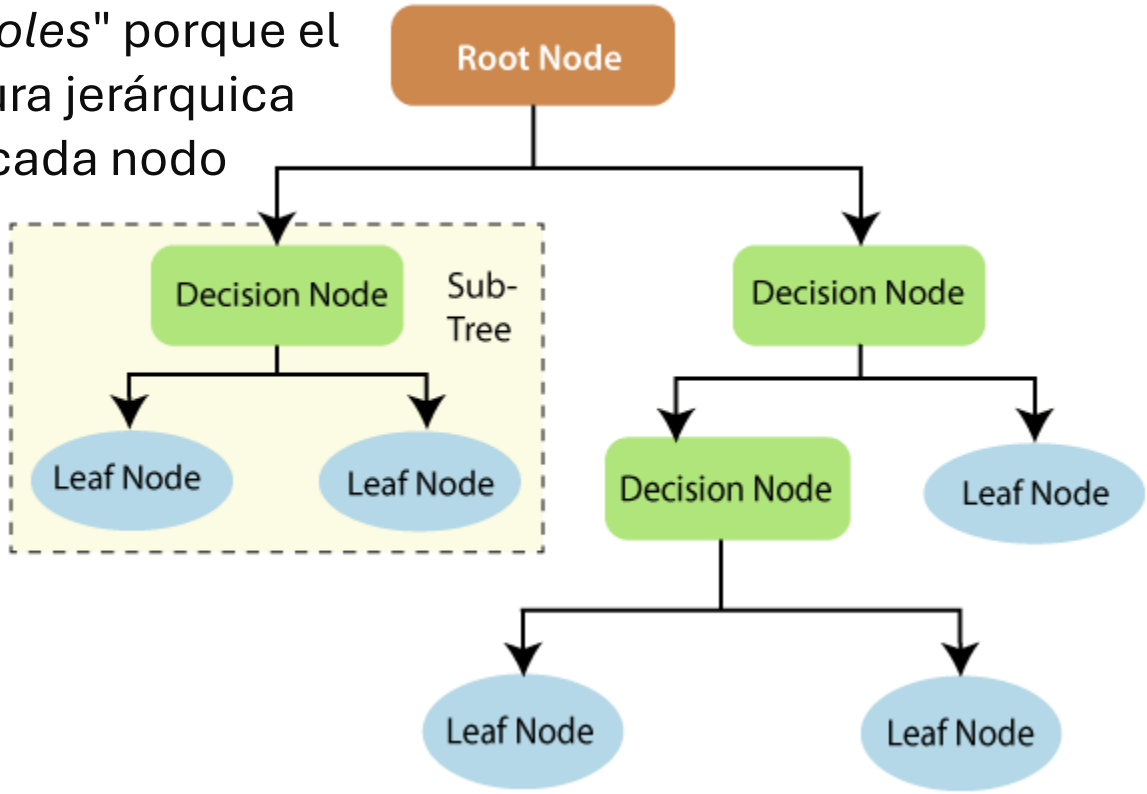
Módulo 4: Aprendizaje Automático

Temario de la Clase 3
23 de agosto del 2024
Árboles de decisión

- Introducción
- Componentes de un árbol de decisión
- Ventajas y desventajas
- Ramificación
- AD para clasificación y regresión
- Métricas
- Validación cruzada
- Poda
- Random Forest

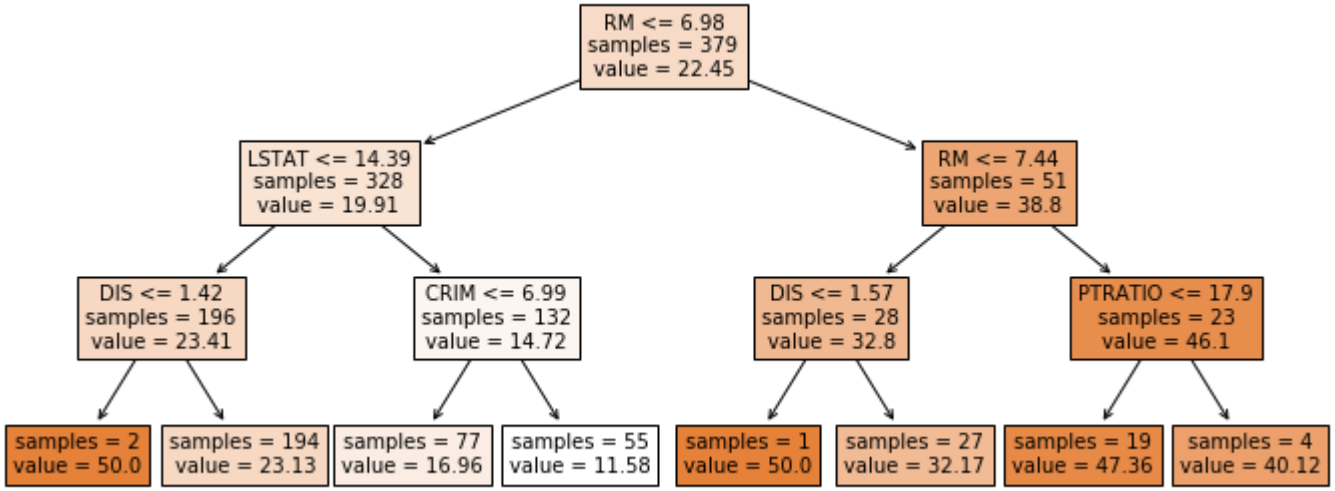
Introducción

Los **árboles de decisión** son un tipo de modelo de aprendizaje supervisado que se utiliza tanto para tareas de **clasificación** como de **regresión**. Se llaman "*árboles*" porque el modelo tiene una estructura jerárquica similar a un árbol, donde cada nodo interno representa una "decisión" basada en un atributo (o característica) del conjunto de datos, y cada hoja final representa una clase o un valor predicho.



Introducción

A diferencia de los árboles reales, este modelo coloca la raíz en la cima, y el AD se desarrolla ramificándose desde arriba hacia abajo, terminando en las hojas.



Componentes de un AD

- **Nodo raíz**: Es el primer nodo del árbol y representa la característica inicial que se usa para dividir el conjunto de datos. Es el nodo del que parten todas las decisiones.
- **Nodos internos o de decisión**: Cada uno de estos nodos representa una característica o atributo del conjunto de datos, y cada bifurcación (o rama) de ese nodo representa una decisión basada en esa característica.
- **Hojas**: Las hojas son los nodos terminales del árbol y representan las clases o valores finales. En el caso de la clasificación, cada hoja corresponde a una clase, y en el caso de la regresión, cada hoja representa un valor numérico.
- **Ramas**: Son las conexiones entre los nodos y representan el resultado de la prueba aplicada a la característica en el nodo padre.

Ventajas

- Fácil de interpretar: La estructura del árbol es intuitiva y sus resultados son altamente comprensibles.
- No requiere mucha preprocesamiento de datos: Los árboles de decisión pueden manejar datos con diferentes tipos de variables y no requieren escalado o normalización de las características.
- Carece de suposiciones sobre la distribución del espacio y la estructura del clasificador.
- Falta de restricciones en tipo de datos: puede aplicarse sobre variables numéricas y categóricas.
- Exploración de datos: permite determinar cuáles variables son más importantes en la estructura de datos y las relaciones entre variables.
- Tiempos de cómputo rápido.
- Mejor tolerancia al ruido.

Desventajas

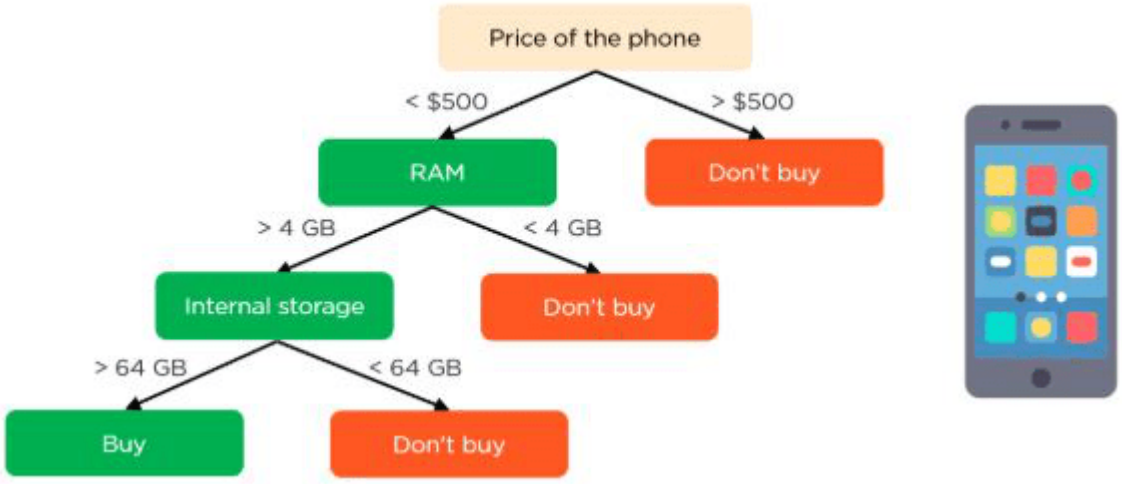
- Propensos al sobreajuste: Un árbol de decisión puede volverse demasiado específico para el conjunto de entrenamiento, lo que reduce su capacidad para generalizar a nuevos datos. Este problema se puede mitigar mediante técnicas como la poda o utilizando un conjunto de árboles como en un Random Forest.
- Una solución consiste en la eliminación de ramas del árbol (poda) y la restricción de ciertos parámetros.

Los árboles de decisión son fundamentales en Machine Learning y se utilizan como base para métodos más complejos, como los bosques aleatorios (Random Forests).

Aplicaciones

- **Clasificación**: en el nodo final, se asume la moda de las observaciones.

- **Regresión**: en el nodo final, se asume el promedio de las observaciones.



Ramificación

El algoritmo determina la estructura del árbol mediante la creación de nodos intermedios, ramas y hojas. Para ello, se utilizan diferentes criterios que se evalúan sobre los datos de entrenamiento para hacer "crecer" el árbol.

