APRENDIZAJE
ÁRBOL DE DECISIÓN
Data Scientist
Jhoan Esteban Ruiz Borja Msc

Definiciones básicas:

- **Aprendizaje inductivo:** Es aquel que se basa en el descubrimiento de patrones a partir de ejemplos.
- **Hipótesis de aprendizaje inductivo:** Cualquier hipótesis encontrada que clasifique un número suficientemente grande de ejemplos de entrenamiento clasificará otros ejemplos no observados.
- **Razonamiento deductivo:** Partiendo de unas premisas se llega necesariamente a una conclusión. No aporta información nueva.
- **Razonamiento abductivo:** Partiendo del conocimiento de unos efectos (síntomas) se llega a la causa (enfermedad).
- **Entropía:** n° de bits necesarios para codificar un suceso. Cuantos más bits más información menos probable del suceso. Es decir, al aparecer más cuando aparece aporta menos información al conjunto que cuando aparece un suceso más raro.
- **Ganancia de información:** Información del conjunto menos la que aporta el atributo X. Cuanto mayor sea menor es la cantidad de información que aporta X. es decir, es un suceso muy probable lo que implica que sea un buen candidato como atributo representativo el conjunto.
- **Consenso:** Se denomina consenso al acuerdo producido por consentimiento entre todos los miembros de un grupo o entre varios grupos.
- **Miope:** Que no tiene la suficiente capacidad o perspicacia para ver las cosas que son muy claras y fáciles de entender o para darse cuenta de algún asunto.
- **Disyuntiva:** Situación en la que hay que elegir entre dos cosas o soluciones diferentes.

Machine Learning [Tom Mitchell-Chapter 3]

El aprendizaje de árboles de decisión es uno de los métodos más utilizados y prácticos para la inferencia inductiva. Es un método para aproximar funciones de valores discretos que son robustos a datos ruidosos y capaces de aprender expresiones disyuntivas. En este capítulo se describe una familia de algoritmos de aprendizaje de árboles de decisión, que incluye algoritmos ampliamente utilizados como **ID3**, **ASSISTANT** y **C4.5**. Estos métodos de aprendizaje de árboles de decisión buscan espacio de hipótesis totalmente expresivas y así evitar las dificultades de los espacios restringidos de la hipótesis. Su sesgo inductivo es una preferencia por árboles pequeños sobre grandes árboles.

3.1 **INTRODUCCIÓN**

El aprendizaje de árbol de decisiones es un método para aproximar funciones objetivo de valores discretos, en el cual la función aprendida es representada por un árbol de decisión. Los árboles aprendidos también se pueden volver a representar como conjuntos de reglas **if-then** (si-entonces) para mejorar la legibilidad humana. Estos métodos de aprendizaje se encuentran entre los más populares de los algoritmos de inferencia inductiva y se han aplicado con éxito a una amplia gama de tareas como aprender a diagnosticar los casos médicos, también para aprender a evaluar el riesgo de crédito de los solicitantes de préstamos.

3.2 **REPRESENTACIÓN DEL ÁRBOL DE DECISIÓN**

Los árboles de decisión clasifican las instancias de clasificación hacía abajo del árbol desde la raíz hasta cierto nodo hoja, que proporciona la clasificación de la instancia. Cada nodo en el árbol especifica una prueba de algún atributo de la instancia, y cada rama descendente del nodo corresponde a uno de los valores posibles para este atributo. Un ejemplo está clasificado por comenzar en el nodo raíz del árbol, probando el atributo especificado por este nodo, a continuación, mover hacia abajo la rama del árbol correspondiente al valor del atributo en el ejemplo dado. Este proceso se repite entonces para el subárbol con raíz en el nuevo nodo.

Figura 3.1 ilustra un típico aprendizaje de árbol decisión. Este árbol de decisión clasifica sábados por la mañana en función de si son adecuados para jugar al tenis.
Por ejemplo, la instancia

(Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong)

Sería solucionada por la rama más a la izquierda de este árbol de decisión y por lo tanto se clasifica como una instancia negativa (es decir, el árbol predice que PlayTennis = no). Este árbol y el ejemplo utilizado en la Tabla 3.2 para ilustrar el algoritmo de aprendizaje **ID3** se han adaptado de (Quinlan 1986).

En general, los árboles de decisión representan una disyunción de conjunciones de restricciones en los valores de los atributos de las instancias. Cada camino desde la raíz del árbol a una hoja corresponde a un conjunto de pruebas de atributo, y el propio árbol a una disyunción de estas conjunciones. Por ejemplo, el árbol de decisión se muestra en la Figura 3.1 corresponde a la expresión

$$(Outlook = Sunny \wedge Humidity = Normal) \vee (Outlook = Overcast) \vee (Outlook = Rain \wedge Wind = Weak)$$

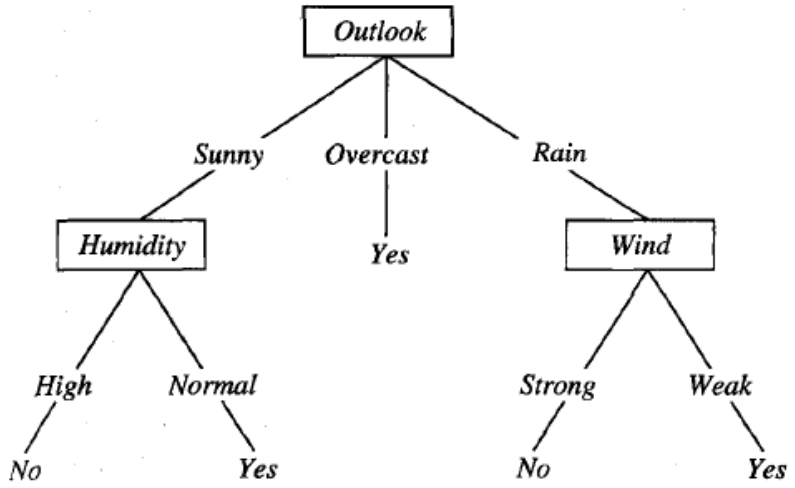

FIGURE 3.1
A decision tree for the concept *PlayTennis*. An example is classified by sorting it through the tree to the appropriate leaf node, then returning the classification associated with this leaf (in this case, *Yes* or *No*). This tree classifies Saturday mornings according to whether or not they are suitable for playing tennis.

3.3 PROBLEMAS ADECUADOS PARA EL APRENDIZAJE DEL ÁRBOL DE DECISIÓN

Aunque una variedad de métodos de aprendizaje de árboles de decisión se ha desarrollado con las capacidades y requisitos algo diferentes, el aprendizaje del árbol de decisión es generalmente el más adecuado a los problemas con las siguientes características:

- **Los casos que son representados por pares atributo-valor**. Los casos son descritos por un conjunto fijo de atributos (por ejemplo, la temperatura) y sus valores (por ejemplo, calientes). La situación más fácil para el aprendizaje del árbol de decisión es cuando cada atributo lleva en un pequeño número de valores disjuntos posibles (por ejemplo, caliente, suave, frío). Sin embargo, extensiones para el algoritmo básico (discuten en la Sección 3.7.2) permiten el manejo de atributos con valores reales, así (por ejemplo, en representación de la temperatura numéricamente).
- **La función objetivo tiene valores de salida discretos**. El árbol de decisiones de la Figura 3.1 asigna una clasificación booleana (por ejemplo, sí o no) para cada ejemplo. Métodos de árbol de decisión se extienden fácilmente a las funciones de aprendizaje con más de dos valores de salida posibles. Una extensión más sustancial permite aprender funciones objetivo con salidas con valores reales, a pesar de la aplicación de los árboles de decisión en este escenario es menos común.
- **Se pueden requerir descripciones disyuntivas**. Como se señaló anteriormente, los árboles de decisión representan naturalmente expresiones disyuntivas.
- **Los datos de entrenamiento pueden contener errores**. Los métodos de aprendizaje de árboles de decisión son robustos a errores, tanto los errores en las

clasificaciones de los ejemplos de entrenamiento y los errores en los valores de los atributos que describen estos ejemplos.

- **Los datos de entrenamiento pueden contener valores de los atributos que faltan**. Métodos de árbol de decisión se pueden utilizar incluso cuando algunos ejemplos de formación tienen valores desconocidos (por ejemplo, si la humedad del día es conocido por sólo algunos de los ejemplos de entrenamiento). Este tema se discute en la Sección 3.7.4.

Muchos problemas prácticos se han encontrado para adaptarse a estas características. El aprendizaje de árboles de decisión, por tanto, se ha aplicado a problemas tales como aprender a clasificar a los pacientes médicos por su enfermedad, mal funcionamiento del equipo por su causa y los solicitantes de préstamo por su probabilidad de incumplimiento en los pagos. Tales problemas, en la que la tarea es clasificar ejemplos en uno de un conjunto discreto de categorías posibles, se refieren a menudo como problemas de clasificación.

El resto de este capítulo está organizado como sigue. Sección 3.4 presenta el algoritmo **ID3** básico para el aprendizaje de árboles de decisión y se muestra su funcionamiento en detalle. Sección 3.5 analiza la búsqueda de espacio de hipótesis realizada por este algoritmo de aprendizaje, contrastando con algoritmos del capítulo 2. Sección 3.6 caracteriza el sesgo inductivo de este árbol de decisión del algoritmo de aprendizaje y explora más generalmente un sesgo inductivo llamado la navaja de Occam, que corresponde a una preferencia por la hipótesis más simple. Sección 3.7 discute el problema de sobreajuste de los datos de entrenamiento, así como estrategias, como poda la regla para resolver este problema. Esta sección también discute un número de temas más avanzados tales como extender el algoritmo para acomodar los atributos de valor real, datos de entrenamiento con inadvertidos atributos y atributos con diferentes costos.

3.4 El ALGORITMO BASICO DE APRENDIZAJE DE ÁRBOL DE DECISIÓN

La mayoría de algoritmos que se han desarrollado para el aprendizaje de árboles de decisión son variaciones sobre un algoritmo de base que emplea una búsqueda codiciosa de arriba hacia abajo a través del espacio de árboles de decisión posible. Este enfoque es el que tiene el algoritmo **ID3 (Quinlan 1986)** y su sucesor **C4.5 (Quinlan 1993)**, que forman el objetivo principal de nuestra discusión aquí. En esta sección presentamos el algoritmo básico de aprendizaje de árboles de decisión, correspondiente aproximadamente al algoritmo **ID3**. En la sección 3.7 consideramos un número de extensiones de este algoritmo básico, incluidas las extensiones incorporadas **C4.5** y otros algoritmos más recientes para el aprendizaje de árboles de decisión.

En nuestro algoritmo básico, **ID3**, los árboles de decisión aprenden mediante la construcción de ellos, desde arriba hacia abajo, ¿comenzando con la pregunta “qué atributo debe probarse en la raíz del árbol? Para responder a esta pregunta, cada atributo de instancia se evalúa mediante una prueba estadística para determinar qué tan bien solo

clasifica los ejemplos de entrenamiento. El mejor atributo es seleccionado y se utiliza como la prueba en el nodo raíz del árbol. Se crea entonces un descendiente del nodo raíz para cada posible valor de este atributo, y los ejemplos de entrenamiento se clasifican en el nodo descendiente apropiado (es decir, por la rama correspondiente valor en el ejemplo para este atributo). El proceso entonces se repite usando los ejemplos de entrenamiento asociados con cada nodo descendiente para seleccionar el mejor atributo para probar en ese punto en el árbol. Esto forma una búsqueda codiciosa de un árbol de decisión aceptable, en la que el algoritmo nunca retrocede para reconsiderar decisiones anteriores. Una versión simplificada del algoritmo, especializado para aprender funciones con valores de booleanos (es decir, el concepto de aprendizaje), se describe en la tabla 3.1.

3.4.1 **¿Qué atributo es el mejor clasificador?**

La elección central en el algoritmo **ID3** es seleccionar qué atributo debe poner a prueba en cada nodo del árbol. Como seleccionar el atributo que es más útil para clasificar los ejemplos. **¿Qué es una buena medida cuantitativa del valor de un atributo?** Definiremos una propiedad estadística, llamada **ganancia de información**, **que mide qué tan bien un atributo dado separa los ejemplos de entrenamiento según su clasificación de destino**. **ID3** utiliza esta medida de **ganancia de información** para seleccionar entre los atributos del candidato en cada paso mientras que crece el árbol.

3.4.1.1 **HOMOGENEIDAD DE LAS MEDIDAS DE ENTROPÍA DE EJEMPLOS**

Para definir la **ganancia de información** precisamente, comenzamos por definir una medida de uso general en teoría de la información, llamada **entropía**, que caracteriza la pureza (impureza) de una colección arbitraria de ejemplos. Dado un conjunto **S**, que contienen los ejemplos positivos y negativos de algún concepto de destino, la entropía de **S** con respecto a esta clasificación booleana es

$$\boldsymbol{Entropía}(\boldsymbol{S}) = -\boldsymbol{P}_{+}\log_2 \boldsymbol{P}_{+} - \boldsymbol{P}_{-}\log_2 \boldsymbol{P}_{-} \qquad (\mathbf{3.1})$$

Donde $\boldsymbol{P}_{+}$, es la proporción de ejemplos positivos en **S** y $\boldsymbol{P}_{-}$, es la proporción de ejemplos negativos en **S**. En todos los cálculos que implican la entropía definimos $\mathbf{0}\log\mathbf{0}$ ser **0**.

Para ilustrar, supongamos que **S** es una colección de **14** ejemplos de algún concepto booleano, que incluyen **9** ejemplos positivos y **5** negativos (nosotros adoptamos la notación **[9+, 5-]** para resumir tal muestra de datos). Entonces la entropía de **S** con respecto a la clasificación booleana es

$$\boldsymbol{Entropía}([\mathbf{9}+,\mathbf{5}-]) = -\left(\frac{\mathbf{9}}{\mathbf{14}}\right)\log_2\left(\frac{\mathbf{9}}{\mathbf{14}}\right) - \left(\frac{\mathbf{5}}{\mathbf{14}}\right)\log_2\left(\frac{\mathbf{5}}{\mathbf{14}}\right) = \mathbf{0.940} \qquad (\mathbf{3.2})$$

ID3(Ejemplos, Atributos destino, Atributos)

Ejemplos son los ejemplos de entrenamiento. Atributos destino es el atributo cuyo valor debe ser predicho por el árbol. Atributos es una lista de otros atributos que pueden ser probados por el aprendizaje del árbol de decisión. Devuelve un árbol de decisión que clasifica correctamente los ejemplos dados en Ejemplos.

- **Crea un nodo raíz para el árbol.**
- **Si todos los Ejemplos son positivos, Retorne un solo nodo raíz del árbol, con etiqueta = +.**
- **Si todos los Ejemplo son negativos, Retorne un solo nodo raíz del árbol, con etiqueta = -.**
- **Si Atributos es vacío, Retorne un solo nodo raíz del árbol, con etiquete = El valor más común de Atributos destino en Ejemplos.**
- **De lo contrario comenzar**
    - **A ← El atributo de Atributos que mejor* clasifica los Ejemplos.**
    - **La decisión para el atributo raíz ←A**
    - **Para cada posible valor, v_i, de A**
        - **Añadir una nueva rama de árbol debajo de raíz, correspondiente a la prueba A = v_i**
        - **Sea $Ejemplos_{v_i}$ ser el subconjunto de Ejemplos que tienen valor de v_i para A**
        - **Si $Ejemplos_{v_i}$ es vacío**
            - **A continuación, debajo de esta nueva rama, añada un nodo hoja con etiqueta = valor más común del Atributo de destino en Ejemplos**
            - **Otro agregue debajo de esta nueva rama el subárbol ID3($Ejemplos_{v_i}$, Atributo destino, Atributos – {A})**
- **Terminar**
- **Retornar raíz**

*** El mejor atributo es el que tiene mayor ganancia de información, como se define en la ecuación (3.4).**

Tabla 3.1

Resumen del algoritmo ID3 especializado para funciones con valores boléanos de aprendizaje. ID3 es un algoritmo voraz que crece el árbol de arriba hacia abajo, en cada nodo seleccionando el atributo que mejor clasifica los ejemplos locales de formación. Este proceso continúa hasta que el árbol perfectamente clasifica los ejemplos de entrenamiento, o hasta que se han utilizado todos los atributos.

Note que la entropía es **0** si todos los miembros de **S** pertenecen a una misma clase. Por ejemplo, si todos los miembros son positivos **($P_+ = 1$)**, entonces P_- es **0**, y $Entropía(S) = -1 * \log_2 1 - 0 * \log_2 0 = -1 * 0 - 0 * \log_2 0 = 0$. Note que la entropía es 1 cuando la colección contiene un igual número de ejemplos positivos y

negativos. Si la colección contiene desigual número de ejemplos positivos y negativos, la entropía está entre **0** y **1**. La figura 3.2 muestra la forma de la función de la entropía en relación a una clasificación booleana, como $\boldsymbol{P_+}$ varía entre **0** y **1**.

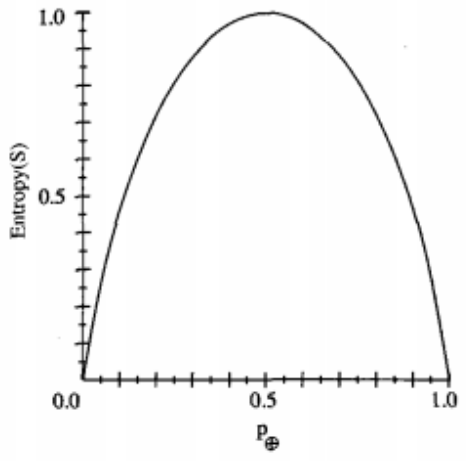

FIGURE 3.2
The entropy function relative to a boolean classification, as the proportion, $p_\oplus$, of positive examples varies between 0 and 1.

Una interpretación de la entropía de la teoría de la información es que específica el número mínimo de bits de información necesaria para codificar la clasificación de un miembro arbitrario de S (es decir, un miembro de S dibujado al azar con probabilidad uniforme). Por ejemplo, si $\boldsymbol{P_+}$ es **1**, el receptor sabe que el ejemplo dibujado será positivo, por lo que no necesita enviar ningún mensaje, y la entropía es cero. Por otro lado, si $\boldsymbol{P_+}$ es **0.5**, un bit es necesario para indicar si en el ejemplo dibujado es positivo o negativo. Si $\boldsymbol{P_+}$ es de **0.8**, entonces una colección de mensajes puede ser codificado utilizando en promedio menos de **1** bit por mensaje asignando códigos más cortos a las colecciones de ejemplos positivos y códigos más largos a ejemplos negativos menos probables.

Hasta ahora hemos hablado sobre entropía en el caso especial donde la clasificación de objetivo es booleana. Más generalmente, si el atributo de destino puede adoptar diferentes valores de **c**, entonces la entropía de **S** en relación con esto **c**-sabia clasificación se define como

$$\boldsymbol{Entropía(S)} = \sum_{i=1}^{c} -\boldsymbol{P_i} \log_2 \boldsymbol{P_i} \qquad (\mathbf{3.3})$$

Donde $\boldsymbol{P_i}$ es la proporción de **S** que pertenecen a la clase $\boldsymbol{i}$. Tenga en cuenta que el logaritmo es todavía base **2** porque la entropía es una medida de la duración prevista de codificación en bits. Tenga en cuenta también que si el atributo de destino puede adoptar valores posibles en **c**, la entropía puede ser tan grande como $\log_2 \boldsymbol{c}$.

3.4.1.2 GANANCIA DE LA INFORMACIÓN MIDE LA REDUCCIÓN ESPERADA DE LA ENTROPÍA

Teniendo en cuenta la entropía como una medida de la impureza en una colección de ejemplos de entrenamiento, ahora podemos definir una medida de la eficacia o efectividad de un atributo en la clasificación de los datos de entrenamiento. La medida que usaremos, llamada **ganancia de información**, es simplemente la reducción esperada de la entropía causada por la partición de los ejemplos de acuerdo con este atributo. Más precisamente, la ganancia de información, $\boldsymbol{Ganancia}(\boldsymbol{S}, \boldsymbol{A})$ de un atributo $\boldsymbol{A}$, con respecto a una colección de ejemplos $\boldsymbol{S}$, se define como

$$\boldsymbol{Ganancia}(\boldsymbol{S}, \boldsymbol{A}) = \boldsymbol{Entropia}(\boldsymbol{S}) - \sum_{\boldsymbol{v} \in \boldsymbol{Valores}(\boldsymbol{A})} \frac{|\boldsymbol{S_v}|}{|\boldsymbol{S}|} \boldsymbol{Entropia}(\boldsymbol{S_v}) \qquad (3.4)$$

Donde $\boldsymbol{Valores}(\boldsymbol{A})$ es el conjunto de todos los posibles valores para el atributo $\boldsymbol{A}$, y $\boldsymbol{S_v}$ es el subconjunto de $\boldsymbol{S}$ para el cual el atributo $\boldsymbol{A}$ tiene valor $\boldsymbol{v}$ (es decir, $\boldsymbol{S_v} = \{\boldsymbol{s} \in \boldsymbol{S} | \boldsymbol{A}(\boldsymbol{s}) = \boldsymbol{v}\}$). Obsérvese que el primer término de la ecuación (3.4) es sólo la entropía de la colección original $\boldsymbol{S}$, y el segundo término es el valor esperado de la entropía después de que $\boldsymbol{S}$ se divide utilizando el atributo $\boldsymbol{A}$. La entropía esperada descrita por este segundo término es simplemente la suma de las entropías de cada subconjunto $\boldsymbol{S_v}$, ponderada por la fracción de ejemplos $\frac{|\boldsymbol{S_v}|}{|\boldsymbol{S}|}$ que pertenecen a $\boldsymbol{S_v}$. El $\boldsymbol{Ganancia}(\boldsymbol{S}, \boldsymbol{A})$ es, por tanto, la reducción esperada de la entropía causada por conocer el valor del atributo $\boldsymbol{A}$. Dicho de otro modo, $\boldsymbol{Ganancia}(\boldsymbol{S}, \boldsymbol{A})$ es la información proporcionada sobre el valor de la función de destino, dado el valor de algún otro atributo $\boldsymbol{A}$. El valor de $\boldsymbol{Ganancia}(\boldsymbol{S}, \boldsymbol{A})$ es el número de bits guardados al codificar el valor objetivo de un miembro arbitrario de $\boldsymbol{S}$, conociendo el valor del atributo $\boldsymbol{A}$.

Por ejemplo, supongamos que $\boldsymbol{S}$ es una colección de días de entrenamiento-ejemplo descritos por atributos incluyendo Viento (Wind), que puede tener los valores Débil (Weak) o Fuerte (Strong). Como antes, supongamos que $\boldsymbol{S}$ es una colección que contiene **14** ejemplos $[\boldsymbol{9+}, \boldsymbol{5-}]$. De estos 14 ejemplos, suponga que **6** de los positivos y **2** de los ejemplos negativos tienen Viento (Wind) = Débil (Strong), y el resto tiene Viento = Fuerte. La ganancia de información debido a la clasificación de los **14** ejemplares originales por el atributo Viento (Wind) puede calcularse como

$$\boldsymbol{Valores}(\boldsymbol{Viento}\ (\boldsymbol{Wind})) = \boldsymbol{Débil}\ (\boldsymbol{Weak}), \boldsymbol{Fuerte}\ (\boldsymbol{Strong})$$

$$\boldsymbol{S} = [\boldsymbol{9+}, \boldsymbol{5-}]$$

$$\boldsymbol{S_{Weak}} = [\boldsymbol{6+}, \boldsymbol{2-}]$$

$$\boldsymbol{S_{Strong}} = [\boldsymbol{3+}, \boldsymbol{3-}]$$

$$\boldsymbol{Ganancia}(\boldsymbol{S}, \boldsymbol{Wind}) = \boldsymbol{Entropia}(\boldsymbol{S}) - \sum_{\boldsymbol{v} \in \boldsymbol{Valores}(\boldsymbol{A})} \frac{|\boldsymbol{S_v}|}{|\boldsymbol{S}|} \boldsymbol{Entropia}(\boldsymbol{S_v})$$

$$= \boldsymbol{Entropia}(\boldsymbol{S}) - \left(\frac{\boldsymbol{8}}{\boldsymbol{14}}\right) \boldsymbol{Entropia}(\boldsymbol{S_{Weak}}) - \left(\frac{\boldsymbol{6}}{\boldsymbol{14}}\right) \boldsymbol{Entropia}(\boldsymbol{S_{Strong}})$$

$$= \mathbf{0.940} - \left(\frac{\mathbf{8}}{\mathbf{14}}\right) * \mathbf{0.811} - \left(\frac{\mathbf{6}}{\mathbf{14}}\right) * \mathbf{1.00} = \mathbf{0.048}$$

La **ganancia de información** es precisamente la medida utilizada por **ID3** para seleccionar el mejor atributo en cada paso en el crecimiento del árbol. El uso de la ganancia de información para evaluar la relevancia de los atributos se resume en la Figura 3.3. En esta figura se calcula la ganancia de información de dos atributos diferentes, Humedad (Humidity) y Viento (wind), para determinar cuál es el mejor atributo para clasificar los ejemplos de entrenamiento mostrados en la Tabla 3.2.

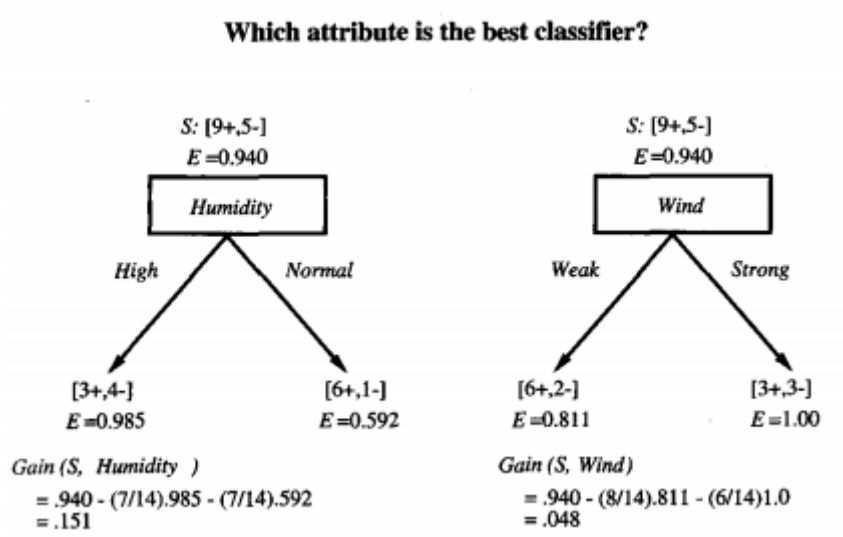

Figura 3.3

La humedad (Humidity) proporciona mayor ganancia de información que el viento (Wind), en relación con la clasificación objetivo. Aquí, $\boldsymbol{E}$ significa entropía y $\boldsymbol{S}$ para la colección original de ejemplos. Dada una colección inicial $\boldsymbol{S}$ de **9** positivos y **5** negativos ejemplos $[\mathbf{9}+, \mathbf{5}-]$, la clasificación de estos por su humedad (Humidity) produce colecciones de $[\mathbf{3}+, \mathbf{4}-]$ (Humedad = Alto)(Humidity = High) y $[6+, 1-]$ (Humedad = Normal)(Humidity = Normal). La información obtenida por este particionamiento es **0.151**, en comparación con una ganancia de sólo **0.048** para el atributo Viento (Wind).

3.4.2 **Un ejemplo ilustrativo**

Para ilustrar la operación de **ID3**, considere la tarea de aprendizaje representada por los ejemplos de entrenamiento de la Tabla 3.2. Aquí el atributo de destino **PlayTennis**, que puede tener valores de sí (yes) o no (no) para diferentes mañanas de los sábados, debe predecirse sobre la base de otros atributos de la mañana en cuestión.

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

TABLE 3.2
Training examples for the target concept *PlayTennis*.

Considere el primer paso a través del algoritmo, en el que se crea el nodo superior del árbol de decisión. **¿Qué atributo se debe probar primero en el árbol? ID3** determina la ganancia de información para cada atributo candidato (es decir, Outlook, Temperatura (Temperature), Humedad (Humidity) y Viento (Wind)), y luego selecciona el que tiene mayor ganancia de información. El cálculo de la ganancia de información para dos de estos atributos se muestra en la figura 3.3. Los valores de ganancia de información para los cuatro atributos son

$$\boldsymbol{Ganancia(S, Outlook) = 0.246}$$
$$\boldsymbol{Ganancia(S, Humidity) = 0.151}$$
$$\boldsymbol{Ganancia(S, Wind) = 0.048}$$
$$\boldsymbol{Ganancia(S, Temperature) = 0.029}$$

Donde $\boldsymbol{S}$ denota la colección de ejemplos de entrenamiento de la Tabla 3.2
De acuerdo con la medida de ganancia de información, el atributo **Outlook** proporciona la mejor predicción del atributo de destino, **PlayTennis**, sobre los ejemplos de entrenamiento. Por lo tanto, **Outlook** se selecciona como atributo de decisión para el nodo raíz y se crean ramas debajo de la raíz para cada uno de sus posibles valores (por ejemplo, **Sunny**, **Overcast** y **Rain**). El árbol de decisión parcial resultante se muestra en la Figura 3.4, junto con los ejemplos de entrenamiento ordenados para cada nuevo nodo descendente. Tenga en cuenta que cada ejemplo para el que **Outlook = Overcast** es también un ejemplo positivo de **PlayTennis**. Por lo tanto, este nodo del árbol se convierte en un nodo de hoja con la clasificación **PlayTennis = yes** (Sí). Por el contrario, los descendientes correspondientes a **Outlook = Sunny** y **Outlook = Rain** aún tienen entropía no nula, y el árbol de decisión se desarrollará más adelante debajo de estos nodos.

El proceso de selección de un nuevo atributo y particionamiento de los ejemplos de entrenamiento se repite ahora para cada nodo descendiente no descendiente, esta vez usando sólo los ejemplos de entrenamiento asociados con ese nodo. Los atributos que se han incorporado más arriba en el árbol se excluyen, por lo que cualquier atributo puede aparecer como máximo una vez a lo largo de cualquier camino a través del árbol. Este proceso continúa para cada nuevo nodo hoja hasta que se cumpla cualquiera de dos condiciones: (1) cada atributo ya ha sido incluido a lo largo de este camino a través del árbol, o (2) los ejemplos de entrenamiento asociados con este nodo hoja tienen el mismo atributo de destino Valor (es decir, su entropía es cero). La Figura 3.4 ilustra los cálculos de la ganancia de información para el siguiente paso en el crecimiento del árbol de decisión. El árbol de decisión final aprendido por **ID3** de los **14** ejemplos de entrenamiento de la Tabla 3.2 se muestra en la Figura 3.1.

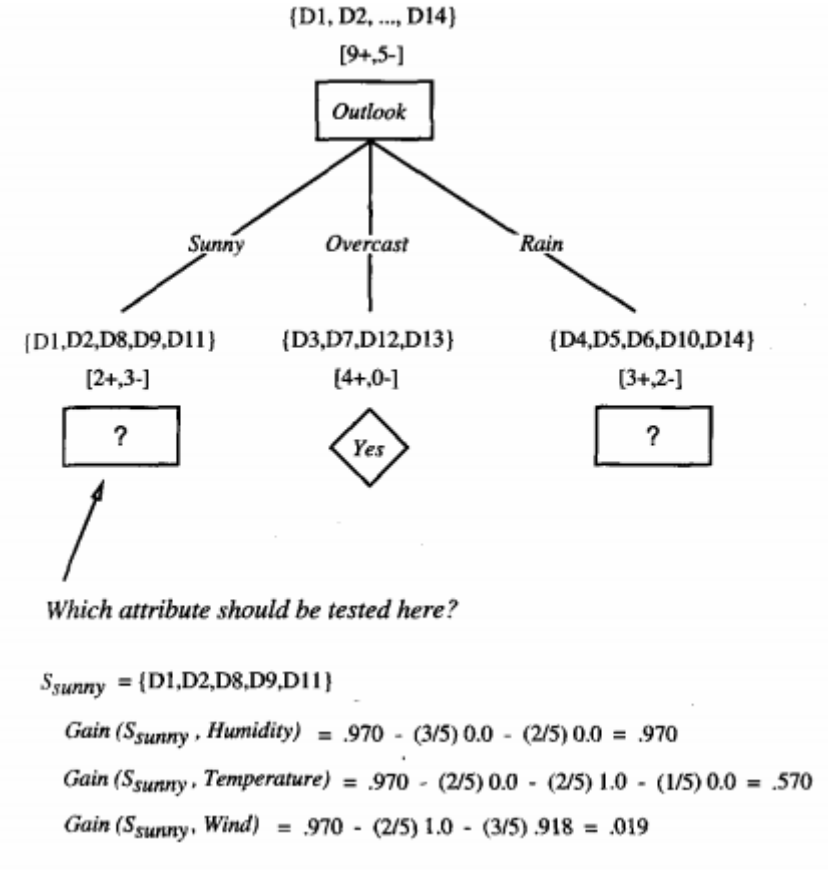

Figura 3.4

El árbol de decisiones parcialmente aprendido resultante del primer paso de **ID3**. Los ejemplos de entrenamiento se clasifican en los nodos descendientes correspondientes. El descendiente Nublado (**Overcast**) sólo tiene ejemplos positivos y por lo tanto se convierte en un nodo de hoja con clasificación (**Yes**) Sí. Los otros dos nodos se ampliarán más, seleccionando el atributo con mayor ganancia de información en relación con los nuevos subconjuntos de ejemplos.

3.5 HIPÓTESIS DE BÚSQUEDA ESPACIAL EN EL APRENDIZAJE DE UN ÁRBOL DE DECISIÓN

Al igual que con otros métodos de aprendizaje inductivo, **ID3** se puede caracterizar por buscar en un espacio de hipótesis uno que se ajuste a los ejemplos de capacitación. El espacio de hipótesis buscado por **ID3** es el conjunto de posibles árboles de decisión. **ID3** realiza una búsqueda de escalada simple a compleja a través de este espacio de hipótesis, comenzando con el árbol vacío, luego considerando hipótesis más elaboradas en busca de un árbol de decisión que clasifique correctamente los datos de entrenamiento.

La función de evaluación que guía esta búsqueda de alpinismo es la medida de **ganancia de información**. Esta búsqueda se muestra en la Figura 3.5.

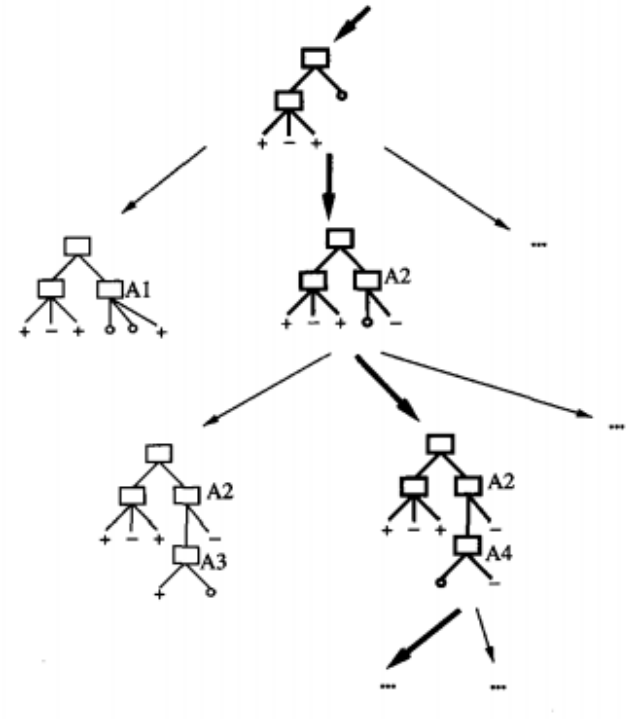

FIGURE 3.5
Hypothesis space search by ID3. ID3 searches through the space of possible decision trees from simplest to increasingly complex, guided by the *information gain heuristic.*

Al ver **ID3** en términos de su espacio de búsqueda y estrategia de búsqueda, podemos obtener una idea de sus capacidades y limitaciones.

El espacio de hipótesis de **ID3** de todos los árboles de decisión es un espacio completo de funciones finitas de valores discretos, relativas a los atributos disponibles. Debido a que cada función finita de valores discretos puede ser representada por algún árbol de decisión, **ID3** evita uno de los mayores riesgos de los métodos que buscan espacios de hipótesis incompletos (como los métodos que consideran solo hipótesis conjuntivas): que el espacio de hipótesis podría no contener la función objetivo.

ID3 solo mantiene una única hipótesis actual mientras busca en el espacio de los árboles de decisión. Esto contrasta, por ejemplo, con el método de la versión anterior candidato-eliminación método que mantiene el conjunto de todas las hipótesis consistentes con los ejemplos de entrenamiento disponibles. Mediante la determinación de sólo una sola hipótesis, **ID3** pierde las capacidades que se derivan explícitamente que representa todas las hipótesis consistentes. Por ejemplo, no tiene la capacidad para determinar cuántos

árboles de decisión alternativos son consistentes con los datos del entrenamiento disponibles, o plantear nuevas consultas de instancia que resuelven forma óptima entre estas hipótesis competidoras.

ID3 en su forma pura no realiza retroceso en su búsqueda. Una vez que selecciona un atributo para probar en un nivel particular en el árbol, nunca retrocede para reconsiderar esta opción. Por lo tanto, es susceptible a los riesgos habituales de búsqueda de escalada sin dar marcha atrás: la convergencia de las soluciones localmente óptimas que no son globalmente óptima. En el caso de **ID3**, una solución localmente óptima corresponde al árbol de decisión que selecciona a lo largo de la ruta de búsqueda única que explora. Sin embargo, esta solución localmente óptima puede ser menos deseable que los árboles que se habrían encontrado a lo largo de una rama diferente de la búsqueda. A continuación, analizamos una extensión que agrega una forma de retroceso (post-poda del árbol de decisión).

ID3 utiliza todos los ejemplos de capacitación en cada paso de la búsqueda para tomar decisiones basadas en estadísticas con respecto a cómo refinar su hipótesis actual. Esto contrasta con los métodos que toman decisiones de forma incremental, basado en ejemplos individuales de formación (por ejemplo, FIND-S o candidato-eliminación). Una ventaja de usar propiedades estadísticas de todos los ejemplos (por ejemplo, ganancia de información) es que la búsqueda resultante es mucho menos sensible a errores en ejemplos de entrenamiento individuales. **ID3** se puede extender fácilmente para manejar datos de entrenamiento ruidosos modificando su criterio de terminación para aceptar hipótesis que se ajusten imperfectamente a los datos de entrenamiento.

3.6 **SESGO INDUCTIVO EN EL APRENDIZAJE DE ÁRBOLES DE DECISIÓN**

¿Cuál es la política por la cual **ID3** se generaliza a partir de ejemplos de capacitación observados para clasificar instancias no vistas? En otras palabras, ¿cuál es su sesgo inductivo? Recuerde del Capítulo 2 que el sesgo inductivo es el conjunto de suposiciones que, junto con los datos de capacitación, deductivamente justifican las clasificaciones asignadas por el alumno a instancias futuras.

Dada una colección de ejemplos de entrenamiento, normalmente hay muchos árboles de decisión consistentes con estos ejemplos. Por lo tanto, describir el sesgo inductivo de **ID3** consiste en describir la base por la cual elige una de estas hipótesis consistentes sobre las demás. ¿Cuál de estos árboles de decisión elige **ID3**? Elige el primer árbol aceptable que encuentra en su búsqueda simple-a-compleja, colina arriba a través del espacio de árboles posibles. Hablando en términos generales, entonces, la estrategia de búsqueda **ID3** (a) selecciona a favor de árboles más cortos sobre los más largos, y (b) selecciona árboles que colocan los atributos con mayor ganancia de información más cercana a la raíz. Debido a la interacción sutil entre la heurística de selección de atributos utilizada por **ID3** y los ejemplos de entrenamiento particulares que encuentra, es difícil caracterizar precisamente el sesgo inductivo exhibido por **ID3**. Sin embargo, podemos caracterizar de manera aproximada su sesgo como preferencia por árboles de decisión corta sobre árboles complejos.

Aproximación inductiva aproximada de ID3: árboles más cortos son preferidos sobre árboles más grandes.

De hecho, uno podría imaginar un algoritmo similar a **ID3** que exhibe precisamente este sesgo inductivo. Considere un algoritmo que comience con el árbol vacío y busque amplitud primero mediante árboles progresivamente más complejos, primero considerando todos los árboles de profundidad 1, luego todos los árboles de profundidad 2, etc. Una vez que encuentra un árbol de decisión consistente con los datos de entrenamiento, devuelve el árbol más pequeño y consistente en esa profundidad de búsqueda (p. ej., el árbol con el menor número de nodos). Llamemos a este algoritmo de búsqueda de amplitud BFS-ID3. BFS-ID3 encuentra un árbol de decisión más corto y por lo tanto exhibe precisamente el sesgo "se prefieren árboles más cortos sobre árboles más largos". **ID3** se puede ver como una aproximación eficiente a BFS-ID3, utilizando una búsqueda heurística codiciosa para intentar encontrar el árbol más corto sin realizar toda la búsqueda de amplitud a través del espacio de hipótesis.
Debido a que **ID3** usa la heurística de ganancia de información y una estrategia de escalada, exhibe un sesgo más complejo que BFS-ID3. En particular, no siempre encuentra el árbol coherente más corto, y está predispuesto para favorecer a los árboles que colocan atributos con alta ganancia de información más cercana a la raíz.

Una aproximación más cercana al sesgo inductivo de ID3: árboles más cortos son preferidos sobre árboles más largos. Los árboles que colocan atributos de alta ganancia de información cerca de la raíz son preferibles a aquellos que no lo hacen.

3.6.1 Sesgos de restricción y sesgos de preferencia

Hay una diferencia interesante entre los tipos de sesgo inductivo exhibidos por **ID3** y por el algoritmo CANDIDATE-ELIMINATION discutido en el Capítulo 2.
Considere la diferencia entre la búsqueda espacial de hipótesis en estos dos enfoques:

- **ID3** busca un espacio de hipótesis completo (es decir, uno capaz de expresar cualquier función finita de valores discretos). Busca de forma incompleta a través de este espacio, desde hipótesis simples hasta complejas, hasta que se cumple su condición de terminación (por ejemplo, hasta que encuentra una hipótesis coherente con los datos). Su sesgo inductivo es únicamente una consecuencia del ordenamiento de hipótesis por su estrategia de búsqueda. Su espacio de hipótesis no introduce ningún sesgo adicional.
- El algoritmo CANDIDATE-ELIMINATION del espacio de versión busca un espacio de hipótesis incompleto (es decir, uno que puede expresar solo un subconjunto de los conceptos potencialmente enseñables). Busca en este espacio por completo, encontrando todas las hipótesis consistentes con los datos de entrenamiento. Su sesgo inductivo es únicamente una consecuencia del poder expresivo de su representación de hipótesis. Su estrategia de búsqueda no introduce ningún sesgo adicional.

En resumen, el sesgo inductivo de **ID3** sigue de su estrategia de búsqueda, mientras que el sesgo inductivo del algoritmo CANDIDATE-ELIMINATION se sigue de la definición de su espacio de búsqueda.

El sesgo inductivo de **ID3** es por lo tanto una preferencia por ciertas hipótesis sobre otras (por ejemplo, para hipótesis cortas), sin restricción estricta en las hipótesis que pueden enumerarse finalmente. Esta forma de sesgo típicamente se denomina sesgo de preferencia (o, alternativamente, un sesgo de búsqueda). En contraste, el sesgo del algoritmo CANDIDATEELIMINATION tiene la forma de una restricción categórica sobre el conjunto de hipótesis consideradas. Esta forma de sesgo se denomina típicamente un sesgo de restricción (o, alternativamente, un sesgo de idioma).

Dado que se requiere alguna forma de sesgo inductivo para generalizar más allá de los datos de entrenamiento (ver Capítulo 2), qué tipo de sesgo inductivo preferimos; ¿un sesgo de preferencia o un sesgo de restricción?

Típicamente, un sesgo de preferencia es más deseable que un sesgo de restricción, ya que permite al alumno trabajar dentro de un espacio de hipótesis completo que se asegura que contiene la función objetivo desconocida. En contraste, un sesgo de restricción que limita estrictamente el conjunto de hipótesis potenciales es generalmente menos deseable, porque introduce la posibilidad de excluir por completo la función objetivo desconocida.

Mientras que **ID3** exhibe un sesgo puramente de preferencia y CANDIDATE-ELIMINATIO un sesgo puramente de restricción, algunos sistemas de aprendizaje combinan ambos. Considere, por ejemplo, el programa descrito en el Capítulo 1 para aprender una función de evaluación numérica para el juego. En este caso, la función de evaluación aprendida está representada por una combinación lineal de un conjunto fijo de características de la placa, y el algoritmo de aprendizaje ajusta los parámetros de esta combinación lineal para ajustarse mejor a los datos de entrenamiento disponibles. En este caso, la decisión de usar una función lineal para representar la función de evaluación introduce un sesgo de restricción (las funciones de evaluación no lineal no se pueden representar de esta forma). Al mismo tiempo, la elección de un método de ajuste de parámetro particular (el algoritmo LMS en este caso) introduce un sesgo de preferencia que surge de la búsqueda ordenada a través del espacio de todos los valores de parámetro posibles.

3.6.2 **¿Por qué preferir hipótesis cortas?**

¿Es el sesgo inductivo de **ID3** que favorece a los árboles de decisión más cortos una base sólida para generalizar más allá de los datos de entrenamiento? Los filósofos y otros han debatido esta cuestión durante siglos, y el debate permanece sin resolver hasta el día de hoy. William of Occam fue uno de los primeros en discutir la cuestión, alrededor del año 1320, por lo que este sesgo a menudo se conoce con el nombre de la navaja de afeitar de Occam.

La navaja de Occam: Prefiera la hipótesis más simple que se ajuste a los datos.

Por supuesto, dar un nombre a las gemas inductivas no lo justifica. ¿Por qué debería uno preferir hipótesis más simples? Observe que los científicos a veces parecen seguir este sesgo inductivo. Los físicos, por ejemplo, prefieren explicaciones simples para los movimientos de los planetas, en lugar de explicaciones más complejas. ¿Por qué? Un argumento es que debido a que hay menos hipótesis cortas que largas (basadas en argumentos combinatorios directos), es menos probable que se encuentre una hipótesis breve que casualmente coincida con los datos de entrenamiento. En contraste, a menudo hay muchas hipótesis muy complejas que se ajustan a los datos de entrenamiento actuales, pero no se pueden generalizar correctamente a los datos posteriores. Considere las hipótesis del árbol de decisión, por ejemplo. Hay muchos más árboles de decisión de 500 nodos que árboles de decisión de 5 nodos. Dado un pequeño conjunto de 20 ejemplos de entrenamiento, podríamos esperar encontrar muchos árboles de decisión de 500 nodos de acuerdo con estos, mientras que estaríamos más sorprendidos si un árbol de decisión de 5 nodos pudiera encajar perfectamente con estos datos. Por lo tanto, podríamos creer que el árbol de 5 nodos es menos probable que sea una coincidencia estadística y preferimos esta hipótesis sobre la hipótesis de 500 nodos.

Tras un examen más detallado, resulta que hay una gran dificultad con el argumento anterior. Con el mismo razonamiento podríamos haber argumentado que uno debería preferir los árboles de decisión que contienen exactamente 17 nodos de hoja con 11 nodos de hoja, que usan el atributo de decisión A1 en la raíz y los atributos de prueba A2 a través de Todos, en orden numérico. Hay relativamente pocos árboles de este tipo, y podríamos argumentar (con el mismo razonamiento anterior) que nuestra posibilidad a priori de encontrar uno consistente con un conjunto arbitrario de datos es, por lo tanto, pequeño. La dificultad aquí es que hay muchos pequeños conjuntos de hipótesis que uno puede definir, la mayoría bastante arcanas. ¿Por qué deberíamos creer que el pequeño conjunto de hipótesis que consiste en árboles de decisión con descripciones cortas debería ser más relevante que la multitud de otros pequeños conjuntos de hipótesis que podríamos definir?

Un segundo problema con el argumento anterior para la navaja de Occam es que el tamaño de una hipótesis está determinado por la representación particular utilizada internamente por el alumno. ¡Dos aprendices que usan diferentes representaciones internas podrían por lo tanto anive en diferentes hipótesis, ambos justificando sus conclusiones contradictorias por la navaja de Occam! Por ejemplo, la función representada por el árbol de decisión aprendido en la figura 3.1 podría representarse como un árbol con solo un nodo de decisión, por un alumno que usa el atributo booleano XYZ, donde definimos el atributo XYZ como verdadero para las instancias que están clasificadas positivo por el árbol de decisión en la figura 3.1 y falso de lo contrario. Por lo tanto, dos estudiantes, ambos aplicando la navaja de Occam, se generalizarían de diferentes maneras si uno usara el atributo XYZ para describir sus ejemplos y el otro solo usara los atributos Outlook, Temperatura, Humedad y Viento.

