9
43600          4.12244   8.98     1684        7.2
```

Tras el ajuste del modelo, podemos usar el método `kneighbors` para identificar las cinco filas más cercanas en el conjunto de entrenamiento con `scikit-learn`:

```
predictors = ['payment_inc_ratio', 'dti', 'revol_bal', 'revol_util']
outcome = 'outcome'

newloan = loan_data.loc[0:0, predictors]
X = loan_data.loc[1:, predictors]
y = loan_data.loc[1:, outcome]

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X, y)

nbrs = knn.kneighbors(newloan)
X.iloc[nbrs[1][0], :]
```

El valor de `revol_bal` en estos vecinos está muy cerca de su valor en el nuevo registro, pero las otras variables predictoras están por todo el mapa y esencialmente no juegan ningún papel en la determinación de los vecinos.

Comparemos esto con KNN aplicado a los datos estandarizados usando la función `scale` de R, que calcula la puntuación z para cada variable:

```
loan_df <- model.matrix(~ -1 + payment_inc_ratio + dti + revol_bal +
                         revol_util, data=loan_data)
loan_std <- scale(loan_df)
newloan_std <- loan_std[1, , drop=FALSE]
loan_std <- loan_std[-1, ]
loan_df <- loan_df[-1, ] ❶
outcome <- loan_data[-1, 1]
knn_pred <- knn(train=loan_std, test=newloan_std, cl=outcome, k=5)
loan_df[attr(knn_pred, "nn.index")]
  payment_inc_ratio      dti    revol_bal    revol_util
2081           2.61091     1.03      1218       9.7
1439           2.34343     0.51       278       9.9
30216          2.71200     1.34      1075       8.5
28543          2.39760     0.74      2917       7.4
44738          2.34309     1.37       488       7.2
```

- ❶ Necesitamos eliminar también la primera fila de `loan_df`, de modo que los números de fila se correspondan entre sí.

El método `sklearn.preprocessing.StandardScaler` se entrena primero con las predictoras y posteriormente se utiliza para transformar el conjunto de datos antes de entrenar el modelo KNN:

```
newloan = loan_data.loc[0:0, predictors]
X = loan_data.loc[1:, predictors]
y = loan_data.loc[1:, outcome]

scaler = preprocessing.StandardScaler()
scaler.fit(X * 1.0)

X_std = scaler.transform(X * 1.0)
newloan_std = scaler.transform(newloan * 1.0)

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_std, y)

nbrs = knn.kneighbors(newloan_std)
X.iloc[nbrs[1][0], :]
```

Los cinco vecinos más cercanos son mucho más parecidos en todas las variables, proporcionando un resultado más sensible. Hay que tener en cuenta que los resultados se muestran en la escala original, pero se ha aplicado KNN a los datos escalados y al nuevo préstamo que se ha pronosticado.



El uso de la puntuación z es solo una forma de cambiar la escala de las variables. En lugar de la media, se podría utilizar una estimación más sólida de la localización, como la mediana. Asimismo, se podría utilizar una estimación de escala diferente, como el rango intercuartil, en lugar de la desviación estándar. A veces, las variables se "aplastan" en el rango 0-1. También es importante darse cuenta de que escalar cada variable para tener varianza unitaria es algo arbitrario. Esto implica que se piensa que cada variable tiene la misma importancia en la capacidad predictiva. Si tenemos un conocimiento subjetivo de que algunas variables son más importantes que otras, entonces estas podrían ampliarse. Por ejemplo, con los datos de los préstamos, es razonable esperar que la relación entre pagos e ingresos sea muy importante.



La normalización (estandarización) no cambia la forma de distribución de los datos, no le da una forma normal si no tenía ya una forma normal (consultar "Distribución normal" en la página 69).

Elección de K

La elección de K es muy importante para el rendimiento de KNN. La opción más simple es establecer $K = 1$, conocido como el clasificador 1-vecino más cercano. El pronóstico es intuitivo: se basa en encontrar en el conjunto de entrenamiento el registro de datos más similar al nuevo registro que se va a pronosticar. Establecer $K = 1$ rara vez es la mejor opción. Casi siempre obtendremos un rendimiento superior al utilizar $K > 1$ vecinos más cercanos.

En términos generales, si K es demasiado bajo, es posible que estemos sobreajustando: incluimos el ruido en los datos. Valores más altos de K proporcionan un suavizado que reduce el riesgo de sobreajuste en los datos de entrenamiento. Por otro lado, si K es demasiado alto, podemos suavizar demasiado los datos y perder la capacidad de KNN para capturar la estructura local en los datos, una de sus principales ventajas.

El valor de K que obtiene un mejor equilibrio entre el sobreajuste y el sobrealizado suele determinarse mediante métricas de precisión y, en particular, la precisión con datos de validación o reserva. No existe una regla general sobre el mejor valor de K , depende en gran medida de la naturaleza de los datos. Para datos muy estructurados con poco ruido, valores más pequeños de K funcionan mejor. Tomando prestado un término del colectivo de procesamiento de señales, este tipo de datos a veces se dice que tienen una alta relación señal/ruido (SNR) (*signal-to-noise ratio [SNR]*). Ejemplos de datos con una SNR típicamente alta son los conjuntos de datos correspondientes a la escritura a mano y los de reconocimiento de voz. Para datos ruidosos con menos estructura (datos con una SNR baja), como los datos de préstamos, son apropiados valores mayores de K . Por lo general, los valores de K están dentro del rango de 1 a 20. A menudo, se elige un número impar para evitar que se produzcan empates.



Equilibrio entre sesgo y varianza

La tensión entre el sobrerealizado y el sobreajuste es un ejemplo del *equilibrio entre sesgo y varianza (bias-variance trade-off)*, un problema omnipresente en el ajuste de modelos estadísticos. La varianza se refiere al error de modelado que se produce debido a la elección de los datos de entrenamiento. Es decir, si tuviéramos que elegir un conjunto diferente de datos de entrenamiento, el modelo resultante sería diferente. Sesgo se refiere al error de modelado que se produce porque no ha identificado correctamente el escenario del mundo real subyacente. Este error no desaparecería si solamente agregáramos más datos de entrenamiento. Cuando un modelo flexible está sobreajustado, la varianza aumenta. Podemos reducirla utilizando un modelo más simple, pero el sesgo puede aumentar debido a la pérdida de flexibilidad en el modelado de la situación subyacente real. Un enfoque general para manejar este equilibrio es a través de la validación cruzada (*cross-validation*). Consultar “Validación cruzada” en la página 155 para ampliar detalles.

KNN como motor de características

KNN se ha hecho más popular debido a su simplicidad y naturaleza intuitiva. En términos de rendimiento, KNN por sí solo no suele competir con técnicas de clasificación más sofisticadas. Sin embargo, en el ajuste práctico del modelo, KNN se puede utilizar para agregar “conocimiento local” en un proceso por etapas con otras técnicas de clasificación:

1. KNN se ejecuta sobre los datos y, para cada registro, se deriva una clasificación (o quasi-probabilidad de una clase).
2. Ese resultado se agrega como una nueva característica al registro, y luego se ejecuta otro método de clasificación sobre los datos. Por tanto, las variables predictoras originales se utilizan dos veces.

Al principio, podríamos preguntarnos si este proceso, dado que utiliza algunas predictoras dos veces, causa un problema de multicolinealidad (consultar “Multicolinealidad” en la página 172). Esto no constituye un problema, ya que la información que se incorpora al modelo de la segunda etapa es altamente local, se deriva solo de unos pocos registros cercanos y, por lo tanto, es información adicional y no redundante.



Podemos pensar en este uso por etapas de KNN como una forma de aprendizaje por conjuntos en el que se utilizan múltiples métodos de modelado predictivo en conjunto. También se puede considerar como una forma de ingeniería de características en la que el objetivo es derivar características (variables predictoras) que tienen capacidad predictiva. A menudo, esto implica una revisión manual de los datos. KNN ofrece una forma bastante automática de hacerlo.

Estadística práctica para ciencia de datos con R y Python

Por ejemplo, consideremos los datos sobre las viviendas del condado de King. Al fijar el precio de una vivienda para la venta, un agente de bienes raíces basará el precio en viviendas similares vendidas recientemente, conocidas como "comps". En esencia, los agentes inmobiliarios están haciendo una versión manual de KNN: al observar los precios de venta de viviendas similares, pueden estimar por cuánto se venderá una vivienda. Podemos crear una nueva característica para un modelo estadístico que imite al profesional inmobiliario aplicando KNN a las ventas recientes. El valor pronosticado es el precio de venta y las variables predictoras existentes podrían incluir la ubicación, el total de pies cuadrados, el tipo de estructura, el tamaño del lote y la cantidad de dormitorios y baños. La nueva variable predictora (característica) que agregamos a través de KNN es la predictora KNN para cada registro (análoga a las composiciones de los agentes inmobiliarios). Dado que estamos pronosticando un valor numérico, se usa el promedio de los K-vecinos más cercanos en lugar del voto mayoritario (conocido como regresión KNN [*KNN regression*]).

De manera similar, para los datos de préstamos, podemos crear características que representen diferentes aspectos del proceso de préstamo. Por ejemplo, el siguiente código R crearía una característica que representa la solvencia de un prestatario:

```
borrow_df <- model.matrix(~ -1 + dti + revol_bal + revol_util + open_acc +
                           delinq_2yrs_zero + pub_rec_zero, data=loan_data)
borrow_knn <- knn(borrow_df, test=borrow_df, cl=loan_data[, 'outcome'],
                   prob=TRUE, k=20)
prob <- attr(borrow_knn, "prob")
borrow_feature <- ifelse(borrow_knn == 'default', prob, 1 - prob)
summary(borrow_feature)
  Min. 1st Qu. Median Mean 3rd Qu. Max.
 0.000 0.400 0.500 0.501 0.600 0.950
```

Con `scikit-learn`, usamos el método `predict_proba` del modelo entrenado para obtener las probabilidades:

```
predictors = ['dti', 'revol_bal', 'revol_util', 'open_acc',
              'delinq_2yrs_zero', 'pub_rec_zero']
outcome = 'outcome'

X = loan_data[predictors]
y = loan_data[outcome]

knn = KNeighborsClassifier(n_neighbors=20)
knn.fit(X, y)

loan_data['borrower_score'] = knn.predict_proba(X)[:, 1]
loan_data['borrower_score'].describe()
```

El resultado es una característica que predice la probabilidad de que un prestatario incurra en incumplimiento de acuerdo con su historial crediticio.

Ideas clave

- K-vecinos cercanos (KNN) clasifica un registro asignándolo a la clase a la que pertenecen registros similares.
- La similitud (distancia) está determinada por la distancia euclídea u otras métricas relacionadas.
- El número de vecinos más cercanos con los que comparar un registro, K , está determinado por el rendimiento del algoritmo en los datos de entrenamiento, utilizando diferentes valores para K .
- Normalmente, las variables predictoras están estandarizadas para que las variables de gran escala no dominen la métrica de distancia.
- KNN se utiliza a menudo como una primera etapa en el modelado predictivo, y el valor pronosticado se agrega nuevamente a los datos como un predictor (*predictor*) para el modelado de segunda etapa (no KNN).

Modelos de árbol

Los modelos de árbol, también llamados *árboles de clasificación y regresión* (CART)¹⁶ (*Classification and Regression Trees [CART]*), *árboles de decisión* (*decision trees*), o simplemente *árboles* (*trees*), son un método de clasificación (y regresión) eficaz y de uso generalizado, desarrollado inicialmente por Leo Breiman y otros en 1984. Los modelos de árbol y sus descendientes más capaces, los *bosques aleatorios* (*random forests*) y los *árboles potenciados* (*boosted trees*) (consultar “Métodos de bagging y bosque aleatorio” en la página 259 y “Boosting” en la página 270) constituyen la base de las herramientas de modelado predictivo más utilizadas y capaces en ciencia de datos para regresión y clasificación.

Términos clave de los modelos de árbol

Partición recursiva

Separamos y volvemos a separar repetidamente los datos con el objetivo de hacer que los resultados en cada subdivisión final sean lo más homogéneos posible.

Valor empleado para la división

Valor predictor que separa los registros en aquellos en los que ese valor predictor es menor que el valor predictor empleado en la división y aquellos en los que es mayor.

Nodo

En un árbol de decisión, o en el conjunto de reglas de ramificación correspondientes, un nodo es la representación gráfica o de las reglas del valor empleado para la división.

¹⁶ El término CART es una marca registrada de Salford Systems relacionada con su implementación específica de modelos de árboles.

Hoja

El final de un conjunto de reglas si-entonces, o ramas de un árbol: las reglas que nos llevan a la hoja proporcionan una de las reglas de clasificación para cualquier registro del árbol.

Pérdida

Número de clasificaciones erróneas en una etapa del proceso de separación.
Cuantas más clasificaciones erróneas, más impurezas.

Impureza

Grado en el que una mezcla de clases se encuentra en una subpartición de los datos (cuanto más mezclada, más impura).

Sinónimo

heterogeneidad

Antónimos

homogeneidad, pureza

Poda

Proceso de tomar un árbol completamente desarrollado y cortar progresivamente sus ramas para reducir el sobreajuste.

Un modelo de árbol es un conjunto de reglas "si-entonces-sino" que son fáciles de entender e implementar. A diferencia de la regresión lineal y logística, los modelos de árbol tienen la capacidad de descubrir patrones ocultos correspondientes a interacciones complejas en los datos. Sin embargo, a diferencia de KNN o Bayes ingenuo, los modelos de árbol simples se pueden expresar en términos de relaciones predictoras que son fácilmente interpretables.



Árboles de decisión en la investigación de operaciones

El término *árboles de decisión* (*decision trees*) tiene un significado diferente (y más antiguo) en la ciencia de la decisión y la investigación de operaciones, donde se refiere al proceso de análisis de decisiones de los individuos. En este sentido, los puntos de decisión, los posibles resultados y sus probabilidades estimadas se presentan en un diagrama de ramificación, y se elige la ruta de decisión con el valor máximo esperado.

Un ejemplo sencillo

Los dos principales paquetes para ajustar los modelos de árbol en R son `rpart` y `tree`. Con el paquete `rpart`, se ajusta el modelo a una muestra de 3000 registros de los datos de préstamos utilizando las variables `payment_inc_ratio` y `borrower_score` (consultar "K vecinos más cercanos" en la página 238 para obtener una descripción de los datos):

```
library(rpart)
loan_tree <- rpart(outcome ~ borrower_score + payment_inc_ratio,
                   data=loan3000, control=rpart.control(cp=0.005))
plot(loan_tree, uniform=TRUE, margin=0.05)
text(loan_tree)
```

`Sklearn.tree.DecisionTreeClassifier` proporciona la implementación de un árbol de decisiones. El paquete `dmab` proporciona una función de conveniencia para crear una visualización dentro de un Jupyter Notebook:

```
predictors = ['borrower_score', 'payment_inc_ratio']
outcome = 'outcome'

X = loan3000[predictors]
y = loan3000[outcome]

loan_tree = DecisionTreeClassifier(random_state=1, criterion='entropy',
                                   min_impurity_decrease=0.003)
loan_tree.fit(X, y)
plotDecisionTree(loan_tree, feature_names=predictors,
                  class_names=loan_tree.classes_)
```

El árbol resultante se muestra en la figura 6.3. Debido a las diferentes implementaciones, encontraremos que los resultados de *R* y *Python* no son idénticos. Esta circunstancia se esperaba. Estas reglas de clasificación se determinan a través de un árbol jerárquico, comenzando en la raíz y moviéndose hacia la izquierda si el nodo es verdadero y hacia la derecha si no lo es, hasta que se alcanza una hoja.

Por lo general, el árbol se traza invertido, por lo que la raíz está en la parte superior y las hojas en la parte inferior. Por ejemplo, si obtenemos un préstamo con un `borrower_score` de 0.6 y un `payment_inc_ratio` de 8.0, terminamos en la hoja más a la izquierda y pronosticamos que el préstamo se pagará.

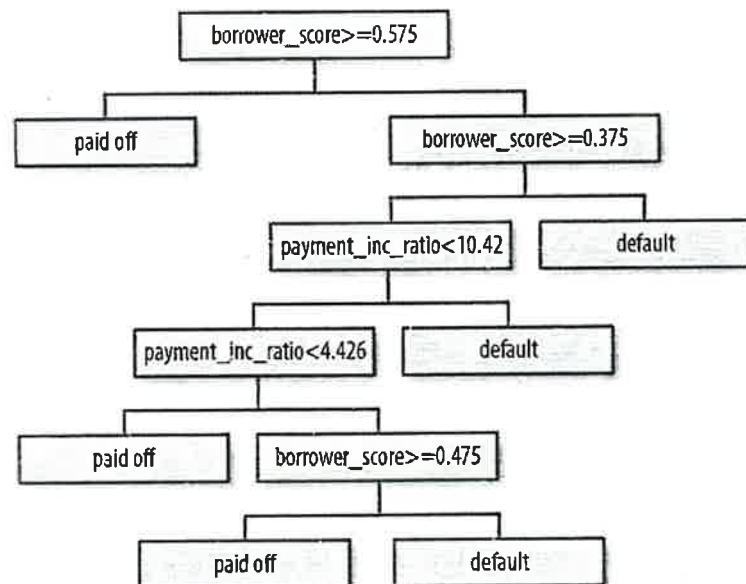


Figura 6.3 Reglas para un modelo de árbol sencillo que se ajustan a los datos de préstamos.

También se puede generar fácilmente en R una versión impresa del árbol:

```
loan_tree
n= 3000

node), split, n, loss, yval, (yprob)
  * denotes terminal node

1) root 3000 1445 paid off (0.5183333 0.4816667)
  2) borrower_score>=0.575 878 261 paid off (0.7027335 0.2972665) *
  3) borrower_score< 0.575 2122 938 default (0.4420358 0.5579642)
    6) borrower_score>=0.375 1639 802 default (0.4893228 0.5106772)
      12) payment_inc_ratio< 10.42265 1157 547 paid off (0.5272256 0.4727744)
        24) payment_inc_ratio< 4.42601 334 139 paid off (0.5838323 0.4161677) *
        25) payment_inc_ratio>=4.42601 823 408 paid off (0.5042527 0.4957473)
          50) borrower_score>=0.475 418 190 paid off (0.5454545 0.4545455) *
          51) borrower_score< 0.475 405 187 default (0.4617284 0.5382716) *
    13) payment_inc_ratio>=10.42265 482 192 default (0.3983402 0.6016598) *
    7) borrower_score< 0.375 483 136 default (0.2815735 0.7184265) *
```

La profundidad del árbol se muestra con el sangrado. Cada nodo corresponde a una clasificación provisional determinada por el resultado predominante en esa partición. La "pérdida" es el número de clasificaciones erróneas producidas por la clasificación provisional en una partición. Por ejemplo, en el nodo 2, ha habido 261 clasificaciones erróneas de un total de 878 registros. Los valores entre paréntesis corresponden a la proporción de registros de pagados o no pagados, respectivamente. Por ejemplo, en el nodo 13, que pronostica el incumplimiento, más del 60 por ciento de los registros son préstamos que están en incumplimiento.

La documentación de `scikit-learn` describe cómo crear una representación de texto de un modelo de árbol de decisión. Incluimos una función de conveniencia en el paquete `dmbs`:

```
print(textDecisionTree(loan_tree))