Existen distintos criterios de división:

- **Ganancia de información**
- **Entropía**
- **Índice Gini**

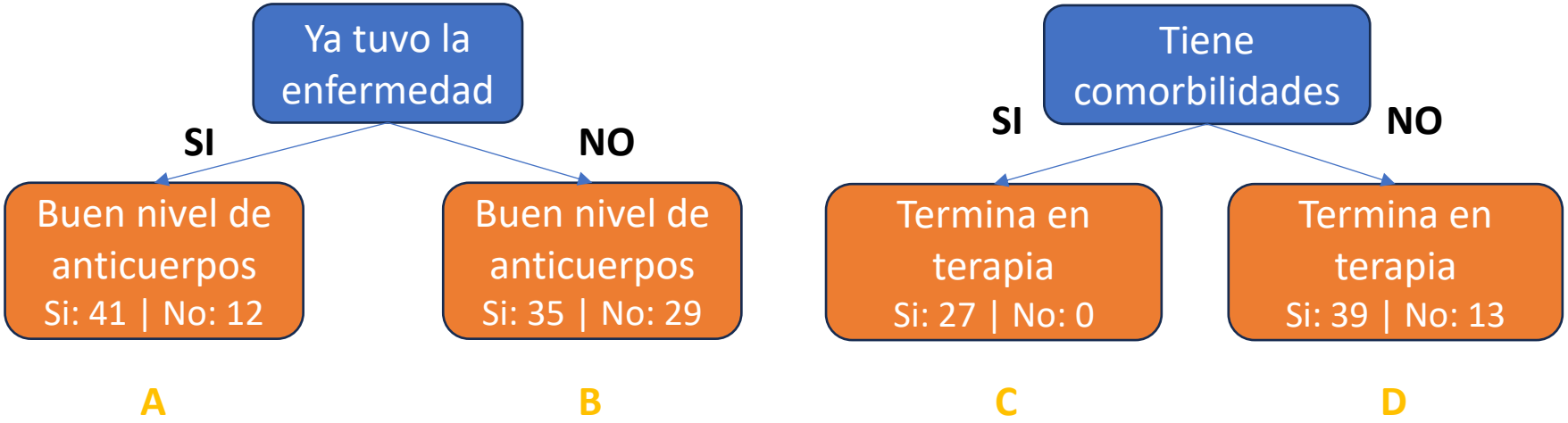
Estos criterios suelen arrojar resultados similares, por lo que nos vamos a focalizar sólo en uno de ellos: el índice Gini.

Ramificación

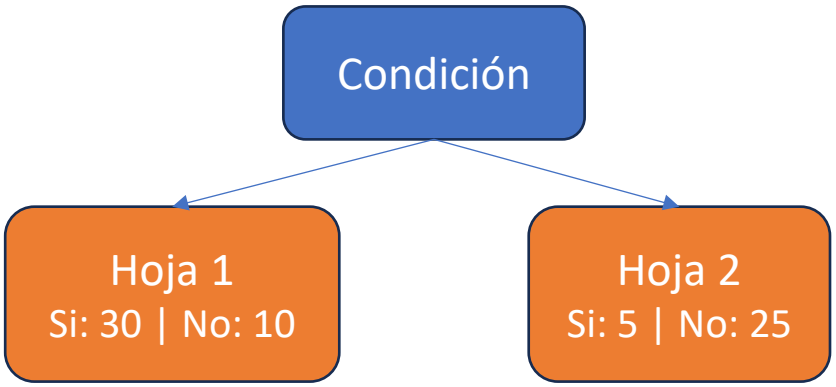
- En teoría, de un nodo pueden salir varias ramas, sin embargo es más fácil trabajar con árboles donde cada nodo tenga solo 2 ramificaciones.
- Cuando se evalúan condiciones, se considera que la rama que sale hacia la izquierda toma la condición como **VERDADERA** y la rama que sale hacia la derecha toma la condición como **FALSA**.
- Si se consideran varias posibles ramificaciones, se elige la que tenga el índice de impureza de Gini con **menor** valor.

Hojas puras e impuras

Una hoja se considera impura cuando contiene mezclas de clasificaciones. Por ejemplo las hojas A, B y D son impuras. Una hoja es pura cuando solo contiene casos de un tipo, por ejemplo la hoja C. Esto nos dice que la ramificación de la derecha tiene mejor poder para clasificar



Índice Gini



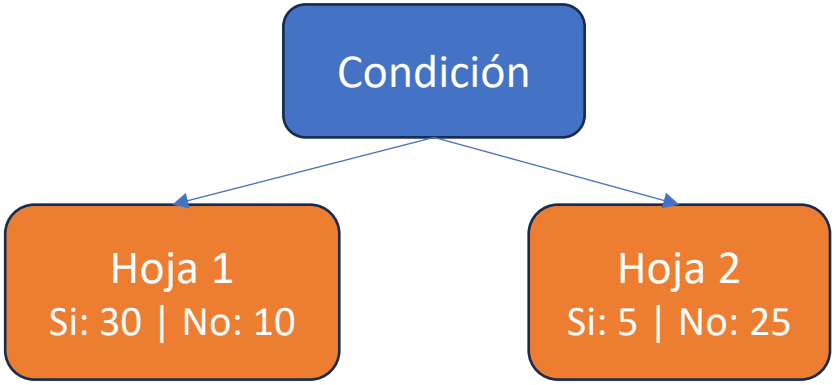
Vamos a calcular el índice Gini de cada hoja para la siguiente división en base a una condición y un nuevo criterio.

$$Impureza\ Gini = 1 - (Probabilidad\ de\ Sí)^2 - (Probabilidad\ de\ No)^2$$

$$Hoja\ 1 = 1 - \left(\frac{30}{30+10}\right)^2 - \left(\frac{10}{30+10}\right)^2 = 0.375$$

$$Hoja2 = 1 - \left(\frac{5}{25+5}\right)^2 - \left(\frac{25}{25+5}\right)^2 = 0.18$$

Índice Gini



Ya que ambas hojas tienen un número distinto de casos en particular, la impureza Gini total para la Condición es la suma ponderada de las impurezas de cada hoja.

$$IG_{Total} = \left(\frac{40}{40+30}\right) * 0.375 + \left(\frac{30}{40+30}\right) * 0.18 = 0.291$$

Métricas de Evaluación

¿Cómo sabemos si nuestro modelo generaliza bien y no hace overfitting?

En cualquier proyecto de Machine Learning, una vez entrenado un algoritmo, siempre llega la parte de evaluarlo.

Índice

1. Matriz de Confusión
2. Métricas de Clasificación
    a. Accuracy
    b. Precision
    c. Recall
    d. F1 Score
    e. Curva ROC / AUC
3. Métricas de Regresión
    a. Mean Squared Error (MSE)

Matriz de Confusión

En el caso de un ***problema binario***, nuestro modelo clasificaría como 0 o 1 un conjunto de datos, y a partir de esto podemos crear la matriz de confusión.

Valores Predichos

Valores Reales

	Positivo	Negativo
Positivo	TP	FP
Negativo	TN	FN

Matriz de Confusión

Aciertos:

TP (True Positive) – Son los valores que el algoritmo clasifica como positivos y que realmente son positivos.

TN (True Negative) – Son valores que el algoritmo clasifica como negativos (0 en este caso) y que realmente son negativos.

Errores:

FP (False Positive) – Falsos positivos, es decir, valores que el algoritmo clasifica como positivo cuando realmente son negativos.

FN (False Negative) – Falsos negativos, es decir, valores que el algoritmo clasifica como negativo cuando realmente son positivos.

Error tipo I: Falsos positivos

Error tipo II: Falsos negativos

Métricas de Clasificación

Accuracy: proporciona una instantánea rápida de qué tan bien se está desempeñando el modelo en términos de predicciones correctas.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Funciona muy bien para casos de equidad de cantidad de muestras para cada clase.

La precisión de la clasificación es buena, pero da una sensación de falso positivo de lograr una alta precisión.

El problema surge debido a que la posibilidad de clasificación errónea de muestras de clases menores es muy alta.

Métricas de Clasificación

Precision (Precisión): indica cuántas de las predicciones positivas realizadas por el modelo son realmente correctas.

$$Precision = \frac{TP}{(TP + FP)}$$

Recall (Sensibilidad): También conocida como el ratio de verdaderos positivos, es utilizada para saber cuantos valores positivos son correctamente clasificados.

