USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
FACULTADE DE MATEMÁTICAS

Traballo Fin de Grao

Árboles de regresión y clasificación

José Ángel Vicente Porres

Julio, 2022

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

GRAO DE MATEMÁTICAS

Traballo Fin de Grao

Árboles de regresión y clasificación

José Ángel Vicente Porres

Julio, 2022

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Trabajo propuesto

Área de Coñecemento: Estatística e Investigación Operativa
Título: Árbores de regresión e clasificación
Breve descrición do contido
As árbores de decisión son unha metodoloxía de aprendizaxe estatística supervisada que se emprega tanto en problemas de clasificación (cando a variable resposta é cualitativa) coma en problemas de regresión (cando a variable resposta é cuantitativa). As árbores constrúense de forma recursiva partindo o espazo das covariables e axustando un modelo de predición en cada paso. Entre os algoritmos máis destacados para a construción de árbores de decisión está o algoritmo CART (Classification and Regression Trees) O obxectivo deste traballo é que o alumno ou a alumna faga unha revisión destes métodos desde o punto de vista teórico e estude a súa implementación e aplicación a datos reais
Recomendacións
Outras observacións

Índice

Resumen

Los árboles de decisión constituyen una de las metodologías más extendidas en el ámbito del aprendizaje estadístico. Se pueden aplicar tanto a problemas de regresión como de clasificación y se caracterizan por su simplicidad y fácil interpretación. En este trabajo se presenta una revisión de la metodología básica necesaria para la construcción de árboles de decisión. Explicaremos paso por paso el proceso de construcción de un árbol, su expansión y podado hasta alcanzar un tamaño adecuado, así como la validación final del modelo. Además, haremos una breve revisión de otras técnicas más complejas basadas en árboles de decisión, que pueden mejorar la capacidad predictiva de los mismos.

Abstract

Decision trees are one of the most spread methodologies in statistical learning. They have been applied in both regression and classification problems, and have become popular thanks to it's simplicity and easy interpretation. This work presents a revision in the basic methodology needed to build decision trees. We will explain how to build a tree, from the expansion and pruning to reach it's appropriate size, to the validation of the final model. Moreover, we will review some other more complex techniques based in decision trees, which allows the creation of far more accurate models.

Introducción

Con la introducción de bases de datos cada vez más extensas, resulta indispensable la creación de modelos capaces de extraer toda la información posible de manera rápida, eficaz y sencilla. A lo largo de los años ochenta y noventa, y gracias a los trabajos de Breiman L. y Friedman J., entre muchos otros, se hizo especialmente popular la aplicación de árboles de decisión para el tratamiento de dichos datos.

La separación de los datos a lo largo de las ramas de un árbol permite establecer de manera sencilla relaciones entre las diversas variables que se observan en un estudio. La gran fuerza y eficacia con la que estos modelos actúan en casos reales, los han convertido en pocos años en una de las herramientas fundamentales de la estadística moderna, la ciencia de datos y la inteligencia artificial.

Por si solo, un árbol de decisión es un modelo bastante sencillo, que se caracteriza además por su fácil interpretación. A pesar de su simplicidad, el uso de árboles de decisión en aprendizaje estadístico se ha extendido de forma generalizada, gracias además a la posibilidad de combinarlos a través de métodos más complejos, también llamados métodos de ensamble. En este trabajo se estudiaran aquellos que actualmente cuentan con una mayor popularidad, Bagging, Boosting y Random Forest.

Qué es un árbol

Un árbol es un tipo de grafo ordenado, caracterizado por su estructura simple repetida de manera recursiva. Cada uno de los nodos de un árbol puede estar conectado a varios hijos, y con cada paso extiende sus ramas cada vez más. Otra característica de este tipo de estructuras es que solo tienen un nodo primero, que no es hijo de ningún otro, denotado como raíz. El resto de nodos solo tienen un único antecedente o padre. Bajo estas condiciones, el camino que conecta un nodo con cualquiera de su descendencia es único.

En este primer capítulo presentaremos los conceptos básicos sobre árboles. A lo largo del

trabajo nos centraremos en árboles binarios, en los que cada nodo tiene como mucho dos hijos. Esto se debe a que son los más sencillos para trabajar, y el añadir un mayor número de ramas no proporciona ningún beneficio a la hora de construir modelos de regresión o clasificación.

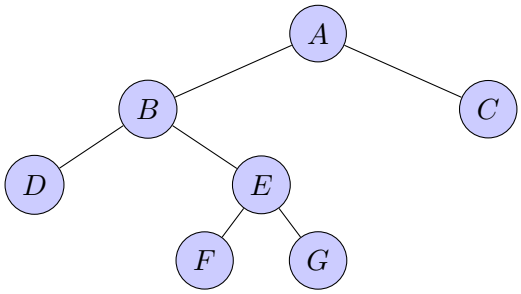


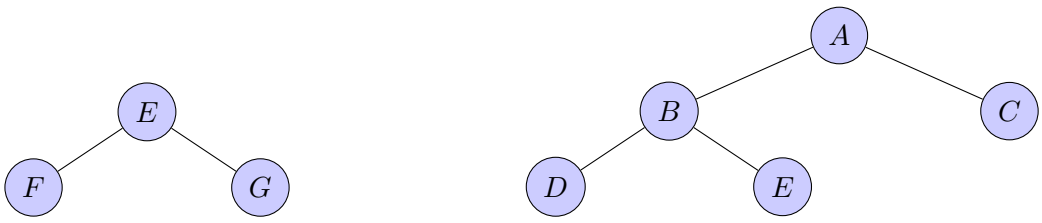
Figura 1: Ejemplo de un árbol binario.

Como ejemplo se muestra en la Figura 1 un árbol, que denotaremos T. En dicho árbol A es el nodo raíz, mientras que B y C son sus hijos. Del mismo modo, D y E son los hijos de B, y F, G son los hijos de E. A partir de esta construcción, resulta intuitivo llamar a los hijos de un nodo t, como el izquierdo y el derecho. Por esta razón se usará la notación t_i y t_d cuando sea necesario identificarlos.

Dado un nodo t, se define una rama T_t, como el subárbol con t de raíz, y que se extiende hacia abajo con la misma estructura que el árbol original. Recuperando el ejemplo anterior, T_E estaría compuesto por E, F y G como se muestra en la Figura 2a.

Se llamará nodo terminal a todo aquel que no tenga descendencia. En el árbol de ejemplo con el que estamos trabajando, los nodos terminales serían C, D, F y G. Se denotará el conjunto de todos los nodos terminales como $\hat{T}$. De la misma forma, se usará la notación $|\hat{T}|$ para determinar el número de nodos terminales del árbol T.

Para la construcción de modelos de regresión y clasificación, también será necesario definir el proceso de podado. De un árbol T se poda la rama T_t al descartar toda la descendencia del nodo t. Este nuevo árbol se denota por $T - T_t$, véase la Figura 2b.



(a) T_E (b) $T - T_E$

Figura 2: Rama E y proceso de podado de la rama E.

Este tipo de grafos es especialmente usado como estructura para guardar información. La forma en la que los nodos están organizados favorece la construcción de algoritmos capaces de ordenar y buscar información de manera eficiente en grandes bases de datos. Sin embargo, en este trabajo nos centraremos en otra de las aplicaciones de este tipo de objetos. Se aprovechará la forma en la que se puede relacionar la información en este tipo de grafos para generar modelos capaces de realizar predicciones sobre nuevos elementos a estudiar.

Capítulo 1

Reglas de clasificación y predicción

1.1. Reglas de clasificación

Supongamos que nos encontramos en un hospital y nos encargan la creación de un modelo capaz de identificar qué pacientes tienen un mayor riesgo de padecer una enfermedad coronaria. Se tiene que los procedimientos que permiten detectar este tipo de enfermedades de manera fiable suelen ser bastante complejos, necesitando en muchos casos la realización de un TAC de gran parte del torso. La obtención y procesado de este tipo de imágenes suele ser lenta, complicada y costosa, por lo que determinar de manera eficiente que pacientes necesitan este tipo de pruebas con mayor urgencia, puede ser un paso fundamental en su tratamiento.

En este tipo de procedimientos, se etiqueta cada uno de los pacientes. Para este ejemplo, diremos que con un 0, si se cree que está fuera de peligro, o 1 si es un caso de riesgo. Este será un problema de clasificación binario, ya que solo hay dos clases entre las que discernir. Si se quiere un modelo más completo, se puede intentar realizar una clasificación más precisa, no solo estudiando si un paciente está sano o no, sino estimando el grado de peligro en el que se encuentra. Formalmente, se llamará conjunto de clases $\mathcal{C} = \{1, 2, 3, \ldots, J\}$ a las distintas categorías que se consideran en el modelo.

Es necesario disponer de información sobre la que realizar dicha clasificación. Por ejemplo, es sabido que la probabilidad de padecer una enfermedad coronaria aumenta enormemente en personas mayores, por lo que conocer la edad del paciente puede ser crucial en la realización un estudio satisfactorio. Otros datos que se podrían utilizar serían antecedentes familiares, peso, colesterol en sangre, síntomas, etc. Toda esta información sobre un paciente se guarda en un vector de medidas $\vec{x}$. Este se estudiará como una variable aleatoria que toma valores siguiendo una distribución del espacio muestral Ω.

Será necesario tener acceso a una base de datos para poder realizar el modelo. Esta podría ser el historial del hospital, ya que contiene información sobre los antiguos pacientes. En este caso, se debe saber el nivel de riesgo real, y los datos suficientes como para construir su vector de medidas. Todo esto se recoge en el conjunto de aprendizaje $\mathcal{L} = (\vec{x}_i, Y_i)_{i=1...N}$, compuesto por N parejas $\vec{x}_i \in \Omega$, $Y_i \in \mathcal{C}$. En el fondo, el proceso de construcción de un modelo se basa en un estudio sobre $\mathcal{L}$, por lo que es necesario asegurarse de que representa una buena muestra del medio.

Definición 1.1. Una regla de clasificación o clasificador d, es una aplicación que a cada punto del espacio muestral Ω, le asigna una de las clases de $\mathcal{C}$.

Regresando al ejemplo del hospital, a partir del conjunto de aprendizaje $\mathcal{L}$, se construirá un clasificador d. Este usará el vector de medidas $\vec{x}$ de un paciente para predecir si está sano, o que enfermedad puede tener. Esto permitiría detectar de manera sencilla aquellos individuos con un mayor riesgo de sufrir la enfermedad estudiada, y por tanto, facilitar su diagnosis.

1.2. Ratios de clasificación incorrecta

Una vez definido lo que es un clasificador d, surge un nuevo problema que será necesario abordar, ya que su resolución será fundamental para la construcción de modelos precisos. ¿Cómo se sabe si d es eficiente? De manera intuitiva, un clasificador es mejor qué otro si es capaz de cometer menos errores al asignar etiquetas a nuevos sujetos. Con el fin de dar un valor numérico a la calidad de un modelo de este tipo, se definen los ratios de clasificación incorrecta.

En una primera aproximación a este problema, se puede definir el riesgo como la probabilidad de que el clasificador cometa un error, $R^*(d) = P[d(\vec{x}) \neq Y]$. Donde $\vec{x} \in \Omega$ representa el vector de medidas de un sujeto, e Y la clase a la que dicho sujeto pertenece en realidad.

Hay que tener en cuenta que la construcción de un clasificador d se lleva a cabo a partir de la experiencia pasada, en forma de conjunto de aprendizaje $\mathcal{L}$. Debido a esta razón, en la gran mayoría de casos será imposible calcular este riesgo, ya que no se dispone de la distribución real del medio. Se proponen por tanto estimadores de esta probabilidad, con el fin de encontrar una medida de la calidad de un clasificador a partir de la información contenida en $\mathcal{L}$

Definición 1.2. Sea d un clasificador, se define el ratio de clasificación incorrecta, $R(d)$, como la proporción de observaciones del conjunto de aprendizaje $\mathcal{L}$ que d no es capaz de clasificar correctamente.

Así, a partir del conjunto de aprendizaje $\mathcal{L} = (\vec{x}_i, Y_i)_{i=1...N}$, y considerando I la función indicador tal que $I(a) = 1$ si la condición a se cumple, y 0 en otro caso. Se calculará por tanto

el ratio de clasificación incorrecta de un modelo como

$$R(d) = \sum_{i=1}^{N} I(d(\vec{x}_i) \neq Y_i)/N$$

Para la construcción de clasificadores, el último ratio es realmente bueno, ya que permite calcular rápidamente la calidad del modelo a medida que este se va construyendo. Sin embargo, se está usando la misma información para construir y para testear el clasificador. Esto suele llevar a modelos demasiado complejos, que pierden información sobre las relaciones que existen entre $\vec{x}$ e Y debido a problemas de sobreajuste.

Con el fin de evitar estos problemas, se suelen aplicar diversos métodos capaces de obtener estimaciones más precisas del error cometido. Sin entrar en detalles, en este trabajo se empleará validación cruzada, que trabaja sobre diversas particiones del conjunto de aprendizaje para comprobar cuánta información se pierde por sobreajuste.

Con el fin de obtener una estimación realista del error cometido, se suele aislar una parte del conjunto de aprendizaje. Construyendo aleatoriamente la partición de $\mathcal{L}$ formada por $\mathcal{L}_e$, conjunto de entrenamiento, y $\mathcal{L}_v$, de validación. De la misma forma, se definirán los ratios de clasificación incorrecta R^e y R^v, definidos de forma similar a R, pero restringido a los conjuntos de entrenamiento y de validación respectivamente.

