Clase 080 — Árboles de decisión: entrenamiento, visualización, CART

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 6 § Training and Visualizing a Decision Tree. · Duración estimada: 60 min.

Clase 080 — Árboles de decisión: entrenamiento, visualización, CART

> *Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 6 § Training and Visualizing a Decision Tree. · Duración estimada: 60 min.*

Objetivo

Que el alumno entrene un DecisionTreeClassifier con el algoritmo CART, entienda cómo se elige cada split (criterio Gini/Entropy), y sepa leer el árbol — tanto el dibujo (plot_tree) como el código (export_graphviz) — para auditar las decisiones del modelo.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Entrenar un DecisionTreeClassifier de scikit-learn sobre un dataset tabular (Iris) y predecir clases / probabilidades.
2. Explicar el algoritmo CART: greedy, binario, busca el par (feature, threshold) que minimiza la impureza ponderada de los dos hijos.
3. Calcular Gini y entropía a mano sobre un nodo con k clases y elegir el split óptimo entre candidatos.
4. Visualizar el árbol entrenado con plot_tree (matplotlib) y export_graphviz (DOT → PNG/SVG), interpretando samples, value, class, gini.
5. Identificar la decision boundary axis-aligned: cada split es ortogonal a un eje, lo que limita al árbol frente a fronteras oblicuas.

Temas

#	Tema	Por qué importa
1	DecisionTreeClassifier API	El estimador base del capítulo y de Random
2	Algoritmo CART	Saber qué hace el .fit() por debajo evita
3	Criterio Gini vs Entropy	Casi siempre dan el mismo árbol; saber cuá
4	Visualización: plot_tree y Graphviz	Auditar el modelo y comunicarlo a no-técni
5	Interpretación de nodos	Leer value=[n0, n1, n2], samples, class pr
6	Decision boundary axis-aligned	Limitación geométrica que explica por qué

Definiciones y características

CART (Classification And Regression Trees)

: Algoritmo que entrena árboles binarios (cada split tiene exactamente 2 hijos). Greedy: en cada nodo elige el par (feature k, threshold t_k) que minimiza el costo J = (m_left/m) · G_left + (m_right/m) · G_right. No vuelve atrás (no es óptimo global).

Impureza Gini

: $G_i = 1 - \Sigma p_{i,k}^2$ sobre las k clases del nodo i. Vale 0 si el nodo es puro (una sola clase) y máximo cuando las clases están balanceadas. Es el default en sklearn.

Entropía (information gain)

: $H_i = -\Sigma p_{i,k} \cdot \log(p_{i,k})$. Mide desorden en bits. Se activa con criterion='entropy'. En la práctica produce árboles muy similares a Gini; Gini es marginalmente más rápido.

Nodo

: Punto del árbol donde se evalúa una condición feature ≤ threshold. Si se cumple, va al hijo izquierdo; si no, al derecho.

Hoja (leaf)

: Nodo terminal, sin hijos. Predice la clase mayoritaria de las muestras de entrenamiento que cayeron en él. La probabilidad estimada es la proporción de cada clase en la hoja.

Profundidad (max_depth)

: Distancia desde la raíz hasta la hoja más lejana. Controla complejidad: sin tope, CART crece hasta que cada hoja es pura → overfitting casi garantizado. Se ataca en la Clase 072.

criterion

: Hiperparámetro de sklearn: 'gini' (default) o 'entropy'. Define la función de impureza que minimiza CART al elegir splits.

Decision boundary axis-aligned

: Cada split parte el espacio con un hiperplano ortogonal a un eje ($x_k = t_k$). La frontera resultante es una unión de rectángulos. Por eso un árbol no puede aprender una diagonal de forma compacta — necesita escaleras.

Dataset / recursos

- Iris (sklearn.datasets.load_iris): 150 muestras, 4 features, 3 clases. Géron lo usa porque permite dibujar el árbol completo en una página y la frontera en 2D (pétalo largo × ancho).
- Opcional: moons (make_moons) para ver la limitación axis-aligned.