$$Recall = \frac{TP}{(TP + FN)}$$

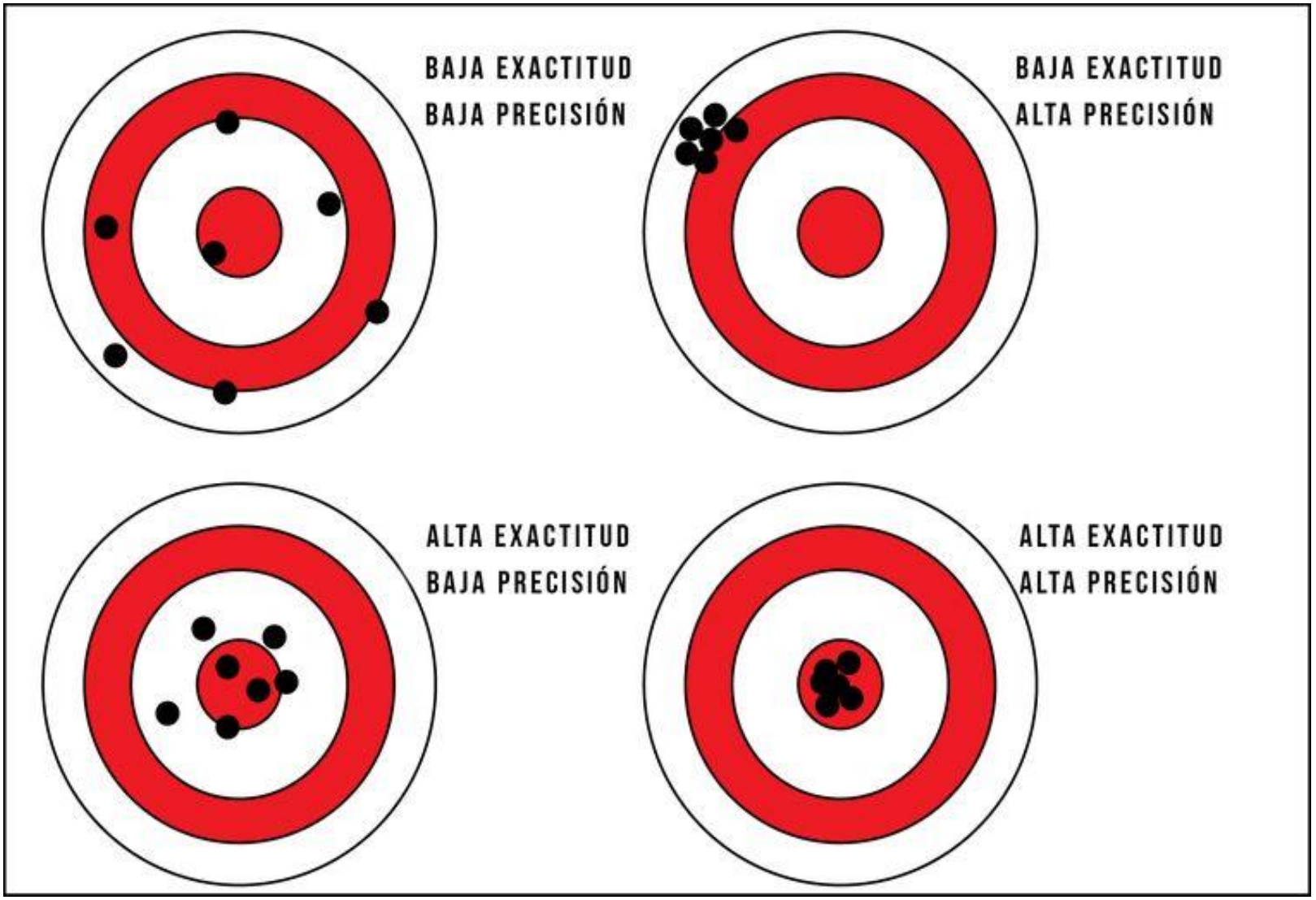
Métricas de Clasificación

Especificidad (la tasa de verdaderos negativos): Se obtiene dividiendo el número total de verdaderos negativos entre la suma de los verdaderos negativos y los falsos positivos.

$$Especificidad = \frac{TN}{TN + FP}$$

Tasa de falsos positivos: Se considera como la tasa de «falsa alarma» y resumen como de común es que una clase negativa sea determinada por el modelo como positiva.

$$Tasa\ de\ falsos\ positivos = \frac{FP}{TN + FP}$$


BAJA EXACTITUD
BAJA PRECISIÓN
BAJA EXACTITUD
ALTA PRECISIÓN
ALTA EXACTITUD
BAJA PRECISIÓN
ALTA EXACTITUD
ALTA PRECISIÓN

Métricas de Clasificación

F1 Score: Muy utilizada en problemas en los que el conjunto de datos a analizar está desbalanceado. Esta métrica combina el ***precision*** y el ***recall***, para obtener un valor mucho más objetivo.

Su rango es [0,1]. Esta métrica generalmente nos dice qué tan preciso (clasifica correctamente cuántas instancias) y robusto (no pierde ningún número significativo de instancias) es nuestro clasificador.

$$F1\ Score = 2 * \frac{precision * recall}{precision + recall}$$

Ejemplo

Valor Real	Predicción
1	1
1	1
1	0
0	1
1	0
0	0

TP: 2
TN: 1
FP: 1
FN: 2

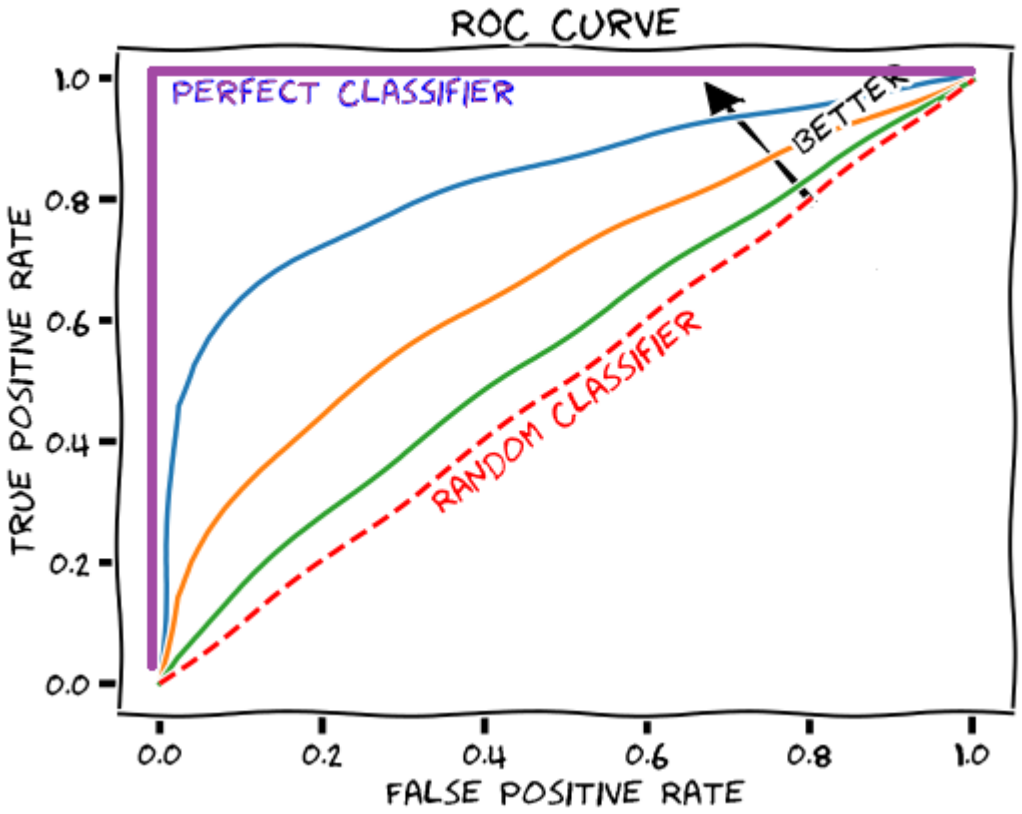
Métricas de Clasificación

Curvas ROC: Receiver Operating Characteristic)
La gráfica representa el porcentaje de verdaderos positivos (***Recall***), contra el ratio de falsos positivos (False Positive Rate), que es la inversa de la ***especificidad***.

La diferencia con el resto de métricas, es que en este caso, el umbral por el que se clasifica un elemento como 0 o 1, se va modificando, para poder ir generando todos los puntos de la gráfica.

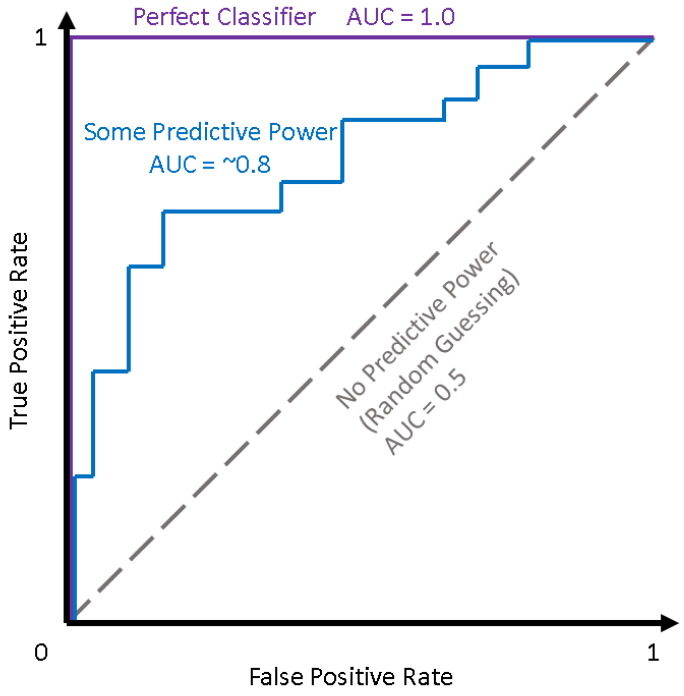
Métricas de Clasificación

Curvas ROC



Métricas de Clasificación

AUC (Area Under Curve): Área Bajo la Curva ROC
Generalmente, cuanto mayor es la puntuación AUC, mejor es el rendimiento de un clasificador binario para una tarea de clasificación dada.



Para un clasificador sin poder predictivo (es decir, de predicción aleatoria), el AUC es 0.5, y para un clasificador perfecto, el AUC es 1.0; esto es el área bajo sus correspondientes curvas ROC.

Métricas de Regresión

Mean Squared Error (MSE) (Error cuadrático medio)

Toma el cuadrado del promedio de los valores predichos y originales.
Es fácil calcular el gradiente.
Al tomar el cuadrado de los errores, se pronuncian más los errores más grandes que los errores más pequeños, por lo que podemos centrarnos más en los errores más grandes.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Cross-Validation

Es una técnica empleadas para probar la eficacia de un modelo de Machine Learning.

Es un procedimiento de “re-sampling” (remuestreo) que permite evaluar un modelo incluso con datos limitados.

Para efectuar una “CV” (cross-validation), hace falta apartar de antemano una parte de los datos de la serie de datos de entrenamiento. Esos datos no se utilizarán para entrenar el modelo, sino más tarde para probarlo y validarlo.

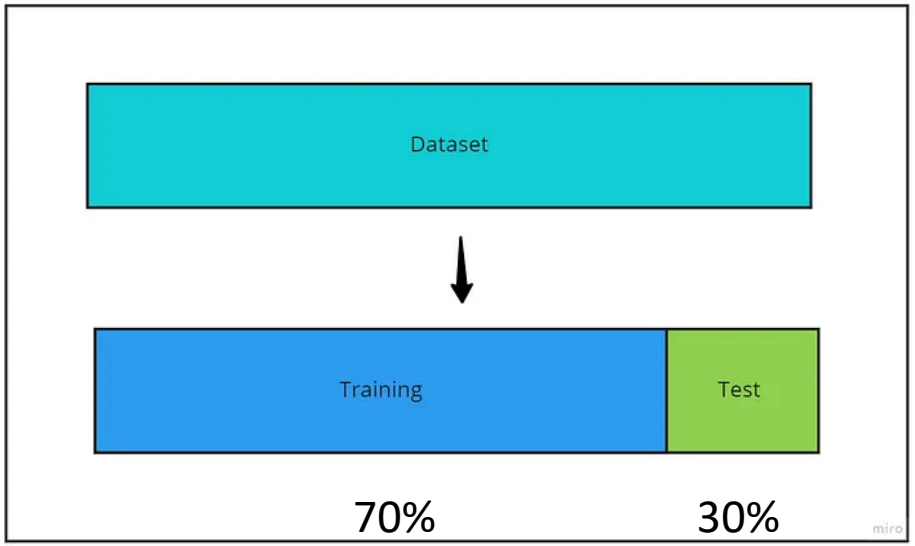
A menudo en Machine Learning se usa la cross-validation para comparar los diferentes modelos y seleccionar el más adecuado para un problema específico.

Técnicas de Validación Cruzada

Holdout Cross-Validation
(Validación Cruzada de Retención)

train_test_split

Esta técnica consiste en separar todo el conjunto de datos en dos grupos, sin solapamiento: conjuntos de entrenamiento y de prueba.
Esta separación se puede realizar barajando los datos o manteniendo su ordenación, según el proyecto.



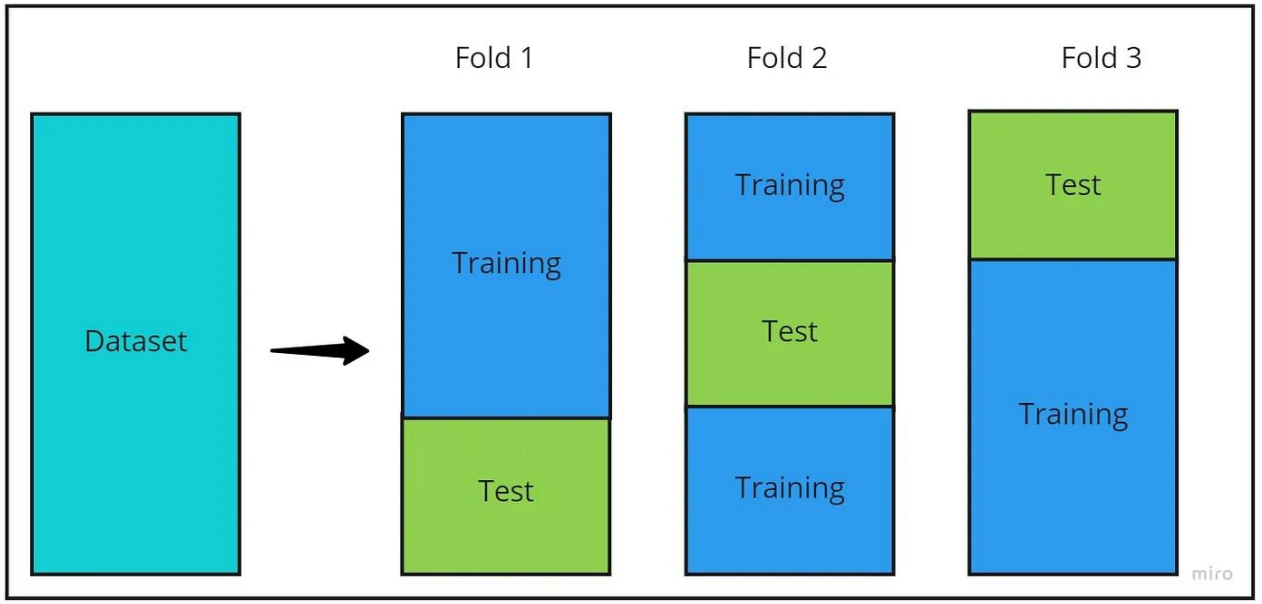
Técnicas de Validación Cruzada

K-Fold Cross-Validation
(Validación Cruzada de K pliegues)

Kfold

Antes de separar los datos en conjuntos de entrenamiento y prueba, la K-Fold CV separa todos los datos en *K* subconjuntos separados con un tamaño aproximado. Solo entonces, cada subconjunto se divide en conjuntos de entrenamiento y prueba.

Cada subconjunto se utiliza para entrenar y probar el modelo.
La técnica genera K modelos diferentes con K resultados diferentes.
El resultado final es el promedio de las métricas individuales de cada subconjunto.



Técnicas de Validación Cruzada

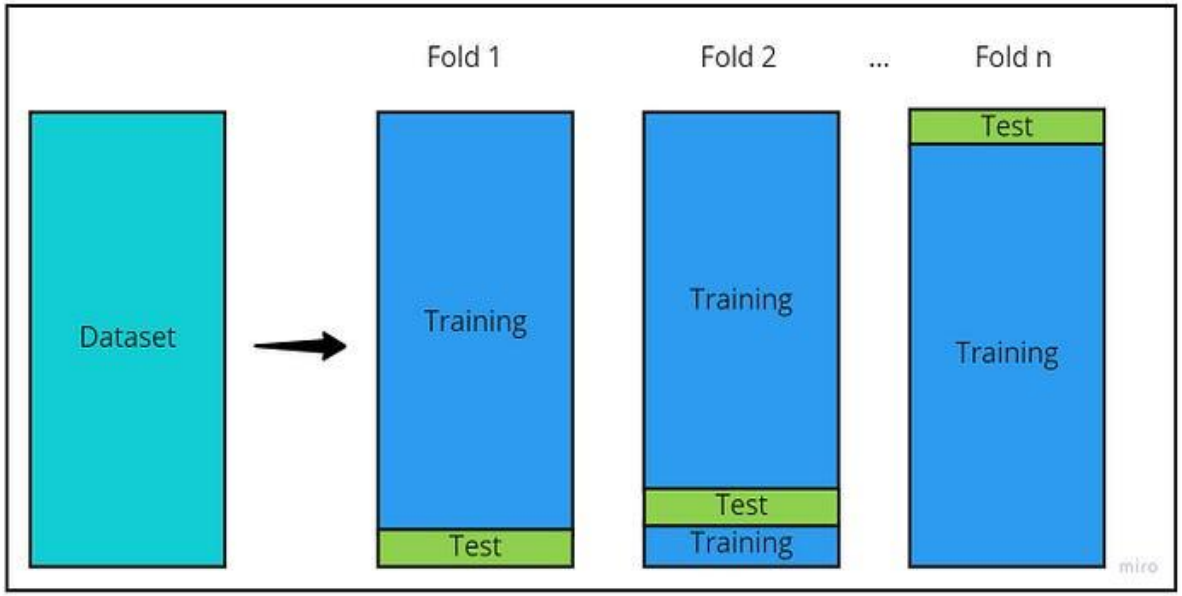
LeaveOneOut

Leave-One-Out Cross-Validation
(Validación cruzada con opción de dejar uno fuera)

Creación de varios conjuntos de prueba y de entrenamiento, donde el conjunto de prueba contiene solo una muestra de los datos originales y el conjunto de entrenamiento consta de todas las demás muestras de los datos originales. Este proceso se repite para todas las muestras del conjunto de datos original.

Desventaja: Gran consumo de cálculos.
n muestras→ *n* veces entrenamiento y validación.

Ventaja: Mayor cantidad de muestras utilizadas para el entrenamiento, y esto puede resultar en mejores modelos desarrollados**.**



Ejemplo: árbol para clasificación binaria

Sobre el siguiente dataset, que combina variables numéricas con cualitativas, vamos a crear un AD para determinar si un paciente es apto para participar en un estudio clínico.

Realiza actividad física	Antecedentes cardiacos	Edad	Apto para el estudio
Si	Si	7	**No**
Si	No	12	**No**
No	Si	18	**Si**
No	Si	35	**Si**
Si	Si	38	**Si**
Si	No	50	**No**
No	No	83	**No**

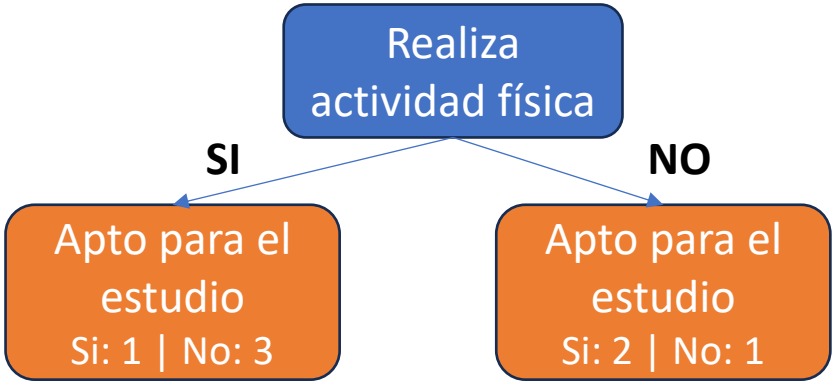
Ejemplo: árbol para clasificación binaria

El primer paso consiste en determinar cuál es la variable que se va a utilizar como nodo raíz. Para ello debemos determinar cuál de las 3 variables independientes (actividad física, antecedentes, edad) tiene el menor índice Gini.

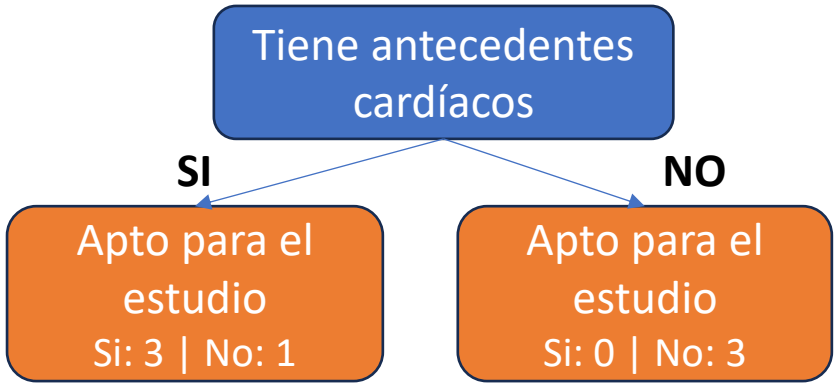
Realiza actividad física	Antecedentes cardiacos	Edad	Apto para el estudio
Si	Si	7	**No**
Si	No	12	**No**
No	Si	18	**Si**
No	Si	35	**Si**
Si	Si	38	**Si**
Si	No	50	**No**
No	No	83	**No**

Ejemplo: árbol para clasificación binaria

El primer paso consiste en determinar cuál es la variable que se va a utilizar como nodo raíz. Para ello debemos determinar cuál de las 3 variables independientes (actividad física, antecedentes, edad) tiene el menor índice Gini.



Impureza Gini = 0.405



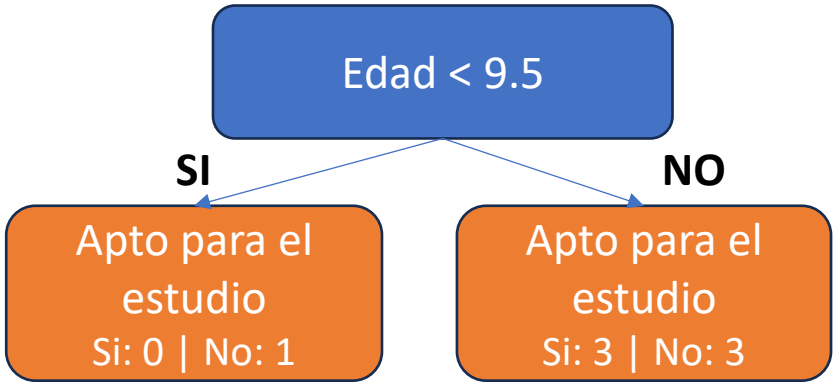
Impureza Gini = 0.214

Ejemplo: árbol para clasificación binaria

En el caso de variables numéricas se toma el promedio entre los valores y se calcula el índice Gini tomando cada promedio como condición de umbral.

Edad	Apto para el estudio
7	No
12	No
18	Si
35	Si
38	Si
50	No
83	No

9.5
15
26.5
36.5
44
66.5



Ejemplo: árbol para clasificación binaria

Vemos que dos condiciones Edad < 15 y Edad < 44 tienen la menor impureza Gini por lo que ambos se pueden utilizar.

Edad	Apto para el estudio
7	**No**
12	**No**
18	**Si**
35	**Si**
38	**Si**
50	**No**
83	**No**

	Impureza Gini
9.5	0.429
15	**0.343**
26.5	0.476
36.5	0.476
44	**0.343**
66.5	0.429

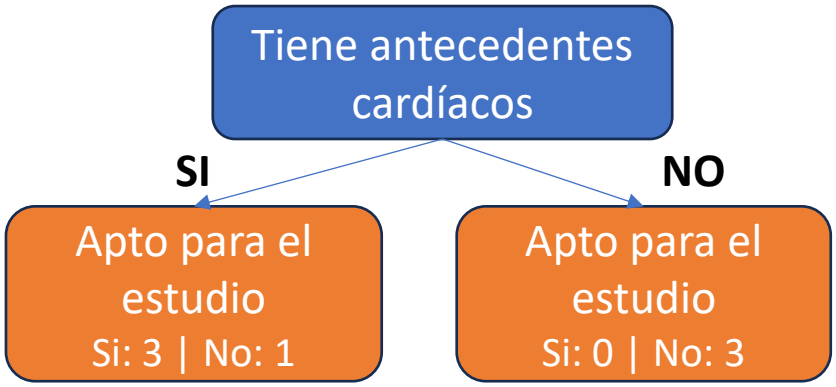
Ejemplo: árbol para clasificación binaria

Ahora que tenemos los índices Gini de las 3 variables independientes, tomamos la menor para determinar el nodo raíz.

	Impureza Gini
Realiza actividad física	0.405
Tiene antecedentes cardíacos	**0.214**
Edad < 44	0.343

Ejemplo: árbol para clasificación binaria

El nodo de la izquierda es impuro, por lo que puede dividirse utilizando el criterio de la **edad** o de la realización de **actividad física**.



Realiza actividad física	Edad	Apto para el estudio
Si	7	**No**
No	18	**Si**
No	35	**Si**
Si	38	**Si**

12.5
26.5
36.5

Ejemplo: árbol para clasificación binaria

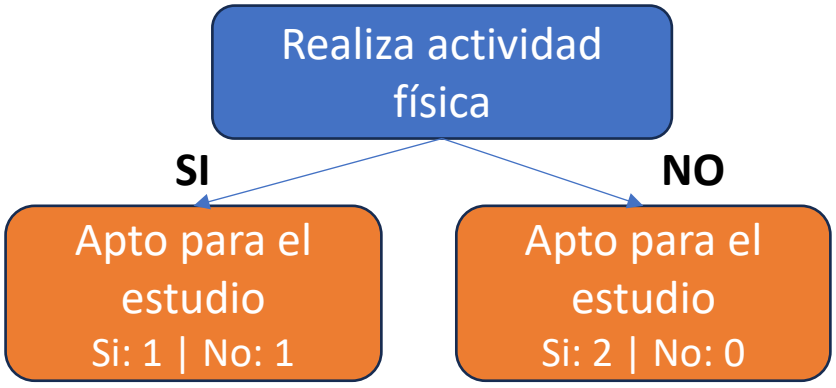
Se determina el criterio que ofrece menor índice Gini

Realiza actividad física	Edad	Apto para el estudio
Si	7	**No**
No	18	**Si**
No	35	**Si**
Si	38	**Si**

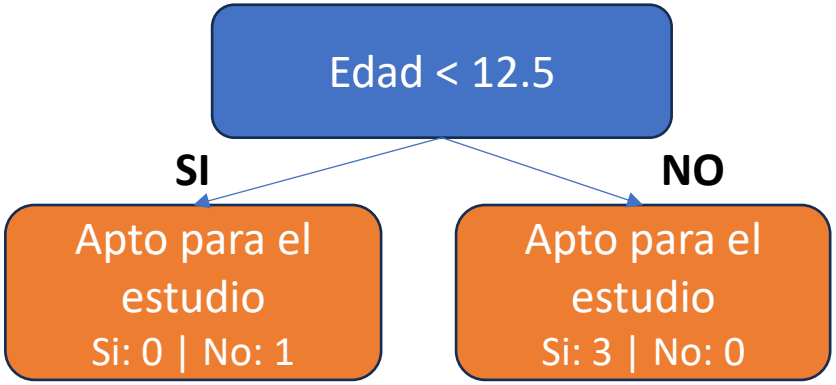
12.5

26.5

36.5



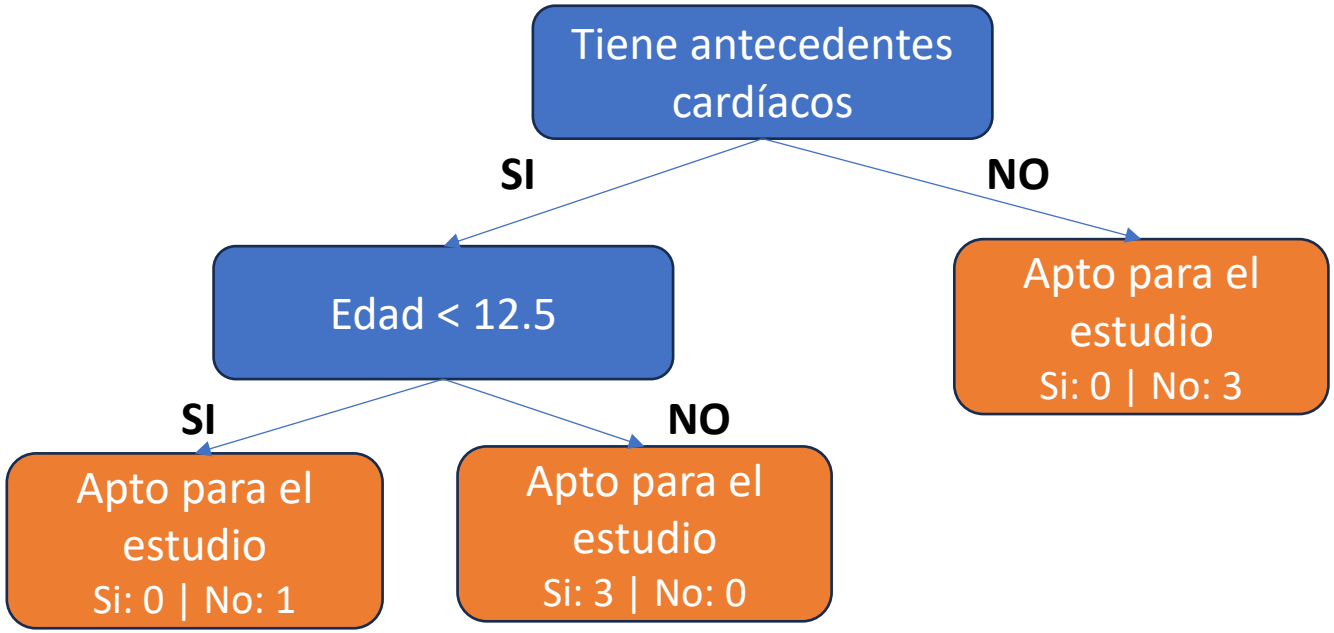
Impureza Gini = 0.25



Impureza Gini = 0

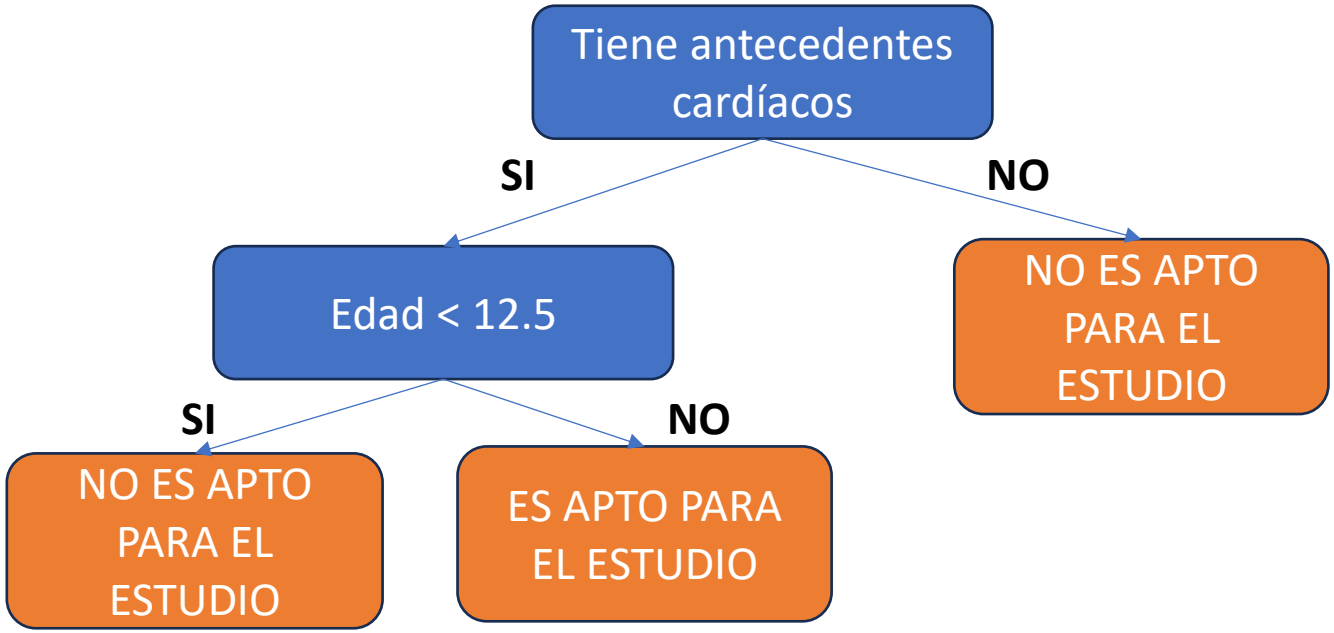
Ejemplo: árbol para clasificación binaria

Se llegó a un árbol cuyos últimos nodos solamente contienen casos del mismo tipo, por lo que no se pueden seguir dividiendo.



Ejemplo: árbol para clasificación binaria

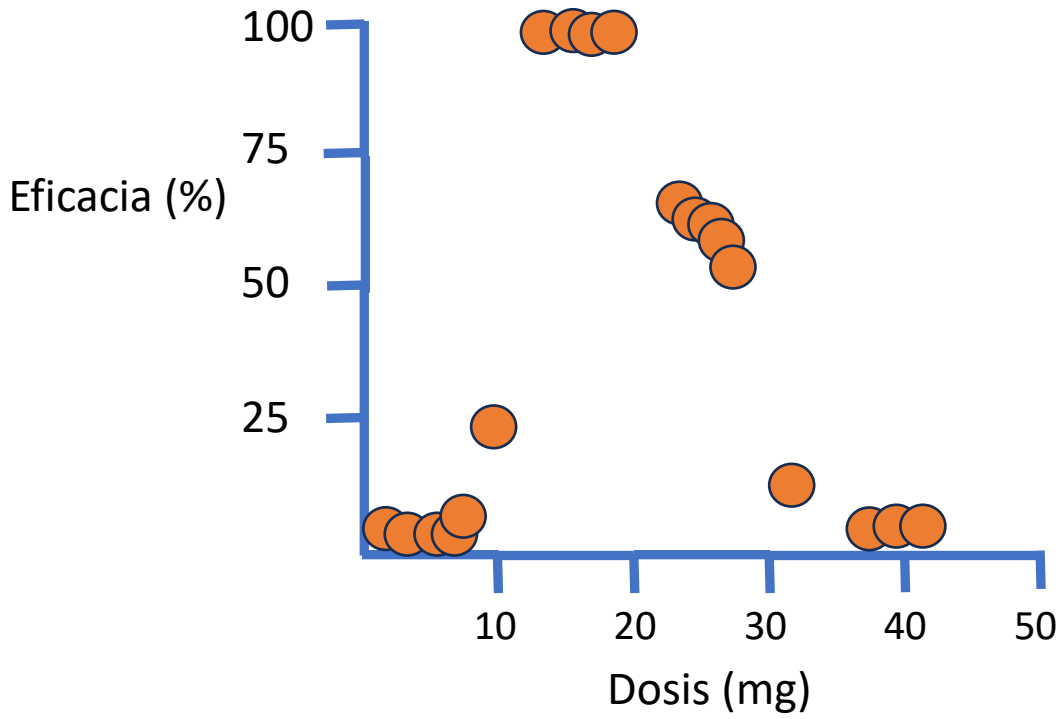
Entonces dichos nodos se convierten en hojas puesto que dan fin a cada rama y permiten clasificar un nuevo caso que se presente.



Nótese como no es necesario incluir el criterio de actividad física en el árbol. Con los dos criterios presentes ya es suficiente.

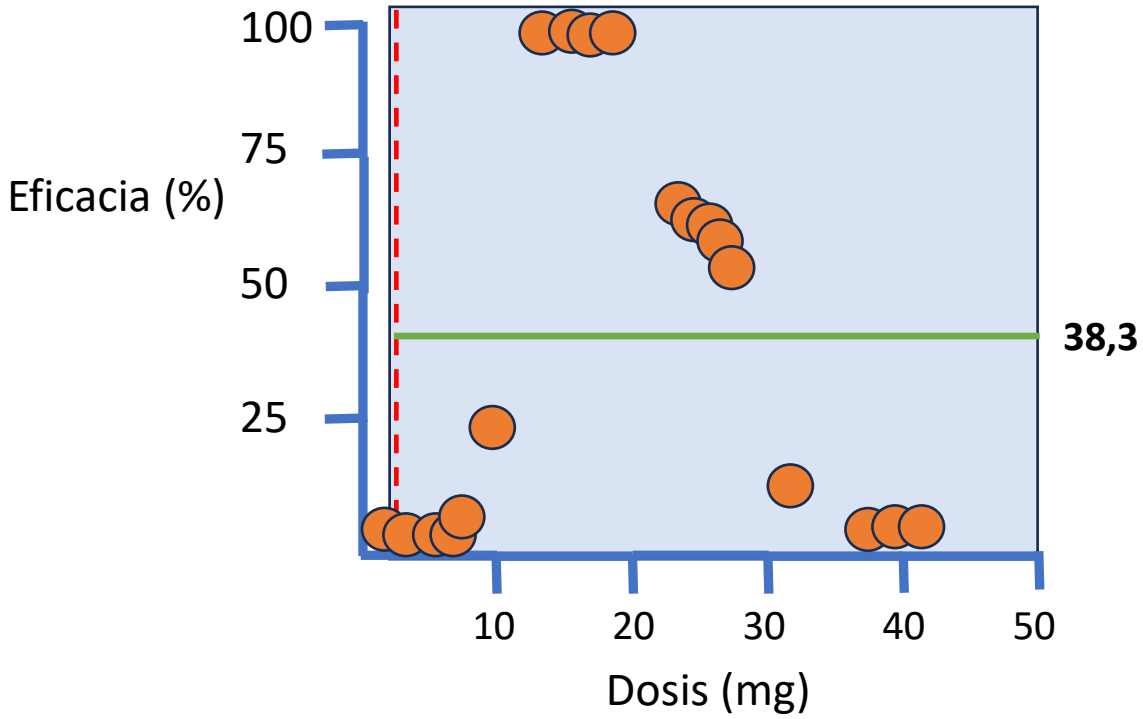
Ejemplo: árbol para regresión

- Supongamos que tenemos los siguientes datos, que relacionan la eficacia de una nueva droga con la dosis de la misma. Se entiende que la relación no es lineal.



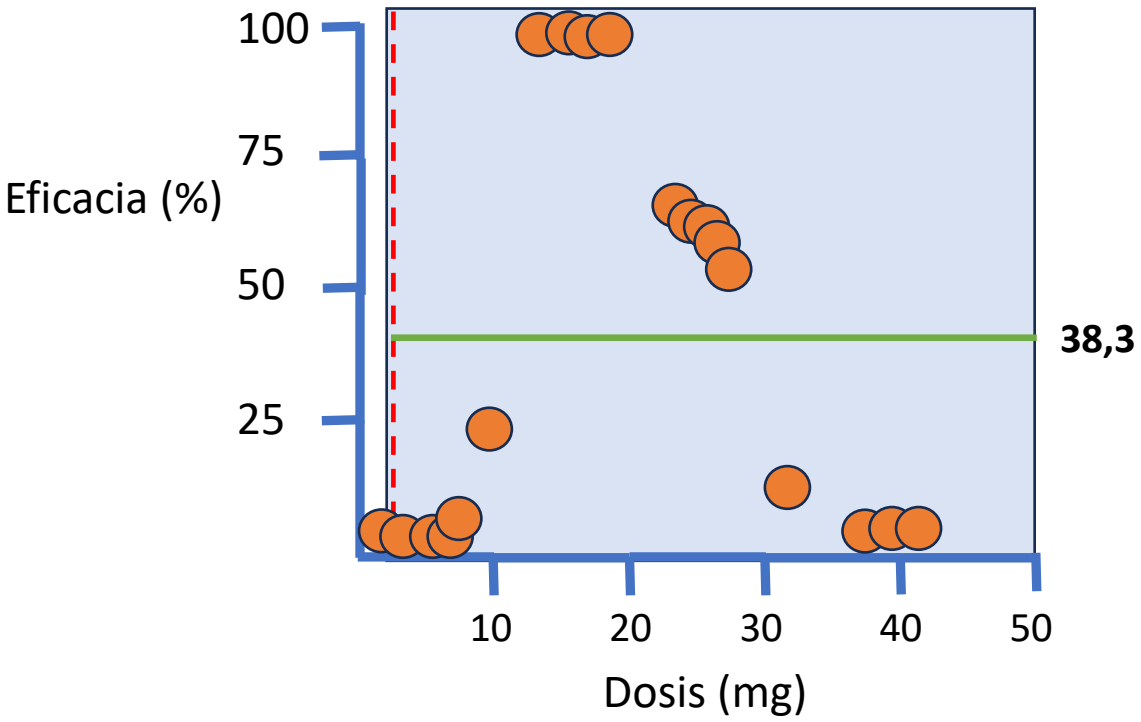
Ejemplo: árbol para regresión

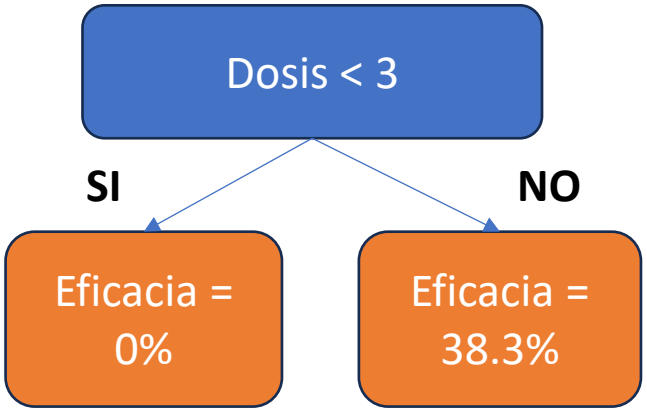
- Veamos como obtener la raíz del árbol. Primero tomamos el promedio de las dosis de las dos primeras muestras (3 mg) y dividimos el dataset en dos. Luego, calculamos la eficacia promedio de los dos grupos resultantes (0% y 38.3%).