En los apartados del trabajo que se centren en la construcción del modelo, y por ende, validar pase a un segundo plano, se usará únicamente la notación $\mathcal{L}$ y R para el desarrollo de la teoría.

1.3. Estimador de la probabilidad de clases

Hasta ahora, en el problema del hospital, a cada paciente se le clasificaba como sano o enfermo, como mucho dándole un grado de peligrosidad. Una aproximación más realista, sería construir un modelo que como respuesta diese una probabilidad de que el paciente padezca dicha enfermedad, o no. De esta forma, el médico puede cotejar los resultados y tener más información a la hora de plantear un tratamiento.

Para este tipo de modelos, al vector de medidas $\vec{x}$ se le deberá asignar un valor para cada una de las clases del estudio $\mathcal{C} = \{1, 2, 3, \ldots, J\}$. De tal forma que la respuesta se de como un vector $\vec{d(x)} = (d(1/\vec{x}), d(2/\vec{x}), \ldots, d(J/\vec{x}))$, donde $d(j/\vec{x})$ es una estimación $p(j/\vec{x})$, la probabilidad de que $\vec{x}$ pertenezca a la clase j. Por esta razón se debe cumplir que $d(j/\vec{x}) \in [0, 1]$ y $\sum_{j \in \mathcal{C}} d(j/\vec{x}) = 1$

Definición 1.3. Un estimador de probabilidad de clases es una aplicación d, que a cada clase del conjunto $\mathcal{C}$, y a cada vector $\vec{x}$ del espacio muestral Ω, les asigna un valor real $d(j/\vec{x}) \in [0, 1]$,

tal que $\sum_{j\in\mathcal{C}} d(j/\vec{x}) = 1$

Se define la precisión del modelo como $R^*(d) = E[\sum_j (p(j/\vec{x}) - d(j/\vec{x}))^2/\vec{x} \in \Omega]$, la esperanza entre las probabilidad real y las estimadas al cuadrado. Al igual que en la distribución de clases, como se desconoce la distribución real del medio, no es posible obtener un cálculo de dicha precisión.

A partir del conjunto de aprendizaje $\mathcal{L} = (\vec{x}_i, Y_i)_{i=1}$, se propone como estimador de la precisión

$$R(d) = \sum_{i=1}^{N} \sum_{j\in\mathcal{C}} (I(i=j) - d(j/\vec{x}_i))^2. \tag{1.1}$$

Esta estimación se obtiene de suponer que la probabilidad $p(Y_i/\vec{x}_i)$ es igual a 1. De la misma forma $p(j/\vec{x}_i) = 0$ para el resto de clases posibles.

1.4. Reglas de predicción

En el ejemplo del hospital, a cada paciente se le clasifica en uno de los niveles de $\mathcal{C}$, estos son los clasificadores. También se puede dar el caso en el que en vez de a una clase, el modelo deba realizar una estimación sobre una magnitud continua. A este tipo de modelos se les llama predictores, ya que se encargan de predecir el valor de una variable.

Definición 1.4. Una regla de predicción o predictor d, es una aplicación que a cada punto del espacio muestral Ω le asigna un valor de $\mathbb{R}$.

A lo largo de este trabajo se denotará con d tanto a los clasificadores como a los predictores. El objetivo final es construir una metodología general para ambos tipos de modelos, por lo que diferenciarlos ahora podría llevar a confusiones más adelante.

Al igual que en el caso anterior, se dispone de un conjunto de aprendizaje $\mathcal{L}$, del que se espera obtener la información suficiente como para realizar predicciones precisas. Sin embargo, en este tipo de modelos no tiene sentido hablar de reglas de clasificación incorrecta, ya que será raro el caso en el que un resultado sea exacto. Por esta razón se suele preferir el uso de un estimador de la esperanza del error cuadrático $R^*(d) = E[(d(\vec{x}) - Y)^2]$ con el fin de medir la calidad del modelo.

Definición 1.5. Sea d un predictor, se define el error de mínimos cuadrados $R(d)$ como la media del error al cuadrado cometido de las diversas observaciones sobre el conjunto de aprendizaje $\mathcal{L}$.

$$R(d) = \sum_{i=1}^{N} (d(\vec{x_i}) - Y_i)^2 / N$$

En este tipo de modelos también se puede separar el conjunto de aprendizaje entre entrenamiento y validación con el fin de obtener estimaciones del error más precisas.

Se tiene que el uso de R para definir tanto el ratio de clasificación incorrecta como el error por mínimos cuadrados. Este abuso de notación permite la formulación de metodología para la construcción de árboles de decisión independientemente de su tipo.

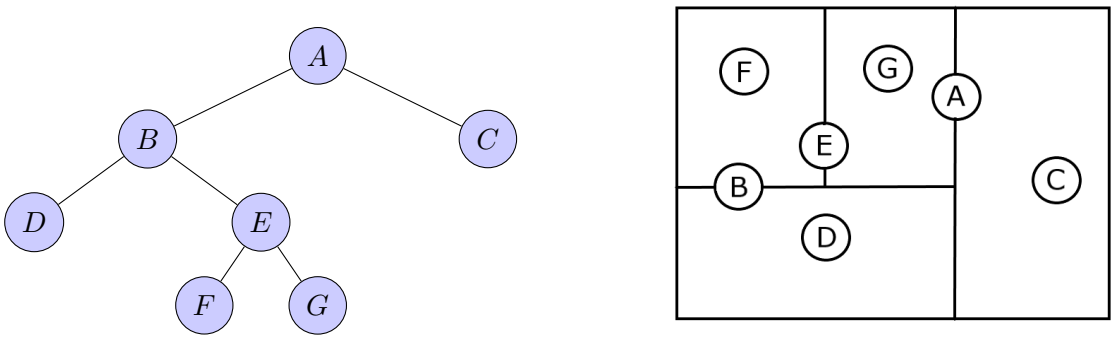
Capítulo 2

Modelos en forma de árbol

La idea que está detrás de usar árboles binarios para la construcción de modelos está en la recursividad que los caracteriza. Cada uno de los nodos del árbol puede estar asociado a otros dos, sus hijos, que a su vez pueden estar unidos a dos más.

En el caso concreto de árboles de decisión, se asocia cada uno de los nodos a una región del espacio muestral Ω. De tal forma que la raíz representa el total, y cada separación es una división del espacio en dos. Este proceso de separar y repartir se puede continuar de manera repetida, aumentando en cada paso el tamaño del árbol.

Más adelante se explicará con detalle la forma en la que se decide la división óptima del espacio. Como introducción, se busca aquella separación que minimice R a partir de la información del conjunto de aprendizaje. La idea es conseguir agrupar en el mismo nodo aquellas zonas del espacio con cualidades similares, para clasificarlas de la misma forma.



(a) Ejemplo simple de árbol binario

(b) Relación con Ω

Figura 2.1: Ejemplo de separación del espacio de medidas a partir de un árbol

En la Figura 2.1 se puede observar un pequeño ejemplo de cómo se emplea un árbol para

dividir el espacio muestral en distintas partes. En este caso, Ω está representado por el rectángulo de la derecha, por notación se considerará que es un espacio bidimensional continuo y $\vec{x} = (x_1, x_2)$. De esta forma, se tiene que $A = \Omega$, el cual se divide entre B y C, B se separa entre E y D, y por último E, entre F y G.

En este caso, todas las separaciones se centran en solo una de las variables, lo que permite que su descripción sea bastante sencilla. Por ejemplo, $B = \{\vec{x} \in A / x_1 < c\}$. Se puede utilizar esta misma estructura para describir cada uno de los subespacios en este ejemplo. Otras posibles formas de dividir el espacio sería mediante el uso de hiperplanos, $\{\vec{x} \in \Omega / \vec{a}^t \vec{x} < c\}$, o bolas $\{\vec{x} \in \Omega / \|\vec{a} - \vec{x}\| < c\}$. A la hora de implementar el modelo, estas dos últimas descripciones son computacionalmente más costosas, por lo que solo se utilizan en el caso de que haya un conocimiento a priori que favorezca su aplicación.

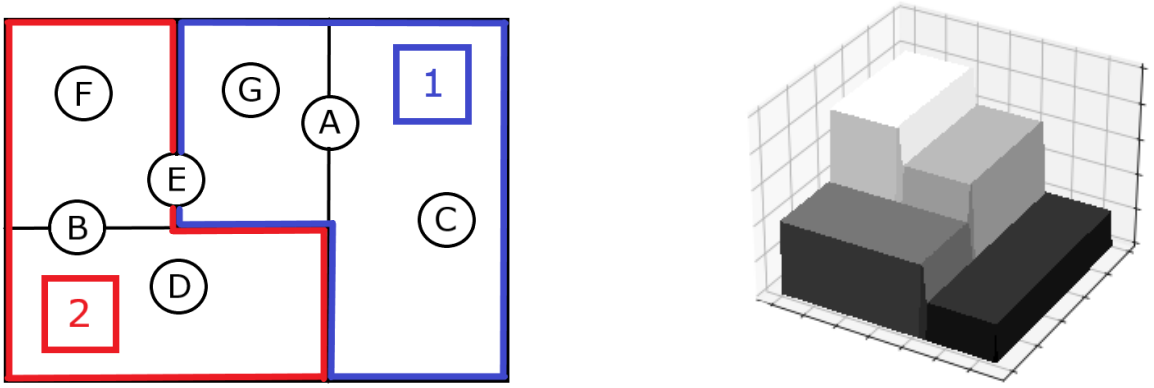
Por lo general, las separaciones se suelen describir en forma de preguntas s. En el caso del ejemplo anterior, se definiría $s_A(\vec{x}) = ¿\vec{x} < c?$, con lo que quedaría $B = \{\vec{x} \in A / s_A(\vec{x}) = \text{sí}\}$, y $C = \{\vec{x} \in A / s_A(\vec{x}) = \text{no}\}$. Esta forma de expresar las divisiones del espacio muestral como un objeto en si mismo, es útil a la hora de construir modelos.

Por como se han construido estos árboles, el conjunto de los subespacios asociados a los nodos terminales son un recubrimiento disjunto del espacio muestral. De esta forma, en el ejemplo anterior, Ω está formado por C, D, G y F. Para completar el modelo d, a cada uno de los nodos terminales se le asigna un valor, en el caso de ser un clasificador, una de las clases de $\mathcal{C}$, y para un predictor, un número real.

Una vez hecho todo esto, el funcionamiento del modelo es bastante sencillo. Los nuevos elementos a estudiar se disponen en la raíz. A estos se les realiza la pregunta s del nodo, y según como respondan, bajan a uno de los hijos. Este proceso se repite a lo largo del árbol, hasta que llega a uno de los nodos terminales. En este punto, se concluye que $\vec{x}$ se encuentra en dicho subespacio terminal, por lo que al sujeto se le asigna el mismo valor que al nodo.

Siguiendo con el ejemplo anterior, en la Figura 2.2 se construye un modelo de clasificación y uno de predicción a partir de la división del espacio ya vista. A la izquierda, se considera un clasificador de dos clases $\mathcal{C} = \{1, 2\}$. Los subespacios G, C se asignan a la clase 1, y los otros dos, F, D, a la clase 2. De la misma forma, se podría definir un recubrimiento de Ω a partir de las clases, $A_1 = G \cup C$ y $A_2 = F \cup D$. Por otro lado, a la derecha se muestra un predictor, en el que a cada uno de los subespacios terminales se les asigna un valor real distinto.

En estos últimos ejemplos se puede ver que la principal diferencia entre un árbol de clasificación y uno de predicción reside en el último paso, la asignación de un valor a un nodo terminal. Antes de empezar a hablar de la manera en la que se divide el espacio a partir este tipo de estruc-



(a) Clasificador (b) Predictor

Figura 2.2: Ejemplo de construcción de modelos a partir de un árbol

turas, es conveniente aclarar las diferencias técnicas entre clasificadores y predictores. De esta forma se espera poder construir toda clase de modelos dejando de lado a que tipo pertenecen.

2.1. Árboles de Clasificación

En la construcción de un modelo se llega hasta A_t, subespacio de Ω asociado a cierto nodo terminal t, de tal forma que todo vector de medidas $\vec{x} \in A_t$, se clasifica como $j \in \mathcal{C} = \{1, 2, 3 \dots J\}$. Se busca la forma óptima de realizar dicha asignación a partir de la información de $\mathcal{L}_t$, conjunto de aprendizaje restringido a esta región.

Sea un nuevo elemento dispuesto para ser clasificado por d, se define $\pi(j)$ la probabilidad de que pertenezca a la clase j sin ningún conocimiento a mayores del sujeto. De esta forma se define $\{\pi(j)\}_{j\in\mathcal{C}}$ el conjunto de probabilidades a priori. Estas representan una distribución ideal del conjunto muestral, por lo que no se suele tener información precisa sobre ellas. A la hora de construir modelos, es trabajo del analista estimar dichas probabilidades por $\{\hat{\pi}(j)\}_{j\in\mathcal{C}}$. Generalmente se suele recurrir al conjunto de aprendizaje para asignarle un valor a dicha magnitud, tomando $\hat{\pi}(j)$ como la proporción de elementos de $\mathcal{L}$ que pertenecen a la clase j.

