Divide y vencerás - clasificación mediante árboles de decisión y reglas, parte I

Al momento de decidir entre ofertas de trabajo, muchas personas comienzan haciendo listas de pros y contras, y luego eliminan opciones usando reglas simples. Por ejemplo, pueden decidir: “Si tengo que viajar más de una hora, seré infeliz” o “Si gano menos de $50,000, no puedo mantener a mi familia”.
De esta manera, la compleja decisión de predecir la felicidad profesional futura de una persona puede reducirse a una serie de decisiones simples.

En esta primera parte del tema divide y conquista, se tratan los árboles de decisión y los aprendices de reglas, dos métodos de aprendizaje automático que también toman decisiones complejas a partir de conjuntos de opciones simples. Estos métodos presentan su conocimiento en forma de estructuras lógicas que pueden entenderse sin conocimientos estadísticos. Este aspecto hace que estos modelos sean particularmente útiles para la estrategia empresarial y la mejora de procesos.

Al final de este capítulo, habrá aprendido:

- Cómo los árboles y las reglas dividen los datos de forma “codiciosa, glotona, …” (*greedy*) en segmentos interesantes
- Los árboles de decisión y los aprendices de reglas de clasificación más comunes, incluidos los algoritmos C5.0, 1R y RIPPER
- Cómo usar estos algoritmos para realizar tareas de clasificación del mundo real, como identificar préstamos bancarios riesgosos y hongos venenosos

Comenzaremos examinando los árboles de decisión y luego veremos las reglas de clasificación. Luego, resumiremos lo que hemos aprendido con una vista previa de los temas posteriores, que analizan los métodos que usan árboles y reglas como base para técnicas de aprendizaje automático más avanzadas.

Comprensión de los árboles de decisión

Los aprendices de árboles de decisión son clasificadores poderosos que utilizan una estructura de árbol para modelar las relaciones entre las características y los resultados potenciales. Como se ilustra en la siguiente figura, esta estructura recibió su nombre porque refleja la forma en que comienza un árbol literal, con un tronco ancho en la base que se divide en ramas cada vez más estrechas a medida que avanza hacia arriba. De la misma manera, un

clasificador de árbol de decisión utiliza una estructura de decisiones ramificadas para canalizar los ejemplos hacia un valor de clase final previsto.

Para entender mejor cómo funciona esto en la práctica, consideremos el siguiente árbol, que predice si se debe aceptar una oferta de trabajo. Una oferta de trabajo bajo consideración comienza en el nodo raíz, desde donde pasa a través de los nodos de decisión, que requieren que se tomen decisiones basadas en los atributos del trabajo. Estas decisiones dividen los datos en ramas que indican los resultados potenciales de una decisión. Aquí se representan como resultados de sí o no, pero en otros casos, puede haber más de dos posibilidades.

Si se puede tomar una decisión final, el árbol termina en **nodos de hoja** (también conocidos como **nodos terminales**) que denotan la acción que se debe tomar como resultado de la serie de decisiones. En el caso de un modelo predictivo, los nodos de hoja proporcionan el resultado esperado dada la serie de eventos en el árbol.

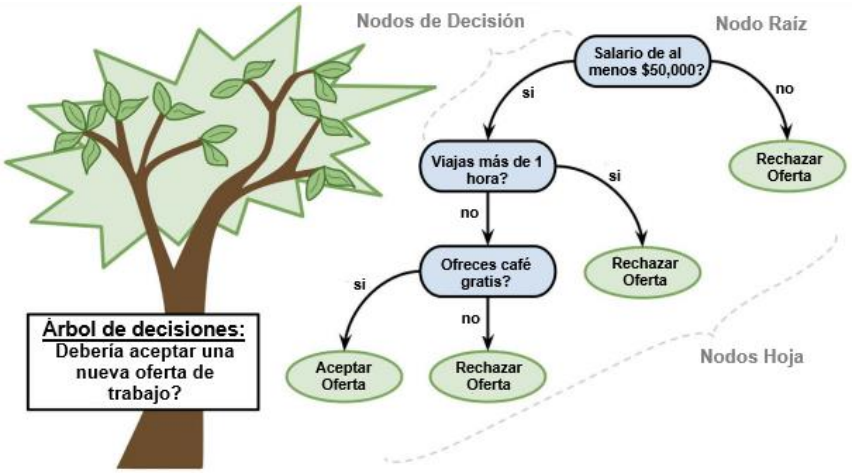

Figura 1: Un árbol de decisión que representa el proceso de determinar si se debe aceptar una nueva oferta de trabajo.

