Lectura 6: Árboles de decisión y métricas avanzadas (precisión, recall, F1)

SI3015 - Fundamentos de Aprendizaje Automático

2026

1. Árboles de Decisión

Los **Árboles de Decisión** son algoritmos de aprendizaje supervisado no paramétricos utilizados tanto para problemas de **clasificación** (predicción de etiquetas categóricas) como de **regresión** (predicción de valores numéricos continuos). Su funcionamiento se basa en la partición binaria recursiva del espacio de características.

1.1. Estructura y Componentes

Un árbol se organiza como un grafo acíclico dirigido que representa un proceso de decisión jerárquico:

- **Nodo Raíz (Root Node):** El punto de inicio que contiene la totalidad de la muestra y realiza la primera partición basada en el atributo más discriminante.
- **Nodos de Decisión (Internal Nodes):** Representan pruebas lógicas sobre una característica específica (ej. $x_i > t$).
- **Ramas (Edges):** Representan el resultado de la condición evaluada, dirigiendo el flujo hacia el siguiente nodo.
- **Nodos Hoja (Leaf Nodes):** Nodos terminales que contienen la predicción final (la moda de la clase o la media del valor objetivo).

1.1.1. Ejemplo básico

En el contexto de Machine Learning (ML), un árbol de decisión no solo sigue reglas lógicas manuales, sino que las **descubre** analizando un conjunto de datos histórico mediante procesos estadísticos.

Caso de Uso: Evaluación de Riesgo Crediticio
Un banco utiliza este modelo para decidir automáticamente si aprueba o rechaza una solicitud de préstamo basándose en el perfil del cliente. El algoritmo entrena sobre una base de datos histórica (ej. $n = 1,000$ clientes) para identificar patrones de pago.

Proceso de Decisión del Modelo

1. **Selección de la Variable Principal (Nodo Raíz):** El algoritmo detecta que el **Historial Crediticio** es el factor con mayor ganancia de información.
   - *¿Tiene deudas impagadas?*
     - **SÍ:** El 90 % de los casos resultan en incumplimiento → **Nodo Hoja: RECHAZAR**.
     - **NO:** Se procede al siguiente nivel de evaluación.
2. **Segmentación por Capacidad Económica (Nodo de Decisión):** Para los clientes sin deudas, el factor determinante es el **Ingreso Mensual**.
   - *¿Ingreso ¡$2,000 USD?*
     - **SÍ:** El riesgo relativo es elevado → **Nodo Hoja: RECHAZAR**.
     - **NO:** Se evalúa la estabilidad laboral.
3. **Refinamiento (Nodo de Decisión):**
   - *¿Antigüedad Laboral ¿2 años?*
     - **SÍ:** Perfil estable → **Nodo Hoja: APROBAR**.
     - **NO:** Perfil incierto → **Nodo Hoja: RECHAZAR** (o requerir aval).

Operatividad Interna en Machine Learning

Cuando se implementa este modelo (por ejemplo, mediante la librería `scikit-learn` en Python), el proceso técnico se resume en tres etapas:

- **Entrenamiento (`.fit`):** El algoritmo evalúa todas las variables predictoras e identifica el umbral t que minimiza la **Impureza de Gini** (la veremos mas adelante).
- **Generación de Reglas:** El modelo traduce los datos en una función escalonada. Si $x_{ingreso} > 2000$, la probabilidad de pertenecer a la clase "Aprobado" aumenta significativamente.
- **Inferencia (`.predict`):** Ante un nuevo vector de entrada $\mathbf{x}$, el modelo recorre el grafo acíclico para asignar una etiqueta en tiempo real.

Representación Geométrica

Matemáticamente, el árbol realiza una partición del espacio de características en hiperrectángulos ortogonales a los ejes. Si representamos los datos en un plano bidimensional (Eje X: Ingresos, Eje Y: Edad), el modelo traza fronteras de decisión rectilíneas que separan las clases.

En resumen:

Uno no le dice al modelo: *Si gana más de 2000, acéptalo.*
Uno le dice: *Aquí tienes 10,000 ejemplos de gente que ganó diferentes montos y si pagaron o no. Encuentra tú mismo el límite de ingresos que separa a los cumplidores de los morosos.*

El resultado de ese entrenamiento es, precisamente, el diagrama de flujo (las reglas) que vimos antes.

