

Sistemas de Aprendizaje Automático

*Conforme a contenidos del «Curso de Especialización
en Inteligencia Artificial y Big Data»*



Sistemas de
Aprendizaje_Automático

Universidad de Castilla-La Mancha

Escuela Superior de Informática
Ciudad Real

Índice general

4. Modelos basados en Árboles de Decisión	1
4.1. Árboles de Decisión	1
4.1.1. Importancia de las características	5
4.2. Conjuntos de Modelos	6
4.2.1. Árboles Individuales	8
4.2.2. Predicciones	10
4.2.3. Importancia de los características	12
4.3. Selección y Optimización de Modelos	13
4.3.1. Selección de Modelos	13
4.3.2. Hiperparámetros	13
4.3.3. Parametrización	13
4.3.4. Optimización de Hiper-parámetros	15

Listado de acrónimos

4

Capítulo

Modelos basados en Árboles de Decisión

Francisco P. Romero

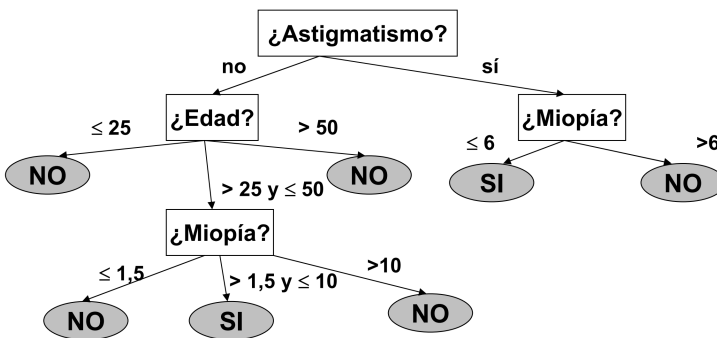
4.1. Árboles de Decisión

Los árboles de decisión son una **técnica de aprendizaje automático por inducción** que permiten identificar conceptos (clases de objetos) a partir de las características de un conjunto de ejemplos que los representan. La información extraída de los mismos queda organizada jerárquicamente en **forma de árbol**, es decir, en forma de **grafo dirigido que consta de nodos y arcos**. Los nodos corresponden a una pregunta o a un test que se hace a los ejemplos [Qui96b].

El árbol de decisión (ver Figura 4.1) se construye a base de ir haciendo preguntas sobre características determinadas a los ejemplos y clasificándolos según la respuesta. Por tanto un árbol de decisión trabaja como un "clasificador". Las diferentes opciones de clasificación (respuesta a las preguntas) son excluyentes entre sí, lo que hace que a partir de casos desconocidos y siguiendo el árbol adecuadamente, se llegue a una única conclusión o decisión a tomar.

La construcción de un árbol de decisión requiere:

1. Un **conjunto de ejemplos** representativos de lo que se desea aprender (Conjunto de entrenamiento);
2. Una **representación simbólica del conocimiento** (Ejemplos y definición de sus características) a través de atributos y sus valores;
3. Un **algoritmo de aprendizaje** (clasificación o regresión) [BFOS17];
4. Un método de **evaluación** o esquema de valoración



¿Cuándo se hace una operación?

Si Astig. = No Y $25 < \text{Edad} \leq 50$ Y $1,5 < \text{Miopía} \leq 10$ **Entonces** Sí

Si Astig. = Sí Y $\text{Miopía} \leq 6$ **Entonces** Sí

En cualquier otro caso No

Figura 4.1: Ejemplo de árbol de decisión.

Un árbol de decisión tiene un nodo raíz, nodos intermedios y hojas. Cualquier nodo intermedio puede ser un nodo raíz de un subárbol. Esto conduce a una definición recursiva de árbol de decisión. Cada nodo intermedio y la raíz tienen asociados separadores que formulan una pregunta o realizan un test acerca de la existencia o no de una característica en cada caso ejemplo. Esto permite clasificar los ejemplos determinar cuáles serían los nodos sucesores.

Las hojas del árbol de decisión representan los conceptos extraídos de manera automática, es decir, una hoja en el árbol corresponde a un conjunto de ejemplos que representan una sola clase. La clase de la hoja se asigna por el criterio de a la que pertenezcan la mayoría de los ejemplos en ella. . Una vez construido un árbol de decisión, un nuevo ejemplo desconocido será representante de la clase en donde caiga recorriendo el árbol desarrollado desde la raíz a las hojas.