Una gran ventaja de los algoritmos de árboles de decisión es que la estructura de árbol, similar a un diagrama de flujo, no es solo para uso interno de la máquina. Una vez creado el modelo, muchos algoritmos de árboles de decisión generan la estructura resultante en un formato legible para humanos. Esto proporciona información sobre cómo y por qué el modelo funciona o no funciona bien para una tarea en particular. Esto también hace que los árboles de decisión sean particularmente apropiados para aplicaciones en las que el mecanismo de clasificación debe ser transparente por razones legales o si los resultados deben compartirse con otros para informar futuras prácticas comerciales. Con esto en mente, algunos usos potenciales incluyen los siguientes:

- Modelos de calificación crediticia en los que los criterios que hacen que se rechace a un solicitante deben estar claramente documentados y libres de sesgos
- Estudios de marketing del comportamiento del cliente, como la satisfacción o la pérdida de clientes, que se compartirán con la administración o las agencias de publicidad
- Diagnóstico de afecciones médicas basado en mediciones de laboratorio, síntomas o tasas de progresión de la enfermedad

Aunque las aplicaciones anteriores ilustran el valor de los árboles para informar los procesos de toma de decisiones, esto no quiere decir que su utilidad termine aquí. De hecho, los árboles de decisión son una de las técnicas de aprendizaje automático más utilizadas y se pueden aplicar para modelar casi cualquier tipo de datos, a menudo con un excelente rendimiento desde el primer momento.

Dicho esto, a pesar de su amplia aplicabilidad, vale la pena señalar que hay algunos escenarios en los que los árboles pueden no ser una opción ideal. Esto incluye tareas en las que los datos tienen muchas características nominales con muchos niveles o una gran cantidad de características numéricas. Estos casos pueden dar como resultado una gran cantidad de decisiones y un árbol demasiado complejo. También pueden contribuir a la tendencia de los árboles de decisión a sobreajustarse a los datos, aunque, como veremos pronto, incluso esta debilidad se puede superar ajustando algunos parámetros simples.

Divide y vencerás

Los árboles de decisión se construyen utilizando una heurística llamada **partición recursiva**. Este enfoque también se conoce comúnmente como divide y vencerás porque divide los datos en subconjuntos, que luego se dividen repetidamente en subconjuntos aún más pequeños, y así sucesivamente, hasta que el proceso se detiene cuando el algoritmo determina que los datos dentro de los subconjuntos son suficientemente homogéneos o se ha cumplido otro criterio de detención.

Para ver cómo la división de un conjunto de datos puede crear un árbol de decisión, imagina un nodo raíz que crecerá hasta convertirse en un árbol maduro. Al principio, el nodo raíz representa todo el conjunto de datos, ya que no se ha producido ninguna división. Aquí, el algoritmo del árbol de decisión debe elegir una característica sobre la que dividir; idealmente, elige la característica más predictiva de la clase de destino.

Luego, los ejemplos se dividen en grupos según los valores distintivos de esta característica, y se forma el primer conjunto de ramas del árbol.