Ejemplo: árbol para regresión

- Podemos considerar un árbol básico que clasifique las dosis menores a 3 mg con 0% y las mayores a 3 mg con 38,3%

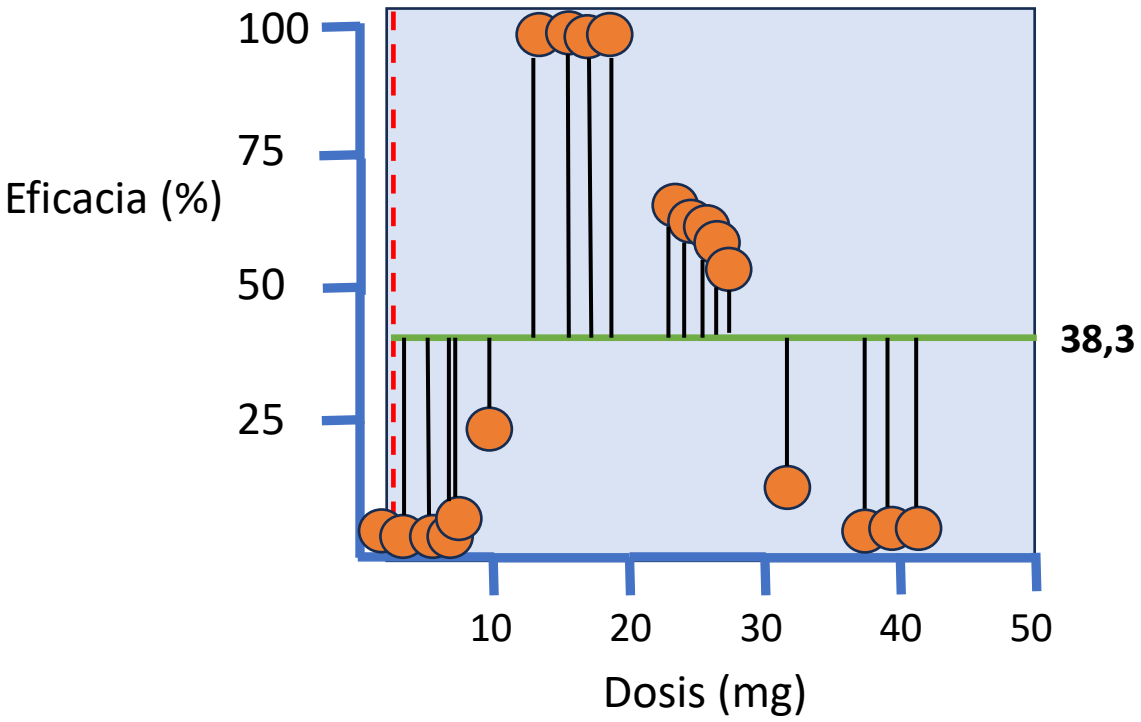


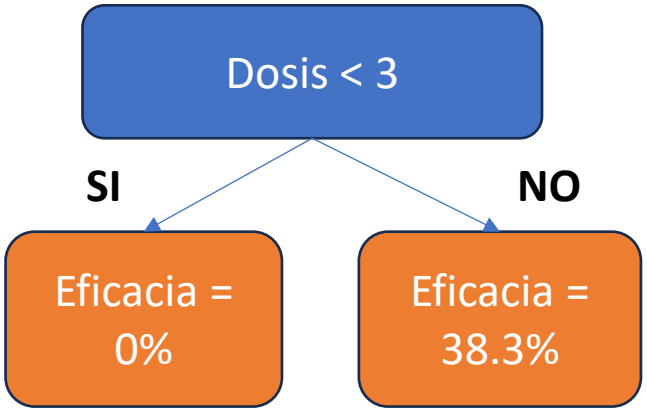


Naturalmente dicha suposición tendrá un error asociado, puesto que conocemos las eficacias reales.

Ejemplo: árbol para regresión

- Calculamos el error de considerar dicho valor predicho por el árbol respecto del real y obtenemos el error cuadrático medio (MSE) de dicho modelo.

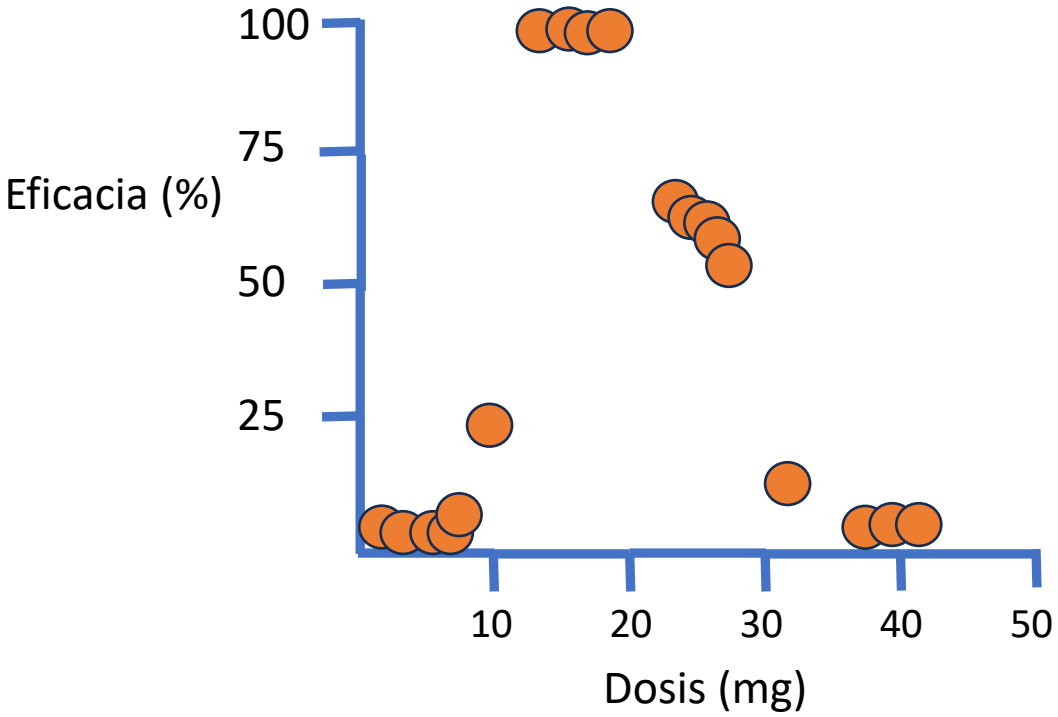




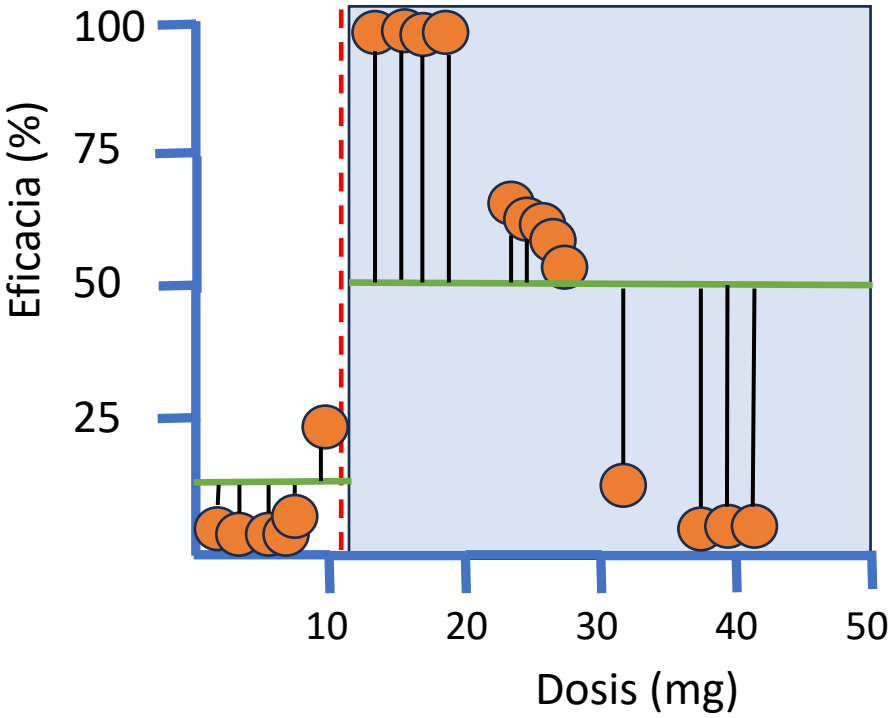
MSE (dosis <3) = 1526,02

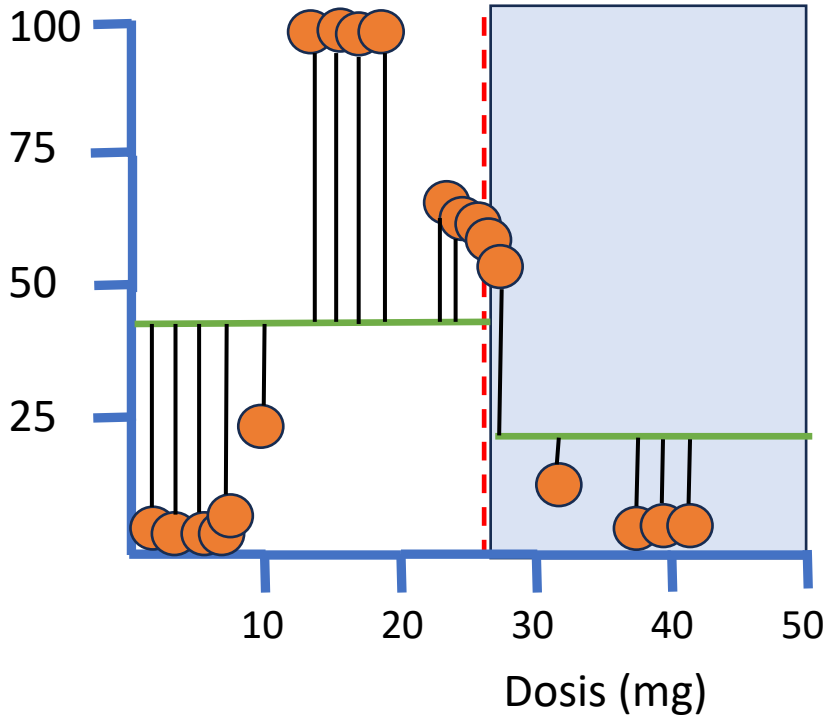
Ejemplo: árbol para regresión

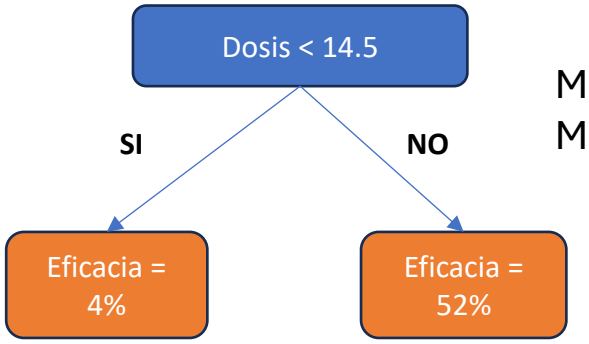
- Se repite el procedimiento sucesivamente, luego tomando la 2da y 3er muestra, luego la 3ra y la 4ta, la 5ta y la 6ta, etc. Se toma el valor promedio de las dosis, se dividen las muestras en 2 grupos, se toma la eficacia promedio de ambos grupos, y se calcula el MSE del árbol resultante de considerar los promedios.