Ejercicios

1. Fit + score baseline. Entrená DecisionTreeClassifier(max_depth=2, random_state=42) sobre Iris (todas las features). Reportá accuracy en train. Probabilidades de la primera flor con .predict_proba.

2. Gini a mano. Para el nodo raíz de Iris (50/50/50), calculá Gini. Verificá contra tree_.impurity[0] del estimador entrenado.

3. Gini vs Entropy. Entrená dos árboles max_depth=3, uno con cada criterio. Compará accuracy y el set de features usadas en los splits (tree_.feature). ¿Cambia algo material?

4. plot_tree. Renderizá el árbol del ejercicio 1 con sklearn.tree.plot_tree(clf, feature_names=..., class_names=..., filled=True). Identificá: feature del root split, threshold, y la clase predicha por cada hoja.

5. Boundary axis-aligned. Entrená un árbol max_depth=4 sobre make_moons(n_samples=300, noise=0.2). Graficá la decision boundary con un meshgrid. Observá los rectángulos.

Homework verificable

Notebook que: (a) carga Iris (solo pétalo largo y ancho), (b) entrena DecisionTreeClassifier(max_depth=2), (c) reporta accuracy y predict_proba para una flor nueva [5, 1.5], (d) exporta el árbol con export_graphviz a un archivo iris_tree.dot y lo convierte a PNG (o usa plot_tree), (e) responde por escrito: "¿por qué la hoja izquierda tiene Gini=0?".

Criterio de aceptación: accuracy ≥ 0.95 en train, el árbol exportado tiene exactamente 3 hojas (porque max_depth=2 con el split que CART elige sobre pétalos), y la respuesta menciona que esa hoja contiene solo setosa (pétalos chicos → linealmente separable).

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
accuracy = 1.0 en train y mucho menos en t	No pusiste max_depth ni regularización. CA
graphviz: command not found al correr expo	El paquete Python graphviz no trae el bina
Cambio criterion='gini' ↔ 'entropy' y el á	Casi nunca pasa en datasets reales — si pa
Resultados distintos en cada .fit() con el	CART desempata splits con un componente al
plot_tree sale ilegible (texto pisado, min	Árbol grande sin figsize. Fix: plt.figure(

Preguntas frecuentes

¿Hay que escalar las features antes de un árbol?

No. CART compara feature ≤ threshold por feature individualmente — escalar/centrar no cambia el orden, así que el árbol es invariante a transformaciones monótonas de cada feature. Esto lo diferencia de SVM, KNN y regresión regularizada.

¿Gini o Entropy, cuál uso?

Gini (default). Es marginalmente más rápido (no calcula log) y produce árboles casi idénticos. Géron lo dice explícito: la elección rara vez importa en la práctica.

¿Por qué los árboles de sklearn son siempre binarios? ID3 hacía multi-way.

Porque sklearn implementa CART, no ID3/C4.5. CART es binario por diseño: un split por nodo, dos hijos. Es más simple, se generaliza a regresión sin cambios, y los multi-way se simulan encadenando binarios.

¿Los árboles manejan features categóricas?

sklearn no nativamente (hasta versión 1.4 hay soporte experimental). Hay que codificar: ordinal si hay orden natural, one-hot si no. Cuidado con one-hot de alta cardinalidad: el árbol nunca puede agrupar dos categorías en el mismo split, lo que sesga la selección hacia features numéricas.

¿Qué predice una hoja cuando empatan dos clases?

Devuelve la clase con índice menor entre las empatadas (orden de clf.classes_). En predict_proba te muestra el empate real ([0.5, 0.5, 0]), así que si te importa, usá probabilidades en vez de la clase argmax.

Referencias

- Géron, cap. 6 — Decision Trees, secciones Training and Visualizing y Making Predictions.
- sklearn — Decision Trees user guide
- DecisionTreeClassifier API
- Graphviz — para export_graphviz + dot.

Siguiente clase

Clase 081 — Regularización de árboles

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb