Ficha Técnica Avanzada: Árboles de Decisión

Jordi Pozo
I.E.S. Ribera de Castilla

Curso 2025/26

1. Concepto

Un **Árbol de Decisión** es un modelo de **aprendizaje supervisado** (utilizado tanto para clasificación como para regresión) que toma decisiones basándose en una estructura de grafo similar a un árbol.

El modelo aprende reglas de decisión simples (p. ej. "¿Es el ingreso $> 50.000\,€$?") inferidas a partir de las características de los datos. La estructura se compone de:

- **Nodo Raíz (Root Node):** El nodo inicial que representa a todo el conjunto de datos.
- **Nodos Internos (Internal Nodes):** Nodos donde se evalúa una característica para dividir los datos.
- **Ramas (Branches):** Representan el resultado de una decisión (p. ej. "Sí" o "No").
- **Nodos Hoja (Leaf Nodes):** Nodos terminales que representan el resultado final.

2. ¿Cómo aprende el árbol? Métricas de División

El algoritmo (como CART o C4.5) construye el árbol de forma recursiva. En cada nodo, busca la "mejor" característica y el "mejor" umbral para dividir los datos: "mejor" significa producir nodos hijos lo más "puros" posible.

2.1. Impureza de Gini (Gini Impurity)

Es la métrica por defecto en `scikit-learn` para clasificación:

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

- **Gini = 0 (Puro):** $1 - (1^2) = 0$.
- **Gini = 0.5 (2 clases, 50/50):** $1 - (0{,}5^2 + 0{,}5^2) = 0{,}5$.

2.2. Entropía y Ganancia de Información

$$H = -\sum_{i=1}^{C} p_i \log_2 p_i$$

La **ganancia de información** (IG) es la reducción de entropía tras dividir:

$$IG(\text{Padre, Split}) = H(\text{Padre}) - \sum_{\text{hijo}} \frac{N_{\text{hijo}}}{N_{\text{Padre}}} H(\text{hijo})$$

En `scikit-learn`: `criterion='entropy'`.

3. Ilustración del Modelo

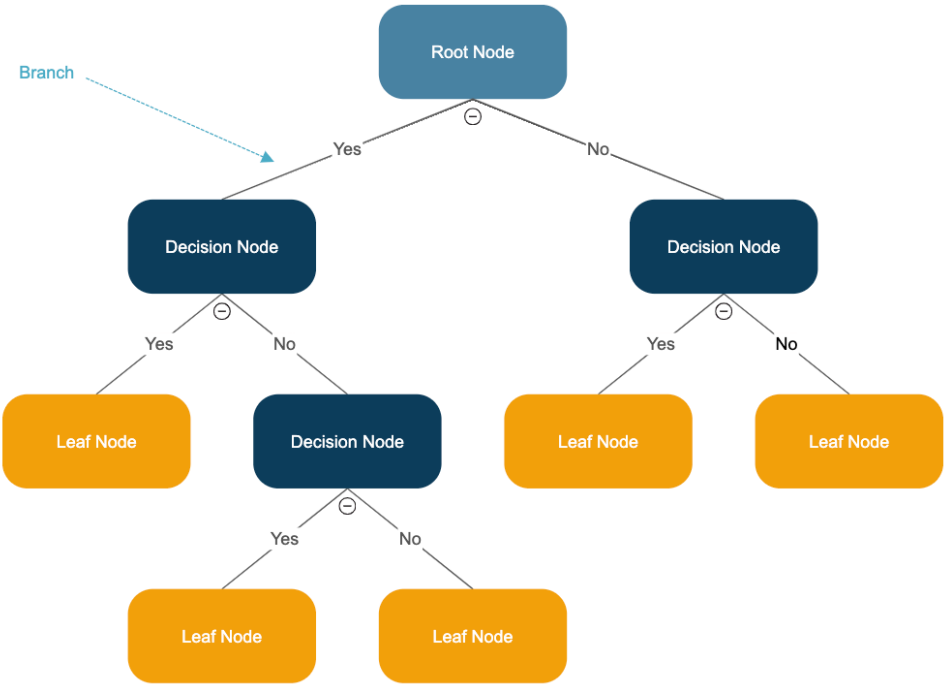

Figura 1: Representación gráfica de un árbol de decisión (placeholder).