node=0 test node: go to node 1 if 0 <= 0.5750000178813934 else to node 6
  node=1 test node: go to node 2 if 0 <= 0.32500000298023224 else to node 3
    node=2 leaf node: [[0.785, 0.215]]
  node=3 test node: go to node 4 if 1 <= 10.42264986038208 else to node 5
    node=4 leaf node: [[0.488, 0.512]]
    node=5 leaf node: [[0.613, 0.387]]
  node=6 test node: go to node 7 if 1 <= 9.19082498550415 else to node 10
    node=7 test node: go to node 8 if 0 <= 0.7249999940395355 else to node 9
      node=8 leaf node: [[0.247, 0.753]]
      node=9 leaf node: [[0.073, 0.927]]
    node=10 leaf node: [[0.457, 0.543]]
```

Algoritmo de partición recursiva

El algoritmo para construir un árbol de decisión, llamado *partición recursiva (recursive partitioning)*, es sencillo e intuitivo. Los datos se someten a partición repetidamente utilizando valores predictivos que hacen lo mejor posible el trabajo de separar los datos

en particiones relativamente homogéneas. La figura 6.4 muestra las particiones creadas para el árbol de la figura 6.3. La primera regla, representada por la regla 1, es $\text{borrower_score} \geq 0.575$ y segmenta la parte derecha de la gráfica. La segunda regla es $\text{borrower_score} < 0.375$ y segmenta la parte izquierda.

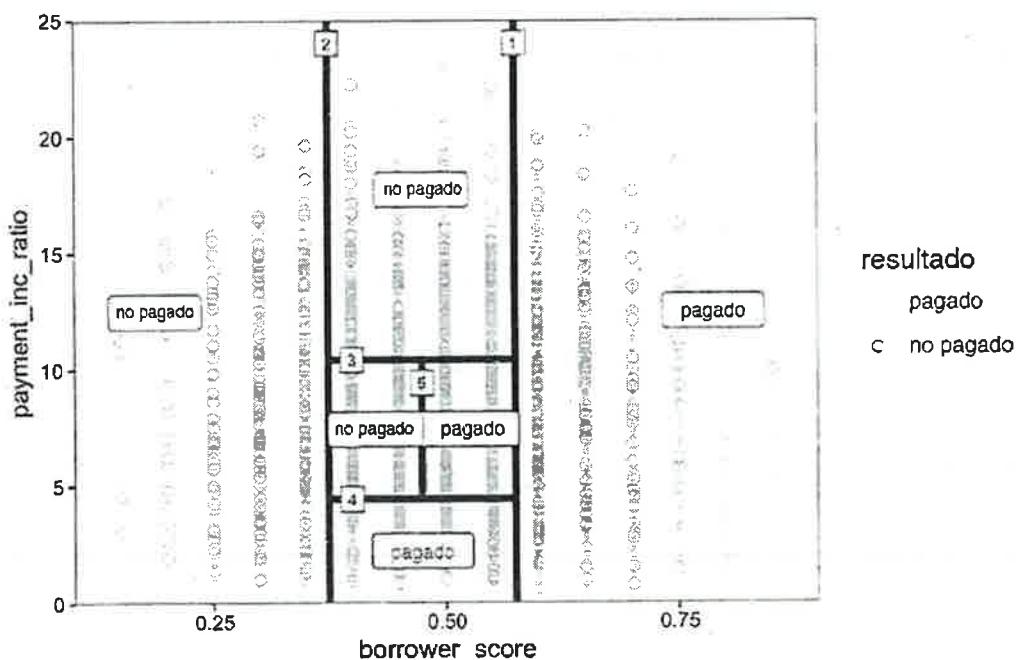


Figura 6.4 Primeras tres reglas para un modelo simple de árbol que se ajustan a los datos de préstamos.

Supongamos que tenemos una variable de respuesta Y y un conjunto de P variables predictoras X_j para $j = 1, \dots, P$. Para una partición A de registros, la partición recursiva encontrará la mejor manera de dividir A en dos subparticiones:

1. Para cada variable predictora X_j :
 - a. Para cada valor s_j de X_j :
 - i. Dividimos los registros en A con valores $X_j < s_j$ formando una partición, y los registros restantes donde $X_j \geq s_j$ forman otra partición.
 - ii. Medimos la homogeneidad de las clases dentro de cada subpartición de A .
 - b. Seleccionamos el valor de s_j que produzca la máxima homogeneidad de clase dentro de la partición.
2. Seleccionamos la variable X_j y el valor empleado para la división s_j que produce la máxima homogeneidad de clase dentro de la partición.

Ahora viene la parte recursiva:

1. Inicializamos A con todo el conjunto de datos.
2. Aplicarnos el algoritmo de partición para dividir A en dos subparticiones: A_1 y A_2 .
3. Repetimos el paso 2 en las subparticiones A_1 y A_2 .
4. El algoritmo termina cuando no se pueden realizar más particiones que mejoren suficientemente la homogeneidad de las particiones.

El resultado final es una partición de los datos, como se muestra en la figura 6.4, excepto en las dimensiones P , donde cada partición pronostica un resultado de 0 o 1 dependiendo del voto mayoritario de la respuesta en esa partición.



Además de un pronóstico binario 0/1, los modelos de árbol pueden producir una estimación de probabilidad basada en el número de 0 y 1 en la partición. La estimación es simplemente la suma de 0 o 1 en la partición dividida por el número de observaciones en la partición:

$$\text{Prob}(Y = 1) = \frac{\text{Número de } 1\text{s en la partición}}{\text{Tamaño de la partición}}$$

La $\text{Prob}(Y = 1)$ estimada se puede convertir en una decisión binaria. Por ejemplo, establecemos la estimación en 1 si $\text{Prob}(Y = 1) > 0.5$.

Medición de la homogeneidad o la impureza

Los modelos de árbol crean de forma recursiva particiones (conjuntos de registros), A , que predicen un resultado de $Y = 0$ o $Y = 1$. Podemos ver en el algoritmo anterior que necesitamos una forma de medir la homogeneidad, también llamada *pureza de clase* (*class purity*), en una partición. O de manera equivalente, necesitamos medir la impureza de una partición. La precisión de los pronósticos es la proporción p de registros mal clasificados dentro de esa partición, que varía de 0 (perfecta) a 0.5 (acierto puramente aleatorio).

Pero resulta que la precisión no es una buena medida de la impureza. En cambio, dos tipos de medida habituales de la impureza son la *impureza de Gini* (*Gini impurity*) y la *entropía de la información* (*entropy of information*). Si bien estas (y otras) medidas de impureza se aplican a problemas de clasificación con más de dos clases, nos centramos en el caso binario. La impureza de Gini para un conjunto de registros A es:

$$I(A) = p(1 - p)$$

La medida de la entropía viene dada por:

$$I(A) = -p \log_2(p) - (1 - p) \log_2(1 - p)$$

La figura 6.5 muestra que las medidas de la impureza (reajustada) de Gini y la entropía son similares, en la que la entropía alcanza puntuaciones de impureza más altas para tasas de precisión moderadas y altas.

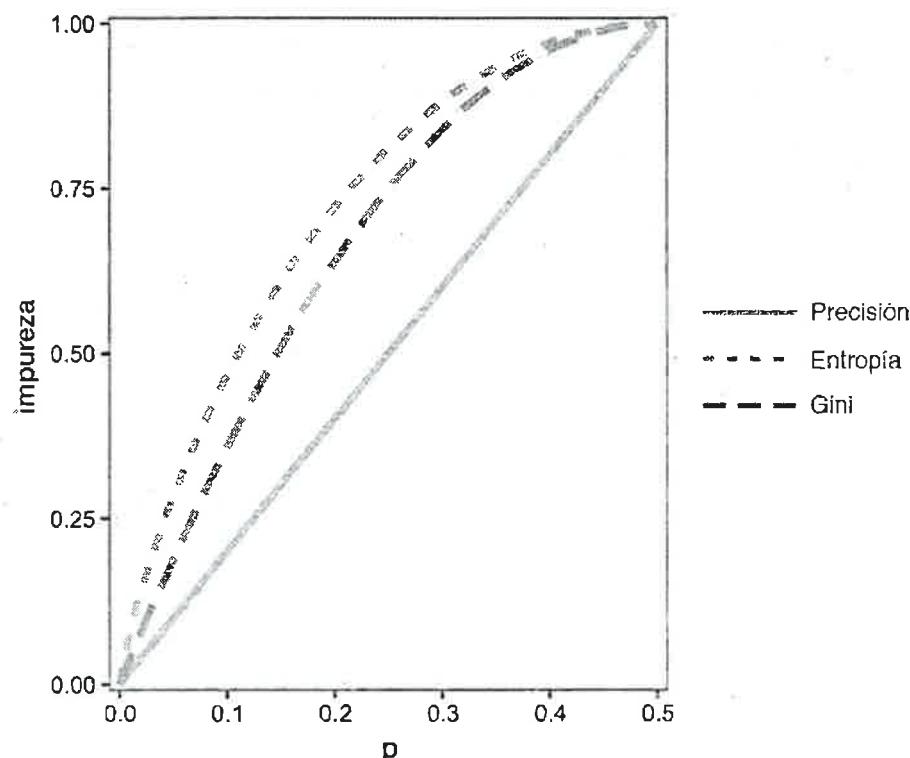


Figura 6.5 Medidas de la impureza y la entropía de Gini.



Coeficiente de Gini

La impureza de Gini no debe confundirse con el coeficiente de Gini (*Gini coefficient*). Representan conceptos similares, pero el coeficiente de Gini se limita al problema de clasificación binaria y está relacionado con la métrica AUC (consultar “AUC” en la página 226).

La métrica de impureza se utiliza en el algoritmo de división descrito anteriormente. Para cada partición propuesta de los datos, se mide la impureza para cada una de las particiones que resultan de la división. Luego se calcula el promedio ponderado y se selecciona la partición (en cada etapa) que arroje el promedio ponderado más bajo.

Detención del crecimiento del árbol

A medida que el árbol crece, las reglas de división se vuelven más detalladas y el árbol cambia gradualmente de identificar “grandes” reglas, que identifican relaciones reales y confiables en los datos, a “pequeñas” reglas que solamente reflejan ruido. Un árbol

completamente desarrollado da como resultado hojas completamente puras y, por lo tanto, una precisión del 100 % en la clasificación de los datos sobre los que está entrenado. Esta precisión es, por supuesto, ilusoria: hemos sobreajustado (consultar "Equilibrio entre sesgo y varianza" en la página 247) los datos, ajustando el ruido en los datos de entrenamiento, no la señal que queremos identificar en los datos nuevos.

Necesitamos alguna forma de determinar cuándo dejar de cultivar un árbol en una etapa que se generalizará a nuevos datos. Hay varias formas de dejar de dividir en *R* y *Python*:

- Evitamos dividir una partición si una subpartición resultante es demasiado pequeña o si una hoja terminal es demasiado pequeña. En *rpart* (*R*), estas restricciones se controlan por separado mediante los parámetros `minsplit` y `minbucket`, respectivamente, con valores predeterminados de 20 y 7. En *DecisionTreeClassifier* de *Python*, podemos controlar las restricciones usando los parámetros `min_samples_split` (predeterminado 2) y `min_samples_leaf` (predeterminado 1).
- No dividimos una partición si la nueva partición no reduce "significativamente" la impureza. En *rpart*, esta decisión la controla el *parámetro de complejidad (complexity parameter)* `cp`, que es una medida de lo complejo que es un árbol: cuanto más complejo, mayor es el valor de `cp`. En la práctica, `cp` se usa para limitar el crecimiento de un árbol agregando una penalización a la complejidad adicional (divisiones) en el árbol. *DecisionTreeClassifier* (*Python*) tiene el parámetro `min_impurity_decrease`, que limita la división en función de un valor ponderado de disminución de impurezas. En este caso, valores más pequeños conducirán a árboles más complejos.

Estos métodos involucran reglas arbitrarias y pueden ser útiles para el trabajo exploratorio, pero no podemos determinar fácilmente los valores óptimos (es decir, valores que maximicen la precisión predictiva con nuevos datos). Necesitamos combinar la validación cruzada bien con el cambio sistemático de los parámetros del modelo o bien con la modificación del árbol mediante la poda.

Control de la complejidad del árbol en *R*

Con el parámetro de complejidad, `cp`, podemos estimar qué tamaño de árbol funcionará mejor con nuevos datos. Si `cp` es demasiado pequeño, el árbol se ajustará a los datos, ajustando el ruido y no la señal. Por otro lado, si `cp` es demasiado grande, entonces el árbol será demasiado pequeño y tendrá poco poder predictivo. El valor predeterminado en *rpart* es 0.01, aunque para conjuntos de datos más grandes, es probable que encontraremos que es demasiado grande. En el ejemplo anterior, `cp` se ha establecido en 0.005, ya que el valor predeterminado conducía a un árbol con una sola división. En el análisis exploratorio, basta con probar algunos valores.

La determinación del `cp` óptimo es un ejemplo del equilibrio entre sesgo y varianza. La forma más común de estimar un buen valor de `cp` es mediante la validación cruzada:

1. Dividimos los datos en conjuntos de entrenamiento y validación (reserva).
2. Hacemos crecer el árbol con los datos de entrenamiento.
3. Lo podamos gradualmente, paso a paso, registrando `cp` (usando los datos de entrenamiento [*training*]) en cada paso.
4. Hay que tener en cuenta el `cp` que corresponde al error mínimo (pérdida) en los datos de validación (*validation*).
5. Repartimos los datos en capacitación y validación, y repetimos el proceso de crecimiento, poda y registro de `cp`.
6. Hacemos esto una y otra vez, y promediamos los `cp` que reflejan el error mínimo para cada árbol.
7. Regresamos a los datos originales, o a los datos futuros, y hacemos crecer el árbol, deteniéndonos en este valor óptimo de `cp`.

En `rpart`, podemos usar el argumento `cptable` para producir una tabla de los valores de `cp` y su error de validación cruzada asociado (`xerror` en *R*), a partir del cual podemos determinar el valor de `cp` que tiene el error de validación cruzada más bajo.

Control de la complejidad del árbol en Python

En la implementación del árbol de decisiones de `scikit-learn`, el parámetro de complejidad se llama `ccp_alpha`. El valor predeterminado es 0, lo que significa que el árbol no se poda. Aumentar el valor conduce a árboles más pequeños. Podemos utilizar `GridSearchCV` para encontrar un valor óptimo.

Hay otra serie de parámetros del modelo que permiten controlar el tamaño del árbol. Por ejemplo, podemos variar `max_depth` en el rango de 5 a 30 y `min_samples_split` entre 20 y 100. El método `GridSearchCV` en `scikit-learn` es una manera conveniente de combinar la búsqueda exhaustiva a través de todas las combinaciones con validación cruzada. A continuación, se selecciona un conjunto de parámetros óptimo utilizando el rendimiento del modelo con validación cruzada.

Pronóstico de un valor continuo

El pronóstico de un valor continuo (también denominado *regresión* [*regression*]) con un árbol sigue la misma lógica y procedimiento, excepto que la impureza se mide por desviaciones cuadráticas de la media (errores cuadráticos) en cada subpartición, y el rendimiento predictivo se juzga por la raíz cuadrada del error cuadrático medio (RECM [*RMSE* en inglés]) en cada partición.

`scikit-learn` tiene el método `sklearn.tree.DecisionTreeRegressor` para entrenar el modelo de regresión del árbol de decisión.

Cómo se utilizan los árboles

Uno de los grandes obstáculos al que se enfrentan los modeladores predictivos en las organizaciones es la percepción de la naturaleza de "caja negra" de los métodos que utilizan, lo que genera la oposición de otros elementos de la organización. En este sentido, el modelo de árbol tiene dos aspectos atractivos:

- Los modelos de árbol proporcionan una herramienta visual para explorar los datos, para tener una idea de qué variables son importantes y cómo se relacionan entre sí. Los árboles pueden capturar relaciones no lineales entre variables predictoras.
- Los modelos de árbol proporcionan un conjunto de reglas que se pueden transmitir de manera eficaz a los no especialistas, ya sea para su implementación o para "vender" un proyecto de minería de datos.

Sin embargo, en lo que respecta al pronóstico, aprovechar los resultados de varios árboles suele ser más eficaz que utilizar un solo árbol. En particular, los algoritmos de bosque aleatorio y árbol potenciado casi siempre proporcionan una precisión y un rendimiento predictivo superiores (consultar "Métodos de bagging y bosque aleatorio" en la página 259 y "Boosting" en la página 270), pero se pierden las ventajas mencionadas anteriormente de un solo árbol.

Ideas clave

- Los árboles de decisión generan un conjunto de reglas para clasificar o pronosticar un resultado.
- Las reglas corresponden a particiones sucesivas de los datos en subparticiones.
- Cada partición, o división, hace referencia a un valor específico de una variable de pronóstico y divide los datos en registros donde ese valor de pronóstico está por encima o por debajo de ese valor empleado para la división.
- En cada etapa, el algoritmo de árbol elige la división que minimiza la impureza del resultado dentro de cada subpartición.
- Cuando no se pueden realizar más divisiones, el árbol está completamente desarrollado y cada nodo terminal, u hoja terminal, tiene registros de una sola clase. A los casos nuevos que sigan la ruta de una determinada regla (división) se les asignará la clase correspondiente.
- Un árbol completamente desarrollado se adapta a los datos y debe podarse para que capture la señal y no el ruido.
- Los algoritmos de árboles múltiples, como los bosques aleatorios y los árboles potenciados, producen un mejor rendimiento predictivo, pero pierden el poder comunicativo basado en reglas de los árboles individuales.

Lecturas complementarias

- Equipo de contenido de Analytics Vidhya, “Tree Based Algorithms: A Complete Tutorial from Scratch (de R & Python) (<https://www.analyticsvidhya.com/blog/2016/04/tree-based-algorithms-complete-tutorial-scratch-in-python/>)”, 12 de abril de 2016.
- Terry M. Therneau, Elizabeth J. Atkinson y Mayo Foundation, “An Introduction to Recursive Partitioning Using the RPART Routines” (<https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf>), 11 de abril de 2019.

Métodos de bagging y bosque aleatorio

En 1906, el estadístico sir Francis Galton se encontraba visitando la feria de un condado en Inglaterra, en la que se estaba llevando a cabo un concurso para adivinar el peso en canal de un buey en exposición. Hubo 800 conjeturas y, si bien las conjeturas individuales variaron ampliamente, tanto la media como la mediana estuvieron dentro del 1 % del peso real del buey. James Surowiecki ha explorado este fenómeno en su libro *The Wisdom of Crowds* (Doubleday, 2004). Este principio también se aplica a los modelos predictivos: promediar (o tomar la mayoría de votos) de múltiples modelos (un *conjunto [ensemble]* de modelos) resulta ser más preciso que seleccionar solamente un modelo.

Términos clave de bagging y bosque aleatorio

Conjunto de modelos

Formar un pronóstico mediante el uso de una colección de modelos.

Sinónimo

promedio del modelo

Bagging

Técnica general para formar una colección de modelos mediante el bootstrapping de los datos.

Sinónimo

agregación de bootstrap

Bosque aleatorio

Un tipo de estimación empaquetada basada en modelos de árbol de decisión.

Sinónimo

árboles de decisión empaquetados

Importancia de la variable

Medida de la importancia de una variable predictora en el rendimiento del modelo.

El enfoque de conjunto se ha aplicado a muchos métodos de modelado diferentes, para decirlo públicamente, en el Premio Netflix, en el que Netflix ofreció un premio de 1 millón de dólares al concursante que creara un modelo que produjera una mejora del 10 % en el pronóstico de la calificación que un cliente de Netflix otorgaría a una película. La versión sencilla de conjuntos es la siguiente:

1. Desarrollamos un modelo predictivo y registramos los pronósticos para un conjunto dado de datos.
2. Repetimos para varios modelos con los mismos datos.
3. Para cada registro que se va a pronosticar, tomamos el promedio (o el promedio ponderado o el voto mayoritario) de los pronósticos.

Los métodos de conjuntos de modelos (ensembles) se han aplicado de forma más sistemática y eficaz a los árboles de decisión. Los modelos de árbol de conjuntos de modelos son tan potentes que proporcionan una forma de construir buenos modelos predictivos con relativamente poco esfuerzo.

Más allá del simple algoritmo de conjuntos, existen dos variantes principales de modelos de conjunto: *bagging* y *boosting* (*bagging and boosting*). En el caso de los modelos de árbol de conjuntos de modelos, estos se denominan modelos de bosque aleatorio (*random forest*) y modelos de árbol potenciado (*boosted tree*). Esta sección se centra en bagging.

Bagging

Bagging, que significa "agregación de bootstrap", lo introdujo Leo Breiman en 1994. Supongamos que tenemos una respuesta Y y P variables predictoras $X = X_1, X_2, \dots, X_p$, con N registros.

Bagging es como el algoritmo básico para conjuntos, excepto que, en lugar de ajustar los distintos modelos a los mismos datos, cada nuevo modelo se ajusta a un remuestreo de bootstrap. A continuación se presenta el algoritmo de manera más formal:

1. Inicializamos M , el número de modelos a ajustar y n , el número de registros a elegir ($n < N$). Establecemos la iteración $m = 1$.
2. Tomamos el remuestreo bootstrap (es decir, con reposición) de n registros de los datos de entrenamiento para formar una submuestra Y_m y X_m (la bolsa).
3. Entrenamos un modelo usando Y_m y X_m para crear un conjunto de reglas de decisión $\hat{f}_m(X)$.
4. Incrementamos el contador del modelo $m = m + 1$. Si $m \leq M$, vamos al paso 2.

En el caso en que \hat{f}_m pronostique la probabilidad $Y = 1$, la estimación empaquetada es:

$$\hat{f} = \frac{1}{M} [\hat{f}_1(X) + \hat{f}_2(X) + \dots + \hat{f}_M(X)]$$

Bosque aleatorio

El *bosque aleatorio* (*random forest*) se basa en la aplicación de bagging a los árboles de decisión, con una ampliación importante: además de muestrear los registros, el algoritmo también muestrea las variables¹⁷. En los árboles de decisión tradicionales, para determinar cómo crear una subpartición de una partición A , el algoritmo elige la variable y el punto de división minimizando un criterio como puede ser la impureza de Gini (consultar "Medición de la homogeneidad o la impureza" en la página 254). Con el método de bosques aleatorios, en cada etapa del algoritmo, la elección de la variable se limita a un *subconjunto aleatorio de variables* (*random subset of variables*). En comparación con el algoritmo de árbol básico (consultar "Algoritmo de partición recursiva" en la página 252), el algoritmo de bosque aleatorio agrega dos pasos más: el método de bagging analizado anteriormente (consultar "Métodos de bagging y bosque aleatorio" en la página 259) y el muestreo de bootstrap de variables en cada división:

1. Tomemos una submuestra de bootstrap (con reposición) de los registros (*records*).
2. Para la primera división, muestreamos $p < P$ variables al azar sin reposición.
3. Para cada una de las variables muestreadas $X_{j(1)}, X_{j(2)}, \dots, X_{j(p)}$, aplicamos el algoritmo de división:
 - a. Para cada valor $s_{j(k)}$ de $X_{j(k)}$:
 - i. Dividimos los registros en la partición A , con $X_{j(k)} < s_{j(k)}$ como una partición y los registros restantes donde $X_{j(k)} \geq s_{j(k)}$ como otra partición.
 - ii. Medimos la homogeneidad de las clases dentro de cada subpartición de A .
 - b. Seleccionamos el valor de $s_{j(k)}$ que produzca la máxima homogeneidad de clase dentro de la partición.
4. Seleccionamos la variable $X_{j(k)}$ y el valor empleado para la división $s_{j(k)}$ que produce la máxima homogeneidad de clase dentro de la partición.
5. Continuamos con la siguiente división y repetimos los pasos anteriores, comenzando con el paso 2.
6. Continuamos con las divisiones adicionales, siguiendo el mismo procedimiento hasta que el árbol crezca.
7. Volvemos al paso 1, extraemos otra submuestra de bootstrap y comenzamos el proceso de nuevo.

¿Cuántas variables hay que muestrear en cada paso? Una regla general es elegir \sqrt{P} donde P es el número de variables predictoras. El paquete `randomForest` implementa el bosque aleatorio en *R*. Después aplica este paquete a los datos de préstamos:

¹⁷ El término bosque aleatorio (*random forest*) es una marca registrada de Leo Breiman y Adele Cutler y tiene licencia de Salford Systems. No existe un nombre estándar que no sea una marca comercial, y el término bosque aleatorio es tan sinónimo del algoritmo como Kleenex lo es de los pañuelos faciales.

Estadística práctica para ciencia de datos con R y Python

```
rf <- randomForest(outcome ~ borrower_score + payment_inc_ratio, data=loan3000)
rf
Call:
randomForest(formula = outcome ~ borrower_score + payment_inc_ratio,
  data = loan3000)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 1

OOB estimate of error rate: 39.17%
Confusion matrix:
      default    paid off   class.error
default     873       572  0.39584775
paid off     603       952  0.38778135
```

En *Python*, utilizamos el método `sklearn.ensemble.RandomForestClassifier`:

```
predictors = ['borrower_score', 'payment_inc_ratio']
outcome = 'outcome'

X = loan3000[predictors]
y = loan3000[outcome]

rf = RandomForestClassifier(n_estimators=500, random_state=1, oob_score=True)
rf.fit(X, y)
```

Por defecto, se entranan 500 árboles. Dado que solo hay dos variables en el conjunto de predictoras, el algoritmo selecciona aleatoriamente la variable sobre la que hacer la división en cada etapa (es decir, una submuestra bootstrap de tamaño 1).

La estimación del error *fuera de bolsa* (*out-of-bag* [OOB]) es la tasa de error para los modelos entrenados, aplicada a los datos que quedan fuera del conjunto de entrenamiento para ese árbol. Utilizando el resultado del modelo, se puede obtener un gráfico del error OOB en relación con el número de árboles en el bosque aleatorio en *R*:

```
error_df = data.frame(error_rate=rf$err.rate[, 'OOB'],
                      num_trees=1:rf$ntree)
ggplot(error_df, aes(x=num_trees, y=error_rate)) +
  geom_line()
```

La implementación de `RandomForestClassifier` no tiene una manera fácil de obtener estimaciones de los errores fuera de bolsa en función del número de árboles en el bosque aleatorio. Podemos entrenar una secuencia de clasificadores con un número creciente de árboles y realizar un seguimiento de los valores de `oob_score_`. Sin embargo, este método no es eficaz:

```
n_estimator = list(range(20, 510, 5))
oobScores = []
for n in n_estimator:
  rf = RandomForestClassifier(n_estimators=n, criterion='entropy',
                             max_depth=5, random_state=1, oob_score=True)
```

```

rf.fit(X, y)
oobScores.append(rf.oob_score_)
df = pd.DataFrame({ 'n': n_estimator, 'oobScore': oobScores })
df.plot(x='n', y='oobScore')

```

El resultado se muestra en la figura 6.6. La tasa de error disminuye rápidamente desde un valor superior a 0.44 antes de estabilizarse alrededor de 0.385. Los valores pronosticados se pueden obtener de la función `predict` y se pueden representar en *R* de la siguiente manera:

```

pred <- predict(rf, prob=TRUE)
rf_df <- cbind(loan3000, pred = pred)
ggplot(data=rf_df, aes(x=borrower_score, y=payment_inc_ratio,
                       shape=pred, color=pred, size=pred)) +
  geom_point(alpha=.8) +
  scale_color_manual(values = c('paid off'='#b8e186', 'default'='#d95f02')) +
  scale_shape_manual(values = c('paid off'=0, 'default'=1)) +
  scale_size_manual(values = c('paid off'=0.5, 'default'=2))

```

En *Python*, podemos crear un gráfico similar de la forma siguiente:

```

predictions = X.copy()
predictions['prediction'] = rf.predict(X)
predictions.head()

fig, ax = plt.subplots(figsize=(4, 4))

predictions.loc[predictions.prediction=='paid off'].plot(
    x='borrower_score', y='payment_inc_ratio', style='.',
    markerfacecolor='none', markeredgecolor='C1', ax=ax)
predictions.loc[predictions.prediction=='default'].plot(
    x='borrower_score', y='payment_inc_ratio', style='o',
    markerfacecolor='none', markeredgecolor='C0', ax=ax)
ax.legend(['paid off', 'default']);
ax.set_xlim(0, 1)
ax.set_ylim(0, 25)
ax.set_xlabel('borrower_score')
ax.set_ylabel('payment_inc_ratio')

```

El gráfico, que se muestra en la figura 6.7, es bastante revelador sobre la naturaleza del bosque aleatorio.

El método de bosque aleatorio es un método de "caja negra". Genera pronósticos más precisos que un árbol sencillo, pero se pierden las reglas de decisión intuitivas del árbol sencillo. Los pronósticos de bosque aleatorio también son algo ruidosos: hay que tener en cuenta que algunos prestatarios con una puntuación muy alta, lo que indica una alta calidad crediticia, aún terminan con un pronóstico de incumplimiento. Esto es el resultado de algunos registros inusuales en los datos y demuestra el peligro de sobreajuste por el bosque aleatorio (consultar "Equilibrio entre sesgo y varianza" en la página 247).

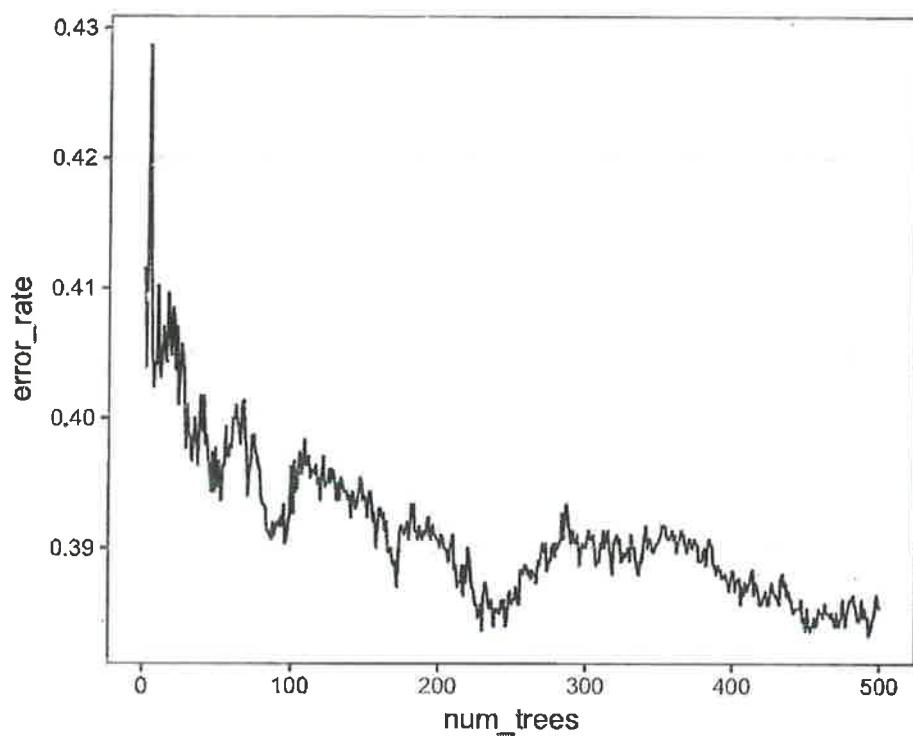


Figura 6.6 Ejemplo de la mejora en la precisión del bosque aleatorio con la adición de más árboles.

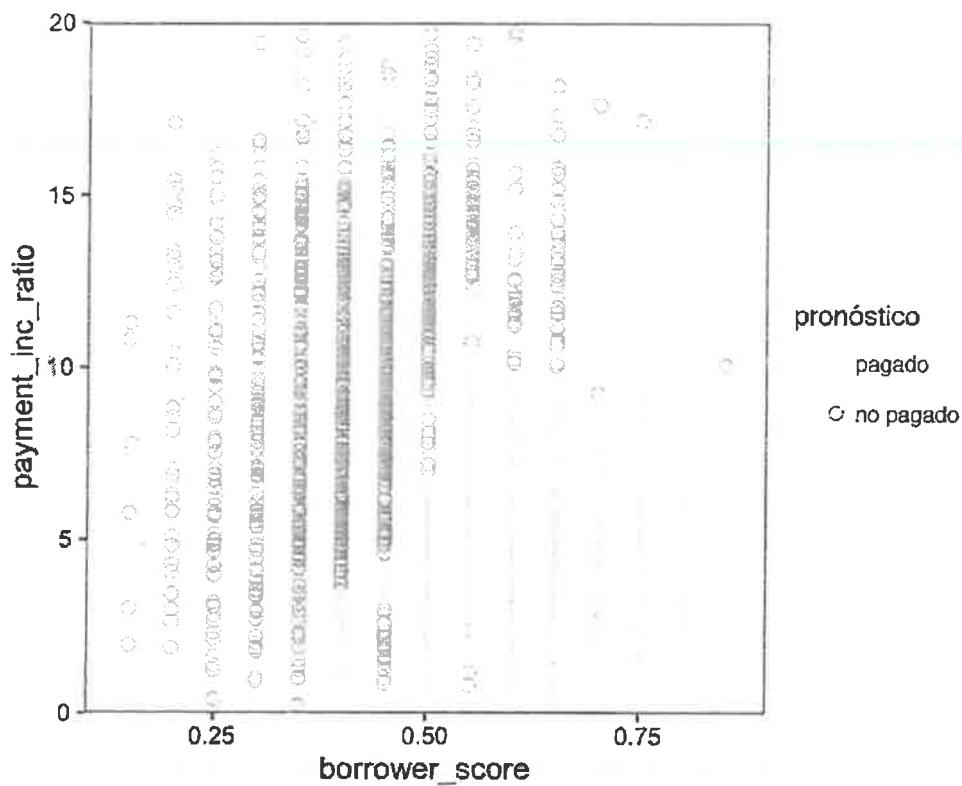


Figura 6.7 Resultados pronosticados de bosque aleatorio aplicado a los datos de préstamos no reembolsados.

Importancia de la variable

El poder del algoritmo de bosque aleatorio se manifiesta cuando creamos modelos predictivos para datos con muchas características y registros. Tiene la capacidad de determinar automáticamente qué predictoras son importantes y descubrir relaciones complejas entre predictoras en relación con los términos de interacción (consultar “Interacciones y efectos principales” en la página 174). Por ejemplo, ajustemos un modelo a los datos de incumplimiento de préstamos con todas las columnas incluidas. Lo siguiente muestra esto en R:

```
rf_all <- randomForest(outcome ~ ., data=loan_data, importance=TRUE)
rf_all
Call:
  randomForest(formula = outcome ~ ., data = loan_data, importance = TRUE)
  Type of random forest: classification
  Number of trees: 500
  No. of variables tried at each split: 4

  OOB estimate of error rate: 33.79%
Confusion matrix:
      paid   off default class.error
paid off    14676       7995  0.3526532
default     7325      15346  0.3231000
```

Y en Python:

```
predictors = ['loan_amnt', 'term', 'annual_inc', 'dti', 'payment_inc_ratio',
             'revol_bal', 'revol_util', 'purpose', 'delinq_2yrs_zero',
             'pub_rec_zero', 'open_acc', 'grade', 'emp_length', 'purpose_',
             'home_', 'emp_len_', 'borrower_score']
outcome = 'outcome'

X = pd.get_dummies(loan_data[predictors], drop_first=True)
y = loan_data[outcome]

rf_all = RandomForestClassifier(n_estimators=500, random_state=1)
rf_all.fit(X, y)
```

El argumento `importance=TRUE` solicita que el bosque aleatorio almacene información adicional sobre la importancia de diferentes variables. La función `varImpPlot` trazará el gráfico del rendimiento relativo de las variables (en relación con la permutación de esa variable):

```
varImpPlot(rf_all, type=1) ❶
varImpPlot(rf_all, type=2) ❷
```

- ❶ decrecimiento medio de la precisión
- ❷ decrecimiento medio de la impureza del nodo

En Python, `RandomForestClassifier` recopila información sobre la importancia de las características durante el entrenamiento y permite que esté disponible con el campo `feature_importances_`:

Estadística práctica para ciencia de datos con R y Python

```
importances = rf_all.feature_importances_
```

La "disminución de Gini" está disponible por la propiedad `feature_importance_` del clasificador ajustado. Sin embargo, la disminución de la precisión no está disponible de fábrica para *Python*. Podemos calcularla (puntuaciones) usando el siguiente código:

```
rf = RandomForestClassifier(n_estimators=500)
scores = defaultdict(list)

# cross-validate the scores on a number of different random splits of the data
for _ in range(3):
    train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.3)
    rf.fit(train_X, train_y)
    acc = metrics.accuracy_score(valid_y, rf.predict(valid_X))
    for column in X.columns:
        X_t = valid_X.copy()
        X_t[column] = np.random.permutation(X_t[column].values)
        shuff_acc = metrics.accuracy_score(valid_y, rf.predict(X_t))
        scores[column].append((acc-shuff_acc)/acc)
```

El resultado se muestra en la figura 6.8. Se puede crear un gráfico similar con este código *Python*:

```
df = pd.DataFrame({ 'feature': X.columns,
                    'Accuracy decrease': [np.mean(scores[column]) for column in X.columns], 'Gini decrease': rf_all.feature_importances_,
                    })
df = df.sort_values('Accuracy decrease')

fig, axes = plt.subplots(ncols=2, figsize=(8, 4.5))
ax = df.plot(kind='barh', x='feature', y='Accuracy decrease', legend=False, ax=axes[0])
ax.set_ylabel("")

ax = df.plot(kind='barh', x='feature', y='Gini decrease', legend=False, ax=axes[1])
ax.set_ylabel("") ax.get_yaxis().set_visible(False)
```

Hay dos formas de medir la importancia de las variables:

- Por la disminución de la precisión del modelo si los valores de una variable se permutan aleatoriamente (`type=1`). La permutación aleatoria de los valores tiene el efecto de eliminar todo el poder predictivo de esa variable. La precisión se calcula a partir de los datos fuera de bolsa (por lo que esta medida es efectivamente una estimación con validación cruzada).
- Por la disminución media en la puntuación de impureza de Gini (consultar “Medición de la homogeneidad o la impureza” en la página 254) para todos los nodos que se dividieron en una variable (`type=2`). Esto mide cuánto mejora la pureza de los nodos con la inclusión de esa variable. Esta medida se basa en el conjunto de entrenamiento y, por lo tanto, es menos confiable que una medida calculada con datos fuera de bolsa.

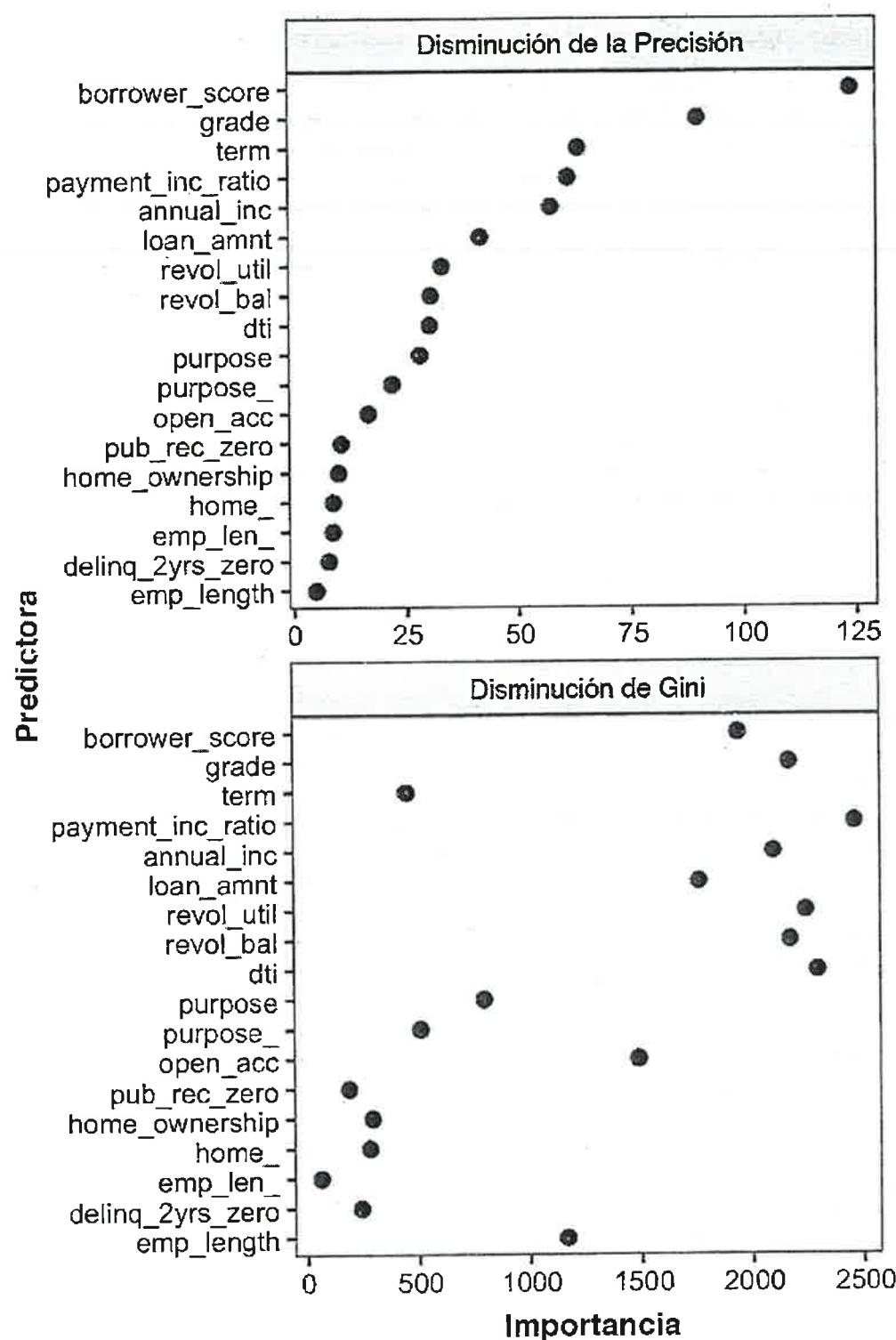


Figura 6.8 Importancia de las variables para el ajuste completo del modelo a los datos de préstamos.

Los paneles superior e inferior de la figura 6.8 muestran una importancia variable según la disminución en la precisión y en la impureza de Gini, respectivamente. Las variables de ambos paneles se clasifican según la disminución de la precisión. Las puntuaciones de importancia variable producidas por estas dos medidas son bastante diferentes.

Dado que la disminución de la precisión es una métrica más confiable, ¿por qué deberíamos usar la medida de disminución de impureza de Gini? Por defecto, `randomForest` calcula solo esta impureza de Gini: la impureza de Gini es un subproducto del algoritmo, mientras que la precisión del modelo por variable requiere cálculos adicionales (permutando aleatoriamente los datos y pronosticando estos datos). En los casos en que la complejidad de cálculo es importante, como es el caso de un entorno de producción donde se ajustan miles de modelos, puede que no valga la pena el esfuerzo computacional adicional. Además, la disminución de Gini arroja luz sobre qué variables está usando el bosque aleatorio para hacer sus reglas de división (hay que tener en cuenta que esta información, fácilmente visible en un árbol simple, se pierde efectivamente en un bosque aleatorio).

Hiperparámetros

El bosque aleatorio, al igual que ocurre con muchos algoritmos estadísticos de aprendizaje automático, se puede considerar un algoritmo de caja negra con botones para ajustar el funcionamiento de la caja. Estos botones se denominan hiperparámetros (*hyperparameters*), que son parámetros que debemos configurar antes de ajustar el modelo; no se optimizan como parte del proceso de entrenamiento. Si bien los modelos estadísticos tradicionales requieren elegir opciones (por ejemplo, la opción de predictoras para usar en un modelo de regresión), los hiperparámetros para el bosque aleatorio son más críticos, especialmente para evitar el sobreajuste. En particular, los dos hiperparámetros más importantes para el bosque aleatorio son:

`nodesize/min_samples_leaf`

El tamaño mínimo de los nodos terminales (hojas del árbol). El valor predeterminado en *R* es 1 para la clasificación y 5 para la regresión. La implementación de `scikit-learn` en *Python* usa el valor predeterminado de 1 para ambas.

`maxnodes/max_leaf_nodes`

El número máximo de nodos en cada árbol de decisión. Por defecto, no hay límite y el árbol más grande se ajustará sujeto a las restricciones de `nodesize`. Hay que tener en cuenta que en *Python* especificamos el número máximo de nodos terminales. Los dos parámetros están relacionados:

$$\text{maxnodes} = 2\text{max_leaf_nodes} - 1$$

Puede resultar tentador ignorar estos parámetros y simplemente utilizar los valores predeterminados. Sin embargo, el uso de los valores predeterminados puede provocar un sobreajuste cuando aplicamos el bosque aleatorio a datos con ruido.

Cuando aumentamos `nodesize/min_samples_leaf` o establecemos `maxnodes/max_leaf_nodes`, el algoritmo se ajustará a árboles más pequeños y es menos probable que cree reglas predictivas falsas. La validación cruzada se puede utilizar para probar los efectos de establecer diferentes valores para los hiperparámetros.

Ideas clave

- Los modelos de conjunto mejoran la precisión del modelo al combinar los resultados de muchos modelos.
- Bagging es un tipo particular de modelo de conjunto que se basa en ajustar muchos modelos a muestras bootstrapped de los datos y promediar los modelos.
- Bosque aleatorio es un tipo especial de bagging que se aplica a los árboles de decisión. Además de volver a muestrear los datos, el algoritmo de bosque aleatorio toma muestras de las variables predictoras al dividir los árboles.
- Un resultado útil de bosque aleatorio es una medida de importancia variable que clasifica a las predictoras en términos de su contribución a la precisión del modelo.
- Bosque aleatorio tiene un conjunto de hiperparámetros que deben ajustarse mediante validación cruzada para evitar el sobreajuste.

Boosting

Los modelos de conjuntos de modelos se han convertido en una herramienta estándar para el modelado predictivo. *Boosting* es una técnica general para crear un conjunto de modelos. Se desarrolló casi al mismo tiempo que *bagging* (consultar “Métodos de bagging y bosque aleatorio” en la página 259). Al igual que *bagging*, *boosting* se usa más frecuentemente con árboles de decisión. A pesar de sus similitudes, *boosting* adopta un enfoque muy diferente, uno que viene con muchas más florituras. Como resultado, mientras que *bagging* se puede hacer con relativamente poco ajuste, *boosting* requiere mucho más cuidado en su aplicación. Si estos dos métodos fueran automóviles, *bagging* podría considerarse un Honda Accord (confiable y estable), mientras que *boosting* podría considerarse un Porsche (potente, pero requiere más cuidados).

En los modelos de regresión lineal, los residuos se examinan a menudo para ver si se puede mejorar el ajuste (consultar “Diagramas de residuos parciales y falta de linealidad” en la página 185). Boosting lleva este concepto mucho más allá y se ajusta a una serie de modelos, en los que cada modelo sucesivo busca minimizar el error del modelo anterior.

Varias variantes del algoritmo se utilizan con frecuencia: *Adaboost*, *potenciación del gradiente* (*gradient boosting*) y *potenciación del gradiente estocástico* (*stochastic gradient boosting*). Esta última, la potenciación del gradiente estocástico, es la más general y más utilizada. De hecho, con la elección correcta de parámetros, el algoritmo puede emular al modelo de bosque aleatorio.

Términos clave de boosting

Conjunto de modelos

Formulación de un pronóstico mediante el uso de una colección de modelos.

Sinónimo

promedio del modelo

Boosting

Técnica general para ajustar una secuencia de modelos dando más peso a los registros con grandes residuos para cada ronda sucesiva.

Adaboost

Versión inicial de boosting que vuelve a ponderar los datos en función de los residuos.

Potenciación del gradiente

Una forma más general de boosting que se proyecta en términos de minimizar una función de coste.

Potenciación del gradiente estocástico

Algoritmo más general de boosting que incorpora remuestreo de registros y columnas en cada ronda.

Regularización

Técnica para evitar el sobreajuste al agregar un término de penalización a la función de coste en función del número de parámetros en el modelo.

Hyperparámetros

Parámetros que deben establecerse antes de ajustar el algoritmo.

El algoritmo boosting

Hay varios algoritmos de boosting, y la idea básica detrás de todos ellos es esencialmente la misma. El más fácil de entender es Adaboost, que procede de la siguiente manera:

1. Inicializamos M , el número máximo de modelos que se ajustarán, y establecemos el contador de iteraciones $m = 1$. Inicializamos las ponderaciones de las observaciones $w_i = 1/N$ para $i = 1, 2, \dots, N$. Inicializamos el modelo de conjunto $\hat{F}_0 = 0$.

2. Con las ponderaciones de las observaciones w_1, w_2, \dots, w_N , entrenamos el modelo \hat{f}_m que minimice el error ponderado e_m definido al sumar las ponderaciones de las observaciones mal clasificadas.
3. Agregamos el modelo al conjunto de modelos:

$$\hat{F}_m = \hat{F}_{m-1} + \alpha_m \hat{f}_m \text{ donde } \alpha_m = \frac{\log 1 - e_m}{e_m}$$

4. Actualizamos las ponderaciones w_1, w_2, \dots, w_N para que estas se incrementen para las observaciones que se clasificaron erróneamente. La magnitud del aumento depende de α_m , valores mayores de α_m conducen a mayores ponderaciones.
5. Incrementamos el contador del modelo $m = m + 1$. Si $m \leq M$, vamos al paso 2.

La estimación boosted viene dada por:

$$\hat{F} = \alpha_1 \hat{f}_1 + \alpha_2 \hat{f}_2 + \dots + \alpha_M \hat{f}_M$$

Al aumentar las ponderaciones de las observaciones que se han clasificado erróneamente, el algoritmo obliga a los modelos a entrenar más intensamente con los datos para los que tuvo un rendimiento deficiente. El factor α_m asegura que los modelos con menor error tengan un mayor peso.

La potenciación del gradiente es similar a Adaboost, pero enfoca el problema como una optimización de una función de costes. En lugar de ajustar las ponderaciones, la potenciación del gradiente ajusta los modelos a un *pseudorresiduo* (*pseudo-residual*), que tiene el efecto de entrenar más intensamente con los residuos más grandes. Con la misma intención del modelo de bosque aleatorio, la potenciación del gradiente estocástico agrega aleatoriedad al algoritmo mediante el muestreo de observaciones y variables predictoras en cada etapa.

XGBoost

El software de dominio público más utilizado para boosting es XGBoost, una implementación de la potenciación del gradiente estocástico desarrollado originalmente por Tianqi Chen y Carlos Guestrin de la Universidad de Washington. Una implementación computacionalmente eficiente con muchas opciones está disponible como paquete para la mayoría de los principales lenguajes de software de ciencia de datos. En *R*, XGBoost está disponible como paquete *xgboost* (<https://xgboost.readthedocs.io>) y con el mismo nombre también para *Python*.

El método *xgboost* tiene muchos parámetros que pueden y deben ajustarse (consultar “Hiperparámetros y validación cruzada” en la página 279). Dos parámetros muy importantes son *subsample*, que controla la fracción de observaciones que deben

Estadística práctica para ciencia de datos con R y Python

muestrearse en cada iteración, y eta, un factor de contracción aplicado a α_m en el algoritmo de boosting (consultar “El algoritmo boosting” en la página 271). El uso de subsample hace que el boosting actúe como bosque aleatorio, excepto que el muestreo se realiza sin reposición. El parámetro de contracción eta es útil para evitar el sobreajuste al reducir el cambio en las ponderaciones (un cambio más pequeño en las ponderaciones significa que es menos probable que el algoritmo se sobreajuste al conjunto de entrenamiento). Lo siguiente aplica xgboost en R a los datos de préstamos con solo dos variables predictoras:

```
predictors <- data.matrix(loan3000[, c('borrower_score', 'payment_inc_ratio')])  
label <- as.numeric(loan3000[, 'outcome']) - 1  
xgb <- xgboost(data=predictors, label=label, objective="binary:logistic",  
                 params=list(subsample=0.63, eta=0.1), nrounds=100)  
  
[1] train-error:0.358333  
[2] train-error:0.346333  
[3] train-error:0.347333  
...  
[99] train-error:0.239333  
[100] train-error:0.241000
```

Hay que tener en cuenta que xgboost no admite la sintaxis de fórmulas, por lo que las predictoras deben convertirse a `data.matrix` y la respuesta debe convertirse a variables 0/1. El argumento `objective` le dice a xgboost el tipo de problema de que se trata. En función de esta información, xgboost elegirá una métrica para la optimización.

En *Python*, xgboost tiene dos interfaces diferentes: una API de `scikit-learn` y una interfaz más funcional como la de *R*. Para ser coherentes con otros métodos de `scikit-learn`, se cambiaron los nombres de algunos parámetros. Por ejemplo, eta se renombró como `learning_rate`. El uso de eta no fallará, pero no tendrá el efecto deseado:

```
predictors = ['borrower_score', 'payment_inc_ratio']  
outcome = 'outcome'  
  
X = loan3000[predictors]  
y = loan3000[outcome]  
  
xgb = XGBClassifier(objective='binary:logistic', subsample=0.63)  
xgb.fit(X, y)  
  
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,  
              colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,  
              max_delta_step=0, max_depth=3, min_child_weight=1, missing=None,  
              n_estimators=100, n_jobs=1, nthread=None, objective='binary:logistic',  
              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,  
              silent=None, subsample=0.63, verbosity=1)
```

La función `predict` de *R* obtiene los valores pronosticados y, dado que solo hay dos variables, se pueden representar gráficamente en función de las predictoras:

```
pred <- predict(xgb, newdata=predictors)  
xgb_df <- cbind(loan3000, pred_default = pred > 0.5, prob_default = pred)  
ggplot(data=xgb_df, aes(x=borrower_score, y=payment_inc_ratio, color=pred_default,
```

```

shape=pred_default, size=pred_default)) + geom_point(alpha=.8) +
scale_color_manual(values = c('FALSE'='#b8e186', 'TRUE'="#d95f02")) +
scale_shape_manual(values = c('FALSE'=0, 'TRUE'=1)) +
scale_size_manual(values = c('FALSE'=0.5, 'TRUE'=2))

```

En *Python* se puede crear el mismo gráfico usando el siguiente código:

```

fig, ax = plt.subplots(figsize=(6, 4))

xgb_df.loc[xgb_df.prediction=='paid off'].plot(
    x='borrower_score', y='payment_inc_ratio', style='.',
    markerfacecolor='none', markeredgecolor='C1', ax=ax)

xgb_df.loc[xgb_df.prediction=='default'].plot(
    x='borrower_score', y='payment_inc_ratio', style='o',
    markerfacecolor='none', markeredgecolor='C0', ax=ax)
ax.legend(['paid off', 'default']);
ax.set_xlim(0, 1)
ax.set_ylim(0, 25)
ax.set_xlabel('borrower_score')
ax.set_ylabel('payment_inc_ratio')

```

El resultado se muestra en la figura 6.9. Cualitativamente, el resultado es similar a los pronósticos de bosque aleatorio. Ver la figura 6.7. Los pronósticos presentan algo de ruido en el sentido de que algunos prestatarios con una puntuación de prestatario muy alta aun así terminan con un pronóstico de incumplimiento.

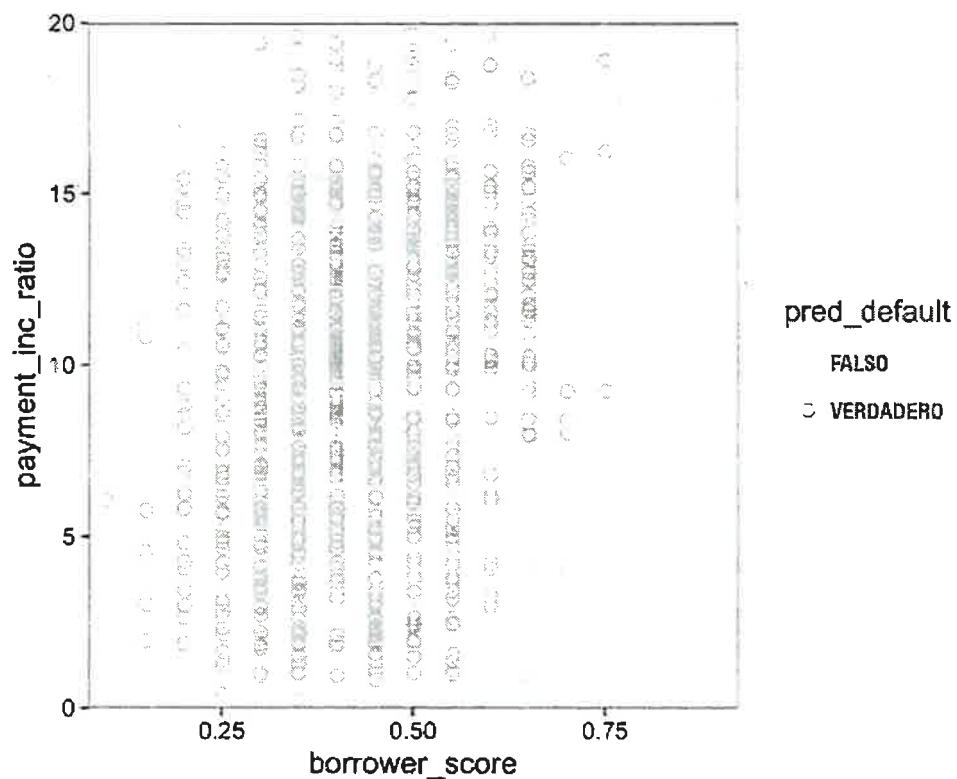


Figura 6.9 Resultados previstos de XGBoost aplicado a los datos de incumplimiento de préstamos.

Regularización: evitación del sobreajuste

La aplicación ciega de `xgboost` puede generar modelos inestables como resultado del sobreajuste (*overfitting*) de los datos de entrenamiento. El problema del sobreajuste es doble:

- Se degradará la precisión del modelo con datos nuevos que no estén en el conjunto de entrenamiento.
- Los pronósticos del modelo son muy variables, lo que genera resultados inestables.

Cualquier técnica de modelado es potencialmente propensa al sobreajustarse. Por ejemplo, si se incluyen demasiadas variables en una ecuación de regresión, el modelo puede terminar ofreciendo pronósticos falsos. Sin embargo, para la mayoría de las técnicas estadísticas, el sobreajuste se puede evitar mediante una cuidadosa selección de las variables predictoras. Incluso el método de bosque aleatorio generalmente crea un modelo razonable sin ajustar los parámetros.

Sin embargo, este no es el caso de `xgboost`. Ajustamos `xgboost` a los datos de préstamos para un conjunto de entrenamiento con todas las variables incluidas en el modelo. En *R*, podemos hacer esto de la siguiente manera:

```
seed <- 400820
predictors <- data.matrix(loan_data[, -which(names(loan_data) %in%
                                             'outcome')])

label <- as.numeric(loan_data$outcome) - 1
test_idx <- sample(nrow(loan_data), 10000)

xgb_default <- xgboost(data=predictors[-test_idx,], label=label[-test_idx],
                        objective='binary:logistic', nrounds=250, verbose=0)
pred_default <- predict(xgb_default, predictors[test_idx,])
error_default <- abs(label[test_idx] - pred_default) > 0.5
xgb_default$evaluation_log[250,]
mean(error_default)

iter train_error
1: 250  0.133043

[1] 0.3529
```

Utilizamos la función `train_test_split` en *Python* para dividir el conjunto de datos en conjuntos de entrenamiento y de prueba:

```
predictors = ['loan_amnt', 'term', 'annual_inc', 'dti', 'payment_inc_ratio',
              'revol_bal', 'revol_util', 'purpose', 'delinq_2yrs_zero',
              'pub_rec_zero', 'open_acc', 'grade', 'emp_length', 'purpose_',
              'home_', 'emp_len_', 'borrower_score']
outcome = 'outcome'

X = pd.get_dummies(loan_data[predictors], drop_first=True)
y = pd.Series([1 if o == 'default' else 0 for o in loan_data[outcome]])

train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=10000)
```

```

xgb_default = XGBClassifier(objective='binary:logistic', n_estimators=250,
                             max_depth=6, reg_lambda=0, learning_rate=0.3,
                             subsample=1)
xgb_default.fit(train_X, train_y)

pred_default = xgb_default.predict_proba(valid_X)[:, 1]

error_default = abs(valid_y - pred_default) > 0.5
print('default: ', np.mean(error_default))

```

El conjunto de prueba consta de 10 000 registros muestreados aleatoriamente a partir del total de los datos, y el conjunto de entrenamiento consta de los registros restantes. Boosting conduce a una tasa de error de solo el 13.3 % para el conjunto de entrenamiento. El conjunto de prueba, sin embargo, tiene una tasa de error mucho mayor, del 35.3 %. Esto es el resultado del sobreajuste: si bien boosting puede explicar muy bien la variabilidad en el conjunto de entrenamiento, las reglas de pronóstico no se aplican a los datos nuevos.

Boosting proporciona varios parámetros para evitar el sobreajuste, incluidos los parámetros eta (o learning_rate) y subsample (consultar “XGBoost” en la página 272). Otro enfoque es la *regularización* (regularization), una técnica que modifica la función de coste para penalizar (penalize) la complejidad del modelo. Los árboles de decisión se ajustan minimizando los criterios de coste, como la puntuación de impureza de Gini (consultar “Medición de la homogeneidad o la impureza” en la página 254). En xgboost, es posible modificar la función de coste agregando un término que mide la complejidad del modelo.

Hay dos parámetros en xgboost para regularizar el modelo: alpha y lambda, que corresponden a la distancia Manhattan (regularización L1) y la distancia euclídea al cuadrado (regularización L2), respectivamente (consultar “Métricas de distancia” en la página 241). El aumento de estos parámetros penalizará los modelos más complejos y reducirá el tamaño de los árboles que se ajustan. Por ejemplo, veamos lo que sucede en R si establecemos lambda en 1000:

```

xgb_penalty <- xgboost(data=predictors[-test_idx,], label=label[-test_idx],
                         params=list(eta=.1, subsample=.63, lambda=1000),
                         objective='binary:logistic', nrounds=250, verbose=0)
pred_penalty <- predict(xgb_penalty, predictors[test_idx,])
error_penalty <- abs(label[test_idx] - pred_penalty) > 0.5
xgb_penalty$evaluation_log[250,]
mean(error_penalty)

iter train_error
1: 250 0.30966

[1] 0.3286

```

En la API de scikit-learn, los parámetros se denominan reg_alpha y reg_lambda:

```

xgb_penalty = XGBClassifier(objective='binary:logistic', n_estimators=250,
                             max_depth=6, reg_lambda=1000, learning_rate=0.1,
                             subsample=0.63)
xgb_penalty.fit(train_X, train_y)

```

Estadística práctica para ciencia de datos con R y Python

```
pred_penalty = xgb_penalty.predict_proba(valid_X)[:, 1]
error_penalty = abs(valid_y - pred_penalty) > 0.5
print('penalty: ', np.mean(error_penalty))
```

Ahora, el error de entrenamiento es solo un poco más bajo que el error en el conjunto de prueba.

El método `predict` en *R* ofrece un argumento útil, `ntreelimit`, que obliga a usar solo los primeros i árboles en el pronóstico. Esto nos permite comparar directamente las tasas de error dentro de la muestra y fuera de la muestra a medida que se incluyen más modelos:

```
error_default <- rep(0, 250)
error_penalty <- rep(0, 250)
for(i in 1:250){
  pred_def <- predict(xgb_default, predictors[test_idx], ntreelimit=i)
  error_default[i] <- mean(abs(label[test_idx] - pred_def) >= 0.5)
  pred_pen <- predict(xgb_penalty, predictors[test_idx], ntreelimit=i)
  error_penalty[i] <- mean(abs(label[test_idx] - pred_pen) >= 0.5)
}
```

En *Python* podemos llamar al método `predict_proba` con el argumento `ntree_limit`:

```
results = []
for i in range(1, 250):
    train_default = xgb_default.predict_proba(train_X, ntree_limit=i)[:, 1]
    train_penalty = xgb_penalty.predict_proba(train_X, ntree_limit=i)[:, 1]
    pred_default = xgb_default.predict_proba(valid_X, ntree_limit=i)[:, 1]
    pred_penalty = xgb_penalty.predict_proba(valid_X, ntree_limit=i)[:, 1]
    results.append({
        'iterations': i,
        'default train': np.mean(abs(train_y - train_default) > 0.5),
        'penalty train': np.mean(abs(train_y - train_penalty) > 0.5),
        'default test': np.mean(abs(valid_y - pred_default) > 0.5),
        'penalty test': np.mean(abs(valid_y - pred_penalty) > 0.5),
    })
results = pd.DataFrame(results)
results.head()
```

La salida del modelo proporciona el error del conjunto de entrenamiento al componente `xgb_default$evaluation_log`. Al combinar lo anterior con los errores fuera de la muestra, podemos trazar un gráfico de los errores en función del número de iteraciones:

```
errors <- rbind(xgb_default$evaluation_log,
                  xgb_penalty$evaluation_log,
                  data.frame(iter=1:250, train_error=error_default),
                  data.frame(iter=1:250, train_error=error_penalty))
errors$type <- rep(c('default train', 'penalty train',
                     'default test', 'penalty test'), rep(250, 4))
ggplot(errors, aes(x=iter, y=train_error, group=type)) +
  geom_line(aes(linetype=type, color=type))
```

Podemos usar el método de diagramas de pandas para crear el gráfico de líneas. El eje proporcionado por el primer gráfico nos permite superponer líneas adicionales en el mismo gráfico. Este es un patrón que admiten muchos de los paquetes de gráficos de *Python*:

```

ax = results.plot(x='iterations', y='default test')
results.plot(x='iterations', y='penalty test', ax=ax)
results.plot(x='iterations', y='default train', ax=ax)
results.plot(x='iterations', y='penalty train', ax=ax)

```

El resultado, que aparece en la figura 6.10, muestra cómo el modelo estándar mejora constantemente la precisión del conjunto de entrenamiento, pero en realidad empeora para el conjunto de prueba. El modelo penalizado no presenta este comportamiento.

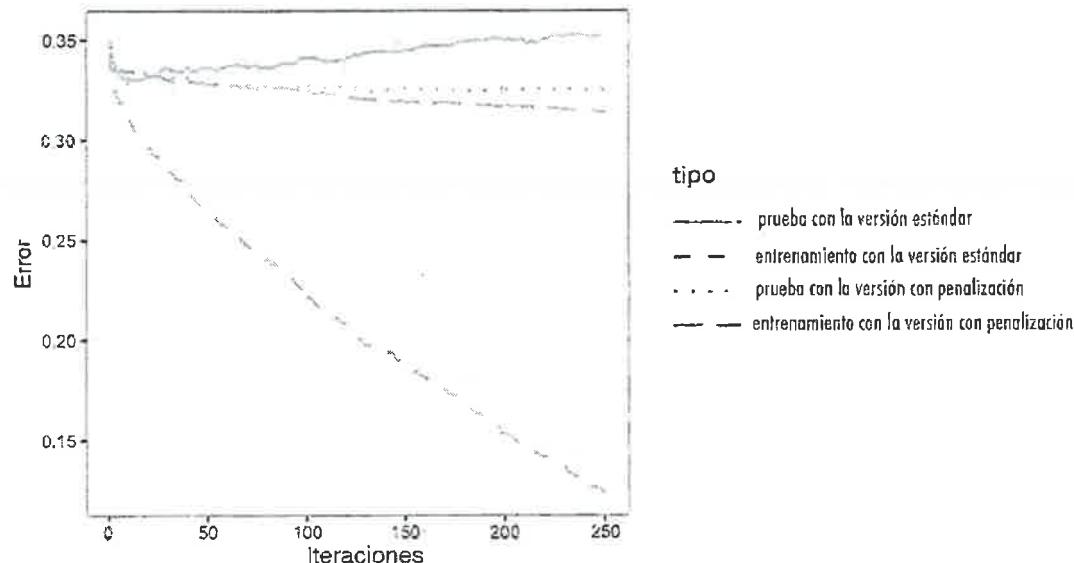


Figura 6.10 Tasa de error de XGBoost estándar frente a una versión con penalización de XGBoost.

Regresión de cresta y lazo

La incorporación de una penalización a la complejidad de un modelo para ayudar a evitar el sobreajuste se remonta a la década de 1970. La regresión de mínimos cuadrados minimiza la suma de los cuadrados de los residuos (RSS). Consultar “Mínimos cuadrados” en la página 148. La regresión de crestas minimiza la suma de los cuadrados de los residuos más un término de penalización que es función del número y tamaño de los coeficientes:

$$\sum_{i=1}^n (Y_i - \hat{b}_0 - \hat{b}_1 X_1 - \cdots - \hat{b}_p X_p)^2 + \lambda (\hat{b}_1^2 + \cdots + \hat{b}_p^2)$$

El valor de λ determina cuánto se penalizan los coeficientes; los valores más grandes producen modelos que tienen menos probabilidades de sobreajustarse a los datos. *Lasso* es similar, excepto que usa la distancia Manhattan en lugar de la distancia euclíadiana como término de penalización:

$$\sum_{i=1}^n (Y_i - \hat{b}_0 - \hat{b}_1 X_1 - \cdots - \hat{b}_p X_p)^2 + \alpha (\|\hat{b}_1\| + \cdots + \|\hat{b}_p\|)$$

El uso de la distancia euclídea también se conoce como regularización L2 y el uso de la distancia Manhattan como regularización L1. Los parámetros `xgboost`, `lambda` (`reg_lambda`) y `alpha` (`reg_alpha`) actúan de manera similar.

Hiperparámetros y validación cruzada

`xgboost` tiene una enorme variedad de hiperparámetros; consultar “Hiperparámetros de XGBoost” en la página 281 para ver una discusión. Como se ve en “Regularización: evitación del sobreajuste” en la página 274, la elección específica puede cambiar drásticamente el ajuste del modelo. Dada una enorme combinación de hiperparámetros para elegir, ¿cómo debemos guiarnos en nuestra elección? Una solución estándar a este problema es utilizar la validación cruzada (*cross-validation*); consultar “Validación cruzada” en la página 155. La validación cruzada divide aleatoriamente los datos en K grupos diferentes, también llamados *pliegues* (*folds*). Para cada pliegue, se entrena un modelo sobre los datos que no están en el pliegue y luego se evalúa sobre los datos en el pliegue. Esto produce una medida de precisión del modelo en datos fuera de la muestra. El mejor conjunto de hiperparámetros es el que proporciona el modelo con el error general más bajo calculado al promediar los errores de cada uno de los pliegues.

Para ilustrar la técnica, la aplicamos a la selección de parámetros para `xgboost`. En este ejemplo, exploramos dos parámetros: el parámetro de contracción `eta` (`learning_rate`; consultar “XGBoost” en la página 272) y la profundidad máxima de los árboles `max_depth`. El parámetro `max_depth` es la profundidad máxima de un nodo hoja a la raíz del árbol con un valor predeterminado de seis. Esto nos brinda otra forma de controlar el sobreajuste: los árboles profundos tienden a ser más complejos y pueden sobreajustar los datos. Primero configuramos los pliegues y la lista de parámetros. En R, esto se hace de la siguiente manera:

```
N <- nrow(loan_data)
fold_number <- sample(1:5, N, replace=TRUE)
params <- data.frame(eta = rep(c(.1, .5, .9), 3),
                     max_depth = rep(c(3, 6, 12), rep(3,3)))
```

Ahora aplicamos el algoritmo anterior para calcular el error de cada modelo y cada pliegue usando cinco pliegues:

```
error <- matrix(0, nrow=9, ncol=5)
for(i in 1:nrow(params)){
  for(k in 1:5){
    fold_idx <- (1:N)[fold_number == k]
    xgb <- xgboost(data=predictors[-fold_idx,], label=label[-fold_idx],
                    params=list(eta=params[i, 'eta'],
                               max_depth=params[i, 'max_depth']),
                    objective='binary:logistic', nrounds=100, verbose=0)
    pred <- predict(xgb, predictors[fold_idx,])
    error[i, k] <- mean(abs(label[fold_idx] - pred) >= 0.5)
  }
}
```

En el siguiente código de *Python*, creamos todas las combinaciones posibles de hiperparámetros y ajustamos y evaluamos modelos con cada combinación:

```

idx = np.random.choice(range(5), size=len(X), replace=True)
error = []
for eta, max_depth in product([0.1, 0.5, 0.9], [3, 6, 9]): ❶
    xgb = XGBClassifier(objective='binary:logistic', n_estimators=250,
                         max_depth=max_depth, learning_rate=eta)
    cv_error = []
    for k in range(5):
        fold_idx = idx == k
        train_X = X.loc[~fold_idx];
        train_y = y[~fold_idx]
        valid_X = X.loc[fold_idx];
        valid_y = y[fold_idx]

        xgb.fit(train_X, train_y)
        pred = xgb.predict_proba(valid_X)[:, 1]
        cv_error.append(np.mean(abs(valid_y - pred) > 0.5))
    error.append({
        'eta': eta,
        'max_depth': max_depth,
        'avg_error': np.mean(cv_error)
    })
print(error[-1])
errors = pd.DataFrame(error)

```

- ❶ Usamos la función `itertools.product` de la biblioteca estándar de *Python* para crear todas las combinaciones posibles de los dos hiperparámetros.

Dado que estamos instalando 45 modelos en total, esto puede llevar un tiempo. Los errores se almacenan como una matriz con los modelos a lo largo de las filas y los pliegues a lo largo de las columnas. Usando la función `rowMeans`, podemos comparar la tasa de error para los diferentes conjuntos de parámetros:

```

avg_error <- 100 * round(rowMeans(error), 4)
cbind(params, avg_error)
  eta max_depth avg_error
1 0.1         3    32.90
2 0.5         3    33.43
3 0.9         3    34.36
4 0.1         6    33.08
5 0.5         6    35.60
6 0.9         6    37.82
7 0.1        12    34.56
8 0.5        12    36.83
9 0.9        12    38.18

```

La validación cruzada sugiere que el uso de árboles menos profundos con un valor menor de `eta/learning_rate` produce resultados más precisos. Dado que estos modelos también son más estables, los mejores parámetros para usar son `eta=0.1` y `max_depth=3` (o posiblemente `max_depth=6`).

Hiperparámetros de XGBoost

Los hiperparámetros de `xgboost` se utilizan principalmente para equilibrar el sobreajuste empleando la precisión y la complejidad computacional. Para una discusión completa de los parámetros, consultar la documentación de `xgboost` (<https://xgboost.readthedocs.io/en/latest/>).

`eta/learning_rate`

Factor de contracción entre 0 y 1 aplicado a α en el algoritmo de boosting. El valor predeterminado es 0.3, pero para datos ruidosos, se recomiendan valores más pequeños (por ejemplo, 0.1). En *Python*, el valor predeterminado es 0.1.

`nrounds/n_estimators`

El número de rondas de boosting. Si `eta` se establece en un valor pequeño, es importante aumentar el número de rondas ya que el algoritmo aprende más lentamente. Siempre que se incluyan algunos parámetros para evitar el sobreajuste, tener más rondas no está de más.

`max_depth`

Profundidad máxima del árbol (el valor predeterminado es 6). En contraste con el bosque aleatorio, que se adapta a árboles muy profundos, el boosting generalmente se adapta a árboles poco profundos. Esto tiene la ventaja de evitar interacciones complejas espurias o falsas en el modelo que pueden surgir de datos ruidosos. En *Python*, el valor predeterminado es 3.

`subsample y colsample_bytree`

Fracción de los registros a muestrear sin reposición y fracción de predictoras a muestrear para usar en el ajuste de los árboles. Estos parámetros, que son similares a los de los bosques aleatorios, ayudan a evitar el sobreajuste. El valor predeterminado es 1.0.

`lambda/reg_lambda y alpha/reg_alpha`

Los parámetros de regularización para ayudar a controlar el sobreajuste (consultar “Regularización: evitación del sobreajuste” en la página 274). Los valores predeterminados para *Python* son `reg_lambda=1` y `reg_alpha=0`. En *R*, ambos valores tienen un valor predeterminado de 0.

Ideas clave

- Boosting es una clase de modelos de conjuntos basados en el ajuste de una secuencia de modelos, con más peso para los registros con grandes errores en rondas sucesivas.
- Boosting de gradiente estocástico es el tipo más general de impulso y ofrece el mejor rendimiento. La forma más común de la potenciación del gradiente estocástico utiliza modelos de árbol.
- XGBoost es un paquete de software popular y computacionalmente eficiente para la potenciación del gradiente estocástico; está disponible en todos los lenguajes habituales que se utilizan en la ciencia de datos.
- Boosting tiende a sobreajustar los datos, y los hiperparámetros deben ajustarse para evitar esto.
- La regularización es una forma de evitar el sobreajuste al incluir un término de penalización en el número de parámetros (por ejemplo, el tamaño del árbol) en un modelo.
- La validación cruzada es especialmente importante para el boosting debido a la gran cantidad de hiperparámetros que deben configurarse.

Resumen

En este capítulo se han descrito dos métodos de clasificación y pronóstico que "aprenden" de manera flexible y localmente a partir de los datos, en lugar de comenzar con un modelo estructural (por ejemplo, la regresión lineal) que se ajusta a todo el conjunto de datos. K-vecinos más cercanos es un proceso sencillo que analiza registros similares y asigna su clase mayoritaria (o valor promedio) al registro que se pronostica. Al probar varios valores de corte (división) de las variables predictoras, los modelos de árbol dividen iterativamente los datos en secciones y subsecciones que son cada vez más homogéneas con respecto a la clase. Los valores divididos más efectivos conforman una ruta, y también una "regla", hacia una clasificación o pronóstico. Los modelos de árbol son una herramienta de pronóstico muy potente y popular, que a menudo supera a otros métodos. Han dado lugar a varios métodos conjuntos (bosques aleatorios, boosting, bagging) que agudizan el poder predictivo de los árboles.

CAPÍTULO 7

Aprendizaje no supervisado

El término *aprendizaje no supervisado* (*unsupervised learning*) se refiere a los métodos estadísticos que extraen información de los datos sin necesidad de entrenar un modelo con datos etiquetados (datos en los que se conoce el resultado que nos interesa). En los capítulos del 4 al 6, el objetivo era construir un modelo (conjunto de reglas) para pronosticar una variable de respuesta a partir de un conjunto de variables predictoras. Este es el aprendizaje supervisado. Por el contrario, el aprendizaje no supervisado también construye un modelo de los datos, pero no distingue entre la variable de respuesta y las variables predictoras.

El aprendizaje no supervisado se puede utilizar para lograr diferentes objetivos. En algunos casos, se puede utilizar para crear una regla predictiva en ausencia de una respuesta etiquetada. Se pueden utilizar métodos de agrupamiento para identificar grupos de datos significativos. Por ejemplo, utilizando los clics en la web y los datos demográficos de un usuario en un sitio web, es posible que podamos agrupar diferentes tipos de usuarios. A continuación, el sitio web podría personalizarse para estos diferentes tipos de visitantes.

En otros casos, el objetivo puede ser reducir la *dimensión* (*dimension*) de los datos a un conjunto de variables más manejable. Este conjunto reducido podría utilizarse posteriormente como entrada para un modelo predictivo, como los de regresión o clasificación. Por ejemplo, podemos tener miles de sensores para monitorizar un proceso industrial. Al reducir los datos a un conjunto más pequeño de características, es posible que podamos construir un modelo más potente e interpretable para pronosticar fallos en el proceso que el modelo que se podría crear al incluir flujos de datos de miles de sensores.

Por último, el aprendizaje no supervisado se puede ver como una extensión del análisis exploratorio de datos (consultar el capítulo 1) en situaciones en las que se enfrenta a una gran cantidad de variables y registros. El objetivo es obtener información sobre un conjunto de datos y cómo se relacionan las diferentes variables entre sí. Las técnicas sin supervisión nos permiten examinar y analizar estas variables y descubrir relaciones.



Aprendizaje no supervisado y pronóstico

El aprendizaje no supervisado puede desempeñar un papel importante en el pronóstico, tanto para problemas de regresión como de clasificación. En algunos casos, queremos pronosticar una categoría en ausencia de datos etiquetados. Por ejemplo, podríamos querer pronosticar el tipo de vegetación en un área a partir de un conjunto de datos sensoriales satelitales. Dado que no tenemos una variable de respuesta para entrenar un modelo, la agrupación en clústeres (grupos) nos brinda una forma de identificar patrones comunes y categorizar las regiones.

La agrupación en clústeres es una herramienta especialmente importante para el "problema del arranque en frío". En este tipo de problemas, como puede ser lanzar una nueva campaña de marketing o identificar posibles nuevos tipos de fraude o spam, es posible que inicialmente no tengamos ninguna respuesta para entrenar el modelo. Con el tiempo, a medida que se recopilan datos, podemos aprender más sobre el sistema y construir un modelo predictivo tradicional. Pero la agrupación en clústeres nos ayuda a iniciar el proceso de aprendizaje más rápidamente al identificar segmentos de población.

El aprendizaje no supervisado también es importante como componente básico de las técnicas de regresión y clasificación. En el caso de big data, si una pequeña subpoblación no está bien representada en la población general, es posible que el modelo entrenado no funcione bien para esa subpoblación. Con la agrupación en clústeres es posible identificar y etiquetar subpoblaciones. Los modelos separados se pueden adaptar a las diferentes subpoblaciones. Alternativamente, la subpoblación se puede representar con su propia característica, lo que obliga al modelo general a considerar explícitamente la identidad de la subpoblación como una predictoría.

Análisis de componentes principales

A menudo, las variables variarán juntas (covariarán), y parte de la variación en una de ellas se duplica en realidad por la variación en la otra (por ejemplo, cheques y propinas de restaurantes). El análisis de componentes principales (principal components analysis [PCA]) es una técnica para descubrir la forma en que covarian las variables numéricas¹⁸.

¹⁸ Esta sección y las siguientes de este capítulo © 2020 Datastats, LLC, Peter Bruce, Andrew Bruce y Peter Gedeck; utilizado con la correspondiente autorización.

Términos clave del análisis de componentes principales

Componente principal

Combinación lineal de las variables predictoras.

Cargas

Ponderaciones que transforman las predictoras en componentes.

Sinónimo

ponderaciones

Diagrama de sedimentación

Diagrama de las varianzas de los componentes, que muestra la importancia relativa de los mismos, ya sea como varianza explicada o como proporción de la varianza explicada.

La idea en PCA es combinar múltiples variables predictoras numéricas en un conjunto más pequeño de variables, que son combinaciones lineales ponderadas del conjunto original. El conjunto más pequeño de variables, los componentes principales (*principal components*), "explica" la mayor parte de la variabilidad del conjunto completo de variables, reduciendo la dimensión de los datos. Las ponderaciones utilizadas para formar los componentes principales revelan las contribuciones relativas de las variables originales a los nuevos componentes principales.

La técnica de PCA la propuso por primera vez Karl Pearson (https://books.google.es/books?id=Sq1JAQAAJ&vq=559&pg=PA559&redir_esc=y#v=onepage&q=559&f=false). En lo que quizás fue el primer artículo sobre aprendizaje no supervisado, Pearson reconoció que en muchos problemas hay variabilidad en las variables predictoras, por lo que desarrolló PCA como una técnica para modelar esta variabilidad. PCA puede verse como la versión no supervisada del análisis discriminante lineal. Consultar "Análisis discriminante" en la página 201.

Un ejemplo sencillo

Para dos variables, X_1 y X_2 , hay dos componentes principales Z_i ($i = 1$ o 2):

$$Z_i = w_{i,1}X_1 + w_{i,2}X_2$$

Las ponderaciones ($w_{i,1}$, $w_{i,2}$) se conocen como cargas (*loadings*) de los componentes. Estas transforman las variables originales en los componentes principales. El primer componente principal, Z_1 , es la combinación lineal que mejor explica la variación total. El segundo componente principal, Z_2 , es ortogonal al primero y explica en la medida de lo posible la mayor parte de la variación restante. (Si hubiera componentes adicionales, cada uno de estos sería ortogonal a los demás.)



También es habitual calcular los componentes principales sobre las desviaciones de las medias de las variables predictoras, en lugar de sobre los valores mismos.

Podemos calcular los componentes principales en R usando la función `princomp`. El siguiente código lleva a cabo PCA sobre las rentabilidades del precio de las acciones de Chevron (CVX) y Exxon-Mobil (XOM):

```
oil_px <- sp500_px[, c('CVX', 'XOM')]
pca <- princomp(oil_px)
pca$loadings
```

Loadings:

	Comp.1	Comp.2
CVX	-0.747	0.665
XOM	-0.665	-0.747

	Comp.1	Comp.2
SS loadings	1.0	1.0
Proportion Var	0.5	0.5
Cumulative Var	0.5	1.0

En *Python* podemos usar la implementación `sklearn.decomposition.PCA` de `scikit-learn`:

```
pcs = PCA(n_components=2)
pcs.fit(oil_px)
loadings = pd.DataFrame(pcs.components_, columns=oil_px.columns)
loadings
```

Las ponderaciones de CVX y XOM para el primer componente principal son -0.747 y -0.665, y para el segundo componente principal son 0.665 y -0.747. ¿Cómo interpretar estas cantidades? El primer componente principal es esencialmente un promedio de CVX y XOM, lo que refleja la correlación entre las dos empresas del sector de la energía. El segundo componente principal mide cuando divergen los precios de las acciones de CVX y XOM.

Es instructivo presentar en una gráfica los componentes principales con los datos. Aquí creamos una visualización en *R*:

```
loadings <- pca$loadings
ggplot(data=oil_px, aes(x=CVX, y=XOM)) +
  geom_point(alpha=.3) +
  stat_ellipse(type='norm', level=.99) +
  geom_abline(intercept = 0, slope = loadings[2,1]/loadings[1,1]) +
  geom_abline(intercept = 0, slope = loadings[2,2]/loadings[1,2])
```

El siguiente código crea una visualización similar en *Python*:

```
def abline(slope, intercept, ax):
    """Calculate coordinates of a line based on slope and intercept"""
    x_vals = np.array(ax.get_xlim())
    return (x_vals, Intercept + slope * x_vals)
ax = oil_px.plot.scatter(x='XOM', y='CVX', alpha=0.3, figsize=(4, 4))
```

```

ax.set_xlim(-3, 3)
ax.set_ylim(-3, 3)
ax.plot(*abline(loadings.loc[0, 'CVX'] / loadings.loc[0, 'XOM'], 0, ax),
        '--', color='C1')
ax.plot(*abline(loadings.loc[1, 'CVX'] / loadings.loc[1, 'XOM'], 0, ax),
        '--', color='C1')

```

El resultado se muestra en la figura 7.1.

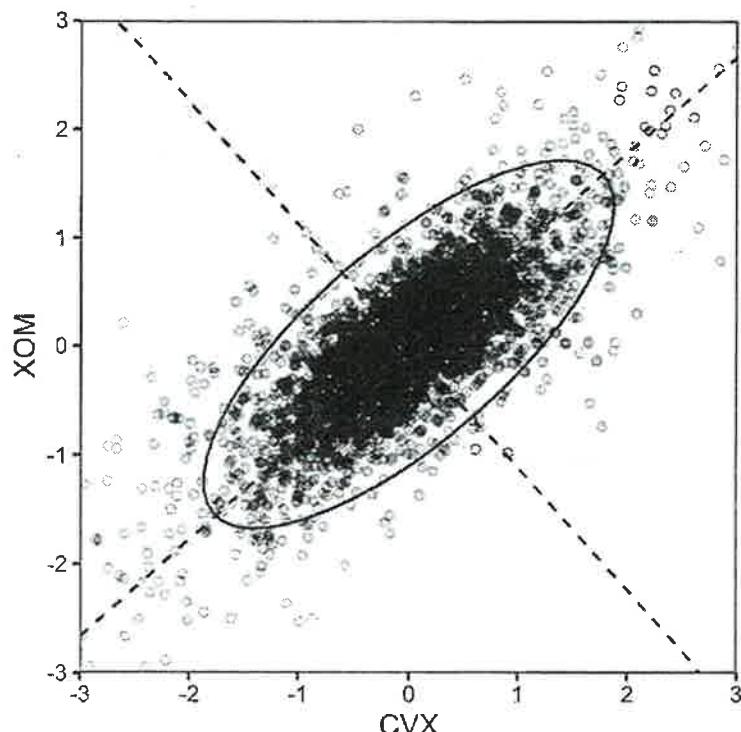


Figura 7.1 Componentes principales de la rentabilidad de las acciones de Chevron (CVX) y ExxonMobil (XOM).

Las líneas discontinuas muestran la dirección de los dos componentes principales: el primero está a lo largo del eje largo de la elipse y el segundo a lo largo del eje corto. Podemos ver que la mayor parte de la variabilidad en las rentabilidades de las dos acciones se explica por el primer componente principal. Esto tiene sentido, ya que los precios de las acciones de la energía tienden a variar como grupo.



Las ponderaciones del primer componente principal son ambas negativas, pero invertir el signo de todas las ponderaciones no cambia el componente principal. Por ejemplo, usar ponderaciones de 0.747 y 0.665 para el primer componente principal es equivalente a las ponderaciones negativas, al igual que una línea infinita definida por el origen y las coordenadas 1, 1 es lo mismo que una definida por el origen y las coordenadas -1, -1.

Cálculo de los componentes principales

Pasar de dos variables a un número mayor de variables es sencillo. Para el primer componente, simplemente incluimos las variables predictoras adicionales en la combinación lineal, asignando ponderaciones que optimicen el conjunto de las variaciones conjuntas de todas las variables predictoras en este primer componente principal (*covarianza [covariance]* es el término estadístico. Consultar “Matriz de covarianza” en la página 202). El cálculo de los componentes principales es un método estadístico clásico, que se basa en la matriz de correlación de los datos o en la matriz de covarianza, y se ejecuta de forma rápida, al no depender de iteraciones. Como se señaló anteriormente, el análisis de componentes principales funciona solo con variables numéricas, no categóricas. El proceso completo se puede describir de la siguiente manera:

1. Al crear el primer componente principal, PCA calcula la combinación lineal de variables predictoras que maximiza el porcentaje de la varianza total indicada más arriba.
2. Esta combinación lineal se convierte entonces en el primer predictor “nuevo”, Z_1 .
3. PCA repite este proceso, usando las mismas variables con diferentes ponderaciones, para crear un segundo predictor, Z_2 . La ponderación se realiza de manera que Z_1 y Z_2 no estén correlacionados.
4. El proceso continúa hasta que tengamos tantas variables nuevas, o componentes, Z_i como variables originales X_i .
5. Optamos por quedarnos con tantos componentes como sean necesarios para dar cuenta de la mayor parte de la variación.
6. El resultado hasta ahora es un conjunto de ponderaciones para cada componente. El paso final es convertir los datos originales en nuevas puntuaciones de componentes principales aplicando las ponderaciones a los valores originales. Estas nuevas puntuaciones se pueden usar luego como el conjunto reducido de variables predictoras.

Interpretación de componentes principales

La naturaleza de los componentes principales a menudo revela información sobre la estructura de los datos. Hay un par de presentaciones visuales tradicionales para ayudarnos a obtener información sobre los componentes principales. Uno de estos métodos es un diagrama de barras para visualizar la importancia relativa de los componentes principales (el nombre se deriva de la semejanza del gráfico con la pendiente de una ladera. Aquí, el eje y es el autovalor): El siguiente código R muestra un ejemplo de algunas de las principales empresas del S&P 500:

```
syms <- c( 'AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM',
         'SLB', 'COP', 'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST')
top_sp <- sp500_px[row.names(sp500_px)>='2005-01-01',
```

```
syms]sp_pca <- princomp(top_sp)
screeplot(sp_pca)
```

La información para crear un diagrama de cargas a partir del resultado de `scikit-learn` está disponible en `explained_variance_`. Aquí, lo convertimos en un marco de datos de pandas y lo usamos para hacer un gráfico de barras:

```
syms = sorted(['AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM', 'SLB', 'COP',
              'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST'])
top_sp = sp500_px.loc[sp500_px.index >= '2011-01-01', syms]

sp_pca = PCA()
sp_pca.fit(top_sp)

explained_variance = pd.DataFrame(sp_pca.explained_variance_)
ax = explained_variance.head(10).plot.bar(legend=False, figsize=(4, 4))
ax.set_xlabel('Component')
```

Como se ve en la figura 7.2, la varianza del primer componente principal es bastante grande (como suele ser el caso), pero los otros componentes principales que le siguen son significativos.

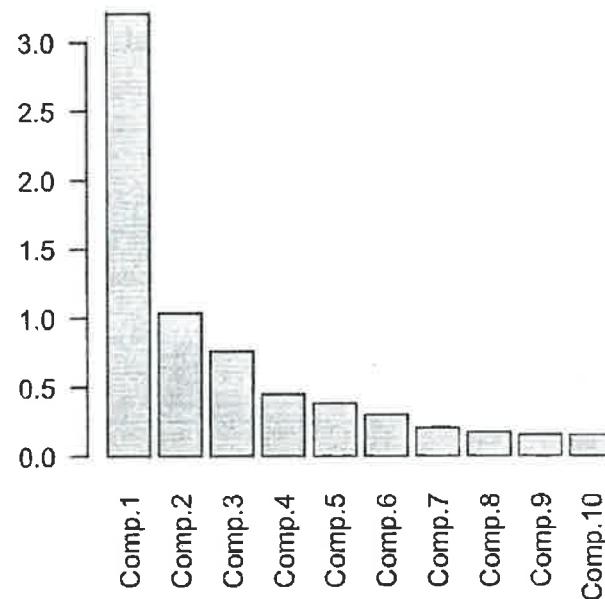


Figura 7.2 Diagrama de barras para el PCA de las principales acciones del S&P 500.

Puede resultar especialmente revelador reflejar en un diagrama las ponderaciones de los componentes principales superiores. Una forma de hacer esto en R es usar la función `gather` del paquete `tidyverse` junto con `ggplot`:

```
library(tidyverse)
loadings <- sp_pca$loadings[,1:5]
loadings$Symbol <- row.names(loadings)
loadings <- gather(loadings, 'Component', 'Weight', -Symbol)
ggplot(loadings, aes(x=Symbol, y=Weight)) +
```

Estadística práctica para ciencia de datos con R y Python

```
geom_bar(stat='identity') +  
facet_grid(Component ~ ., scales='free_y')
```

Aquí está el código para crear la misma visualización en *Python*:

```
loadings = pd.DataFrame(sp_pca.components_[0:5, :], columns=top_sp.columns)  
maxPC = 1.01 * np.max(np.max(np.abs(loadings.loc[0:5, :])))  
  
f, axes = plt.subplots(5, 1, figsize=(5, 5), sharex=True)  
for i, ax in enumerate(axes):  
    pc_loadings = loadings.loc[i, :]  
    colors = ['C0' if l > 0 else 'C1' for l in pc_loadings]  
    ax.axhline(color='#888888')  
    pc_loadings.plot.bar(ax=ax, color=colors)  
    ax.set_ylabel(f'PC{i+1}')  
    ax.set_ylim(-maxPC, maxPC)
```

Las cargas de los cinco componentes superiores se muestran en la figura 7.3. Las cargas para el primer componente principal tienen el mismo signo: esto es típico de los datos en los que todas las columnas comparten un factor común (en este caso, la tendencia general del mercado de valores). El segundo componente captura los cambios del precio de las acciones del sector energético en comparación con las otras acciones. El tercer componente es principalmente un contraste en los movimientos de Apple y CostCo. El cuarto componente contrasta los movimientos de Schlumberger (SLB) con las otras acciones del sector energético. Finalmente, el quinto componente está dominado mayoritariamente por empresas financieras.

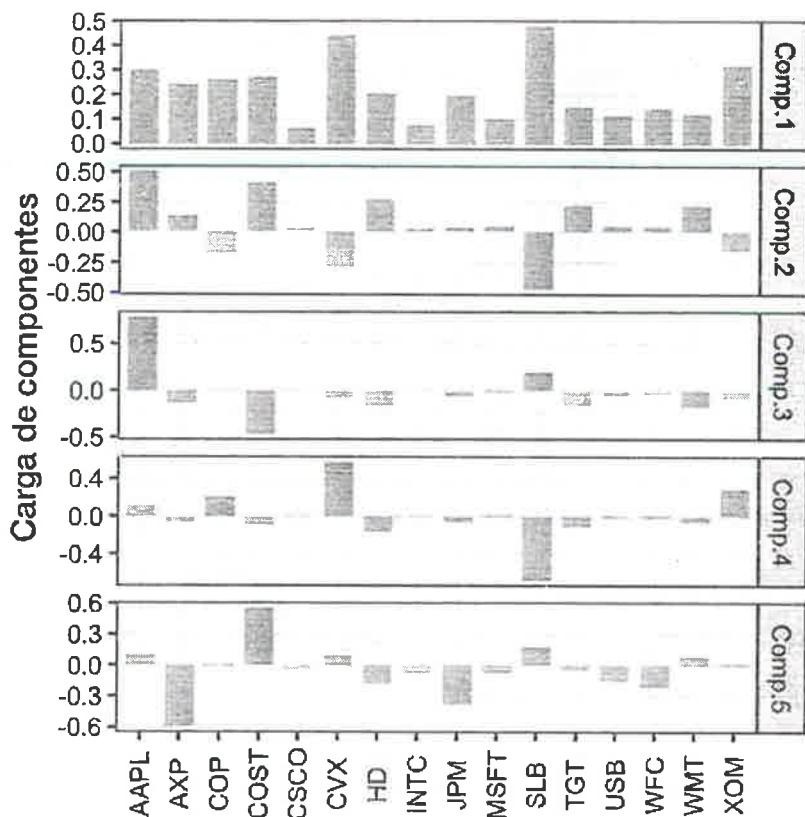


Figura 7.3 Cargas de los cinco componentes principales de la rentabilidad del precio de las acciones.



¿Cuántos componentes elegir?

Si nuestro objetivo es reducir la dimensión de los datos, debemos decidir cuántos componentes principales seleccionar. El planteamiento más frecuente es utilizar una regla *ad hoc* para seleccionar los componentes que explican "la mayor parte" de la varianza. Podemos hacer esto visualmente a través del diagrama de barras, como, por ejemplo, en la figura 7.2. Opcionalmente, podemos seleccionar los componentes superiores de manera que la varianza acumulada exceda un umbral, por ejemplo, el 80%. Además, podemos inspeccionar las cargas para determinar si el componente tiene una interpretación intuitiva. La validación cruzada proporciona un método más formal para seleccionar el número de componentes significativos (consultar "Validación cruzada" en la página 155 para obtener más información).

Análisis de correspondencias

PCA no se puede utilizar para datos categóricos. Sin embargo, una técnica relacionada de alguna manera es el *análisis de correspondencias* (*correspondence analysis*). El objetivo es reconocer asociaciones entre categorías o entre características categóricas. Las similitudes entre el análisis de correspondencias y el análisis de componentes principales son difíciles de apreciar a primera vista: el álgebra matricial para el escalado de dimensiones. El análisis de correspondencias se utiliza principalmente para el análisis gráfico de datos categóricos de baja dimensionalidad y no se utiliza de la misma forma que PCA para la reducción de dimensiones como paso preparatorio en proyectos de big data.

La entrada puede verse como una tabla, con las filas que representan una variable, las columnas otra, y las celdas representan recuentos de registros. El resultado (después de algo de álgebra matricial) es un *biplot*, un diagrama de dispersión con ejes escalados (y con porcentajes que indican la proporción de varianza que explica cada una de las dimensiones). El significado de las unidades en los ejes no está conectado intuitivamente con los datos originales, y el valor principal del diagrama de dispersión es ilustrar gráficamente las variables que están asociadas entre sí (por proximidad en el diagrama). Veamos, por ejemplo, la figura 7.4, en la que las tareas del hogar se ordenan según se realicen de forma conjunta o individual (eje vertical) y si la esposa o el marido tiene la máxima responsabilidad (eje horizontal). El análisis de correspondencias tiene muchas décadas, al igual que el espíritu de este ejemplo, a juzgar por la asignación de tareas.

Hay una variedad de paquetes para el análisis de correspondencias en R. Aquí, usamos el paquete ca:

```
ca_analysis <- ca(housetasks)
plot(ca_analysis)
```

En Python podemos usar el paquete prince, que implementa el análisis de correspondencias usando la API scikit-learn:

```
ca = prince.CA(n_components=2)
ca = ca.fit(housetasks)

ca.plot_coordinates(housetasks, figsize=(6, 6))
```

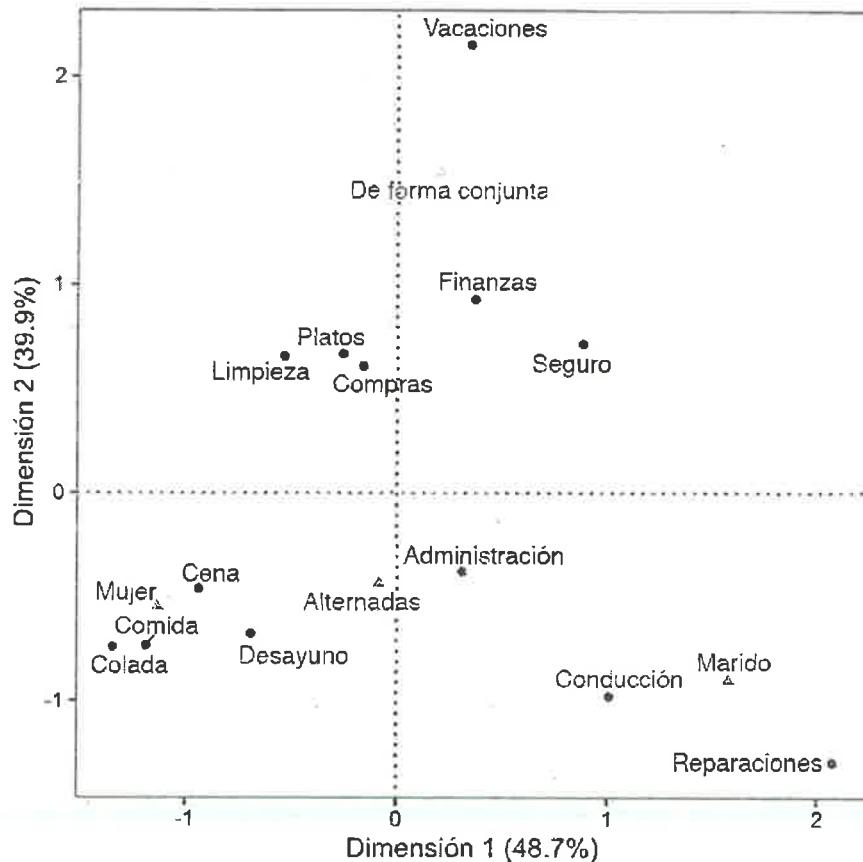


Figura 7.4 Representación gráfica de un análisis de correspondencias de datos de tareas domésticas.

Ideas clave

- Los componentes principales son combinaciones lineales de las variables predictoras (solo datos numéricos).
- Los componentes principales se calculan para minimizar la correlación entre componentes, reduciendo la redundancia.
- Un número limitado de componentes normalmente explicará la mayor parte de la varianza en la variable de resultado.
- El conjunto limitado de componentes principales se puede utilizar en lugar de las predictoras originales (más numerosas), reduciendo la dimensión.
- Una técnica algo similar para datos categóricos es el análisis de correspondencias, pero no es útil en un contexto de big data.

Lecturas complementarias

Para obtener una descripción detallada del uso de la validación cruzada en los componentes principales, consultar Rasmus Bro, K. Kjeldahl, A.K. Smilde y Henk A. L. Kiers, "Cross-Validation of Component Models: A Critical Look at Current Methods" (https://www.researchgate.net/publication/5638191_Cross-validation_of_component_models_A_critical_look_at_current_methods), *Analytical and Bioanalytical Chemistry* 390, n.º 5 (2008).

Agrupación K-means

La agrupación en clústeres es una técnica para dividir los datos en diferentes grupos, donde los registros de cada grupo son similares entre sí. Un objetivo de la agrupación en clústeres es identificar grupos de datos importantes y con sentido. Los grupos se pueden utilizar directamente, analizar con más profundidad o pasarlo como una característica o un resultado a un modelo de clasificación o regresión predictiva. *K-means* fue el primer método de agrupación que se desarrolló. Todavía se utiliza ampliamente, debido a su popularidad, a la relativa simplicidad del algoritmo y a su capacidad para escalar a grandes conjuntos de datos.

Términos clave de la agrupación K-means

Grupo

Grupo de registros que son similares.

Media de grupo

Vector de las medias de las variables de los registros de un grupo.

K

Número de grupos.

K-means divide los datos en K grupos, minimizando la suma de las distancias al cuadrado de cada registro a la *media* (*mean*) del grupo al que ha sido asignado. A esta operación se le conoce como *suma de cuadrados dentro del grupo* (*within-cluster sum of squares*) o *SS dentro del grupo* (*within-cluster SS*). K-means no garantiza que los grupos vayan a tener el mismo tamaño, pero busca los grupos que más separados están.



Normalización

Es típico normalizar (estandarizar) las variables continuas restando la media y dividiendo por la desviación estándar. De lo contrario, las variables de gran escala dominarán el proceso de agrupamiento (consultar "Estandarización [normalización, puntuación z]" en la página 243).

Un ejemplo sencillo

Empecemos por considerar un conjunto de datos con n registros y solo dos variables, x e y . Supongamos que queremos dividir los datos en $K = 4$ grupos. Esto significa asignar cada registro (x_i, y_i) a un grupo k . Dada una asignación de n_k registros al grupo k , el centro del grupo (\bar{x}_k, \bar{y}_k) es la media de los puntos del grupo:

$$\bar{x}_k = \frac{1}{n_k} \sum_{\substack{i \in \\ \text{Cluster } k}} x_i$$

$$\bar{y}_k = \frac{1}{n_k} \sum_{\substack{i \in \\ \text{Cluster } k}} y_i$$



Media de grupo

En la agrupación de registros con múltiples variables (el caso típico), el término media de grupo (*cluster mean*) se refiere no a un único número sino al vector de las medias de las variables.

La suma de cuadrados dentro de un grupo viene dada por:

$$SS_k = \sum_{i \in \text{Cluster } k} (x_i - \bar{x}_k)^2 + (y_i - \bar{y}_k)^2$$

K-means encuentra la asignación de registros que minimiza la suma de cuadrados dentro del grupo en los cuatro grupos $SS_1 + SS_2 + SS_3 + SS_4$:

$$\sum_{k=1}^4 SS_k$$

Un uso típico de la agrupación en clústeres es localizar, en los datos, grupos separados de forma natural. Otra aplicación es dividir los datos en un número predeterminado de grupos separados, donde se utiliza la agrupación para garantizar que los grupos sean lo más diferentes posible entre sí.

Por ejemplo, supongamos que queremos dividir las rentabilidades diarias de las acciones en cuatro grupos. La agrupación K-means se puede utilizar para separar los datos en las mejores agrupaciones. Hay que tener en cuenta que las rentabilidades diarias de las acciones se informan de una manera que, de hecho, está estandarizada, por lo que no es necesario normalizar los datos. En R, la agrupación K-means se puede realizar utilizando la función `kmeans`. Por ejemplo, lo que viene a continuación encuentra cuatro grupos basados en dos variables: las rentabilidades diarias de las acciones de ExxonMobil (XOM) y Chevron (CVX):

```
df <- sp500_px[row.names(sp500_px)>='2011-01-01', c('XOM', 'CVX')]
km <- kmeans(df, centers=4)
```

En *Python* utilizamos el método `sklearn.cluster.KMeans` de *scikit-learn*:

```
df = sp500_px.loc[sp500_px.index >= '2011-01-01'];
['XOM', 'CVX']]kmeans = KMeans(n_clusters=4).fit(df)
```

La asignación de grupo para cada registro se devuelve como el componente `cluster` (*R*):

```
> df$cluster <- factor(km$cluster)
> head(df)
      XOM      CVX cluster
2011-01-03 0.73680496  0.2406809    2
2011-01-04 0.16866845 -0.5845157    1
2011-01-05 0.02663055  0.4469854    2
2011-01-06 0.24855834 -0.9197513    1
2011-01-07 0.33732892  0.1805111    2
2011-01-10 0.00000000 -0.4641675    1
```

En *scikit-learn*, las etiquetas del grupo están disponibles en el campo `labels_`:

```
df['cluster'] = kmeans.labels_
df.head()
```

Los primeros seis registros se asignan al grupo 1 o al grupo 2. También devuelve las medias de los grupos (*R*):

```
> centers <- data.frame(cluster=factor(1:4), km$centers)
> centers
  cluster      XOM      CVX
1       1 -0.3284864 -0.5669135
2       2  0.2410159  0.3342130
3       3 -1.1439800 -1.7502975
4       4  0.9568628  1.3708892
```

En *scikit-learn*, los centros de los grupos están disponibles en el campo `cluster_centers_`:

```
centers = pd.DataFrame(kmeans.cluster_centers_, columns=['XOM', 'CVX'])
centers
```

Los grupos 1 y 3 representan mercados "a la baja", mientras que los grupos 2 y 4 representan "mercados al alza".

Como el algoritmo K-means utiliza puntos de partida aleatorios, los resultados pueden diferir entre subsecuentes ejecuciones y diferentes implementaciones del método. En general, debemos verificar que las fluctuaciones no sean demasiado grandes.

En este ejemplo, con solo dos variables, es sencillo visualizar los grupos y sus medias:

```
ggplot(data=df, aes(x=XOM, y=CVX, color=cluster, shape=cluster)) +
  geom_point(alpha=.3) +
  geom_point(data=centers, aes(x=XOM, y=CVX), size=3, stroke=2)
```

La función `scatterplot` de *seaborn* facilita el color (hue) y el estilo (style) de los puntos mediante una propiedad:

```
fig, ax = plt.subplots(figsize=(4, 4))
ax = sns.scatterplot(x='XOM', y='CVX', hue='cluster', style='cluster',
                      ax=ax, data=df)
ax.set_xlim(-3, 3)
ax.set_ylim(-3, 3)
centers.plot.scatter(x='XOM', y='CVX', ax=ax, s=50, color="black")
```

El gráfico resultante, que se muestra en la figura 7.5, presenta las asignaciones y las medias de los grupos. Hay que tener en cuenta que K-means asignará registros a grupos, incluso si esos grupos no están bien separados (lo que puede ser útil si necesitamos separar los registros en grupos de manera óptima).

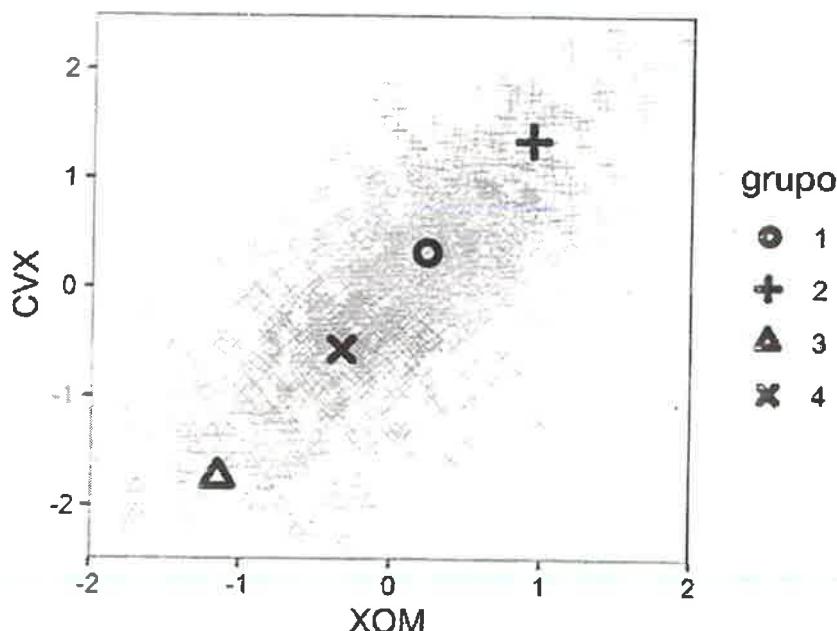


Figura 7.5 Grupos de K-means aplicados a la rentabilidad diaria de las acciones de ExxonMobil y Chevron (los centros de los grupos están resaltados con símbolos de color negro).

Algoritmo K-means

En general, K-means se pueden aplicar a un conjunto de datos con p variables, X_1, \dots, X_p . Si bien la solución exacta de K-means es muy difícil desde el punto de vista del cálculo, los algoritmos heurísticos proporcionan una forma eficiente de calcular una solución localmente óptima.

El algoritmo comienza con una K especificada por el usuario y un conjunto inicial de las medias de los grupos y luego itera los siguientes pasos:

1. Asignamos cada registro al grupo con la media más cercana medida por la distancia al cuadrado.
2. Calculamos las nuevas medias del grupo en función de la asignación de los registros.

El algoritmo converge cuando la asignación de registros a los grupos no cambia.

Para la primera iteración, debemos especificar un conjunto inicial de las medias de los grupos. Por lo general, hacemos esto asignando aleatoriamente cada registro a uno de los K grupos y luego encontramos las medias de esos grupos.

Dado que no se garantiza que este algoritmo encuentre la mejor solución posible, se recomienda ejecutarlo varias veces utilizando diferentes muestras aleatorias para inicializar el algoritmo. Cuando se usa más de un conjunto de iteraciones, el resultado de K-means viene dado por la iteración que tiene la suma de cuadrados más baja dentro de cada grupo.

El parámetro `nstart` de la función `kmeans` de R nos permite especificar el número de inicios aleatorios que se van a intentar. Por ejemplo, el siguiente código ejecuta K-means para encontrar 5 grupos utilizando 10 medias iniciales de grupos diferentes:

```
syms <- c( 'AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM', 'SLB', 'COP',
         'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST')
df <- sp500_px[row.names(sp500_px) >= '2011-01-01', syms]
km <- kmeans(df, centers=5, nstart=10)
```

La función proporciona automáticamente la mejor solución de los 10 puntos de partida diferentes. Podemos utilizar el argumento `iter.max` para establecer el número máximo de iteraciones que se permite al algoritmo para cada inicio aleatorio.

El algoritmo `scikit-learn` se repite 10 veces de forma predeterminada (`n_init`). El argumento `max_iter` (predeterminado 300) se puede usar para controlar el número de iteraciones:

```
syms = sorted(['AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM', 'SLB', 'COP',
              'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST'])
top_sp = sp500_px.loc[sp500_px.index >= '2011-01-01', syms]
kmeans = KMeans(n_clusters=5).fit(top_sp)
```

Interpretación de los grupos

Una parte importante del análisis de grupos puede involucrar la interpretación de los grupos. Los dos resultados más importantes de `kmeans` son el tamaño y la media de cada grupo. Para el ejemplo de la subsección anterior, los tamaños de los grupos resultantes vienen dados por el siguiente comando de R:

```
km$size  
[1] 106 186 285 288 266
```

En `Python`, podemos usar la clase `collections.Counter` de la biblioteca estándar para obtener esta información. Debido a las diferencias en la implementación y la aleatoriedad inherente del algoritmo, los resultados variarán:

```
from collections import Counter  
Counter(kmeans.labels_)  
Counter({4: 302, 2: 272, 0: 288, 3: 158, 1: 111})
```

Estadística práctica para ciencia de datos con R y Python

Los tamaños de los grupos están relativamente equilibrados. Los grupos desequilibrados pueden ser el resultado de valores atípicos distantes o de grupos de registros muy distintos del resto de los datos; ambas causas pueden justificar una inspección adicional.

Podemos trazar los centros de los grupos utilizando la función `gather` junto con `ggplot`:

```
centers <- as.data.frame(t(centers))
names(centers) <- paste("Cluster", 1:5)
centers$Symbol <- row.names(centers)
centers <- gather(centers, 'Cluster', 'Mean', -Symbol)
centers$Color = centers$Mean > 0
ggplot(centers, aes(x=Symbol, y=Mean, fill=Color)) +
  geom_bar(stat='identity', position='identity', width=.75) +
  facet_grid(Cluster ~ ., scales='free_y')
```

El código para crear esta visualización en *Python* es similar al que usamos para PCA:

```
centers = pd.DataFrame(kmeans.cluster_centers_, columns=syms)

f, axes = plt.subplots(5, 1, figsize=(5, 5), sharex=True)
for i, ax in enumerate(axes):
    center = centers.loc[i, :]
    maxPC = 1.01 * np.max(np.max(np.abs(center)))
    colors = ['C0' if l > 0 else 'C1' for l in center]
    ax.axhline(color="#888888")
    center.plot.bar(ax=ax, color=colors)
    ax.set_ylabel(f'Cluster {i + 1}')
    ax.set_ylim(-maxPC, maxPC)
```

El gráfico resultante se muestra en la figura 7.6 y revela la naturaleza de cada grupo. Por ejemplo, los grupos 4 y 5 corresponden a días en los que el mercado está a la baja y al alza, respectivamente. Los grupos 2 y 3 se caracterizan por días de mercado alcista para las acciones de bienes de consumo y días de mercado a la baja para las acciones del sector energético, respectivamente. Por último, el grupo 1 captura los días en los que las existencias del sector energético subieron y las de bienes de consumo bajaron.



Análisis de grupos frente a PCA

El diagrama de las medias de los grupos es similar en espíritu a observar las cargas para el análisis de componentes principales (PCA). Consultar “Interpretación de componentes principales” en la página 289. Una distinción importante es que, a diferencia de PCA, el signo de las medias del grupo es significativo. PCA identifica las principales direcciones de variación, mientras que el análisis de grupos encuentra grupos de registros ubicados uno cerca del otro.

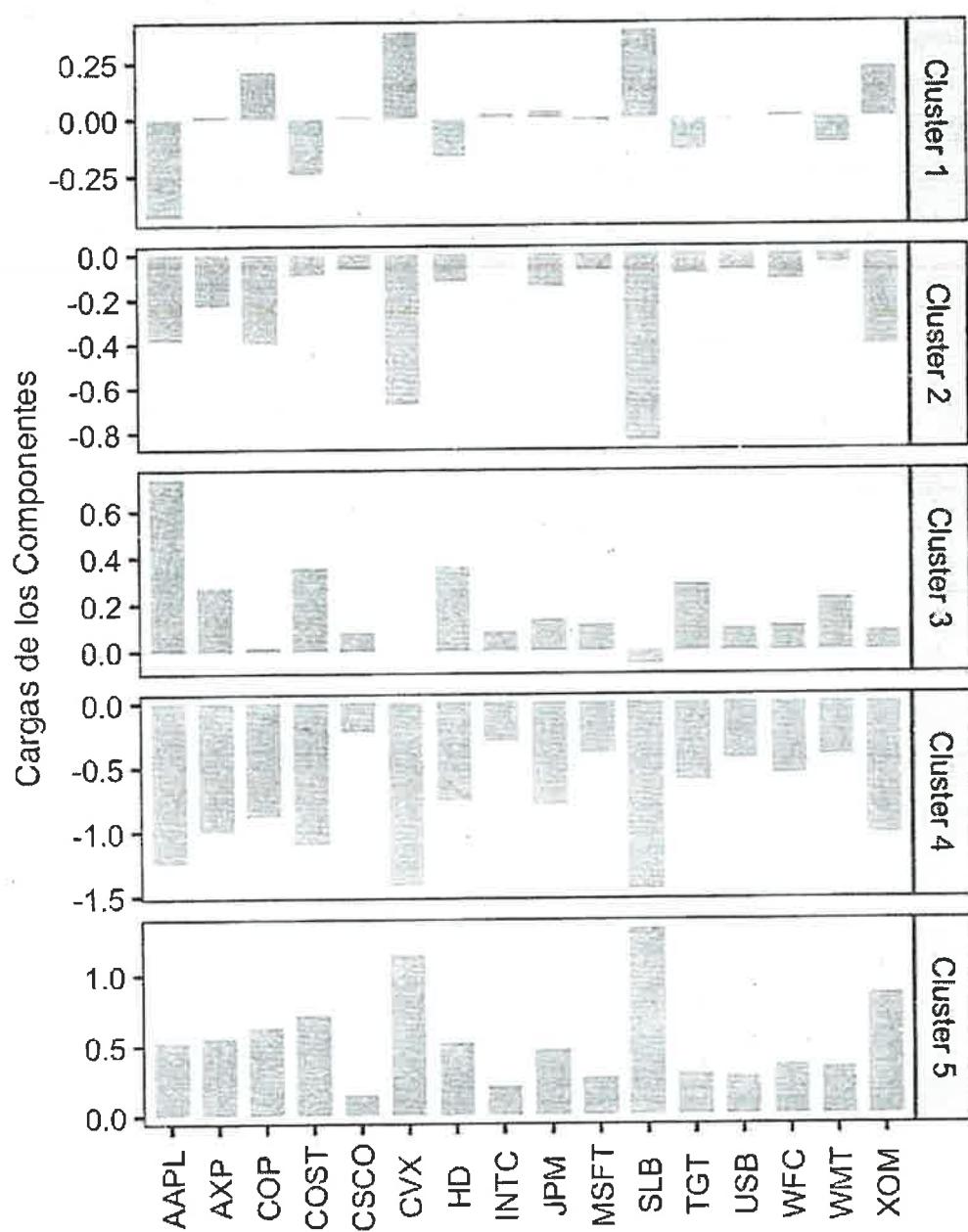


Figura 7.6 Las medias de las variables en cada grupo ("medias de grupo").

Selección del número de grupos

El algoritmo K-means requiere que especifiquemos el número K de grupos. A veces, es la aplicación la que decide este número. Por ejemplo, una empresa que gestiona un departamento comercial puede querer agrupar a los clientes en "personas" para enfocar y orientar las llamadas de ventas. En tal caso, las consideraciones administrativas dictarían el número de segmentos deseados de clientes. Por ejemplo, dos podrían no producir una diferenciación útil de clientes, mientras que ocho podrían ser demasiados para administrarlos.

Estadística práctica para ciencia de datos con R y Python

En ausencia de un número de grupos dictado por consideraciones prácticas o de gestión, se podría utilizar un enfoque estadístico. No existe un método estándar único para encontrar el "mejor" número de grupos.

Un enfoque habitual, llamado método del codo (*elbow method*), es identificar cuándo el conjunto de grupos explica "la mayor parte" de la varianza en los datos. Agregar nuevos grupos más allá de este conjunto contribuye relativamente poco a la varianza explicada. El codo es el punto donde la varianza explicada acumulada se aplana después de subir abruptamente, de ahí el nombre del método.

La figura 7.7 muestra el porcentaje acumulado de varianza explicada para las acciones para un número de grupos que van de 2 a 14. ¿Dónde está el codo en este ejemplo? No hay un candidato obvio, ya que el aumento incremental de la varianza explicada cae gradualmente. Esto es bastante típico en los datos que no tienen grupos bien definidos. Este es quizás un inconveniente del método del codo, pero revela la naturaleza de los datos.

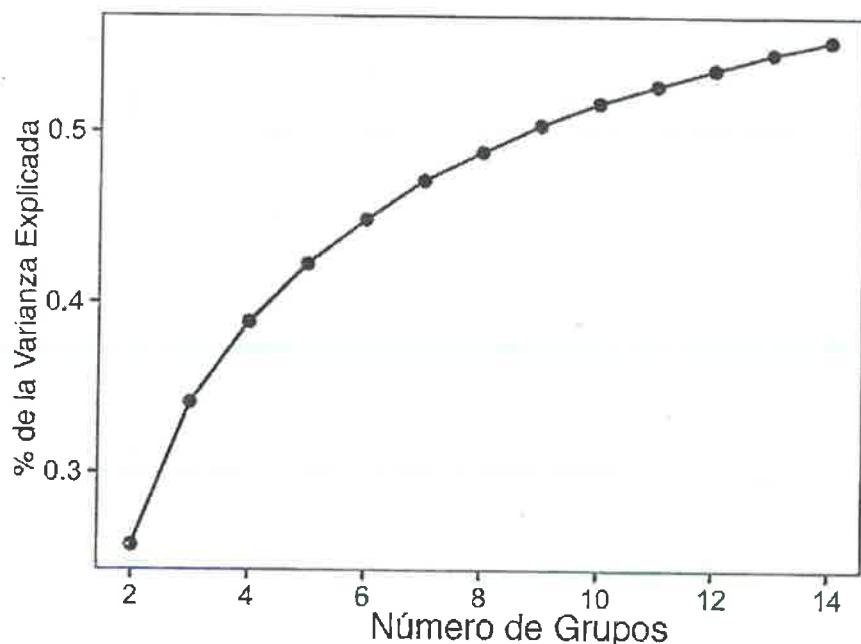


Figura 7.7 Método del codo aplicado a los datos de las acciones.

En *R*, la función kmeans no proporciona ningún comando para aplicar el método del codo, pero se puede aplicar fácilmente a partir del resultado de kmeans como se muestra aquí:

```
pct_var <- data.frame(pct_var = 0,
                      num_clusters = 2:14)
totalss <- kmeans(df, centers=14, nstart=50, iter.max=100)$totss
for (i in 2:14) {
  kmCluster <- kmeans(df, centers=i, nstart=50, iter.max=100)
  pct_var[i-1, 'pct_var'] <- kmCluster$betweenss / totalss
}
```

Para el resultado de Kmeans, obtenemos esta información de la propiedad `inertia_`. Después de la conversión a un marco de datos de pandas, podemos usar su método `plot` para crear el gráfico:

```

inertia = []
for n_clusters in range(2, 14):
    kmeans = KMeans(n_clusters=n_clusters,
                    random_state=0).fit(top_sp)
    inertia.append(kmeans.inertia_ / n_clusters)

inertias = pd.DataFrame({'n_clusters': range(2, 14), 'inertia': inertia})
ax = inertias.plot(x='n_clusters', y='inertia')
plt.xlabel('Number of clusters(k)')
plt.ylabel('Average Within-Cluster Squared Distances')
plt.ylim((0, 1.1 * inertias.inertia.max()))
ax.legend().set_visible(False)

```

Al evaluar cuántos grupos retener, quizá la prueba más importante sea la siguiente: ¿qué probabilidad hay de que los grupos se repliquen en nuevos datos? ¿Los grupos son interpretables y se relacionan con una característica general de los datos o simplemente reflejan una instancia específica? Podemos evaluar esto, en parte, mediante validación cruzada; consultar “Validación cruzada” en la página 155.

En general, no existe una regla única que oriente de manera confiable sobre cuántos grupos crear.



Hay varias formas más formales de determinar el número de conglomerados basados en la teoría estadística o de la información. Por ejemplo, Robert Tibshirani, Guenther Walther y Trevor Hastie proponen una estadística de "brecha" (<https://web.stanford.edu/~hastie/Papers/gap.pdf>) basada en la teoría estadística para identificar el codo. Para la mayoría de las aplicaciones, un enfoque teórico probablemente no sea necesario, ni siquiera apropiado.

Ideas clave

- El número K de clústeres deseados lo elige el usuario.
- El algoritmo desarrolla agrupaciones asignando registros iterativamente a la media de grupo más cercana hasta que las asignaciones de agrupaciones no cambien.
- Las consideraciones prácticas suelen dominar la elección de K ; no hay un número óptimo de grupos determinado estadísticamente.

Agrupación jerárquica

La *agrupación jerárquica* (*hierarchical clustering*) es una alternativa a K-means que pueden producir grupos muy diferentes. La agrupación jerárquica permite al usuario visualizar el efecto de especificar diferentes números de agrupaciones. Es más sensible a la hora de descubrir grupos o registros distantes o aberrantes. La agrupación jerárquica también se presta a una visualización gráfica intuitiva, lo que facilita la interpretación de los grupos.

Términos clave de la agrupación jerárquica

Dendrograma

Representación visual de los registros y la jerarquía de los grupos a los que pertenecen.

Distancia

Medida de lo cerca que está un *registro* de otro.

Disimilitud

Medida de lo cerca que está un *grupo* de otro.

La flexibilidad de la agrupación jerárquica tiene un coste, y la agrupación jerárquica no se adapta bien a grandes conjuntos de datos con millones de registros. Incluso para datos de tamaño modesto con solo decenas de miles de registros, la agrupación jerárquica puede requerir recursos informáticos intensivos. De hecho, la mayoría de las aplicaciones de la agrupación jerárquica se centran en conjuntos de datos relativamente pequeños.

Un ejemplo sencillo

La agrupación jerárquica funciona en un conjunto de datos con n registros y p variables y se basa en dos bloques de construcción básicos:

- Una métrica de distancia $d_{i,j}$ para medir la distancia entre dos registros i y j .
- Una métrica de disimilitud $D_{A,B}$ para medir la diferencia entre dos grupos A y B en función de las distancias $d_{i,j}$ entre los miembros de cada grupo.

Para aplicaciones que involucran datos numéricos, la opción más importante es la métrica de disimilitud. La agrupación jerárquica comienza estableciendo cada registro como su propio grupo y se repite para combinar los grupos menos diferentes.

En R, la función `hclust` se puede utilizar para realizar agrupaciones jerárquicas. Una gran diferencia de `hclust` frente `kmeans` es que opera con las distancias por pares $d_{i,j}$ en lugar de con los datos en sí. Podemos calcularlas usando la función `dist`. Por ejemplo,

lo siguiente aplica la agrupación jerárquica a la rentabilidad de las acciones de un conjunto de empresas:

```
syms1 <- c('GOOGL', 'AMZN', 'AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM', 'SLB',
         'COP', 'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST')
# take transpose: to cluster companies, we need the stocks along the rows
df <- t(sp500_px[row.names(sp500_px) >= '2011-01-01', syms1])
d <- dist(df)
hcl <- hclust(d)
```

Los algoritmos de agrupamiento agruparán los registros (filas) del marco de datos. Dado que queremos agrupar las empresas, necesitamos transponer (*t*) el marco de datos y colocar las acciones a lo largo de las filas y las fechas a lo largo de las columnas.

El paquete `scipy` ofrece varios métodos diferentes para la agrupación jerárquica en el módulo `scipy.cluster.hierarchy`. Aquí usamos la función `linkage` con el método "completo":

```
syms1 = ['AAPL', 'AMZN', 'AXP', 'COP', 'COST', 'CSCO', 'CVX', 'GOOGL', 'HD',
         'INTC', 'JPM', 'MSFT', 'SLB', 'TGT', 'USB', 'WFC', 'WMT', 'XOM']
df = sp500_px.loc[sp500_px.index >= '2011-01-01', syms1].transpose()

Z = linkage(df, method='complete')
```

El dendrograma

La agrupación jerárquica se presta a una visualización gráfica natural como un árbol, denominado *dendrograma* (*dendrogram*). El nombre proviene de las palabras griegas *dendro* (árbol) y *gramma* (dibujo). En R, podemos crearlo fácilmente usando el comando `plot`:

```
plot(hcl)
```

Podemos utilizar el método `dendrogram` para hacer un gráfico del resultado de la función `linkage` en Python:

```
fig, ax = plt.subplots(figsize=(5, 5))
dendrogram(Z, labels=list(df.index), color_threshold=0)
plt.xticks(rotation=90)
ax.set_ylabel('distance')
```

El resultado se muestra en la figura 7.8 (hay que tener en cuenta que es en este instante, no hace días, cuando hacemos un gráfico de empresas que son similares entre sí). Las hojas del árbol corresponden a los registros. La longitud de la rama en el árbol indica el grado de disimilitud entre los grupos correspondientes. Las rentabilidades de Google y Amazon son bastante diferentes entre sí y entre las rentabilidades de las otras acciones. Las acciones del petróleo (SLB, CVX, XOM, COP) están en su grupo, Apple (AAPL) está sola y el resto son similares entre sí.

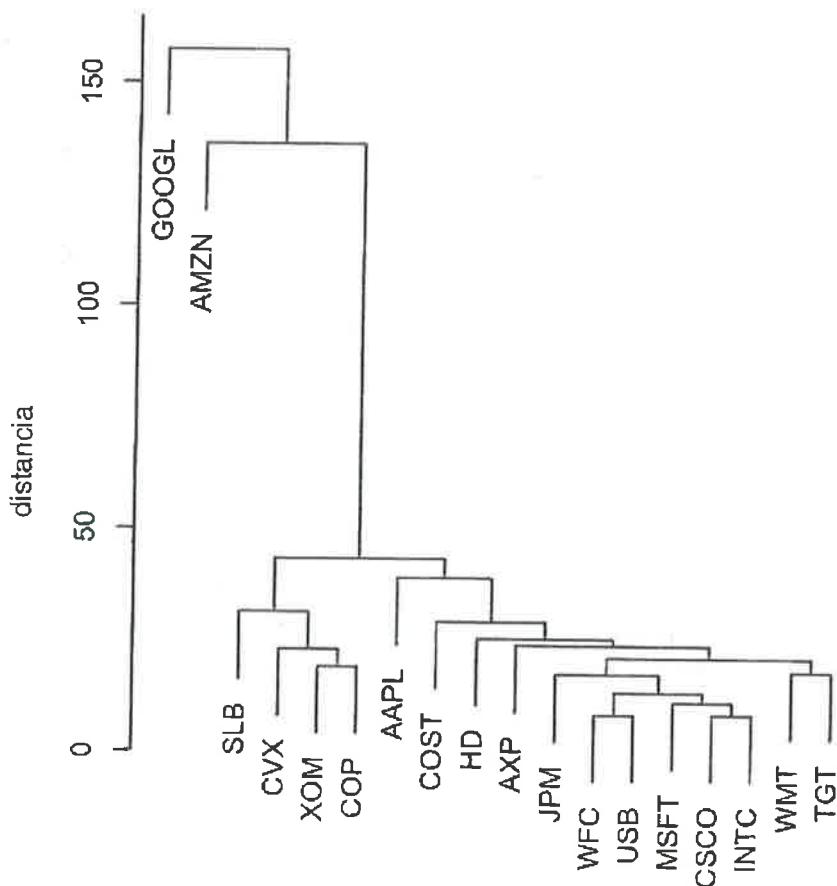


Figura 7.8 Dendrograma de algunas acciones.

A diferencia de K-means, no es necesario especificar previamente el número de grupos. Gráficamente, podemos identificar diferentes números de grupos mediante una línea horizontal que se desliza hacia arriba o hacia abajo. Un grupo se define donde la línea horizontal se cruza con las líneas verticales. Para extraer un número específico de grupos, podemos utilizar la función `cutree`:

```
cutree(hcl, k=4)
GOOGL AMZN AAPL MSFT CSCCO INTC CVX XOM SLB COP JPM WFC
1 2 3 3 3 3 4 4 4 4 4 3 3
USB AXP WMT TGT HD COST
3 3 3 3 3 3
```

En *Python* logramos lo mismo con el método `fcluster`:

```
memb = fcluster(Z, 4, criterion='maxclust')
memb = pd.Series(memb, index=df.index)
for key, item in memb.groupby(memb):
    print(f'{key} : {", ".join(item.index)}')
```

El número de grupos a extraer se establece en 4 y podemos ver que Google y Amazon pertenecen a su propio grupo. Todas las acciones de petróleo pertenecen a otro grupo. Las poblaciones restantes están en el cuarto grupo.

El algoritmo de aglomeración

El algoritmo principal para la agrupación jerárquica es el algoritmo de *aglomeración* (*agglomerative*), que fusiona de forma iterativa agrupaciones similares. El algoritmo de *aglomeración* (*agglomerative*) comienza con cada registro que constituye su propio grupo de registro único y luego crea grupos cada vez más grandes. El primer paso es calcular las distancias entre todos los pares de registros.

Para cada par de registros (x_1, x_2, \dots, x_p) , e (y_1, y_2, \dots, y_p) , medimos la distancia entre los dos registros, $d_{x,y}$, utilizando una métrica de distancia (consultar "Métricas de distancia" en la página 241). Por ejemplo, podemos usar la distancia euclidiana:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_p - y_p)^2}$$

Pasamos ahora a la distancia entre grupos. Consideremos dos grupos A y B, cada uno con un conjunto distintivo de registros, $A = (a_1, a_2, \dots, a_m)$ y $B = (b_1, b_2, \dots, b_q)$. Podemos medir la disimilitud entre los grupos $D(A, B)$ usando las distancias entre los miembros de A y los miembros de B.

Una medida de disimilitud es el método de *vinculación completa* (*complete-linkage*), que es la distancia máxima entre todos los pares de registros entre A y B:

$$D(A, B) = \max d(a_i, b_j) \text{ para todos los pares } i, j$$

Esta expresión define la disimilitud como la mayor diferencia entre todos los pares.

Los principales pasos del algoritmo de aglomeración son los siguientes:

1. Creamos un conjunto inicial de grupos en el que cada grupo consta de un solo registro para todos los registros de los datos.
2. Calculamos la disimilitud $D(C_k, C_\ell)$ entre todos los pares de grupos k, ℓ .
3. Fusionamos los dos grupos C_k y C_ℓ que sean menos diferentes según lo medido por $D(C_k, C_\ell)$.
4. Si nos queda más de un grupo, regresamos al paso 2. De lo contrario, hemos terminado.

Medidas de disimilitud

Hay cuatro medidas habituales de disimilitud: *vinculación completa* (*complete linkage*), *vinculación simple* (*single linkage*), *vinculación promedio* (*average linkage*) y *varianza mínima* (*minimum variance*). Estas (más otras medidas) están respaldadas por la mayoría del software de agrupación jerárquica, incluidos `hclust` y `linkage`. El método de vinculación completa definido anteriormente tiende a producir grupos con miembros similares. El método de vinculación simple es la distancia mínima entre los registros en dos grupos:

$$D(A, B) = \min d(a_i, b_j) \text{ para todos los pares } i, j$$

Este es un método "codicioso" y produce grupos que pueden contener elementos bastante dispares. El método de vinculación promedio es el promedio de todos los pares de distancias y representa un compromiso entre los métodos de vinculación simple y completa. Por último, el método de varianza mínima, también conocido como método de *Ward*, es similar a K-means, ya que minimiza la suma de cuadrados dentro del grupo (consultar "Agrupación K-means" en la página 294).

La figura 7.9 aplica la agrupación jerárquica utilizando las cuatro medidas para las rentabilidades de las acciones de ExxonMobil y Chevron. Para cada medida, se retienen cuatro grupos.

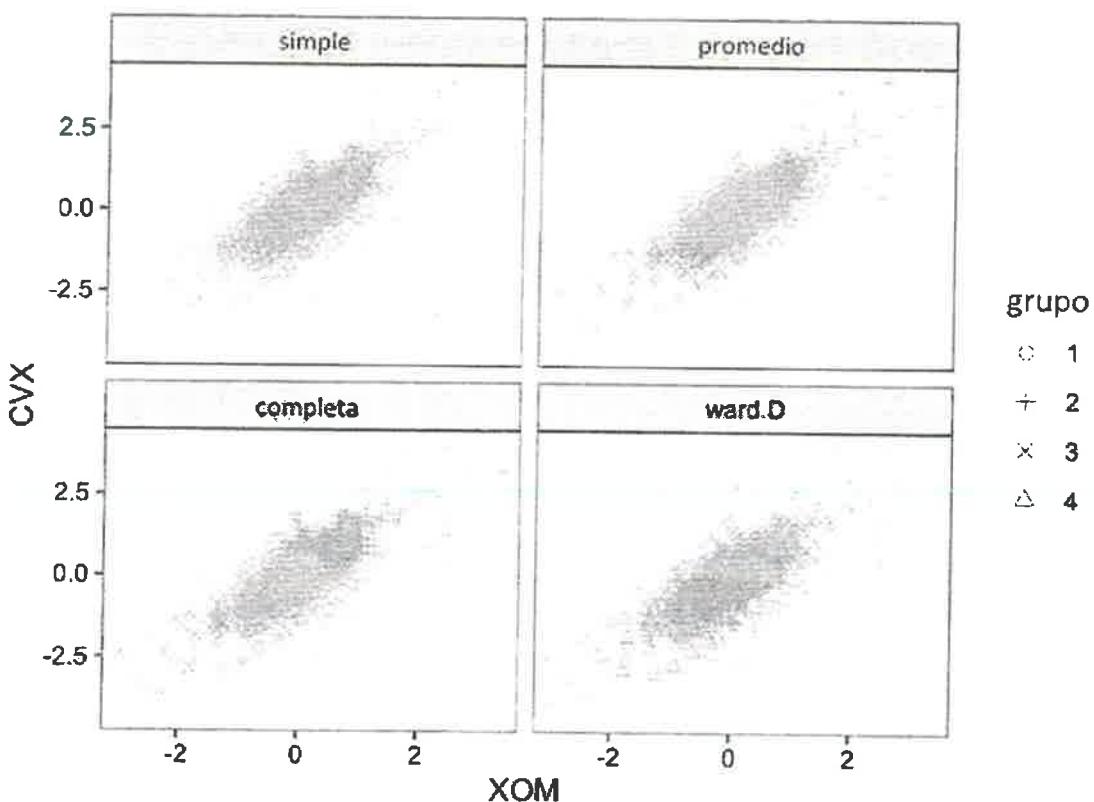


Figura 7.9 Comparación de medidas de disimilitud aplicadas a los datos de stock.

Los resultados son sorprendentemente diferentes: la medida de vinculación única asigna casi todos los puntos a un solo grupo. Excepto por el método de varianza mínima (R: *Ward.D*; Python: *ward*), todas las medidas terminan con al menos un grupo con solamente unos pocos puntos periféricos. El método de varianza mínima es más similar al grupo de K-means. Compararlo con la figura 7.5.

Ideas clave

- La agrupación jerárquica comienza con cada registro en su propio grupo.
- Progresivamente, los grupos se unen a los grupos cercanos hasta que todos los registros pertenecen a un solo grupo (algoritmo de aglomeración).
- El historial de aglomeraciones se conserva y se traza, y el usuario (sin especificar el número de grupos de antemano) puede visualizar el número y la estructura de los grupos en diferentes etapas.
- Las distancias entre grupos se calculan de diferentes formas, todas basándose en el conjunto de todas las distancias entre registros.

Agrupación basada en el modelo

Los métodos de agrupación, como la agrupación jerárquica y K-means, se basan en la heurística y su función principalmente es la búsqueda de grupos cuyos miembros estén cerca unos de otros, medidos directamente con los datos (sin modelo de probabilidad involucrado). En los últimos 20 años, se ha dedicado un esfuerzo importante al desarrollo de métodos de *agrupamiento basado en el modelo* (*model-based clustering*). Adrian Raftery y otros investigadores de la Universidad de Washington hicieron contribuciones críticas a la agrupación en grupos basada en modelos, incluida tanto la teoría como el software. Las técnicas se basan en la teoría estadística y proporcionan formas más rigurosas de determinar la naturaleza y el número de grupos. Podrían usarse, por ejemplo, en casos en los que pudiera haber un grupo de registros similares entre sí, pero no necesariamente cercanos entre sí (por ejemplo, acciones tecnológicas con alta variación de rentabilidades), y otro grupo de registros que son similares y también cercanos (por ejemplo, acciones de servicios públicos con baja variación).

Distribución normal multivariante

Los métodos más utilizados de agrupación basados en modelos tienen como base la distribución *normal multivariante* (*multivariate normal*). La distribución normal multivariante es una generalización de la distribución normal a un conjunto de p variables X_1, X_2, \dots, X_p . La distribución está definida por un conjunto de medias $\mu = \mu_1, \mu_2, \dots, \mu_p$ y una matriz de covarianza Σ . La matriz de covarianza es una medida de cómo las variables se correlacionan entre sí (consultar “Matriz de covarianza” en la página 202 para ampliar detalles sobre la covarianza). La matriz de covarianzas Σ consta de p varianzas $\sigma_1^2, \sigma_2^2, \dots, \sigma_p^2$ y covarianzas $\sigma_{i,j}$ para todos los pares de variables $i \neq j$. Con las variables colocadas a lo largo de las filas y duplicadas a lo largo de las columnas, la matriz se ve así:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} & \cdots & \sigma_{1,p} \\ \sigma_{2,1} & \sigma_2^2 & \cdots & \sigma_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p,1} & \sigma_{p,2}^2 & \cdots & \sigma_p^2 \end{bmatrix}$$

Hay que tener en cuenta que la matriz de covarianza es simétrica a ambos lados de la diagonal que va desde la parte superior izquierda a la inferior derecha. Dado que $\sigma_{i,j} = \sigma_{j,i}$, solo hay $(p \times p - 1)/2$ términos de covarianza. En total, la matriz de covarianza tiene $(p \times (p - 1))/2 + p$ parámetros. La distribución se denota por:

$$(X_1, X_2, \dots, X_p) \sim N_p(\mu, \Sigma)$$

Esta es una forma simbólica de decir que todas las variables están distribuidas normalmente y que la distribución general está completamente descrita por el vector de medias de las variables y la matriz de covarianza.

La figura 7.10 muestra los contornos de probabilidad para una distribución normal multivariante para dos variables X e Y (el contorno de probabilidad 0.5, por ejemplo, contiene el 50% de la distribución).

Las medias son $\mu_x = 0.5$ y $\mu_y = -0.5$, y la matriz de covarianza es:

$$\Sigma = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$$

Dado que la covarianza σ_{xy} es positiva, X e Y están correlacionadas positivamente.

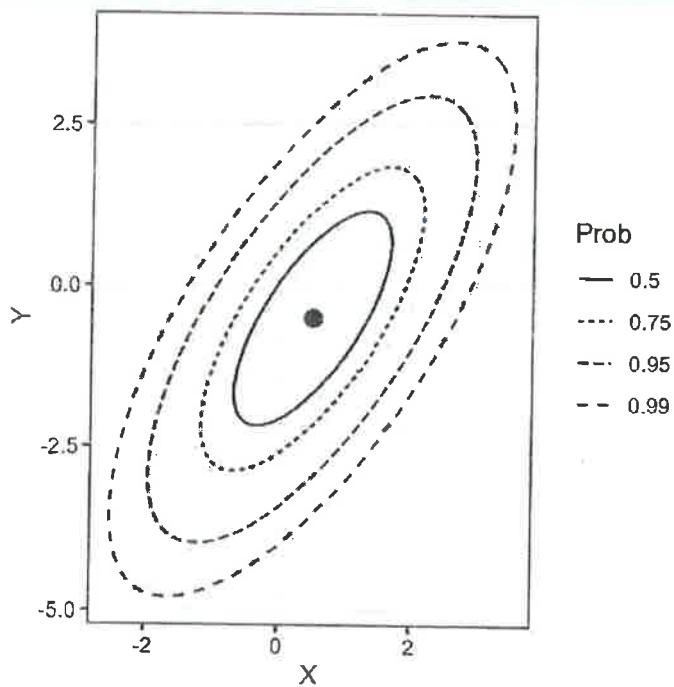


Figura 7.10 Contornos de probabilidad para una distribución normal bidimensional.

Mezclas de distribuciones normales

La idea clave detrás de la agrupación basada en el modelo es que se supone que cada registro se distribuye en una de las K distribuciones normales multivariantes, donde K es el número de agrupaciones. Cada distribución tiene una media μ diferente y una matriz de covarianza Σ . Por ejemplo, si tenemos dos variables, X e Y , entonces cada fila (X_i, Y_i) se modela como una muestra de una de las K distribuciones normales multivariantes $N(\mu_1, \Sigma_1), N(\mu_2, \Sigma_2), \dots, N(\mu_K, \Sigma_K)$.

R tiene un paquete muy rico de agrupación basada en el modelo llamado `mclust`, desarrollado originalmente por Chris Fraley y Adrian Raftery. Con este paquete, podemos aplicar la agrupación basada en el modelo a los datos de rentabilidad de las acciones que analizamos anteriormente utilizando K-means y la agrupación jerárquica:

```
> library(mclust)
> df <- sp500_px[row.names(sp500_px) >= '2011-01-01', c('XOM', 'CVX')]
> mcl <- Mclust(df)
> summary(mcl)
Mclust VEE (ellipsoidal, equal shape and orientation) model with 2 components:

log.likelihood    n      df      BIC      ICL
-2255.134   1131      9 -4573.546 -5076.856

Clustering table:
 1 2
963 168
```

`scikit-learn` tiene la clase `sklearn.mixture.GaussianMixture` para la agrupación basada en el modelo:

```
df = sp500_px.loc[sp500_px.index >= '2011-01-01', ['XOM', 'CVX']]
mclust = GaussianMixture(n_components=2).fit(df)
mclust.bic(df)
```

Si ejecutamos este código, notaremos que el cálculo lleva mucho más tiempo que otros procedimientos. Si extraemos las asignaciones de grupos usando la función `predict`, podemos visualizar los grupos:

```
cluster <- factor(predict(mcl)$classification)
ggplot(data=df, aes(x=XOM, y=CVX, color=cluster, shape=cluster)) +
  geom_point(alpha=.8)
```

A continuación se detalla el código de *Python* para crear una figura similar:

```
fig, ax = plt.subplots(figsize=(4, 4))
colors = [f'C{c}' for c in mclust.predict(df)]
df.plot.scatter(x='XOM', y='CVX', c=colors, alpha=0.5, ax=ax)
ax.set_xlim(-3, 3)
ax.set_ylim(-3, 3)
```

Hay dos grupos: un grupo en la zona central de los datos y un segundo grupo en el borde exterior. Este resultado es muy diferente del de los grupos obtenidos usando K-means (figura 7.5) y agrupamiento jerárquico (figura 7.9), que encuentran grupos compactos.

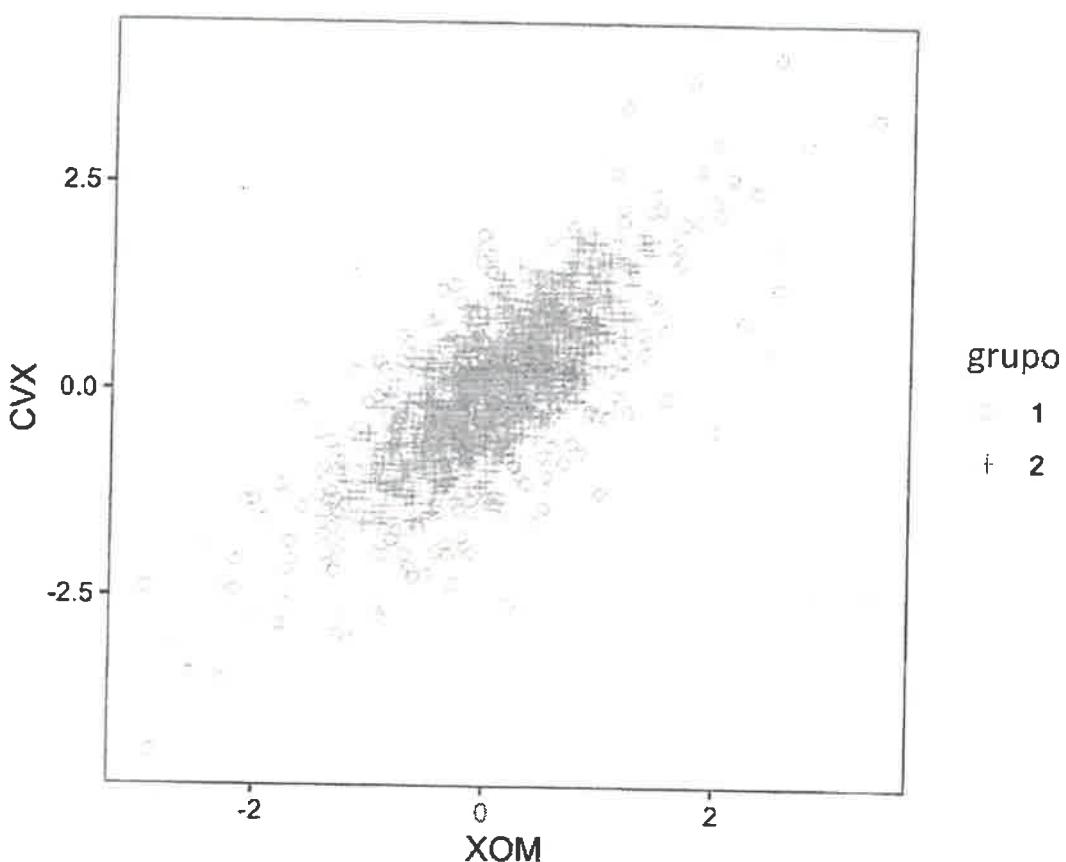


Figura 7.11 Se obtienen dos grupos para los datos de rentabilidades de las acciones utilizando mclust.

Podemos extraer los parámetros de las distribuciones normales utilizando la función `summary`:

```
> summary(mcl, parameters=TRUE)$mean
      [,1]      [,2]
XOM  0.05783847 -0.04374944
CVX  0.07363239 -0.21175715
> summary(mcl, parameters=TRUE)$variance
, , 1
      XOM      CVX
XOM  0.3002049  0.3060989
CVX  0.3060989  0.5496727
, , 2
      XOM      CVX
XOM  1.046318   1.066860
CVX  1.066860   1.915799
```

En `Python` obtenemos esta información de las propiedades del resultado `means_` y `covariances_`:

```
print('Mean')
print(mclust.means_)
```

```
print('Covariances')
print(mclust.covariances_)
```

Las distribuciones tienen medias y correlaciones similares, pero la segunda distribución tiene varianzas y covarianzas mucho mayores. Debido a la aleatoriedad del algoritmo, los resultados pueden variar ligeramente entre diferentes ejecuciones.

Los grupos de `mclust` pueden parecer sorprendentes, pero de hecho ilustran la naturaleza estadística del método. El objetivo de la agrupación basada en el modelo es encontrar el conjunto de distribuciones normales multivariantes que mejor se ajuste. Los datos de las acciones parecen tener una forma normal: ver los contornos de la figura 7.10. En realidad, sin embargo, las rentabilidades de las acciones tienen una distribución de cola más larga que una distribución normal. Para tratar esto, `mclust` ajusta la distribución para la mayor parte de los datos, pero luego ajusta una segunda distribución con una varianza mayor.

Selección del número de grupos

A diferencia de K-means y la agrupación jerárquica, `mclust` selecciona automáticamente el número de grupos en R (en este caso, dos). Para ello, elige el número de grupos para los que el *criterio de información bayesiano* (*Bayesian Information Criteria [BIC]*) tiene el valor más alto (BIC es similar a AIC; consultar “Selección del modelo y regresión escalonada” en la página 156). BIC funciona seleccionando el modelo que mejor se ajusta con una penalización por el número de parámetros del modelo. En el caso de la agrupación en grupos basada en modelos, agregar más grupos siempre mejorará el ajuste a expensas de la introducción de parámetros adicionales en el modelo.



Hay que tener en cuenta que, en la mayoría de los casos, BIC suele minimizarse. Los autores del paquete `mclust` decidieron definir BIC para que tuviera el signo opuesto y facilitar así la interpretación de los gráficos.

`mclust` se adapta a 14 modelos diferentes con un número creciente de componentes y elige un modelo óptimo automáticamente. Podemos trazar los valores BIC de estos modelos usando una función en `mclust`:

```
plot(mcl, what='BIC', ask=FALSE)
```

El número de grupos, o el número de diferentes modelos normales multivariados (componentes), se muestra en el eje x (ver la figura 7.12).

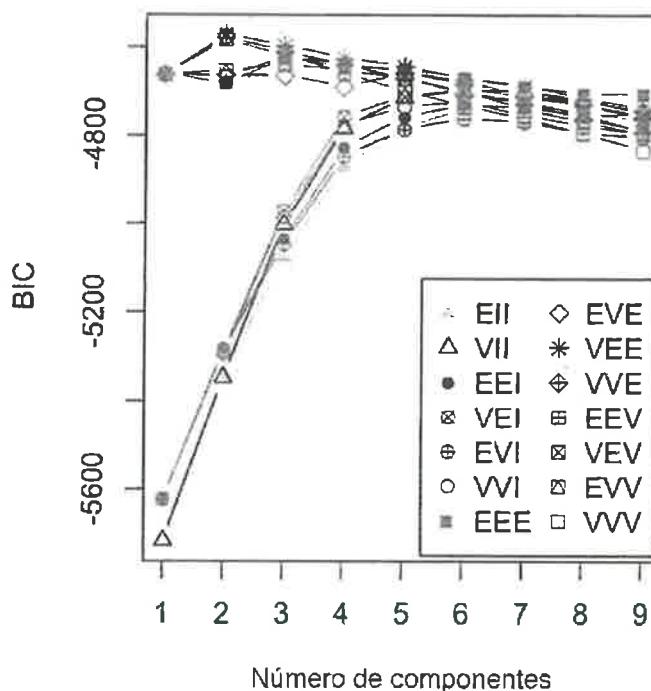


Figura 7.12 Valores BIC para 14 modelos de datos de las rentabilidades de las acciones con un número creciente de componentes.

La implementación de GaussianMixture, por otro lado, no probará varias combinaciones. Como se muestra aquí, es sencillo ejecutar múltiples combinaciones usando *Python*. Esta implementación define BIC como siempre. Por lo tanto, el valor BIC calculado será positivo y debemos minimizarlo.

```

results = []
covariance_types = ['full', 'tied', 'diag', 'spherical']
for n_components in range(1, 9):
    for covariance_type in covariance_types:
        mclust = GaussianMixture(n_components=n_components, warm_start=True,
                                 covariance_type=covariance_type) ①
        mclust.fit(df)
        results.append({
            'bic': mclust.bic(df),
            'n_components': n_components,
            'covariance_type': covariance_type,
        })
results = pd.DataFrame(results)

colors = ['C0', 'C1', 'C2', 'C3']
styles = ['C0-','C1-','C0-','C1-']

fig, ax = plt.subplots(figsize=(4, 4))
for i, covariance_type in enumerate(covariance_types):
    subset = results.loc[results.covariance_type == covariance_type, :]
    subset.plot(x='n_components', y='bic', ax=ax, label=covariance_type,
                kind='line', style=styles[i])

```

- Con el argumento `warm_start`, el cálculo reutilizará la información del ajuste anterior. Esto acelerará la convergencia de los cálculos posteriores.

Este gráfico es similar al gráfico de codo que se usa para identificar el número de grupos a elegir para K-means, excepto que el valor que se representa es BIC en lugar del porcentaje de varianza explicada (ver figura 7.7). Una gran diferencia es que, en lugar de una línea, `mclust` muestra 14 líneas diferentes! Esto se debe a que `mclust` en realidad se ajusta a 14 modelos diferentes para cada tamaño de grupo y, en última instancia, elige el modelo que mejor se ajusta. `GaussianMixture` implementa menos enfoques, por lo que el número de líneas será solo de cuatro.

¿Por qué `mclust` se ajusta a tantos modelos para determinar el mejor conjunto de normales multivariantes? Es porque hay diferentes formas de parametrizar la matriz de covarianza Σ para ajustar un modelo. En su mayor parte, no necesitamos preocuparnos por los detalles de los modelos y simplemente podemos usar el modelo elegido por `mclust`. En este ejemplo, de acuerdo con BIC, tres modelos diferentes (llamados VEE, VEV y VVE) dan el mejor ajuste usando dos componentes.



El agrupamiento basado en el modelo es un área de estudio viva y en rápido desarrollo, y su tratamiento en este texto abarca solo una pequeña parte de su campo de acción. De hecho, el archivo de ayuda de `mclust` tiene actualmente 154 páginas. Navegar por los matices de la agrupación en grupos basada en modelos es probablemente más esfuerzo del que se necesita para la mayoría de los problemas que encuentran los científicos de datos.

Las técnicas de agrupación en grupos basadas en modelos tienen algunas limitaciones. Los métodos requieren una suposición subyacente de un modelo para los datos, y los resultados del grupo dependen en gran medida de esa suposición. Los requisitos de cálculo son más exigentes incluso que la agrupación jerárquica en grupos, lo que dificulta el escalado a cantidades grandes de datos. Finalmente, el algoritmo es más sofisticado y menos accesible que el de otros métodos.

Ideas clave

- Se supone que los grupos derivan de diferentes procesos de generación de datos con diferentes distribuciones de probabilidad.
- Se ajustan diferentes modelos, asumiendo diferentes números de distribuciones (típicamente normales).
- El método elige el modelo (y el número asociado de grupos) que se ajusta bien a los datos sin usar demasiados parámetros (es decir, sobreajuste).

Lecturas complementarias

Para ampliar detalles sobre la agrupación en grupos basada en modelos, consultar la documentación de mclust (<https://stat.uw.edu/sites/default/files/files/reports/2012/tr597.pdf>) y GaussianMixture (<https://scikit-learn.org/stable/modules/mixture.html>).

Variables categóricas y escalado

Las técnicas de aprendizaje no supervisado generalmente requieren que los datos se escalen adecuadamente. Esto es diferente de muchas de las técnicas de regresión y clasificación en las que la escala no es importante (una excepción es K-vecinos más cercanos. Consultar “K-vecinos más cercanos” en la página 238).

Términos clave del escalado de datos

Escalado

Contracción o expansión de datos, generalmente para traer múltiples variables a la misma escala.

Normalización

Método de escalado: restar la media y dividir por la desviación estándar.

Sinónimo

estandarización

Distancia de Gower

Algoritmo de escalado aplicado a datos categóricos y numéricos mixtos para llevar todas las variables al rango de 0 a 1.

Por ejemplo, con los datos de préstamos personales, las variables tienen unidades y magnitudes muy diferentes. Algunas variables tienen valores relativamente pequeños (por ejemplo, el número de años empleados), mientras que otras tienen valores muy grandes (por ejemplo, el monto del préstamo en dólares). Si los datos no se escalan, entonces PCA, K-means y otros métodos de agrupación estarán dominados por las variables con valores grandes e ignorarán las variables con valores pequeños.

Los datos categóricos pueden plantear un problema especial para algunos procedimientos de agrupación. Al igual que con K-vecinos más cercanos, las variables de tipo factor no ordenadas generalmente se convierten en un conjunto de variables binarias (0/1) utilizando una codificación activa (consultar “Codificador One-Hot” en la página 242). No solo es probable que las variables binarias estén en una escala diferente de otros datos, sino que el hecho de que las variables binarias tengan solo dos valores puede resultar problemático con técnicas como PCA y K-means.

Escalado de variables

Las variables con escalas y unidades muy diferentes deben normalizarse adecuadamente antes de aplicar un procedimiento de agrupamiento. Por ejemplo, apliquemos kmeans a un conjunto de datos de impagos de préstamos sin normalizar:

```
defaults <- loan_data[loan_data$outcome=='default',]
df <- defaults[, c('loan_amnt', 'annual_inc', 'revol_bal', 'open_acc',
                  'dti', 'revol_util')]
km <- kmeans(df, centers=4, nstart=10)
centers <- data.frame(size=km$size, km$centers)
round(centers, digits=2)

  size  loan_amnt annual_inc    revol_bal   open_acc    dti revol_util
1   52    22570.19  489783.40    85161.35   13.33    6.91    59.65
2  1192    21856.38  165473.54    38935.88   12.61   13.48    63.67
3 13902    10606.48  42500.30    10280.52    9.59   17.71    58.11
4  7525    18282.25  83458.11    19653.82   11.66   16.77    62.27
```

Aquí está el código *Python* correspondiente:

```
defaults = loan_data.loc[loan_data['outcome'] == 'default']
columns = ['loan_amnt', 'annual_inc', 'revol_bal', 'open_acc', 'dti', 'revol_util']

df = defaults[columns]
kmeans = KMeans(n_clusters=4, random_state=1).fit(df)
counts = Counter(kmeans.labels_)

centers = pd.DataFrame(kmeans.cluster_centers_, columns=columns)
centers['size'] = [counts[i] for i in range(4)]
centers
```

Las variables *annual_inc* y *revol_bal* dominan los grupos, y los grupos tienen tamaños muy diferentes. El grupo 1 tiene solo 52 miembros con ingresos comparativamente altos y saldo de crédito renovable.

Un enfoque muy utilizado para escalar las variables es convertirlas en puntuación *z* restando la media y dividiendo por la desviación estándar. Esto se denomina estandarización (*standardization*) o normalización (*normalization*) (consultar “Estandarización [normalización, puntuación z]” en la página 243 para obtener más información sobre el uso de puntuación *z*):

$$z = \frac{x - \bar{x}}{s}$$

Veamos qué sucede con los grupos cuando se aplica kmeans a los datos normalizados:

```
df0 <- scale(df)
km0 <- kmeans(df0, centers=4, nstart=10)
centers0 <- scale(km0$centers, center=FALSE,
                  scale=1 / attr(df0, 'scaled:center'))
centers0 <- scale(centers0, center=-attr(df0, 'scaled:center'), scale=FALSE)
centers0 <- data.frame(size=km0$size, centers0)
round(centers0, digits=2)
```

	size	loan_amnt	annual_inc	revol_bal	open_acc	dti	revol_util
1	7355	10467.65	51134.87	11523.31	7.48	15.78	77.73
2	5309	10363.43	53523.09	6038.26	8.68	11.32	30.70
3	3713	25894.07	116185.91	32797.67	12.41	16.22	66.14
4	6294	13361.61	55596.65	16375.27	14.25	24.23	59.61

En *Python*, podemos usar `StandardScaler` de `scikit-learn`. El método `inverse_transform` permite convertir los centros del grupo a la escala original:

```
scaler = preprocessing.StandardScaler()
df0 = scaler.fit_transform(df * 1.0)

kmeans = KMeans(n_clusters=4, random_state=1).fit(df0)
counts = Counter(kmeans.labels_)

centers = pd.DataFrame(scaler.inverse_transform(kmeans.cluster_centers_),
                       columns=columns)
centers['size'] = [counts[i] for i in range(4)]
centers
```

Los tamaños de los grupos están más equilibrados y los grupos no están dominados por `annual_inc` y `revol_bal`, lo que revela una estructura más interesante en los datos. Hay que tener en cuenta que los centros se reescalan a las unidades originales en el código anterior. Si los hubiésemos dejado sin escalar, los valores resultantes estarían en términos de puntuación z y, por lo tanto, serían menos interpretables.



El escalado también es importante para PCA. Usar la puntuación z equivale a usar la matriz de correlación (consultar “Correlación” en la página 30) en lugar de la matriz de covarianza al calcular los componentes principales. El software para calcular PCA generalmente tiene una opción para usar la matriz de correlación (en *R*, la función `princomp` tiene el argumento `cor`).

Variables dominantes

Incluso en los casos en que las variables se miden en la misma escala y reflejan con precisión la importancia relativa (por ejemplo, el movimiento de los precios de las acciones), a veces puede ser útil cambiar la escala de las variables.

Supongamos que agregamos Google (GOOGL) y Amazon (AMZN) al análisis en “Interpretación de componentes principales” en la página 289. Vemos cómo se hace esto en *R* a continuación:

```
syms <- c('GOOGL', 'AMZN', 'AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM',
        'SLB', 'COP', 'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST')
top_sp1 <- sp500_px[row.names(sp500_px) >= '2005-01-01', syms]
sp_pca1 <- princomp(top_sp1)
screeplot(sp_pca1)
```

En *Python*, obtenemos el gráfico de sedimentación de la siguiente manera:

```
syms = ['GOOGL', 'AMZN', 'AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM',
        'SLB', 'COP', 'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST']
top_sp1 = sp500_px.loc[sp500_px.index >= '2005-01-01', syms]

sp_pca1 = PCA()
sp_pca1.fit(top_sp1)

explained_variance = pd.DataFrame(sp_pca1.explained_variance_)
ax = explained_variance.head(10).plot.bar(legend=False, figsize=(4, 4))
ax.set_xlabel('Component')
```

El gráfico de sedimentación muestra las variaciones de los componentes principales. En este caso, el gráfico de sedimentación de la figura 7.13 revela que las varianzas del primer y segundo componentes son mucho mayores que las de los demás. Esto a menudo indica que una o dos variables dominan las cargas. Este es, de hecho, el caso en este ejemplo:

```
round(sp_pca1$loadings[,1:2], 3)
      Comp.1 Comp.2
GOOGL  0.781  0.609
AMZN   0.593 -0.792
AAPL    0.078  0.004
MSFT    0.029  0.002
CSCO    0.017 -0.001
INTC    0.020 -0.001
CVX     0.068 -0.021
XOM     0.053 -0.005
...
...
```

En *Python*, usamos lo siguiente:

```
loadings = pd.DataFrame(sp_pca1.components_[0:2, :], columns=top_sp1.columns)
loadings.transpose()
```

Los dos primeros componentes principales están dominados casi por completo por GOOGL y AMZN. Esto se debe a que los movimientos del precio de las acciones de GOOGL y AMZN dominan la variabilidad.

Para manejar esta situación, podemos incluirlos tal cual, cambiar la escala de las variables (consultar “Escalado de variables” en la página 319) o excluir las variables dominantes del análisis y manejarlas por separado. No existe un enfoque “correcto” y el tratamiento depende de la aplicación.

Datos categóricos y distancia de Gower

En el caso de datos categóricos, debemos convertirlos en datos numéricos, ya sea por clasificación (para un factor ordenado) o codificándolos como un conjunto de variables binarias (ficticias). Si los datos constan de una mezcla de variables continuas y binarias, normalmente desearemos escalar las variables para que los rangos sean similares; consultar “Escalado de variables” en la página 319. Un método popular es utilizar la distancia de Gower (*Gower's distance*).

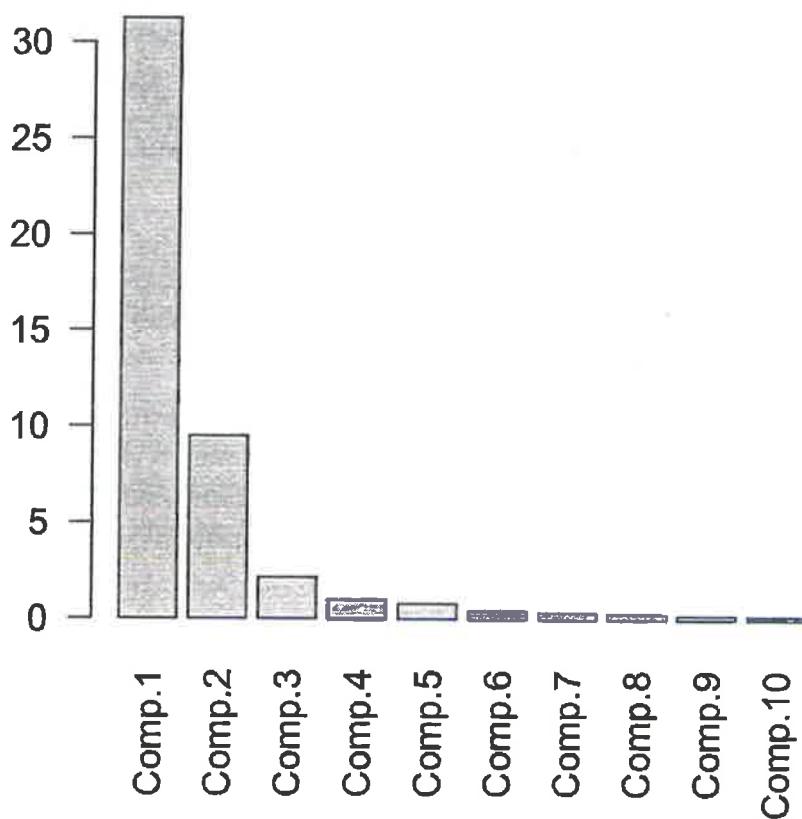


Figura 7.13 Gráfico de sedimentación para la PCA de las principales acciones del S&P 500, incluidos GOOGL y AMZN.

La idea básica detrás de la distancia de Gower es aplicar una métrica de distancia diferente a cada variable según el tipo de datos:

- Para variables numéricas y factores ordenados, la distancia se calcula como el valor absoluto de la diferencia entre dos registros (distancia Manhattan) (*Manhattan distance*).
- Para variables categóricas, la distancia es 1 si las categorías entre dos registros son diferentes y 0 si las categorías son las mismas.

La distancia de Gower se calcula de la siguiente manera:

1. Calculamos la distancia d_{ij} para todos los pares de variables i y j para cada registro.
2. Escalamos cada par d_{ij} de modo que el mínimo sea 0 y el máximo 1.
3. Sumamos las distancias escaladas por pares entre variables juntas, usando una media simple o ponderada, para crear la matriz de distancias.

Para ilustrar la distancia de Gower, utilizamos algunas filas de los datos de préstamos en R:

```
> x <- loan_data[1:5, c('dti', 'payment_inc_ratio', 'home_', 'purpose_')]
> x
# A tibble: 5 × 4
```

	dti	payment_inc_ratio	home	purpose
	<dbl>	<dbl>	<fctr>	<fctr>
1	1.00	2.39320	RENT	major_purchase
2	5.55	4.57170	OWN	small_business
3	18.08	9.71600	RENT	other
4	10.08	12.21520	RENT	debt_consolidation
5	7.06	3.90888	RENT	other

La función `daisy` del paquete `cluster` en R se puede usar para calcular la distancia de Gower:

```
library(cluster)
daisy(x, metric='gower')
Dissimilarities :
      1       2       3       4
2 0.6220479
3 0.6863877 0.8143398
4 0.6329040 0.7608561 0.4307083
5 0.3772789 0.5389727 0.3091088 0.5056250

Metric : mixed ; Types = I, I, N, N
Number of objects : 5
```

En el momento de escribir estas líneas, la distancia de Gower no está disponible en ninguno de los paquetes populares de Python. Sin embargo, se están llevando a cabo actuaciones para incluirlo en `scikit-learn`. Actualizaremos el código fuente adjunto una vez que se publique la implementación.

Las distancias están entre 0 y 1. El par de registros con la mayor distancia es 2 y 3: ninguno tiene los mismos valores para `home` y `purpose`, y tienen niveles muy diferentes de `dti` (debt-to-income) (deuda-ingresos) y `payment_inc_ratio`. Los registros 3 y 5 tienen la distancia más pequeña porque comparten los mismos valores para `home` y `purpose`.

Podemos pasar la matriz de distancias de Gower calculada de `daisy` a `hclust` para la agrupación jerárquica (consultar "Agrupación jerárquica" en la página 304):

```
df <- defaults[sample(nrow(defaults), 250), c('dti', 'payment_inc_ratio', 'home', 'purpose')]
d = daisy(df, metric='gower')
hcl <- hclust(d)
dnd <- as.dendrogram(hcl)
plot(dnd, leaflab='none')
```

El dendrograma resultante se muestra en la figura 7.14. Los registros individuales no se pueden distinguir en el eje x, pero podemos cortar el dendrograma horizontalmente en 0.5 y examinar los registros en uno de los subárboles con este código:

```
dnd_cut <- cut(dnd, h=0.5)
df[labels(dnd_cut$lower[[1]])]
      dti payment_inc_ratio home_ purpose_
44532     21.22        8.37694   OWN debt_consolidation
39826     22.59        6.22827   OWN debt_consolidation
13282     31.00        9.64200   OWN debt_consolidation
```

31510	26.21	11.94380	OWN	debt_consolidation
6693	26.96	9.45600	OWN	debt_consolidation
7356	25.81	9.39257	OWN	debt_consolidation
9278	21.00	14.71850	OWN	debt_consolidation
13520	29.00	18.86670	OWN	debt_consolidation
14668	25.75	17.53440	OWN	debt_consolidation
19975	22.70	17.12170	OWN	debt_consolidation
23492	22.68	18.50250	OWN	debt_consolidation

Este subárbol consta en su totalidad de propietarios con un propósito de préstamo etiquetado como “debt_consolidation”. Si bien la separación estricta no es cierta para todos los subárboles, esto ilustra que las variables categóricas tienden a agruparse en los grupos.

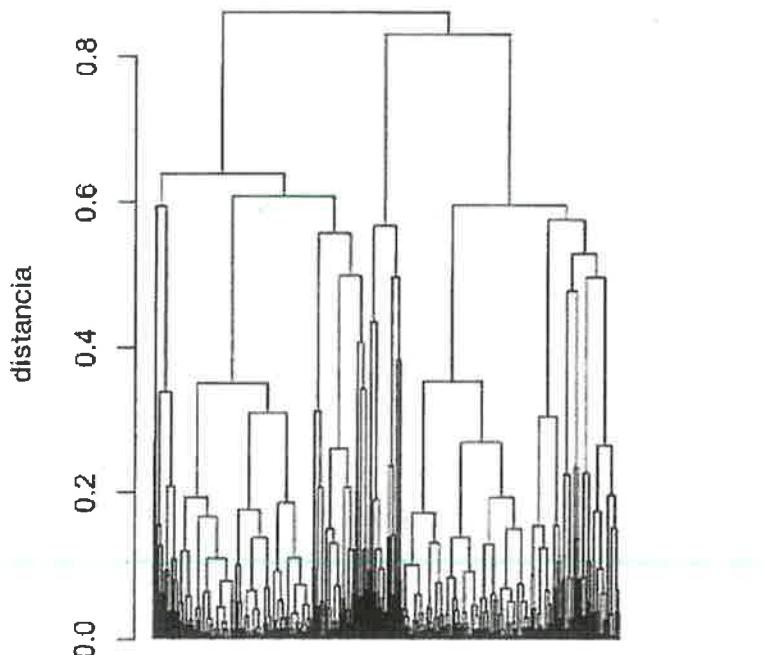


Figura 7.14 Dendrograma de `hclust` aplicado a una muestra de datos de incumplimiento de préstamos con tipos de variables mixtas.

Problemas con la agrupación de datos mixtos

K-means y PCA son los más apropiados para variables continuas. Para conjuntos de datos más pequeños, es mejor utilizar la agrupación jerárquica con la distancia de Gower. En principio, no hay ninguna razón por la que K-means no se pueda aplicar a datos binarios o categóricos. Por lo general, utilizariamos la representación del “codificador One-Hot” (consultar “Codificador One-Hot” en la página 242) para convertir los datos categóricos en valores numéricos. Sin embargo, en la práctica, el uso de K-means y PCA con datos binarios puede resultar difícil.

Si se utilizan la puntuación z estándar, las variables binarias dominarán la definición de los grupos. Esto se debe a que las variables 0/1 toman solo dos valores, y K-means puede obtener una pequeña suma de cuadrados dentro del grupo asignando todos los registros

con un 0 o 1 a un solo grupo. Por ejemplo, aplicamos kmeans a los datos de incumplimiento de préstamos, incluidas las variables de home y pub_rec_zero, en R que se muestran a continuación:

```
df <- model.matrix(~ -1 + dti + payment_inc_ratio + home_ + pub_rec_zero,
                   data=defaults)
df0 <- scale(df)
km0 <- kmeans(df0, centers=4, nstart=10)
centers0 <- scale(km0$centers, center=FALSE,
                   scale=1/attr(df0, 'scaled:center'))
round(scale(centers0, center=-attr(df0, 'scaled:center'), scale=FALSE), 2)
```

	dti	payment_inc_ratio	home_MORTGAGE	home_OWN	home_RENT	pub_rec_zero
1	17.20	9.27	0.00	1	0.00	0.92
2	16.99	9.11	0.00	0	1.00	1.00
3	16.50	8.06	0.52	0	0.48	0.00
4	17.46	8.42	1.00	0	0.00	1.00

En Python:

```
columns = ['dti', 'payment_inc_ratio', 'home_', 'pub_rec_zero']
df = pd.get_dummies(defaults[columns])