El atributo que se selecciona como separador dentro de un nodo debe de cumplir el objetivo de que su posición en algún punto del árbol genere un subárbol tan simple como sea posible y dé una concreta clasificación. De esta forma, cuando se construye un árbol de decisión, es necesario tener un medio para determinar tanto los atributos importantes requeridos para la clasificación, como el orden de uso de esos atributos importantes. Es por ello por lo que es clave la elección del criterio de selección de separadores lo cual dará lugar a diferentes algoritmos de construcción de árboles de decisión.

Algoritmo

El algoritmo general para todo método basado en árboles de decisión sería el siguiente:

1. **Se calcula la calidad** (criterio de selección de separadores, impureza, ganancia de información, etc.) **del nodo** (conjunto de ejemplos).
2. **Si un conjunto de ejemplos en un nodo no tiene suficiente calidad, se calcula el mejor atributo separador.** Una vez que se obtiene, se añade a la base de reglas, y **se utiliza para dividir el conjunto de ejemplos en al menos dos conjuntos nuevos de mayor calidad.**
3. **Seguir separando hasta que se cumpla el criterio de parada.**

Criterios de Selección

Los **criterios de selección basados en la calidad del nodo** (ganancia de información) más habituales en clasificación son los siguientes:

- **Índice de Entropía:** se calcula el índice de entropía para cada una de las variables: Si la variable es categórica, se obtiene sumando el índice de entropía de cada una de sus clases. En cambio, si es numérica, previamente se obtiene uno o varios puntos de corte por métodos iterativos [Qui96a]. **Se elegirá aquella variable que tenga menor índice de entropía.** La entropía se calcula según la siguiente fórmula:

$$H(\mathcal{S}) = - \sum p_i \log p_i$$

donde, where p_i es la probabilidad de aparición de la clase i



- Consideremos que se dispone de 110 manzanas. De ellas, 89 son verdes y 21 rojas. Para calcular la entropía de este conjunto se necesita la probabilidad de cada clase.
- Si existen 89 manzanas verdes de 110, la probabilidad de las manzanas verdes es 0,8091. Además, esta probabilidad de la manzana verde multiplicada por el logaritmo es igual a 0,2473.
- Para las manzanas rojas. Se dispone de 21 de 110, por lo que la probabilidad resultante es 0,1909. Al multiplicar la probabilidad por el logaritmo de la probabilidad obtenemos 0,4561.
- La Entropía de este subconjunto de datos globales viene dada la suma de las entropías individuales, entonces, $0,2473 + 0,4561 = 0,7034$

- **Índice de Gini:** mide la probabilidad de no sacar dos registros con el mismo valor para la variable objetivo dentro del mismo nodo (ver Tabla 4.1. **Cuando menor es el índice de Gini mayor es la pureza del corte.** Por lo tanto, el corte propuesto en primer lugar será el de aquella variable que tenga menor valor del índice de Gini.

$$Gini = \sum_{i=1}^C p(i) * (1 - p(i))$$

donde, C es el número total de clases y $p(i)$ la probabilidad de encontrar elementos de la clase i en el conjunto de elementos que se está analizando.

Ejemplos Clase 1	Ejemplos Clase 2	Cálculo de GINI
5	5	$0,5 * (1 - 0,5) + 0,5 * (1 - 0,5) = 0,5$
5	0	$1 * (1 - 1) + 0 * (1 - 0) = 0$
5	1	$1/6 * (1 - 1/6) + 5/6 * (1 - 5/6) = 0,278$

Tabla 4.1: Ejemplo de cálculo del índice de Gini




- *Entropía/Ganancia de información:* Sesgado hacia atributos con muchos valores diferentes.
- *Índice de Gini:* Funciona peor cuando hay muchas clases y tiende a favorecer particiones de tamaño y pureza similares.

Criterios de Parada y Poda

Además, otro de los aspectos más importantes al construir un árbol de decisión es el **criterio de parada** con el fin de evitar el sobreaprendizaje. Los más habituales son los siguientes:

- **Máximo número de ramas por nodo** (*Maximum Branch*): número de ramas máximo en que puede dividirse un nodo.
- **Número mínimo de observaciones por nodo final** (*Leaf Size*): número mínimo de observaciones que tiene que tener un nodo final para que se construya la regla.
- **Número mínimo de observaciones para dividir un nodo** (*Split Size*): número mínimo de observaciones que tiene que tener un nodo para que se pueda cortar por la variable seleccionada.
- **Variables discriminantes** (*Discriminant Variables*): no encontrar ninguna variable que sea lo suficientemente discriminante en el nodo es motivo de parada.