1.2. Fundamentos Matemáticos (Métricas de Selección)

El objetivo del algoritmo en cada división es maximizar la homogeneidad (pureza) de los nodos hijos. Las métricas más comunes son:

1.2.1. Impureza de Gini (Algoritmo CART)

La Impureza de Gini es una métrica fundamental en el algoritmo CART (Classification and Regression Trees) que sirve para medir qué tan "mezclados.^están los datos en un nodo de un árbol de decisión. En otras palabras, nos dice qué tan probable es que clasifiquemos mal un elemento si lo eligiéramos al azar y le asignáramos una etiqueta basándonos en la distribución de clases de ese nodo.

Para entenderlo, imagina que tienes una bolsa con pelotas de colores. Un nodo es Nodo Puro (Gini = 0) si todas las pelotas son del mismo color (por ejemplo, todas verdes). No hay incertidumbre; si sacas una, sabes con certeza qué es. Por otro lado, un nodo es un Nodo Impuro (Máximo Gini) si tienes la mitad de pelotas verdes y la mitad de rojas. Es el punto de máxima confusión o desorden.

La probabilidad de clasificar incorrectamente un elemento elegido al azar se define como:

$$G = 1 - \sum_{i=1}^{n} p_i^2$$

Donde n es el número de clases y p_i es la proporción de instancias de la clase i en el nodo.

EJEMPLO:

Tenemos un grupo de **10 personas** para analizar si comprarán o no un producto:

- 6 personas compraron (Sí).
- 4 personas no compraron (No).

Paso 1: Impureza de Gini del Nodo Raíz

Calculamos el desorden inicial antes de realizar cualquier división. Las probabilidades son $P(\text{Sí}) = 0{,}6$ y $P(\text{No}) = 0{,}4$.

$$G_{raiz} = 1 - \sum_{i=1}^{n} (p_i)^2$$

$$G_{raiz} = 1 - (0{,}6^2 + 0{,}4^2)$$

$$G_{raiz} = 1 - (0{,}36 + 0{,}16) = \mathbf{0{,}48}$$

Paso 2: División de los Datos

Evaluamos la variable **"¿Es Socio?"**, resultando en dos subgrupos:

1. **Grupo A (Socios):** 4 personas (3 Sí, 1 No).
2. **Grupo B (No Socios):** 6 personas (3 Sí, 3 No).

Paso 3: Impureza de cada Rama

Impureza Grupo A (Socios)

$$G_A = 1 - \left[\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2\right]$$

$$G_A = 1 - (0{,}5625 + 0{,}0625) = \mathbf{0{,}375}$$

Impureza Grupo B (No Socios)

$$G_B = 1 - \left[\left(\frac{3}{6}\right)^2 + \left(\frac{3}{6}\right)^2\right]$$

$$G_B = 1 - (0{,}25 + 0{,}25) = \mathbf{0{,}50}$$

Paso 4: Gini Ponderado de la División

Calculamos el promedio de impureza ajustado por el tamaño de cada grupo ($N = 10$):

$$G_{total} = \left(\frac{n_A}{N} \times G_A\right) + \left(\frac{n_B}{N} \times G_B\right)$$

$$G_{total} = \left(\frac{4}{10} \times 0{,}375\right) + \left(\frac{6}{10} \times 0{,}50\right)$$

$$G_{total} = 0{,}15 + 0{,}30 = \mathbf{0{,}45}$$

Conclusión

Dado que el **Gini final (0.45)** es menor que el **Gini inicial (0.48)**, la división es considerada útil por el algoritmo CART para reducir la incertidumbre en el modelo.

1.2.2. El algoritmo CART (Classification and Regression Trees)

El algoritmo **CART** es la piedra angular de los árboles de decisión modernos. Su funcionamiento se basa en una estrategia denominada **partición binaria recursiva**. A diferencia de otros algoritmos como ID3 o C4.5, CART siempre divide un nodo en exactamente **dos hijos** (Izquierda/Derecha).