Este último argumento muestra que la navaja de Occam producirá dos hipótesis diferentes a partir de los mismos ejemplos de entrenamiento cuando es aplicada por dos estudiantes que perciben estos ejemplos en términos de diferentes representaciones internas. Sobre esta base, podríamos sentirnos tentados a rechazar la navaja de Occam

por completo. Sin embargo, considere la siguiente situación que examina la cuestión de qué representaciones internas pueden surgir de un proceso de evolución y selección natural. Imagínese una población de agentes de aprendizaje artificial creados por un proceso evolutivo simulado que implica reproducción, mutación y selección natural de estos agentes. Supongamos que este proceso evolutivo puede alterar los sistemas perceptivos de estos agentes de generación en generación, cambiando así los atributos internos por los cuales perciben su mundo. En aras de la argumentación, supongamos también que los agentes de aprendizaje emplean un algoritmo de aprendizaje fijo (digamos ID3) que no puede ser alterado por la evolución. Es razonable suponer que con el tiempo la evolución producirá una representación interna que hará que estos agentes sean cada vez más exitosos dentro de su entorno. Suponiendo que el éxito de un agente depende en gran medida de su capacidad para generalizar con precisión, por lo tanto, esperamos que la evolución desarrolle representaciones internas que funcionen bien con cualquier algoritmo de aprendizaje y sesgo inductivo presente. Si la especie de agentes emplea un algoritmo de aprendizaje cuyo sesgo inductivo es la navaja de Occam, entonces esperamos que la evolución produzca representaciones internas para las cuales la navaja de Occam es una estrategia exitosa. La esencia del argumento aquí es que la evolución creará representaciones internas que hacen que el sesgo inductivo del algoritmo de aprendizaje sea una profecía autocumplida, simplemente porque puede alterar la representación más fácilmente de lo que puede alterar el algoritmo de aprendizaje.

Por ahora, dejamos el debate sobre la navaja de Occam. Lo revisaremos en el Capítulo 6, donde discutiremos el principio de Longitud mínima de la descripción, una versión de la navaja de Occam que puede interpretarse dentro de un marco bayesiano.

3.7 PROBLEMAS DE APRENDIZAJE DE ÁRBOLES DE DECISIÓN

Los aspectos prácticos en el aprendizaje de los árboles de decisión incluyen determinar cómo crecer profundamente el árbol de decisiones, manejar atributos continuos, elegir una medida de selección de atributos apropiada, transferir datos de entrenamiento con valores de atributo faltantes, manejar atributos con costos diferentes y mejorar la eficiencia computacional. A continuación, analizamos cada uno de estos problemas y extensiones del algoritmo **ID3** básico que los aborda. **ID3** se ha extendido para abordar la mayoría de estos problemas, y el sistema resultante se renombró **C4.5** (Quinlan 1993).

3.7.1 Evitando sobreajustar los datos

El algoritmo descrito en la Tabla 3.1 crece cada rama del árbol con la profundidad suficiente para clasificar perfectamente los ejemplos de entrenamiento. Aunque a veces esto es una estrategia razonable, de hecho, puede provocar dificultades cuando hay ruido en los datos, o cuando el número de ejemplos de entrenamiento es demasiado pequeño para producir una muestra representativa de la verdadera función objetivo. En

cualquiera de estos casos, este algoritmo simple puede producir árboles que sobrepasan los ejemplos de entrenamiento.
Diremos que una hipótesis sobrepasa los ejemplos de capacitación si otras hipótesis que se ajustan peor a los ejemplos de capacitación realmente funcionan mejor en toda la distribución de las instancias (es decir, incluidas las instancias más allá del conjunto de capacitación).

Definición: Dado un espacio de hipótesis $\boldsymbol{H}$, se dice que una hipótesis $\boldsymbol{h} \in \boldsymbol{H}$ sobreajusta los datos de entrenamiento si existe alguna hipótesis alternativa $\boldsymbol{h'} \in \boldsymbol{H}$, tal que $\boldsymbol{h}$ tiene un error menor que $\boldsymbol{h'}$ sobre los ejemplos de entrenamiento, pero $\boldsymbol{h'}$ tiene un error menor que $\boldsymbol{h}$ sobre toda la distribución de instancias.

La Figura 3.6 ilustra el impacto del sobreajuste en una aplicación típica del aprendizaje del árbol de decisión. En este caso, el algoritmo **ID3** se aplica a la tarea de aprender qué pacientes médicos tienen una forma de diabetes. El eje horizontal de este gráfico indica el número total de nodos en el árbol de decisión, a medida que se construye el árbol. El eje vertical indica la precisión de las predicciones hechas por el árbol. La línea continua muestra la precisión del árbol de decisión sobre los ejemplos de entrenamiento, mientras que la línea discontinua muestra la precisión medida sobre un conjunto independiente de ejemplos de prueba (no incluidos en el conjunto de entrenamiento). Previsiblemente, la precisión del árbol sobre los ejemplos de entrenamiento aumenta monótonamente a medida que crece el árbol. Sin embargo, la precisión medida sobre los ejemplos de prueba independientes primero aumenta, luego disminuye. Como se puede ver, una vez que el tamaño del árbol excede aproximadamente 25 nodos, la elaboración adicional del árbol disminuye su precisión sobre los ejemplos de prueba a pesar de aumentar su precisión en los ejemplos de entrenamiento.

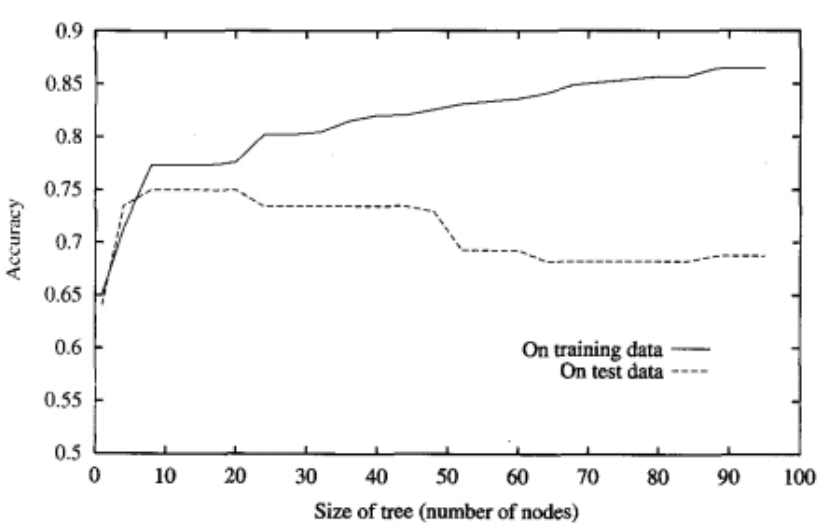

FIGURE 3.6
Overfitting in decision tree learning. As ID3 adds new nodes to grow the decision tree, the accuracy of the tree measured over the training examples increases monotonically. However, when measured over a set of test examples independent of the training examples, accuracy first increases, then decreases. Software and data for experimenting with variations on this plot are available on the World Wide Web at http://www.cs.cmu.edu/~tom/mlbook.html.

¿Cómo puede ser posible que el árbol $\boldsymbol{h}$ adapte los ejemplos de entrenamiento mejor que $\boldsymbol{h'}$, pero que funcione peor en los ejemplos siguientes? Una forma en que esto puede ocurrir es cuando los ejemplos de entrenamiento contienen errores aleatorios o ruido. Para ilustrar, considere el efecto de agregar el siguiente ejemplo de entrenamiento positivo, incorrectamente etiquetado como negativo, a los ejemplos (de lo contrario correctos) en la Tabla 3.2.

(Outlook = Sunny, Temperature = Hot, Humidity = Normal,
Wind = Strong, PlayTennis = No)

Dados los datos originales sin error, **ID3** produce el árbol de decisión que se muestra en la Figura 3.1. Sin embargo, la adición de este ejemplo incorrecto ahora causará que **ID3** construya un árbol más complejo. En particular, el nuevo ejemplo se ordenará en el segundo nodo de hoja desde la izquierda en el árbol aprendido de la Figura 3.1, junto con los ejemplos positivos anteriores **D9** y **D11**. Debido a que el nuevo ejemplo está etiquetado como un ejemplo negativo, **ID3** buscará para más refinamientos al árbol debajo de este nodo. Por supuesto, siempre que el nuevo ejemplo erróneo difiera de alguna manera arbitraria de los otros ejemplos afiliados a este nodo, **ID3** logrará encontrar un nuevo atributo de decisión para separar este nuevo ejemplo de los dos ejemplos positivos anteriores en este nodo de árbol. El resultado es que **ID3** generará un árbol de decisión ($\boldsymbol{h}$) que es más complejo que el árbol original de la figura 3.1 ($\boldsymbol{h'}$). Por supuesto, $\boldsymbol{h}$ encajará perfectamente en la colección de ejemplos de entrenamiento, mientras que el $\boldsymbol{h}$ más simple no lo hará. Sin embargo, dado que el nuevo nodo de decisión es simplemente una consecuencia de ajustar el ejemplo de entrenamiento

ruidoso, esperamos que h supere a h' en los datos posteriores extraídos de la misma distribución de instancia.

El ejemplo anterior ilustra cómo el ruido aleatorio en los ejemplos de entrenamiento puede llevar a un ajuste excesivo. De hecho, el sobreajuste es posible incluso cuando los datos de entrenamiento están libres de ruido, especialmente cuando se asocian pequeños números de ejemplos con los nodos de las hojas. En este caso, es bastante posible que ocurran regularidades coincidentes, en las que algunos atributos dividen muy bien los ejemplos, a pesar de no estar relacionados con la función objetivo real. Siempre que tales regularidades coincidentes existan, existe el riesgo de sobreajuste.

El sobreajuste es una dificultad práctica importante para el aprendizaje del árbol de decisión y muchos otros métodos de aprendizaje. Por ejemplo, en un estudio experimental de **ID3** que involucró cinco tareas de aprendizaje diferentes con datos no deterministas y ruidosos (Mingers 1989b), se descubrió que el sobreajuste disminuía la precisión de los árboles de decisión aprendidos en un 10-25% en la mayoría de los problemas.

Hay varios enfoques para evitar el sobreajuste en el aprendizaje del árbol de decisión. Estos se pueden agrupar en dos clases:

- enfoques que detienen el crecimiento del árbol más temprano, antes de que llegue al punto en que clasifique perfectamente los datos de entrenamiento,
- enfoques que permiten que el árbol sobreajuste los datos, y luego posponer el árbol.

Aunque el primero de estos enfoques podría parecer más directo, se ha encontrado que el segundo enfoque de los árboles de superposición posteriores a la poda es más exitoso en la práctica. Esto se debe a la dificultad en el primer enfoque de estimar con precisión cuándo dejar de crecer el árbol.

Independientemente de si se encuentra el tamaño correcto del árbol al detenerse temprano o mediante una poda posterior, una pregunta clave es qué criterio se debe usar para determinar el tamaño final correcto del árbol. Los enfoques incluyen:

- Use un conjunto separado de ejemplos, distinto de los ejemplos de capacitación, para evaluar la utilidad de los nodos posteriores a la poda del árbol.
- Utilice todos los datos disponibles para el entrenamiento, pero aplique una prueba estadística para estimar si es probable que la expansión (o poda) de un nodo particular produzca una mejora más allá del conjunto de entrenamiento. Por ejemplo, Quinlan (1986) utiliza una prueba de chi-cuadrado para estimar si es probable que expandir un nodo mejore el rendimiento en toda la distribución de la instancia, o solo en la muestra actual de datos de entrenamiento.
- Utilice una medida explícita de la complejidad para codificar los ejemplos de capacitación y el árbol de decisiones, deteniendo el crecimiento del árbol cuando se minimiza este tamaño de codificación. Este enfoque, basado en una heurística denominada principio de Longitud mínima de la descripción, se analiza con más detalle en el Capítulo 6, así como en Quinlan y Rivest (1989) y Mehta et al. (199.5).

El primero de los enfoques anteriores es el más común y a menudo se lo conoce como un enfoque conjunto de entrenamiento y validación. Discutimos las dos variantes principales de este enfoque a continuación. En este enfoque, los datos disponibles se separan en dos conjuntos de ejemplos: un conjunto de entrenamiento, que se utiliza para formar la hipótesis aprendida, y un conjunto de validación por separado, que se utiliza para evaluar la precisión de esta hipótesis sobre datos posteriores y, en particular, para evaluar el impacto de podar esta hipótesis. La motivación es esta: aunque el alumno puede ser engañado por errores aleatorios y regularidades coincidentes dentro del conjunto de entrenamiento, es poco probable que el conjunto de validación muestre las mismas fluctuaciones aleatorias. Por lo tanto, se puede esperar que el conjunto de validación proporcione una verificación de seguridad contra el sobreajuste de las características espurias del conjunto de entrenamiento. Por supuesto, es importante que el conjunto de validación sea lo suficientemente grande como para proporcionar una muestra estadísticamente significativa de las instancias. Una heurística común es retener un tercio de los ejemplos disponibles para el conjunto de validación, utilizando los otros dos tercios para el entrenamiento.

3.7.1.1 **PODA DE REDUCCIÓN DEL ERROR**

¿Cómo podemos usar un conjunto de validación para evitar el sobreajuste? Un enfoque, llamado reducción de errores reducidos (Quinlan 1987), es considerar cada uno de los nodos de decisión en el árbol como candidatos para la poda. Podar un nodo de decisión consiste en eliminar el subárbol enraizado en ese nodo, convirtiéndolo en un nodo hoja y asignarle la clasificación más común de los ejemplos de formación afiliados con ese nodo. Los nodos se eliminan solo si el árbol podado resultante no funciona peor que el original sobre el conjunto de validación. Esto tiene el efecto de que cualquier nodo de hoja agregado debido a regularidades coincidentes en el conjunto de entrenamiento es probable que se reduzca porque es poco probable que estas coincidencias ocurran en el conjunto de validación. Los nodos se podan de forma iterativa, siempre eligiendo el nodo cuya eliminación aumenta la precisión del árbol de decisión sobre el conjunto de validación. La poda de los nodos continúa hasta que la poda adicional sea dañina (es decir, disminuye la precisión del árbol sobre el conjunto de validación).

El impacto de la reducción de errores en la precisión del árbol de decisión se ilustra en la Figura 3.7. Como en la Figura 3.6, la precisión del árbol se muestra medida tanto en los ejemplos de entrenamiento como en los ejemplos de prueba. La línea adicional en la Figura 3.7 muestra la precisión sobre los ejemplos de prueba a medida que el árbol es podado. Cuando comienza la poda, el árbol está en su tamaño máximo y con la precisión más baja sobre el conjunto de prueba. A medida que avanza la poda, se reduce el número de nodos y aumenta la precisión sobre el conjunto de prueba. Aquí, los datos disponibles se han dividido en tres subconjuntos: los ejemplos de capacitación, los ejemplos de validación utilizados para podar el árbol y un conjunto de ejemplos de prueba utilizados para proporcionar una estimación imparcial de la precisión sobre los ejemplos futuros no vistos. La gráfica muestra precisión sobre los conjuntos de

entrenamiento y prueba. La precisión sobre el conjunto de validación utilizado para la poda no se muestra.

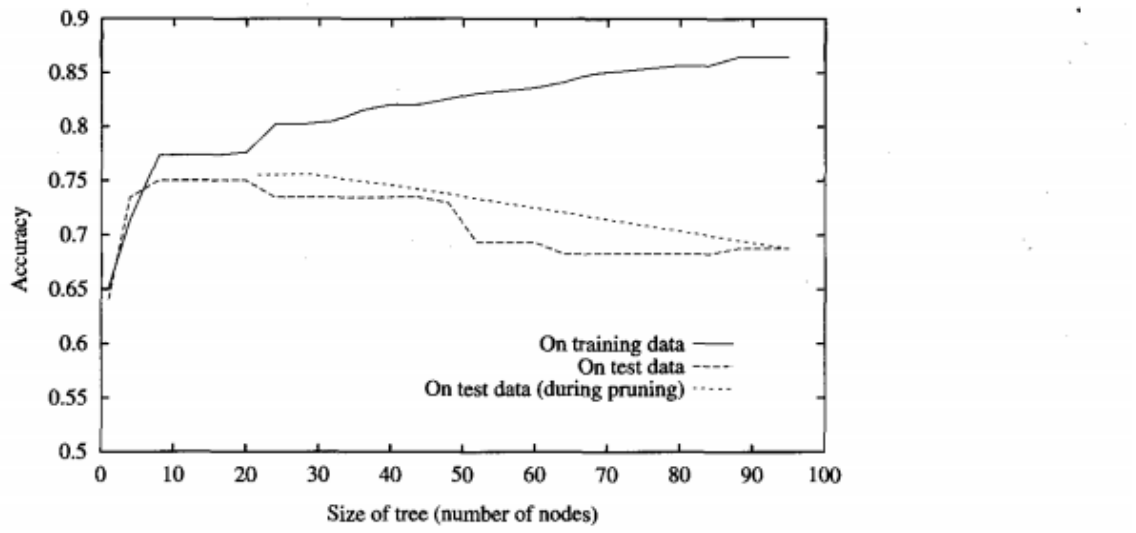

FIGURE 3.7
Effect of reduced-error pruning in decision tree learning. This plot shows the same curves of training and test set accuracy as in Figure 3.6. In addition, it shows the impact of reduced error pruning of the tree produced by ID3. Notice the increase in accuracy over the test set as nodes are pruned from the tree. Here, the validation set used for pruning is distinct from both the training and test sets.

Usar un conjunto separado de datos para guiar la poda es un enfoque efectivo siempre que haya una gran cantidad de datos disponibles. El principal inconveniente de este enfoque es que cuando los datos son limitados, retener parte de ellos para el conjunto de validación reduce aún más el número de ejemplos disponibles para la capacitación. La siguiente sección presenta un enfoque alternativo a la poda que se ha encontrado útil en muchas situaciones prácticas donde los datos son limitados. También se han propuesto muchas técnicas adicionales, que implican dividir los datos disponibles en diferentes momentos de múltiples maneras, y luego promediar los resultados. Las evaluaciones empíricas de métodos alternativos de poda de árboles son informadas por Mingers (1989b) y por Malerba et al. (1995).

3.7.1.2 REGLA POST-PODA

En la práctica, un método bastante exitoso para encontrar hipótesis de alta precisión es una técnica que llamaremos regla posterior a la poda. Una variante de este método de poda es utilizada por **C4.5** (Quinlan 1993), que es una mejora del algoritmo original **ID3**. La regla posterior a la poda implica los siguientes pasos:

1. Inferir el árbol de decisión del conjunto de entrenamiento, cultivar el árbol hasta que los datos de entrenamiento se ajusten lo mejor posible y permitiendo que ocurra un ajuste excesivo.
2. Convierta el árbol aprendido en un conjunto equivalente de reglas creando una regla para cada ruta desde el nodo raíz a un nodo hoja.
3. Pode (generalice) cada regla eliminando cualquier precondición que resulte en la mejora de su precisión estimada.

4. Ordene las reglas podadas por su precisión estimada, y considérelas en esta secuencia cuando clasifique instancias posteriores.

Para ilustrar, considere nuevamente el árbol de decisiones en la Figura 3.1. En la regla posterior a la poda, se genera una regla para cada nodo hoja en el árbol. Cada prueba de atributo a lo largo de la ruta desde la raíz hasta la hoja se convierte en un antecedente de regla (precondición) y la clasificación en el nodo hoja se convierte en la regla consecuente (condición posterior). Por ejemplo, la ruta más a la izquierda del árbol en la Figura 3.1 se traduce en la regla

IF (Outlook = Sunny) ∧ (Humidity = High)
THEN PlayTennis = No

A continuación, cada regla se elimina eliminando cualquier antecedente o precondición, cuya eliminación no empeora su precisión estimada. Dada la regla anterior, por ejemplo, la regla posterior a la poda consideraría eliminar las condiciones previas (Outlook = Sunny (Soleado)) y (Humidity (Humedad) = High (Alto)). Seleccionaría cualquiera de estos pasos de poda que produjera la mayor mejora en la precisión estimada de la regla, luego consideraría podar la segunda condición previa como un paso de poda adicional. No se realiza ningún paso de poda si reduce la precisión estimada de la regla.
Como se señaló anteriormente, un método para estimar la precisión de las reglas es usar un conjunto de ejemplos de validación disjuntos del conjunto de entrenamiento. Otro método, utilizado por C4.5, es evaluar el rendimiento basado en el propio conjunto de entrenamiento, utilizando una estimación pesimista para compensar el hecho de que los datos de entrenamiento dan una estimación sesgada a favor de las reglas. Más precisamente, C4.5 calcula su estimación pesimista calculando la precisión de la regla sobre los ejemplos de entrenamiento a los que se aplica, luego calcula la desviación estándar en esta precisión estimada suponiendo una distribución binomial. Para un nivel de confianza dado, la estimación del límite inferior se toma como la medida del rendimiento de la regla (por ejemplo, para un intervalo de confianza del 95%, la precisión de la regla se estima pesimistamente por la precisión observada sobre el conjunto de entrenamiento, menos 1,96 veces el estándar estimado desviación). El efecto neto es que, para grandes conjuntos de datos, la estimación pesimista es muy cercana a la precisión observada (por ejemplo, la desviación estándar es muy pequeña), mientras que crece más allá de la precisión observada a medida que disminuye el tamaño del conjunto de datos. Aunque este método heurístico no es estadísticamente válido, se ha encontrado útil en la práctica. Ver el Capítulo 5 para una discusión de enfoques estadísticamente válidos para estimar los medios y los intervalos de confianza.

¿Por qué convertir el árbol de decisión a reglas antes de podar? Hay tres ventajas principales.

- La conversión a reglas permite distinguir entre los diferentes contextos en los que se usa un nodo de decisión. Debido a que cada ruta distinta a través del nodo

del árbol de decisión produce una regla distinta, la decisión de poda con respecto a esa prueba de atributo se puede hacer de manera diferente para cada ruta. Por el contrario, si el árbol en sí se podara, las únicas dos opciones serían eliminar por completo el nodo de decisión o retenerlo en su forma original.

- La conversión a reglas elimina la distinción entre pruebas de atributos que ocurren cerca de la raíz del árbol y aquellas que ocurren cerca de las hojas. Por lo tanto, evitamos problemas desordenados de contabilidad como, por ejemplo, cómo reorganizar el árbol si el nodo raíz se elimina mientras se conserva parte del subárbol debajo de esta prueba.
- La conversión a reglas mejora la legibilidad. Las reglas a menudo son más fáciles de entender.

3.7.2 Incorporación de atributos de valor continúo

Nuestra definición inicial de **ID3** está restringida a los atributos que toman un conjunto discreto de valores. En primer lugar, el atributo objetivo cuyo valor es predicho por el árbol aprendido debe tener un valor discreto. En segundo lugar, los atributos probados en los nodos de decisión del árbol también deben tener un valor discreto. Esta segunda restricción se puede eliminar fácilmente para que los atributos de decisión de valor continuo puedan incorporarse en el árbol aprendido. Esto se puede lograr definiendo dinámicamente nuevos atributos discretos que particionen el valor de atributo continuo en un conjunto discreto de intervalos. En particular, para un atributo A que es de valor continuo, el algoritmo puede crear dinámicamente un nuevo atributo booleano $\boldsymbol{A_c}$ que es verdadero si $\boldsymbol{A} < \boldsymbol{c}$ y falso en caso contrario. La única pregunta es cómo seleccionar el mejor valor para el umbral $\boldsymbol{c}$.

Como ejemplo, supongamos que deseamos incluir el atributo de valor continuo Temperatura al describir los días de ejemplo de entrenamiento en la tarea de aprendizaje de la Tabla 3.2. Supongamos además que los ejemplos de entrenamiento asociados con un nodo particular en el árbol de decisión tienen los siguientes valores para Temperatura y el atributo objetivo PlayTennis.

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

¿Qué atributo booleano basado en el umbral debe definirse en función de la temperatura? Claramente, nos gustaría elegir un umbral $\boldsymbol{c}$, que produzca la mayor ganancia de información. Al ordenar los ejemplos según el atributo continuo $\boldsymbol{A}$, identificando ejemplos adyacentes que difieren en su clasificación de objetivos, podemos generar un conjunto de umbrales candidatos a medio camino entre los valores correspondientes de $\boldsymbol{A}$. Se puede demostrar que el valor de $\boldsymbol{c}$ que **maximiza la ganancia de información** siempre debe encontrarse en dicho límite (Fayyad 1991). Estos umbrales candidatos pueden evaluarse computando la **ganancia de información** asociada con cada uno. En el ejemplo actual, hay dos umbrales candidatos, que corresponden a los valores de temperatura a los cuales cambia el valor de PlayTennis: (48 + 60) / 2 y (80 + 90) / 2. La **ganancia de información** puede entonces calcularse

para cada uno de los atributos candidatos, Temperatura > 54 y Temperatura > 85, y se puede seleccionar lo mejor (Temperatura > 54). Este atributo booleano creado dinámicamente puede competir con los otros atributos candidatos de valores discretos disponibles para hacer crecer el árbol de decisiones. Fayyad e Irani (1993) discuten una extensión de este enfoque que divide el atributo continuo en múltiples intervalos en lugar de solo dos intervalos basados en un único umbral. Utgoff y Brodley (1991) y Murthy et al. (1994) discuten los enfoques que definen las características mediante el umbral de combinaciones lineales de varios atributos de valor continuo.