A medida que avanza por cada rama, el algoritmo continúa dividiendo y conquistando los datos, eligiendo la mejor característica candidata cada vez para crear otro nodo de decisión hasta que se alcanza un criterio de detención. La división y conquista podría detenerse en un nodo si:

- Todos (o casi todos) los ejemplos en el nodo tienen la misma clase
- No quedan características restantes para distinguir entre los ejemplos
- El árbol ha crecido hasta un límite de tamaño predefinido

Para ilustrar el proceso de construcción del árbol, consideremos un ejemplo sencillo. Imagina que trabaja para un estudio de Hollywood, donde su función es decidir si el estudio debe seguir adelante con la producción de los guiones presentados por nuevos autores prometedores. Después de regresar de unas vacaciones, su escritorio está repleto de propuestas. Sin tiempo para leer cada propuesta de principio a fin, decide desarrollar un algoritmo de árbol de decisiones para predecir si una posible película entraría en una de tres categorías: *éxito de la crítica*, *éxito comercial* o *fracaso de taquilla*.

Para obtener datos para crear el modelo de árbol de decisiones, recurres a los archivos del estudio para examinar los factores que conducen al éxito o al fracaso de los 30 estrenos más recientes de la empresa. Rápidamente notas una relación entre el presupuesto estimado de rodaje de la película, la cantidad de celebridades de primera línea que esperan papeles protagónicos y el nivel de éxito de la película. Emocionado por este hallazgo, creas un diagrama de dispersión para ilustrar el patrón:

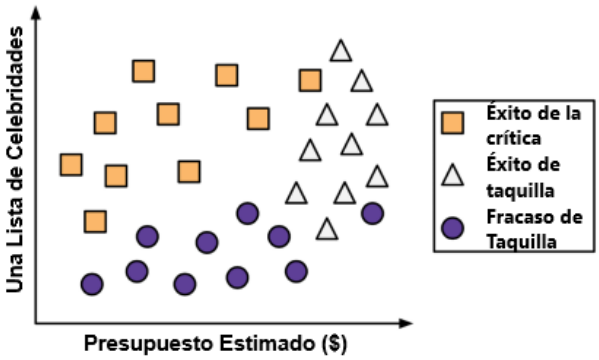

Figura 2: Diagrama de dispersión que representa la relación entre el presupuesto de una película y el número de celebridades.

Usando la estrategia de divide y vencerás, puedes construir un árbol de decisiones simple a partir de estos datos. Primero, para crear el nodo raíz del árbol, divides la característica que indica el número de celebridades, dividiendo las películas en grupos con y sin un número significativo de estrellas de primera línea:

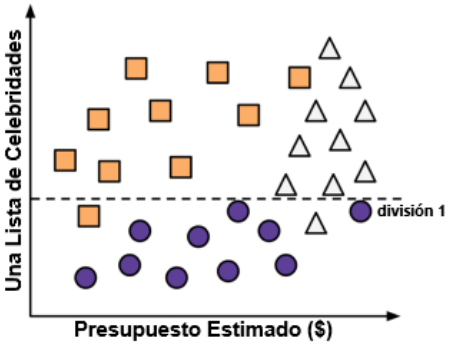

Figura 3: La primera división del árbol de decisiones divide los datos en películas con un número alto y bajo de celebridades.

A continuación, entre el grupo de películas con un número mayor de celebridades, realizas otra división entre películas con y sin un presupuesto alto:

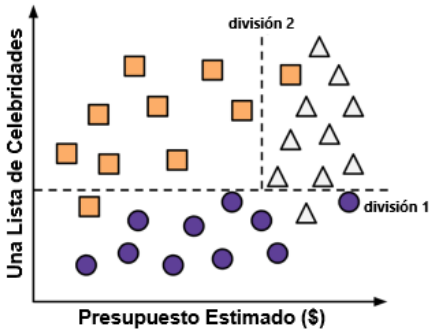

Figura 4: La segunda división del árbol de decisiones divide aún más las películas con un número alto de celebridades en aquellas con presupuestos altos y bajos.

En este punto, has dividido los datos en tres grupos. El grupo en la esquina superior izquierda del diagrama está compuesto enteramente por películas aclamadas por la crítica. Este grupo se distingue por una gran cantidad de celebridades y un presupuesto relativamente bajo. En la esquina superior derecha, casi todas las películas son éxitos de taquilla con presupuestos altos y muchas celebridades. El grupo final, que tiene poco poder ‘estelar’ pero presupuestos que van desde pequeños a grandes, contiene los fracasos.