En último lugar es recomendable podar (pruning) el árbol. Consiste en reducirlo, haciéndolo más sencillo dejando sólo los nodos más importantes y a su vez eliminando los redundantes. Para ello, antes de llamar recursivamente al algoritmo para cada nodo, se calcula tanto la tasa de ramificación del conjunto actual como de los subconjuntos, además de comprobar otros valores como el número de nodos de cada hoja, etc. A partir de esta tasa de ramificación se puede decidir si seguir desarrollando el árbol o no.



La poda se basa en un algoritmo de mínimo coste/complejidad de poda considerando 1) el coste de clasificación errónea del árbol, 2) el número hojas actuales del árbol y 3) el coste de introducir una nueva hoja al mismo.

4.1.1. Importancia de las características

La importancia de cada característica es su contribución relativa a las predicciones del objetivo. Así, una característica con mayor importancia tendrá un mayor impacto en las predicciones. Habitualmente este factor se calcula estimando la reducción agregada y normalizada del error de predicción de cada característica para cada división de un árbol (estas mismas estimaciones se pueden utilizar en los procesos de poda) [Bre01].

Por ejemplo, si obtenemos como resultado el árbol que se muestra en la Figura 4.2, se puede observar como en el nodo raíz la variable *temperature* es la que se utiliza para la división y se vuelve a utilizar en el segundo nivel de los árboles. Por otro lado, la variable *humidity* únicamente se utiliza en el segundo nivel y la variable *windspeed* no aparece en el árbol. Los valores concretos de importancia de cada una de las características se puede observar en la Tabla 4.2. Qué el atributo *windspeed* obtenga un valor de cero sólo significa que el modelo prefiere la utilización de otras características al elegir las divisiones según el criterio establecido.

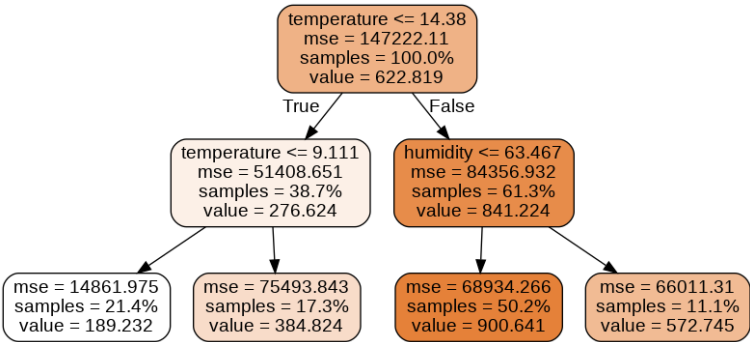


Figura 4.2: Ejemplo de árbol de decisión.

Característica	Importancia
temperature	0,89
humidity	0.11
windspeed	0.00

Tabla 4.2: Importancia de las variables a partir de un árbol de decisión



- Los modelos basados en árboles de decisión se construyen dividiendo los datos en particiones para que cada una de ellas maximice la ganancia de información para modelos de clasificación o minimice el error medio cuadrático para los modelos de regresión.
- Cada partición, tiene un predicado asociado que la define, por ejemplo, $b < 1,0$.
- El proceso de partición de las instancias de un conjunto de datos es recursivo, es decir, las particiones se dividen a su vez en otras más pequeñas, formando así una jerarquía de particiones con sus predicados asociados.
- En cada paso de este proceso, también se calcula una serie de estadísticas de resumen, como el número de instancias que pertenecen a la partición, su confianza, etc.

4.2. Conjuntos de Modelos

Los modelos individuales de aprendizaje supervisado son capaces de resolver un buen número de problemas de clasificación y regresión, sin embargo, otros muchos de estos problemas no se pueden resolver con un solo modelo.

Uno de los inconvenientes de los modelos individuales es que puede sobreaprender en el caso de hacerse demasiado complejos, es decir, sobreajustar sus datos, por lo que su rendimiento sobre los datos de entrenamiento resultaría excelente, pero no se generalizan bien a los datos nuevos. Esto hace que los modelos individuales de árboles de decisión sean peores modelos para su uso habitual. Para solucionar este problema aparecen los conjuntos de modelos.