La razón por la que se introduce estas probabilidades a priori π, es para paliar los casos en los que $\mathcal{L}$ no sea representativo del medio que se va a estudiar. Por ejemplo, en el hospital, tiene sentido realizar el entrenamiento con muchos informes de pacientes con síntomas raros. De esta forma, el modelo se blinda ante estos síntomas a costa de que los comunes tengan una menor representación.

Para trabajar con el conjunto de aprendizaje $\mathcal{L}$, se tendrá que está compuesto por N elemen-

tos, de los cuales N_j pertenecen a cada una de las clases j, y N^t a cada nodo t. De la misma forma, se tendrá que hay N_j^t elementos de $\mathcal{L}$ que pertenezcan a la clase j y al nodo t.

A la hora de construir el modelo, es necesario minimizar la probabilidad de cometer un error. Para ello, a cada nodo terminal t se le debe asignar la clase j más probable. Esta se describe como la probabilidad condicionada $p(j/t)$. Se puede simplificar por el teorema de Bayes a $p(j/t) = p(j,t)/p(t)$, donde $p(j,t)$ representa la probabilidad de la intersección (o probabilidad conjunta), y $p(t)$ la del nodo t.

Primero se estima $p(j,t)$, la probabilidad de que un elemento arbitrario pertenezca a la clase j y al nodo t. Con este fin se aplica una vez más el teorema de Bayes, $p(j,t) = p(t/j) \cdot p(j)$. La probabilidad de pertenecer a la clase j se estima por $\hat{\pi}(j)$, mientras que la condicionada se extrae del conjunto de aprendizaje, $\hat{p}(t/j) = N_j^t/N_j$. De esta forma se obtiene finalmente la estimación

$$\hat{p}(j,t) = \hat{\pi}(j) \cdot N_j^t/N_j.$$

La estimación de la probabilidad del nodo t se realizará de manera más sencilla. Se parte de que $p(t) = \sum_{j\in\mathcal{C}} p(j,t)$ y así se estima $\hat{p}(t) = \sum_{j\in\mathcal{C}} \hat{p}(j,t)$. Por último, se puede obtener una estimación de la probabilidad de que un elemento que llegue al nodo t, pertenezca a la clase j, como

$$\hat{p}(j/t) = \hat{p}(j,t)/\hat{p}(t).$$

Bajo estas definiciones, se tendrá que $\sum_{j\in\mathcal{C}} \hat{p}(j/t) = 1$, ya que todo sujeto clasificado por el nodo t deberá pertenecer necesariamente a una de las clases de $\mathcal{C}$. Por tanto resulta conveniente asignar al nodo t, la clase j^* que maximice la estimación de las probabilidades $\hat{p}(j^*/t) = \text{máx}_{j\in\mathcal{C}}\, \hat{p}(j/t)$

Se recuerda que la calidad de un clasificador viene dada por el ratio de clasificación incorrecta 1.2. Considerando que t es clasificado de manera óptima, se puede definir el ratio de error de un nodo como $r(t) = 1 - \hat{p}(j^*/t)$. Sea $\hat{T}$ el conjunto de los nodos terminales, se estimará el error de clasificación incorrecta de un árbol de clasificación T a partir de la suma de los errores de cada una de sus hojas

$$R(T) = \sum_{t\in\hat{T}} \hat{p}(t) r(t) = \sum_{t\in\hat{T}} R(t).$$

Puede darse el caso de que esta definición de ratio de clasificación incorrecta de un modelo en forma de árbol sea demasiado sencilla para la construcción de un modelo. Por ejemplo, en una consulta, considerar equivocadamente que un paciente está sano, es un error mucho más grave que predecir una enfermedad, y comprobar en pruebas posteriores que ha sido un falso positivo. Con el fin de incorporar este tipo de consideraciones al modelo, se definen los costes de error de clasificación $C(i/j)$, como un castigo por clasificar como i, un elemento de la clase j. Estos costes deben ser no negativos, y cumplir $C(j/j) = 0$.

Definición 2.1. Sea t un nodo de un árbol de clasificación, se define $j^*(t)$ clase del nodo a aquel elemento $i \in \mathcal{C}$ que minimice $\sum_{j\in\mathcal{C}} C(i/j)\hat{p}(j/t)$

Bajo esta definición, el ratio de clasificación incorrecta de un nodo vendrá dado por

$$r(t) = \min_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}} C(i/j)\hat{p}(j/t) = \sum_{j\in\mathcal{C}} C(j^*/j)\hat{p}(j/t).$$

Se observa que a pesar de tener una definición distinta, se mantiene el error de clasificación de un árbol como la suma de los errores en las hojas

$$R(T) = \sum_{t\in\hat{T}} \hat{p}(t)r(t) = \sum_{t\in\hat{T}} R(t).$$

2.1.1. Árboles de probabilidad de clases

En este caso, la construcción de los modelos deriva directamente de los árboles de clasificación. En el apartado anterior, se explica como obtener $\hat{p}(j/t)$, estimación de la probabilidad de que un elemento del nodo t pertenezca a la clase j. Resulta natural definir entonces $d(j/\vec{x}) = \hat{p}(j/t)$, siendo t el nodo terminal al que $\vec{x}$ desciende.

Trabajando sobre la ecuación 1.1, se tiene que el error estimado en cada uno de los elementos del conjunto de aprendizaje $\mathcal{L} = (\vec{x}_i, Y_i)_i$ viene dado por

$$\sum_j (I(j = Y_i) - d(j/\vec{x}_i))^2 = (1 - \hat{p}(Y_i/t))^2 + \sum_{j\neq Y_i} \hat{p}^2(j/t) = 1 - 2\hat{p}(Y_i/t) + S_t,$$

donde t es el nodo al que pertenece $(\vec{x}_i, Y_i)$, y S_t viene dado por $\sum_j \hat{p}^2(j/t)$. Para conseguir una aproximación del error cometido en cada uno de los nodos, se usa que $\hat{p}(j/t)$ es una estimación de la distribución de probabilidades de Y_i para integrar sobre las clases $j \in \mathcal{C}$

$$r(t) = \sum_i^N (1 - 2\hat{p}(Y_i/t) + S_t) \approx \sum_{j\in\mathcal{C}} (1 - 2\hat{p}(j/t) + S_t)\hat{p}(j/t).$$

Aplicando que por construcción $\sum_j \hat{p}(j/t) = 1$, la ecuación anterior se reduce a

$$r(t) = 1 - S_t = 1 - \sum_j \hat{p}^2(j/t). \tag{2.1}$$

Debido a la gran similitud que tienen este tipo de modelos con los clasificadores, muchos programas suelen construir únicamente estimadores de probabilidad, para posteriormente asociar cada nodo a aquella clase con una mayor estimación.

2.2. Árboles de Predicción

En el caso del que se este construyendo un modelo de predicción en vez de uno de clasificación, no tiene sentido hablar de probabilidades a priori ni de errores de clasificación incorrecta. Sin embargo sí que es necesario determinar que valor se le asigna a un nodo terminal t bajo su conjunto de aprendizaje restringido $\mathcal{L}_t$.

En un modelo de predicción, se usa el error cuadrático medio 1.5 como un estimador de $E[(d(\vec{x}) - Y)^2/\vec{x} \in \Omega]$. Estudiando la hoja t asociada al subespacio $\Omega_t \subset \Omega$, se tendrá que d realizará la misma predicción para todo $\vec{x}$. Aplicando que la media $\mu(t) = E[Y/\vec{x} \in \Omega_t]$ minimiza $E[(d(\vec{x}) - Y)^2/\vec{x} \in \Omega_t]$, se deduce que el valor óptimo que se puede asignar al nodo t será la media muestral $\bar{\mu}(t) = \sum_i Y_i/N^t$

Definición 2.2. Sea t un nodo terminal de un árbol de predicción T, se define $\nu(t)$ predicción del nodo a la estimación de la esperanza de la variable respuesta en el subespacio Ω_t. Este se calcula a partir de la media muestral $\bar{\mu} = \sum_i Y_i/N^t$

Para el estudio del error cometido, se utiliza el ratio sobre el conjunto restringido $\mathcal{L}_t$

$$r(t) = \sum_{i \in \mathcal{L}_t} (Y_i - \bar{\mu}(t))^2/N^t$$

Como un estimador de la varianza del nodo t, $\sigma^2(t) = Var(Y/\vec{x} \in \Omega_t) = E[(Y - \mu(t))^2/\vec{x} \in \Omega_t]$. De esta forma, si se calcula la probabilidad de que un elemento llegue a t como el ratio de los elementos del conjunto de entrenamiento, $\hat{p}(t) = N^t/N$, se puede aproximar el error cometido en todo el árbol por

$$R(T) = \sum_{t \in \hat{T}} \hat{p}(t) r(t) = \sum_{t \in \hat{T}} R(t).$$

2.3. Ejemplos

Presentamos a continuación un pequeño ejemplo de la forma en la que se puede usar un árbol con el fin de construir un clasificador. De este modo ilustraremos con un caso práctico cómo el modelo separa la información para realizar predicciones sobre la misma.

Se construirá un clasificador sobre *"Wine Data Set"*, conjunto de datos disponible en [7]. En él se recoge información sobre una serie de vinos de tres cultivos diferentes de la misma región de Italia. Se centra en su composición química, con datos sobre la cantidad de ciertas substancias como alcohol, ceniza y un serie de proteínas y aminoácidos. Se pude consultar información más detallada sobre el estudio en [7].

Para la construcción y presentación de árboles, se usará el paquete *"rpart"* de R, véase [13], ya que ofrece una gran versatilidad a la hora de acotar parámetros, perfecto para un ejemplo como este.

Con el fin de realizar un estudio más preciso, se separó la base de datos en dos partes, un conjunto de entrenamiento y otro de validación. De esta forma el modelo solo se construye con el 80 % de los datos escogidos de manera aleatoria, y el resto se utiliza para testearlo.

En la Figura 2.3 se presenta el árbol construido a partir del conjunto de entrenamiento. En cada nodo se ofrece información sobre el cultivo por el que es clasificado, la probabilidad estimada condicionada, y el porcentaje de elementos del conjunto de entrenamiento contenidos.

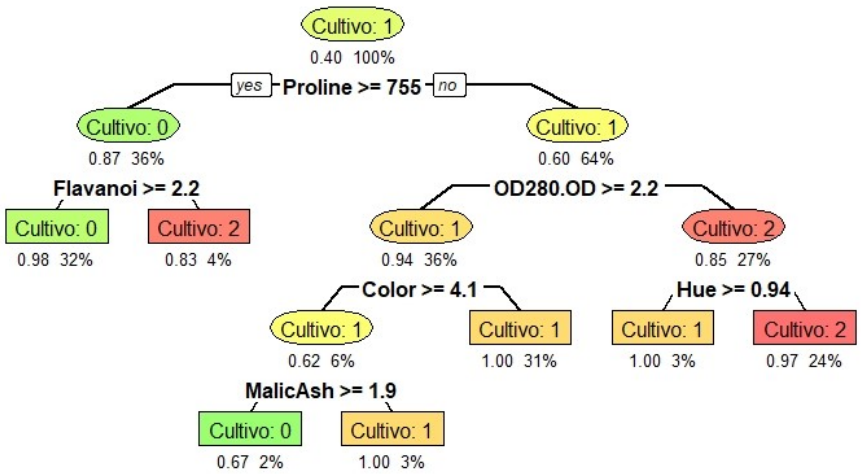


Figura 2.3: Árbol de clasificación final

Se puede observar que la estimación de la probabilidad es elevada en los nodos terminales. Esto es un indicador de que están formados en su mayoría por elementos de un solo cultivo, dato que se ve corroborado con el ratio $R^e = 0,028$. Al realizar la estimación del riesgo a partir del conjunto de validación, se obtiene $R^v = 0,147$. Obtener errores tan bajos a partir de un modelo tan simple indica que los datos se adaptan bastante bien a este método.

Las separaciones de los nodos 1 y 3, muestran que la concentración del aminoácido prolina y de las proteínas OD280/OD315 en los vinos, son buenos indicadores del cultivo al que pertenecen. Por esta razón en la Figura 2.4 se presenta la distribución del conjunto de entrenamiento con respecto a estas dos variables.

Se puede observar que los vinos del cultivo 1, tienden a tener una cantidad de prolina muy superior. La linea vertical que separa el espacio en dos muestra la separación del nodo 1 *Proline*

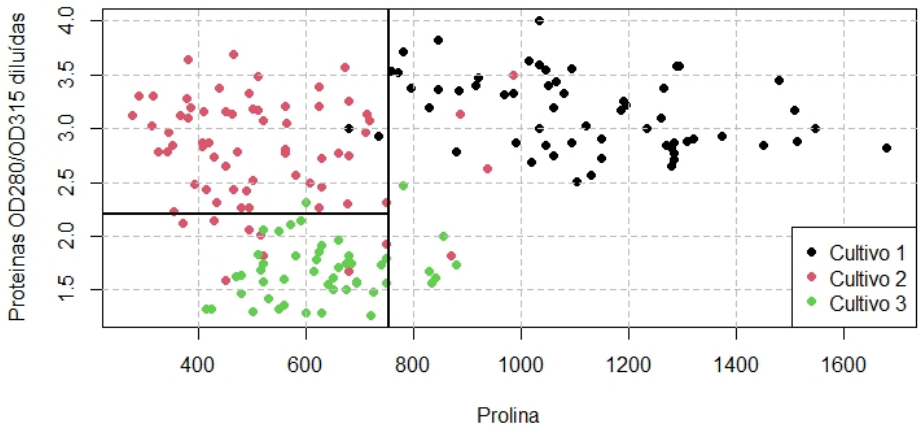


Figura 2.4: Ejemplo de cómo el árbol separa los datos

$\geq$ *755* , el cual consigue aislar gran parte de los elementos de dicho cultivo de manera sencilla. La recta horizontal que corta el espacio de la izquierda, muestra la separación del nodo 3, $OD280.OD315 \geq 2.2$, la cual consigue distinguir entre las clases restantes.