3.7.3 Medidas alternativas para seleccionar atributos

Existe un sesgo natural en la medida de ganancia de información que favorece los atributos con muchos valores sobre aquellos con pocos valores. Como ejemplo extremo, considere el atributo Fecha, que tiene una gran cantidad de valores posibles (por ejemplo, 4 de marzo de 1979). Si tuviéramos que agregar este atributo a los datos en la Tabla 3.2, tendría la mayor ganancia de información de cualquiera de los atributos. Esto se debe a que la fecha por sí sola predice perfectamente el atributo objetivo sobre los datos de entrenamiento. Por lo tanto, se seleccionará como el atributo de decisión para el nodo raíz del árbol y conducirá a un árbol (bastante amplio) de profundidad uno, que clasifica perfectamente los datos de entrenamiento. Por supuesto, a este árbol de decisiones le irá mal en los ejemplos posteriores, porque no es un predictor útil a pesar de que separa perfectamente los datos de entrenamiento.

¿Qué pasa con el atributo Fecha? En pocas palabras, tiene tantos valores posibles que está obligado a separar los ejemplos de entrenamiento en subconjuntos muy pequeños. Debido a esto, tendrá una ganancia de información muy alta en relación con los ejemplos de entrenamiento, a pesar de ser un predictor muy pobre de la función objetivo en instancias no vistas.

Una forma de evitar esta dificultad es seleccionar atributos de decisión basados en alguna medida que no sea la ganancia de información. Una medida alternativa que se ha utilizado con éxito es la relación de ganancia (Quinlan 1986). La medida de la relación de ganancia penaliza los atributos como Fecha incorporando un término, llamado información dividida, que es sensible a la forma en que el atributo divide los datos de manera amplia y uniforme:

$$\boldsymbol{SplitInformation}(\boldsymbol{S},\boldsymbol{A}) = -\sum_{\boldsymbol{i=1}}^{\boldsymbol{c}} \frac{|\boldsymbol{S_i}|}{|\boldsymbol{S}|} \boldsymbol{log_2} \frac{|\boldsymbol{S_i}|}{|\boldsymbol{S}|} \qquad (\mathbf{3.5})$$

Donde $\boldsymbol{S_1}$ a $\boldsymbol{S}$, son los subconjuntos $\boldsymbol{c}$ de ejemplos resultantes de la partición $\boldsymbol{S}$ por el atributo $\boldsymbol{A}$ de valores $\boldsymbol{c}$. Tenga en cuenta que SplitInformation es en realidad la entropía de $\boldsymbol{S}$ con respecto a los valores del atributo $\boldsymbol{A}$. Esto está en contraste con nuestros usos previos de entropía, en la cual consideramos solo la entropía de $\boldsymbol{S}$ con respecto al atributo objetivo cuyo valor debe ser predicho por el árbol aprendido.

La medida de GainRatio se define en términos de la medida de ganancia anterior, así como esta división dividida, de la siguiente manera

$$\boldsymbol{GainRatio(S,A) = \frac{Gain(S,A)}{SplitInformation(S,A)}} \qquad (\mathbf{3.6})$$

Tenga en cuenta que el término de Splitlnformation desalienta la selección de atributos con muchos valores distribuidos uniformemente. Por ejemplo, considere una colección de n ejemplos que están completamente separados por el atributo $\boldsymbol{A}$ (p. Ej., Fecha). En este caso, el valor de Splitlnformation será $\boldsymbol{log_2 n}$. Por el contrario, un atributo booleano $\boldsymbol{B}$ que divide los mismos $\boldsymbol{n}$ ejemplos exactamente a la mitad tendrá Splitlnformation de $\mathbf{1}$. Si los atributos $\boldsymbol{A}$ y $\boldsymbol{B}$ producen la misma ganancia de información, entonces claramente $\boldsymbol{B}$ puntuará más alto de acuerdo con la medida de GainRatio.

Una cuestión práctica que surge al utilizar GainRatio en lugar de Gain para seleccionar atributos es que el denominador puede ser cero o muy pequeño cuando $|\boldsymbol{S_i}| \approx |\boldsymbol{S}|$ para uno de los $\boldsymbol{S_i}$. Esto hace que GainRatio sea indefinido o muy grande para los atributos que tienen el mismo valor para casi todos los miembros de $\boldsymbol{S}$. Para evitar seleccionar atributos puramente sobre esta base, podemos adoptar alguna heurística como calcular primero la Ganancia de cada atributo, luego aplicando la prueba de GainRatio solo considerando aquellos atributos con ganancia superior a la media (Quinlan 1986).

Una alternativa al GainRatio, diseñada para abordar directamente la dificultad anterior, es una medida basada en la distancia introducida por López de Mantaras (1991). Esta medida se basa en la definición de una métrica de distancia entre las particiones de los datos. Cada atributo se evalúa en función de la distancia entre la partición de datos que crea y la partición perfecta (es decir, la partición que clasifica perfectamente los datos de entrenamiento). Se elige el atributo cuya partición está más cerca de la partición perfecta. López de Mantaras (1991) define esta medida de distancia, demuestra que no está sesgada hacia atributos con grandes cantidades de valores e informa estudios experimentales que indican que la precisión predictiva de los árboles inducidos no es significativamente diferente de la obtenida con la ganancia y ganancia Medidas de proporción Sin embargo, esta medida de distancia evita las dificultades prácticas asociadas con la medida GainRatio, y en sus experimentos produce árboles significativamente más pequeños en el caso de conjuntos de datos cuyos atributos tienen números de valores muy diferentes.

También se han propuesto una variedad de otras medidas de selección (por ejemplo, véase Breiman y otros 1984, Mingers 1989a, Kearns y Mansour 1996, Dietterich et al. 1996). Mingers (1989a) proporciona un análisis experimental de la efectividad relativa de varias medidas de selección sobre una variedad de problemas. Él informa diferencias significativas en los tamaños de los árboles sin podar producidos por las diferentes medidas de selección. Sin embargo, en sus dominios experimentales, la elección de la medida de selección de atributos parece tener un impacto menor en la precisión final que el alcance y el método de la poda posterior.

3.7.4 Manejo de ejemplos de entrenamiento con valores de atributo faltantes

En ciertos casos, los datos disponibles pueden ser valores perdidos para algunos atributos. Por ejemplo, en un dominio médico en el que deseamos predecir el resultado del paciente según diversas pruebas de laboratorio, es posible que el resultado del análisis de sangre en la prueba de laboratorio esté disponible solo para un subconjunto de pacientes. En tales casos, es común estimar el valor del atributo faltante en base a otros ejemplos para los cuales este atributo tiene un valor conocido.

Considere la situación en la que se calcula $\boldsymbol{Gain}(\boldsymbol{S}, \boldsymbol{A})$ en el nodo $\boldsymbol{n}$ en el árbol de decisión para evaluar si el atributo $\boldsymbol{A}$ es el mejor atributo para evaluar en este nodo de decisión. Supongamos que $(\boldsymbol{x}, \boldsymbol{c}\,(\boldsymbol{x}))$ es uno de los ejemplos de entrenamiento en $\boldsymbol{S}$ y que el valor $\boldsymbol{A}(\boldsymbol{x})$ es desconocido.

Una estrategia para lidiar con el valor del atributo que falta es asignarle el valor que es más común entre los ejemplos de capacitación en el nodo $\boldsymbol{n}$. Alternativamente, podríamos asignarle el valor más común entre los ejemplos en el nodo $\boldsymbol{n}$ que tienen la clasificación $\boldsymbol{c}(\boldsymbol{x})$. El ejemplo de entrenamiento elaborado que usa este valor estimado para $\boldsymbol{A}(\boldsymbol{x})$ puede usarse directamente mediante el algoritmo de aprendizaje de árbol de decisión existente. Esta estrategia es examinada por Mingers (1989a).

Un segundo procedimiento, más complejo, es asignar una probabilidad a cada uno de los valores posibles de $\boldsymbol{A}$ en lugar de simplemente asignarle el valor más común a $\boldsymbol{A}(\boldsymbol{x})$. Estas probabilidades se pueden estimar nuevamente basándose en las frecuencias observadas de los diversos valores para $\boldsymbol{A}$ entre los ejemplos en el nodo $\boldsymbol{n}$. Por ejemplo, dado un atributo booleano $\boldsymbol{A}$, si el nodo $\boldsymbol{n}$ contiene seis ejemplos conocidos con $\boldsymbol{A} = \boldsymbol{1}$ y cuatro con $\boldsymbol{A} = \boldsymbol{0}$, entonces diríamos que la probabilidad de que $\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{1}$ sea 0.6, y la probabilidad de que $\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{0}$ es 0.4. Ahora se distribuye una fracción de 0.6 de instancia $\boldsymbol{x}$ por la rama para $\boldsymbol{A} = \boldsymbol{1}$, y una fracción de 0.4 de $\boldsymbol{x}$ por la otra rama de árbol. Estos ejemplos fraccionarios se utilizan con el fin de calcular la ganancia de información y se pueden subdividir en ramas posteriores del árbol si se debe probar un segundo valor de atributo faltante. Este mismo fraccionamiento de ejemplos también se puede aplicar después del aprendizaje, para clasificar nuevas instancias cuyos valores de atributo son desconocidos. En este caso, la clasificación de la nueva instancia es simplemente la clasificación más probable, calculada sumando los pesos de los fragmentos de instancia clasificados de diferentes maneras en los nodos de hoja del árbol. Este método para manejar los valores de atributo faltantes se usa en C4.5 (Quinlan 1993).

3.7.5 Manejo de Atributos con Diferentes Costos

En algunas tareas de aprendizaje, los atributos de la instancia pueden tener costos asociados. Por ejemplo, al aprender a clasificar las enfermedades médicas, podríamos describir a los pacientes en términos de atributos como temperatura, resultados de biopsia, pulso, análisis de sangre, etc. Estos atributos varían significativamente en sus costos, tanto en términos de costo monetario como de costo para la comodidad del paciente. En tales tareas, preferiríamos árboles de decisión que usen atributos de bajo

costo donde sea posible, confiando en atributos de alto costo solo cuando sea necesario para producir clasificaciones confiables.

ID3 se puede modificar para tener en cuenta los costos de los atributos al introducir un término de costo en la medida de selección de atributos. Por ejemplo, podríamos dividir la Ganancia por el costo del atributo, de modo que se preferirían los atributos de menor costo. Si bien estas medidas sensibles a los costos no garantizan la búsqueda de un árbol de decisión óptimo sensible a los costos, sí sesgan la búsqueda a favor de los atributos de bajo costo.

Tan y Schlimmer (1990) y Tan (1993) describen uno de estos enfoques y lo aplican a una tarea de percepción del robot en la que el robot debe aprender a clasificar diferentes objetos según cómo puedan ser captados por el manipulador del robot. En este caso, los atributos corresponden a diferentes lecturas de sensor obtenidas por un sonar móvil en el robot. El costo del atributo se mide por el número de segundos requeridos para obtener el valor del atributo colocando y operando el sonar. Demuestran que se aprenden estrategias de reconocimiento más eficientes, sin sacrificar la precisión de clasificación, reemplazando la medida de selección de atributos de ganancia de información por la siguiente medida

$$\frac{\boldsymbol{Gain^2(S,A)}}{\boldsymbol{Cost(A)}}$$

Núñez (1988) describe un enfoque relacionado y su aplicación al aprendizaje de las reglas de diagnóstico médico. Aquí los atributos son síntomas diferentes y pruebas de laboratorio con costos diferentes. Su sistema usa una medida de selección de atributos algo diferente

$$\frac{\boldsymbol{2^{Gain(S,A)}-1}}{\boldsymbol{\left(Cost(A+1)\right)^{w}}}$$

Donde $\boldsymbol{w \in [0,1]}$ es una constante que determina la importancia relativa del costo frente a la ganancia de información. Núñez (1991) presenta una comparación empírica de estos dos enfoques en una variedad de tareas.

3. 8 RESUMEN Y LECTURA ADICIONAL

Los puntos principales de este capítulo incluyen:

- El aprendizaje del árbol de decisiones proporciona un método práctico para el aprendizaje conceptual y para el aprendizaje de otras funciones de valores discretos. La familia de algoritmos ID3 infiere los árboles de decisión haciéndolos crecer desde la raíz hacia abajo, seleccionando ávidamente el siguiente mejor atributo para cada nueva rama de decisión añadida al árbol.
- ID3 busca un espacio de hipótesis completo (es decir, el espacio de árboles de decisión puede representar cualquier función de valores discretos definida sobre instancias de valores discretos). Por lo tanto, evita la mayor dificultad asociada con los enfoques que consideran solo conjuntos restringidos de hipótesis: que la función objetivo puede no estar presente en el espacio de la hipótesis.
- El sesgo inductivo implícito en ID3 incluye una preferencia por árboles más pequeños; es decir, su búsqueda a través del espacio de hipótesis hace que el

árbol sea tan grande como sea necesario para clasificar los ejemplos de capacitación disponibles.

- Sobreajustar los datos de entrenamiento es un tema importante en el aprendizaje del árbol de decisión. Debido a que los ejemplos de capacitación son solo una muestra de todas las instancias posibles, es posible agregar ramas al árbol que mejoren el rendimiento en los ejemplos de capacitación mientras se disminuye el rendimiento en otras instancias fuera de este conjunto. Los métodos para posponer el árbol de decisión son importantes para evitar el sobreajuste en el aprendizaje del árbol de decisión (y otros métodos de inferencia inductiva que emplean un sesgo de preferencia).
- Una gran variedad de extensiones para el algoritmo ID3 básico ha sido desarrollada por diferentes investigadores. Estos incluyen métodos para podar árboles, manejar los atributos reales, acomodar ejemplos de entrenamiento con valores de atributo faltantes, refinar incrementalmente los árboles de decisión a medida que nuevos ejemplos de entrenamiento estén disponibles, usar medidas de selección de atributos que no sean ganancia de información, y considerar los costos asociados con atributos de instancia.

Entre los primeros trabajos sobre el aprendizaje del árbol de decisiones se encuentra el Sistema de Aprendizaje Conceptual (CLS) de Hunt (Hunt et al., 1966) y el trabajo de Friedman y Breiman que resulta en el sistema CART (Friedman 1977; Breiman et al., 1984). El sistema ID3 de Quinlan (Quinlan 1979, 1983) forma la base de la discusión en este capítulo. Otros trabajos iniciales sobre el aprendizaje del árbol de decisión incluyen ASSISTANT (Kononenko y otros 1984, Cestnik et al., 1987). Las implementaciones de los algoritmos de inducción del árbol de decisión ahora están disponibles comercialmente en muchas plataformas informáticas.

Para obtener más detalles sobre la inducción del árbol de decisión, un excelente libro de Quinlan (1993) analiza muchos problemas prácticos y proporciona código ejecutable para C4.5. Mingers (1989a) y Buntine y Niblett (1992) proporcionan dos estudios experimentales que comparan diferentes medidas de selección de atributos. Mingers (1989b) y Malerba et al. (1995) proporcionan estudios de diferentes estrategias de poda. Los experimentos que comparan el aprendizaje del árbol de decisión y otros métodos de aprendizaje se pueden encontrar en numerosos documentos, incluidos (Dietterich y col., 1995, Fisher y McKusick 1989, Quinlan 1988a, Shavlik y cols., 1991, Thrun y cols., 1991, Weiss y Kapouleas 1989).

Métodos basados en el árbol

(Capítulo 8 del libro An Introduction to Statistical Learning)

En este capítulo, describimos los métodos basados en el árbol para regresión y clasificación. Estos implican la **estratificación** o **segmentación** del espacio del predictor en un número de regiones simples. Para hacer una predicción de una observación determinada, normalmente utilizamos la media o el modo de las observaciones de la formación en la región a la que pertenece. Puesto que el conjunto de

reglas usadas para segmentar el espacio del predictor de separación puede resumirse en un árbol, este tipo de enfoques es conocido como métodos de árbol de decisión.

Los métodos basados en árboles son simples y útiles para la interpretación. Sin embargo, por lo general no son competitivos con los mejores enfoques de aprendizaje supervisado, como los que se ven en los Capítulos 6 y 7, en términos de exactitud de la predicción. Por lo tanto, en este capítulo también introducimos el **bagging**, **random forests** (los bosques aleatorios) y el **boosting**. Cada uno de estos enfoques implica la producción de múltiples árboles que luego se combinan para producir una sola predicción de consenso. Veremos que la combinación de un gran número de árboles a menudo puede dar lugar a mejoras dramáticas en la exactitud de la predicción, a expensas de cierta pérdida de interpretación.

(8.1) Los conceptos básicos de árboles de decisión

Árboles de decisión pueden aplicarse a problemas de regresión y clasificación. En primer lugar, consideramos problemas de regresión y luego pasar a la clasificación.

(8.1.1) Árboles de regresión

Para motivar árboles de regresión, comenzamos con un ejemplo simple.

FIGURE 8.1. *For the* `Hitters` *data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year. At a given internal node, the label (of the form* $X_j < t_k$*) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to* $X_j \geq t_k$*. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to* `Years<4.5`*, and the right-hand branch corresponds to* `Years>=4.5`*. The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.*

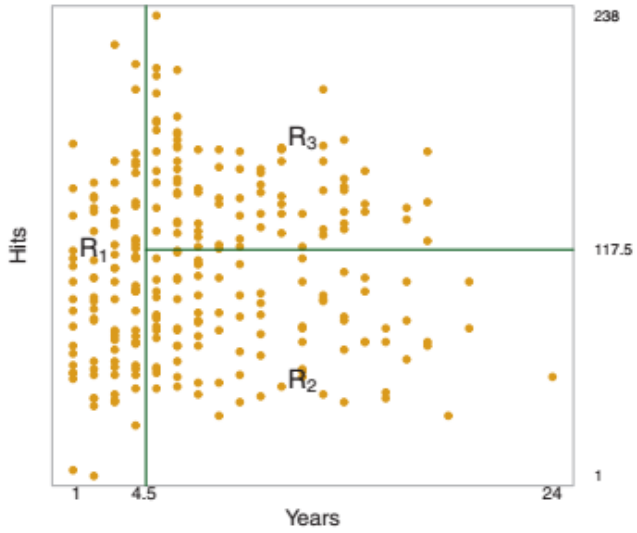

FIGURE 8.2. *The three-region partition for the* `Hitters` *data set from the regression tree illustrated in Figure 8.1.*

Predicción de los sueldos de los jugadores de béisbol utilizando árboles de regresión

Usamos el conjunto de datos de **Hitters** para predecir el **salario** (**Salary**) de un jugador de béisbol en función de los **años** (**Years**) (el número de años que ha jugado en las ligas mayores) y los **hits** (**Hits**) (el número de hits que realizó en el año anterior). Primero eliminamos las observaciones que faltantes en los valores de **salario** (**Salary**) y el **salario** (**Salary**) en la transformación de registro para que su distribución tenga una forma de campana más típica. (Recuerde que el salario se mide en miles de dólares).

La figura 8.1 muestra un árbol de regresión que se ajusta a esta información. Consiste en una serie de reglas de división, comenzando en la parte superior del árbol. La división superior asigna observaciones que tienen **Años** (**Years<4.5)** a la rama izquierda. El salario previsto para estos jugadores viene dado por el valor de respuesta promedio para los jugadores en el conjunto de datos con **Años** (**Years<4.5**). Para esos jugadores, el salario medio de registro es **5.107**, por lo que hacemos una predicción de $e^{5.107}$ miles de dólares, es decir, **$165,174**, para estos jugadores. Los jugadores con Años (**Years> = 4.5)** se asignan a la rama derecha, y luego ese grupo se subdivide en **Hits**. En general, el árbol estratifica o segmenta a los jugadores en tres regiones de espacio predictivo: jugadores que han jugado durante cuatro o menos años, jugadores que han jugado durante cinco o más años y que obtuvieron menos de 118 hits el año pasado, y jugadores que han jugado durante cinco o más años y quien hizo al menos 118 hits el año pasado. Estás regiones pueden ser escritas como $\boldsymbol{R_1 = \{X|Years < 4.5\}}$ $\boldsymbol{R_2 = \{X|Years \geq 4.5, Hits < 117.5\}}$, y $\boldsymbol{R_3 = \{X|Years \geq 4.5, Hits \geq 117.5\}}$. La Figura 8.2 ilustra las regiones como una función de Años (**Years**) e **Hits**. Los salarios previstos para estos tres grupos son \$1,000×$e^{5.107}$ = \$ 65,174, \$1,000×$e^{5.999}$ = \$402,834 y \$1,000 ×$e^{6.740}$ = \$845,346, respectivamente.

De acuerdo con la analogía del árbol, las regiones $\boldsymbol{R_1, R_2}$ $\boldsymbol{y}$ $\boldsymbol{R_3}$ se conocen como nodos terminales u hojas del árbol. Como en el caso de la Figura 8.1, los árboles de decisión se dibujan típicamente al revés, en el sentido de que las hojas están en la parte inferior del árbol. Los puntos a lo largo del árbol donde se divide el espacio del predictor se denominan nodos internos. En la Figura 8.1, los dos nodos internos están indicados por el texto Años (**Years**) < 4.5 e Impactos (**Hits**) < 117.5. Nos referimos a los segmentos de los árboles que conectan los nodos como ramas.

Podríamos interpretar el árbol de regresión que se muestra en la Figura 8.1 de la siguiente manera: Años (**Years**) es el factor más importante para determinar el Salario (**Salary**), y los jugadores con menos experiencia ganan salarios más bajos que los jugadores más experimentados. Dado que un jugador tiene menos experiencia, el número de hits que hizo en el año anterior parece tener un rol pequeño en su salario. Pero entre los jugadores que han estado en las ligas mayores durante cinco o más años, la cantidad de hits realizados en el año anterior afecta el salario, y los jugadores que lograron más hits el año pasado tienden a tener salarios más altos. El árbol de regresión que se muestra en la Figura 8.1 es probable que sea una simplificación excesiva de la verdadera relación entre Impactos (**Hits**), Años (**Years**) y Salario (**Salary**). Sin embargo, tiene ventajas sobre otros tipos de modelos de regresión (como los que se ven en los Capítulos 3 y 6): es más fácil de interpretar y tiene una buena representación gráfica.

Predicción a través de la estratificación del espacio de características

Ahora discutimos el proceso de construcción de un árbol de regresión. En términos generales, hay dos pasos.

1. Dividimos el espacio del predictor, es decir, el conjunto de posibles valores para $\boldsymbol{X_1, X_2, \ldots, X_p}$ en $\boldsymbol{J}$ regiones distintas no superpuestas $\boldsymbol{R_1, R_2, \ldots, R_J}$.
2. Para cada observación que cae en la región de $\boldsymbol{R_j}$, hacemos la misma predicción, que es simplemente la media de los valores de respuesta para las observaciones de entrenamiento en $\boldsymbol{R_j}$.

Por ejemplo, supongamos que en el paso **1** se obtienen dos regiones, $\boldsymbol{R_1}$ y $\boldsymbol{R_2}$, y que la media de respuesta de las observaciones de entrenamiento en la primera región es **10**, mientras que la media de respuesta de las observaciones de entrenamiento en la segunda región es de **20**. Entonces para una determinada observación $\boldsymbol{X = x}$, si $\boldsymbol{x \in R_1}$ a predecir un valor de **10**, y si $\boldsymbol{x \in R_2}$ será predecir un valor de **20**.

Ahora elaboramos el paso 1. ¿Cómo construimos las regiones $\boldsymbol{R_1, R_2, \ldots, R_J}$? En teoría, las regiones podrían tener cualquier forma. Sin embargo, decidimos dividir el espacio del predictor en rectángulos multidimensional, o cajas, por simplicidad y para facilitar la interpretación del modelo predictivo resultante. El objetivo es encontrar cajas $\boldsymbol{R_1, R_2, \ldots, R_J}$ que minimizan el RSS, por

$$\boldsymbol{RSS} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{J}} \sum_{\boldsymbol{i \in R_j}} \left(\boldsymbol{y_i - \hat{y}_{R_j}}\right) \qquad (\mathbf{8.1})$$

Donde $\boldsymbol{\hat{y}_{R_j}}$ es la respuesta media para las observaciones de entrenamiento dentro de la j-exima caja. Lamentablemente, es computacionalmente inviable considerar cada posible partición del espacio de característica en J cajas. Por esta razón, tomamos un enfoque top-down, codicioso que se conoce como división binaria recursiva. El enfoque es de arriba hacia abajo porque comienza en la parte superior del árbol (momento en el que todas las observaciones pertenecen a una sola región) y, a continuación, divide sucesivamente el espacio predictor; cada división se indica a través de dos nuevas ramas más abajo en el árbol. Es codicioso porque en cada paso del proceso de construcción de árboles, la mejor división se hace en ese paso en particular, en lugar de mirar hacia delante y escoger una división que conducirá a un árbol mejor en algún paso futuro.

Para llevar a cabo división binaria recursiva, primero seleccionamos el predictor X_j y el punto de corte s que divide el espacio del predictor en las regiones $\{X|X_j < s\}$ y $\{X|X_j \geq s\}$ que conducen a la mayor reducción posible en RSS. (La notación $\{X|X_j < s\}$ significa la región del espacio del predictor X_j toma un valor menor que s.) Es decir, consideramos todos predictores $X_1, \ldots, X_p$ y todos los valores posibles del punto de corte s para cada uno de los predictores elija el predictor y punto de corte tal que el árbol resultante tiene el RSS más bajo. En mayor detalle, para cualquier j y s, definimos el par de medianos planos

$$\boldsymbol{R_1(j,s) = \{X | X_j < s\}} \qquad \boldsymbol{y} \qquad \boldsymbol{R_2(j,s) = \{X | X_j \geq s\}} \qquad (\mathbf{8.2})$$

Y buscamos el valor de j y s que minimizan la ecuación

$$\sum_{\boldsymbol{i: x_i \in R_1(j,s)}} \left(\boldsymbol{y_i - \hat{y}_{R_1}}\right)^2 + \sum_{\boldsymbol{i: x_i \in R_2(j,s)}} \left(\boldsymbol{y_i - \hat{y}_{R_2}}\right)^2 \qquad (\mathbf{8.3})$$

Donde $\boldsymbol{\hat{y}_{R_1}}$ es la respuesta media para las observaciones de entrenamiento en $\boldsymbol{R_1(j,s)}$, y $\boldsymbol{\hat{y}_{R_2}}$ es la respuesta media para las observaciones de entrenamiento en $\boldsymbol{R_2(j,s)}$. Encontrar los valores de j y s que minimizan (8.3) se puede hacer bastante rápido, especialmente cuando el número de características p no es demasiado grande.

A continuación, repetimos el proceso, buscando el mejor predictor y el mejor punto de corte para dividir aún más los datos a fin de minimizar el RSS dentro de cada una de las regiones resultantes. Sin embargo, esta vez, en lugar de dividir todo el espacio del predictor, dividimos una de las dos regiones previamente identificadas. Ahora tenemos tres regiones. Una vez más, buscamos dividir una de estas tres regiones más, para minimizar el RSS. El proceso continúa hasta que se alcanza un criterio de detención; por ejemplo, podemos continuar hasta que ninguna región contenga más de cinco observaciones.

Una vez que las regiones $\boldsymbol{R_1, R_2, \ldots, R_J}$ se han creado, predecimos la respuesta para una observación de prueba dada utilizando la media de las observaciones de entrenamiento en la región a la que pertenece la observación de prueba.

Un ejemplo de cinco regiones de este enfoque se muestra en la Figura 8.3.

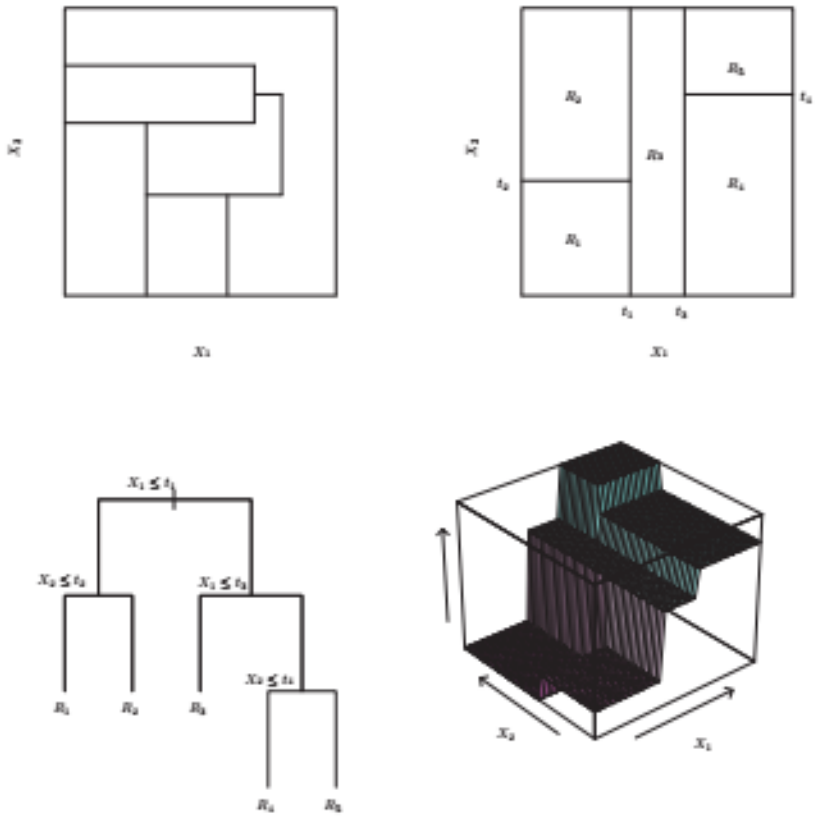

FIGURE 8.3. Top Left: *A partition of two-dimensional feature space that could not result from recursive binary splitting.* Top Right: *The output of recursive binary splitting on a two-dimensional example.* Bottom Left: *A tree corresponding to the partition in the top right panel.* Bottom Right: *A perspective plot of the prediction surface corresponding to that tree.*

Poda de árboles

El proceso descrito anteriormente puede producir buenas predicciones en el conjunto de entrenamiento, pero es probable que sobreajuste los datos, lo que conduce a un rendimiento del conjunto de prueba pobre. Esto se debe a que el árbol resultante puede ser demasiado complejo. Un árbol más pequeño con menos divisiones (es decir, menos regiones $R_1, \ldots, R_j$) puede conducir a una menor varianza y una mejor interpretación a costa de un pequeño sesgo. Una posible alternativa al proceso descrito anteriormente es construir el árbol siempre que la disminución en el RSS debido a cada división exceda algún umbral (alto). Esta estrategia dará como resultado árboles más pequeños, pero es demasiado miope, ya que una división aparentemente inútil al inicio del árbol podría estar seguida por una muy buena división, es decir, una división que conduce a una gran reducción en RSS más adelante.

Por lo tanto, una mejor estrategia es hacer crecer un árbol muy grande T_0, y luego podarlo para obtener un árbol secundario. ¿Cómo determinamos la mejor manera de podar el árbol? Intuitivamente, nuestro objetivo es seleccionar un subárbol que conduzca a la tasa de error de prueba más baja. Dado un subárbol, podemos estimar su error de prueba utilizando la validación cruzada o el enfoque del conjunto de validación. Sin embargo, estimar el error de validación cruzada para cada subárbol posible sería demasiado engorroso, ya que existe una gran cantidad de subárboles posibles.

En cambio, necesitamos una forma de seleccionar un pequeño conjunto de subárboles para su consideración.

La reducción de la complejidad de los costos, también conocida como la poda de enlaces más débil, nos permite hacer esto. En lugar de considerar todos los subárboles posibles, consideramos una secuencia de árboles indexada por un parámetro de ajuste no negativo α.

Algoritmo 8.1: Construyendo un árbol de regresión

1. Use división binaria recursiva para hacer crecer un árbol grande en los datos de entrenamiento, deteniéndose solo cuando cada nodo terminal tenga menos de un número mínimo de observaciones.
2. Aplique la poda de complejidad de costos al árbol grande para obtener una secuencia de los mejores subárboles, como una función de $\boldsymbol{\alpha}$.
3. Use la validación cruzada de $\boldsymbol{K}$-fold para elegir α. Es decir, divida las observaciones de entrenamiento en K pliegues. Para cada $\boldsymbol{k = 1, \ldots, K}$:
    a. Repita los pasos 1 y 2 en todos menos en el pliegue de orden k-eximo de los datos de entrenamiento.
    b. Evalúe el error de predicción cuadrático medio sobre los datos en el pliegue $\boldsymbol{k}$-eximo del lado izquierdo, como una función de $\boldsymbol{\alpha}$.

    Promedie los resultados para cada valor de $\boldsymbol{\alpha}$, y elija $\boldsymbol{\alpha}$ para minimizar el error promedio.
4. Devuelve el subárbol del Paso 2 que corresponde al valor elegido de $\boldsymbol{\alpha}$.

Para cada valor de $\boldsymbol{\alpha}$, corresponde un subárbol $\boldsymbol{T \subset T_0}$ que es lo más pequeño posible.

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2 + \alpha |T| \qquad (8.4)$$

Aquí $\boldsymbol{|T|}$ indica el número de nodos terminales del árbol $\boldsymbol{T}$, $\boldsymbol{R_m}$ es el rectángulo (es decir, el subconjunto del espacio del predictor) correspondiente al $\boldsymbol{m}$-ésimo nodo terminal, y $\boldsymbol{\hat{y}_{R_m}}$ es la respuesta pronosticada asociada con $\boldsymbol{R_m}$, es decir, la media de las observaciones de entrenamiento en $\boldsymbol{R_m}$. El parámetro de ajuste $\boldsymbol{\alpha}$ controla una compensación entre la complejidad del subárbol y su ajuste a los datos de

entrenamiento. Cuando $\boldsymbol{\alpha = 0}$, entonces el subárbol $\boldsymbol{T}$ simplemente será igual a $\boldsymbol{T_0}$, porque entonces (8.4) simplemente mide el error de entrenamiento. Sin embargo, a medida que $\boldsymbol{\alpha}$ aumenta, hay que pagar un precio por tener un árbol con muchos nodos terminales, por lo que la cantidad (8.4) tenderá a minimizarse para un subárbol más pequeño. La ecuación 8.4 es una reminiscencia del lazo (6.7) del capítulo 6, en el que se utilizó una formulación similar para controlar la complejidad de un modelo lineal.

Resulta que a medida que aumentamos $\boldsymbol{\alpha}$ desde cero en (8.4), las ramas se podan del árbol de forma anidada y predecible, por lo que es fácil obtener la secuencia completa de subárboles en función de $\boldsymbol{\alpha}$. Podemos seleccionar un valor de $\boldsymbol{\alpha}$ usando un conjunto de validación o usando validación cruzada. Luego volvemos al conjunto completo de datos y obtenemos el subárbol correspondiente a $\boldsymbol{\alpha}$. Este proceso se resume en el algoritmo 8.1.

Las Figuras 8.4 y 8.5 muestran los resultados de ajustar y podar un árbol de regresión en los datos de Hitters, utilizando nueve de las características. Primero, dividimos aleatoriamente el conjunto de datos a la mitad, produciendo 132 observaciones en el conjunto de entrenamiento y 131 observaciones en el conjunto de prueba. Luego construimos un gran árbol de regresión en los datos de entrenamiento y variamos α en (8.4) para crear subárboles con diferentes números de nodos terminales. Finalmente, realizamos una validación cruzada de seis veces para estimar el MSE con validación cruzada de los árboles en función de $\boldsymbol{\alpha}$. (Elegimos realizar una validación cruzada de seis veces porque 132 es un múltiplo exacto de seis). El árbol de regresión sin podar se muestra en la Figura 8.4. La curva verde en la figura 8.5 muestra el error CV como una función del número de hojas, 2 mientras que la curva naranja indica el error de prueba. También se muestran barras de error estándar alrededor de los errores estimados. Como referencia, la curva de error de entrenamiento se muestra en negro. El error de CV es una aproximación razonable del error de prueba: el error de CV toma su mínimo para un árbol de tres nodos, mientras que el error de prueba también se reduce en el árbol de tres nodos (aunque toma su valor más bajo a los diez -árbol de nodos). El árbol podado que contiene tres nodos terminales se muestra en la Figura 8.1.

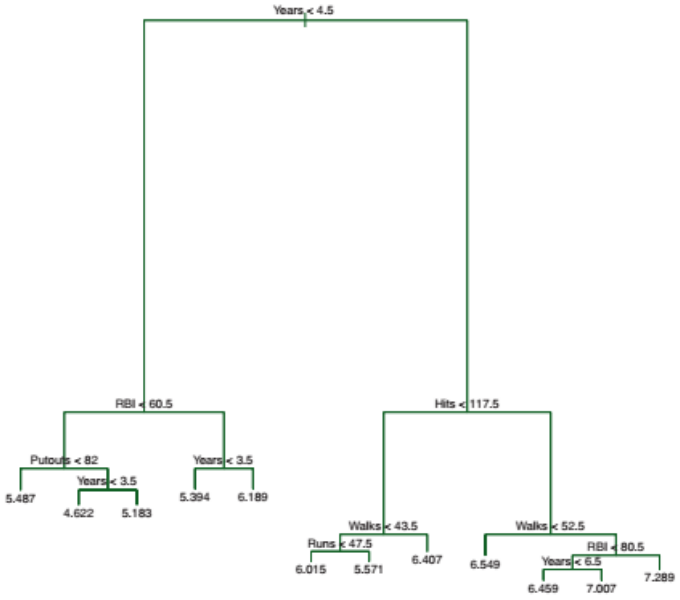

FIGURE 8.4. *Regression tree analysis for the* `Hitters` *data. The unpruned tree that results from top-down greedy splitting on the training data is shown.*

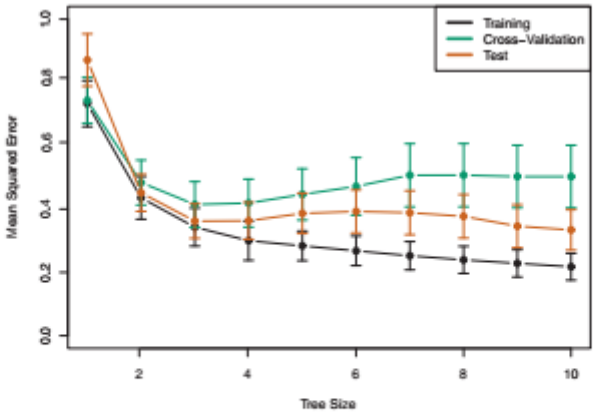

FIGURE 8.5. *Regression tree analysis for the* `Hitters` *data. The training, cross-validation, and test MSE are shown as a function of the number of terminal nodes in the pruned tree. Standard error bands are displayed. The minimum cross-validation error occurs at a tree size of three.*

(8.1.2) Árbol de clasificación

Un árbol de clasificación es muy similar a un árbol de regresión, excepto que se usa para predecir una respuesta cualitativa en lugar de una cuantitativa. Vuelva a llamar eso para un árbol de regresión, la respuesta pronosticada para una observación viene dada por la respuesta media de las observaciones de entrenamiento que pertenecen al mismo nodo terminal. En contraste, para un árbol de clasificación, predecimos que cada observación pertenece a la **clase más común de observaciones de entrenamiento** en la región a la que pertenece. Al interpretar los resultados de un árbol de clasificación, a menudo nos interesa no solo la predicción de clase correspondiente a una región de nodo terminal particular, sino también las **proporciones** de clase entre las observaciones de entrenamiento que caen dentro de esa región.

La tarea de hacer crecer un árbol de clasificación es bastante similar a la tarea de hacer crecer un árbol de regresión. Al igual que en la configuración de regresión, utilizamos la **división binaria recursiva** para hacer crecer un árbol de clasificación. Sin embargo, en la configuración de clasificación, RSS no se puede utilizar como criterio para realizar las divisiones binarias. Una alternativa natural a RSS es la tasa de **error de clasificación**. Como planeamos asignar una observación en una región determinada a la clase más común de observaciones de entrenamiento en esa región, la tasa de **error de clasificación** es simplemente la **fracción de las observaciones de entrenamiento en esa región que no pertenecen a la clase más común**:

$$\boldsymbol{E = 1 - \max_{k}(\hat{p}_{mk})}$$

Aquí $\boldsymbol{\hat{p}_{mk}}$ representa la **proporción de observaciones de entrenamiento en la región m-ésima que son de la clase k-ésima**. Sin embargo, resulta que el error de clasificación no es lo suficientemente sensible para el cultivo de árboles, y en la práctica son preferibles otras dos medidas.

El Índice de Gini es definido por

$$\boldsymbol{G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})}$$

Una medida de la varianza total en las K clases. No es difícil ver que el índice de Gini toma un pequeño valor si todos los $\boldsymbol{\hat{p}_{mk}}$ están cerca de cero o uno. Por esta razón, se hace referencia al índice de Gini como una medida de la **pureza** del nodo: un valor pequeño indica que un nodo contiene predominantemente observaciones de una sola clase.

Una alternativa al índice de Gini es la entropía cruzada, dada por

$$\boldsymbol{D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}}$$

Dado que $0 \leq \boldsymbol{\hat{p}_{mk}} \leq 1$, se deduce que $0 \leq -\boldsymbol{\hat{p}_{mk}} \log \boldsymbol{\hat{p}_{mk}}$. Uno puede mostrar que la entropía cruzada tomará un valor cercano a cero si los $\boldsymbol{\hat{p}_{mk}}$ están todos cerca de cero o cerca de uno. Por lo tanto, al igual que el índice de Gini, la entropía cruzada tomará un pequeño valor si el m-ésimo nodo es puro. De hecho, resulta que el índice de Gini y la entropía cruzada son bastante similares numéricamente.

Al construir un árbol de clasificación, el índice de Gini o la entropía cruzada se utilizan generalmente para evaluar la calidad de una división en particular, ya que estos dos enfoques son más sensibles a la **pureza** del nodo que la **tasa de error de clasificación**. Cualquiera de estos tres enfoques podría usarse al podar el árbol, pero la **tasa de error de clasificación** es preferible si la meta es la precisión de predicción del árbol podado final.

La Figura 8.6 muestra un ejemplo en el conjunto de datos **Heart**. Estos datos contienen un resultado binario **HD** para 303 pacientes que presentaron dolor en el pecho. Un valor de resultado de **Sí** (**Yes**) indica la presencia de enfermedad cardíaca basada en una prueba angiográfica, mientras que **No** (**No**) significa que no hay enfermedad cardíaca. Hay 13 predictores que incluyen la **edad** (**Age**), el **sexo** (**Sex**), el **colesterol** (**Chol**) (una medición de colesterol) y otras mediciones de la función cardíaca y pulmonar. La validación cruzada da como resultado un árbol con seis nodos terminales.

En nuestra discusión hasta ahora, hemos asumido que las variables predictoras toman valores continuos. Sin embargo, los árboles de decisión se pueden construir incluso en presencia de variables predictivas cualitativas. Por ejemplo, en los datos **Heart**, algunos de los predictores, como **Sex**, **Thal** (prueba de estrés de talio) y **ChestPain**, son cualitativos.

Por lo tanto, una división en una de estas variables equivale a asignar algunos de los valores cualitativos a una rama y asignar el resto a la otra rama. En la Figura 8.6, algunos de los nodos internos corresponden a la división de variables cualitativas. Por ejemplo, el nodo interno superior corresponde a la división de **Thal**. El texto **Thal: a** indica que la rama izquierda que sale de ese nodo se compone de observaciones con el primer valor de la variable **Thal** (normal), y el nodo derecho consiste en las observaciones restantes (defectos fijos o reversibles). El texto **ChestPain: bc** dos divisiones en el árbol de la izquierda indica que la rama izquierda que sale de ese nodo consta de observaciones con el segundo y tercer valor de la variable **ChestPain**, donde los valores posibles son angina típica, angina atípica, dolor no anginoso y asintomático.

La figura 8.6 tiene una característica sorprendente: algunas de las divisiones producen dos nodos terminales que tienen el **mismo valor predicho**. Por ejemplo, considere la división **RestECG<1** cerca de la parte inferior derecha del árbol no podado. Independientemente del valor de **RestECG**, se predice un valor de respuesta de **Sí** (**Yes**) para estas observaciones. ¿Por qué, entonces, se realiza la división en absoluto? La división se realiza porque conduce a una mayor pureza de nodo. Es decir, las 9 observaciones correspondientes a la hoja de la derecha tienen un valor de respuesta de **Sí** (**Yes**), mientras que 7/11 de las que corresponden a la hoja de la izquierda tienen un valor de respuesta de **Sí** (**Yes**). ¿Por qué es importante la pureza del nodo? Supongamos que tenemos una observación de prueba que pertenece a la región dada por esa hoja de la derecha. Entonces podemos estar bastante seguros de que su valor de respuesta es **Sí (Yes)**. Por el contrario, si una observación de prueba pertenece a la región dada por la hoja de la izquierda, entonces su valor de respuesta es probablemente **Sí** (**Yes**), pero estamos mucho menos seguros. Aunque la división **RestECG <1** no reduce el error de clasificación, mejora el índice de Gini y la entropía cruzada, que son más sensibles a la pureza del nodo.

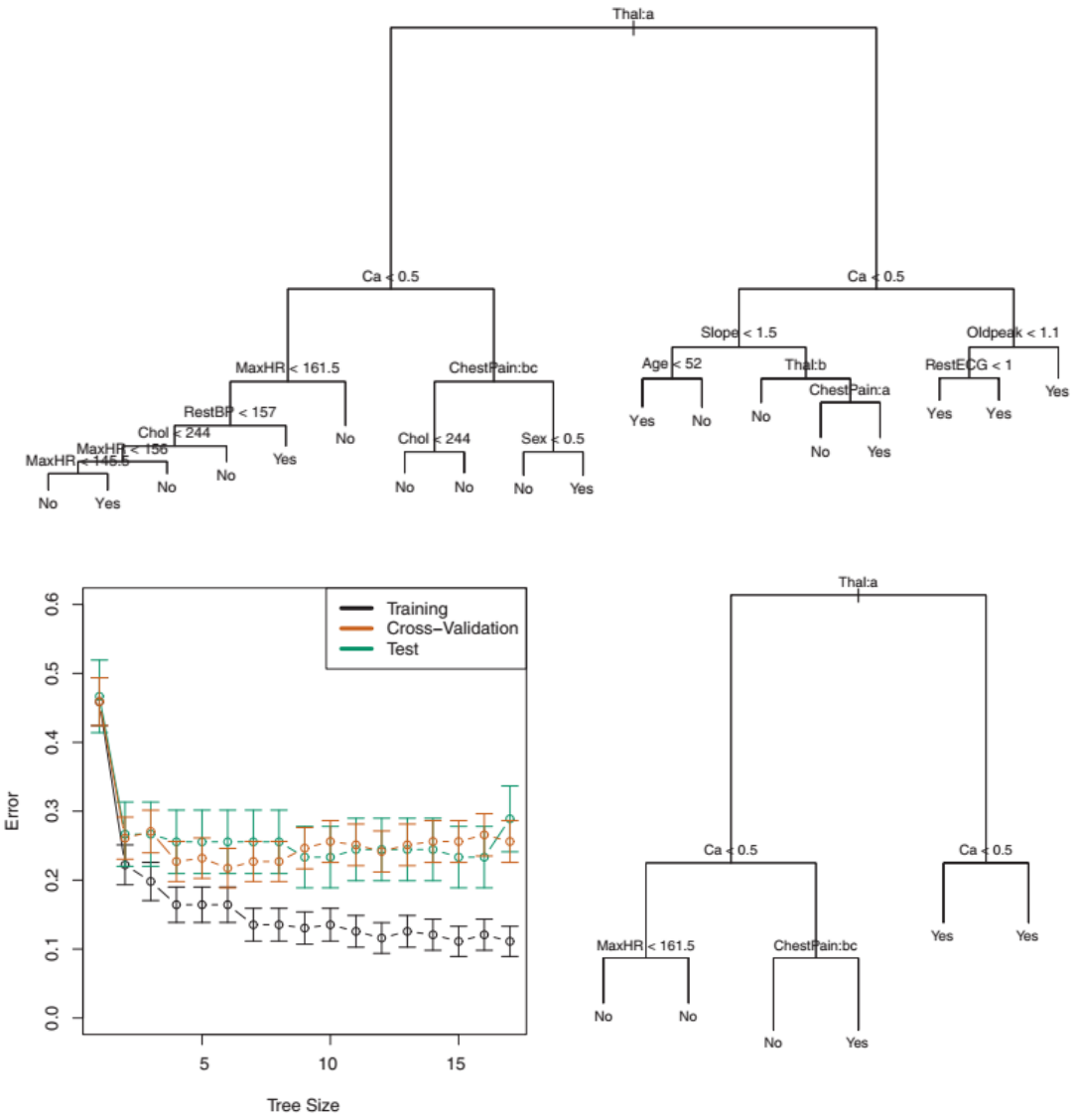

FIGURA 8.6. Datos del corazón (Heart). Arriba: el árbol sin podar. Abajo a la izquierda: error de validación cruzada, entrenamiento y error de prueba para diferentes tamaños del árbol podado. Abajo a la derecha: el árbol podado que corresponde al mínimo error de validación cruzada.

8.1.3 Árboles versus modelos lineales

Los árboles de regresión y clasificación tienen un sabor muy diferente de los enfoques más clásicos para la regresión y la clasificación presentados en los Capítulos 3 y 4. En particular, la regresión lineal supone un modelo de la forma

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j, \qquad (8.8)$$

mientras que los árboles de regresión asumen un modelo de la forma

$$f(X) = \beta_0 + \sum_{m=1}^{M} c_m * 1_{(X \epsilon R_m)}, \qquad (8.9)$$

donde $R_1, R_2, \ldots, R_M$ representa una partición del espacio de características, como en la Figura 8.3.

¿Qué modelo es mejor? Depende del problema en cuestión. Si la relación entre las características y la respuesta es bien aproximada por un modelo lineal como en (8.8), entonces un enfoque como la regresión lineal probablemente funcionará bien, y superará un método como un árbol de regresión que no explota esta estructura lineal. Si, en cambio, existe una relación altamente no lineal y compleja entre las características y la respuesta según lo indicado por el modelo (8.9), entonces los árboles de decisión pueden superar los enfoques clásicos. Un ejemplo ilustrativo se muestra en la Figura 8.7. Los rendimientos relativos de los enfoques basados en árboles y clásicos pueden evaluarse estimando el error de la prueba, utilizando la validación cruzada o el enfoque del conjunto de validación (Capítulo 5).

Por supuesto, otras consideraciones más allá del simple error de prueba pueden entrar en juego al seleccionar un método de aprendizaje estadístico; por ejemplo, en ciertas configuraciones, la predicción usando un árbol puede ser preferible por razones de interpretación y visualización.

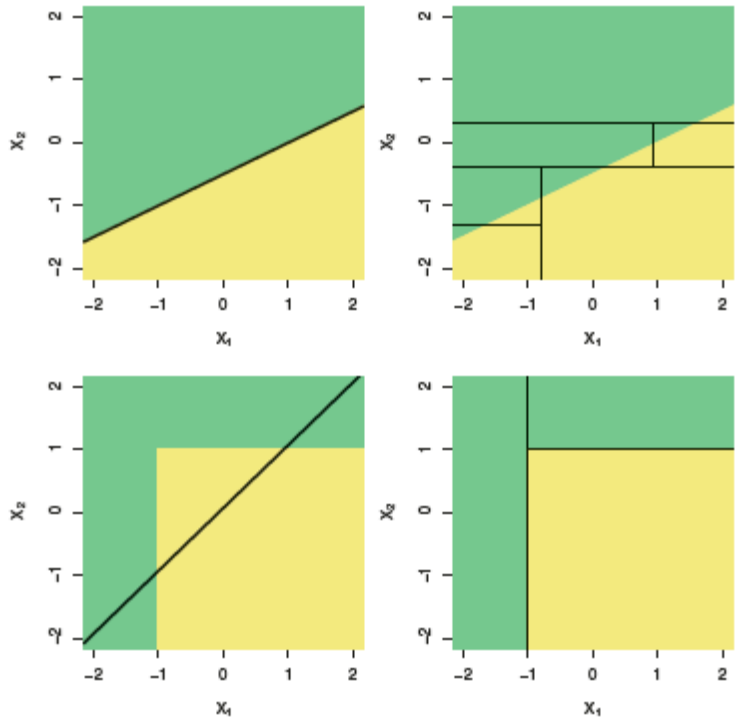

FIGURE 8.7. Top Row: *A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right).* Bottom Row: *Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).*

Fila superior: un ejemplo de clasificación bidimensional en el que el límite de decisión verdadero es lineal, y está indicado por las regiones sombreadas. Un enfoque clásico que asume un límite lineal (izquierda) superará a un árbol de decisión que realiza divisiones paralelas a los ejes (derecha). Fila inferior: aquí el límite de decisión verdadero no es lineal. Aquí un modelo lineal no puede capturar el límite de decisión verdadero (izquierda), mientras que un árbol de decisión es exitoso (derecha).

8.1.4 Ventajas y desventajas de los árboles

Los árboles de decisión para la regresión y la clasificación tienen una serie de ventajas sobre los enfoques más clásicos que se ven en los Capítulos 3 y 4:

- Los árboles son muy fáciles de explicar a las personas. De hecho, ¡son incluso más fáciles de explicar que la regresión lineal!
- Algunas personas creen que los árboles de decisiones reflejan más estrechamente la toma de decisiones humanas que los enfoques de regresión y clasificación vistos en los capítulos anteriores.
- Los árboles se pueden mostrar gráficamente y son fácilmente interpretables incluso por un no experto (especialmente si son pequeños).
- Los árboles pueden manejar fácilmente los predictores cualitativos sin la necesidad de crear variables ficticias.
- Desafortunadamente, los árboles generalmente no tienen el mismo nivel de precisión predictiva que algunos de los otros enfoques de regresión y clasificación que se ven en este libro.

Sin embargo, al agregar muchos árboles de decisión, utilizando métodos como el embolsado (bagging), los bosques aleatorios (random forests) y el refuerzo (boosting), el rendimiento predictivo de los árboles puede mejorarse sustancialmente. Presentamos estos conceptos en la siguiente sección.

8.2 Bagging, Random Forests, Boosting

Bagging, Random Forests, y el boosting utilizan los árboles como bloques de construcción para construir modelos de predicción más potentes.

8.2.1 Bagging

El bootstrap, presentado en el Capítulo 5, es una idea extremadamente poderosa. Se usa en muchas situaciones en las que es difícil o incluso imposible calcular directamente la desviación estándar de una cantidad de interés. Vemos aquí que el bootstrap se puede utilizar en un contexto completamente diferente, con el fin de mejorar los métodos de aprendizaje estadísticos, como los árboles de decisión.

Los árboles de decisión discutidos en la Sección 8.1 sufren de **alta varianza**.

Esto significa que, si dividimos los datos de entrenamiento en dos partes al azar, y ajustamos un árbol de decisión a ambas mitades, los resultados que obtengamos podrían ser bastante diferentes. En contraste, un procedimiento con **baja varianza** arrojará

resultados similares si se aplica repetidamente a conjuntos distintos de datos; la regresión lineal tiende a tener baja varianza, si la relación de $\boldsymbol{n}$ a $\boldsymbol{p}$ es moderadamente grande. La **agregación de Bootstrap**, o **bagging**, es un procedimiento de propósito general para reducir la varianza de un método de aprendizaje estadístico; lo presentamos aquí porque es particularmente útil y se usa con frecuencia en el contexto de los árboles de decisión.

Recuerde que dado un conjunto de $\boldsymbol{n}$ observaciones independientes $\boldsymbol{Z_1, \ldots, Z_n}$, cada uno con varianza $\boldsymbol{\sigma^2}$, la varianza de la media $\boldsymbol{\overline{Z}}$ of de las observaciones viene dada por $\frac{\boldsymbol{\sigma^2}}{\boldsymbol{n}}$. En otras palabras, **promediar un conjunto de observaciones reduce la varianza**. Por lo tanto, una forma natural de reducir la varianza y aumentar la precisión de predicción de un método de aprendizaje estadístico es tomar muchos conjuntos de entrenamiento de la población, construir un modelo de predicción separado usando cada conjunto de entrenamiento y promediar las predicciones resultantes. En otras palabras, podríamos calcular $\boldsymbol{\hat{f}^1(x), \hat{f}^2(x), \ldots, \hat{f}^B(x)}$ usando $\boldsymbol{B}$ conjuntos de entrenamiento separados, y los promedia para obtener un modelo de aprendizaje estadístico de baja varianza, dado por

$$\boldsymbol{\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x)}$$

Por supuesto, esto no es práctico porque generalmente no tenemos acceso a múltiples conjuntos de entrenamiento. En cambio, podemos arrancar, tomando muestras repetidas del conjunto de datos de entrenamiento (individual). En este enfoque, generamos *B* diferentes conjuntos de datos de entrenamiento bootstrapped. Luego, entrenamos nuestro método en el $\boldsymbol{b}$-eximo conjunto de entrenamiento bootstrapped para obtener $\boldsymbol{\hat{f}^{*b}(x)}$, y finalmente promediar todas las predicciones, para obtener

$$\boldsymbol{\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)}$$

Esto se llama bagging.

Si bien el bagging puede mejorar las predicciones para muchos métodos de regresión, es particularmente útil para los árboles de decisión. Para aplicar el bagging a los árboles de regresión, simplemente construimos $\boldsymbol{B}$ árboles de regresión usando $\boldsymbol{B}$ conjuntos de entrenamiento bootstrap, y promediando las predicciones resultantes. Estos árboles crecen en profundidad y no están podados. Por lo tanto, cada árbol individual tiene una gran varianza, pero un sesgo bajo. Promediar estos $\boldsymbol{B}$ árboles reduce la varianza. Se ha demostrado que el bagging brinda mejoras impresionantes en precisión combinando cientos o incluso miles de árboles en un solo procedimiento.

Hasta ahora, hemos descrito el procedimiento del bagging en el contexto de regresión, para predecir un resultado cuantitativo $\boldsymbol{Y}$. ¿Cómo se puede extender el bagging a un problema de clasificación donde $\boldsymbol{Y}$ es cualitativo? En esa situación, hay algunos

enfoques posibles, pero el más simple es el siguiente. Para una observación de prueba dada, podemos registrar la clase predicha por cada uno de los $\boldsymbol{B}$ árboles, y obtener una mayoría de votos: la predicción general es la clase más común entre las $\boldsymbol{B}$ predicciones.

La figura 8.8 muestra los resultados del bagging en árboles en los datos del corazón (**Heart**). La tasa de error de prueba se muestra como una función de $\boldsymbol{B}$, la cantidad de árboles construidos utilizando conjuntos de datos de entrenamiento bootstrapped. Vemos que la tasa de error de la prueba del bagging es un poco más baja en este caso que la tasa de error de prueba obtenida de un solo árbol. El número de $\boldsymbol{B}$ árboles no es un parámetro crítico en el bagging; usar un valor muy grande de $\boldsymbol{B}$ no dará lugar a un ajuste excesivo. En la práctica, utilizamos un valor de $\boldsymbol{B}$ lo suficientemente grande como para que el error se haya establecido. Usar $\boldsymbol{B = 100}$ es suficiente para lograr un buen rendimiento en este ejemplo.

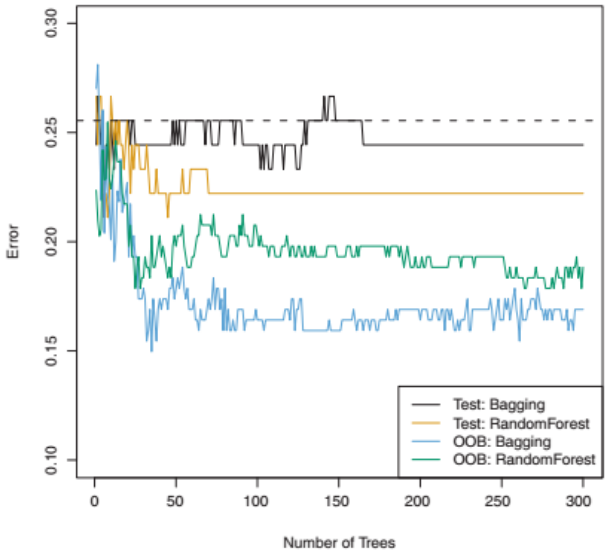

FIGURE 8.8. *Bagging and random forest results for the* `Heart` *data. The test error (black and orange) is shown as a function of* B*, the number of bootstrapped training sets used. Random forests were applied with* $m = \sqrt{p}$*. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which in this case is considerably lower.*

Out-of-Bag Error Estimate

Resulta que hay una manera muy directa de estimar el error de prueba de un modelo bagging, sin la necesidad de llevar a cabo la validación cruzada o el enfoque del conjunto de validación. Recuerde que la clave del bagging es que los árboles se ajustan repetidamente a subconjuntos de las observaciones (bootstrapped). Se puede demostrar que, en promedio, cada árbol bagging hace uso de alrededor de dos tercios de las observaciones. El tercio restante de las observaciones que no se usan para caber en un árbol dado se conoce out-of-bag (OOB). Podemos predecir la respuesta para la $\boldsymbol{i}$-ésima observación usando cada uno de los árboles en los que esa observación fue OOB. Esto producirá alrededor de $\boldsymbol{B/3}$ predicciones para la $\boldsymbol{i}$-ésima observación. Para obtener una

predicción única para la i-ésima observación, podemos promediar estas respuestas pronosticadas (si la regresión es el objetivo) o podemos tomar un voto mayoritario (si la meta es la clasificación). Esto lleva a una única predicción OOB para la i-ésima observación. De esta manera, se puede obtener una predicción OOB para cada una de las n observaciones, a partir de las cuales se puede calcular el MSE OOB general (para un problema de regresión) o el error de clasificación (para un problema de clasificación). El error OOB resultante es una estimación válida del error de prueba para el modelo bagging, ya que la respuesta para cada observación se predice usando solo los árboles que no se ajustaron usando esa observación. La figura 8.8 muestra el error OOB en los datos del corazón (Heart). Se puede demostrar que con B suficientemente grande, el error OOB es virtualmente equivalente al error de validación cruzada de dejar salir uno. El enfoque OOB para estimar el error de la prueba es particularmente conveniente cuando se realiza bagging en grandes conjuntos de datos para los cuales la validación cruzada sería computacionalmente pesada.

Medidas de Importancia Variable

Como ya hemos comentado, el bagging generalmente da como resultado una precisión mejorada sobre la predicción usando un solo árbol. Desafortunadamente, puede ser difícil interpretar el modelo resultante. Recuerde que una de las ventajas de los árboles de decisión es el diagrama atractivo y fácil de interpretar que resulta, como el que se muestra en la Figura 8.1. Sin embargo, cuando empaquetamos una gran cantidad de árboles, ya no es posible representar el procedimiento de aprendizaje estadístico resultante usando un solo árbol, y ya no está claro qué variables son más importantes para el procedimiento. Por lo tanto, el bagging mejora la precisión de predicción a expensas de la interpretabilidad.

Aunque la recolección de árboles en el bagging es mucho más difícil de interpretar que un solo árbol, se puede obtener un resumen general de la importancia de cada predictor usando RSS (para el bagging de árboles de regresión) o el índice de Gini (para el bagging de árboles de clasificación). En el caso de los árboles de regresión del bagging, podemos registrar la cantidad total que el RSS (8.1) se reduce debido a las divisiones sobre un predictor dado, promediado sobre todos los B árboles. Un valor grande indica un predictor importante. De manera similar, en el contexto de los árboles de clasificación del bagging, podemos sumar la cantidad total que el índice de Gini (8.6) se reduce por divisiones sobre un predictor dado, promediado sobre todos los B árboles.

En la Figura 8.9 se muestra una representación gráfica de las variables importantes en los datos corazón (Heart). Vemos la disminución media en el índice de Gini para cada variable, en relación con el más grande. Las variables con la mayor disminución media en el índice de Gini son Thal, Ca y ChestPain.

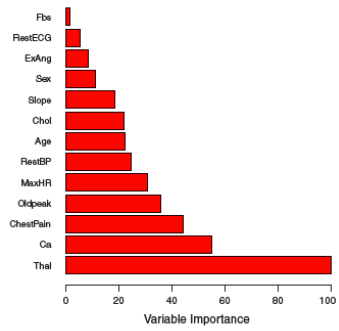

FIGURE 8.9. *A variable importance plot for the* Heart *data. Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.*

8.2.2 Random forests

Los bosques aleatorizados proporcionan una mejora sobre los árboles que utilizan el bagging por medio de una pequeña modificación que decora los árboles. Al igual que en el bagging, construimos varios árboles de decisión en muestras de entrenamiento bootstrapped. Pero al construir estos árboles de decisión, cada vez que se considera una división en un árbol, se elige una muestra aleatoria de m predictores como candidatos que dividen el conjunto completo de p predictores. La división solo permite usar uno de esos m predictores. Se toma una nueva muestra de m predictores en cada división, y típicamente elegimos $\boldsymbol{m \approx \sqrt{p}}$, es decir, el número de predictores considerados en cada división es aproximadamente igual a la raíz cuadrada del número total de predictores (4 de los 13 para los datos del Heart).

En otras palabras, al construir un bosque aleatorio, en cada división del árbol, el algoritmo ni siquiera tiene en cuenta la mayoría de los predictores disponibles. Esto puede parecer una locura, pero tiene un razonamiento ingenioso. Supongamos que hay un predictor muy fuerte en el conjunto de datos, junto con una serie de otros predictores moderadamente fuertes. Luego, en la colección de árboles en bagging, la mayoría o todos los árboles utilizarán este fuerte predictor en la división superior. En consecuencia, todos los árboles en el bagging se verán bastante similares entre sí. Por lo tanto, las predicciones de los árboles en el bagging estarán altamente correlacionadas. Desafortunadamente, promediar muchas cantidades altamente correlacionadas no conduce a una reducción de la varianza tan grande como promediar muchas cantidades no correlacionadas. En particular, esto significa que el bagging no conducirá a una reducción sustancial de la varianza sobre un solo árbol en este entorno.

Los bosques aleatorios superan este problema forzando a cada división a considerar solo un subconjunto de predictores. Por lo tanto, en promedio $\boldsymbol{(p-m)/p}$ de las divisiones ni siquiera considerarán el fuerte predictor, por lo que otros predictores tendrán más posibilidades. Podemos pensar que este proceso decora los árboles, haciendo que el promedio de los árboles resultantes sea menos variable y, por lo tanto, más confiable.

La principal diferencia entre el embolsado (bagging) y los bosques aleatorios es la elección del tamaño del subconjunto predictor $\boldsymbol{m}$. Por ejemplo, si se construye un bosque al azar usando $\boldsymbol{m = p}$, entonces esto equivale simplemente el bagging. En los

datos Heart, los bosques aleatorios que usan $\boldsymbol{m = \sqrt{p}}$ conducen a una reducción en el error de prueba y el error OOB sobre el bagging (Figura 8.8).

Usar un valor pequeño de $\boldsymbol{m}$ en la construcción de un bosque aleatorio generalmente será útil cuando tenemos un gran número de predictores correlacionados. Aplicamos bosques aleatorios a un conjunto de datos biológicos de alta dimensión que consistía en mediciones de expresión de 4,718 genes medidos en muestras de tejido de 349 pacientes. Existen alrededor de 20,000 genes en humanos, y los genes individuales tienen diferentes niveles de actividad o expresión, en particular células, tejidos y condiciones biológicas. En este conjunto de datos, cada una de las muestras de pacientes tiene una etiqueta cualitativa con 15 niveles diferentes: uno normal o uno de cada 14 tipos diferentes de cáncer. Nuestro objetivo fue utilizar bosques aleatorios para predecir el tipo de cáncer según los 500 genes que tienen la mayor varianza en el conjunto de entrenamiento.

Dividimos al azar las observaciones en un conjunto de entrenamiento y prueba, y aplicamos bosques aleatorios al conjunto de entrenamiento para tres valores diferentes del número de variables de división $\boldsymbol{m}$. Los resultados se muestran en la Figura 8.10. La tasa de error de un solo árbol es 45.7%, y la tasa nula es 75.4% .4 Vemos que usar 400 árboles es suficiente para dar un buen rendimiento, y que la elección $\boldsymbol{m = \sqrt{p}}$ dio una pequeña mejora en el error de prueba sobre el bagging ($\boldsymbol{m = p}$) en este ejemplo. Al igual que con el bagging, los bosques aleatorios no se sobreagotarán si aumentamos $\boldsymbol{B}$, por lo que en la práctica utilizamos un valor de $\boldsymbol{B}$ lo suficientemente grande para que la tasa de error se haya estabilizado.

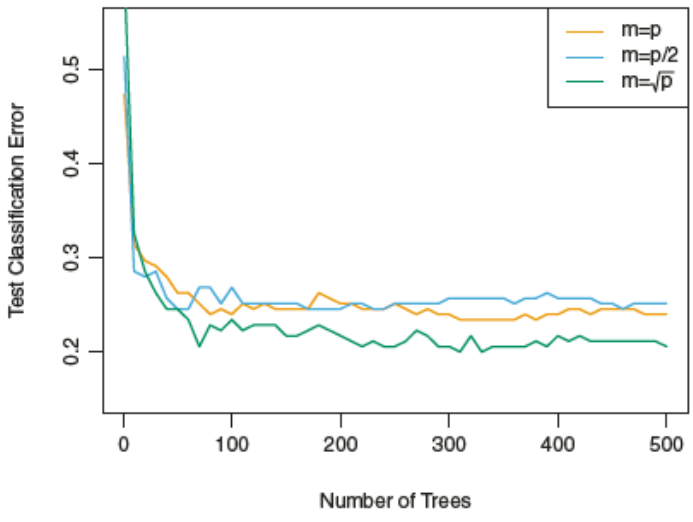

FIGURE 8.10. *Results from random forests for the 15-class gene expression data set with $p = 500$ predictors. The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of m, the number of predictors available for splitting at each interior tree node. Random forests ($m < p$) lead to a slight improvement over bagging ($m = p$). A single classification tree has an error rate of 45.7%.*

Chapter 11 [Data Mining and Predictive Analytics (Larose-2015)] Decision Trees

11.1 ¿QUÉ ES UN ÁRBOL DE DECISIÓN?

En este capítulo, continuamos nuestro examen de los métodos de clasificación para la minería de datos. Un método de clasificación atractivo implica la construcción de un árbol de decisión, una colección de nodos de decisión, conectados por ramas, que se extienden hacia abajo desde el nodo raíz hasta que terminan en nodos de hoja. Comenzando en el nodo raíz, que por convención se coloca en la parte superior del diagrama del árbol de decisiones, los atributos se prueban en los nodos de decisión, y cada resultado posible da como resultado una bifurcación. Cada rama luego conduce a otro nodo de decisión o a un nodo de hoja de terminación. La figura 11.1 proporciona un ejemplo de un árbol de decisión simple.

La variable objetivo para el árbol de decisión en la Figura 11.1 es el riesgo de crédito (credit risk), y los clientes potenciales se clasifican como riesgos de crédito buenos (good) o malos (bad). Las variables de predicción son ahorros (savings) (bajo (low), medio (médium) y alto (high)), activos (assets) (bajo (low) o no bajo (not low)) e ingresos (income) (≤ $30,000 o > $30,000). Aquí, el nodo raíz representa un nodo de decisión, que prueba si cada registro tiene un nivel de ahorro bajo (low savings), medio (médium savings) o alto (high savings) (según lo definido por el analista o experto de dominio). El conjunto de datos es particionado o se divide, de acuerdo con los valores de este atributo. Los registros con ahorros bajos se envían a través de la rama más a la izquierda (ahorros = bajo (savings = low)) a otro nodo de decisión. Los registros con grandes ahorros (high savings) se envían a través de la rama más a la derecha a un nodo de decisión diferente.

Los registros con ahorros medios (médium savings) se envían a través de la rama central directamente a un nodo hoja, lo que indica la terminación de esta rama. ¿Por qué un nodo hoja y no otro nodo de decisión? Porque, en el conjunto de datos (no se muestra), todos los registros con niveles de ahorro medios se han clasificado como buenos riesgos crediticios. No es necesario otro nodo de decisión, porque nuestro conocimiento de que el cliente tiene ahorros medios predice un buen crédito con un 100% de precisión en el conjunto de datos.

Para los clientes con bajos ahorros (low savings), el siguiente nodo de decisión prueba si el cliente tiene activos bajos (low assets). Aquellos con activos bajos (low assets) se clasifican como malos riesgos crediticios (bad risk); los otros se clasifican como buenos riesgos de crédito (good risk). Para los clientes con grandes ahorros (high savings), el siguiente nodo de decisión prueba si el cliente tiene un ingreso de menos ($income \leq$ 30.000). Los clientes con ingresos de $30,000 o menos se clasifican entonces como riesgos de mal crédito (bad risk), mientras que los otros se clasifican como buenos riesgos crediticios (good risk).

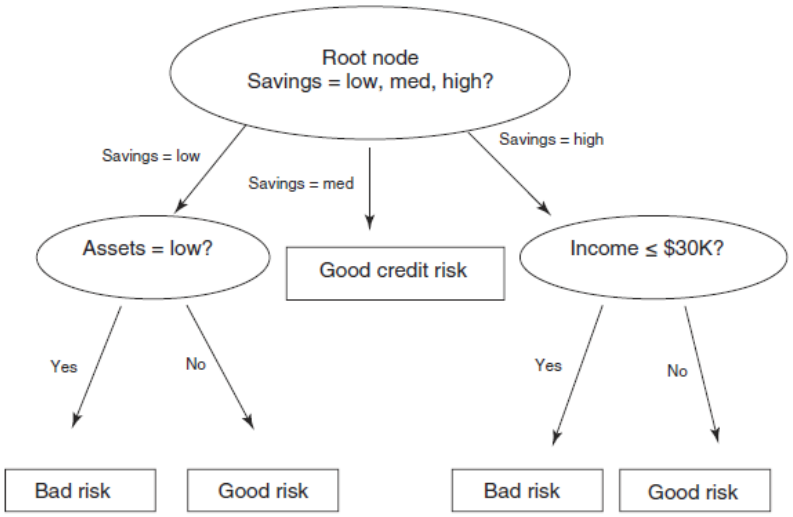

Figure 11.1 Simple decision tree.

Cuando no se pueden hacer más divisiones, el algoritmo del árbol de decisiones deja de crecer nuevos nodos. Por ejemplo, supongamos que todas las ramas terminan en nodos hoja "puros", donde la variable objetivo es única para los registros en ese nodo (por ejemplo, cada registro en el nodo hoja es un buen riesgo de crédito). Entonces no se necesitan más divisiones, por lo que no se cultivan más nodos.

Sin embargo, hay instancias en las que un nodo particular contiene atributos "diversos" (con valores no únicos para el atributo de destino) y, sin embargo, el árbol de decisiones no puede hacer una división. Por ejemplo, supongamos que consideramos los registros de la Figura 11.1 con grandes ahorros y bajos ingresos (≤ $30,000). Supongamos que hay cinco registros con estos valores, todos los cuales también tienen activos bajos. Finalmente, supongamos que tres de estos cinco clientes han sido clasificados como malos riesgos de crédito y dos como buenos riesgos de crédito, como se muestra en la Tabla 11.1. En el mundo real, a menudo se encuentran situaciones como esta, con valores variados para la variable de respuesta, incluso para exactamente los mismos valores para las variables de predicción.

Aquí, como todos los clientes tienen los mismos valores de predicción, no hay forma posible de dividir los registros de acuerdo con las variables de predicción que conducirán a un nodo hoja puro. Por lo tanto, dichos nodos se convierten en diversos nodos de hoja, con valores mixtos para el atributo de destino. En este caso, el árbol de decisiones puede informar que la clasificación para tales clientes es "mala", con un 60% de confianza, según lo determinado por las tres quintas partes de los clientes en este nodo que son malos riesgos crediticios. Tenga en cuenta que no todos los atributos se prueban para todos los registros. Los clientes con bajos ahorros y activos bajos, por ejemplo, no se prueban con respecto a los ingresos en este ejemplo.

TABLE 11.1 Sample of records that cannot lead to pure leaf node

Customer	Savings	Assets	Income	Credit Risk
004	High	Low	≤$30,000	Good
009	High	Low	≤$30,000	Good
027	High	Low	≤$30,000	Bad
031	High	Low	≤$30,000	Bad
104	High	Low	≤$30,000	Bad

11.2 REQUISITOS PARA EL USO DE ÁRBOLES DE DECISIÓN

Se deben cumplir los siguientes requisitos antes de que se apliquen los algoritmos del árbol de decisión:

- Los algoritmos del árbol de decisión representan el aprendizaje supervisado y, como tal, requieren variables de destino preclasificadas. Se debe suministrar un conjunto de datos de entrenamiento, que proporciona al algoritmo los valores de la variable objetivo.
- Este conjunto de datos de entrenamiento debe ser rico y variado, proporcionando al algoritmo una sección transversal saludable de los tipos de registros para los cuales la clasificación puede ser necesaria en el futuro. Los árboles de decisión aprenden con el ejemplo, y si faltan sistemáticamente ejemplos para un subconjunto definible de registros, la clasificación y predicción para este subconjunto será problemática o imposible.
- Las clases de atributos de destino deben ser discretas. Es decir, uno no puede aplicar el análisis del árbol de decisión a una variable objetivo continua. Más bien, la variable objetivo debe tomar valores que están claramente demarcados como pertenecientes a una clase particular o no pertenecientes.

¿Por qué, en el ejemplo anterior, el árbol de decisión eligió el atributo de ahorro para la división del nodo raíz? ¿Por qué no eligió activos o ingresos en su lugar? Los árboles de decisión buscan crear un conjunto de nodos de hoja lo más "puros" posible; es decir, donde cada uno de los registros en un nodo hoja particular tiene la misma clasificación. De esta forma, el árbol de decisiones puede proporcionar asignaciones de clasificación con la mayor medida de confianza disponible.

Sin embargo, ¿cómo se mide la uniformidad o, a la inversa, ¿cómo se mide la heterogeneidad? Examinaremos dos de los muchos métodos para medir la pureza del nodo de la hoja, que conducen a los siguientes dos algoritmos principales para construir árboles de decisión:

- Algoritmo de clasificación y árboles de regresión (CART);
- Algoritmo C4.5.

11.3 CLASIFICACIÓN Y REGRESIÓN DE ÁRBOLES

El método CART fue sugerido por Breiman et al.1 en 1984. Los árboles de decisión producidos por CART son estrictamente binarios, que contienen exactamente dos ramas para cada nodo de decisión.

CART particiona recursivamente los registros en el conjunto de datos de entrenamiento en subconjuntos de registros con valores similares para el atributo de destino. El algoritmo CART cultiva el árbol realizando para cada nodo de decisión, una búsqueda exhaustiva de todas las variables disponibles y todos los posibles valores de división, seleccionando la división óptima de acuerdo con los siguientes criterios (de Kennedy et al.2).

Deje $\boldsymbol{\Phi(s|t)}$ ser una medida de la "bondad" de una segmentación candidata s en el nodo t, donde

$$\boldsymbol{\Phi(s|t) = 2P_L P_R \sum_{j=1}^{\#Clases} |P(j|t_L) - P(j|t_R)|} \qquad (\mathbf{11.1})$$

Y donde

$\boldsymbol{t_L}$ = nodo hijo izquierdo del nodo $\boldsymbol{t}$

$\boldsymbol{t_R}$ = nodo secundario derecho del nodo $\boldsymbol{t}$

$$\boldsymbol{P_L = \frac{cantidad\ de\ registros\ en\ t_L}{número\ de\ registros\ en\ el\ conjunto\ de\ entrenamiento}}$$

$$\boldsymbol{P_R = \frac{cantidad\ de\ registros\ en\ t_R}{número\ de\ registros\ en\ el\ conjunto\ de\ entrenamiento}}$$

$$\boldsymbol{P(j|t_L) = \frac{número\ de\ registros\ de\ clase\ j\ en\ t_L}{número\ de\ registros\ en\ t}}$$

$$\boldsymbol{P(j|t_R) = \frac{número\ de\ registros\ de\ clase\ j\ en\ t_R}{número\ de\ registros\ en\ t}}$$

Entonces, la división óptima es la que divide maximiza esta medida $\boldsymbol{\Phi(s|t)}$ sobre todas las divisiones posibles en el nodo $\boldsymbol{t}$.

Veamos un ejemplo. Supongamos que tenemos el conjunto de datos de capacitación que se muestra en la Tabla 11.2 y estamos interesados en usar CART para construir un árbol de decisión para predecir si un cliente en particular debe clasificarse como un riesgo de crédito bueno o malo.

TABLE 11.2 Training set of records for classifying credit risk

Customer	Savings	Assets	Income ($1000s)	Credit Risk
1	Medium	High	75	Good
2	Low	Low	50	Bad
3	High	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
6	High	High	25	Good
7	Low	Low	25	Bad
8	Medium	Medium	75	Good

En este pequeño ejemplo, los ocho registros de entrenamiento ingresan al nodo raíz. Como CART está restringido a divisiones binarias, el candidato se divide en que el algoritmo CART evaluaría la partición inicial en el nodo raíz y se muestran en la Tabla 11.3. Aunque el ingreso es una variable continua, CART aún puede identificar una lista finita de posibles divisiones basadas en el número de valores diferentes que la variable realmente toma en el conjunto de datos. Alternativamente, el analista puede elegir categorizar la variable continua en un número menor de clases.

TABLE 11.3 Candidate splits for t = root node

Candidate Split	Left Child Node, t_L	Right Child Node, t_R
1	*Savings* = *low*	*Savings* ∈ {*medium, high*}
2	*Savings* = *medium*	*Savings* ∈ {*low, high*}
3	*Savings* = *high*	*Savings* ∈ {*low, medium*}
4	*Assets* = *low*	*Assets* ∈ {*medium, high*}
5	*Assets* = *medium*	*Assets* ∈ {*low, high*}
6	*Assets* = *high*	*Assets* ∈ {*low, medium*}
7	*Income* ≤ $25,000	*Income* > $25,000
8	*Income* ≤ $50,000	*Income* > $50,000
9	*Income* ≤ $75,000	*Income* > $75,000

Para cada división candidata, examinemos los valores de los diversos componentes de la medida de optimalidad $\boldsymbol{\Phi(s|t)}$ en la Tabla 11.4. Usando estos valores observados, podemos investigar el comportamiento de la medida de optimalidad en diversas condiciones. ¿Por ejemplo, cuándo es $\boldsymbol{\Phi(s|t)}$ grande? Vemos que **Φ(s|t)** es grande cuando sus dos componentes principales son grandes: $\boldsymbol{2P_L P_R}$ y $\sum_{j=1}^{\#Clases} |\boldsymbol{P(j|t_L) - P(j|t_R)}|$.

Considere $\boldsymbol{Q(s|t) = \sum_{j=1}^{\#Clases} |P(j|t_L) - P(j|t_R)|}$. ¿Cuándo es grande el componente $\boldsymbol{Q(s|t)}$? $\boldsymbol{Q(s|t)}$ es grande cuando la distancia entre $\boldsymbol{P(j|t_L)}$ y $\boldsymbol{P(j|t_L)}$ se maximiza en cada clase (valor de la variable objetivo). En otras palabras, este componente se maximiza cuando las proporciones de registros en los nodos secundarios para cada valor particular de la variable objetivo son lo más diferentes posibles. El valor máximo por lo tanto ocurriría cuando para cada clase los nodos secundarios son completamente uniformes (puros). El valor máximo teórico para $\boldsymbol{Q(s|t)}$ es $\boldsymbol{k}$, donde $\boldsymbol{k}$ es el número de clases para la variable objetivo. Como nuestro riesgo de crédito variable de salida toma dos valores, bueno y malo, $\boldsymbol{k = 2}$ es el máximo para este componente.

El componente $\boldsymbol{2P_L P_R}$ se maximiza cuando $\boldsymbol{P_L}$ y $\boldsymbol{P_R}$ son grandes, lo que ocurre cuando las proporciones de registros en los nodos secundarios izquierdo y derecho son iguales. Por lo tanto, $\boldsymbol{\Phi(s|t)}$ tenderá a favorecer divisiones equilibradas que dividen los datos en nodos secundarios que contienen aproximadamente el mismo número de registros. Por lo tanto, la medida de optimalidad $\boldsymbol{\Phi(s|t)}$ prefiere divisiones que proporcionarán nodos secundarios que son homogéneos para todas las clases y tienen aproximadamente el mismo número de registros. El máximo teórico para $\boldsymbol{2P_L P_R}$ es 2 (0.5) (0.5) = 0.5.

En este ejemplo, solo el candidato dividido 5 tiene un valor observado para $\boldsymbol{2P_L P_R}$ que alcanza el máximo teórico para esta estadística, 0.5, porque los registros están divididos por igual en dos grupos de cuatro. El máximo teórico para $\boldsymbol{Q(s|t)}$ se obtiene solo cuando cada nodo secundario resultante es puro y, por lo tanto, no se logra para este conjunto de datos.

TABLE 11.4 Values of the components of the optimality measure $\Phi(s|t)$ for each candidate split, for the root node (best performance highlighted)

Split	P_L	P_R	$P(j\|t_L)$	$P(j\|t_R)$	$2P_L P_R$	$Q(s\|t)$	$\Phi(s\|t)$
1	0.375	0.625	G: 0.333 B: 0.667	G: 0.8 B: 0.2	0.46875	0.934	0.4378
2	0.375	0.625	G: 1 B: 0	G: 0.4 B: 0.6	0.46875	1.2	0.5625
3	0.25	0.75	G: 0.5 B: 0.5	G: 0.667 B: 0.333	0.375	0.334	0.1253
4	0.25	0.75	G: 0 B: 1	G: 0.833 B: 0.167	0.375	1.667	**0.6248**
5	0.5	0.5	G: 0.75 B: 0.25	G: 0.5 B: 0.5	0.5	0.5	0.25
6	0.25	0.75	G: 1 B: 0	G: 0.5 B: 0.5	0.375	1	0.375
7	0.375	0.625	G: 0.333 B: 0.667	G: 0.8 B: 0.2	0.46875	0.934	0.4378
8	0.625	0.375	G: 0.4 B: 0.6	G: 1 B: 0	0.46875	1.2	0.5625
9	0.875	0.125	G: 0.571 B: 0.429	G: 1 B: 0	0.21875	0.858	0.1877

El máximo valor observado para $\boldsymbol{\Phi(s|t)}$ entre las divisiones candidatas se obtiene por la división 4, con $\boldsymbol{\Phi(s|t) = 0.6248}$. Por lo tanto, CART elige hacer la partición inicial del conjunto de datos usando la división candidata 4, activos = bajo versus activos ∈ {medio, alto}, como se muestra en la Figura 11.2.

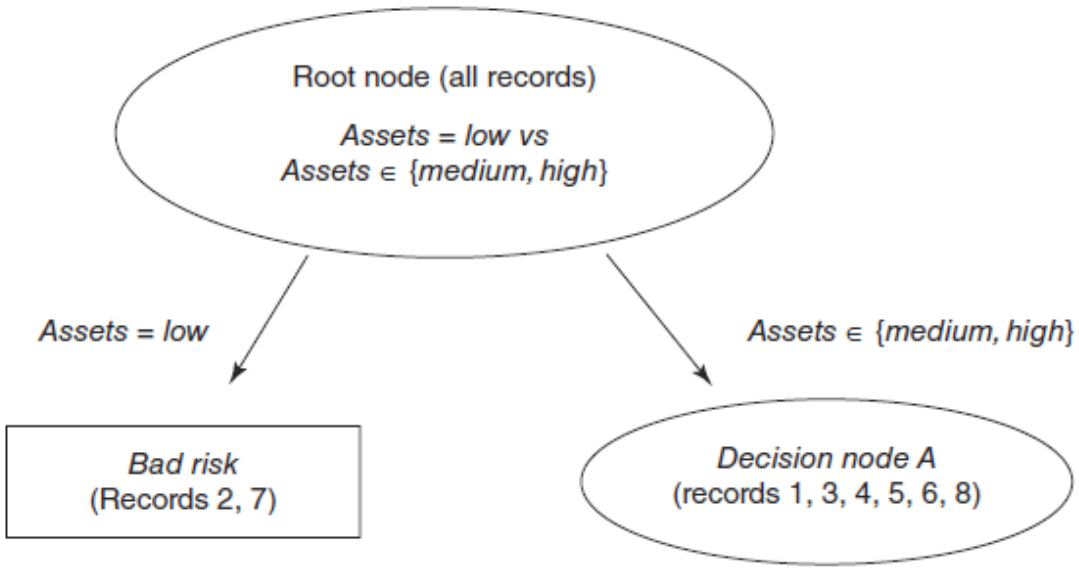

Figure 11.2 CART decision tree after initial split.

El nodo secundario izquierdo resulta ser un nodo de hoja terminal, porque los dos registros que se pasaron a este nodo tenían un riesgo de crédito incorrecto. El nodo hijo derecho, sin embargo, es diverso y requiere una mayor partición.

Nuevamente compilamos una tabla de las divisiones candidatas (todas están disponibles excepto la división 4), junto con los valores para la medida de optimalidad (Tabla 11.5). Aquí dos divisiones candidatas (3 y 7) comparten el valor más alto para $\boldsymbol{\Phi(s|t)}$, 0.4444. Seleccionamos arbitrariamente la primera división encontrada, división 3, ahorro = alto versus ahorro ∈ {bajo, medio}, para el nodo de decisión A, con el árbol resultante mostrado en la Figura 11.3.

TABLE 11.5 Values of the components of the optimality measure $\Phi(s|t)$ for each candidate split, for decision node *A* (best performance highlighted)

Split	P_L	P_R	$P(j\|t_L)$	$P(j\|t_R)$	$2P_LP_R$	$Q(s\|t)$	$\Phi(s\|t)$
1	0.167	0.833	G: 1 B: 0	G: 0.8 B: 0.2	0.2782	0.4	0.1113
2	0.5	0.5	G: 1 B: 0	G: 0.667 B: 0.333	0.5	0.6666	0.3333
3	0.333	0.667	G: 0.5 B: 0.5	G: 1 B: 0	0.4444	1	**0.4444**
5	0.667	0.333	G: 0.75 B: 0.25	G: 1 B: 0	0.4444	0.5	0.2222
6	0.333	0.667	G: 1 B: 0	G: 0.75 B: 0.25	0.4444	0.5	0.2222
7	0.333	0.667	G: 0.5 B: 0.5	G: 1 B: 0	0.4444	1	**0.4444**
8	0.5	0.5	G: 0.667 B: 0.333	G: 1 B: 0	0.5	0.6666	0.3333
9	0.833	0.167	G: 0.8 B: 0.2	G: 1 B: 0	0.2782	0.4	0.1113

Como el nodo de decisión **B** es diverso, de nuevo necesitamos buscar la división óptima. Solo quedan dos registros en este nodo de decisión, cada uno con el mismo valor para ahorro (alto (high)) e ingreso (income) (25). Por lo tanto, la única división posible es activos (assets) = alto (high) versus activos (assets) = medio (médium), que nos proporciona la forma final del árbol de decisión CART para este ejemplo, en la figura 11.4. Compare la Figura 11.4 con la Figura 11.5, el árbol de decisión generado por el algoritmo CART de Modeler.

Dejemos de lado este ejemplo ahora, y pensemos cómo funcionaría CART en un conjunto de datos arbitrario. En general, CART procedería recursivamente a visitar cada nodo de decisión restante y aplicaría el procedimiento anterior para encontrar la división óptima en cada nodo. Eventualmente, no quedan nodos de decisión, y el "árbol completo" ha crecido. Sin embargo, como hemos visto en la Tabla 11.1, no todos los nodos de hoja son necesariamente homogéneos, lo que conduce a un cierto nivel de error de clasificación.

Por ejemplo, supongamos que, como no podemos seguir dividiendo los registros en la Tabla 11.1, clasificamos los registros contenidos en este nodo hoja como un riesgo de crédito malo. Entonces la probabilidad de que un registro elegido al azar de este nodo hoja se clasifique correctamente es 0,6, porque tres de los cinco registros (60%) en realidad se clasifican como riesgos de crédito malo. Por lo tanto, nuestra tasa de error de clasificación para esta hoja en particular sería 0,4 o 40%, porque dos de los cinco registros en realidad se clasifican como buenos riesgos crediticios. CART calcularía entonces la tasa de error para todo el árbol de decisión para ser el promedio ponderado de las tasas de error de hojas individuales, con los pesos iguales a la proporción de registros en cada hoja.

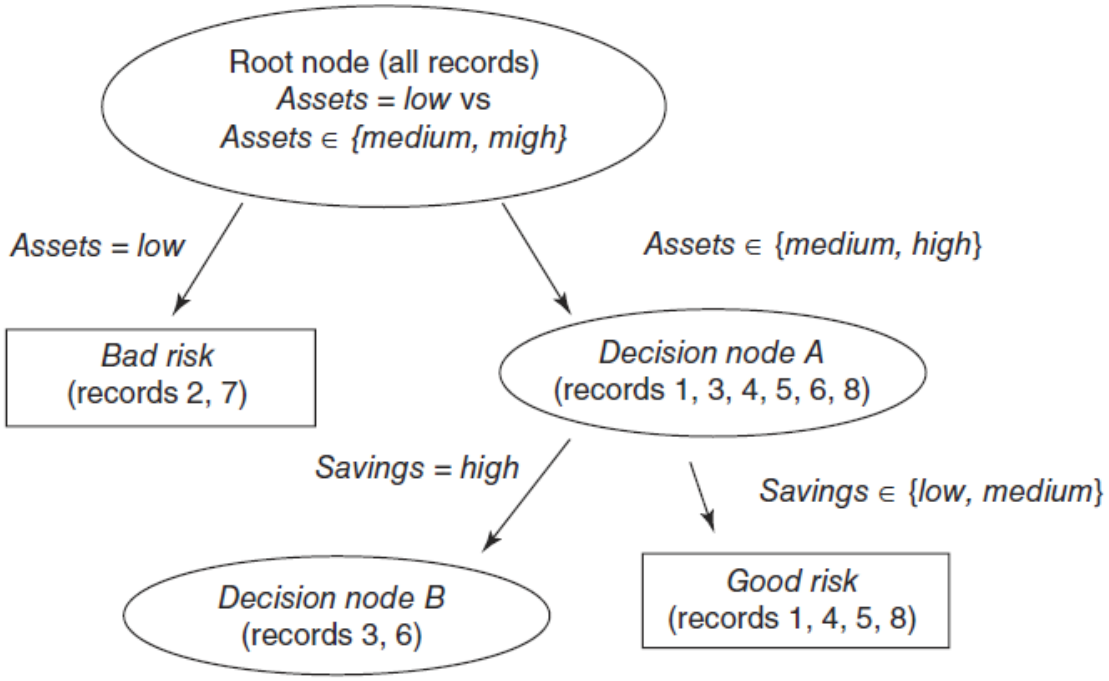

Figure 11.3 CART decision tree after decision node *A* split.

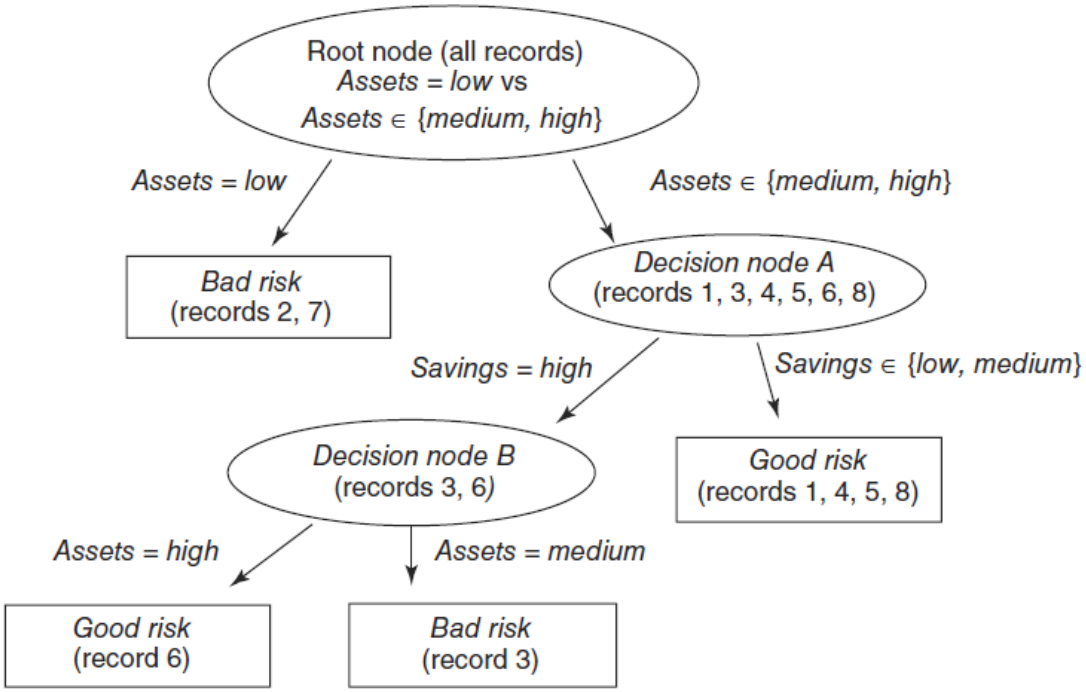

Figure 11.4 CART decision tree, fully grown form.