Si lo deseas, puedes continuar dividiendo y conquistando los datos dividiéndolos en rangos cada vez más específicos de presupuesto y cantidad de celebridades hasta que cada uno de los valores actualmente mal clasificados se clasifiquen correctamente en su propia partición diminuta.

Sin embargo, no es aconsejable sobreajustar un árbol de decisiones de esta manera. Aunque no hay nada que impida que el algoritmo divida los datos indefinidamente, las decisiones demasiado específicas no siempre se generalizan de manera más amplia. Por lo tanto, eliges

evitar el problema del sobreajuste deteniendo el algoritmo aquí, ya que más del 80 por ciento de los ejemplos en cada grupo son de una sola clase. Este es el criterio de detención para el modelo de árbol de decisiones.

Es posible que hayas notado que las líneas diagonales podrían haber dividido los datos de manera aún más clara. Esta es una limitación de la representación del conocimiento del árbol de decisiones, que utiliza **divisiones paralelas al eje**. El hecho de que cada división considere una característica a la vez impide que el árbol de decisiones forme límites de decisión más complejos. Por ejemplo, una línea diagonal podría crearse mediante una decisión que pregunte: "¿La cantidad de celebridades es mayor que el presupuesto estimado?" Si es así, entonces "será un éxito rotundo".

El modelo para predecir el éxito futuro de las películas se puede representar en un árbol simple, como se muestra en el siguiente diagrama. Cada paso del árbol muestra la fracción de ejemplos que caen en cada clase, lo que muestra cómo los datos se vuelven más homogéneos a medida que las ramas se acercan a una hoja.

Para evaluar un nuevo guión de película, sigues las ramas a través de cada decisión hasta que se haya predicho el éxito o el fracaso del guión. Con este enfoque, podrás identificar rápidamente las opciones más prometedoras entre la lista de guiones pendientes y volver a trabajar en un trabajo más importante, como escribir un discurso de aceptación de los Premios de la Academia.

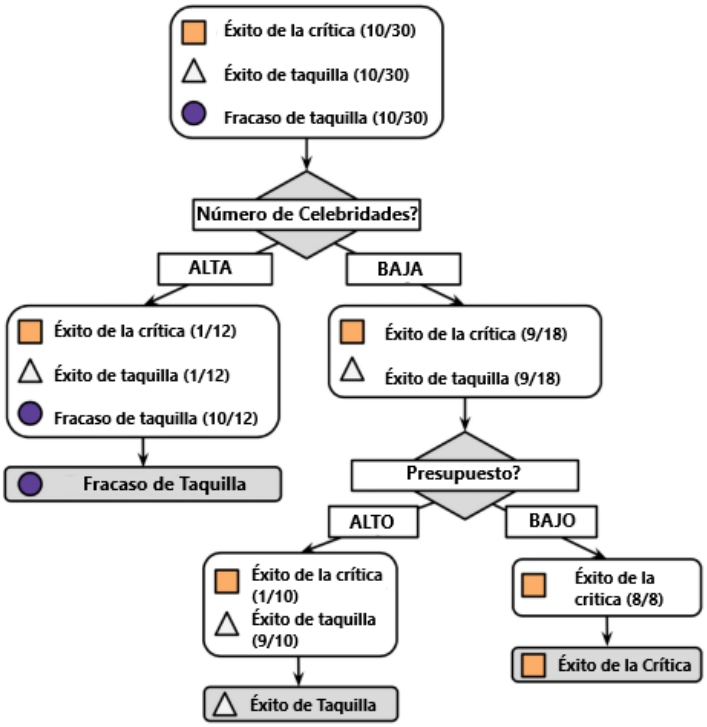

Figura 5: Un árbol de decisión creado a partir de datos históricos de películas puede predecir el rendimiento de películas futuras.

Dado que los datos del mundo real contienen más de dos características, los árboles de decisión rápidamente se vuelven mucho más complejos, con muchos más nodos, ramas y hojas. En la siguiente sección, aprenderás acerca de un algoritmo popular para crear modelos de árboles de decisión de manera automática.

El algoritmo de árbol de decisión C5.0

Existen numerosas implementaciones de árboles de decisión, pero una de las más conocidas es el algoritmo C5.0. Este algoritmo fue desarrollado por el científico informático J. Ross Quinlan como una versión mejorada de su algoritmo anterior, C4.5, que a su vez es una mejora con respecto a su algoritmo **Iterative Dichotomiser 3** (ID3). Aunque Quinlan comercializa C5.0 para clientes comerciales (consulta http://www.rulequest.com/ para obtener más detalles), el código fuente de una versión de un solo subproceso del algoritmo se hizo público y, por lo tanto, se incorporó a programas como R.

> Para complicar aún más las cosas, una alternativa popular de código abierto basada en Java a C4.5, llamada J48, está incluida en el paquete RWeka de R. Como las diferencias entre C5.0, C4.5 y J48 son menores, los principios de este documento se aplican a cualquiera de estos tres métodos y los algoritmos deben considerarse sinónimos.

El algoritmo C5.0 se ha convertido en el estándar de la industria para producir árboles de decisión porque funciona bien para la mayoría de los tipos de problemas desde el primer momento. En comparación con otros modelos avanzados de aprendizaje automático, como los que se describen posteriormente, los árboles de decisión creados por C5.0 generalmente funcionan casi tan bien, pero son mucho más fáciles de entender e implementar. Además, como se muestra en la siguiente tabla, las debilidades del algoritmo son relativamente menores y se pueden evitar en gran medida.

Fortalezas	**Debilidades**
• Un clasificador multipropósito que funciona bien en muchos tipos de problemas • Proceso de aprendizaje altamente automático, que puede manejar características numéricas o nominales, así como datos faltantes • Excluye características no importantes • Puede usarse en conjuntos de datos pequeños y grandes • Da como resultado un modelo que puede interpretarse sin conocimientos matemáticos (para árboles relativamente pequeños) • Es más eficiente que otros modelos complejos	• Los modelos de árboles de decisión suelen estar sesgados hacia divisiones en características que tienen una gran cantidad de niveles • Es fácil sobreajustar o subajustar el modelo • Puede tener problemas para modelar algunas relaciones debido a la dependencia de divisiones paralelas a los ejes • Pequeños cambios en los datos de entrenamiento pueden resultar en grandes cambios en la lógica de decisión • Los árboles grandes pueden ser difíciles de interpretar y las decisiones que toman pueden parecer contraintuitivas

Para simplificar las cosas, nuestro ejemplo anterior de árbol de decisión ignoró las matemáticas involucradas en cómo una máquina emplearía una estrategia de dividir y vencer. Exploremos esto con más detalle para examinar cómo funciona esta heurística en la práctica.

Elección de la mejor división

El primer desafío que enfrentará un árbol de decisión es identificar qué característica dividir. En el ejemplo anterior, buscamos una forma de dividir los datos de modo que las particiones resultantes contuvieran ejemplos principalmente de una sola clase.

El grado en el que un subconjunto de ejemplos contiene solo una sola clase se conoce como **pureza**, y cualquier subconjunto compuesto solo por una sola clase se denomina **puro**.

Existen varias mediciones de pureza que se pueden utilizar para identificar el mejor candidato para la división del árbol de decisión. C5.0 utiliza la **entropía**, un concepto tomado de la teoría de la información que cuantifica la aleatoriedad o el desorden dentro de un conjunto de valores de clase. Los conjuntos con alta entropía son muy diversos y brindan poca información sobre otros elementos que también pueden pertenecer al conjunto, ya que no hay una similitud aparente. El árbol de decisión espera encontrar divisiones que reduzcan la entropía, lo que en última instancia aumenta la homogeneidad dentro de los grupos.