scaler = preprocessing.StandardScaler()
df0 = scaler.fit_transform(df * 1.0)
kmeans = KMeans(n_clusters=4, random_state=1).fit(df0)
centers = pd.DataFrame(scaler.inverse_transform(kmeans.cluster_centers_),
                       columns=df.columns)
centers
```

Los cuatro grupos principales son esencialmente los sustitutos de los diferentes niveles de las variables de tipo factor. Para evitar este comportamiento, podemos escalar las variables binarias para que tengan una varianza menor que otras variables. Alternativamente, para conjuntos de datos muy grandes, podemos aplicar la agrupación en grupos a diferentes subconjuntos de datos tomando valores categóricos específicos. Por ejemplo, podríamos aplicar la agrupación por separado a los préstamos otorgados a alguien que tiene una hipoteca, es propietario de una vivienda o alquila.

Ideas clave

- Las variables medidas en diferentes escalas deben transformarse a escalas similares para que su impacto en los algoritmos no esté determinado principalmente por su escala.
- Un método de escala común es la normalización (estandarización): restar la media y dividir por la desviación estándar.
- Otro método es la distancia de Gower, que escala todas las variables al rango 0-1 (a menudo se usa con datos numéricos y categóricos mixtos).

Resumen

Para la reducción de dimensiones de los datos numéricos, las herramientas principales son el análisis de componentes principales o la agrupación K-means. Ambas requieren atención al escalado adecuado de los datos para garantizar una reducción significativa de los datos.

Para la agrupación en grupos con datos muy estructurados en los que los grupos están bien separados, es probable que todos los métodos produzcan un resultado similar. Cada método ofrece su propia ventaja. K-means escala a datos muy grandes y se entiende fácilmente. La agrupación jerárquica se puede aplicar a tipos de datos mixtos (numéricos y categóricos) y se presta a una visualización intuitiva (el dendrograma). La agrupación basada en el modelo parte de la teoría estadística y proporciona un enfoque más riguroso, en contraposición a los métodos heurísticos. Sin embargo, para datos muy grandes, K-means es el método principal utilizado.

Con datos ruidosos, como los datos de préstamos y las acciones (y gran parte de los datos a los que se enfrentará un científico de datos), la elección es más clara. K-means, la agrupación jerárquica y especialmente la agrupación basada en el modelo producen soluciones muy diferentes. ¿Cómo debe proceder un científico de datos? Desafortunadamente, no existe una regla general que oriente la elección. En última instancia, el método utilizado dependerá del tamaño de los datos y el objetivo de la aplicación.

Bibliografía

- [Baumer, 2017] Baumer, Benjamin, Daniel Kaplan y Nicholas Horton. *Modern Data Science with R*. Boca Ratón, Fla.: Chapman & Hall/CRC Press, 2017.
- [bokeh] Bokeh Development Team. “Bokeh: Python library for interactive visualization” (2014). <https://bokeh.pydata.org>.
- [Deng-Wickham, 2011] Deng, Henry, y Hadley Wickham. “Density Estimation in R.” Septiembre de 2011. https://oreil.ly/-Ny_6.
- [Donoho, 2015] Donoho, David. “50 Years of Data Science.” 18 de septiembre, 2015. <https://oreil.ly/kqFb0>.
- [Duong, 2001] Duong, Tarn. “An Introduction to Kernel Density Estimation.” 2001. <https://oreil.ly/Z5A7W>.
- [Few, 2007] Few, Stephen. “Save the Pies for Dessert.” *Visual Business Intelligence Newsletter*. Perceptual Edge. Agosto, 2007. https://oreil.ly/_iGAL.
- [Freedman, 2007] Freedman, David, Robert Pisani y Roger Purves. *Statistics*. 4.^a ed. Nueva York: W. W. Norton, 2007.
- [Hintze-Nelson, 1998] Hintze, Jerry L., y Ray D. Nelson. “Violin Plots: A Box Plot-Density Trace Synergism.” *The American Statistician* 52, n.^o 2 (mayo, 1998): 181-184.
- [Galton, 1886] Galton, Francis. “Regression Towards Mediocrity in Hereditary Stature.” *The Journal of the Anthropological Institute of Great Britain and Ireland* 15 (1886): 246-263. <https://oreil.ly/DqoAk>.
- [ggplot2] Wickham, Hadley. *ggplot2: Elegant Graphics for Data Analysis*. Nueva York: Springer-Verlag New York, 2009. <https://oreil.ly/O92vC>.
- [Hyndman-Fan, 1996] Hyndman, Rob J. y Yanan Fan. “Sample Quantiles in Statistical Packages.” *American Statistician* 50, n.^o 4 (1996): 361-365.
- [lattice] Sarkar, Deepayan. *Lattice: Multivariate Data Visualization with R*. Nueva York: Springer, 2008. <http://lmdvr.r-forge.r-project.org>.

Estadística práctica para ciencia de datos con R y Python

- [Legendre] Legendre, Adrien-Marie. *Nouvelle méthodes pour la détermination des orbites des comètes*. París: F. Didot, 1805. <https://oreil.ly/8FITJ>.
- [NIST-Handbook, 2012] “Measures of Skewness and Kurtosis.” En *NIST/SEMATECH e-Handbook of Statistical Methods*. 2012. <https://oreil.ly/IAdHA>.
- [R-base, 2015] R Core Team. “R: A Language and Environment for Statistical Computing.” R Foundation for Statistical Computing. 2015. <https://www.r-project.org>.
- [Salsburg, 2001] Salsburg, David. *The Lady Tasting Tea: How Statistics Revolutionized Science in the Twentieth Century*. Nueva York: W. H. Freeman, 2001.
- [seaborn] Waskom, Michael. “Seaborn: Statistical Data Visualization.” 2015. <https://seaborn.pydata.org>.
- [Trellis-Graphics] Becker, Richard A., William S. Cleveland, Ming-Jen Shyu, y Stephen P. Kaluzny. “A Tour of Trellis Graphics.” 15 de abril, 1996. <https://oreil.ly/LVnOV>.
- [Tukey, 1962] Tukey, John W. “The Future of Data Analysis.” *The Annals of Mathematical Statistics* 33, n.º 1 (1962): 1-67. <https://oreil.ly/qrYNW>.
- [Tukey, 1977] Tukey, John W. *Exploratory Data Analysis*. Reading, Mass.: Addison-Wesley, 1977.
- [Tukey, 1987] Tukey, John W. *The Collected Works of John W. Tukey*. Vol. 4, *Philosophy and Principles of Data Analysis: 1965-1986*, edición de Lyle V. Jones. Boca Ratón, Fla.: Chapman & Hall/CRC Press, 1987.
- [Zhang-Wang, 2007] Zhang, Qi y Wei Wang. “A Fast Algorithm for Approximate Quantiles in High Speed Data Streams.” *19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)*. Piscataway, NJ: IEEE, 2007. También disponible en <https://oreil.ly/qShjk>.

Índice onomástico

A

- aciertos, 132
- acondicionamiento, 43
- Adaboost, 264
- algoritmo boosting, 264
- agregación de bootstrap (ver bagging)
- agrupación, 278
 - basada en el modelo, 301-316
 - distribución normal multivariante, 301
 - selección del número de grupos, 297-295
 - jerárquica, 296-301
 - algoritmo de aglomeración, 299
 - métricas o medidas de disimilitud, 299-301
 - representación en un dendrograma, 297-298
 - un ejemplo sencillo, 296
 - K-means, 287-295
 - interpretación de los grupos, 291
 - algoritmo K-means, 290
 - selección del número de grupos, 293
 - un ejemplo sencillo, 296
 - problemas con la agrupación de datos mixtos, 314
 - variables categóricas plantean problemas a la, 308
- agrupación basada en el modelo, 301-316
 - limitaciones de la, 307
 - mezclas de distribuciones normales, 303-305
 - multivariate normal distribution, 301
- agrupación hexagonal, 36
 - extensión con acondicionamiento de variables, 43
 - y contornos, gráfico de la representación entre dos valores numéricos, 36-39
- agrupación jerárquica, 296-301
 - algoritmo aglomerativo, 299
 - métricas o medidas de disimilitud, 299-301
 - representación en dendrograma, 297-298

Estadística práctica para ciencia de datos con R y Python

- un ejemplo sencillo, 296
- uso con la distancia de Gower, 308
- agrupación K-means, 287-295
 - algoritmo K-means, 290
 - aplicación a datos normalizados, 309
 - aplicación a datos sin normalizar, 309
 - interpretación de los grupos, 291
 - selección del número de grupos, 293
 - un ejemplo sencillo, 296
 - utilizada con datos binarios, 314
- AIC (Akaike's Information Criteria), 153-155
- AICc, 153
- ajuste de valores p, 112
- ajuste del modelo
 - bosque aleatorio aplicado a los datos de préstamos no reembolsados, 255
 - equilibrio entre sesgo y varianza, 247
 - K-vecinos más cercanos, ventajas del, 232
 - reglas para un modelo sencillo que se ajustan a los datos de préstamos, 245
 - regresión logística frente a la lineal, 203
- aleatoriedad, infravaloración e interpretación errónea de la, 94
- aleatorización, 88
- alejamiento de las expectativas, 79
- alfa, 103, 107
 - inflación alfa, 112
- algoritmo de aglomeración, 299
- algoritmo de Bayes ingenuo, 192-197
 - por qué la clasificación bayesiana exacta no es práctica, 193
 - solución del, 194-197
 - variables predictoras numéricas con, 196
- algoritmo epsilon-greedy para la prueba A/B, 132
- algoritmo SMOTE, 228
- algoritmos Bandit, 91
 - (ver también Multi-Arm Bandit)
- algoritmos greedy, 133
- American Statistical Association (ASA), declaración sobre el uso del valor p, 107
- análisis bivariante, 36
- análisis de componentes principales, 278-279, 282
 - análisis de correspondencias, 285-286
 - análisis de grupos frente a, 292
 - cálculo de los componentes principales, 282
 - decisión de cuántos componentes elegir, 285
 - escalado de variables, 309
 - interpretación de los componentes principales, 282

- un ejemplo sencillo, 288
- análisis de correspondencias, 285-286
- análisis de datos, 1
 - (ver también análisis exploratorio de datos)
- análisis de la varianza (ANOVA), 81, 117-123
 - ANOVA bidireccional, 122
 - descomposición de la varianza, 122
 - estadístico F, 121
- análisis discriminante, 197-203
 - análisis discriminante lineal (LDA), 198
 - discriminante lineal de Fisher, 199
 - matriz de covarianza, 198
- análisis discriminante cuadrático (QDA), 202
- análisis discriminante lineal (LDA), 198
 - (ver también análisis discriminante)
- análisis de componentes principales como versión no supervisada, 279
- análisis exploratorio de datos, 1-46
 - aprendizaje no supervisado como extensión de, 277
 - correlación, 30-36
 - datos binarios y categóricos, 27-30
 - distribución de datos, 19-27
 - estimación de la localización, 7-13
 - estimación de la variabilidad, 13-19
 - exploración de dos o más variables, 36-46
- análisis exploratorio de datos (Tukey), 1
- análisis multivariante, 36
- análisis univariable, 36
- ANOVA bidireccional, 122
- ANOVA unidireccional, 122
- apalancamiento
 - definición, 175
 - valores influyentes en regresión, 175
- aprendizaje automático, 231
 - (ver también aprendizaje automático estadístico; aprendizaje supervisado; aprendizaje no supervisado)
 - estadística frente a, 232
 - riesgo de sobreajuste, mitigación, 113
 - uso del remuestreo para mejorar los modelos, 96
- aprendizaje automático estadístico, 231
 - aprendizaje no supervisado, 277
 - (ver también aprendizaje no supervisado)
 - métodos de bagging y bosque aleatorio, 253
 - bagging, 254

- bosques aleatorios, 255
- hiperparámetros, 262
- importancia de la variable, 259
- boosting, 263
 - algoritmo boosting, 264
 - hiperparámetros y validación cruzada, 272
 - sobreajuste, evitación con la regularización, 268
 - software XGBoost, 265
- K-vecinos más cercanos, 232
 - codificador One-Hot, 236
 - ejemplo, pronóstico de incumplimiento de un préstamo, 228
 - estandarización, 237
 - KNN como motor de características, 241
 - métricas de distancia, 235
 - modelos de árbol, 243
 - cómo se utilizan los árboles, 252
 - detención del crecimiento del árbol, 249
 - medición de la homogeneidad o de la impureza, 248
 - partición recursiva, 246
 - pronóstico de un valor continuo, 251
 - un ejemplo sencillo, 244
- aprendizaje no supervisado, 277
- agrupación jerárquica, 296
 - algoritmo de aglomeración, 299
- agrupación K-means, 287
 - algoritmo K-means, 290
 - interpretación de los grupos, 291
 - selección del número de grupos, 293
 - un ejemplo sencillo, 288
- agrupación basada en el modelo, 315
 - distribución normal multivariante, 301
 - mezclas de distribuciones normales, 303-305
 - selección del número de grupos, 305-307
- análisis de componentes principales, 278
 - análisis de correspondencias, 285
 - cálculo de componentes principales, 282
 - interpretación de componentes principales, 282
 - un ejemplo sencillo, 279
- aprendizaje por conjuntos, 231
- aprendizaje supervisado, 139, 191, 277
- árboles de clasificación y regresión (CART), 243
 - (ver también modelos de árbol)
- árboles de decisión, 243

aprendizaje por conjuntos aplicado a, 231
antiguo significado en el análisis de toma de decisiones de los individuos, 244
ejecución múltiple en muestras bootstrap, 65
árboles potenciados, 243, 252
arms (Multi-Arm Bandit), 130
asimétrico, 72
asimetría, 24
AUC (área bajo la curva ROC), 221
aumento/disminución de la ponderación, 226

B

b-spline (basis spline), 186
bagging, 64, 96, 253
 boosting frente a, 263
bases de datos, tipos de datos en, 4
big data
 modelos predictivos en, 47
 valor de, 52
bigotes (en diagramas de caja), 21
biplot, 285
boosting, 263
 algoritmo boosting, 264
 frente a bagging, 263
 hiperparámetros y validación cruzada, 272-274
 regularización, evitación del sobreajuste con, 268-271
 software XGBoost, 265-267
bootstrap, 65
 algoritmo para el remuestreo bootstrap de la media, 62
 bootstrap y pruebas de permutación, 102
 en el remuestreo, 96
 error estándar y, 60
 generación de intervalos de confianza, 65, 158
 remuestreo frente a bootstrapping, 65
bosques aleatorios, 243, 255
 hiperparámetros, 262
 importancia de la variable en, 259
Breiman, Leo, 231, 232, 243, 254, 255
búsqueda
 necesidad de una gran cantidad de datos, 54
 efecto de búsqueda masiva, 55

C

- calidad de los datos, 48
- causalidad, regresión y, 146
- ciencia de datos
- estadístico t y, 151
 - importancia de las pruebas de chi cuadrado, 123
 - multiplicidad y, 113-115
 - prueba A/B en, 88
 - pruebas de permutación, valor de, 97
 - valor de heterocedasticidad para, 177
 - valores p y, 109
- clasificación,
 - algoritmo de Bayes ingenuo, 192
 - aplicación a variables predictoras numéricas, 196
 - análisis discriminante, 197
 - discriminante lineal de Fisher, 199
 - matriz de covarianza, 198
 - un ejemplo sencillo, 200
 - estrategias para datos que no están equilibrados, 230
 - clasificación basada en los costes, 228
 - generación de datos, 228
 - exploración de pronósticos, 229
 - sobremuestreo y aumento/disminución de la ponderación, 226
 - submuestreo, 225
 - evaluación de modelos, 215
 - curva ROC, 219
 - matriz de confusión, 216
 - métrica AUC, 221
 - precisión, exhaustividad y especificidad, 218
 - problema de las clases raras, 218
 - sustentación, 222
 - regresión logística, 203
 - comparación con la regresión lineal, 210
 - función de respuesta logística y logit, 204
 - interpretación de los coeficientes y de la razón de oportunidades, 208
 - modelos lineales generalizados, 207
 - valores pronosticados de, 208
 - codificación de contraste, 163
 - codificación de desviación, 160, 163, 167
 - codificación de referencia, 160, 171, 206, 211
 - codificación One-Hot, 160, 162-163, 211, 237
 - codificación polinomial, 163

- coeficiente de correlación, 31
 - cálculo del coeficiente de correlación de Pearson, 31
 - otros tipos de, 34
- coeficiente de correlación de Pearson, 31
- coeficiente de determinación, 150
- coeficiente de Gini, 249
- coeficientes
 - en regresión lineal múltiple, 147
 - en regresión lineal simple, 139
 - en regresión logística, 209
 - intervalos de confianza y, 158
- coeficientes de regresión, 177
 - comparación con todos los datos y con los datos influyentes eliminados, 177
 - intervalos de confianza y, 158
 - predictoras correlacionadas y, 166
- colas, 72
- comparaciones por pares, 118-119
- componentes principales, 278
- conjunto de modelos, 253
 - bagging y boosting, 254
- contenedores, en tablas de frecuencias e histogramas, 22
- contrastos de suma, 163
- correlación, 30-36
 - conceptos clave, 35
 - diagramas de dispersión, 34
 - ejemplo, correlación entre retornos ETF, 32
 - términos clave para, 31
- covarianza, 198, 282
- criterio de información bayesiano (BIC), 153, 305-307
- cuantiles, 17 (ver también percentiles)
 - funciones para, 17
- curtosis, 24
- curva de precisión-exhaustividad (PR), 220
- curva Receiver Operating Characteristics
 - (ver curva ROC)
- curva ROC, 219
 - métrica AUC, 249

D

- d.f. (grados de libertad), 115
 - (ver también grados de libertad)

Estadística práctica para ciencia de datos con R y Python

- datos binarios, 2
 - exploración de, 27-30
- datos categóricos
 - exploración, 27-30
 - moda, 29
 - probabilidad, 30
 - valor esperado, 29
 - exploración de dos variables categóricas, 39
 - exploración de variables numéricas agrupadas por variables categóricas, 41
 - importancia del concepto, 3
- datos continuos, 3
- datos discretos, 3
- datos estructurados, 2-4
- datos numéricos, 3
 - agrupados de acuerdo con una variable categórica, exploración, 41
 - exploración de la relación entre dos variables numéricas, 36-39
 - reducción de la dimensión de, 316
 - importancia del concepto, 3
- datos ordinales, 3
- datos rectangulares, 4
 - diferencias terminológicas, 6
 - términos clave para, 5
- dendrogramas, 297
- descomposición de la varianza, 117, 122
- desviación, 213
 - intento de minimizarla en la regresión logística, 227
 - desviación absoluta mediana de la mediana (MAD), 14, 16, 18-19
- desviación media absoluta, 14
 - fórmula para el cálculo, 15
- desviaciones
 - definición de desviaciones, 14
 - desviación estándar y estimaciones relacionadas, 14
- detección de anomalías, 139
 - valores atípicos y, 11
- diagrama de caja, 19
 - comparación de datos numéricos y categóricos, 41
 - diagramas de violín versus, 42
 - extensión a variables condicionantes, 44
 - percentiles y, 20-21
- diagramas de barras, 282-283, 285
 - para PCA de las principales acciones, 285
- diagramas de burbujas, 176
- diagramas de contornos, 36

- usados con agrupaciones hexagonales, 36-39
- diagramas de densidad, 20
 - y estimación, 24
- diagramas de influencia, 176
- diagramas de residuos parciales, 180, 184
 - falta de linealidad y, 180
 - para la regresión spline, 183
- diagramas de violín, 36
 - diagramas de caja frente a, 42
- diagramas QQ, 69
 - de las rentabilidades de las acciones de NFLX, 73
 - distribución normal estándar y, 70
- discriminante lineal de Fisher, 199
- dispersión, 13
 - (ver también variabilidad)
- distancia de Cook, 176
- distancia de Gower, 311
 - uso para escalar variables categóricas, 311
- distancia de Mahalanobis, 199, 236
- distancia euclídea, 235-236
- distancia Manhattan, 236, 269, 272, 312
- distribución beta, 133
- distribución binomial, 77-79
- distribución chi cuadrado, 80, 123
- distribución de cola larga, 72
- distribución de datos, 19-27
 - diagramas y estimación de la curva de densidad, 24-27
 - distribución del muestreo versus, 58
 - percentiles y diagramas de caja, 20-21
 - tabla de frecuencias e histograma, 22-24
- distribución exponencial, 83
- distribución F, 81
- distribución gaussiana, 69
 - (ver también distribución normal)
- distribución muestral, 57-61
 - distribución de datos frente a la, 58
 - error estándar, 60
 - teorema del límite central, 60
- distribución normal, 69-72
 - conceptos clave, 72
 - error acerca de la, 69
 - multivariante, 301
 - normal estándar y diagramas QQ, 70

distribución normal estándar, 69
y diagramas QQ, 70
distribución normal multivariante, 301
mezclas de distribuciones normales, 303-305
distribución de Poisson, 81
distribución t de Student, 74
distribución aleatoria uniforme, 128
distribución de Weibull, 84
Donoho, David, 1

E

efecto de búsqueda masiva, 54-55
efecto del tamaño, 134
efectos principales, 171
interacciones y, 174-176
Elder, John, 55
elevación, 223
(ver también sustentación)
eliminación hacia atrás, 154-155
encuesta de Literary Digest de 1936, 49
enfoque bayesiano en el muestreo de Thompson, 133
enfoque centrado en los datos, 203
ensayos, 77
ensayos binomiales, 78-79
ensayos en distribuciones binomiales, 78-79
entropía de la información, 248
equilibrio entre sesgo y varianza, 250
error cuadrático medio (RMSE), 147, 149-150, 153, 157, 210, 212, 251
error estándar, 57, 60-61
error estándar residual (RSE), 147, 149
errores, 69
errores de pronóstico, 143
(ver también residuos)
errores de heterocedasticidad, 177
errores de multicolinealidad, 116, 168
errores de tipo 1, 103, 108, 112
errores de tipo 2, 103, 108
escalado, 318
escalado y variables categóricas, 308-316
escalado de variables, 309-310
problemas con la agrupación de datos mixtos, 314

- variables categóricas y distancia de Gower, 311
- variables dominantes, 310-311
- especificidad, 218-222
 - equilibrio con la exhaustividad, 218
- espionaje de datos, 56
- estadístico chi cuadrado, 123
- estadístico de prueba, 89, 91, 110-111
- estadístico Durbin-Watson, 179
- estadístico F, 120
- estadístico muestral, 57
- estadístico t-s, 151
- estadísticos de orden, 17
- estandarización, 70, 237
 - de variables continuas, 311
 - en K-vecinos más cercanos, 233
- estimación no sesgada, 16
- estimación
 - notación \hat{y} , 143
 - métricas y , 9
- estimación de error fuera de bolsa, 256
- estimación de la densidad del núcleo, 20, 24
 - (ver también diagramas de densidad)
- estimación de la localización, 7, 173
- estimación puntual, 66
- estimación robusta de la correlación, 33
- estimación robusta de la localización, 11
 - ejemplo, estimación de la localización de la población y tasas de homicidios, 12
 - mediana, 10
 - mediana ponderada, 11
 - otras métricas robustas para, 11
- estimación robusta de la variabilidad, desviación absoluta mediana de la mediana, 14
- estimación robusta de la varianza,
 - cálculo de la MAD robusta, 18
- estimaciones sesgadas, 15
- estructuras de datos en forma de redes, 7
- estructuras de datos no rectangulares, 6
- estudios ciegos, 91
- estudios doble ciego, 91
- estrategias para datos que no están equilibrados en
 - modelos de clasificación, 224
 - clasificación basada en costes, 228
 - exploración de pronósticos, 229
 - generación de datos, 228

Estadística práctica para ciencia de datos con R y Python

- sobremuestreo y aumento/disminución de la ponderación, 226
- estructuras de datos espaciales, 6
- exhaustividad, 219
 - (ver también sensibilidad)
 - compensación con la especificidad, 219
- éxito, 78
- expectativas o resultados esperados, 123
 - a partir de, 123
- experimentos estadísticos y pruebas significativas, 87-139
 - análisis de la varianza (ANOVA), 117-123
 - grados de libertad, 115-117
 - Multi-Arm Bandit, 130-133
 - potencia y tamaño de la muestra, 135-139
 - prueba A/B, 88
 - prueba de chi cuadrado, 123-130
 - pruebas de hipótesis, 93
 - pruebas múltiples, 111
 - pruebas t, 109
 - remuestreo, 96-103
 - significación estadística y valores p, 103
 - alfa, 107
 - ciencia de datos y valores p, 109
 - controversia sobre valores p, 107
 - errores tipo 1 y tipo 2, 109
 - valores p, 106
- explicación (elaboración de perfiles),
 - pronóstico frente a, 146
- extrapolación
 - peligro de, 158
 - definida, 157

F

- facetas, 43
- Fisher, R. A., 199
- fraude científico, detección del, 128
- Friedman, Jerome H. (Jerry), 231
- función de oportunidades inversa, 205
- función de pérdida, 227
- función de respuesta logística, 204
- función discriminatoria, 198
- función discriminatoria lineal, 198, 202

función logaritmo de la razón de oportunidades

(ver función logit)

función logit, 204-205

función logit inversa, 204

G

Gallup Poll, 49

Galton, Francis, 56, 253

ganancias, 223

(ver también sustentación)

Gauss, Carl Friedrich, 70, 145

generación de datos, 228

Google Analytics, 92, 98

Gosset, W. S., 75

grados de libertad, 15, 115, 121

para la distribución chi cuadrado, 126

distribución t y, 75

gráfico de ganancias acumuladas, 223

gráficos, 7

en informática frente a estadística, 7

gráficos de barras, 27

histogramas y, 28

gráficos de ganancias en deciles, 223

gráficos en forma de tarta, 27, 28

gráficos Trellis, 44

grupo de control, 88

ventajas de su uso, 90

grupo de tratamiento, 88

grupos, 293

guía de W3Schools para SQL, 4

H

heterocedasticidad, 177

importancia para la ciencia de datos, 177

hiperparámetros, 262

hipótesis alternativa, 95

hipótesis alternativa bidireccional, 95

hipótesis nula, 94

en la prueba de tasa de clics para titulares web, 124

uso de la hipótesis alternativa con la, 95

Estadística práctica para ciencia de datos con R y Python

- mezcla de objetivos, 55
- moda, 29
 - ejemplos en datos categóricos, 30
- modelado predictivo
 - KNN como primera fase del, 241
 - modelo lineal (lm), 142
 - modelos aditivos generalizados (GAM), 183, 187, 189
 - ajuste de regresión logística con, 229
 - modelos de árbol, 243
 - algoritmo de partición recursiva, 246
 - cómo se utilizan los árboles, 252
 - conjuntos de modelos, bosque aleatorio y árboles potenciados, 253
 - detención del crecimiento del árbol, 249
 - medición de la homogeneidad y de la impureza, 248
 - pronóstico de un valor continuo, 251
 - un ejemplo sencillo, 244
 - ventajas de los, 252
 - modelos lineales generalizados (GLM), 207
 - momentos de una distribución, 24
 - muestra aleatoria simple, 48
 - muestra de bootstrap, 62
 - muestras
 - tamaño de la muestra, potencia y, 135-139
 - diferencias terminológicas, 6
 - muestras no aleatorias, 55
 - muestreo, 48
 - bootstrap, 61
 - con y sin sustitución, 97
 - distribución binomial, 77-79
 - distribución chi cuadrado, 79-80
 - distribución de Poisson y distribuciones relacionadas, 81
 - distribución de Poisson, 82
 - distribución exponencial, 83
 - distribución de Weibull, 84
 - distribución F, 81
 - distribución normal, 69-72
 - distribución t de Student, 74-77
 - distribuciones de cola larga, 72-74
 - intervalos de confianza, 65
 - muestreo aleatorio y sesgo de la muestra, 48-54
 - sesgo de selección, 54-57
 - muestreo aleatorio, 48-54
 - media muestral frente a media poblacional, 53

selección aleatoria, 51
sesgo, 50
tamaño frente a calidad, 52
muestreo bootstrap multivariante, 64
muestreo de Thompson, 133
muestreo estratificado, 52
Multi-Arm Bandit, 91, 130-133
multicolinealidad, 116, 168
problemas causados por la codificación One-Hot, 161, 211, 237
y predictoras utilizadas dos veces en KNN, 241

N

N (o n) se refiere al número total de registros, 9
 $n \text{ o } n - 1$, divisor en la fórmula de la varianza, 15
 $n \text{ o } n - 1$, divisor en la fórmula de la varianza o de la desviación estándar, 16
n o tamaño de la muestra, 75
navaja de Occam, 153
nivel de confianza, 68
nivel de significación, 134, 137
nodos, 187, 260
normalización, 70, 237
(ver también estandarización)
en estadística, vs. normalización de bases de datos, 237
escalado de variables, 309
normalización de una base de datos versus
normalización en estadística, 237
notación hat, valores estimados frente a valores conocidos, 143
nudos, 183, 186, 189
números aleatorios, generación a partir de la distribución de Poisson, 81

O

objetivo, 6
oportunidades, 204
obtención de la probabilidad a partir de las, 204

P

parámetro de suavizado, uso con el algoritmo de Bayes ingenuo, 197
partición recursiva, 246
particiones en árboles, 246

Estadística práctica para ciencia de datos con R y Python

- algoritmo de partición recursiva, 246
- bosque aleatorio, 255
- PCA
 - (ver análisis de componentes principales)
 - (ver también coeficientes de regresión)
- Pearson, Karl, 279
- penalización de la complejidad del modelo, 271
- pendiente, 141
- percentiles, 8, 14
 - definición precisa de, 17
 - estimación basada en, 17
 - y diagramas de caja, 20-21
- pérdida, 227, 244, 246
 - en un ejemplo de árbol sencillo, 245
- permisos para pruebas científicas y médicas, 92
- pliegues, 272
- población, 48
 - media de la población vs. media de la muestra, 53
- poda, 244, 250
- ponderación
 - aumento y disminución de la ponderación, 226
 - utilización para cambiar la función de pérdida en la clasificación, 277
- ponderaciones de los componentes principales
 - (ver cargas)
- ponderaciones discriminatorias, 198
- potencia y tamaño de la muestra, 135-139
 - cálculo, componentes en, 137
- potencia, 135
 - tamaño de la muestra, 135
- potenciación del gradiente, 264
- precisión, 218
- probabilidad, 30, 195
 - asociada con los intervalos de confianza, 67
 - producida por los modelos de árbol, 248
 - salida de K-vecinos más cercanos, 232
- probabilidad *a posteriori*, 193
- probabilidad condicional, 193
- problema de arranque en frío, uso de la agrupación para, 278
- problema de las clases raras, 218
- pronóstico, 139
 - (ver también clasificación)
 - aprendizaje no supervisado y, 278
 - aprovechamiento de los resultados de varios árboles, 252

- de bosque aleatorio, 231
- de XGBoost aplicado a los datos de incumplimiento de préstamos, 267
- exploración de pronósticos de los modelos de clasificación, 229
- pronóstico de incumplimiento de préstamos con K-vecinos más cercanos, 233
- pronóstico de un valor continuo con un modelo de árbol, 251
- pronóstico frente a explicación en regresión lineal simple, 146
- valores ajustados y residuos en la regresión lineal simple, 143
- valores pronosticados de regresión logística, 208
- utilización de la regresión, 158
- intervalos de confianza y de pronóstico, 158-160
- peligros de la extrapolación, 158
- propensión, 195 (ver también probabilidad)
- prueba A/B, 88-93
 - algoritmo epsilon-greedy para la, 133
 - beneficios de utilizar un grupo de control, 90
 - ejemplos de, 88
 - hipótesis en la, 93
 - importancia de la obtención de permisos, 92
 - Multi-Arm Bandit frente a, 133
 - tradicional, defecto de, 91
- prueba de chi cuadrado, 123
 - enfoque de remuestreo, 124
- prueba exacta de Fisher, 127
 - relevancia para la ciencia de datos, 129
 - teoría estadística, 126
- prueba de chi cuadrado de Pearson, 125
- prueba exacta de Fisher, 127
 - relevancia para la ciencia de datos, 129
- pruebas bidireccionales, 95
- pruebas de detección médica, falsos positivos y, 217
- pruebas de hipótesis, 93-96
 - estructuradas para minimizar errores de tipo I, 108
 - hipótesis alternativa, 95
 - hipótesis nula, 94
- pruebas en uno y dos sentidos, 95
 - tasa de falsos descubrimientos, 112
- pruebas de permutación exhaustiva, 102
- pruebas exactas, 102
 - prueba exacta de Fisher, 127
- pruebas de permutación, 102
 - ejemplo de adherencia de la web, 104
 - estimación de valores p de, 106
 - exhaustiva y bootstrap, 102

Estadística práctica para ciencia de datos con R y Python

- para ANOVA, 120
- para la prueba de chi cuadrado, 123
- valor para la ciencia de datos, 102
- pruebas múltiples, 111
- pruebas ómnibus, 117
- pruebas significativas, 93
 - (ver también pruebas de hipótesis)
 - minusvaloración y malinterpretación de los eventos aleatorios en las, 93
- pruebas unidireccionales, 93, 95
- pruebas web
 - adherencia de la web,
 - ejemplo, 98-102
 - popularidad del algoritmo Bandit en, 132
 - prueba A/B en ciencia de datos, 102
 - tasa de clics para tres titulares diferentes, 124
- pseudorresiduos, 265
- puntos finales del intervalo, 66
- puntuación de Fisher, 210, 212
- puntuación z, 69-72, 237
 - conversión de datos a, distribución normal y, 69
 - en estandarización de datos para KNN, 237
 - utilización para el escalado de variables, 309
- pureza de clase, 248

R

- R cuadrado, 147, 150, 157, 210
- R cuadrado ajustado, 151
- rango, 17
- rango intercuartílico, 17
 - cálculo, 17
- razón de oportunidades, 208
 - relación con la razón del logaritmo de oportunidades, 208
- razón del logaritmo de oportunidades y razón de oportunidades, 208
- reducción de la dimensión de los datos, 285, 316
- registros pertinentes (en búsquedas), 53
- regla de la raíz cuadrada de n, 60
- reglas si-entonces-sino (modelos de árbol), 244
- regresión, 139
 - ANOVA como primer paso hacia un modelo estadístico, 122
 - aprendizaje no supervisado como componente básico, 277

- diagnósticos, 172
 - diagramas de residuos parciales y falta de linealidad, 180
 - heterocedasticidad, anormalidad, y errores correlacionados, 177
 - valores influyentes, 175
 - valores atípicos, 173
- interpretación de la ecuación de regresión, 166
 - interacciones y principales efectos, 170
 - multicolinealidad, 168
 - variables de confusión, 168
 - variables predictoras correlacionadas, 166
- regresión lineal múltiple, 147
 - ejemplo, estimación del valor de las viviendas, 148
 - evaluación del modelo, 149
 - regresión ponderada, 156
 - selección del modelo y regresión escalonada, 152
 - validación cruzada, 151
- regresión lineal simple, 139
 - ecuación de regresión, 141
 - mínimos cuadrados, 145
 - pronóstico frente a explicación, 146
 - valores ajustados y residuos, 143
- regresión logística, 203
 - comparación con la regresión lineal, 210
- regresión polinomial y por spline, 183
 - modelos aditivos generalizados, 187
 - polinomial, 183
 - splines, 185
- variables de tipo factor en la, 160
 - con muchos niveles, 163
 - representación de variables ficticias, 161
 - variables de tipo factor ordenadas, 165
- pronóstico con, 157
 - intervalos de pronóstico y de confianza, 158
 - peligros de la extrapolación, 158
- regresión a la media, 55
- regresión de cresta, 271
- regresión de mínimos cuadrados, 145
- regresión de mínimos cuadrados ordinarios (MCO), 145
- regresión de todos los subconjuntos, 154
- regresión lasso, 155, 271
- regresión lineal
 - comparación con la regresión logística, 210
 - examen de los residuos para ver si se puede mejorar el ajuste, 264

- modelo lineal generalizado (GLM), 206
- problemas de multicolinealidad causados por la codificación One-Hot, 162
- múltiple, 147
 - ejemplo, estimación del valor de las viviendas, 148
 - evaluación del modelo, 149
 - validación cruzada, 151
 - regresión ponderada, 156
 - selección del modelo y regresión por pasos, 152
 - simple, 139
 - ecuación de regresión, 141
 - mínimos cuadrados, 145
 - valores ajustados y residuos, 143
- regresión lineal logística, 229
- regresión logística, 203
 - ajuste utilizando el modelo aditivo
 - generalizado, 229
 - comparación con la regresión
 - lineal, 210
 - evaluación del modelo, 211
 - función de respuesta logística y logit, 204
 - interpretación de los coeficientes y de la razón de oportunidades, 208
 - problemas de multicolinealidad causados por la codificación One-Hot, 162
 - valores pronosticados por, 208
 - y el modelo lineal generalizado, 207
- regresión no lineal, 183
- regresión penalizada, 155, 171
- regresión polinomial, 183
- regresión ponderada, 156
- regresión por pasos, 154
- regresión spline, 183, 185
- regularización, 268
 - evitación del sobreajuste con, 268
 - regularización L1, 269
 - regularización L2, 269
- relación señal/ruido (SNR), 240
- remuestreo, 65, 96-103, 124
 - bootstrapping frente a, 65
 - pruebas de permutación, 97-103, 120
 - pruebas exhaustiva y bootstrap, 102
 - ejemplo de adherencia de la web, 98-102
 - uso en las pruebas de chi cuadrado, 124, 126
 - pruebas de permutación y bootstrap, 102
- representación de objetos (datos espaciales), 6

- representatividad, 48
 - a través del muestreo aleatorio, 48
- residuos, análisis de,
 - en regresión lineal simple, 143
- residuos anormales, 172
- residuos de Pearson, 124
- residuos estandarizados
 - definidos, 172
 - examen para detectar valores atípicos, 174
 - histogramas de, para la regresión de los datos de viviendas, 179
- respuesta, 6, 142, 143
 - aislamiento de la relación entre las variables predictoras y la, 189
- resultado, 5
- resultados binarios, 77
- RMSE
 - (ver error cuadrático medio)
 - Rho de Spearman, 34
- robusto, 8
- RSE (ver error residual estándar)
- RSS (ver suma de los cuadrados de los residuos)

S

- selección de características
 - usando el análisis discriminante, 200
- selección hacia delante, 154-155
- selección hacia atrás, 155
- sensibilidad, 218-219, 222
- sesgo, 50
 - estimaciones sesgadas del clasificador de Bayes ingenuo, 196
- sesgo de la muestra, 49
- sesgo de selección, 54-57
 - formas típicas del, 55
 - regresión a la media, 55-56
- sesgo del muestreo de autoselección, 50
- significación estadística, 103
- significación práctica frente a, 108
- significación práctica vs. significación estadística, 108
- sitio web de R-Tutorial, 4
- sobreajuste, 112
 - evitación usando la regularización, 268

Estadística práctica para ciencia de datos con R y Python

sobremuestreo, 226
y aumento/disminución de la ponderación, 226
SQL (Structured Query Language), 4
SS
(ver suma de cuadrados)
suavizadores de diagramas de dispersión, 178
subconjunto aleatorio de variables (en bosque aleatorio), 255
submuestreo, 225
(ver también sobremuestreo)
sujetos, 88
suma de cuadrados (SS), 117, 121
suma de cuadrados dentro del grupo (SS), 287
suma de los cuadrados de los residuos (RSS), 145, 271
sustentación, 220, 228
elevación y, 229
sustitución (en muestreo), 48, 97
en pruebas de permutación bootstrap, 102
muestra con sustitución, 62
Synthetic Minority Oversampling Technique
(ver algoritmo SMOTE)

T

tablas de contingencia, 40
resumen de dos variables categóricas, 39
tablas de frecuencias, 22
ejemplo, tasas de homicidios por estado, 25
tablas pivot, 40
(ver también tablas de contingencia)
tasa de fallos, estimación, 83
tasa de falsos descubrimientos, 112, 114
tasa de falsos positivos, 222
tau de Kendall, 34
tendencia central (ver estimación de la localización)
teorema del límite central, 57, 60
distribución t de Student y, 74
teoría del cisne negro, 72
tipos de datos
recursos de lecturas complementarias, 4
términos clave para, 2
tratamiento, 88
Tukey, John Wilder, 1

V

- validación cruzada, 151
 - uso en hiperparámetros, 279-281
 - uso para probar los valores de los hiperparámetros, 270
 - uso para seleccionar los componentes principales, 292
- validación cruzada de k-fold, 152
- validación fuera de la muestra, 152
- valor con circunflejo, 175
- valor esperado, 29
- valor medio (ver media)
- valoración de máxima verosimilitud (MLE), 210
- valores p, 103-110, 121
 - alfa, 107
 - ajuste de, 113
 - ciencia de datos y, 109
 - controversia sobre el uso de, 107
 - distribución de chi cuadrado y, 126
 - errores de tipo 1 y 2, 108
 - estadístico t y, 154
 - significado práctico y, 108
- valores ajustados, 143
 - regresión lineal simple, 143
- valores atípicos, 8, 11
 - coeficiente de correlación y, 31
 - en diagnósticos de regresión, 172
 - en diagramas de caja, 21
- valores influyentes, 175
- valores pronosticados, 143
 - (ver también valores ajustados)
- variabilidad, estimación de la, 13-19
 - desviación estándar y estimaciones relacionadas, 15
 - ejemplo, tasas de homicidios por población estatal, 18
 - percentiles, 17
 - terminología clave, 14
- variabilidad muestral, 58
- variables
 - exploración de dos o más, 36
 - datos numéricos y categóricos, 41
 - términos clave, 36
 - uso de la agrupación hexagonales y el diagrama de contorno, 36-39
 - variables categóricas, 39

Estadística práctica para ciencia de datos con R y Python

- visualización de varias variables, 43
- variables binarias, 160-161, 237, 308, 311, 314
- variables categóricas, 160-161, 308
 - (ver también variables de tipo factor)
- conversión a variables ficticias, 116, 160
- variables de confusión, 164, 166, 168
- variables de tipo factor, 160
 - binarias, razón de oportunidades para, 208
 - codificaciones diferentes, 163
 - representaciones de variables ficticias, 161
 - ordenadas, 165
- variables de tipo factor ordenadas, 165
- variables dependientes, 6, 143
 - (ver también respuesta)
- variables dominantes, 310
- variables ficticias, 160-161
 - representación de variables de tipo factor en la regresión, 161
- variables ficticias binarias, 163, 236
- variables independientes, 170
 - (ver también variables predictoras)
- variables indicadoras, 5, 160
- variables numéricas
 - conversión de variables de tipo factor ordenadas a, 165
 - distancia de Mahalanobis, 199, 236
- variables predictoras, 6, 54, 143
 - correlacionadas, 166
 - estadístico t y, 155
 - interacciones y efectos principales, 170
- variables predictoras correlacionadas, 166
- variables proxy, 98
- variación debida al azar, 101
- varianza, 14
 - análisis de la (ANOVA), 81, 117
 - equilibrio entre sesgo y varianza en el ajuste del modelo, 241
- vecinos (en K-vecinos más cercanos), 239
- visualizaciones, 7 (ver también gráficos)

X

- XGBoost, 265
 - hiperparámetros, 265, 272, 274
 - utilización de la regularización para evitar el sobreajuste, 274

Sobre los autores

Peter Bruce es el fundador y artífice de la expansión del Institute for Statistics Education at Statistics.com, en el que se imparten actualmente unos cien cursos de estadística. De estos, aproximadamente un tercio están orientados a científicos de datos. Con la contratación de los mejores autores como instructores y la creación de una estrategia de marketing para llamar la atención de científicos de datos profesionales, Peter ha desarrollado tanto una visión amplia del mercado objetivo como su propia experiencia para alcanzarlo.

Andrew Bruce cuenta con más de 30 años de experiencia en estadística y ciencia de datos que ha desarrollado trabajando en academias, para el gobierno y en distintas empresas. Tiene un doctorado en estadística por la Universidad de Washington y ha publicado numerosos artículos en revistas especializadas. Ha desarrollado soluciones basadas en estadística para resolver una gran variedad de problemas a los que se enfrentan muchas industrias, desde firmas financieras consolidadas hasta nuevas empresas de Internet, facilitando una comprensión profunda del lado práctico de la ciencia de datos.

Peter Gedeck tiene una experiencia de más de 30 años en informática científica y ciencia de datos. Después de trabajar 20 años como químico informático en Novartis, actualmente trabaja como científico de datos senior en Collaborative Drug Discovery. Se ha especializado en el desarrollo de algoritmos de aprendizaje automático para pronosticar propiedades biológicas y fisicoquímicas de posibles futuros fármacos. Coautor de *Data Mining for Business Analytics*, es doctor en química por la Universidad de Erlangen-Nürnberg en Alemania y estudió matemáticas en Fernuniversität Hagen, Alemania.

Colofón

El animal que aparece en la portada de *Estadística práctica para ciencia de datos con R y Python* es un cangrejo de orilla forrado (*Pachygrapsus crassipes*), también conocido como cangrejo de orilla rayado. Se encuentra a lo largo de las costas y playas del océano Pacífico en América del Norte, América Central, Corea y Japón. Estos crustáceos viven debajo de las rocas, en charcas y en grietas. Pasan aproximadamente la mitad de su tiempo en tierra y periódicamente regresan al agua para humedecer las branquias.

El cangrejo de orilla rayado recibe su nombre por las rayas verdes de su caparazón marrón y negro. Tiene las pinzas rojas y las patas moradas, presentando también un colorido de rayas o moteado. El cangrejo crece hasta alcanzar un tamaño generalmente de 3 a 5 centímetros; las hembras son un poco más pequeñas. Los ojos los tienen situados al final de unas proyecciones flexibles que salen del caparazón y pueden rotar, proporcionándoles un campo de visión completo mientras se desplazan.

Los cangrejos son omnívoros, se alimentan principalmente de algas, pero también de moluscos, gusanos, hongos, animales muertos y otros crustáceos (dependiendo de lo que encuentren). Mudan la carcasa muchas veces a medida que crecen hasta llegar a la edad adulta, para ello toman agua para expandirse y romper la vieja cáscara. Una vez lo han conseguido, viven unas horas difíciles hasta liberarse y luego deben esconderse hasta que la nueva cáscara se endurezca.

Muchos de los animales de las portadas de O'Reilly están en peligro de extinción; todos ellos son importantes para el mundo.

La ilustración de la portada es de Karen Montgomery, inspirada en un grabado en blanco y negro del *Pictorial Museum of Animated Nature*.