El algoritmo sigue una lógica *greedy* (codiciosa), tomando la mejor decisión inmediata en cada paso:

1. **Evaluación de características:** El algoritmo analiza todas las variables disponibles ($X_1, X_2, \ldots, X_n$) y todos los posibles puntos de corte (umbrales) para cada una.

2. **Cálculo de Impureza:** Para cada posible división, se calcula la reducción del desorden:
    - En **Clasificación**, utiliza el **Índice de Gini**.
    - En **Regresión**, utiliza el **Error Cuadrático Medio (MSE)**.
3. **Selección de la mejor división:** Elige la combinación de variable y umbral que maximiza la "pureza"de los nodos resultantes.
4. **Recursión:** El proceso se repite para cada uno de los nodos hijos creados, convirtiéndolos en nuevos "nodos raíz"de sus propios subárboles.
5. **Criterio de parada:** El algoritmo se detiene cuando se cumple una condición (ej. el nodo es puro, se alcanzó la profundidad máxima o hay muy pocos datos para continuar).

La regla para decidir el corte varía según la naturaleza de la variable objetivo:

Característica	**CART de Clasificación**	**CART de Regresión**
Variable Objetivo	Categórica (Sí/No, Clase)	Continua (Precio, Valor)
Criterio de Selección	Índice de Gini	Reducción de Varianza / MSE
Predicción Final	Clase más frecuente (Moda)	Promedio de los valores (Media)

Cuadro 1: Comparativa entre CART de Clasificación y Regresión.

Para evitar el **overfitting** (sobreajuste), CART utiliza una técnica llamada **Cost-Complexity Pruning**. El algoritmo busca un equilibrio entre la complejidad del árbol y su error, eliminando ramas que no aportan una mejora significativa en la precisión general para garantizar que el modelo pueda generalizar con datos nuevos.

Fortalezas del Algoritmo

- **No requiere escalado:** Las variables no necesitan normalización.
- **Maneja Outliers:** Los valores extremos tienen un impacto limitado en las divisiones.
- **Interpretabilidad:** La lógica de decisión es transparente y fácil de visualizar.

1.3. Métodos de Ensamble (Ensemble Learning)

Para superar las limitaciones de un solo clasificador (árbol), se utilizan arquitecturas combinadas:

1.3.1. Bagging

El **Bagging**, abreviatura de **Bootstrap Aggregating**, es una técnica de Ensemble Learning (aprendizaje en conjunto) diseñada para mejorar la precisión y la estabilidad de los algoritmos de aprendizaje automático, especialmente de los árboles de decisión.

Su objetivo principal es reducir la varianza de un modelo, evitando que se sobreajuste (overfitting) a los datos de entrenamiento.

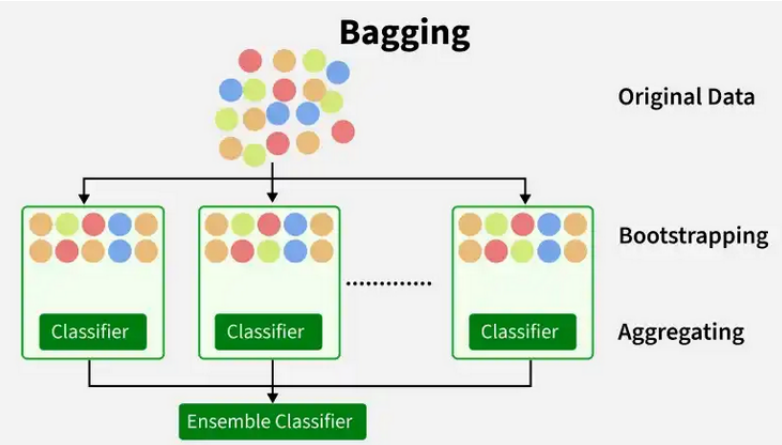

Figura 1: Bagging

Funcionamiento general

A.Bootstrap (Muestreo aleatorio)
En lugar de entrenar un solo modelo con todos los datos, el Bagging crea múltiples subconjuntos de datos aleatorios. Se toma el dataset original y se generan n muestras.

Cada muestra se crea mediante muestreo con reemplazo, lo cual significa que un mismo dato puede aparecer varias veces en una submuestra y otros pueden no aparecer (datos Out-of-Bag).