Normalmente, la entropía se mide en **bits**. Si solo hay dos clases posibles, los valores de entropía pueden variar de 0 a 1. Para *n* clases, la entropía varía de 0 a $\log_2(n)$. En cada caso, el valor mínimo indica que la muestra es completamente homogénea, mientras que el valor máximo indica que los datos son lo más diversos posible y ningún grupo tiene ni siquiera una pequeña pluralidad.

En la noción matemática, la entropía se especifica como:

$$\text{Entropia}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

En esta fórmula, para un segmento dado de datos (S), el término c se refiere al número de niveles de clase y p_i se refiere a la proporción de valores que caen en el i-ésimo nivel de clase. Por ejemplo, supongamos que tenemos una partición de datos con dos clases: rojo (60 por ciento) y blanco (40 por ciento). Podemos calcular la entropía como:

```
> -0.60 * log2(0.60) - 0.40 * log2(0.40)
[1] 0.9709506
```

Podemos visualizar la entropía para todos los posibles arreglos de dos clases. Si sabemos que la proporción de ejemplos en una clase es *x*, entonces la proporción en la otra clase es (1 – *x*). Usando la función curve(), podemos entonces trazar la entropía para todos los valores posibles de *x*:

```
> curve(-x * log2(x) - (1 - x) * log2(1 - x),
+  col = "red", xlab = "x", ylab = "Entropia", lwd = 4)
```

Esto da como resultado el siguiente gráfico:

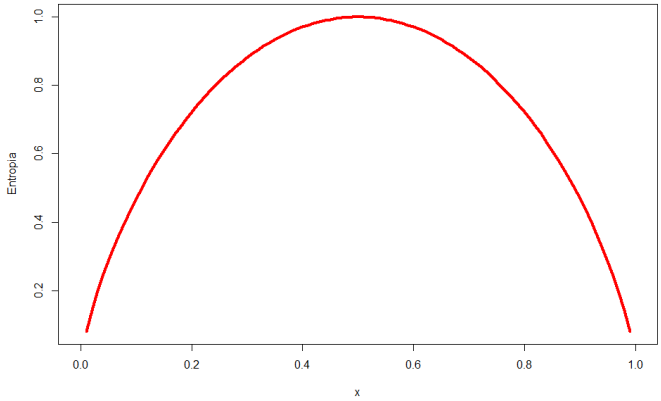

Figura 6: La entropía total a medida que varía la proporción de una clase en un resultado de dos clases.

Como lo ilustra el pico en $x = 0.50$, una división 50-50 da como resultado la entropía máxima. A medida que una clase domina cada vez más a la otra, la entropía se reduce a cero.

Para usar la entropía para determinar la característica óptima sobre la cual dividir, el algoritmo calcula el cambio en homogeneidad que resultaría de una división sobre cada característica posible, una medida conocida como **ganancia de información**. La ganancia de información para una característica F se calcula como la diferencia entre la entropía en el segmento antes de la división (S_1) y las particiones resultantes de la división (S_2):

$$\text{GananciaInformacion}(F) = \text{Entropia}(S_1) - \text{Entropia}(S_2)$$

Una complicación es que después de una división, los datos se dividen en más de una partición. Por lo tanto, la función para calcular Entropia(S_2) debes considerar la entropía total en todas las particiones resultantes de la división. Para ello, ponderas la entropía de cada partición según la proporción de todos los registros que caen en esa partición. Esto se puede expresar en una fórmula como:

$$\text{Entropia}(S) = \sum_{i=1}^{n} w_i \text{Entropia}(P_i)$$

En términos simples, la entropía total resultante de una división es la suma de la entropía de cada una de las *n* particiones ponderada por la proporción de ejemplos que caen en la partición (w_i).

Cuanto mayor sea la ganancia de información, mejor será una característica para crear grupos homogéneos después de una división en esa característica. Si la ganancia de información es cero, no hay reducción en la entropía para la división en esta característica. Por otro lado, la ganancia de información máxima es igual a la entropía antes de la división. Esto implicaría que la entropía después de la división es cero, lo que significa que la división da como resultado grupos completamente homogéneos.