Un conjunto de modelos (*ensemble*) es una colección de árboles de decisión múltiples que se combinan para crear un modelo más sólido con un mejor rendimiento predictivo. Un conjunto de modelos basado en el remuestreo de los datos puede convertirse en un buen predictor al promediar los errores de cada modelo individual. En general, los conjuntos realizan su tarea de clasificación o regresión mejor que un solo árbol de decisión porque son menos sensibles a los valores atípicos en sus datos de entrenamiento lo cual les ayuda a mitigar el riesgo de sobreajuste y generalizar mejor cuando se los aplica a nuevos datos.

En resumen, dependiendo de la naturaleza de sus datos y los valores específicos para los parámetros del conjunto, puede aumentar significativamente el rendimiento sobre el uso de un solo modelo además de permitir la paralelización para la construcción y explotación del mismo.

La idea básica es remuestrear los datos y calcular las predicciones sobre el conjunto de datos remuestreados. Al promediar varios modelos conjuntamente se obtiene un mejor ajuste debido a que se reducen tanto los modelos con sesgo como los modelos con alta varianza. Si el algoritmo predice datos categóricos, entonces el voto de la mayoría dará la clase dominante o con mejor predicción. Si se está realizando predicción sobre datos numéricos, entonces se realizará la media sobre las predicciones.



- **Sesgo (Bias):** El sesgo es la diferencia entre el valor proporcionado por el modelo y el valor real (error).
- **Varianza:** Cantidad de variación de la predicción del modelo según los datos que utilizemos se usen para entrenamiento.

Para saber más: Thomas G. Dietterich. Ensemble Methods in Machine Learning. Link: <http://web.engr.oregonstate.edu/~tgdp/publications/mcs-ensembles.pdf>

Las técnicas que se pueden utilizar para construir conjuntos de árboles de decisión se basan en los métodos de Bagging/Boosting tal y como se explica a continuación.

- **Bagging** (Bootstrap Aggregating) [D⁺02]: este algoritmo crea cada árbol individual a partir del aprendizaje sobre una muestra aleatoria de los elementos del conjunto de datos de entrenamiento. Por defecto las muestras son tomadas utilizando un ratio del 100 %, reduciéndose si es conjunto de datos es muy grande, con reemplazo, esto es, cada miembro del conjunto es entrenado con una muestra diferente del conjunto de entrenamiento. El tamaño es el mismo que el conjunto de entrenamiento original pero no su composición. El resultado final es el promedio de la predicción de cada uno de los miembros del conjunto. Es la estrategia más simple de las tres, pero obtiene unos rendimientos muy buenos sobre otras estrategias más complejas y costosas.
- **Random Forests:** sigue una estrategia similar al Bagging pero añadiendo un elemento adicional de aleatoriedad eligiendo para cada árbol un subconjunto de las características del conjunto de entrenamiento [Bre01]. En subespacios aleatorios (*random subspaces*) [Ho98], cada miembro es entrenado con todos los ejemplos, pero con un subconjunto de los atributos. La dimensión de los subespacios es uno de los parámetros de este algoritmo. De nuevo el resultado es el promedio de los árboles obtenidos. Es uno de los métodos más precisos de todos los métodos de clasificación.

- **Boosted Trees** (Boosting, Adaboost, Gradient Boosted Trees) en este caso el algoritmo **construye secuencialmente un conjunto de árboles de decisión que supone modelos débiles** (*weak learners*). Un modelo débil es cualquier conjunto de aprendizaje que es al menos un poco mejor que la predicción aleatoria (>50 %). Posteriormente combina sus salidas de forma aditiva para obtener una predicción final. En cada iteración del algoritmo, cada modelo individual intenta corregir los errores cometidos en la iteración previa mediante la optimización de una función de pérdida [FSA99].

Todos estos algoritmos tienen en común que están compuestos por varios árboles individuales y su resultado se combina para obtener una predicción final. El conjunto siempre tiene un mejor rendimiento que cada uno de los clasificadores o regresores individuales que lo componen. La principal diferencia entre ellos reside en la forma en que crecen los árboles individuales y la forma en que se combinan sus predicciones para obtener la predicción final del conjunto.

En los siguientes apartados se detallan las principales características y diferencias de estos métodos.