B. Entrenamiento en paralelo
Se entrena un modelo independiente (generalmente un árbol de decisión CART) para cada una de las muestras generadas. Como cada modelo ve un conjunto de datos ligeramente distinto, cada uno aprende patrones diferentes.

C. Aggregation (Agregación)
Para dar una respuesta final, se combinan las predicciones de todos los modelos:

- En Clasificación: Se utiliza la votación mayoritaria (la clase que más modelos predijeron).
- En Regresión: Se calcula el promedio de todas las predicciones.

En Machine Learning, los árboles de decisión individuales son muy sensibles al ruido de los datos. El Bagging "suaviza" esos errores al promediar muchos árboles. En este sentido, el algoritmo **Random Forest** es, esencialmente, Bagging llevado al siguiente nivel, es decir:

- Utiliza Bagging para crear las muestras de datos.
- Añade una mejora: En cada división de los árboles, solo permite elegir entre un subconjunto aleatorio de variables (columnas). Esto hace que los árboles sean aún más diferentes entre sí, mejorando la robustez del conjunto.

1.3.2. Boosting

Si el Bagging se basa en la independencia (entrenar muchos modelos a la vez y promediarlos), el **Boosting** se basa en la **interdependencia**. Es una estrategia de aprendizaje secuencial donde cada nuevo modelo intenta corregir los errores cometidos por los modelos anteriores.

En lugar de una democracia donde todos los votos valen igual, el Boosting es como un entrenamiento intensivo: el algoritmo se enfoca cada vez más en los datos que son difíciles de clasificar.

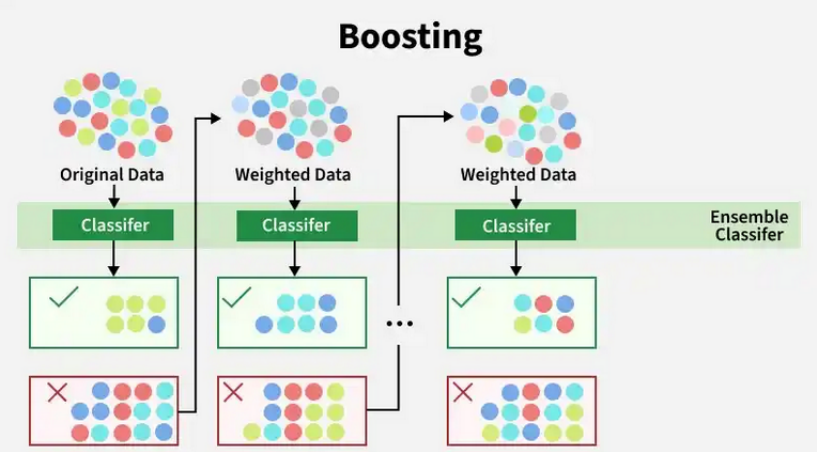

Figura 2: Boosting

Funcionamiento general

A diferencia del Bagging, aquí los modelos no se entrenan en paralelo, sino uno tras otro:

- **Modelo Inicial:** Se entrena un modelo sencillo (normalmente un árbol muy corto, llamado stub) con todos los datos. Sus predicciones serán mediocres.
- **Identificación del Error:** El algoritmo identifica qué datos se clasificaron mal o tuvieron mucho error.

- **Ajuste de Pesos:** Se les da más ïmportancia"(peso) a esos datos erróneos.
- **Siguiente Modelo:** El nuevo modelo se entrena priorizando los datos que el anterior falló.
- **Combinación Final:** El resultado es una suma ponderada de todos los modelos. Los modelos que fueron más precisos tienen más peso en la decisión final.

Las implementaciones más famosas de Boosting son:

- **AdaBoost (Adaptive Boosting):** El pionero. Ajusta los pesos de las instancias (filas) que fueron mal clasificadas.
- **Gradient Boosting (GBM):** En lugar de ajustar pesos, cada nuevo modelo intenta predecir los residuos (el error remanente) del modelo anterior usando el descenso de gradiente.
- **XGBoost / LightGBM:** Versiones ultra-optimizadas del Gradient Boosting que son las reinas absolutas en competencias de Ciencia de Datos (como Kaggle) por su velocidad y precisión.