El modelo posteriormente realiza más separaciones sobre el resto de variables, las cuales no son tan fáciles de visualizar. Sin embargo, la Figura 2.4 permite gráficamente mostrar cómo el árbol divide el espacio muestral para clasificar las nuevas observaciones.

Capítulo 3

Construcción de un árbol

Como se puede ver en la Figura 2.1, la estructura en forma de árbol binario simplifica enormemente la construcción de este tipos de modelos. Esto es así, hasta el punto que se puede resumir el proceso en tres pasos que se deben aplicar a todas las ramas a medida que el modelo se expande.

- Decidir si un nodo es terminal.
- Si lo es, asignarle un valor a dicha hoja
- En caso contrario, extender el árbol con dos nuevas ramas saliendo del nodo, dividiendo el subespacio asociado.

El primer punto, consiste en decidir si se divide el nodo una vez más, o si por el contrario se clasifica como nodo terminal y se le asigna un valor final. Para poder responder esta pregunta, es necesario comprobar si el modelo se ajusta correctamente a los datos del conjunto de aprendizaje. El Capítulo 5 se dedicará a estudiar cómo evoluciona esta relación entre el número de nodos terminales de un árbol y su función como modelo, y los distintos métodos empleados para determinar el tamaño adecuado de un árbol de decisión.

Todo lo referente al segundo punto y a encontrar el valor adecuado para un nodo terminal, ha sido estudiado anteriormente para los distintos tipos de modelos posibles. Por último, este Capítulo estará dedicado al tercer punto, correspondiente a encontrar la forma en las que se puede dividir un nodo, y decidir cuál es la más adecuada con el fin de construir un buen modelo.

3.1. Función de Impureza

Hasta ahora, se ha trabajado con una función riesgo R, de tal forma que es posible cuantificar de manera sencilla la calidad de un modelo. Como ya hemos mencionado, esta magnitud toma

distintas definiciones en función del tipo de árbol que se intenta construir, llamándose ratio de clasificación incorrecta en el caso de ser un clasificador, o error de mínimos cuadrados en un predictor, véanse Definiciones 1.2, 1.5.

A pesar de ser dos definiciones distintas, ambas están construidas de tal forma que la siguiente igualdad se cumple. Sea s una separación del nodo t en t_i y t_d, la disminución del riesgo del árbol viene dado por

$$\Delta R(t,s) = \hat{p}(t) \cdot [r(t) - \hat{p}_i \cdot r(t_i) - \hat{p}_d \cdot r(t_d)],$$

donde $\hat{p}(t)$ estima la probabilidad de que un elemento de $\mathcal{L}$ caiga en t. Por otro lado, $\hat{p}_i = \hat{p}(t_i)/\hat{p}(t)$, y $\hat{p}_d = \hat{p}(t_d)/\hat{p}(t)$. En el caso de que las $\hat{\pi}(j)$ no sean homogéneas, este cálculo se puede complicar, pero generalmente serán el porcentaje de los elementos de $\mathcal{L}_t$ que bajan por cada rama. Por construcción, se cumple que $\hat{p}_i + \hat{p}_d = 1$.

Sin embargo, en ciertos casos, la optimización de este ratio puede no ser un buen criterio para construir un árbol. En la Figura 3.1 se presenta un ejemplo tomado de [5], en el que se tienen dos posibles separaciones de un nodo con elementos en las clases $1, 2$.

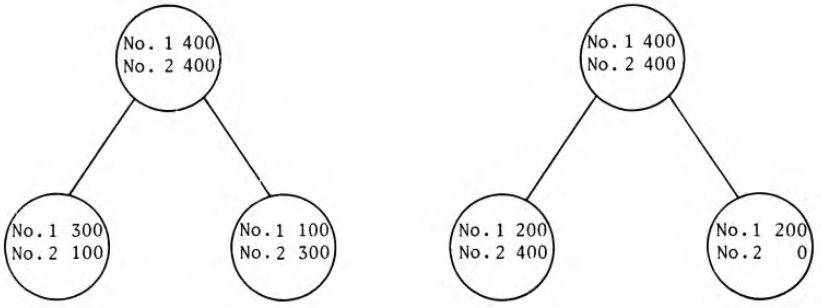


Figura 3.1: Dos separaciones de un nodo de 800 elementos.

Al calcular el ratio de clasificación incorrecta de ambos árboles, se obtiene un valor $R(T) =$ 0,25. Según este criterio, ambas separaciones son igual de buenas. Sin embargo, uno de los nodos del segundo árbol está compuesto únicamente por elementos de la primera clase y no será necesario dividirlo en partes más pequeñas. Teniendo esto en cuenta, la segunda partición es más beneficiosa a la hora de construir un árbol de clasificación.

Este tipo de sucesos se hacen más comunes a medida que se construyen modelos más complejos. Por esta razón, se propone como criterio de separación la reducción de una función $i(t)$. Esta debe ser capaz de medir la "impureza" de un árbol, tomando valores elevados cuanto más homogénea sea la distribución de clases.

Definición 3.1. Se dice que una función ϕ simétrica no negativa es de impureza, si cumple las condiciones:

- Definida sobre vectores J-dimensionales $\vec{p} = (p_1, p_2 \dots p_J)$ tal que $p_j \in [0,1]$ y $\sum_j p_j = 1$.
- El punto homogéneo $(\frac{1}{J}, \frac{1}{J} \dots \frac{1}{J})$ representa el único punto máximo de ϕ.
- $\phi(p_1, p_2 \dots p_J) = 0$ si $p_j = 1$ para algún j.

La primera de las condiciones está impuesta para que la función de impureza este definida sobre todo vector de probabilidades $\vec{p}(t) = (\hat{p}(1/t), \hat{p}(2/t) \dots \hat{p}(J/t))$, siendo $\hat{p}(j/t)$ una estimación de la probabilidad de que un elemento que llegue al nodo t, pertenezca a la clase j

Definición 3.2. Sea ϕ una función de impureza, se define la impureza de un nodo t al valor obtenido de aplicar ϕ al vector de probabilidades $\vec{p}(t)$

$$i(t) = \phi(\hat{p}(1/t), \hat{p}(2/t) \dots \hat{p}(J/t)).$$

Se observa que si todos los elementos de $\mathcal{L}_t$ pertenecen a una única clase j, la impureza será $i(t) = 0$. Esto se debe a que bajo estas condiciones $\hat{p}(j/t) = 1$.

Al igual que con la función de riesgo, se puede definir la impureza de un árbol $I(T)$ como la suma de las impureza de cada uno de sus nodos terminales.

$$I(T) = \sum_{t \in \tilde{T}} I(t) = \sum_{t \in \tilde{T}} \hat{p}(t) \cdot i(t)$$

De la misma forma, sea s una separación del nodo t, se tiene que la disminución de la impureza debido a s viene dada por:

$$\Delta I(s,t) = \hat{p}(t) \cdot \Delta i(s,t) = \hat{p}(t) \cdot [i(t) - \hat{p}_i \cdot i(t_i) - \hat{p}_d \cdot i(t_d)].$$

Más adelante, en 4.1 se demostrará que $\Delta R(s,t) \geq 0$ para todo t y s. Esta es una propiedad importante, ya que asegura la existencia de una separación que mejore la calidad del árbol. Por esta razón, es interesante comprobar si también se cumple $\Delta I(s,t) \geq 0$ para toda pareja (s,t).

Proposición 3.3. *Sea $i(t) = \phi(\hat{p}(1/t), \hat{p}(2/t) \dots \hat{p}(J/t))$ la impureza de t, y s una separación del nodo entre los hijos t_i y t_d. Si ϕ es convexa sobre su dominio, se tendrá:*

$$\Delta i(s,t) = i(t) - \hat{p}_i \cdot i(t_i) - \hat{p}_d \cdot i(t_d) \geq 0 \tag{3.1}$$

Alcanzando la igualdad en el caso de que $\hat{p}(j/t) = \hat{p}(j/t_i) = \hat{p}(j/t_d)$ para todas las clases j.

Demostración. Por la definición usual de función convexa, se cumple que

$$\begin{aligned} p_i \cdot i(t_i) + \hat{p}_d \cdot i(t_d) = \hat{p}_i \cdot \phi(\hat{p}(1/t_i), \dots \hat{p}(J/t_i)) + \hat{p}_d \cdot \phi(\hat{p}(1/t_d), \dots \hat{p}(J/t_d)) \leq \\ \leq \phi(\hat{p}_i \cdot \hat{p}(1/t_i) + \hat{p}_d \cdot \hat{p}(1/t_d), \dots \hat{p}_i \cdot \hat{p}(J/t_i) + \hat{p}_d \cdot \hat{p}(J/t_d)). \end{aligned}$$

En este caso, como $\hat{p}_i + \hat{p}_d = 1$, se alcanzará la igualdad si la estimación del vector de probabilidad es igual para los tres nodos, $\vec{p}(t) = \vec{p}(t_i) = \vec{p}(t_d)$.

Se recuerda que $\hat{p}(j, t_i) = \hat{p}(t_i) \cdot \hat{p}(j/t_i)$, es la estimación de la probabilidad combinada de que un elemento llegue a t_i y pertenezca a la clase j. Esto unido a que $\hat{p}_i = \hat{p}(t_i)/\hat{p}(t)$ por construcción, se llega a que:

$$\hat{p}_i \cdot \hat{p}(j/t_i) + \hat{p}_d \cdot \hat{p}(j/t_d) = \frac{\hat{p}(t_i) \cdot \hat{p}(j/t_i) + \hat{p}(t_d) \cdot \hat{p}(j/t_d)}{\hat{p}(t)} = \frac{\hat{p}(j, t_i) + \hat{p}(j, t_d)}{\hat{p}(t)} = \hat{p}(j/t)$$

Uniendo ambos resultados, se llega finalmente a la desigualdad

$$\hat{p}_i \cdot i(t_i) + \hat{p}_d \cdot i(t_d) \leq \phi(\hat{p}(1/t), \dots \hat{p}(J/t)) = i(t),$$

con lo que se concluye la demostración. □

3.2. Criterio de Gini

Una vez definido el criterio para separar los nodos de un árbol a partir de la impureza, surge el problema de encontrar una función ϕ capaz de medirla. Por la Proposición 3.3, es conveniente que sea convexa. Tampoco puede beneficiar demasiado a los nodos "puros", pues se podrían acabar perdiendo los beneficios que tiene estructurar el modelo en forma de árbol.

Al final, es trabajo del analista definir dicha función de tal forma que se ajuste bien al problema. Como primera aproximación, la función de impureza más simple viene dada por:

$$\phi(\vec{p}) = 1 - \max_{j=1\dots J} p_j = \min_{j=1\dots J} p_j.$$

Esta aproximación al problema no es muy popular ya que no beneficia lo suficiente a los nodos "puros". En los orígenes del método, probando cómo distintas funciones se comportaban ante los experimento, se hizo especialmente popular usar como medida de la impureza:

$$\phi(\vec{p}) = -\sum_{j=1}^{J} p_j \cdot \ln p_j.$$

Similar a este, se empezó a utilizar el método de Gini. Este criterio se deduce de una aproximación de la probabilidad de clasificar incorrectamente un elemento que llegue al nodo t. Se estima que $\hat{p}(j/t)$ es la probabilidad de que un elemento cualquiera de t sea clasificado por j. Estimando por $\hat{p}(i/t)$ la probabilidad de que en realidad pertenezca a la clase i y sumando sobre todas las combinaciones $i \neq j$, se deduce el índice de diversidad de Gini:

$$\phi(\vec{p}) = \sum_{i \neq j} p_i \cdot p_j = 1 - \sum_{j=1}^{J} p_j^2 \tag{3.2}$$

Se tiene que el índice de diversidad de Gini está compuesto por un polinomio cuadrático no negativo. Por esta razón cumple la condición de convexidad, y por la Proposición 3.3, $\Delta i(s,t) \geq 0$ para toda separación s.

Se puede observar que en la construcción del criterio de Gini, se estima la probabilidad de clasificar incorrectamente un elemento que llegue a t. Sin embargo, todos los errores se consideran iguales, $C(i/j) = \delta_{ij}$. Con el fin de tener en cuenta esta construcción, se puede ampliar la definición del índice de diversidad de Gini a :

$$i(t) = \sum_{i \neq j} C(i/j)\hat{p}(i/t)\hat{p}(j/t).$$

La aplicación de esta fórmula puede llegar a resultados satisfactorios en la construcción de árboles. Sin embargo, bajo esta definición general es imposible asegurar la convexidad del problema. Determinar si la aplicación de este método es beneficioso o no, recae una vez más en el analista.

En el caso de los estimadores de probabilidad de clasificación, se observa que el índice de diversidad de Gini 3.2, es similar a la estimación del error en cada nodo 2.1. Por esta razón, el uso del criterio de Gini es ideal para la construcción de este tipo de árboles.