4.2.1. Árboles Individuales

Tanto en las técnicas de *Bagging* como en las de *Random Forests* cada árbol individual intenta predecir el campo objetivo utilizando cierto nivel de aleatoriedad, ya sea seleccionando un porcentaje aleatorio de las instancias del conjunto de datos (*Bagging*) y/o seleccionando un subconjunto aleatorio de los campos de entrada en cada división (*Random Forests*). Sin embargo, **en los *Boosted Trees*, cada árbol en lugar de predecir el campo objetivo, intenta aprender de los errores cometidos por el modelo anterior ajustando un paso de gradiente para minimizar el error del clasificador anterior.**



El **descenso de gradiente** [Mor94] es un algoritmo que estima numéricamente dónde una función genera sus valores más bajos, es decir estima mínimos locales. Si comenzamos en un punto x_0 y nos movemos una distancia positiva α (**paso del gradiente**) en la dirección del gradiente negativo, entonces nuestro nuevo y mejorado x_1 será

$$x_1 = x_0 - \alpha \nabla f(x_0)$$

$$x_{n+1} = x_n - \alpha \nabla f(x_n)$$

Es decir, a partir del punto de partida se mejora iterativamente hasta conseguir el mínimo local.

Para problemas de regresión, el error se mide habitualmente mediante el error cuadrático medio (la diferencia al cuadrado entre los verdaderos objetivos y la predicción actual). Para los problemas de clasificación, se entrena un árbol para cada clase en cada iteración. Las puntuaciones de los árboles se normalizan mediante

la función *softmax* para obtener una distribución de probabilidad sobre las clases dadas en un punto de datos. El error es la diferencia entre esta distribución y la "verdadera distribución" sobre las clases para los puntos de datos, que es uno para la clase correcta y cero para para todas las demás.



En matemáticas, la función *softmax*, o función exponencial normalizada es una generalización de la Función logística. Se emplea para comprimir un vector K -dimensional z , de valores reales arbitrarios en un vector K -dimensional, $\sigma(z)$, de valores reales en el rango $[0, 1]$. La función está dada por:

$$\sigma : \mathbb{R}^K \rightarrow [0, 1]^K$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$

para $j = 1, \dots, K$. En teoría de la probabilidad, la salida de la función *softmax* puede ser utilizada para representar una distribución categórica— la distribución de probabilidad sobre K diferentes posibles salidas.

Otra característica de los *Boosted Trees* es que tienen un peso asociado a cada árbol que mide su importancia para calcular la predicción final del conjunto. Los pesos se eligen mediante un procedimiento de búsqueda en la que los posibles pesos se evalúan en un grupo de puntos de prueba para encontrar un peso que sea casi óptimo. Los puntos de prueba para este procedimiento de búsqueda se seleccionan aleatoriamente de los datos de entrenamiento. Por último, los pesos se multiplican por la tasa de aprendizaje.



En optimización, la estrategia de búsqueda de línea (*line search*) es uno de los dos enfoques iterativos básicos para encontrar un mínimo local x^* de una función objetivo $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Primero encuentra una dirección de descenso a lo largo de la cual la función objetivo f se reducirá y luego calcula un tamaño de paso que determina hasta qué punto x debe moverse a lo largo de esa dirección. La dirección de descenso puede ser calculada por varios métodos, como el descenso de gradiente o el método *quasi-Newton*. El tamaño del paso puede determinarse de forma exacta o inexacta.



La **confianza** se calcula como el límite inferior del *intervalo de puntuación de Wilson*, lo que significa que proporciona una confianza pesimista.



La tasa de aprendizaje o *learning rate* es un hiperparámetro utilizado para controlar la rapidez a la que un algoritmo actualiza las estimaciones de los parámetros o aprende los valores de los mismos.

4.2.2. Predicciones

Los conjuntos se componen de varios árboles. Cada árbol devuelve una predicción diferente dados los datos de entrada. Estas predicciones individuales deben combinarse para obtener una predicción final para el conjunto.

En el caso de los *Boosted Trees*, la agregación de predicciones del modelo único es aditiva en lugar de promediada. Esto es, sólo se obtienen las probabilidades de clase en el caso de los conjuntos de clasificación, ni la confianza ni el error esperado. En el caso de los conjuntos de regresión, la predicción final se genera calculando la suma de la predicción de cada árbol multiplicada por su peso. El error esperado no puede calcularse en estos modelos, por lo que no se devuelve una medida de calidad para los problemas de regresión. Las predicciones de los conjuntos de clasificación son similares, se realizan sumas ponderadas separadas para cada clase objetivo. El vector resultante de las sumas ponderadas se transforma en probabilidades de clase mediante la función softmax. De este modo, la probabilidad de cada una de las clases en el campo objetivo se devuelve para medir la calidad de la predicción.

En el caso de las técnicas de **Bagging** y de **Random Forests**, las predicciones de los árboles individuales se promedian para obtener una predicción final. Las mismas medidas de calidad obtenidas al construir un modelo individual también se obtienen para estos modelos: confianza y probabilidades para los problemas de clasificación, y error esperado, para los problemas de regresión.

■ Probabilidad:

- **Clasificación:** Las probabilidades por clase se promedian teniendo en cuenta todos los árboles que componen el conjunto. La clase con la mayor probabilidad es la clase ganadora. Por ejemplo, En la Tabla 4.3 se muestra un conjunto de clasificación construido con tres árboles de decisión que intentan predecir dos clases, "Verdadero" "Falso":

Árbol	Verdadero	Falso
1	80 %	20 %
2	40 %	60 %
3	60 %	40 %
Conjunto	60 %	40 %

Tabla 4.3: Ejemplo de clasificación utilizando probabilidad

- **Regresión:** Se promedian promedian las predicciones de los árboles que componen el conjunto. Por ejemplo, considerando de nuevo tres árboles diferentes en un conjunto pero prediciendo una salida numérica esta vez (ver Tabla 4.4). El resultado final va a ser $210 = [200 + 250 + 180]/3$ con un error de $1,98 = [2,4 + 2,1 + 1,45]/3$.

Árbol	Ventas	Error
1	200	2,40
2	250	2,10
3	180	1,45
Conjunto	210	1,98

Tabla 4.4: Ejemplo de regresión utilizando probabilidad

- **Confianza:**
 - **Clasificación:** La confianza por clase se promedian teniendo en cuenta todos los árboles que componen el conjunto. La clase con la mayor confianza es la clase ganadora. Se calcula de la misma manera que la predicción de la Tabla 4.3, pero utilizando las confianzas por clase en lugar de las probabilidades.
 - **Regresión:** Se promedian las predicciones de los árboles que componen el conjunto de la misma manera que se explica para las probabilidades (Tabla 4.4) en este caso de forma ponderada de acuerdo al error esperado.
- **Votación:**
 - **Clasificación:** Cada una predicciones que arrojan los árboles del conjunto se consideran un voto. Los "votos"de una clase determinada es el porcentaje de árboles del conjunto que votan por esa clase. En la Tabla 4.5 se muestra un ejemplo de un conjunto de clasificación construido con tres árboles de decisión para predecir dos clases, "Verdadero" o "Falso". Como hay más árboles que predijeron la clase "Verdadero"(dos frente a un solo árbol que predijo la clase "Falso"), la predicción final es "Verdadero" con el 66,67 % de los votos ($2/3 = 0,6667$). Si dos o más clases tienen el mismo número de votos se tendrá que establecer un sistema de prioridad, lo cual normalmente se resuelve por orden alfabético o de forma aleatoria.

Árbol	Clase Resultado
1	Verdadero
2	Falso
3	Verdadero
Conjunto	Verdadero

Tabla 4.5: Ejemplo de clasificación por el método de la votación

- **Regresión:** Para los conjuntos de regresión, la opción de votos promedia las predicciones de los árboles que componen el conjunto. Ofrece los mismos resultados que la probabilidad.



En el caso de los problemas de regresión, todas las predicciones de los árboles individuales se promedian para obtener una única predicción. En el caso de los conjuntos de clasificación, la clase con la mayor confianza o probabilidad agrupada de todos los árboles es la que se obtiene como resultado. Existe una técnica adicional para calcular las predicciones de los problemas de clasificación denominada "votación". Se basa en el porcentaje de árboles que votan por cada clase en el conjunto para seleccionar la clase ganadora.

4.2.3. Importancia de los características

Al igual que con los árboles de decisión individuales, se puede obtener una medida de la importancia de una características en relación con las demás. Se calcula tomando una media ponderada de cuánto reduce cada característica el error de predicción del árbol en cada división.

Cabe destacar que los valores de importancia de los campos de los *Random Forests* suelen ser más significativos que los de un solo árbol (ver Tabla 4.6). En el caso de los árboles individuales esta medida no es del todo correcta, ya que supone que la estructura del árbol es correcta, pero para los conjuntos de modelos es una medida mucho más significativa.

