8.4 Árboles Bagged (Ensamblados)

En la década de 1990, comenzaron a aparecer técnicas de conjunto (métodos que combinan las predicciones de muchos modelos). Bagging, abreviatura de bootstrap aggregation (agregación bootstrap), fue propuesto originalmente por Leo Breiman y fue una de las primeras técnicas de conjunto desarrolladas (Breiman 1996a). Bagging es un enfoque general que utiliza el bootstrap (ver Sección 4.4) en conjunto con cualquier modelo de regresión (o clasificación; ver Sección 14.3) para construir un conjunto. El método es bastante simple en su estructura y consta de los pasos del Algoritmo 8.1. Cada modelo en el conjunto se utiliza luego para generar una predicción para una nueva muestra, y estas m predicciones se promedian para dar la predicción del modelo bagged.

Los modelos Bagged proporcionan varias ventajas sobre los modelos que no están bagged. En primer lugar, bagging reduce eficazmente la varianza de una predicción a través de su proceso de agregación (consulte la Sección 5.2 para obtener una discusión sobre el trade-off entre sesgo y varianza). Para los modelos que producen una predicción inestable, como los árboles de regresión, la agregación de muchas versiones de los datos de entrenamiento realmente reduce la varianza en la predicción y, por lo tanto, hace que la predicción sea más estable.

Fig. 8.13: Linear model coefficients for the rule-based version of M5. The coefficients have been normalized to be on the same scale as Fig. 8.11. *White blocks* indicate that the predictor was not involved in linear regression equation for that rule

```
1 for i = 1 to m do
2 |   Generate a bootstrap sample of the original data
3 |   Train an unpruned tree model on this sample
4 end
```

Algorithm 8.1: Bagging

En este ejemplo, se generaron seis muestras de arranque (bootstrap) de los datos de solubilidad y se construyó un árbol de máxima profundidad para cada muestra. Estos árboles varían en su estructura (comparar Fig. 8.14b, d, que tienen estructuras diferentes en los lados derecho e izquierdo de cada árbol), y por lo tanto, la predicción para las muestras variará de un árbol a otro. Cuando las predicciones para una muestra se promedian entre todos los árboles individuales, la predicción promedio tiene una varianza más baja que la varianza entre las predicciones individuales. Esto significa que si generáramos una secuencia diferente de muestras de arranque, construyéramos un modelo en cada muestra y promediáramos las predicciones entre los modelos, probablemente obtendríamos un valor predicho muy similar para la muestra seleccionada, al igual que con el modelo de bagging anterior. Esta característica también mejora el rendimiento predictivo de un modelo bagged sobre un modelo que no está baggeado. Si el objetivo del esfuerzo de modelado es encontrar la mejor predicción, entonces el bagging tiene una ventaja distintiva.

Para modelos estables y de baja varianza como la regresión lineal y MARS, en cambio, el bagging ofrece menos mejora en el rendimiento predictivo. Considere la Fig. 8.15, donde se aplicó bagging a árboles, modelos lineales y MARS para los datos de solubilidad y también para datos de un estudio de mezclas de concreto (consulte el Capítulo 10). Para cada conjunto de datos, el rendimiento del conjunto de prueba basado en RMSE se representa en función del número de iteraciones de bagging. Para los datos de solubilidad, la disminución en RMSE a lo largo de las iteraciones es similar para árboles, regresión lineal y MARS, lo cual no es un resultado típico. Esto sugiere que las predicciones del modelo de regresión lineal y MARS tienen alguna inestabilidad inherente para estos datos que se puede mejorar mediante un conjunto de bagged o que los árboles son menos efectivos para modelar los datos. Los resultados de bagging para los datos de concreto son más típicos, en los que la regresión lineal y MARS se mejoran menos a través del conjunto, mientras que las predicciones para los árboles de regresión mejoran drásticamente.

Como demostración adicional de la capacidad del bagging para reducir la varianza de la predicción de un modelo, considere la onda sinusoidal simulada en la Fig. 5.2. Para esta ilustración, se simularon 20 ondas senoidales y, para cada conjunto de datos, se calcularon modelos de árboles de regresión y MARS. Las líneas rojas en los paneles muestran la tendencia real, mientras que las múltiples líneas negras muestran las predicciones de cada modelo. Tenga en cuenta que el panel de CART tiene más ruido alrededor de la verdadera curva seno que el modelo MARS, que solo muestra variación en los puntos de cambio del patrón. Esto ilustra la alta varianza en el árbol de regresión debido a la inestabilidad del modelo. Los paneles inferiores de la figura muestran los resultados para 20 árboles de regresión bagged y modelos MARS (cada uno con 50 iteraciones del modelo). La variación alrededor de la verdadera curva se reduce en gran medida para los árboles de regresión, y para MARS, la variación solo se reduce alrededor de las partes curvilíneas del patrón. Utilizando un conjunto de prueba simulado para cada modelo, la reducción promedio en RMSE al hacer bagging del

árbol fue del 8.6%, mientras que el modelo MARS más estable tuvo una reducción correspondiente del 2% (Fig. 8.16).

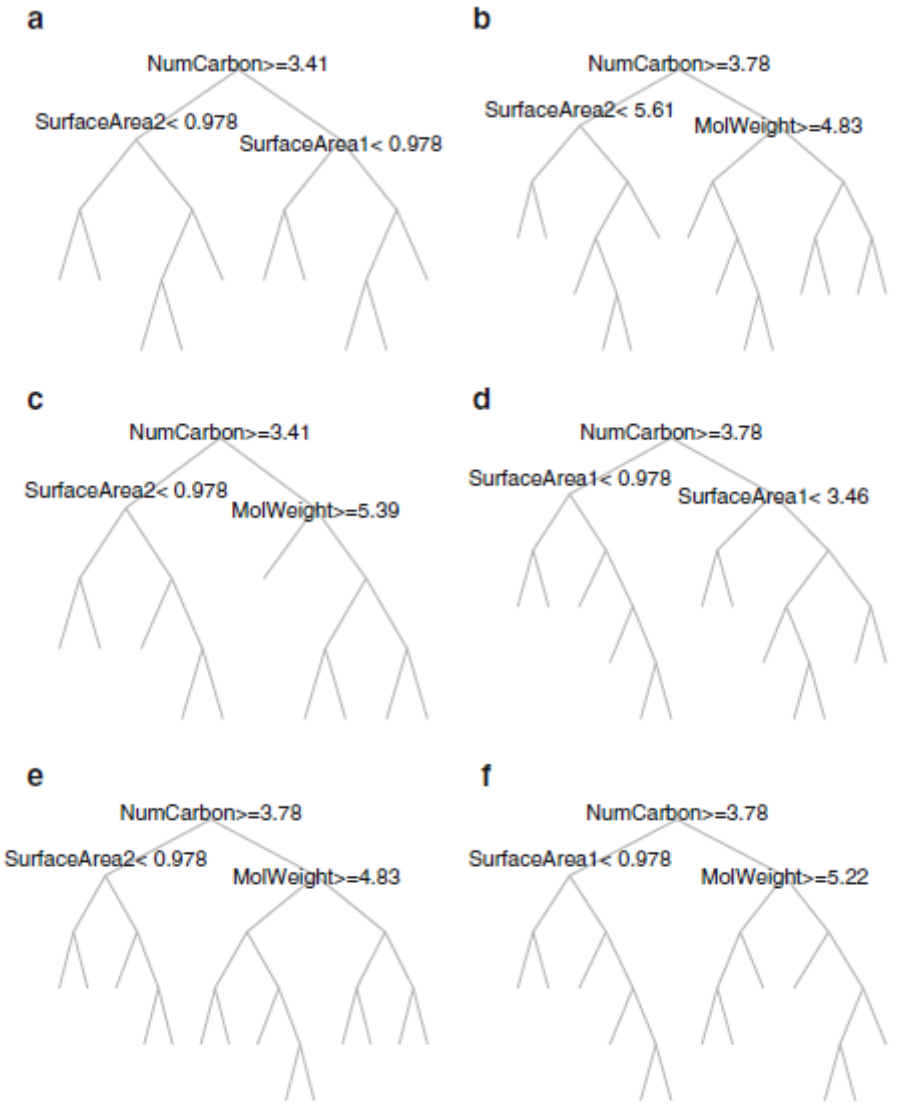

Fig. 8.14: Example of trees of maximum depth from bagging for the solubility data. Notice that the trees vary in structure, and hence the predictions will vary from tree to tree. The prediction variance for the ensemble of trees will be less than the variance of predictions from individual trees. (**a**) Sample 1. (**b**) Sample 2. (**c**) Sample 3. (**d**) Sample 4. (**e**) Sample 5. (**f**) Sample 6

