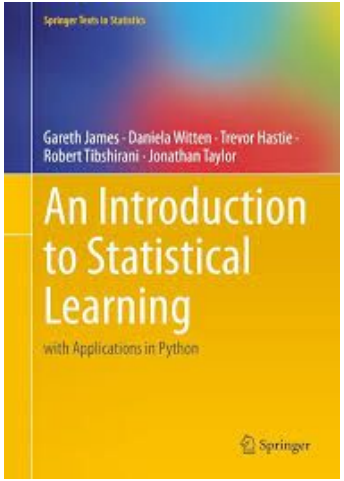
clusterAI
ciencia de datos en ingeniería industrial
UTN BA
curso I5521

clase_07: Árboles de Decisión

Agenda clase: Arboles de Decision

- ¿Qué es un árbol de decisión?
- ¿Cómo se construye un árbol de decisión?
- Criterios de división
- ¿Cómo determinamos el mejor corte ?
- Ventajas y desventajas de los árboles de decisión
- Ensamble de árboles y tipos
- Método de Bagging
- RandomForest
- Método de Boosting
- Boosting y descenso del gradiente
- XGBoost y LightGBM
- Diferencias entre XGBoost y LightGBM

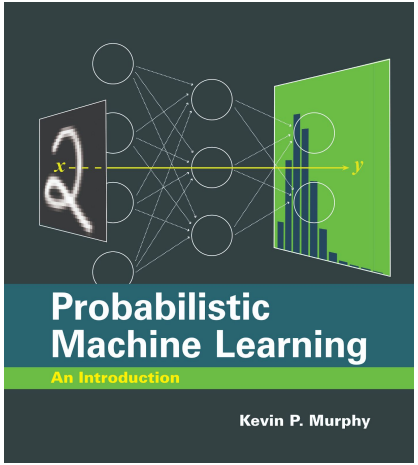
Lectura sugerida



Introduction to Statistical Learning (with applications in Python)
https://www.statlearning.com/
https://hastie.su.domains/ISLP/ISLP_website.pdf.download.html

- Capitulo 8 -> Tree-Based Methods

Lectura sugerida



Probabilistic Machine Learning: An Introduction
https://probml.github.io/pml-book/book1.html

- Capítulo 18: Trees, Forests, Bagging, and Boosting

¿Que es un árbol de decisión?

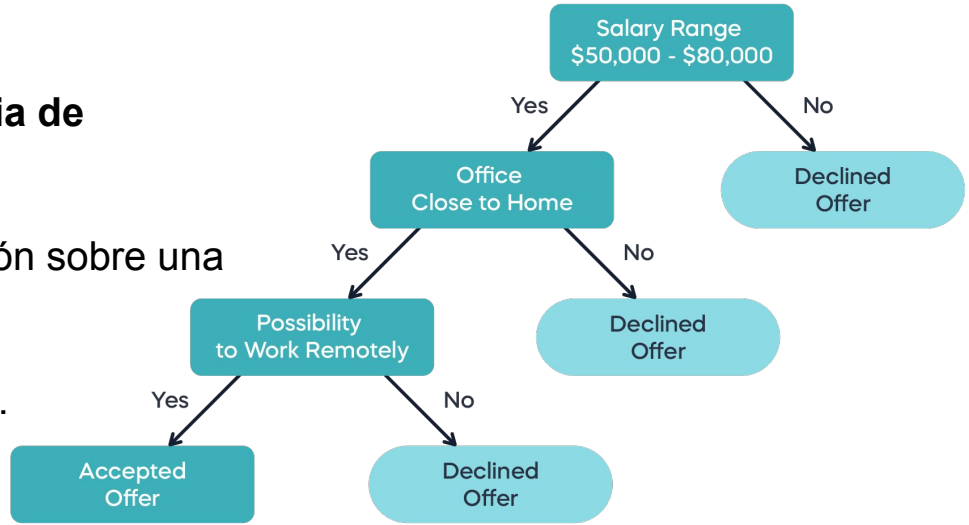
Es un **modelo predictivo supervisado** que utiliza una estructura similar a un árbol para tomar decisiones. Pueden utilizarse tanto en clasificación como en regresión.

Representa decisiones como **una secuencia de preguntas** sobre las variables.

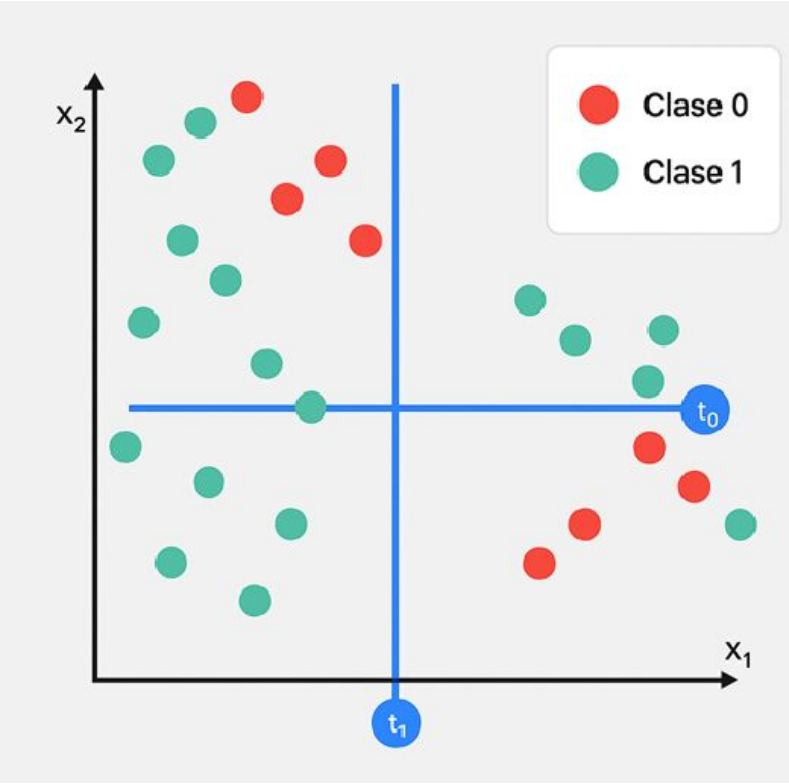
Cada **nodo interno** representa una condición sobre una variable.

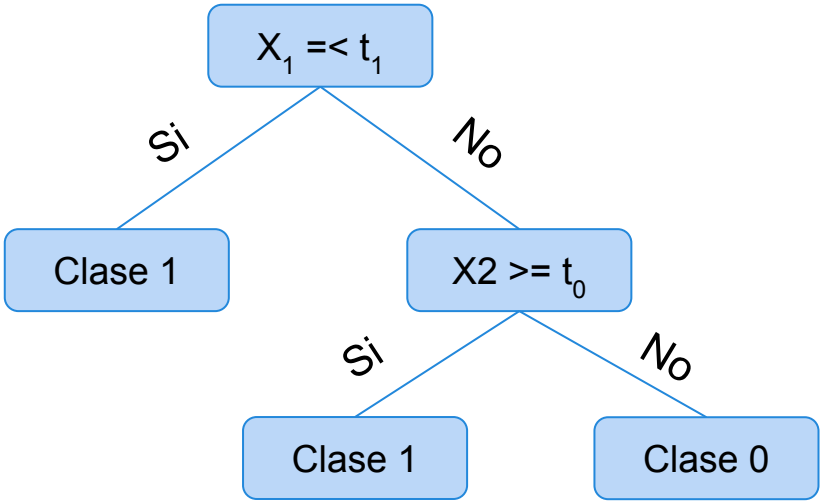
Cada **rama** representa un resultado posible.

Cada **hoja** representa una **predicción** o **valor estimado** (clase o número).



Ejemplo





Este mecanismo se denomina Recursive Binary Splitting

Criterios de división

Se necesita medir que tan bueno o no es un corte. Para eso en clasificación utilizamos el concepto de impureza

La Impureza es una manera de cuantificar que tan mezcladas están dos clases dentro de un determinado corte.

Formas de medir:

- Coeficiente de Gini
- Entropía

Coeficiente de Gini y Entropía

Entropía

$$Entropia(S) = \sum_{c \in Clases(S)} -p_c \cdot \log_2(p_c)$$

donde $p_c = \frac{|\{x \in S, clase(x) = c\}|}{|S|}$

Índice de Gini

$$Gini(S) = \sum_{c \in Clases(S)} p_c(1 - p_c)$$

$$= 1 - \sum_{c \in Clases(S)} p_c^2$$

Donde pc es la proporción de la clase c que hay en S

¿Cómo determinamos el mejor corte?

Buscamos reducir la impureza utilizando la entropía:

$$IG = Entropia\ \ nodo\ padre\ \ - Promedio\ Ponderado(Entropia\ de\ los\ hijos)$$

Esto se denomina **Information Gain**. También se puede utilizar el índice de Gini

Ejemplo: Siendo P nodo padre, F al feature, x al punto de corte en F:

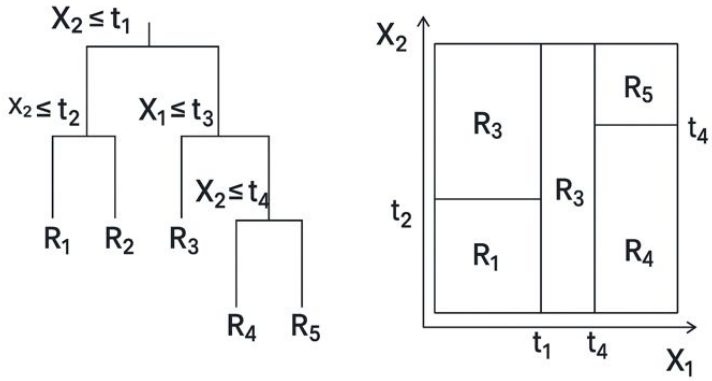
$$IG(F_i, x\ ) = Entropia(P) - \sum_{Pi} \frac{\mathrm{N_i}}{\mathrm{N}} Entropia\ (P_i)$$

N es la cantidad de datos que tiene el nodo padre y Ni será la cantidad que hay en cada nodo hijo

Para Regresión…

- No existe el concepto de Impureza porque no hay clases
- Pero utilizamos el Error Cuadrático Medio (ECM). Lo que hacemos es reducir el ECM de las distintas regiones del árbol.

$$\mathbf{ECM} = \sum_{j=1} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$



- Para evaluar que tan bueno es un corte queremos elegir el que más minimice el error cuadrático

$$ECM = \sum_{i \in R_1} (y_i - \overline{y}_{R_1})^2 + \sum_{i \in R_2} (y_i - \overline{y}_{R_2})^2$$

donde R1 y R2 son las dos regiones creadas por el corte Rj y ŷ es la media de la respuesta en esa región.

Ventajas y Desventajas

Ventajas

- Fáciles de interpretar y visualizar.
- Manejan variables categóricas y numéricas. (va a depender de la implementación de cada librería)
- Aceptan datos faltantes (también va a depender de la implementación)
- Se entrenan rapidamente
- Útiles para explicar a audiencias no técnicas.

Desventajas

- Alta propensión al overfitting (si no se podan o regulan).
- De manera individual pueden no ser los mejores predictores

¿Que pasa si en lugar de un árbol de decisión utilizamos muchos?

(Cientos)

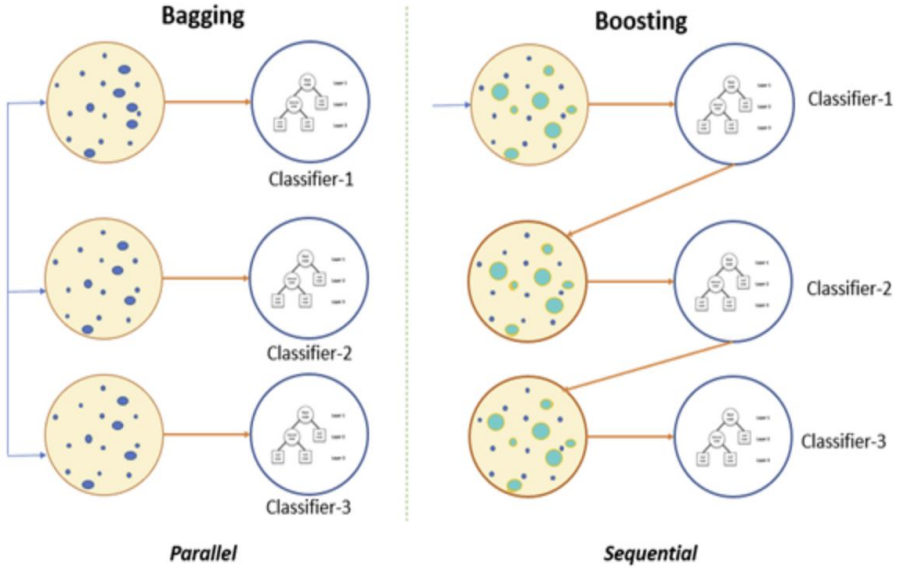
Ensambles de Árboles

Un ensamble de árboles combina muchos árboles de decisión para mejorar la precisión y obtener mejores resultados.

Existen 2 familias de ensambles de árboles:

1) Método de Bagging o promedios: Técnica que entrena muchos árboles en paralelo usando distintas muestras de los datos (con reemplazo) y luego promedia (regresión) o vota (clasificación) los resultados.

2) Método de Boosting: Técnica que entrena árboles de forma secuencial, donde cada nuevo árbol corrige los errores del anterior.



Método de Bagging

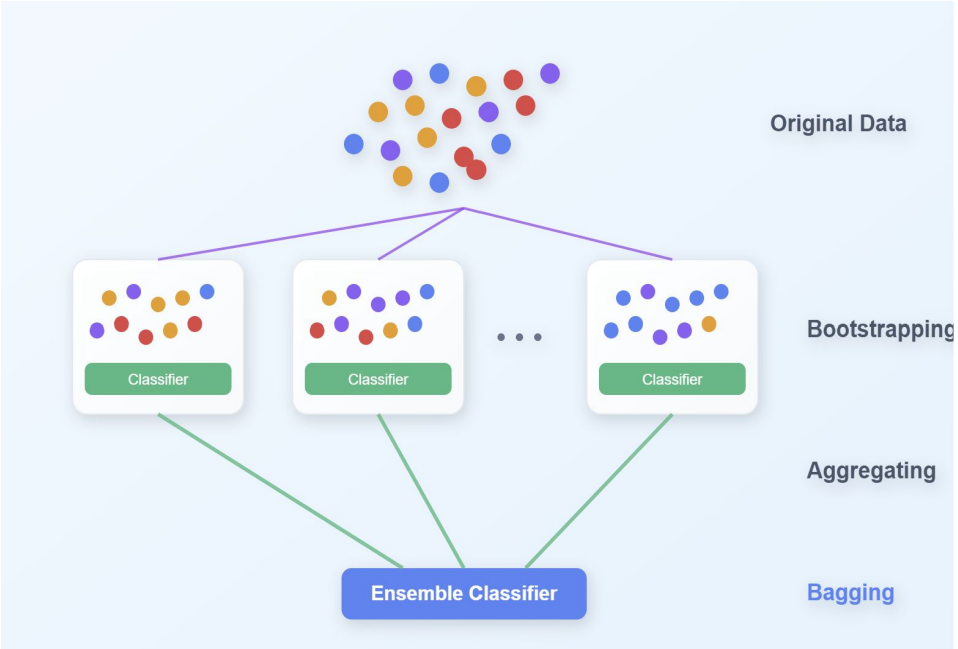
Se fundamentan en la creación de múltiples estimadores entrenados sobre muestras aleatorias de la población objetivo. La finalidad de este enfoque es disminuir la varianza que puede presentar un único estimador, como lo sería un solo árbol de decisión.

Para lograrlo, introduce aleatoriedad en el proceso de construcción y combina los modelos obtenidos mediante el promedio de sus predicciones.

Ejemplo: Random Forest

Ventajas: Fácil de implementar y de controlar.

Desventajas: Son más débiles para obtener buenas predicciones



Hiperparámetros RandomForest

- n_estimators: Número de árboles.
- max_depth: Profundidad máxima de cada árbol.
- min_samples_split: Mínimo número de muestras para dividir un nodo.
- min_samples_leaf: Mínimo número de muestras en una hoja.
- max_features: Número de columnas consideradas al dividir.
- bootstrap: Si se usa muestreo con reemplazo.
- criterion: Función para medir la calidad de la división (por ejemplo, gini o entropy).

Método de Boosting

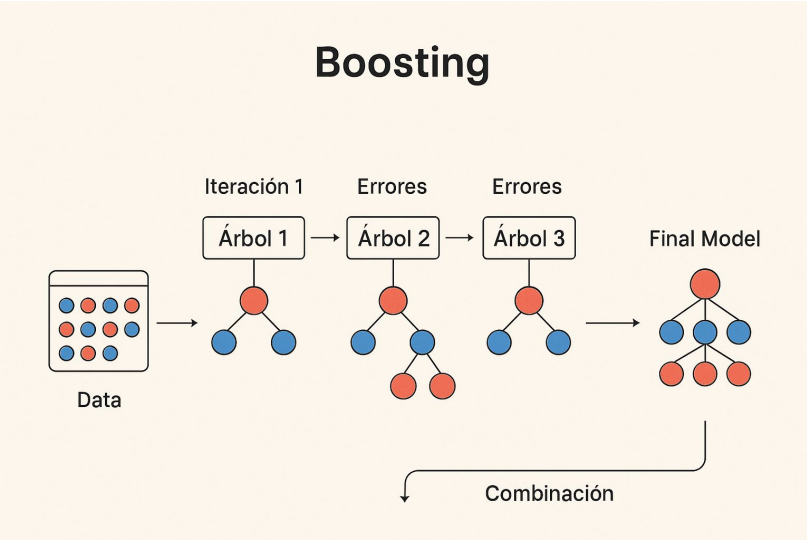
El método de boosting es una técnica poderosa de aprendizaje automático supervisado, especialmente eficaz en datos tabulares. Aunque fue diseñado originalmente para clasificación, también se aplica exitosamente a regresión.

La idea principal consiste en mejorar el rendimiento del modelo mediante un proceso iterativo, donde cada árbol débil se entrena para corregir los errores del modelo anterior.

Cada instancia enfatiza los errores cometidos previamente, permitiendo que el sistema aprenda de ellos y se vuelva más preciso.
Ejemplo: XGBoost; LightGBM

Ventajas: Suelen predecir mejor que otros métodos

Desventajas: Mucho costo computacional y ademas, tienden a overfittear.



Boosting y Descenso del Gradiente

El descenso del gradiente es un algoritmo de optimización que se utiliza para minimizar una función de costo (o pérdida) en problemas de aprendizaje automático. Cómo funciona paso a paso:

1) Definir la función de costo:

   Esta función mide el error del modelo. En boosting, puede ser el error cuadrático medio, la entropía cruzada, etc.

2) Calcular el gradiente:

   El gradiente indica la dirección de mayor incremento de la función de costo. Para reducir el error, debemos movernos en la dirección opuesta al gradiente.

Boosting y Descenso del Gradiente

3) Actualizar los parámetros:

Se ajustan los parámetros del modelo en pasos pequeños, controlados por la tasa de aprendizaje (v), siguiendo la regla:

$$\theta_{\text{nueva}} = \theta_{\text{actual}} - v \cdot \nabla J(\theta)$$

donde J(θ) es la función de costo y $\nabla J(\theta)$ su gradiente.

4) Iterar hasta converger

Se repite el proceso hasta que el cambio en la función de costo sea mínimo o se alcance un número máximo de iteraciones.

Boosting y Descenso del Gradiente

En boosting:
Cada nuevo árbol se entrena para aproximar el gradiente negativo de la función de pérdida respecto a las predicciones actuales. Así, el conjunto de modelos se construye como una suma de correcciones sucesivas:

$$F_m(x) = F_{m-1}(x) + \nu \cdot h_m(x)$$

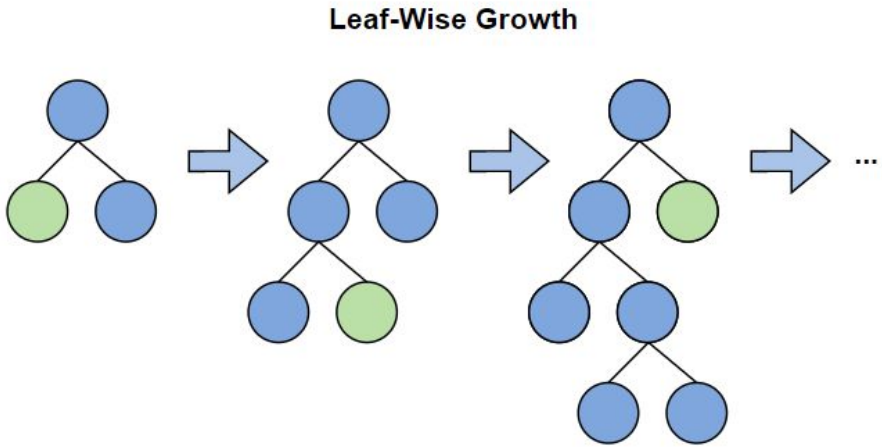
donde hm(x) es el modelo que corrige los errores del anterior.

Hiperparámetros XGBoost y LightGBM

- num_leaves/max_leaves/: Número de hojas.
- learning_rate: Tasa de aprendizaje.
- max_depth: Profundidad máxima de los árboles.
- subsample/bagging_fraction: Proporción de datos usados por árbol.
- colsample_bytree/feature_fraction: Proporción de características usadas por árbol .
- gamma/min_gain_to_split: Mínima ganancia requerida para dividir un nodo.
- reg_alpha & reg_lambda / lambda_l1 & lambda_l2 : Regularización L1 y L2.

LightGBM vs XGBoost

LightGBM



XGBoost