Tabla 4.6: Tabla de Relevancia de variables obtenida mediante RandomForests

Característica	Importancia
temperature	0,68
humidity	0.23
windspeed	0.09



Algoritmo Adaboost [HRZZ09]

1. Inicialmente a todos los datos del conjunto de entrenamiento se les asigna un peso idéntico, $w_i = \frac{1}{n}$, donde n es el tamaño del conjunto de datos.
2. Se entrena el modelo usando el set de entrenamiento.
3. Se calcula error del modelo en el set de entrenamiento, se cuentan cuántos objetos han sido mal clasificados y se identifican cuáles son.
4. Se incrementan pesos en los casos de entrenamiento que el modelo calcula erróneamente.
5. Se entrena un nuevo modelo usando el conjunto de pesos modificados.
6. Volver al punto 3 (repetir hasta el número de iteraciones fijadas inicialmente).
7. Modelo final: votación ponderada por los pesos de todos los modelos

4.3. Selección y Optimización de Modelos

4.3.1. Selección de Modelos

No hay una respuesta fácil a la pregunta de qué modelo produce los mejores resultados. Por lo general se realiza una prueba experimental de las opciones disponibles. Es cierto, que dependiendo de las características del conjunto de datos, se puede disponer de una idea inicial de cuál puede tener un mejor rendimiento.

En los *Boosted Trees*, el efecto de los árboles adicionales es esencialmente una expansión del espacio de hipótesis, esto no ocurre en las otras dos técnicas. Por lo tanto, si se espera que la función de decisión sea muy compleja y se dispone de muchos datos, el *boosting* puede funcionar mejor. Por otro lado, si el conjunto de datos tiene mucho ruido y el sobreajuste es un problema, tanto el *Bagging* como *Random Forests* pueden ser la mejor opción ya que el *boosting* es más vulnerable al ruido en los datos.

Por último, si se sospecha que se enfrenta una función "fácilmente aprendible", la potencia adicional que ofrecen el *Boosting*, o incluso los *Random Forests*, puede que no sirva de nada; un método más sencillo, como el *Bagging* puede funcionar mejor que las otras dos opciones.

4.3.2. Hiperparámetros

Un parámetro puede considerarse interno al modelo y puede obtenerse después de que el modelo haya aprendido de los datos. Ejemplos de parámetros son los coeficientes de regresión en la regresión lineal, los vectores de soporte en las máquinas de vectores de soporte y los pesos en las redes neuronales.

Un hiperparámetro puede considerarse externo al modelo y puede ser fijado arbitrariamente por el programador. Algunos ejemplos de hiperparámetros son la k en los vecinos más cercanos, el número de árboles y el número máximo de características en *RandomForest*, la tasa de aprendizaje en los *BoostedTrees*. Estos hiperparámetros sirven para construir el modelo antes de pasar a su fase de entrenamiento y test.

4.3.3. Parametrización

En los conjuntos de modelos existen diferentes parámetros relevantes [Lou14]. Uno de los más importantes es el número de árboles de decisión que van a conformar el *ensemble* en los casos del *Bagging* y del *Random Forests*. Por defecto, el número de modelos se recomienda que se establezca en un mínimo de 10 y un máximo 1.024 árboles. En el caso de los *Boosted Trees*, el número de modelos vendrá determinado por el número de iteraciones, recomendándose un máximo de 2.048.

En general, el aumento del número de árboles dará mejores resultados. Además, no hay ningún inconveniente excepto un mayor tiempo de cálculo. Las situaciones en las que es probable que un mayor número de modelos proporcione una mayor mejora en las que el conjunto de datos no es muy grande (por ejemplo, en los miles de instancias o menos), los datos son muy ruidosos y cuando hay muchas características correlacionadas en el caso de la utilización de *Random Forests*. Hay que tener en cuenta que cada modelo adicional tiende a ofrecer una mejora marginal menor, por lo que si la diferencia entre nueve y diez modelos es muy pequeña, es muy poco probable que un undécimo modelo suponga una gran diferencia.

Se puede fijar un **umbral de crecimiento** de un árbol dentro del conjunto. Un umbral más bajo simplifica el conjunto y ayuda a evitar el sobreajuste. Sin embargo, también puede reducir el poder de predicción del conjunto en comparación con conjuntos más profundos. El número ideal de nodos puede depender del tamaño del conjunto de datos y del número de características. Los conjuntos de datos más grandes con muchas características importantes pueden requerir conjuntos más complejos. Reducir el número de nodos también puede ser útil para obtener una comprensión inicial de los patrones de datos básicos para posteriormente, crecer el conjunto a partir de ahí.