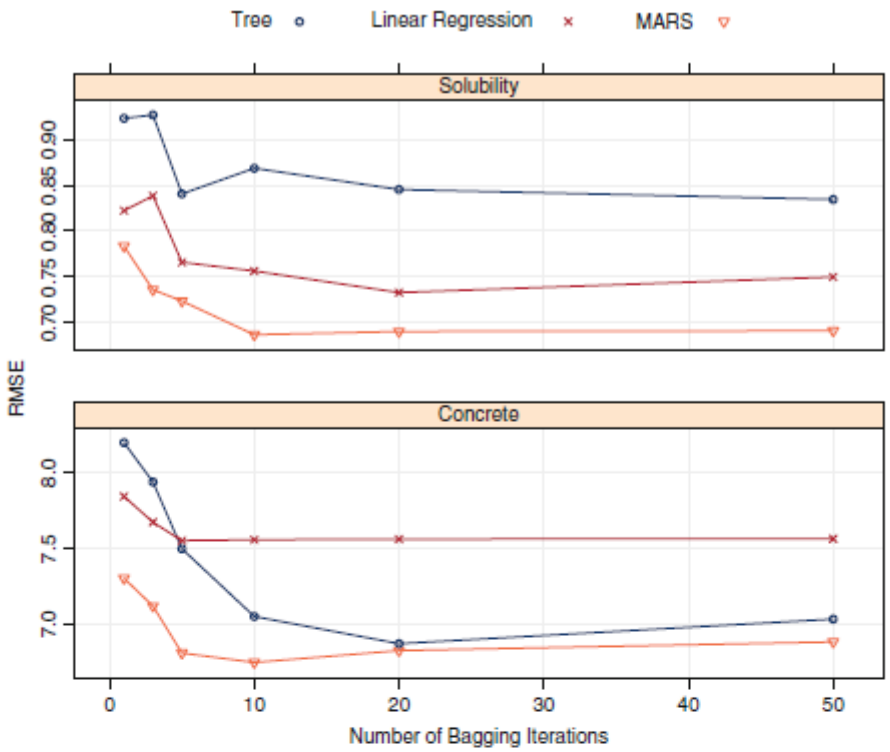

Fig. 8.15: Test set RMSE performance profiles for bagging trees, linear models, and MARS for the solubility data (*top plot*) and concrete data (*bottom plot*; see Chap. 10 for data details) by number of bootstrap samples. Bagging provides approximately the same magnitude of improvement in RMSE for all three methods for the solubility data, which is atypical. A more typical pattern of improvement from bagging is shown for the concrete data. In this case, trees benefit the most

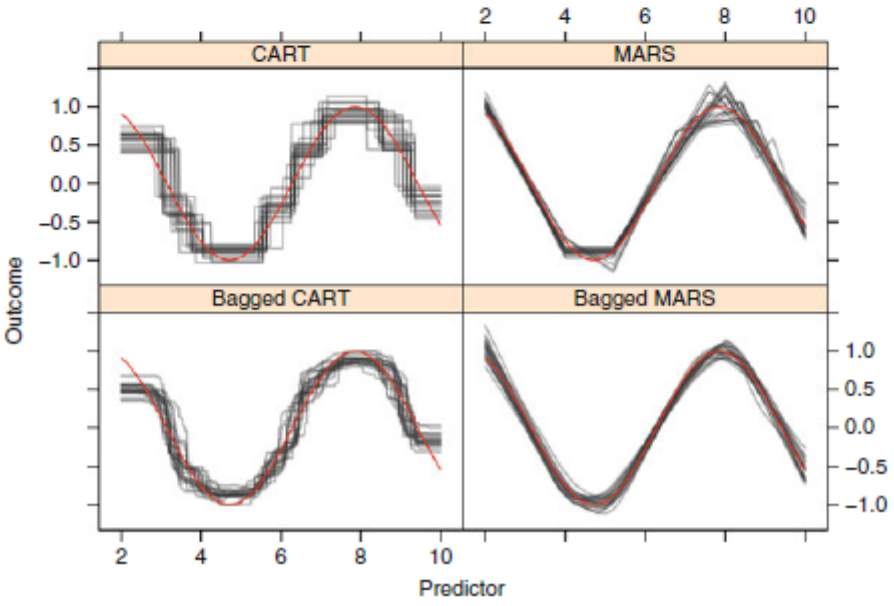

Fig. 8.16: The effect of bagging regression trees and MARS models

Otra ventaja de los modelos de bagging es que pueden proporcionar su propia estimación interna del rendimiento predictivo, que se correlaciona bien con las estimaciones de la validación cruzada o las estimaciones del conjunto de prueba. Aquí está el motivo: al construir

una muestra de arranque para cada modelo en el conjunto, ciertas muestras quedan excluidas. Estas muestras se llaman out-of-bag y se pueden utilizar para evaluar el rendimiento predictivo de ese modelo específico, ya que no se utilizaron para construir el modelo. Por lo tanto, cada modelo en el conjunto genera una medida del rendimiento predictivo cortesía de las muestras out-of-bag. El promedio de las métricas de rendimiento out-of-bag se puede utilizar para evaluar el rendimiento predictivo de todo el conjunto, y este valor generalmente se correlaciona bien con la evaluación del rendimiento predictivo que se puede obtener con la validación cruzada o desde un conjunto de prueba. Esta estimación del error se conoce comúnmente como la estimación out-of-bag.

En su forma básica, el usuario tiene una opción que hacer para el bagging: el número de muestras de arranque para agregar, m. A menudo, observamos una disminución exponencial en la mejora predictiva a medida que aumenta el número de iteraciones; la mayor mejora en el rendimiento de predicción se obtiene con un pequeño número de árboles ($m < 10$). Para ilustrar este punto, considere la Fig. 8.17 que muestra el rendimiento predictivo (RMSE) para diferentes números de muestras de arranque para árboles CART. Observamos que el rendimiento predictivo mejora a través de diez árboles y luego se estabiliza con mejoras muy limitadas más allá de ese punto. En nuestra experiencia, aún se pueden lograr pequeñas mejoras utilizando conjuntos de bagging de hasta 50 árboles. Si el rendimiento no alcanza un nivel aceptable después de 50 iteraciones de bagging, entonces sugerimos probar otros métodos de conjunto más potentes, como bosques aleatorios y boosting, que se describirán en las siguientes secciones.

Para los datos de solubilidad, los árboles CART sin bagging producen un RMSE de validación cruzada óptimo de 0.97 con un error estándar de 0.021. Al aplicar bagging, el rendimiento mejora y alcanza un RMSE de 0.9, con un error estándar de 0.019. Los árboles de inferencia condicional, al igual que los árboles CART, también se pueden someter a bagging. Como comparación, los árboles de inferencia condicional sin bagging tienen un RMSE y un error estándar óptimos de 0.93 y 0.034, respectivamente. Los árboles de inferencia condicional sometidos a bagging reducen el RMSE óptimo a 0.8 con un error estándar de 0.018. Para ambos tipos de modelos, el bagging mejora el rendimiento y reduce la varianza de la estimación. En este ejemplo específico, parece que bagging de árboles de inferencia condicional tiene una ligera ventaja sobre los árboles CART en rendimiento predictivo medido por RMSE. Los valores R2 del conjunto de prueba son paralelos al rendimiento de RMSE validado cruzado, con árboles de inferencia condicional ligeramente mejor (0.87) que los árboles CART (0.85).

Aunque el bagging generalmente mejora el rendimiento predictivo para modelos inestables, hay algunas advertencias. En primer lugar, los costos computacionales y los requisitos de memoria aumentan a medida que aumenta el número de muestras de arranque. Esta desventaja se puede mitigar en su mayoría si el modelador tiene acceso a la computación en paralelo porque el proceso de bagging se puede paralelizar fácilmente. Recuerde que cada muestra de arranque y modelo correspondiente es independiente de cualquier otra muestra y modelo. Esto significa que cada modelo se puede construir por separado y todos los modelos se pueden unir al final para generar la predicción.

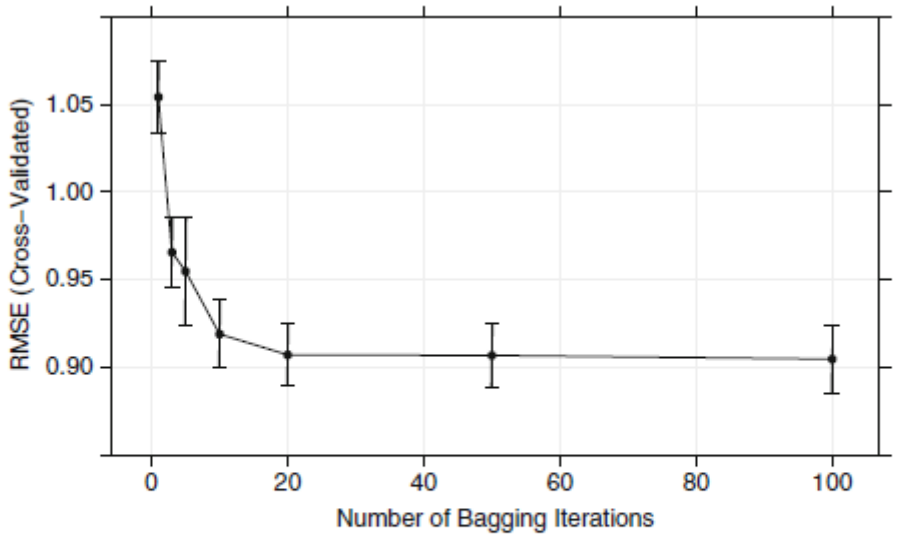

Fig. 8.17: Cross-validated performance profile for bagging CART trees for the solubility data by number of bootstrap samples. Vertical lines indicate ± one-standard error of $RMSE$. Most improvement in predictive performance is obtained aggregating across ten bootstrap replications

Otra desventaja de este enfoque es que un modelo bagged es mucho menos interpretable que un modelo que no está bagged. No se pueden obtener reglas convenientes como las que se pueden obtener de un solo árbol de regresión, como se muestra en la Fig. 8.4. Sin embargo, se pueden construir medidas de importancia de variables al combinar medidas de importancia de los modelos individuales en todo el conjunto. Se discutirá más sobre la importancia de variables en la siguiente sección cuando examinemos los bosques aleatorios.

8.5 Bosques Aleatorios

Como se ilustra con los datos de solubilidad, el ensacado de árboles (o cualquier técnica de alta varianza y bajo sesgo) mejora el rendimiento predictivo en comparación con un solo árbol al reducir la varianza de la predicción. La generación de muestras de arranque introduce un componente aleatorio en el proceso de construcción del árbol, lo que induce una distribución de árboles y, por lo tanto, también una distribución de valores predichos para cada muestra. Sin embargo, los árboles en el ensacado no son completamente independientes entre sí, ya que todos los predictores originales se consideran en cada división de cada árbol. Se puede imaginar que si comenzamos con un número suficientemente grande de muestras originales y una relación entre predictores y respuesta que puede ser adecuadamente modelada por un árbol, entonces los árboles de diferentes muestras de arranque pueden tener estructuras similares entre sí (especialmente en la parte superior de los árboles) debido a la relación subyacente. Esta característica se conoce como correlación de árboles y evita que el ensacado reduzca óptimamente la varianza de los valores predichos. La Figura 8.14 proporciona una ilustración directa de este fenómeno. A pesar de tomar muestras de arranque, cada árbol comienza a dividirse en el número de átomos de carbono a un valor escalado de aproximadamente 3.5. Las divisiones de segundo nivel varían un poco más, pero están restringidas a ambos predictores de área superficial y peso molecular. Si bien cada árbol es único en última instancia, ninguno de los árboles es exactamente igual, todos comienzan con una estructura similar y, por lo tanto, están relacionados entre sí. Por lo tanto, la reducción de varianza proporcionada por el ensacado podría mejorarse. Para una explicación matemática

del fenómeno de correlación de árboles, consulte Hastie et al. (2008). Reducir la correlación entre los árboles, conocido como decorrelación de árboles, es el siguiente paso lógico para mejorar el rendimiento del ensacado.

Desde una perspectiva estadística, reducir la correlación entre los predictores se puede lograr mediante la adición de aleatoriedad al proceso de construcción del árbol. Después de que Breiman presentara el ensacado, varios autores ajustaron el algoritmo al agregar aleatoriedad al proceso de aprendizaje. Dado que los árboles eran un método popular de aprendizaje para el ensacado, Dietterich (2000) desarrolló la idea de la selección aleatoria de divisiones, donde se construyen árboles utilizando un subconjunto aleatorio de los primeros k predictores en cada división del árbol. Otro enfoque fue construir árboles completos basados en subconjuntos aleatorios de descriptores (Ho 1998; Amit y Geman 1997). Breiman (2000) también intentó agregar ruido a la respuesta para perturbar la estructura del árbol. Después de evaluar cuidadosamente estas generalizaciones del algoritmo original de ensacado, Breiman (2001) construyó un algoritmo unificado llamado bosques aleatorios. Un algoritmo general de bosques aleatorios para un modelo basado en árboles se puede implementar como se muestra en el Algoritmo 8.2.

Cada modelo en el conjunto se utiliza para generar una predicción para una nueva muestra y estas m predicciones se promedian para dar la predicción del bosque. Dado que el algoritmo selecciona aleatoriamente predictores en cada división, la correlación entre árboles será necesariamente menor. Como ejemplo, las primeras divisiones para los primeros seis árboles en el bosque aleatorio para los datos de solubilidad son NumNonHBonds, NumCarbon, NumNonHAtoms, NumCarbon, NumCarbon y NumCarbon, que son diferentes de los árboles ilustrados en la Figura 8.14.

```
Select the number of models to build, m
for i = 1 to m do
    Generate a bootstrap sample of the original data
    Train a tree model on this sample
    for each split do
        Randomly select k (< P) of the original predictors
        Select the best predictor among the k predictors and
        partition the data
    end
    Use typical tree model stopping criteria to determine when a
    tree is complete (but do not prune)
end
```

Algorithm 8.2: Basic Random Forests

El parámetro de ajuste de los bosques aleatorios es el número de predictores seleccionados aleatoriamente, k, para elegir en cada división, y comúnmente se conoce como mtry. En el contexto de regresión, Breiman (2001) recomienda establecer mtry en un tercio del número de predictores. Para ajustar el parámetro mtry, dado que los bosques aleatorios son intensivos computacionalmente, sugerimos comenzar con cinco valores de k distribuidos de manera uniforme en el rango de 2 a P. El practicante también debe especificar el número de árboles para el bosque. Breiman (2001) demostró que los bosques aleatorios están protegidos contra

el sobreajuste; por lo tanto, el modelo no se verá afectado negativamente si se construyen un gran número de árboles para el bosque. Hablando prácticamente, cuanto más grande sea el bosque, mayor será la carga computacional que tendremos que asumir para entrenar y construir el modelo. Como punto de partida, sugerimos usar al menos 1,000 árboles. Si los perfiles de rendimiento de validación cruzada siguen mejorando a 1,000 árboles, entonces incorpore más árboles hasta que el rendimiento se estabilice.

Breiman demostró que la combinación lineal de muchos aprendices independientes reduce la varianza del conjunto en comparación con cualquier aprendiz individual en el conjunto. Un modelo de bosque aleatorio logra esta reducción de varianza al seleccionar aprendices fuertes y complejos que muestran un sesgo bajo. Este conjunto de muchos aprendices independientes y fuertes produce una mejora en las tasas de error. Dado que cada aprendiz se selecciona de forma independiente de todos los aprendices anteriores, los bosques aleatorios son robustos ante una respuesta ruidosa. Detallamos más este punto en la Sección 20.2 y proporcionamos una ilustración del efecto del ruido en los bosques aleatorios, así como en muchos otros modelos. Al mismo tiempo, la independencia de los aprendices puede generar un ajuste insuficiente de los datos cuando la respuesta no es ruidosa (Fig. 5.1).

En comparación con el ensacado, los bosques aleatorios son más eficientes computacionalmente en una base árbol por árbol, ya que el proceso de construcción del árbol solo necesita evaluar una fracción de los predictores originales en cada división, aunque generalmente se requieren más árboles en los bosques aleatorios. Combinar este atributo con la capacidad de procesar en paralelo la construcción de árboles hace que los bosques aleatorios sean más eficientes computacionalmente que el aumento (Sección 8.6).

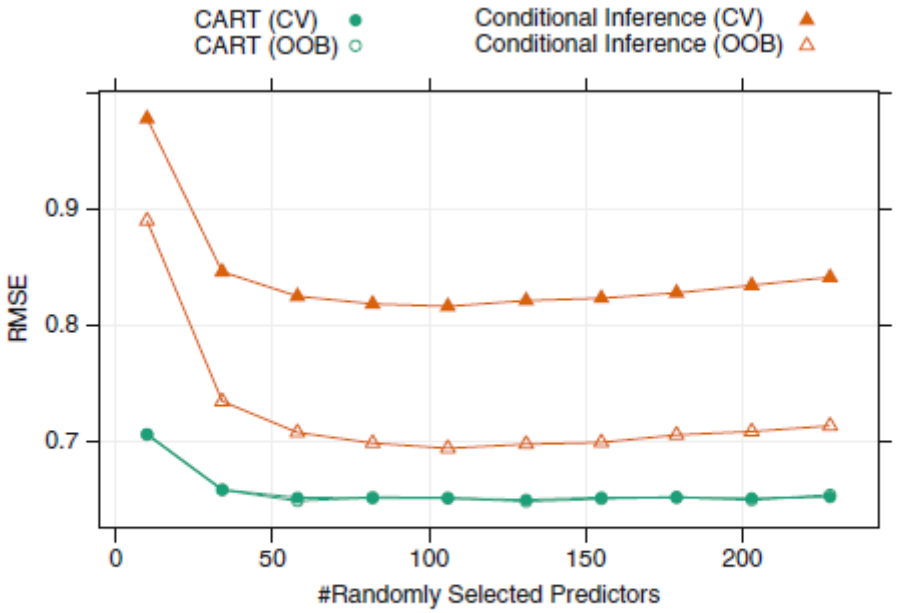

Fig. 8.18: Cross-validated RMSE profile for the CART and conditional inference approaches to random forests