Las fórmulas anteriores suponen características nominales, pero los árboles de decisión también utilizan la ganancia de información para la división en características numéricas. Para ello, una práctica habitual consiste en probar varias divisiones que dividen los valores en grupos mayores o menores que un umbral. Esto reduce la característica numérica a una característica categórica de dos niveles que permite calcular la ganancia de información de la forma habitual. Se elige para la división el punto de corte numérico que produce la mayor ganancia de información.

Aunque se utiliza en C5.0, la ganancia de información no es el único criterio de división que se puede utilizar para construir árboles de decisión. Otros criterios de uso común son el **índice de Gini**, la **estadística de chi-cuadrada** y la **razón de ganancia**. Para una revisión de estos criterios (y muchos más), consulta el artículo de J. Mingers, *An Empirical Comparison of Selection Measures for Decision-Tree Induction*, J. Machine Learning, 1989, Vol. 3, pp. 319-342.

Poda del árbol de decisiones

Como se mencionó anteriormente, un árbol de decisiones puede seguir creciendo indefinidamente, eligiendo características de división y dividiéndose en particiones cada vez más pequeñas hasta que cada ejemplo esté perfectamente clasificado o el algoritmo se quede sin características para dividir. Sin embargo, si el árbol crece demasiado, muchas de las decisiones que tome serán demasiado específicas y el modelo se ajustará en exceso a los datos de entrenamiento.

El proceso de **poda** de un árbol de decisiones implica reducir su tamaño de modo que se generalice mejor a los datos no vistos.

Una solución a este problema es detener el crecimiento del árbol una vez que alcanza una cierta cantidad de decisiones o cuando los nodos de decisión contienen solo una pequeña cantidad de ejemplos. Esto se llama **detención temprana** o **poda previa** del árbol de decisiones. Como el árbol evita realizar trabajo innecesario, esta es una estrategia atractiva. Sin embargo, una desventaja de este enfoque es que no hay forma de saber si el árbol pasará por alto patrones sutiles pero importantes que habría aprendido si hubiera crecido a un tamaño mayor.

Una alternativa, llamada **poda posterior**, implica hacer crecer un árbol que sea intencionalmente demasiado grande y podar los nodos de las hojas para reducir el tamaño del árbol a un nivel más apropiado. Este suele ser un enfoque más eficaz que la poda previa porque es bastante difícil determinar la profundidad óptima de un árbol de decisión sin hacerlo crecer primero. Podar el árbol más tarde permite al algoritmo estar seguro de que se descubrieron todas las estructuras de datos importantes.

Los detalles de implementación de las operaciones de poda son muy técnicos y exceden el alcance de este curso. Para una comparación de algunos de los métodos disponibles, consulta el trabajo de: Esposito, F, Malerba, D, Semeraro, G.: A Comparative Analysis of Methods for Pruning Decision Trees, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, vol. 19, pp. 476-491.

Uno de los beneficios del algoritmo C5.0 es que tiene opiniones firmes sobre la poda: se ocupa de muchas de las decisiones automáticamente utilizando valores predeterminados razonables. Su estrategia general es podar el árbol posteriormente. Primero, se crea un árbol grande que se ajusta en exceso a los datos de entrenamiento. Luego, se eliminan los nodos y las ramas que tienen poco efecto en los errores de clasificación. En algunos casos, se mueven ramas enteras hacia arriba en el árbol o se reemplazan por decisiones más simples. Estos procesos de injerto de ramas se conocen como **elevación de subárboles** y **reemplazo de subárboles**, respectivamente.

Obtener el equilibrio correcto entre sobreajuste y subajuste es un poco un arte, pero si la precisión del modelo es vital, puede que valga la pena invertir algo de tiempo con varias opciones de poda para ver si mejora el rendimiento del conjunto de datos de prueba. Como verás pronto, una de las fortalezas del algoritmo C5.0 es que es muy fácil ajustar las opciones de entrenamiento.