Para evitar la memorización del conjunto de entrenamiento, el algoritmo CART necesita comenzar a podar nodos y ramas que de lo contrario reducirían la generalización de los resultados de clasificación. A pesar de que el árbol completamente crecido tiene la tasa de error más baja en el conjunto de entrenamiento, el modelo resultante puede ser demasiado complejo, lo que resulta en un ajuste excesivo. A medida que crece cada nodo de decisión, el subconjunto de registros disponibles para el análisis se vuelve más pequeño y menos representativo de la población general. Podar el árbol aumentará la capacidad de generalización de los resultados. La forma en que el algoritmo CART lleva a cabo la poda de árboles se explica en Breiman et al. [pag. 66]. Básicamente, se encuentra una tasa de error general ajustada que penaliza al árbol de decisiones por tener demasiados nodos de hojas y, por lo tanto, demasiada complejidad.

11.4 ALGORITMO C4.5

El algoritmo C4.5 es la extensión de Quinlan de su propio algoritmo iterativo dichotomizer 3 (ID3) para generar árboles de decisión. Al igual que con CART, el algoritmo C4.5 visita recursivamente cada nodo de decisión, seleccionando la división óptima, hasta que no se dividan más posible. Sin embargo, existen las siguientes diferencias interesantes entre CART y C4.5:

- A diferencia de CART, el algoritmo C4.5 no está restringido a divisiones binarias. Mientras que CART siempre produce un árbol binario, C4.5 produce un árbol de forma más variable.
- Para los atributos categóricos, C4.5 de forma predeterminada produce una rama separada para cada valor del atributo categórico. Esto puede resultar en más

"espesor" de lo deseado, porque algunos valores pueden tener baja frecuencia o pueden estar asociados de forma natural con otros valores.

- El método C4.5 para medir la homogeneidad del nodo es bastante diferente del método CART y se examina en detalle a continuación.

El algoritmo C4.5 utiliza el concepto de ganancia de información o reducción de entropía para seleccionar la división óptima. Supongamos que tenemos una variable $\boldsymbol{X}$ cuyos $\boldsymbol{k}$ valores posibles tienen probabilidades $\boldsymbol{p_1, p_2, \ldots, p_k}$. ¿Cuál es el número más pequeño de bits, en promedio por símbolo, necesarios para transmitir un flujo de símbolos que representa los valores de X observados? La respuesta se llama entropía de $\boldsymbol{X}$ y se define como

$$\boldsymbol{H(X) = -\sum_j p_j \log_2(p_j)}$$

¿De dónde viene esta fórmula para la entropía? Para un evento con probabilidad p, la cantidad promedio de información en bits requerida para transmitir el resultado es $\boldsymbol{-\log_2(p)}$. Por ejemplo, el resultado de un lanzamiento de moneda, con una probabilidad de 0.5, se puede transmitir usando $\boldsymbol{\log_2(0.5) = 1}$ bit, que es un 0 o 1, dependiendo del resultado del lanzamiento. Para variables con varios resultados, simplemente usamos una suma ponderada de $\boldsymbol{\log_2(p_j)\, s}$, con pesos iguales a las probabilidades de resultado, resultando en la fórmula

$$\boldsymbol{H(X) = -\sum_j p_j \log_2(p_j)}$$

C4.5 usa este concepto de entropía de la siguiente manera. Supongamos que tenemos un candidato dividido $\boldsymbol{S}$, que divide el conjunto de datos de entrenamiento T en varios subconjuntos, $\boldsymbol{T_1, T_2, \ldots, T_k}$. El requisito de información promedio se puede calcular como la suma ponderada de las entropías para los subconjuntos individuales, de la siguiente manera:

$$\boldsymbol{H_S(T) = -\sum_{i=1}^{k} p_i\, H_S(T_i)} \qquad \boldsymbol{(11.2)}$$

donde $\boldsymbol{p_i}$ representa la proporción de registros en el subconjunto $\boldsymbol{i}$. Entonces podemos definir que nuestra ganancia de información sea ganancia $\boldsymbol{gain(S) = H(T) - H_S(T)}$, es decir, el aumento en la información producida al particionar los datos de entrenamiento $\boldsymbol{T}$ de acuerdo con esta segmentación candidata $\boldsymbol{S}$. En cada nodo de decisión, C4.5 elige la división óptima para ser la división que tiene la mayor ganancia de información, ganancia $\boldsymbol{gain(S)}$.

Para ilustrar el algoritmo C4.5 en el trabajo, volvamos al conjunto de datos en la Tabla 11.2 y apliquemos el algoritmo C4.5 para construir un árbol de decisión para clasificar el riesgo de crédito, tal como lo hicimos anteriormente usando CART. Una vez más,

estamos en el nodo raíz y estamos considerando todas las divisiones posibles usando todos los datos (Tabla 11.6).

Ahora, debido a que cinco de los ocho registros se clasifican como un buen riesgo crediticio, y los tres registros restantes se clasifican como riesgo de crédito malo, la entropía antes de la división es

$$\boldsymbol{H(T) = -\sum_j p_j \log_2(p_j) = -\frac{5}{8}\log_2\left(\frac{5}{8}\right) - \frac{3}{8}\log_2\left(\frac{3}{8}\right) = 0.9544}$$

Compararemos la entropía de cada candidato dividido con esta $\boldsymbol{H(T) = 0.9544}$, para ver qué división da como resultado la mayor reducción de entropía (o ganancia de información).

Para el candidato dividido 1 ahorros (savings), dos de los registros tienen grandes ahorros (high savings), tres de los registros tienen ahorros medios (médium savings), y tres de los registros tienen ahorros bajos (low savings), entonces tenemos $\boldsymbol{p_{high} = \frac{2}{8}}$, $\boldsymbol{p_{medium} = \frac{3}{8}}$, $\boldsymbol{p_{low} = \frac{3}{8}}$. De los registros con grandes ahorros, uno es un buen riesgo de crédito y uno es malo, dando una probabilidad de 0.5 de elegir el registro con un buen riesgo crediticio. Por lo tanto, la entropía para grandes ahorros (high savings) es $\boldsymbol{-\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1}$, que es similar al lanzamiento de una moneda justa. Los tres registros con ahorros medios (médium savings) son buenos riesgos de crédito, por lo que la entropía para el medio (médium) es $\boldsymbol{-\frac{3}{3}\log_2\left(\frac{3}{3}\right) - \frac{0}{3}\log_2\left(\frac{0}{3}\right) = 0}$, donde por convención definimos $\boldsymbol{\log_2(0) = 0}$.

En las aplicaciones de ingeniería, la información es análoga a la señal, y la entropía es análoga al ruido, por lo que tiene sentido que la entropía para ahorros medios sea cero, porque la señal es clara como el cristal y no hay ruido: si el cliente tiene ahorros medios, o ella es un buen riesgo de crédito, con 100% de confianza. La cantidad de información requerida para transmitir la calificación crediticia de estos clientes es cero, siempre que sepamos que tienen ahorros medios.

TABLE 11.6 Candidate splits at root node for C4.5 algorithm

Candidate Split	Child Nodes		
1	*Savings = low*	*Savings = medium*	*Savings = high*
2	*Assets = low*	*Assets = medium*	*Assets = high*
3	*Income* ≤ $25,000		*Income* > $25,000
4	*Income* ≤ $50,000		*Income* > $50,000
5	*Income* ≤ $75,000		*Income* > $75,000

Uno de los registros con bajos ahorros (low savings) es un buen riesgo de crédito, y dos registros con bajos ahorros (low savings) son malos riesgos crediticios, lo que nos da nuestra entropía para un bajo riesgo de crédito como $-\frac{1}{3}\mathbf{log_2}\left(\frac{1}{3}\right)-\frac{2}{3}\mathbf{log_2}\left(\frac{2}{3}\right)=\mathbf{0.9183}$. Combinamos las entropías de estos tres subconjuntos, usando la ecuación (11.2) y las proporciones de los subconjuntos $\boldsymbol{p_i}$, de modo que $\boldsymbol{H_{savings}(T)=\frac{2}{8}(1)+\frac{3}{8}(0)+\frac{3}{8}(0.9183)=0.5944}$. Luego, la ganancia de información representada por la división en el atributo de ahorro se calcula como $\boldsymbol{H(T)-H_{savings}(T)=0.9544-0.5944=0.36}$ bits.

¿Cómo debemos interpretar estas medidas? Primero, $\boldsymbol{H(T)=0.9544}$ significa que, en promedio, uno necesitaría 0.9544 bits (0's o 1's) para transmitir el riesgo crediticio de los ocho clientes en el conjunto de datos. Ahora, $\boldsymbol{H_{savings}(\mathbf{T})=0.5944}$ significa que la partición de los clientes en tres subconjuntos ha reducido el requisito de bits promedio para transmitir el estado de riesgo de crédito de los clientes a 0.5944 bits. La baja entropía es buena. Esta reducción de entropía puede verse como ganancia de información, de modo que hemos ganado en promedio $\boldsymbol{H(T)-H_{savings}(T)=0.9544-0.5944=0.36}$ bits de información al usar la partición de ahorro (savings). Compararemos esto con la información obtenida por las otras divisiones candidatas, y elegiremos la división con la mayor ganancia de información como la división óptima para el nodo raíz.

Para el candidato dividido 2 activos (assets), dos de los registros tienen activos altos (high assets), cuatro de los registros tienen activos medios (médium assets) y dos de los registros tienen activos bajos (low assets), por lo que tenemos $\boldsymbol{p_{high}=\frac{2}{8}}$, $\boldsymbol{p_{medium}=\frac{4}{8}}$, $\boldsymbol{p_{low}=\frac{2}{8}}$. Ambos registros con altos activos se clasifican como buenos riesgos crediticios, lo que significa que la entropía para activos altos será cero, tal como lo fue para ahorros medios superiores.

Tres de los registros con activos medianos son buenos riesgos de crédito y uno es un riesgo de crédito malo, que nos da entropía $-\frac{3}{4}\mathbf{log_2}\left(\frac{3}{4}\right)-\frac{1}{4}\mathbf{log_2}\left(\frac{1}{4}\right)=\mathbf{0.8113}$. Y ambos registros con bajos activos son malos riesgos de crédito, lo que resulta en una entropía de activos bajos que equivale a cero. Combinando las entropías de estos tres subconjuntos, usando la ecuación (11.2) y las proporciones de los subconjuntos $\boldsymbol{p_i}$, tenemos $\boldsymbol{H_{assets}=\frac{2}{8}(0)+\frac{4}{8}(0.8113)+\frac{2}{8}(0)=0.4057}$. La entropía para la división de activos es menor que la entropía (0.5944) para la división de ahorros, lo que indica que la división de activos contiene menos ruido y es preferible a la división de ahorros. Esto se mide directamente usando la ganancia de información, como sigue: $\boldsymbol{H(T)-H_{assets}(T)=0.9544-0.4057=0.5487}$ bits. Esta ganancia de información de 0.5487 bits es mayor que la de la división de ahorros de 0.36 bits, verificando que la división de activos es preferible.

Mientras que C4.5 divide las variables categóricas de forma diferente a CART, las particiones para las variables numéricas son similares. Aquí tenemos cuatro valores observados para el ingreso (income): 25,000, 50,000, 75,000 y 100,000, que nos proporcionan tres umbrales para las particiones, como se muestra en la Tabla 11.6. Para el candidato dividido 3 de la Tabla 11.6, ingreso $(income \leq \$\,25{,}000\ versus\ income > \$25{,}000)$, tres de los registros tienen ingresos $(income \leq \$\,25{,}000)$, con los otros cinco registros con ingresos $(income > \$\,25{,}000)$, que nos da $p_{income \leq \$25{,}000} = \frac{3}{8},\ p_{income > \$25{,}000} = \frac{5}{8}$. De los registros con ingresos $(income \leq \$\,25{,}000)$, uno es un buen riesgo de crédito y dos son malos, lo que nos da la entropía de ingresos $(income \leq \$\,25{,}000)$ como $-\frac{1}{3}\log_2\left(\frac{1}{3}\right) - \frac{2}{3}\log_2\left(\frac{2}{3}\right) = 0.9483$. Cuatro de los cinco registros con un ingreso> $ 25,000 son buenos riesgos de crédito, por lo que la entropía para el ingreso $(income > \$\,25{,}000)$ es $-\frac{4}{5}\log_2\left(\frac{4}{5}\right) - \frac{1}{5}\log_2\left(\frac{1}{5}\right) = 0.7219$. Combinando, encontramos que la entropía para el candidato dividido 3 es $H_{income \leq \$25{,}000}(T) = \left(\frac{3}{8}\right)(0.9183) + \left(\frac{5}{8}\right)(0.7219) = 0.7956$. Entonces la ganancia de información para esta división es $H(T) - H_{income \leq \$25{,}000}(T) = 0.9544 - 0.7956 = 0.1588$ bits, que es nuestra opción más pobre hasta el momento.

Para el candidato dividido 4, $(income \leq \$\,50{,}000\ versus\ income > \$50{,}000)$, dos de los cinco registros con ingresos $(income \leq \$50{,}000)$ son buenos riesgos de crédito, y tres son malos, mientras que los tres de los registros con ingresos $(income > \$50{,}000)$ son buenos riesgos crediticios. Esto nos da la entropía para el candidato dividido 4 como

$$H_{income \leq \$50{,}000}(T) = \frac{5}{8}\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{3}{8}\left(-\frac{3}{3}\log_2\frac{3}{3} - \frac{0}{3}\log_2\frac{0}{3}\right) = 0.6069$$

La ganancia de información para esta división es, por lo tanto, $H(T) - H_{income \leq \$50{,}000}(T) = 0.9544 - 0.6069 = 0.3475$, que no es tan buena como para los activos. Finalmente, para la división del candidato 5, ingreso $(income \leq \$75{,}000\ versus\ income > \$75{,}000)$, cuatro de los siete registros con ingreso $(income \leq \$75{,}000)$ son buenos riesgos de crédito, y tres son malos, mientras que el único registro con ingresos $(income > \$75{,}000)$ es un buen riesgo de crédito. Por lo tanto, la entropía para el candidato dividido 4 es

$$H_{income \leq \$75{,}000}(T) = \frac{7}{8}\left(-\frac{4}{7}\log_2\frac{4}{7} - \frac{3}{7}\log_2\frac{3}{7}\right) + \frac{1}{8}\left(-\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1}\right) = 0.8621$$

La ganancia de información para esta división es $H(T) - H_{income \leq \$75{,}000}(T) = 0.9544 - 0.8621 = 0.0923$, haciendo que esta división sea la más pobre de las cinco divisiones candidatas.

La Tabla 11.7 resume la ganancia de información para cada candidato dividido en el nodo raíz. Candidato dividido 2, activos, tiene la ganancia de información más grande, y así se elige para la división inicial por el algoritmo C4.5. Tenga en cuenta que esta

elección para una división óptima concuerda con la partición preferida por CART, que se divide en activos = bajo versus activos = {medio, alto} (assets = low versus assets ={meium, high}). El árbol de decisión parcial resultante de la división inicial de C4.5 se muestra en la figura 11.6.

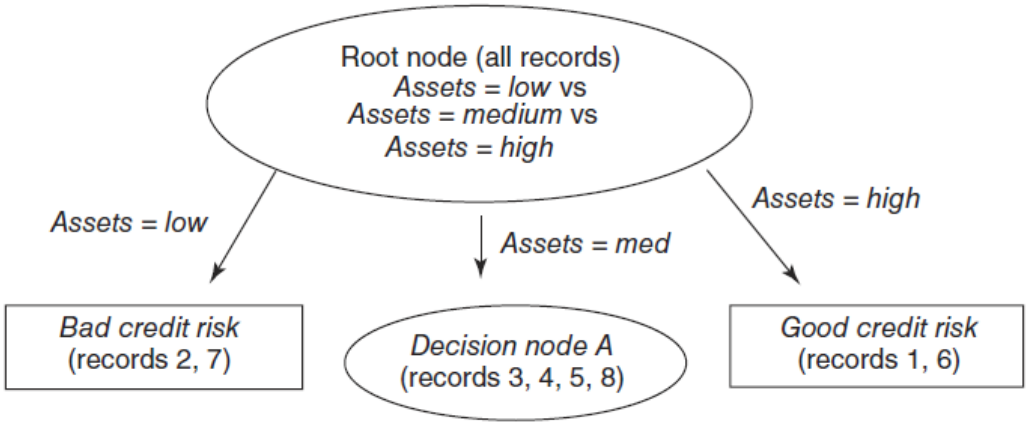

Figure 11.6 C4.5 concurs with CART in choosing *assets* for the initial partition.

TABLE 11.7 Information gain for each candidate split at the root node

Candidate Split	Child Nodes	Information Gain (Entropy Reduction)
1	*Savings* = *low* *Savings* = *medium* *Savings* = *high*	0.36 bits
2	*Assets* = *low* *Assets* = *medium* *Assets* = *high*	0.5487 bits
3	*Income* ≤ $25,000 *Income* > $25,000	0.1588 bits
4	*Income* ≤ $50,000 *Income* > $50,000	0.3475 bits
5	*Income* ≤ $75,000 *Income* > $75,000	0.0923 bits

La división inicial ha resultado en la creación de dos nodos de hoja terminales y un nuevo nodo de decisión. Como ambos registros con activos bajos tienen un riesgo de crédito malo, esta clasificación tiene una confianza del 100% y no se requieren más divisiones. Es similar para los dos registros con altos activos. Sin embargo, los cuatro registros en el nodo de decisión A (activos = medio) (assets = médium) contienen riesgos de crédito buenos y malos, por lo que se requiere una mayor división.

Procedemos a determinar la división óptima para el nodo de decisión A, que contiene los registros 3, 4, 5 y 8, como se indica en la Tabla 11.8. Debido a que tres de los cuatro registros se clasifican como buenos riesgos crediticios, y el registro restante se clasifica como riesgo de crédito malo, la entropía antes de la división es

$$H(A) = -\sum_j p_j \log_2(p_j) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) = 0.8113$$

TABLE 11.8 Records available at decision node *A* for classifying credit risk

Customer	Savings	Assets	Income ($1000s)	Credit Risk
3	High	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
8	Medium	Medium	75	Good

El candidato se divide para el nodo de decisión A se muestra en la Tabla 11.9.

TABLE 11.9 Candidate splits at decision node *A*

Candidate Split	Child Nodes		
1	*Savings = low*	*Savings = medium*	*Savings = high*
3	*Income* ≤ $25,000		*Income* > $25,000
4	*Income* ≤ $50,000		*Income* > $50,000
5	*Income* ≤ $75,000		*Income* > $75,000

Para la división del candidato 1, ahorros (savings), el único registro con bajos ahorros (low savings) es un buen riesgo de crédito, junto con los dos registros con ahorros medios (médium savings). Tal vez contra intuición, el único registro con grandes ahorros (high savings) es un riesgo de crédito malo. Entonces la entropía para cada una de estas tres clases es igual a cero, porque el nivel de ahorro determina el riesgo de crédito por completo. Esto también resulta en una entropía combinada de cero para la división de activos, $H_{assets}(A) = 0$, que es óptima para el nodo de decisión A. La ganancia de información para esta división es por lo tanto $H(A) - H_{assets}(A) = 0.8113 - 0.0 = 0.8113$. Esta es, por supuesto, la máxima ganancia de información posible para el nodo de decisión A. Por lo tanto, no es necesario que continuemos nuestros cálculos, porque ninguna otra división puede dar como resultado una mayor ganancia de información. Da la casualidad que el candidato dividido 3, ingreso ($income \leq \$25{,}000 \quad versus \quad income > \$25{,}000$), también resulta en la ganancia máxima de información, pero nuevamente seleccionamos arbitrariamente la primera división de ese tipo, la división de ahorro.

La Figura 11.7 muestra la forma del árbol de decisión después de la división de ahorros. Tenga en cuenta que esta es la forma desarrollada por completo, porque todos los nodos son ahora nodos hoja, y C4.5 no crecerá ningún otro nodo. Comparando el árbol C4.5 en la figura 11.7 con el árbol CART en la figura 11.4, vemos que el árbol C4.5 es más "boscoso", proporcionando una mayor amplitud, mientras que el árbol CART es un nivel más profundo. Ambos algoritmos coinciden en que los activos son la variable más importante (la división raíz) y que los ahorros también son importantes. Finalmente, una vez que el árbol de decisión está completamente desarrollado, C4.5 se embarca en una postpuning pesimista. Los lectores interesados pueden consultar a Kantardzic.

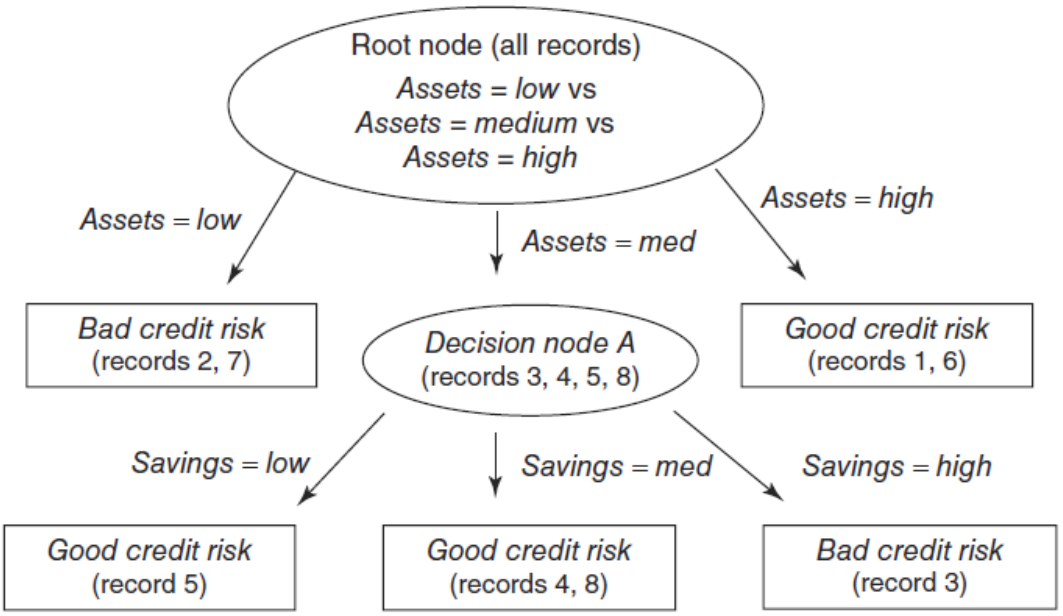

Figure 11.7 C4.5 Decision tree: fully grown form.

11.5 REGLAS DE DECISIÓN

Uno de los aspectos más atractivos de los árboles de decisión reside en su interpretabilidad, especialmente con respecto a la construcción de reglas de decisión. Las reglas de decisión se pueden construir a partir de un árbol de decisión simplemente atravesando una ruta dada desde el nodo raíz a cualquier hoja. El conjunto completo de reglas de decisión generadas por un árbol de decisión es equivalente (a efectos de clasificación) al propio árbol de decisión. Por ejemplo, desde el árbol de decisiones de la Figura 11.7, podemos construir las reglas de decisión que figuran en la Tabla 11.10.

Las reglas de decisión vienen en la forma si son antecedentes, luego consecuentes, como se muestra en la Tabla 11.10. Para reglas de decisión, el antecedente consiste en los valores de atributo de las ramas tomadas por la ruta particular a través del árbol, mientras que el consecuente consiste en el valor de clasificación para la variable objetivo dada por el nodo hoja particular.

El soporte de la regla de decisión se refiere a la proporción de registros en el conjunto de datos que se encuentran en ese nodo de hoja terminal particular. La confianza de la regla se refiere a la proporción de registros en el nodo hoja para el cual la regla de decisión es verdadera. En este pequeño ejemplo, todos nuestros nodos de hojas son puros, lo que da como resultado niveles de confianza perfectos de 100% = 1.00. En ejemplos del mundo real, como en la siguiente sección, uno no puede esperar niveles de confianza tan altos.

TABLE 11.10 Decision rules generated from decision tree in Figure 11.7

Antecedent	Consequent	Support	Confidence
If *assets* = *low*	then *bad credit risk*	$\frac{2}{8}$	1.00
If *assets* = *high*	then *good credit risk*	$\frac{2}{8}$	1.00
If *assets* = *medium* and *savings* = *low*	then *good credit risk*	$\frac{1}{8}$	1.00
If *assets* = *medium* and *savings* = *medium*	then *good credit risk*	$\frac{2}{8}$	1.00
If *assets* = *medium* and *savings* = *high*	then *bad credit risk*	$\frac{1}{8}$	1.00