Al igual que con el ensacado, los árboles CART o los árboles de inferencia condicional se pueden utilizar como aprendices base en los bosques aleatorios. Ambos de estos aprendices base se utilizaron, junto con validación cruzada de 10 pliegues y validación out-of-bag, para entrenar modelos en los datos de solubilidad. El parámetro mtry se evaluó en diez valores, desde 10 hasta 228. Los perfiles de RMSE para estas combinaciones se presentan en la Figura

8.18. Contrario al ensacado, los árboles CART tienen un mejor rendimiento que los árboles de inferencia condicional en todos los valores del parámetro de ajuste. Cada uno de los perfiles muestra un plateau plano entre mtry = 58 y mtry = 155. El modelo de bosque aleatorio basado en CART fue óptimo numéricamente en mtry = 131, independientemente del método utilizado para estimar el RMSE. Nuestra experiencia indica que el parámetro de ajuste del bosque aleatorio no tiene un efecto drástico en el rendimiento. En estos datos, la única diferencia real en el RMSE ocurre cuando se utiliza el valor más pequeño (10 en este caso). A menudo, un valor tan pequeño no está asociado con un rendimiento óptimo. Sin embargo, hemos visto ejemplos raros donde valores pequeños del parámetro de ajuste generan los mejores resultados. Para obtener una evaluación rápida del rendimiento del modelo de bosque aleatorio, el valor predeterminado del parámetro de ajuste para la regresión (mtry = P/3) tiende a funcionar bien. Si hay un deseo de maximizar el rendimiento, ajustar este valor puede resultar en una ligera mejora.

En la Figura 8.18, también se observa que los modelos de bosque aleatorio construidos con árboles CART tuvieron resultados de RMSE extremadamente similares con la estimación de error out-of-bag y la validación cruzada (cuando se comparan a través de los parámetros de ajuste). No está claro si el patrón visto en estos datos se generaliza, especialmente bajo circunstancias diferentes como tamaños de muestra pequeños. El uso de la tasa de error out-of-bag reduciría drásticamente el tiempo computacional para ajustar modelos de bosque aleatorio. Para los bosques creados utilizando árboles de inferencia condicional, el error out-of-bag fue mucho más optimista que el RMSE validado cruzadamente. Nuevamente, la lógica detrás de este patrón no está clara.

La naturaleza de conjunto de los bosques aleatorios hace imposible comprender la relación entre los predictores y la respuesta. Sin embargo, como los árboles son el aprendiz típico para este método, es posible cuantificar el impacto de los predictores en el conjunto. Breiman (2000) propuso originalmente permutar aleatoriamente los valores de cada predictor para la muestra out-of-bag de un predictor a la vez para cada árbol. La diferencia en el rendimiento predictivo entre la muestra no permutada y la muestra permutada para cada predictor se registra y se agrega en todo el conjunto. Otro enfoque es medir la mejora en la pureza del nodo basada en la métrica de rendimiento para cada predictor en cada ocurrencia de ese predictor en todo el conjunto. Estos valores individuales de mejora para cada predictor se agregan luego en todo el conjunto para determinar la importancia general del predictor.

Aunque esta estrategia para determinar la influencia relativa de un predictor es muy diferente al enfoque descrito en la Sección 8 para árboles de regresión individuales, sufre de las mismas limitaciones relacionadas con el sesgo. Además, Strobl et al. (2007) mostraron que las correlaciones entre predictores pueden tener un impacto significativo en los valores de importancia. Por ejemplo, los predictores no informativos con altas correlaciones con predictores informativos tenían valores de importancia anormalmente grandes. En algunos casos, su importancia era mayor o igual que la de variables débilmente importantes. También demostraron que el parámetro de ajuste mtry tiene un efecto serio en los valores de importancia.

Otro impacto de las correlaciones entre predictores es diluir las importancias de los predictores clave. Por ejemplo, supongamos que un predictor crítico tiene una importancia de X. Si otro predictor es igualmente crítico pero está casi perfectamente correlacionado con el primero, la importancia de estos dos predictores será aproximadamente X/2. Si tres predictores de este tipo estuvieran en el modelo, los valores disminuirían aún más a X/3 y así

sucesivamente. Esto puede tener implicaciones profundas para algunos problemas. Por ejemplo, los datos de perfiles de expresión de ARN tienden a medir el mismo gen en muchos lugares y, como resultado, las variables dentro del gen tienden a tener correlaciones muy altas. Si este gen fuera importante para predecir algún resultado, agregar todas las variables a un modelo de bosque aleatorio haría que el gen pareciera menos importante de lo que realmente es.

Strobl et al. (2007) desarrollaron un enfoque alternativo para calcular la importancia en modelos de bosques aleatorios que tiene en cuenta las correlaciones entre predictores. Su metodología reduce el efecto de la redundancia entre predictores. Sin embargo, no ajusta el efecto de dilución mencionado anteriormente. Los valores de importancia de variables para los 25 principales predictores de los datos de solubilidad se presentan en la Figura 8.19. Para este modelo, MolWeight, NumCarbon, SurfaceArea2 y SurfaceArea1 ascienden en la métrica de importancia, y los valores de importancia comienzan a disminuir con las huellas dactilares. Los valores de importancia para las huellas dactilares 116 y 75 son los mejores, lo que puede indicar que las estructuras representadas por estas huellas dactilares tienen un impacto en la solubilidad de un compuesto.

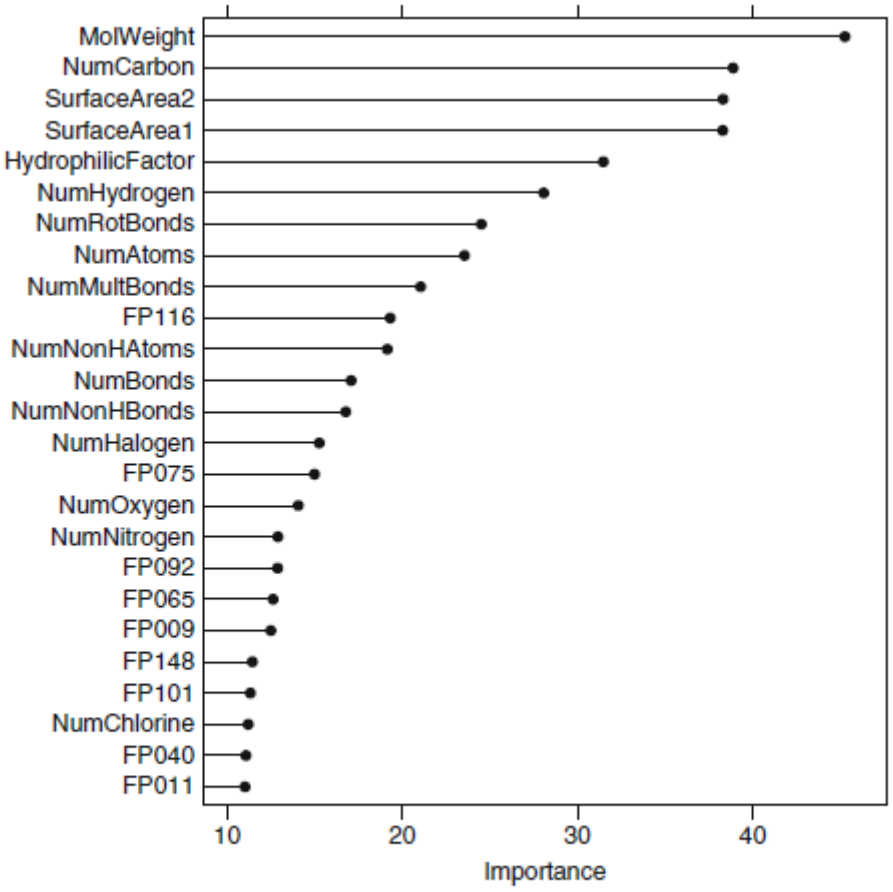

Fig. 8.19: Variable importance scores for the top 25 predictors used in the random forest CART tree model for solubility

Al contrastar los resultados de importancia de bosques aleatorios con un solo árbol CART (Figura 8.6), observamos que 2 de los 4 principales predictores son los mismos (SurfaceArea2 y NumCarbon) y 14 de los 16 principales son iguales. Sin embargo, los órdenes de importancia son muy diferentes. Por ejemplo, NumNonHBonds es el predictor principal para un árbol CART,

pero termina clasificado en el 14º lugar para los bosques aleatorios; los bosques aleatorios identifican a MolWeight como el predictor principal, mientras que un árbol CART lo clasifica en quinto lugar. Estas diferencias no deberían ser desconcertantes; más bien, enfatizan que la codicia de un solo árbol prioriza los predictores de manera diferente a un bosque aleatorio.

8.6 Boosting

Boosting (impulso) modelos fueron desarrollados originalmente para problemas de clasificación y posteriormente se extendieron al ámbito de la regresión. Los lectores que no estén familiarizados con el boosting pueden beneficiarse al leer primero sobre el boosting para clasificación (Sección 14.5) y luego regresar a esta sección. Para la completitud de esta sección, proporcionaremos una historia del boosting para establecer un puente desde el desarrollo original de boosting en clasificación hasta su uso en el contexto de la regresión. Esta historia comienza con el algoritmo AdaBoost y evoluciona hacia la máquina de boosting de gradiente estocástico de Friedman, que ahora es ampliamente aceptada como el algoritmo de boosting de elección entre los profesionales.

En la década de 1990, aparecieron algoritmos de boosting (Schapire 1990; Freund 1995; Schapire 1999) influenciados por la teoría del aprendizaje (Valiant 1984; Kearns y Valiant 1989), en los cuales se combinaban (o potenciaban) varios clasificadores débiles (un clasificador que predice marginalmente mejor que al azar) para producir un clasificador de conjunto con una tasa de error de clasificación generalizada superior. Los investigadores lucharon durante un tiempo para encontrar una implementación efectiva de la teoría de boosting, hasta que Freund y Schapire colaboraron para producir el algoritmo AdaBoost (Schapire 1999). AdaBoost (ver Algoritmo 14.2) proporcionó una implementación práctica del concepto de Kerns y Valiant de potenciar un aprendiz débil en un aprendiz fuerte (Kearns y Valiant 1989)

El boosting, especialmente en la forma del algoritmo AdaBoost, demostró ser una herramienta de predicción poderosa, superando generalmente a cualquier modelo individual. Su éxito atrajo la atención de la comunidad de modelado y su uso se extendió con aplicaciones en expresión génica (Dudoit et al. 2002; Ben-Dor et al. 2000), quimiometría (Varmuza et al. 2003) e identificación de géneros musicales (Bergstra et al. 2006), por mencionar algunas.

El algoritmo AdaBoost funcionó claramente, y después de su llegada exitosa, varios investigadores (Friedman et al. 2000) conectaron el algoritmo AdaBoost con conceptos estadísticos de funciones de pérdida, modelado aditivo y regresión logística, y demostraron que el boosting se puede interpretar como un modelo aditivo secuencial hacia adelante que minimiza la pérdida exponencial. Esta comprensión fundamental del boosting condujo a una nueva perspectiva que facilitó varias generalizaciones algorítmicas para problemas de clasificación (Sección 14.5). Además, esta nueva perspectiva también permitió que el método se extendiera a problemas de regresión.

La capacidad de Friedman para ver el marco estadístico del boosting produjo un algoritmo simple, elegante y altamente adaptable para diferentes tipos de problemas (Friedman 2001). Llamó a este método "máquinas de boosting de gradiente" que abarca tanto la clasificación como la regresión. Los principios básicos del boosting de gradiente son los siguientes: dada una función de pérdida (por ejemplo, error cuadrático para regresión) y un aprendiz débil (por ejemplo, árboles de regresión), el algoritmo busca encontrar un modelo aditivo que minimice la función de pérdida. El algoritmo generalmente se inicializa con la mejor suposición de la respuesta (por ejemplo, la media de la respuesta en regresión). Se calcula el gradiente (por

ejemplo, residuo), y luego se ajusta un modelo a los residuos para minimizar la función de pérdida. El modelo actual se suma al modelo anterior y el procedimiento continúa durante un número especificado de iteraciones.

Como se describe a lo largo de este texto, cualquier técnica de modelado con parámetros de ajuste puede producir un rango de capacidad predictiva, desde débil hasta fuerte. Dado que el boosting requiere un aprendiz débil, casi cualquier técnica con parámetros de ajuste se puede convertir en un aprendiz débil. Los árboles, como resulta, son un excelente aprendiz base para boosting por varias razones. Primero, tienen la flexibilidad para ser aprendices débiles simplemente restringiendo su profundidad. En segundo lugar, los árboles separados se pueden agregar fácilmente, al igual que los predictores individuales se pueden agregar en un modelo de regresión, para generar una predicción. Y tercero, los árboles se pueden generar muy rápidamente. Por lo tanto, los resultados de los árboles individuales se pueden agregar directamente, haciéndolos inherentemente adecuados para un proceso de modelado aditivo.

Cuando se utilizan árboles de regresión como aprendices base, el boosting de gradiente simple para regresión tiene dos parámetros de ajuste: la profundidad del árbol y el número de iteraciones. La profundidad del árbol en este contexto también se conoce como profundidad de interacción, ya

que cada división subsecuente se puede pensar como un término de interacción de nivel superior con todos los otros predictores de divisiones anteriores. Si se utiliza el error cuadrático como función de pérdida, entonces se puede encontrar un algoritmo de boosting simple utilizando estos parámetros de ajuste en el Algoritmo 8.3.

1 Select tree depth, D, and number of iterations, K
2 Compute the average response, $\overline{y}$, and use this as the initial predicted value for each sample
3 **for** $k = 1$ *to* K **do**
4 Compute the residual, the difference between the observed value and the *current* predicted value, for each sample
5 Fit a regression tree of depth, D, using the residuals as the response
6 Predict each sample using the regression tree fit in the previous step
7 Update the predicted value of each sample by adding the previous iteration's predicted value to the predicted value generated in the previous step
8 **end**

Algorithm 8.3: Simple Gradient Boosting for Regression

Claramente, la versión de boosting presentada en el Algoritmo 8.3 tiene similitudes con los bosques aleatorios: la predicción final se basa en un conjunto de modelos, y se utilizan árboles como aprendices base. Sin embargo, la forma en que se construyen los conjuntos difiere sustancialmente entre cada método. En los bosques aleatorios, todos los árboles se crean de manera independiente, cada árbol se crea con la máxima profundidad y cada árbol contribuye por igual al modelo final. Los árboles en el boosting, sin embargo, dependen de los árboles

anteriores, tienen una profundidad mínima y contribuyen de manera desigual al modelo final. A pesar de estas diferencias, tanto los bosques aleatorios como el boosting ofrecen un rendimiento predictivo competitivo. El tiempo de computación para el boosting a menudo es mayor que para los bosques aleatorios, ya que los bosques aleatorios pueden procesarse fácilmente en paralelo dado que los árboles se crean de manera independiente.

Friedman reconoció que su máquina de boosting de gradiente podría ser susceptible al sobreajuste, ya que el aprendiz empleado, incluso en su capacidad de aprendiz débilmente definido, tiene la tarea de ajustarse de manera óptima al gradiente. Esto significa que el boosting seleccionará al aprendiz óptimo en cada etapa del algoritmo. A pesar de utilizar aprendices débiles, el boosting sigue empleando la estrategia ávida de elegir el aprendiz débil óptimo en cada etapa. Aunque esta estrategia genera una solución óptima en la etapa actual, tiene las desventajas de no encontrar el modelo global óptimo y de sobreajustar los datos de entrenamiento.

Un remedio para la avidez es limitar el proceso de aprendizaje mediante la aplicación de regularización, o reducción, de la misma manera que se ilustra en la Sección 6.4. En el Algoritmo 8.3, una estrategia de regularización se puede introducir en la última línea del bucle. En lugar de agregar el valor predicho para una muestra al valor predicho de la iteración anterior, solo se agrega una fracción del valor predicho actual al valor predicho de la iteración anterior. Esta fracción comúnmente se conoce como la tasa de aprendizaje y se parametriza con el símbolo λ. Este parámetro puede tomar valores entre 0 y 1 y se convierte en otro parámetro de ajuste para el modelo. Ridgeway (2007) sugiere que los valores pequeños del parámetro de aprendizaje (< 0.01) funcionan mejor, pero también señala que el valor del parámetro es inversamente proporcional al tiempo de computación necesario para encontrar un modelo óptimo, ya que se necesitan más iteraciones. Tener más iteraciones también implica que se necesita más memoria para almacenar el modelo.

Después de que Friedman publicó su máquina de boosting de gradiente, consideró algunas de las propiedades de la técnica de bagging de Breiman. Específicamente, la naturaleza de muestreo aleatorio del bagging ofrecía una reducción en la varianza de la predicción. Friedman actualizó el algoritmo de la máquina de boosting con un esquema de muestreo aleatorio y denominó al nuevo procedimiento "stochastic gradient boosting" (boosting de gradiente estocástico). Para hacer esto, Friedman insertó el siguiente paso antes de la línea dentro del bucle: seleccionar aleatoriamente una fracción de los datos de entrenamiento. Los residuos y los modelos en los pasos restantes de la iteración actual se basan solo en la muestra de datos. La fracción de datos de entrenamiento utilizada, conocida como la fracción de bagging, luego se convierte en otro parámetro de ajuste para el modelo. Resulta que esta modificación simple mejoró la precisión de la predicción del boosting al tiempo que reducía los recursos computacionales requeridos. Friedman sugiere usar una fracción de bagging de alrededor de 0.5; sin embargo, este valor se puede ajustar como cualquier otro parámetro.

La Figura 8.20 presenta los resultados de RMSE (error cuadrático medio) cruzado validado para árboles boosteados a través de los parámetros de ajuste de la profundidad del árbol (1-7), el número de árboles (100-1,000) y la reducción (0.01 o 0.1); la fracción de bagging en esta ilustración se fijó en 0.5. Al examinar esta figura, el valor más grande de reducción (gráfico de la derecha) tiene un impacto en la reducción del RMSE para todas las opciones de profundidad del árbol y número de árboles. Además, el RMSE disminuye a medida que aumenta la profundidad del árbol cuando la reducción es 0.01. El mismo patrón se mantiene para el RMSE cuando la reducción es 0.1 y el número de árboles es inferior a 300.

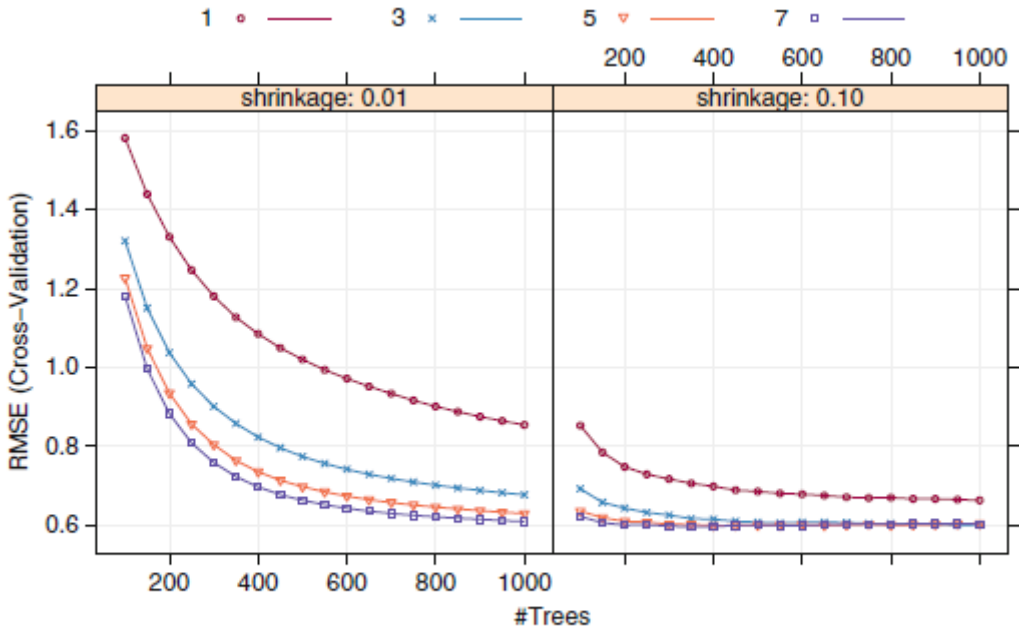

Fig. 8.20: Cross-validated RMSE profiles for the boosted tree model

Usando la regla de un error estándar, el árbol boosteado óptimo tiene una profundidad de 3 con 400 árboles y una reducción de 0.1. Estas configuraciones producen un RMSE (error cuadrático medio) cruzado validado de 0.616.

La importancia de las variables para el boosting es una función de la reducción en el error cuadrático. Específicamente, la mejora en el error cuadrático debido a cada predictor se suma dentro de cada árbol en el conjunto (es decir, cada predictor obtiene un valor de mejora para cada árbol). Los valores de mejora para cada predictor se promedian en todo el conjunto para obtener un valor de importancia general (Friedman 2002; Ridgeway 2007). Las 25 principales variables predictoras para el modelo se presentan en la Figura 8.21. NumCarbon y MolWeight destacan en este ejemplo como las más importantes, seguidas por SurfaceArea1 y SurfaceArea2; los valores de importancia disminuyen después de aproximadamente 7 predictores. Al comparar estos resultados con los bosques aleatorios, vemos que ambos métodos identifican los mismos 4 principales predictores, aunque en un orden diferente. El perfil de importancia para el boosting tiene una pendiente de importancia mucho más pronunciada que la de los bosques aleatorios. Esto se debe a que los árboles del boosting dependen entre sí y, por lo tanto, tendrán estructuras correlacionadas a medida que el método sigue el gradiente. Por lo tanto, muchos de los mismos predictores se seleccionarán en los árboles, aumentando su contribución a la métrica de importancia. Las diferencias en el orden y la magnitud de la importancia de las variables entre los bosques aleatorios y el boosting no deben ser motivo de preocupación. En cambio, se deben considerar como dos perspectivas diferentes de los datos y utilizar cada vista para proporcionar una comprensión de las relaciones generales entre los predictores y la respuesta.

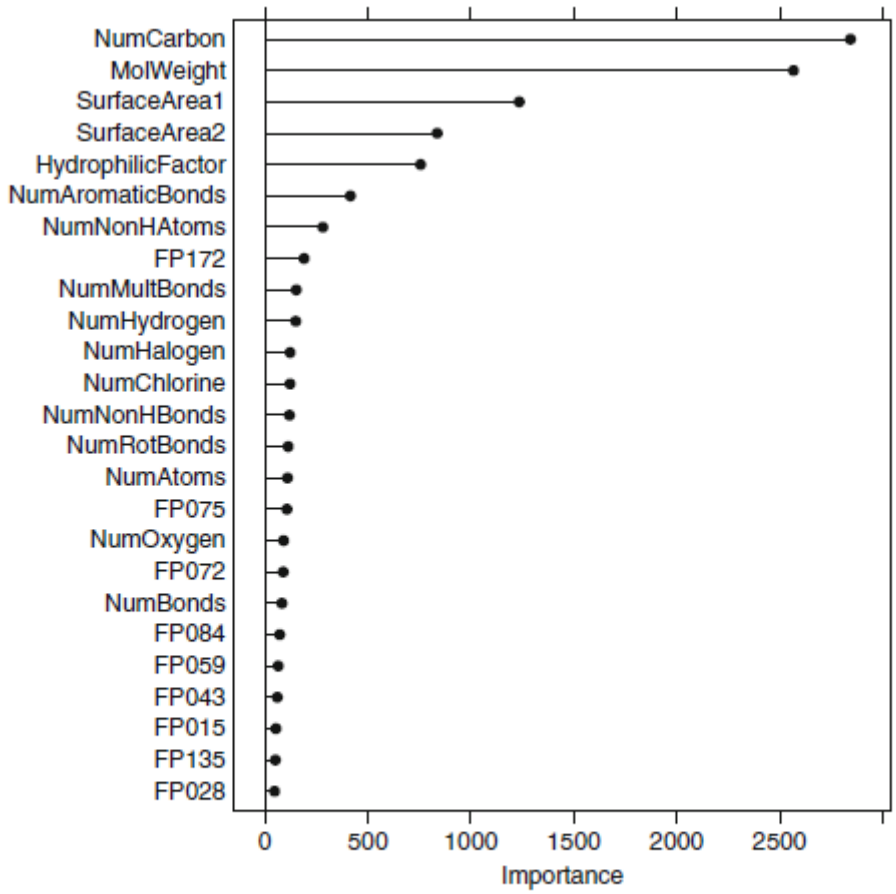

Fig. 8.21: Variable importance scores for the top 25 predictors used in the stochastic gradient boosting model for solubility

14.3 Árboles Bootstrap (Bagged Trees)

El Bootstrap para clasificación es una modificación simple del Bootstrap para regresión (Sección 8.4). Específicamente, el árbol de regresión en el Algoritmo 8.1 se reemplaza con un árbol de clasificación no podado para modelar C clases. Al igual que en el entorno de regresión, cada modelo en el conjunto se utiliza para predecir la clase de la nueva muestra. Dado que cada modelo tiene un peso igual en el conjunto, se puede pensar que cada modelo emite un voto por la clase a la que cree que pertenece la nueva muestra. El número total de votos dentro de cada clase se divide entonces por el número total de modelos en el conjunto (M) para producir un vector de probabilidad predicho para la muestra. La nueva muestra se clasifica luego en el grupo que tiene más votos y, por lo tanto, la probabilidad más alta.

Table 14.1: The 2008 holdout set confusion matrix for the random forest model

	Observed class	
	Successful	Unsuccessful
Successful	491	144
Unsuccessful	79	843

This model had an overall accuracy of 85.7 %, a sensitivity of 86.1 %, and a specificity of 85.4 %

Para los datos de subvención, se construyeron modelos de bootstrap utilizando ambas estrategias para predictores categóricos. Como se discutió en el capítulo de árboles de regresión, el rendimiento de Bootstrap a menudo se estabiliza con alrededor de 50 árboles, por lo que se seleccionó 50 como el número de árboles para cada uno de estos modelos. La Figura 14.7 ilustra el rendimiento del conjunto Bootstrap utilizando categorías independientes o agrupadas. Ambas de estas curvas ROC son más suaves que las curvas producidas con árboles de clasificación o J48, lo que indica la capacidad de Bootstrap para reducir la varianza mediante el conjunto. Además, ambos modelos de Bootstrap tienen mejores AUC (0.92 para ambos) que cualquiera de los modelos anteriores. Para estos datos, no parece haber una diferencia obvia en el rendimiento de Bootstrap al utilizar categorías independientes o agrupadas; las curvas ROC, sensibilidades y especificidades son todas casi idénticas. El rendimiento en el conjunto de retención en la Figura 14.7 muestra una mejora respecto a los resultados de J48 (Figura 14.6).

Al igual que en el entorno de regresión, se pueden calcular medidas de importancia de variables agregando los valores de importancia de variables de los árboles individuales en el conjunto. La importancia de las variables de los 16 mejores predictores para ambos conjuntos de modelos de Bootstrap, categorías independientes y agrupadas, se presenta en la Figura 14.15, y se deja la comparación de estos resultados al lector en el Ejercicio 14.1.

14.4 Bosques Aleatorios

Los bosques aleatorios para clasificación requieren un ajuste simple en el algoritmo de bosques aleatorios de regresión (Algoritmo 8.2): se utiliza un árbol de clasificación en lugar de un árbol de regresión. Al igual que con Bootstrap, cada árbol en el bosque emite un voto para la clasificación de una nueva muestra, y la proporción de votos en cada clase en todo el conjunto es el vector de probabilidad predicho.

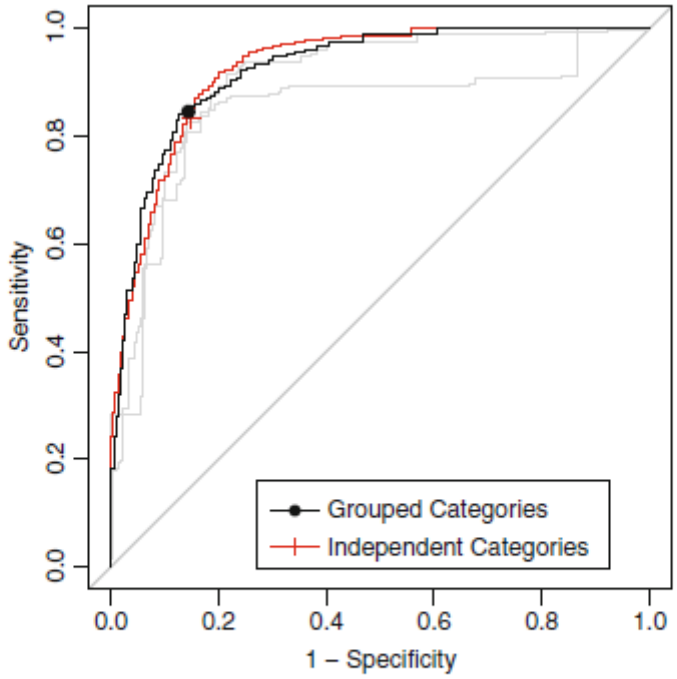

Fig. 14.7: The ROC curves for the bagged classification tree model. The area under the curves for both models was 0.92. The sensitivities and specificities were 82.98 and 85.71, respectively

Aunque el tipo de árbol cambia en el algoritmo, el parámetro de ajuste del número de predictores seleccionados aleatoriamente para elegir en cada división es el mismo (denominado mtry). Al igual que en regresión, la idea detrás del muestreo aleatorio de predictores durante el entrenamiento es descorrelacionar los árboles en el bosque. Para problemas de clasificación, Breiman (2001) recomienda establecer mtry en la raíz cuadrada del número de predictores. Para ajustar mtry, recomendamos comenzar con cinco valores distribuidos de manera uniforme en el rango de 2 a P, donde P es el número de predictores. También recomendamos comenzar con un conjunto de 1,000 árboles e incrementar ese número si el rendimiento aún no está cerca de un plateau.

En su mayor parte, los bosques aleatorios para clasificación tienen propiedades muy similares a la análoga de regresión discutida anteriormente, incluyendo:

- El modelo es relativamente insensible a los valores de mtry.

- Al igual que con la mayoría de los árboles, los requisitos de preprocesamiento de datos son mínimos.

- Se pueden calcular medidas de rendimiento out-of-bag, incluyendo precisión, sensibilidad, especificidad y matrices de confusión.

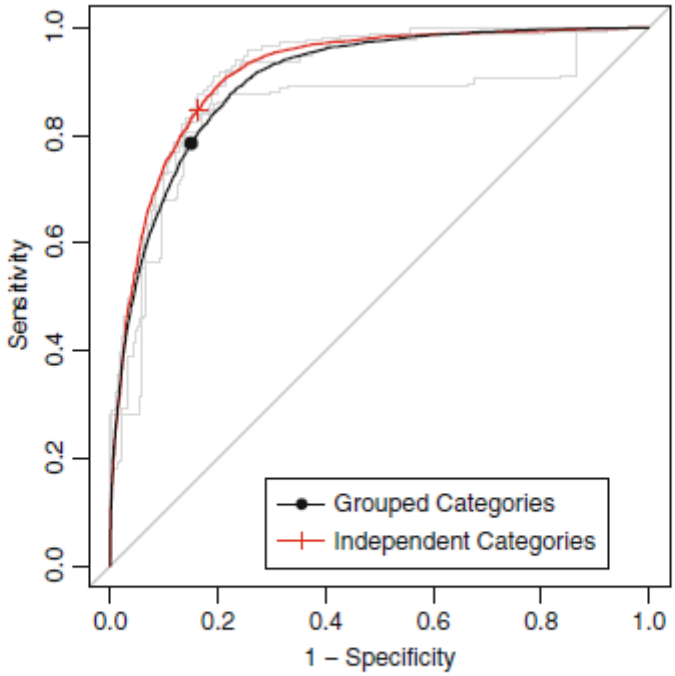

Fig. 14.8: The ROC curves for the random forest model. The area under the curve for independent categories was 0.92 and for the grouped category model the AUC was 0.9

Una diferencia es la capacidad para ponderar las clases de manera diferencial. Este aspecto del modelo se discute con más detalle en el Capítulo 16.

Se construyeron modelos de bosques aleatorios tanto para modelos de categorías independientes como para categorías agrupadas. El parámetro de ajuste, mtry, se evaluó en valores que van desde 5 hasta 1,000. Para las categorías independientes, el valor óptimo ajustado de mtry fue 100, y para las categorías agrupadas, el valor también fue 250. La Figura 14.8 presenta los resultados, y en este caso, las categorías independientes tienen un AUC ligeramente mayor (0.92) que el enfoque de categoría agrupada (0.9). El modelo de predictor binario también tiene una mayor sensibilidad (86.1% vs. 84.7%) pero una especificidad ligeramente menor (85.4% vs. 87.2%).

Para árboles individuales, la importancia de las variables se puede determinar mediante la agregación de la mejora en el objetivo de optimización para cada predictor. Para bosques aleatorios, el criterio de mejora (por defecto, suele ser el índice de Gini) se agrega en todo el conjunto para generar una medida general de importancia de la variable. Alternativamente, el impacto de los predictores en el conjunto se puede calcular utilizando un enfoque de permutación (Breiman 2000), como se discute en la Sección 8.5. Se han calculado valores de importancia de variables basados en la mejora agregada para los datos de subvenciones para ambos tipos de predictores, y los predictores más importantes se presentan en la Figura 14.15. La interpretación se deja al lector en el Ejercicio 14.1.

Los árboles de inferencia condicional también se pueden utilizar como aprendizaje base para los bosques aleatorios. Sin embargo, las implementaciones actuales de la metodología son computacionalmente pesadas para problemas del tamaño relativo de los datos de

subvenciones. Se explora una comparación del rendimiento de los bosques aleatorios utilizando árboles CART e árboles de inferencia condicional en el Ejercicio 14.3.

14.5 Boosting

Aunque ya hemos discutido el impulso en el contexto de la regresión, el método se desarrolló originalmente para problemas de clasificación (Valiant 1984; Kearns y Valiant 1989), en los cuales muchos clasificadores débiles (por ejemplo, un clasificador que predice marginalmente mejor que al azar) se combinaron en un clasificador fuerte. Hay muchas especies de algoritmos de impulso, y aquí discutimos los principales.

AdaBoost

A principios de la década de 1990, aparecieron varios algoritmos de impulso (Schapire 1990; Freund 1995) para implementar la teoría original. Freund y Schapire (1996) proporcionaron finalmente la primera implementación práctica de la teoría de impulso en su famoso algoritmo AdaBoost; se proporciona una versión intuitiva en el Algoritmo 14.2.

En resumen, AdaBoost genera una secuencia de clasificadores débiles, donde en cada iteración el algoritmo encuentra el mejor clasificador basado en los pesos de la muestra actuales. Las muestras que se clasifican incorrectamente en la k-ésima iteración reciben más peso en la (k + 1)-ésima iteración, mientras que las muestras que se clasifican correctamente reciben menos peso en la iteración subsiguiente. Esto significa que las muestras que son difíciles de clasificar reciben pesos cada vez mayores hasta que el algoritmo identifica un modelo que clasifica correctamente estas muestras. Por lo tanto, cada iteración del algoritmo debe aprender un aspecto diferente de los datos, centrándose en regiones que contienen muestras difíciles de clasificar. En cada iteración, se calcula un peso de etapa en función de la tasa de error en esa iteración. La naturaleza del peso de etapa descrito en el Algoritmo 14.2 implica que los modelos más precisos tienen valores positivos más altos y los modelos menos precisos tienen valores negativos más bajos.5 La secuencia general de clasificadores ponderados se combina luego en un conjunto y tiene un fuerte potencial para clasificar mejor que cualquiera de los clasificadores individuales.