4. Hiperparámetros Clave

4.1. Parámetros genéricos

- **criterion** (Gini/Entropía en clasificación; ECM en regresión).
- **max_depth**: tope de niveles del árbol.
- **min_samples_split**: mínimo para intentar dividir.
- **min_samples_leaf**: mínimo por hoja tras dividir.
- **ccp_alpha**: poda por complejidad.

4.2. Implementación en `scikit-learn`

- `criterion`: `'gini'`/`'entropy'` (`'squared_error'` en regresión).
- `max_depth`: `None` por defecto; conviene fijarlo (p. ej. 3, 5, 8).
- `min_samples_split`: por defecto 2.
- `min_samples_leaf`: por defecto 1 (sube para suavizar).
- `ccp_alpha`: 0.0 por defecto (sin poda).

5. Características

5.1. Ventajas

- Interpretables (“white box”).

- Poco preprocesado.
- Selección implícita de variables.

5.2. Desventajas

- Sobreajuste si no se limita.
- Inestables a pequeñas variaciones.
- Sesgo hacia variables con muchos niveles.

6. Usos

- **Clasificación**: diagnóstico, riesgo crediticio, spam.
- **Regresión**: precios, demanda.
- **Base de ensembles**: **Random Forest**, **Gradient Boosting (XGBoost)**.

7. Ejemplos con Python y Scikit-learn

7.1. Ejemplo 1: Clasificación (Iris)

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt

# 1. Cargar datos
iris = load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = iris.target

# 2. Dividir
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
  random_state=42)

# 3. Crear y entrenar (Ajustando max_depth)
clf = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=42)
clf.fit(X_train, y_train)

# 4. Evaluar
y_pred = clf.predict(X_test)
print(f"Accuracy (Precision): {accuracy_score(y_test, y_pred):.2f}")
print("Matriz de Confusion:\n", confusion_matrix(y_test, y_pred))

# 5. Visualizar el arbol
plt.figure(figsize=(15, 10))
plot_tree(clf,
filled=True,
rounded=True,
class_names=iris.target_names,
feature_names=iris.feature_names)
plt.title("Arbol de Decision - Clasificacion Iris (max_depth=3)")

# Guardar la imagen para usarla en LaTeX:
```

```
# plt.savefig("decision_tree_diagram.png")
```

Listing 1: Clasificacion con Iris (criterion='gini', max_depth=3).

7.2. Ejemplo 2: Regresión (Diabetes)

```
from sklearn.datasets import load_diabetes
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

# 1. Cargar datos
diabetes = load_diabetes()
X = pd.DataFrame(diabetes.data, columns=diabetes.feature_names)
y = diabetes.target

# 2. Dividir
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
  random_state=42)

# 3. Crear y entrenar
reg = DecisionTreeRegressor(criterion='squared_error', max_depth=4, random_
  state=42)
reg.fit(X_train, y_train)

# 4. Evaluar
y_pred_reg = reg.predict(X_test)
mse = mean_squared_error(y_test, y_pred_reg)
r2 = r2_score(y_test, y_pred_reg)
print(f"MSE: {mse:.2f}")
print(f"R2: {r2:.2f}")

# 5. Visualizar
plt.figure(figsize=(20, 12))
plot_tree(reg, filled=True, rounded=True, feature_names=diabetes.feature_names
  )
plt.title("Arbol de Decision - Regresion Diabetes (max_depth=4)")
# plt.savefig("diabetes_tree.png")
```

Listing 2: Regresion con Diabetes (criterion='squared_error', max_depth=4).