En los *Random Forests* cada árbol del conjunto se construye a partir de una muestra extraída con reemplazo del conjunto de entrenamiento. Además, al dividir cada nodo durante la construcción de un árbol, la mejor división se encuentra a partir de todas las características de entrada o de un subconjunto aleatorio de tamaño (**máximo de características**). Este es un parámetro importante que normalmente se suele fijar como la raíz cuadrada o el logaritmo del número de características totales.

Por último, los siguientes parámetros son propios de los *Boosted Trees*.

- Las opciones de **parada temprana** (early stopping) tratan de encontrar el número óptimo de iteraciones probando los modelos individuales después de cada iteración y dando lugar a una parada temprana si no se produce una mejora significativa. En consecuencia, el total de iteraciones para los *Boosted Trees*, puede ser inferior al establecido en el parámetro de número de iteraciones.
- La **tasa de aprendizaje** (learning rate), también conocida como paso de gradiente, controla la agresividad con la que el algoritmo se ajusta a los datos. La elección de este valor afecta desde dos puntos de vista 1) la rapidez con la que el algoritmo aprende y 2) si la función de coste se minimiza o no. Se puede establecer valores mayores que 0 y menores que 1. Los valores más grandes evitarán el sobreajuste, pero valores más pequeños suelen funcionar mejor (normalmente 0,1 o menos).

4.3.4. Optimización de Hiper-parámetros

Se puede obtener la mejor combinación de los valores de los parámetros, es decir, aquellos que optimizaran el resultado predictivo del modelo, probando diferentes combinaciones y buscando el mejor rendimiento en una prueba de validación cruzada.

En concreto, para encontrar la mejor combinación de valores de un modelo determinado se necesita buscar con las siguientes condiciones:

- un modelo (Árbol de Decisión, Random Forests, Boosted Trees),
- un conjunto de combinaciones de parámetros en la que buscar;
- un método de búsqueda o muestreo de candidatos
- un esquema de validación cruzada;
- una función de puntuación.

Existen **dos enfoques** genéricos para la búsqueda de la mejor combinación de hiperparámetros para un modelo. Estos son los siguientes:

- **Búsqueda exhaustiva** (GridSearch): Genera exhaustivamente candidatos a partir de una cuadrícula de conjuntos de valores especificados para cada hiper-parámetro del modelo. Posteriormente evalúa evalúan todas las combinaciones generadas y selecciona la mejor combinación.
- **Búsqueda aleatoria** (Random Search): Puede muestrear un número determinado de candidatos de un espacio de parámetros con una distribución específica, esto es, cada prueba que se realiza se ajusta a una posible combinación de hiper-parámetros de entre los especificados en la definición inicial. Esto permite una ejecución una exploración más rápida de espacios amplios de parámetros. Para ejecutar esta búsqueda es necesario especificar el número de combinaciones candidatas a muestrear. Para cada parámetro, se puede especificar una distribución sobre posibles valores o una lista de opciones

Otro factor importante para la búsqueda de los mejores hiper-parámetros es la **función de evaluación del modelo**. Normalmente se utiliza la exactitud (o *accuracy*) para clasificación y el coeficiente R^2 para regresión. Pero esta selección por defecto debe ser estudiada para que sea adecuada al problema. Por ejemplo, ante un problema de clasificación no balanceada, la evaluación de los modelos mediante la precisión o la exactitud no suele ser informativo. De esta manera será necesario especificar la función de valoración que mejor se adecue al problema.

Hay que tener en cuenta que **es habitual que un pequeño subconjunto de esos hiperparámetros pueda tener un gran impacto en el rendimiento** predictivo o de cálculo del modelo, mientras que otros pueden dejarse con sus valores por defecto. Es por ello que es necesario conocer bien cuáles son los parámetros relevantes para un modelo concreto y de esta forma comprender más profundamente el comportamiento esperado.

Bibliografía

- [BFOS17] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.
- [Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [D⁺02] Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2(1):110–125, 2002.
- [FSA99] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.
- [Ho98] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.
- [HRZZ09] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class ada-boost. *Statistics and its Interface*, 2(3):349–360, 2009.
- [Lou14] Gilles Louppe. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*, 2014.
- [Mor94] Antonio Moreno. *Aprendizaje automático*. Edicions UPC, 1994.
- [Qui96a] J Ross Quinlan. Improved use of continuous attributes in c4. 5. *Journal of artificial intelligence research*, 4:77–90, 1996.
- [Qui96b] J. Ross Quinlan. Learning decision tree classifiers. *ACM Computing Surveys (CSUR)*, 28(1):71–72, 1996.