Comparativa entre Bagging y Boosting

Cuadro 2: Comparativa Técnica: Random Forest (Bagging) vs. XGBoost (Boosting)

Característica	**Random Forest (Bagging)**	**XGBoost (Boosting)**
Filosofía	Independencia: Promedia múltiples árboles para reducir varianza.	Dependencia: Construcción secuencial para reducir el sesgo.
Facilidad de Uso	**Muy Alta.** Pocos parámetros críticos; difícil de arruinar.	**Media/Baja.** Requiere ajuste fino de hiperparámetros.
Overfitting	Muy robusto. Es casi imposible que sobreajuste por añadir más árboles.	Riesgo elevado. Necesita regularización y control de iteraciones.
Entrenamiento	**Paralelo.** Cada árbol se entrena de forma independiente.	**Secuencial.** Aunque XGBoost usa paralelización interna de nodos.
Sensibilidad	Ignora bastante bien el ruido y los valores atípicos.	Muy sensible al ruido (intenta ajustar los errores a toda costa).
Precisión	Excelente como modelo base estable.	**Estado del arte.** Suele dominar en tablas de clasificación.

2. Métricas avanzadas

Para ilustrar las métricas, utilizaremos un modelo médico diseñado para detectar una enfermedad rara en un grupo de **100 personas**:

- **Enfermos reales (Positivos):** 10 personas.
- **Sanos reales (Negativos):** 90 personas.

2.1. Matriz de Confusión

Es la tabla que cruza la realidad con la predicción del modelo, permitiendo identificar cuatro tipos de resultados:

- **Verdadero Positivo (TP):** Enfermo detectado correctamente.
- **Verdadero Negativo (TN):** Sano identificado correctamente.
- **Falso Positivo (FP):** Sano clasificado como enfermo (*Error Tipo I*).
- **Falso Negativo (FN):** Enfermo clasificado como sano (*Error Tipo II*).

2.2. Accuracy (Exactitud)

Mide el porcentaje total de aciertos sobre el total de casos.

- **Definición:** ¿Qué proporción de predicciones fueron correctas?
- **Fórmula:** $\text{Acc} = \frac{TP+TN}{TP+TN+FP+FN}$
- **Ejemplo:** Si el modelo predice que todos están sanos, tendría un **90 % de accuracy** (acierta los 90 sanos), pero no detectaría ningún enfermo.

2.3. Precision (Precisión)

Se enfoca en la calidad de las predicciones positivas.

- **Definición:** De todos los que el modelo marcó como "Enfermos", ¿cuántos lo estaban realmente?
- **Fórmula:** $\text{Prec} = \frac{TP}{TP+FP}$
- **Ejemplo:** Si el modelo marca a 10 personas como enfermos pero solo 2 lo están, la precisión es del **20 %**. El modelo genera muchos falsos positivos.

2.4. Recall (Sensibilidad / Exhaustividad)

Se enfoca en la capacidad del modelo para capturar los casos reales.

- **Definición:** De todos los enfermos reales, ¿cuántos logró encontrar el modelo?
- **Fórmula:** $\text{Recall} = \frac{TP}{TP+FN}$
- **Ejemplo:** Si hay 10 enfermos y el modelo solo detecta a 2, el recall es del **20 %**. El modelo es "distraído" y omite casos críticos.

2.5. F1-Score

Es la media armónica entre Precision y Recall, proporcionando un balance robusto.

- **Definición:** Un balance que penaliza valores extremos en Precision o Recall. Ideal para datasets desbalanceados.
- **Fórmula:** $F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
- **Ejemplo:** Con una Precision de 0.20 y un Recall de 0.20, el F1 es **0.20**, reflejando un desempeño deficiente a pesar de un accuracy alto.

Tabla Resumen de Fórmulas

Métrica	**Fórmula**	**Objetivo Principal**
Accuracy	$\frac{TP+TN}{\text{Total}}$	Desempeño global.
Precision	$\frac{TP}{TP+FP}$	Minimizar Falsos Positivos.
Recall	$\frac{TP}{TP+FN}$	Minimizar Falsos Negativos.
F1-Score	$2 \cdot \frac{\text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}}$	Balancear calidad y cobertura.