3.3. Criterio de Twoing

Se plantea una segunda aproximación al problema de dividir el espacio con la ayuda de un conjunto de entrenamiento. En este caso, la clasificación se reduce a únicamente dos categorías, $C_1 = \{j_1, j_2 \ldots, j_n \leq J\}$, subconjunto de $\mathcal{C} = \{1, 2, 3, \ldots, J\}$, y su complementario $C_2 = \mathcal{C} - C_1$. De esta forma, dada una función de impureza $\phi(p_1, p_2)$, y una separación s del nodo t, se define de manera usual $\Delta i(s, t, C_1)$, disminución de la impureza restringido al conjunto C_1.

El criterio de Twoing propone para cada uno de los C_1 subconjuntos de $\mathcal{C}$, buscar aquella separación $s^*(C_1)$ que maximice el incremento de impureza. Una vez completado, escoger aquella de las superclases con la que se obtenga un mayor $\Delta i(s^*(C_1), t, C_1)$.

El mayor inconveniente que surge de aplicar este criterio es el coste computacional, que crece exponencialmente con la complejidad del problema. Se tiene que para un conjunto de J clases, existen 2^{J-1} posibles superclases, y que en cada paso de la construcción del árbol, hay que encontrar la división óptima $s^*(C_1)$ para todos los C_1.

Este problema se simplifica enormemente si como función de impureza se aplica Gini en bidimensional $\phi(p_1, p_2) = p_1 \cdot p_2$. Se puede demostrar que en este caso, el aumento de la impureza

asociado viene dado por

$$\Delta i\,(s,t) = \frac{\hat{p}_i \hat{p}_d}{4} \left[\sum_{j=1}^{J} \mid p\,(j/t_i) - p\,(j/t_d) \mid \right]^2 . \tag{3.3}$$

Siendo $C_1^*(s) = \{j/p(j/t_i) \geq p(j/t_d)\}$ la superclase que caracteriza esta separación. Poder calcular la impureza antes de separar las clases, hace posible aplicar este criterio con la misma eficiencia que Gini.

El principal inconveniente de aplicar este criterio para la construcción de un modelo, es que al juntar varias clases en una categoría conjunta, no tiene sentido medir la impureza total de un árbol. Por esta razón, la única forma de saber si el método funciona correctamente, es una vez completado el árbol.

A nivel computacional, si se tiene en cuenta la ecuación 3.3, no hay diferencia entre un criterio u otro. En pruebas realizadas mediante la construcción de árboles por ambos métodos y con la misma base de datos, se ha podido comprobar que en los primeros pasos se comportan prácticamente igual, llegando en muchos casos al mismo resultado final.

Aunque es posible construir ejemplos en los que un método es mejor que otro, se suele usar el criterio de Gini, ya que agrupa nodos terminales puros con mayor frecuencia. A diferencia del criterio de Twoing, que tiende más a mantener todos los nodos con el mismo número de elementos, lo que puede llevar a que en algunos casos la eficacia del método sea un poco peor.

3.4. Criterio en un árbol de predicción

Sea T un árbol de predicción en el que el valor asociado a un nodo $\nu(t)$ viene dado por la media muestral $\bar{\mu} = \sum_{(\vec{x},Y)\in\mathcal{L}_t} Y/N^t$, se tiene que el riesgo de uno de sus nodos se estima por $r(t) = \sum_{(\vec{x},Y)\in\mathcal{L}_t}(Y - \bar{\mu}(t))^2/N^t$. De esta forma, al realizar una separación por s, se observará una disminución del ratio:

$$\Delta r(t,s) = r(t) - \hat{p}_i \cdot r(t_i) - \hat{p}_d \cdot r(t_d), \tag{3.4}$$

donde $\hat{p}_i$ y $\hat{p}_d$ representan la proporción del conjunto de entrenamiento restringido $\mathcal{L}_t$ que caen en cada parte de la división.

Calcular esta magnitud para cada una de las parejas (t,s) es un proceso computacionalmente complejo, ya que requiere de elevar una gran cantidad de números al cuadrado. Por esta razón, es importante encontrar un criterio que permita optimizar esta Δr eficientemente.

Se considera por tanto que $\hat{p}_i = N^{t_i}/N^t$ es un estimador de $P_i = P[\vec{x} \in \Omega_{t_i}/\vec{x} \in \Omega_t]$, probabilidad del espacio muestral asociado al nodo t_i restringido a Ω_t. Recordando que $r(t)$ es

un estimador de la varianza del nodo $\sigma(t) = E[(Y - \mu(t))^2/\vec{x} \in \Omega_t]$, se llega a la conclusión que Δr estima δr.

$$\delta r(t,s) = \sigma(t) - P_i \cdot \sigma(t_i) - P_d \cdot \sigma(t_d). \tag{3.5}$$

Por construcción, $\mu(t) = P_i \cdot \mu(t_i) + P_d \cdot \mu(t_d)$. De la misma forma se cumple la igualdad $E[(Y - \mu(t))^2/\vec{x} \in \Omega_t] = P_i \cdot E[(Y - \mu(t_i))^2/\vec{x} \in \Omega_{t_i}] + P_d \cdot E[(Y - \mu(t_d))^2/\vec{x} \in \Omega_{t_d}]$

$$\begin{aligned} & E\left[(Y-\mu(t))^2 - (Y-\mu(t_u))^2/\vec{x} \in \Omega_t\right] = \\ = & E\left[\mu(t)^2 - \mu\left(t_i\right)^2 - 2Y\left(\mu\left(t_i\right) - \mu(t)\right)/\vec{x} \in \Omega_t\right] = \\ = & \ \mu(t)^2 - \mu(t_i)^2 - 2\mu\left(t_i\right)\left(\mu\left(t_i\right) - \mu(t)\right) = \\ = & \left(\mu(t) - \mu\left(t_i\right)\right)^2 \end{aligned} \tag{3.6}$$

Juntando todo lo anterior, se puede llegar a la conclusión:

$$\begin{aligned} \delta r(t,s) &= P_i\left(\mu\left(t_i\right) - \mu(t)\right)^2 + P_d\left(\mu\left(t_d\right) - \mu(t)\right)^2 \\ &= P_i\mu^2\left(t_i\right) + P_d\mu^2\left(t_d\right) - \mu^2\left(t^2\right) \\ &= P_iP_d\left(\mu\left(t_i\right) - \mu\left(t_d\right)\right)^2. \end{aligned} \tag{3.7}$$

Regresando al mundo de los estimadores, se obtiene el resultado final $\Delta r(t,s) = p_i \cdot p_d(\nu(t_i) - \nu(t_d))^2$. De donde se concluye que para estimar la disminución del error cuadrático medio al realizar una separación s, es suficiente con estudiar las medias muéstrales en sus nodos hijos. Este resultado se suele utilizar como criterio para construir árboles de predicción.

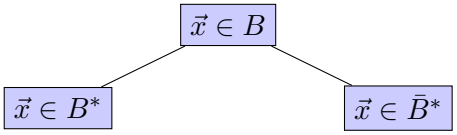
3.5. Conjuntos de preguntas

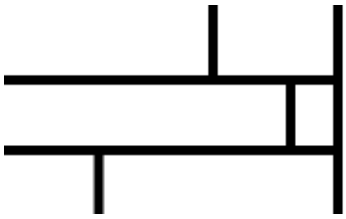
A lo largo de este capítulo se ha estado visitando los distintos tipos de árboles de clasificación y qué problemas pueden surgir a la hora de construir un modelo de este tipo. A la conclusión a la que se llega en cada apartado, es que para cada nodo t y separación s, se puede construir una función $\Delta i(t,s)$ que cuantifica el incremento de la pureza del árbol al realizar dicha división.

Bajo los criterios establecidos, se plantea el problema de optimización de encontrar la separación que maximice el incremento de pureza. Por lo general, comprobar todas las separaciones posibles resulta computacionalmente prohibitivo, por lo que se suele acotar a conjuntos concretos de preguntas Q, y se busca el $\max_{s \in Q} \Delta i(t,s)$

Con el fin de construir conjuntos de preguntas, en [6] se presentan las siguientes como las particiones más comúnmente usadas. El vector de medidas se describe con la notación $\vec{x} = (x_1, \ldots x_M)$

En el caso de que x_m sea una variable categórica con B el conjunto de categorías, se tendrá que las preguntas posibles son de la forma $s = ¿x_m \in B^*?$, donde $B^* \subset B$





Para el caso de variables continuas, la partición más sencilla es mediante la aplicación de hiperrectángulos. Para un x_m y un valor real c se define la pregunta $s = ¿x_m < c?$

Se puede complicar un poco más los cortes lineales mediante la aplicación de hiperplanos, se toma un vector de pesos $\vec{a} = (a_1, \dots a_n)$, un valor real c y se define la pregunta $s = ¿\vec{a}^t \cdot \vec{x} < c?$

Por último, también se suelen construir lo que se denominará como árboles esféricos, en los que las divisiones del espacio vectorial se determinan entre dentro y fuera de una bola. Se toma un centro $\vec{z}$ y un radio c y se define $s = ¿ \parallel \vec{x} - \vec{z} \parallel < c?$

Todo esto se puede complicar mediante la aplicación de recursos más complejos o mediante la combinación de preguntas. Por ejemplo si x_1 es una variable categórica y x_2 una continua, $s = ¿x_1 \in B^* \ \& \ x_2 < c?$ sería una pregunta válida.

3.6. Conjunto estándar de preguntas

A la hora de construir un árbol, la inclusión de demasiados tipos de preguntas puede llevar a la construcción de un conjunto Q demasiado grande, lo que resultaría en un coste computacional demasiado elevado para una mejora de $\Delta i(t, s)$ mínima. Por esta razón, generalmente se emplea el conjunto de preguntas estándar, el cual ofrece un gran número de preguntas, una manera sencilla de obtenerlas, y en la práctica, un tiempo de fabricación del árbol aceptable.

Sea $\mathcal{L} = (\vec{x}_i, Y_i)_{i=1\dots N}$ el conjunto de entrenamiento en el que $\vec{x}_i = (x_1^i, x_2^i, \dots, x_M^i)$ representa el vector de medidas de un modelo, el conjunto de preguntas estándar estará compuesto por la unión $Q = \bigcup_{m=1}^{M} Q_m$. Las preguntas de cada conjunto Q_m únicamente hacen referencia a la variable x_m, y el primer paso en la separación de un nodo, será la construcción de cada uno de

estos subconjuntos.

En el caso de que x_m sea una variable continua, Q_m estará compuesto por preguntas de la forma ¿$x_m < c$?. Para facilitar la explicación, se considera que el conjunto de mediciones de x_m está ordenado, $\{x_m^1 \leq x_m^2 \leq \cdots \leq x_m^N\}$.

Se define así $c_m^i = (x_m^{i+1} - x_m^i)/2$ sucesión de puntos medios, y se construye el conjunto de preguntas $Q_m = \{¿x_m < c_m^i?/i = 1 \ldots N-1\}$. Hay que tener en cuenta que todos los cortes c en el intervalo (x_m^i, x_m^{i+1}) representan la misma pregunta ¿$x_m < c$?, ya que llevan a cabo la misma separación del conjunto de entrenamiento. Se decide tomar el punto medio ya tiende a separar el espacio muestral en partes equiprobables.

En el caso de que x_m represente una variable categórica sobre el conjunto $B = \{b_1, \ldots b_n\}$, Q_m estará compuesto por todos los posibles divisiones de B, $Q_m = \{¿x_m \in B^*?/B^* \subset B\}$. Esto representa $2^{|B|-1} - 2$ preguntas posibles, ya que $\emptyset$ y B no ofrecen información al modelo y por simetría B^* lleva a cabo la misma separación que su complementario $\bar{B}^*$.

Capítulo 4

Podado y tamaño adecuado

La principal cuestión en la construcción de este tipo de modelos, es asegurarse que el árbol final sea una buena representación del medio real al que se va a enfrentar. Por desgracia, a la hora de construirlo, solo se dispone de una cantidad de información limitada por el conjunto de aprendizaje, lo que dificulta juzgar si el modelo realiza predicciones buenas o no.

Con este fin, se suele dividir el conjunto de aprendizaje en dos partes, uno de entrenamiento $\mathcal{L}_e$, usado para la construcción, y uno de validación, $\mathcal{L}_v$, al margen del anterior, dedicado a juzgar si el modelo realiza predicciones correctas. Se ha definido el ratio $R(T)$ como un estimador del error cometido por un modelo, para este apartado se distinguirá entre R^e y R^v, en función si se han utilizados los datos del conjunto de entrenamiento o de validación.

Es importante que el árbol final sea del tamaño adecuado, demasiado pequeño, y no extraerán todas las relaciones contenidas en el conjunto de aprendizaje, demasiado grande, y se llegarán a errores de sobredimensión, en los que se pierde parte de la información. En 4.1 se grafican datos de un estudio de [5]. A partir de una serie de experimentos consistentes en construir clasificadores para diversas bases de datos, se presenta la evolución de los errores medios en función del número de nodos terminales del árbol.

Se puede observar el efecto de sobredispersión en R^v, a partir de los 10 nodos terminales, el error aumenta de nuevo. Por otro lado, R^e siempre decrece, da igual el número de nodos o qué división se realice. Este resultado que se demuestra en 4.1.

Proposición 4.1. *Sea s una separación del nodo t entre t_i y t_d, se tiene que $\Delta r^e(t,s) > 0$*

Demostración. En el caso de que el modelo creado sea de predicción, en 3.7 se deduce que $\Delta r^e(t,s) = p_i \cdot p_d(\nu^e(t_i) - \nu^e(t_d))^2$, claramente superior a 0.

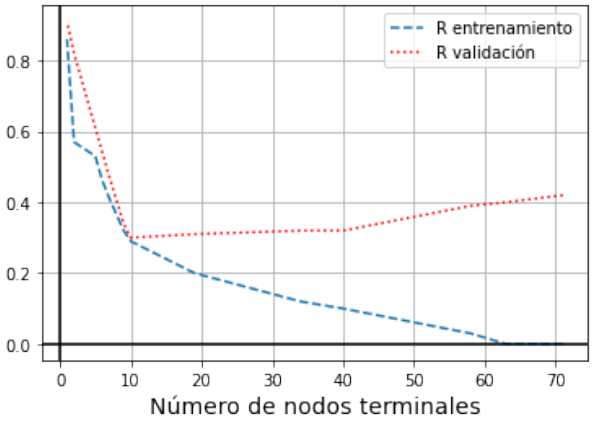


Figura 4.1: Comparación R^e con R^v en un mismo árbol

En el caso de un modelo de clasificación, se trabaja con la definición de error de clasificación incorrecta de un árbol 2.1

$$\begin{gathered} R^e(t) = \sum_j C\left(j^*(t)/j\right) \hat{p}(j,t) = \\ = \sum_j C\left(j^*(t)/j\right) \left[\hat{p}\left(j,t_i\right) + \hat{p}\left(j,t_d\right)\right] \geqslant \\ \geqslant \text{mín}_i \sum_j C(i/j)\hat{p}(j,t_i) + \text{mín}_i \sum_j C(i/j)\hat{p}(j,t_d) = \\ = \sum_j C\left(j^*(t_i)/j\right) \hat{p}(j,t_i) + \sum_j C\left(j^*(t_d)/j\right) \hat{p}(j,t_d) \\ = R^e\left(t_i\right) + R^e\left(t_d\right) \end{gathered} \tag{4.1}$$

De donde se deduce que $\Delta R^e(s,t) = R^e(t) - R^e(t_i) - R^e(t_d) \geqslant 0$ □

4.1. Reglas de frenado

El primer intento para resolver este problema, fue buscar una buena regla de frenado, un criterio capaz de determinar en que momento hay que dejar de aumentar el árbol. La aproximación que tuvo más éxito fue imponer una cota superior a la impureza. No realizar ninguna separación que produjese una disminución de esta menor a cierto β.

Se tiene que a la hora de separar un nodo en dos, no es posible saber cómo evolucionará el árbol en pasos futuros. Lo que proboca que sea imposible generar una regla de frenado capaz de ofrecer resultados satisfactorios. No es difícil construir ejemplos en los que se cumpla $\Delta i(t,s) < \beta$, pero que al seguir expandiendo el árbol se consigan grandes reducciones de la impureza.

4.2. Podado

Se propone una aproximación al problema distinta, primero se hace crecer el árbol hasta T_{max}. Este debe ser lo suficientemente grande como para asegurar que la realización de más separaciones no provoquen una gran reducción del error. Un buen criterio que se suele tomar es no tener ningún nodo por debajo de los 5 elementos del conjunto de entrenamiento.

El algoritmo suele consistir en la construcción de una sucesión de árboles $T_{max}, T_1, T_2, \dots \{t_1\}$, donde $\{t_1\}$ es únicamente la raíz, y cada T_H tiene H nodos terminales y se construye mediante podados sobre T_{max}. Idealmente, se cumplirá que $R^e(T_H) = \text{máx}_{|\hat{T}|=H} R^e(T)$, con $\mid \hat{T} \mid$ número de nodos terminales. A partir de esta sucesión, se toma el modelo que minimice el error de validación.

A nivel práctico, este planteamiento no es factible, no hay ninguna relación real entre T_H y T_{H+1}, lo que imposibilita la aplicación práctica de este algoritmo. Sería necesario el estudio de todos los subárboles de T_{max} construidos mediante podados. Una aproximación real a este método se encuentra en los trabajos de Breiman L. y Stone C.J. [5].

4.3. Podado coste-complejidad

Se propone castigar el tamaño de un árbol aplicándole un precio al número de nodos terminales. De esta forma se define la medida de coste-complejidad $R_\alpha(T) = R^e(T) + \alpha \cdot \mid \hat{T} \mid$. Sea t un nodo terminal, se define $R_\alpha(t) = R(t) + \alpha$, cumpliendo así la relación $R_\alpha(T) = \sum_{t \in \hat{T}} R_\alpha(t)$. Denotando por T_D y T_I las dos ramas principales de un árbol T, se tendrá que $R_\alpha(T) = R_\alpha(T_I) + R_\alpha(T_D)$, ya que todo nodo terminal pertenece necesariamente a uno de los dos subárboles.

Definición 4.2. Un subárbol $T_1 \leq T_{max}$ se dice que es de podado óptimo para cierto α, si $R_\alpha(T_1) = \text{mín}_{T' \leq T_{max}} R_\alpha(T')$. En el caso de que exista un subárbol de podado óptimo menor que el resto, se denotará por $T(\alpha)$

Teorema 4.3. *Para todo árbol T y número real α, existe un subárbol mínimo de podado óptimo $T(\alpha)$*

Demostración. Esta demostración se realiza por inducción sobre el número de nodos terminales. La idea es estableces una relación entre el subárbol de podado óptimo de un árbol, y el de sus ramas principales T_D, T_I.

Se parte de la hipótesis de que todo árbol menor que T, tiene un podado óptimo para α. Con esto se asegura la existencia de $T_I(\alpha)$ y $T_D(\alpha)$. Como inicio de la inducción, en el caso de

los árboles compuestos solo por la raíz $\{t_1\}$, el podado óptimo es si mismo, ya que es el único subárbol posible.

El primer paso de la demostración, es que solo hay dos aspirantes a subárbol de podado óptimo de T, estos son la raíz, $T(\alpha) = \{t_1\}$, y la conexión de la raíz con los podados óptimos de las ramas principales, $T(\alpha) = \{t_1\} \bigcup T_I(\alpha) \bigcup T_D(\alpha)$.

Supongamos que existe un árbol de podado óptimo T' que no entra en una de estas dos categorías. Como no es la raíz, se puede descomponer en $T' = \{t_1\} \bigcup T'_I \bigcup T'_D$. Bajo estas condiciones, se debe cumplir que

$$R_\alpha(T') = R_\alpha(T'_I) + R_\alpha(T'_D) < R_\alpha(T_I(\alpha)) + R_\alpha(T_D(\alpha)) = R_\alpha(T(\alpha)).$$

Esto contradice la hipótesis de que las dos ramas principales tienen subárboles de podado óptimos, ya que se habrá encontrado un árbol que cumple $R_\alpha(T'_I) <_\alpha (T_I(\alpha))$ ó $R_\alpha(T'_D) <_\alpha (T_D(\alpha))$.

Una vez que se han reducido los candidatos a $T(\alpha) = \{t_1\}$ y a $T(\alpha) = \{t_1\} \bigcup T_I(\alpha) \bigcup T_D(\alpha)$, el árbol de podado óptimo de T será aquel con un menor ratio coste-complejidad R_α □

El el caso de que α sea próximo a 0, el coste no afectará a la estimación de R_α, por lo que $T(\alpha) = T_{max}$. Por el contrario, si se toma un valor de α demasiado alto, solo se tendrá el nodo raíz $T(\alpha) = \{t_1\}$. Por tanto el problema se reduce a encontrar el α que ofrezca un árbol de podado óptimo del tamaño adecuado.

Se plantea ahora un algoritmo por el cual, a partir de podados sucesivos se construye una sucesión de árboles T_{max}, $T_{max} = T_1, T_2, \ldots T_K = \{t_1\}$. Se pude demostrar que bajo este proceso, también se obtienen los números reales $\alpha_{min} = \alpha_1 < \alpha_2 < \ldots \alpha_K = \infty$, de tal forma que en el intervalo $[\alpha_k, \alpha_{k+1})$, se cumple que $T_{max}(\alpha) = T_k$

Algoritmo Coste Complejidad

Entrada: $\alpha_1 = 0$, árbol máximo T_1 y parámetro inicial $k = 1$
While $T_k \neq \{t_1\}$:
  For $t \in T_k - \hat{T}_k$:
    Calcular valor crítico $g_k(t) = \frac{R(t) - R(T_t)}{|\hat{T}| - 1}$
    Guardar $\hat{t}$, nodo con menor g_k
  $\alpha_{k+1} = \min g_k(\hat{t})$
  $T_{k+1} = T_k - T_{\hat{t}}$
  $k = k + 1$

Partiendo del árbol máximo $T_{max} = T_1$, se busca la rama más débil, aquella que al aumentar el valor de α, supone un mayor coste al ratio R_α. Se define así el valor crítico de un nodo no terminal $g_1(t) = \frac{R(t)-R(T_t)}{|\hat{T}|-1}$. Caracterizado con que $R_\alpha(T_t) < R_\alpha(\{t\})$ sí y solo si $\alpha < g_1(t)$.

Esto quiere decir, que el valor crítico de un nodo no terminal determina el rango de costes en los que es preferible no podar. Por lo que la rama más débil $\hat{t}$ será aquella con el menor g_1. Se define por tanto $\alpha_2 = \text{mín}_{t \in T-\hat{T}}\, g_1(t)$. Al aplicar costes α superiores a este, el subárbol óptimo pasará a ser $T(\alpha) = T_1 - T_{\hat{t}} = T_2$.

Este proceso se puede continuar calculando ahora los valores críticos g_2 de todos los nodos no terminales de T_2, obteniendo de manera sucesiva los árboles $T_3, T_4, \ldots$. El algoritmo termina en el momento que el árbol de podado óptimo es la raíz $T_k = \{t_1\}$

4.4. Estimación del mejor árbol podado

Del algoritmo presentado en el apartado anterior se obtiene una serie de árboles, cada uno podado del anterior $T_1 > T_2 > \ldots T_K = \{t_1\}$. Y cada uno de ellos cumpliendo que $T_k = t(\alpha_k)$, árbol mínimo de podado óptimo para una sucesión de $_k$. La cuestión ahora, es encontrar una forma "honesta"de estimar el riesgo, de tal forma que el árbol final se obtiene de minimizar dicho error $\text{mín}_k\, \hat{R}(T_k)$.

La forma más simple de obtener dicho error, es usando un subconjunto de $\mathcal{L}$ distinto al usado para crear el árbol. Este podría ser el conjunto de validación, o preferiblemente, tomar un porcentaje de los datos del conjunto de entrenamiento y reservarlos para este paso.

Hay modelos que plantean realizar un modelo de la distribución Ω a partir del conjunto de aprendizaje, con el fin de obtener la mejor estimación posible del riesgo. Estos tienden a conseguir resultados muy buenos, pero computacionalmente son extremadamente caros, en mucho casos no merece la pena construirlos.

4.5. Ejemplo podado coste complejidad

Con el fin de mostrar un ejemplo de cómo emplear el podado por coste complejidad, se continua con el modelo construido en el Capítulo 3 , 2.3. Se recuerda que a partir de una base de datos extraída de [7], se procuraba obtener la procedencia de un vino a partir del estudio de sus componentes químicos. De la misma forma, el conjunto de aprendizaje se dividía en uno de entrenamiento y uno de validación.

En este ejemplo, y con el fin de simplificar la explicación, los testeos se usarán con el fin de

construir el modelo, practica inaceptable en un entorno real. Existen métodos más complejos que permiten escoger el modelo adecuado sin necesidad de usar el conjunto de validación.

El primer modelo que se presentó, se obtuvo a partir de cortar el crecimiento del árbol si no producían un aumento de la impureza de $\Delta I(T) \geq 0,005$. De esta forma se llegó al modelo 4.4, el cual tiene $R^e = 0,028$ y $R^v = 0,147$.

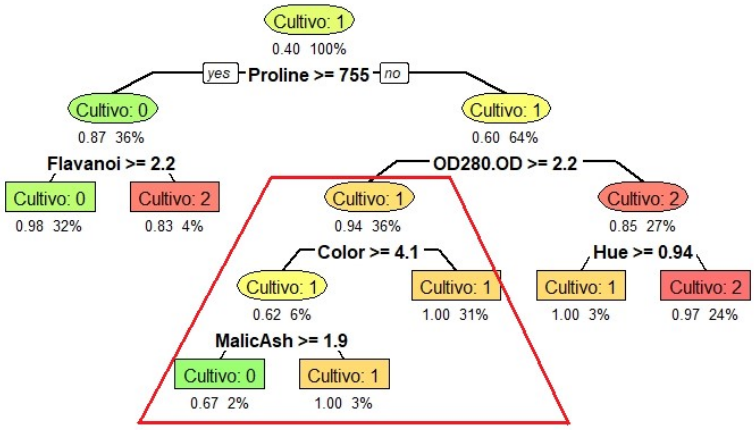


Figura 4.2: Árbol de clasificación máximo T_{max}

El paquete *rpart* [13] que se está utilizando para la realización de estos ejemplos, calcula el valor coste complejidad de todas las ramas a medida que las construye, lo que facilita enormemente el proceso de podado. En los árbol de este apartado, se presentará en rojo la rama más débil, aquella que se debe podar con mayor rapidez. En el caso del árbol máximo, se tiene que el valor crítico es $\alpha_2 = 0,0116$.

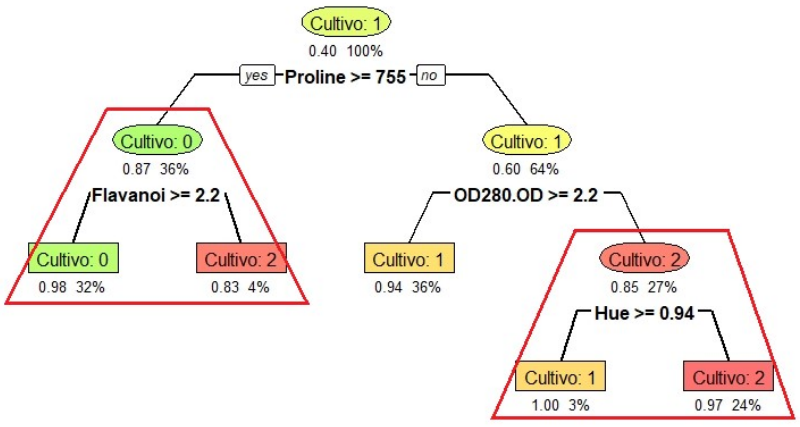


Figura 4.3: Árbol de clasificación T_2

Al podar la rama más débil se obtiene el árbol 4.3. Este mejora en el ratio de validación $R^v = 0,113$, lo que lo hace un modelo más fiable que el máximo. En este caso se tienen dos ramas igual de débiles, por lo que se deben de podar las dos a la vez en el siguiente paso. El valor crítico del segundo árbol es de $\alpha_3 = 0{,}0536$

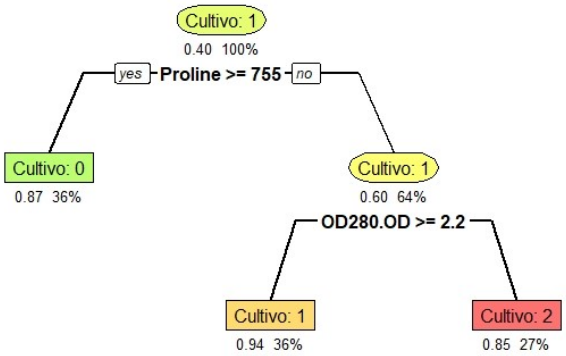


Figura 4.4: Árbol de clasificación T_3

Este último árbol aumenta de nuevo el ratio de validación, $R^v = 0{,}143$. La sucesiva poda de ramas no produce ninguna mejora futura en este modelo, por lo que se decide parar aquí con el proceso.

Capítulo 5

Métodos alternativos

5.1. Boostrap y Bagging

El método Boostrap, introducido por Efron B. en [8], permite la reducción de la varianza en el cálculo de un estimador a partir de una base de datos $\mathcal{L} = \{x_1, \dots x_N\}$. Por ejemplo, se podría tener $s(x)$, un estimador de cierto parámetro θ, como la media muestral $\bar{x} = \frac{1}{N}\sum x_i$ es un estimador de la esperanza de la variable respuesta.

El algoritmo consiste en la construcción de un conjunto de Boostrap $\mathcal{L}^{(B)} = \{\mathcal{L}^1, \dots \mathcal{L}^M\}$ en el que cada uno de sus elementos es un subconjunto diferente de $\mathcal{L}$. Generalmente, cada una de las muestras de Boostrap se construye mediante la toma con reemplazo de puntos de $\mathcal{L}$. De esta forma, todos son del mismo tamaño, pero tienen algunos elementos repetidos. La fuerza del método se basa en estas pequeñas diferencias, ya que cada árbol desarrollará distintas sutilezas, que suman en el modelo final.

A partir de aquí, para cada uno de los $\mathcal{L}^m$ se aplica s para conseguir M estimadores distintos de cierto θ. Se define por último el estimador de Boostrap $\hat{\theta}^{(B)}$, la media aritmética del calculado a partir de cada uno de las muestras del conjunto de Boostrap. Se puede demostrar que este método permite la construcción de estimadores con una menor varianza real. Para más detalles en este tema remitir al libro de Efron and Tibshirani(1994) [9], en concreto el capítulo 6 en el que demuestran las ventajas de este método.

5.1.1. Aplicación a árboles

Después de su creación, y debido al éxito que tenía el método Boostrap para reducir la varianza en estimadores muestrales, se intentó aplicar la misma técnica a distintos problemas

con el fin de obtener resultados, en Breiman(1996) [2] se plante su aplicación a los modelos en forma de árbol. A esta técnica se le acabó llamando Bagging (Boostrap Agregating).

El primer paso es el mismo que en el método general, a partir del conjunto de entrenamiento, se construye un conjunto de Boostrap $\mathcal{L}^{(B)}$. Este estará formado por distintos conjuntos elaborados a partir de los elementos de $\mathcal{L}$.

Una vez que se tiene un árbol construido a partir de cada una de las muestras del conjunto de Boostrap, se unen en el modelo de Bagging $d^B(\vec{x})$. En el caso de que el modelo a desarrollar sea un predictor, se aplica la media muestral $d^B(\vec{x}) = \sum d^m(\vec{x})/M$, y para un clasificador, se tiende a usar un método por votación. A partir del vector de medidas $\vec{x}$, se estima la probabilidad de que pertenezca a la clase j, $\hat{p}_j(\vec{x})$, como el porcentaje de los árboles que predicen $d^m(\vec{x}) = j$. De esta forma, el clasificador de Bagging, toma como $d^B(\vec{x})$ aquella clase con una mayor $\hat{p}_j(\vec{x})$. Estos mismos resultados se pueden emplear para construir modelos de estimación de la probabilidad a partir de los $\hat{p}(\vec{x})$

La segunda parte en la construcción de un árbol, consiste en el podado. Al final de este proceso, a la hora de tomar un modelo final, apartado 4.4, se precisa de un conjunto de testeo, un muestreo con una distribución similar al conjunto de entrenamiento, pero independiente de este. Esto es sencillo de conseguir en Bagging, cualquier subconjunto de $\mathcal{L}$ obtenido bajo los mismos criterios que las muestras de Boostrap, cumplen condiciones similares a las deseadas. De esta forma es posible conseguir los test necesarios sin usar el conjunto de validación.

5.1.2. Por qué Bagging funciona

A pesar de que en un principio los métodos Boostrap no fueron creados para la construcción de estimadores, se puede comprobar de manera sencilla que los modelos diseñados por este método, tienden a ser mejores que los árboles construidos directamente con todo el conjunto de aprendizaje.

Se define el predictor agregado $d_A(\vec{x}) = E_{\mathcal{L}}[d(\vec{x}, L)]$, siendo $E_{\mathcal{L}}$ la esperanza medida sobre la variable aleatoria L. Esta toma la forma de diversos subconjuntos de aprendizaje siguiendo una distribución $\mathcal{P}$.

Dado un elemento conocido $(\vec{x}, y)$, se tiene que la esperanza media del error cuadrático cometido viene dada por:

$$E_{\mathcal{L}}[(y - d(\vec{x}, L))^2] = y^2 - 2yE_{\mathcal{L}}[d(\vec{x}, L)] + E_{\mathcal{L}}[d(\vec{x}, L)^2] \tag{5.1}$$

Considerando que por definición $d_A(\vec{x}) = E_{\mathcal{L}}[d(\vec{x}, L)]$ y aplicando la propiedad $E[Z^2] \geq E^2[Z]$

para toda distribución Z, se cumple la desigualdad:

$$E_{\mathcal{L}}[(y - d(\vec{x}, L))^2] \geq y^2 - 2yd_A(\vec{x}) + d_A^2(\vec{x}) = (y - d_A(\vec{x}))^2 \tag{5.2}$$

De donde se deduce que el predictor agregado tiende a comportarse mejor que los diversos predictores construidos. En el plano muestral, se tiene que d_B es un estimador de d_A, donde el conjunto de Bagging $\{\mathcal{L}^{(B)}\}$ es una muestra de la distribución $\mathcal{P}$

Este método funciona especialmente bien sobre bases de datos inestables, una gran dispersión en el conjunto de aprendizaje, hace que los modelos reducidos $d(\vec{x}, L)$ den resultados muy diversos. En estas situaciones, la desigualdad 5.2 se hace especialmente marcada, y los modelos obtenidos por Bagging sobresalen con respecto al resto. Por otra parte, en el caso de bases estables, los diversos predictores $d(\vec{x}, L)$ son similares los unos a los otros, por lo que d_B no ofrecerá ninguna información extra, a pesar de ser un método computacionalmente costoso.

5.1.3. Resultados experimentales

La principal razón por la que se emplea Bagging en la construcción de árboles, es porque en bases de datos reales se puede observar una disminución significativa del error cometido. Con el fin de presentar resultados reales, se mencionan conclusiones del artículo Breiman(1996) [2], en el que se estudia la mejor forma de construir estimadores de Bagging a partir de diversas bases de datos.

Para la construcción de clasificadores, se observa que 10 muestras de Boostrap tienden a ser suficientes para conseguir mejoras del 30 % en la disminución del error por clasificación incorrecta. Por lo general, el uso de conjuntos de Bagging más grandes no suelen ofrecer mejoras significativas de los modelos.

En el caso de predictores, por lo general se tiende a necesitar un mayor número de muestras para conseguir mejoras considerables. Sin embargo, se consiguen mejores resultados que en los clasificadores. Bajo los casos estudiados, usando conjuntos de Boosting de 25 elementos, se consiguen mejoras de en torno al 40 % del error cometido.

Un fenómeno que ha sido observado al aplicar estos métodos, es que si a los árboles no se les aplica el podado, los resultados finales no empeoran. La explicación que se le suele dar a esto, es que no podar proboca errores en áreas locales del espacio muestral. Al generar un gran número de modelos, estos fallos puntuales tienden a suavizarse al combinar todos los árboles.

5.2. Boosting

En [12], Schapire R. E. plantea un nuevo método para la construcción de estimadores precisos. Boosting permite reducir el error cometido de cualquier clasificador, solo es necesario que sea un poco mejor que si la elección de clase fuese aleatoria. Se llamarán modelos "debiles."[a] aquellos que no sean mucho mejores que la clasificación aleatoria.

Boosting se basa en la generación continua de modelos "débiles", para finalmente recombinarlos en un clasificado compuesto. En este último paso se le da más importancia a aquellos modelos parciales que se ajustan mejor al conjunto de aprendizaje.

Cada una de las parejas $(\vec{x}_i, Y_i)$, tienen asociado un peso w_i, que se aplica a la función de error a la hora de construir el árbol. De esta forma, los puntos con un mayo w, tienen mayor importancia en el modelo. En cada paso, estos varían, haciendo que cada uno de los árboles d_m sean distintos entre sí.

5.2.1. AdaBoost

Entre las distintas propuestas de Boosting, la que ha tenido más popularidad ha sido el algoritmo AdaBoost, aquí se hablará de los trabajos de Freund Y. y Schapire R. E., en concreto de [10], ya que presenta las bases necesarias como para poder aplicar el algoritmo a problemas reales.

Algoritmo AdaBoost.M1

Entrada: conjunto de aprendizaje de tamaño M, $\mathcal{L} = (\vec{x}_i, Y_i)$, con $Y \in \mathcal{C} = \{1, \dots J\}$
Inicializar los pesos $w_i = 1/M$
For $t \in \{1, \dots T\}$:
    Construir modelo d_t a partir de los pesos w_i
    Calcular error, $\epsilon_t = \sum_{d_t(\vec{x}_i) \neq Y_i} w_i$
    Asignar $\beta_t = \epsilon_t / (1 - \epsilon_t)$
    Actualizar pesos, si $d_t(\vec{x}_i = Y_i)$, $w_i = \beta_t w_i$
    Normalizar los pesos a 1
Modelo final, $d(\vec{x}) = arg\,\text{máx}_{j \in \mathcal{C}} \sum_{t/d_t(\vec{x})=j} log\frac{1}{\beta_t}$

Este planteamiento del problema sirve únicamente para creación de clasificadores. En el caso de que se quiera usar en predictores, se puede modificar cambiando la definición del error a $\epsilon_t = \sum_{(\vec{x}_i, Y_i)} w_i (d_t(\vec{x}_i) - Y_i)^2$. Sin embargo, esto puede introducir nuevos inconvenientes, y requiere un estudio del problema más profundo. De momento el trabajo se centrará en árboles de clasificación.

En cada paso del algoritmo se construye un modelo d_t distinto, este no tiene por qué ser un árbol, incluso hay estudios de como afecta Boosting a los distintos métodos de aprendizaje, véase [10].

Centrándonos en la creación de árboles de decisión, es necesaria la introducción de los w_i en el proceso de construcción. La idea detrás de estos pesos, es hacer que los modelos se centren en clasificar correctamente aquellos puntos con un mayor w.

En el caso de árboles, este proceso es un poco más complicado, ya que la construcción de estos modelos no está preparada para introducir pesos en el conjunto de aprendizaje. La solución propuesta es similar a las muestras Boostrap, coger con reemplazo elementos de $\mathcal{L}$, para construir un conjunto de entrenamiento de d_t. Se aprovecha que los w's tienen forma de distribución para hacer que la probabilidad de tomar cada punto sea w_i. De esta forma se consigue que los árboles tiendan a clasificar correctamente los puntos de mayor peso.

Notar que para el primer modelo, los pesos son uniformes, de tal forma que el árbol que se genera, es el mismo que si no se estuviese aplicando Boosting.

La eficacia del algoritmo se basa en cómo se actualizan los pesos. Los nuevos modelos deben ser capaces de clasificar correctamente aquellos puntos que los antiguos fallaron. De esta forma se introduce el factor $\beta_t \in [0, 1)$, que reduce el peso de las clasificaciones acertadas. Antes de avanzar al siguiente paso, se renormalizan los $w's$ para que mantengan la forma de distribución.

Una vez repetido este proceso un número T de veces, se construye el modelo final. Dado un vector de medidas $\vec{x}$, se observa en que clase lo clasifican cada uno de los d_t, y se le asigna aquella clase j que maximice $\sum_{t/d_t(\vec{x})=j} \log \frac{1}{b_t}$.

En cada paso se calcula un β_t, este debe tender a cero si el error del modelo es bajo, y preferiblemente menor que uno. Experimentalmente se ha comprobado que $\beta_t = \epsilon_t/(1-\epsilon_t)$ tiende a construir modelos precisos. Es capaz de mantener el equilibrio entre sus dos funciones, modificar los pesos w_i y cuantificar la influencia de d_t en el modelo final.

A pesar de que se ha comprobado que este método funciona especialmente bien con modelos "débiles", tiene problemas graves si el error cometido es mayor que $\epsilon_t > 1/2$. Esta condición es complicada de cumplir si el número de clases disponibles es elevado.

El algoritmo AdaBoost.M2 intenta corregir esto. Para ello aumenta la dimensión del problema, haciendo que cada pareja punto/clase (i, j) tenga un peso distinto, y que los modelos parciales hagan hipótesis $d_t(i, j) \in [0, 1]$. Si se quiere llevar a árboles, será necesario el uso de estimadores de la probabilidad de clasificación. El algoritmo es algo más complicado, pero hay pruebas experimentales de que si el número de clases es elevado, esta versión da resultados mucho mejores que la M1.

5.2.2. Comparación con Bagging

Por lo general, en experimentos se ha comprobado que Boosting tiende a generar modelos más precisos que Bagging. Esto se debe a que la formación de los conjuntos de aprendizaje parciales es menos arbitraria, contiene más información. Esta diferencia se reduce a medida los árboles generados son más complejos.

En [3], Breiman L. hace sus propios experimentos comparando Bagging con Boosting. Llega a la conclusión de que el proceso de votación con pesos del AdaBoost hace que los modelos finales tiendan a ser mas precisos que los obtenidos por Boosting.

5.3. Ejemplos

Por último, se presentan una serie de modelos construidos sobre un mismo conjunto de aprendizaje con el fin de estudiar cómo se comportan. Como base de datos se usará una vez más "*Wine Data Set*"de [7], que trata de clasificar el cultivo al que pertenece una serie de vinos a partir de un análisis químico.

Se ha podido comprobar que esta base de datos se clasifica realmente bien, en ejemplos anteriores se han conseguido precisiones elevadas a partir de modelos muy básicos. Con el fin de mostrar la potencia de los métodos más complejos, se excluyen del estudio aquellas variables que discriminan bien entre cultivos, en concreto la cantidad de alcohol, flavanoides, proteínas OD280.OD315, prolinas y la intensidad del color del vino.

Con el fin de realizar un estudio más profundo de los modelos, se separa la base de datos entre un conjunto de entrenamiento y uno de validación. El primero se construye cogiendo al azar el 80 % de los elementos totales.

5.3.1. Árbol de clasificación

Con el fin de mostrar un modelo de referencia, se construye un árbol clásico a partir de los mismos datos que se usarán para el resto de modelos. Para este paso se utiliza el paquete *caret* [11], ya que ofrece una recopilación de los distintos métodos de entrenamiento. En concreto, se utiliza la creación de árboles del *rpart* reforzado por una implementación de la validación cruzada.

El modelo presentado en 5.1 tiene un error de validación $R^v = 0,269$. Además de este, se intentaron construir varios árboles a partir de la misma información, sin conseguir una mejora

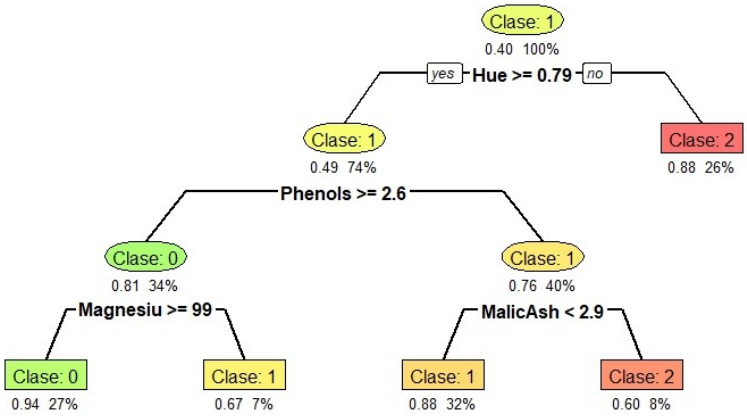


Figura 5.1: Árbol de clasificación

considerable en el error cometido. No parece sencillo construir un modelo simple satisfactorio a partir de la base de datos con la que se está trabajando, lo que exige la aplicación de métodos más complejos.

5.3.2. Bagging y Boosting

Se tiene que los modelos construidos por Bagging y Boosting son bastante similares, por lo que se estudian a la vez con el fin de compararlos entre sí. Ambos métodos se basan en la formación sucesiva de muestras de Boostrap a partir del conjunto de entrenamiento, esta elección tiene una componente aleatoria, lo que dificulta un estudio preciso de la metodología.

A falta de un estudio preciso de como se comportan ambos métodos en función del número de replicas Boostrap creadas y el método utilizado para construir cada uno de los árboles individuales, se opta por crear una serie de modelos de ambos tipos bajo las mismas condiciones y comparar el error de validación medio.

Para la creación de modelos se usa el paquete "*adabag*"de R [1], ya que ofrece implementaciones de ambos algoritmos empleando el mismo método de construcción de árboles. En todos los casos se construirán clasificadores a partir de 20 muestras diferentes. En Boosting se aplica el algoritmo AdaBoost.M1 tal cual introducido en este Capítulo, con los coeficientes de Freund $\beta_t = \epsilon_t/(1-\epsilon_t)$. En 5.2 y 5.3 se muestran los cuatro primeros árboles construidos a partir de una de las iteraciones de cada uno de los métodos aquí estudiados.

Observando a simple ojo los árboles construidos por ambas metodologías, no es posible apreciar ninguna diferencia fundamental las dos series. Todos han sido construidos bajo las mismas

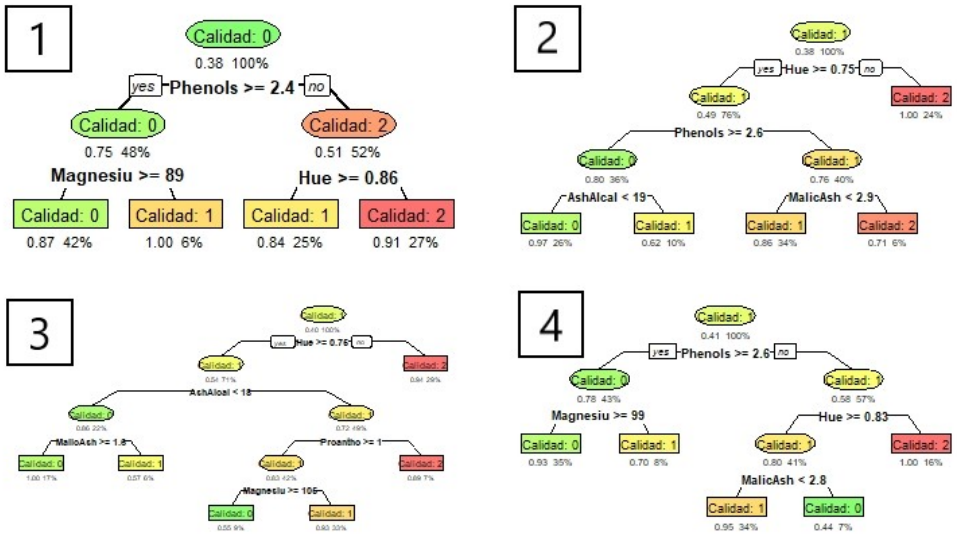


Figura 5.2: Sucesión de árboles obtenida aplicando Bagging

condiciones, variando ligeramente el conjunto de aprendizaje.

La diferencia fundamental entre Bagging y Boosting, es que el segundo utiliza información del árbol anterior en la construcción del siguiente. En estos experimentos, se puede observar que los modelos Boosting tienden a tener un menor ratio de validación. Haciendo media sobre 10, se mide $R^v = 0,0847$ para los construidos por AdaBoost y $R^v = 0,172$ para los Bagging.

A pesar de ser claramente inferiores, los modelos construidos por Bagging son claramente superiores al árbol de clasificación estándar. La única desventaja de estos métodos es el tiempo, cada proceso consta de la fabricación de 20 modelos simples, lo que retrasa enormemente su creación.

5.3.3. Random Forest

En [4], Breiman propone un nuevo método para la generación de modelos a partir de árboles de decisión. Random Forest es una versión de su anterior Bagging que ofrece resultados tan precisos como los Boosting con un coste computacional incluso menor.

En este método, para la construcción de cada árbol parcial, no solo se escoge de manera aleatoria el conjunto de entrenamiento, sino que también se limitan la cantidad de separaciones posibles. Este proceso sigue una actualización de costes similar al algoritmo AdaBoost.

Para el experimento, se usará la implementación presente en el método rf"del paquete [11] de R. Tras hacer una serie de experimentos aplicando Random Forest con 20 pasos, se ha conseguido

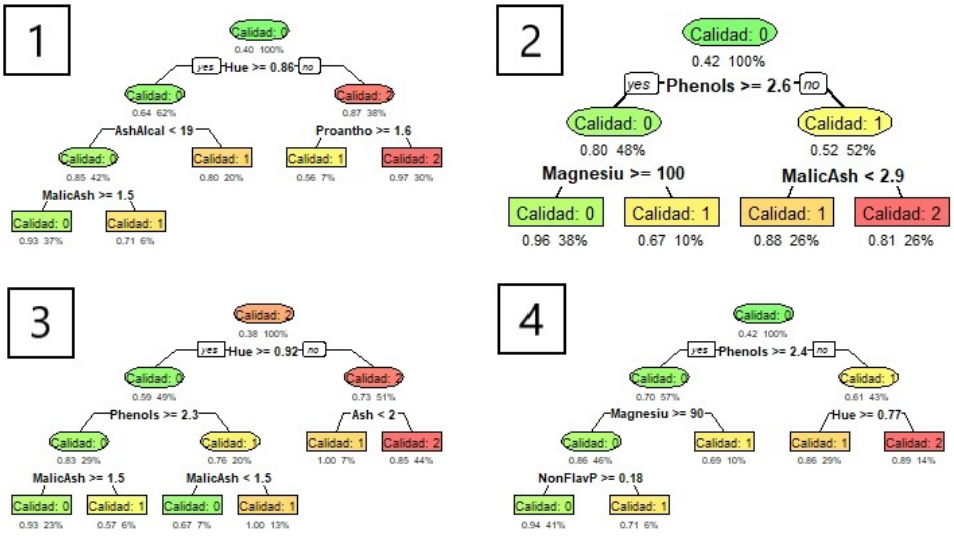


Figura 5.3: Sucesión de árboles obtenida aplicando AdaBoost M.1

una media $R^v = 0,00723$. Los modelos obtenidos de esta forma son realmente precisos, y sin entrar en un estudio preciso de los árboles parciales o el tiempo de ejecución, si que tiende a parecer que en este método consigue mejores resultados de manera más eficiente que los AdaBoost del apartado anterior.

Bibliografía

[1] Alfaro, E., Gámez, M., and García, N. adabag: An R package for classification with boosting and bagging. *Journal of Statistical Software 54*, 2 (2013), 1–35.

[2] Breiman, L. Bagging predictors. *Mach. Learn. 24*, 2 (aug 1996), 123–140.

[3] Breiman, L. Rejoinder: Arcing classifiers. *The Annals of Statistics 26*, 3 (1998), 841–849.

[4] Breiman, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[5] Breiman, L., Friedman, J., Stone, C., and Olshen, R. *Classification and Regression Trees*. Taylor & Francis, 1984.

[6] Devroye, L., Györfi, L., and Lugosi, G. *A Probabilistic Theory of Pattern Recognition*. Stochastic Modelling and Applied Probability. Springer New York, 2013.

[7] Dua, D., and Graff, C. UCI machine learning repository, 2017.

[8] Efron, B. *Bootstrap Methods: Another Look at the Jackknife*. Springer New York, New York, NY, 1992, pp. 569–593.

[9] Efron, B., and Tibshirani, R. *An Introduction to the Bootstrap*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1994.

[10] Freund, Y., and Schapire, R. E. Experiments with a new boosting algorithm. In *ICML* (1996).

[11] Kuhn, M. *caret: Classification and Regression Training*, 2022. R package version 6.0-92.

[12] Schapire, R. E. The strength of weak learnability. *Machine Learning 5*, 2 (1990), 197–227.

[13] Therneau, T., and Atkinson, B. *rpart: Recursive Partitioning and Regression Trees*, 2022. R package version 4.1.16.