1 Let one class be represented with a value of +1 and the other with a value of -1

2 Let each sample have the same starting weight $(1/n)$

3 **for** $k = 1$ *to* K **do**

4 Fit a weak classifier using the weighted samples and compute the kth model's misclassification error (err_k)

5 Compute the kth stage value as $\ln\left(\left(1 - err_k\right)/err_k\right)$.

6 Update the sample weights giving more weight to incorrectly predicted samples and less weight to correctly predicted samples

7 **end**

8 Compute the boosted classifier's prediction for each sample by multiplying the kth stage value by the kth model prediction and adding these quantities across k. If this sum is positive, then classify the sample in the +1 class, otherwise the -1 class.

Algorithm 14.2: AdaBoost algorithm for two-class problems

El aumento se puede aplicar a cualquier técnica de clasificación, pero los árboles de clasificación son un método popular para el aumento, ya que pueden convertirse en aprendices débiles al restringir la profundidad del árbol para crear árboles con pocas divisiones (también conocidos como troncos). Breiman (1998) proporciona una explicación de por qué los árboles de clasificación funcionan especialmente bien para el aumento. Dado que los árboles de clasificación son una técnica de bajo sesgo/alta varianza, el conjunto de árboles ayuda a reducir la varianza, produciendo un resultado con bajo sesgo y baja varianza. Trabajando a través del algoritmo AdaBoost, Johnson y Rayens (2007) mostraron que los métodos de baja varianza no pueden mejorarse significativamente mediante el aumento. Por lo tanto, métodos de aumento como LDA o KNN no mostrarán tanta mejora como métodos de aumento como redes neuronales (Freund y Schapire 1996) o Naïve Bayes (Bauer y Kohavi 1999).

Refuerzo Gradiente Estocástico

Como se mencionó en la Sección 8.6, Friedman et al. (2000) trabajaron para proporcionar una visión estadística del algoritmo AdaBoost. Para el problema de clasificación, mostraron que se podía interpretar como un modelo aditivo secuencial hacia adelante que minimiza una función de pérdida exponencial. Este marco condujo a generalizaciones algorítmicas como Real AdaBoost, Gentle AdaBoost y LogitBoost. Posteriormente, estas generalizaciones se integraron en un marco unificador llamado máquinas de refuerzo de gradiente, que se discutió previamente en el capítulo de árboles de regresión.

1 Initialized all predictions to the sample log-odds: $f_i^{(0)} = \log \frac{\widehat{p}}{1-\widehat{p}}$.

2 **for** *iteration* $j = 1 \ldots M$ **do**

3 Compute the residual (i.e. gradient) $z_i = y_i - \widehat{p}_i$

4 Randomly sample the training data

5 Train a tree model on the random subset using the residuals as the outcome

6 Compute the terminal node estimates of the Pearson residuals: $r_i = \frac{1/n \sum_i^n (y_i - \widehat{p}_i)}{1/n \sum_i^n \widehat{p}_i(1-\widehat{p}_i)}$

7 Update the current model using $f_i = f_i + \lambda f_i^{(j)}$

8 **end**

Algorithm 14.3: Simple gradient boosting for classification (2-class)

Similar al entorno de regresión, cuando se utilizan árboles como aprendiz base, el aumento de gradiente básico tiene dos parámetros de ajuste: la profundidad del árbol (o profundidad de interacción) y el número de iteraciones. Una formulación del aumento de gradiente estocástico modela una probabilidad de evento, similar a lo que vimos en la regresión logística, mediante

$$\widehat{p}_i = \frac{1}{1 + exp\left[-f(x)\right]},$$

Aquí tienes la traducción:

Donde f(x) es una predicción del modelo en el rango de [−∞, ∞]. Por ejemplo, una estimación inicial del modelo podría ser la probabilidad logarítmica de la muestra, F_i^(0) = log(p/(1-p)), donde p es la proporción de muestra de una clase del conjunto de entrenamiento.

Utilizando la distribución de Bernoulli, se muestra el algoritmo para el aumento de gradiente estocástico para dos clases en el Algoritmo 14.3 (Fig. 14.9). En este caso, el modelo de categoría independiente tiene un rendimiento mejor que el modelo de categoría agrupada en función del Área Bajo la Curva (ROC). Sin embargo, el número de árboles en cada modelo fue sustancialmente diferente, lo cual tiene lógica ya que el conjunto de predictores binarios es más grande que las categorías agrupadas.

Un examen de los perfiles de parámetros de ajuste para los predictores de categoría agrupada y de categoría independiente, mostrados en las Figs. 14.10 y 14.11, revela algunas contrastes interesantes. En primer lugar, el aumento de los predictores de categoría independiente tiene un rendimiento predictivo casi uniformemente mejor en comparación con el aumento de los predictores de categoría agrupada en todas las configuraciones de parámetros de ajuste. Este patrón es probablemente porque solo un valor para muchos de los predictores importantes de categoría agrupada contiene información predictiva significativa. Por lo tanto, los árboles que utilizan los predictores de categoría independiente pueden encontrar esa información más fácilmente, lo que impulsa el proceso de aumento. Dentro de los predictores de categoría

agrupada, aumentar el parámetro de contracción de manera casi uniforme degrada el rendimiento predictivo en la profundidad del árbol. Estos resultados implican que, para los predictores de categoría agrupada, el aumento obtiene la mayor parte de su información predictiva de un árbol inicial de tamaño moderado, lo cual se evidencia por Áreas Bajo la Curva (AUC) comparables entre un solo árbol (0.89) y el árbol aumentado óptimo (0.92).

El aumento de los predictores de categoría independiente muestra que a medida que aumenta el número de árboles, el rendimiento del modelo mejora para valores bajos de contracción y empeora para valores altos de contracción. Pero, ya sea que se seleccione un valor bajo o alto de contracción, cada enfoque encuentra el rendimiento predictivo máximo en un Área Bajo la Curva (ROC) de aproximadamente 0.94. Este resultado sugiere que, para estos datos, el aumento puede encontrar una configuración óptima bastante rápidamente sin necesidad de mucha contracción.

La importancia de las variables para el aumento en el entorno de clasificación se calcula de manera similar al entorno de regresión: dentro de cada árbol en el conjunto, se agrega la mejora basada en los criterios de división para cada predictor. Estos valores de importancia se promedian luego en todo el conjunto de aumento.

El usuario puede adaptar el algoritmo de manera más específica seleccionando una función de pérdida apropiada y su gradiente correspondiente (Hastie et al., 2008). La contracción puede implementarse en el último paso del Algoritmo 14.3. Además, este algoritmo puede incorporarse en el marco de trabajo de aumento de gradiente estocástico mediante la adición de un esquema de muestreo aleatorio antes del primer paso en el bucle interno. Detalles sobre este proceso se pueden encontrar en la Sección 8.6.

Para los datos de la subvención, se construyó una cuadrícula de parámetros de ajuste donde la profundidad de interacción varió de 1 a 9, el número de árboles varió de 100 a 2,000, y la contracción varió de 0.01 a 0.1. Esta cuadrícula se aplicó para construir un modelo de aumento donde las variables categóricas se trataron como categorías independientes y por separado como categorías agrupadas. Para el modelo de categoría independiente, el área óptima bajo la curva ROC fue 0.94, con una profundidad de interacción de 9, número de árboles 1,300 y contracción 0.01. Para el modelo de categoría agrupada, el área óptima bajo la curva ROC fue 0.92, con una profundidad de interacción de 7, número de árboles 100 y contracción 0.01 (ver Fig. 14.9). En este caso, el modelo de categoría independiente tiene un rendimiento mejor que el modelo de categoría agrupada en función de la curva ROC. Sin embargo, el número de árboles en cada modelo fue sustancialmente diferente, lo cual tiene lógica ya que el conjunto de predictores binarios es más grande que las categorías agrupadas. Un examen de los perfiles de parámetros de ajuste para los predictores de categoría agrupada y de categoría independiente, mostrados en las Figs. 14.10 y 14.11, revela algunas contrastes interesantes. En primer lugar, el aumento de los predictores de categoría independiente tiene un rendimiento predictivo casi uniformemente mejor en comparación con el aumento de los predictores de categoría agrupada en todas las configuraciones de parámetros de ajuste. Este patrón es probablemente porque solo un valor para muchos de los predictores importantes de categoría agrupada contiene información predictiva significativa. Por lo tanto, los árboles que utilizan los predictores de categoría independiente pueden encontrar esa información más fácilmente, lo que impulsa el proceso de aumento. Dentro de los predictores de categoría agrupada, aumentar el parámetro de contracción de manera casi uniforme degrada el rendimiento predictivo en la profundidad del árbol. Estos resultados implican que, para los predictores de categoría agrupada, el aumento obtiene la mayor parte de su información predictiva de un

árbol inicial de tamaño moderado, lo cual se evidencia por Áreas Bajo la Curva (AUC) comparables entre un solo árbol (0.89) y el árbol aumentado óptimo (0.92). El aumento de los predictores de categoría independiente muestra que a medida que aumenta el número de árboles, el rendimiento del modelo mejora para valores bajos de contracción y empeora para valores altos de contracción. Pero, ya