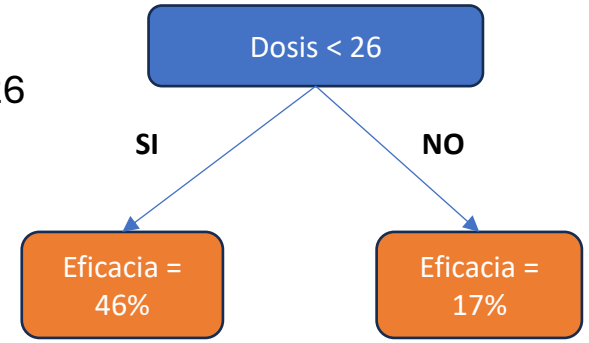
Ejemplo: árbol para regresión





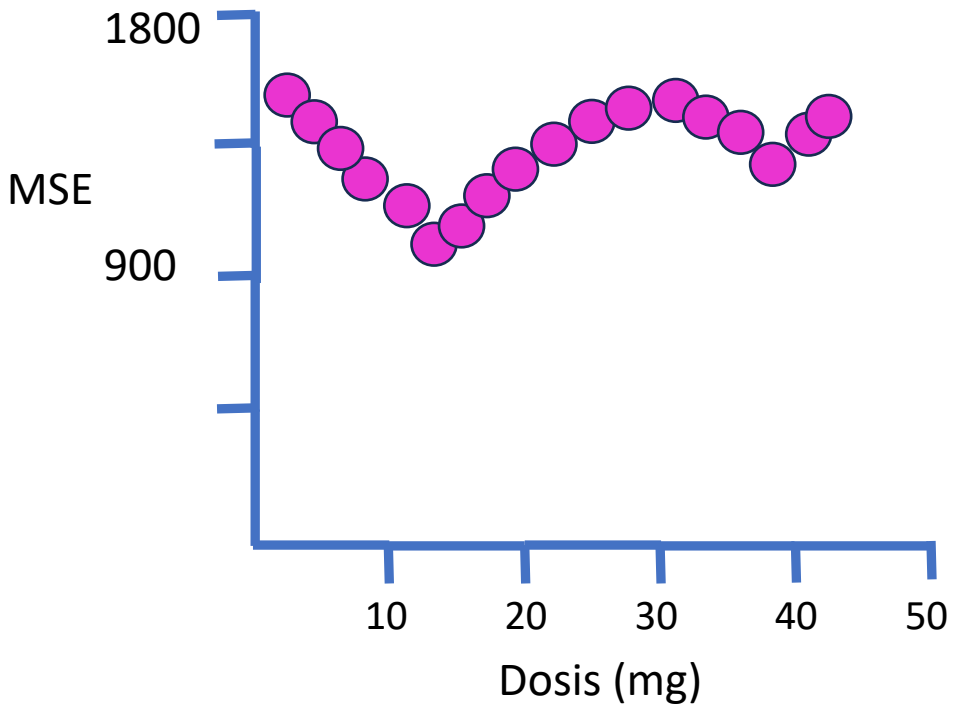


MSE (dosis <14.5)= 1026
MSE (dosis <26)= 1658



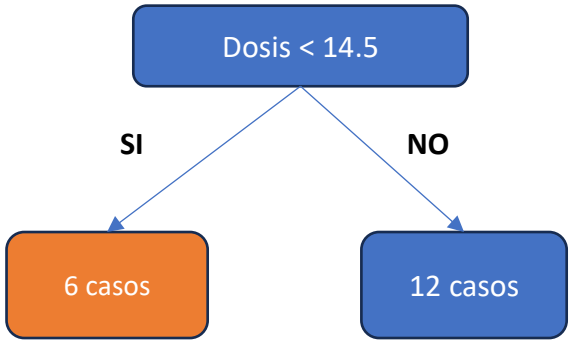
Ejemplo: árbol para regresión

- Una vez que repetí para todos los pares de valores consecutivos, y calculé los correspondientes MSE, tengo que buscar el caso que tuvo el MSE menor. El cual resulta para 14.5 mg.



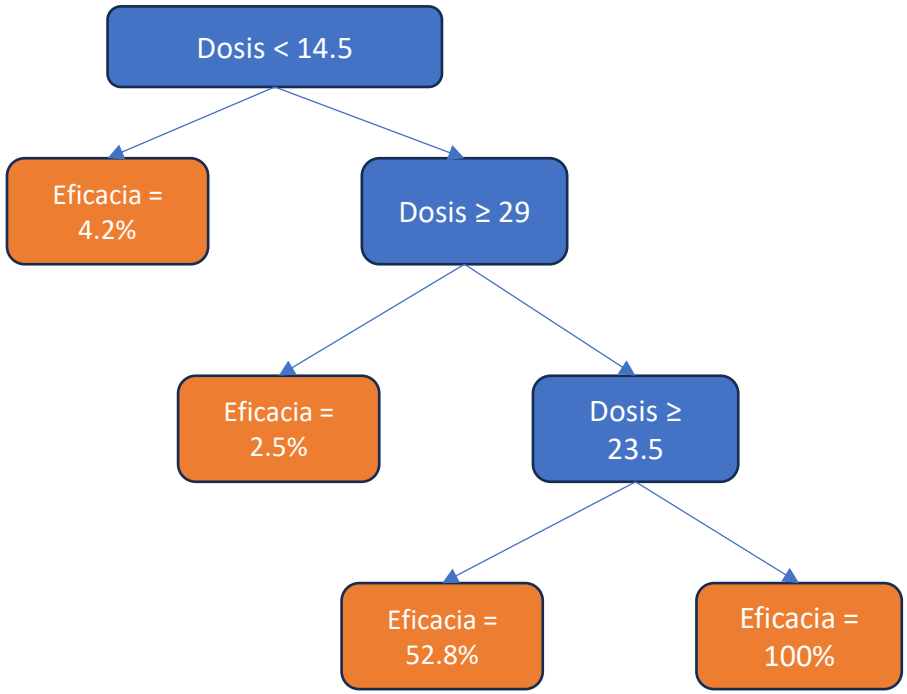
Ejemplo: árbol para regresión

- Entonces utilizo Dosis < 14.5 como nodo raíz del árbol y divido el dataset en 2 grupos. Los pasos seguidos anteriormente pueden repetirse sobre cada uno de estos grupos resultando en divisiones con cada vez menos casos. La hoja de la izquierda solo tiene 6 casos. Tener hojas que contengan pocos casos es equivalente a **sobreajustar** el árbol a los **datos de entrenamiento**, por lo que es recomendable determinar un número mínimo de casos que deba contener un nodo para poder dividirse, para que el árbol no pierda generalidad. Por ejemplo podemos fijar dicho valor en 7.



Ejemplo: árbol para regresión

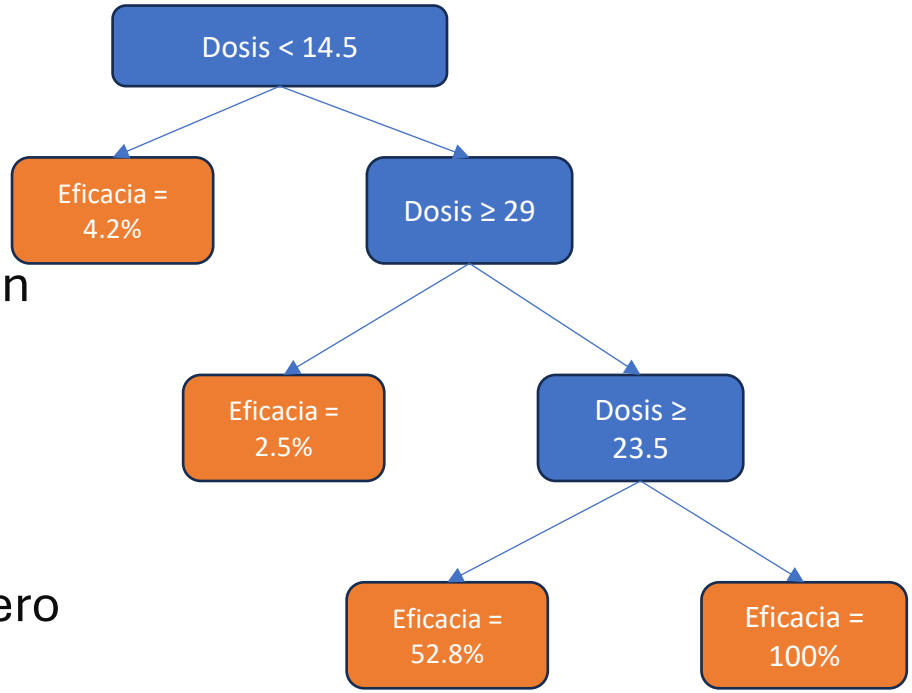
- Puesto que la hoja de la derecha tiene más de 7 casos, se puede seguir dividiendo realizando los pasos anteriores. Continúo el proceso hasta llegar a hojas con menos de 7 casos. El valor que predice la hoja es el promedio de las observaciones que contiene.



Optimización de los AD

Anteriormente establecimos un número mínimo de ejemplos que deben pertenecer a un nodo para poder dividirlo. Esto se hizo para evitar el sobreajuste del árbol a los datos de entrenamiento. Intuitivamente, podemos tener la noción de que un mayor número de hojas resultará en un árbol con menor generalidad.

Otro criterio para optimizar un árbol es establecer un valor máximo de niveles de ramificación (profundidad del árbol) como parámetro de parada para el crecimiento.



También se puede definir un número máximo de hojas en el AD.

Poda

- La poda (pruning) consiste en la eliminación de hojas y ramas para producir árboles menos complejos. El árbol “completo” presenta mas complejidad pero menor error sobre el dataset.

- Árboles más simples poseen menos bifurcaciones lo cual reduce la propensión al mismo al sobreajuste pero producirán mayor error.

- Existen técnicas basadas en la identificación de parámetros que permitan construir y comparar distintas versiones de un árbol para determinar cuál conviene elegir.

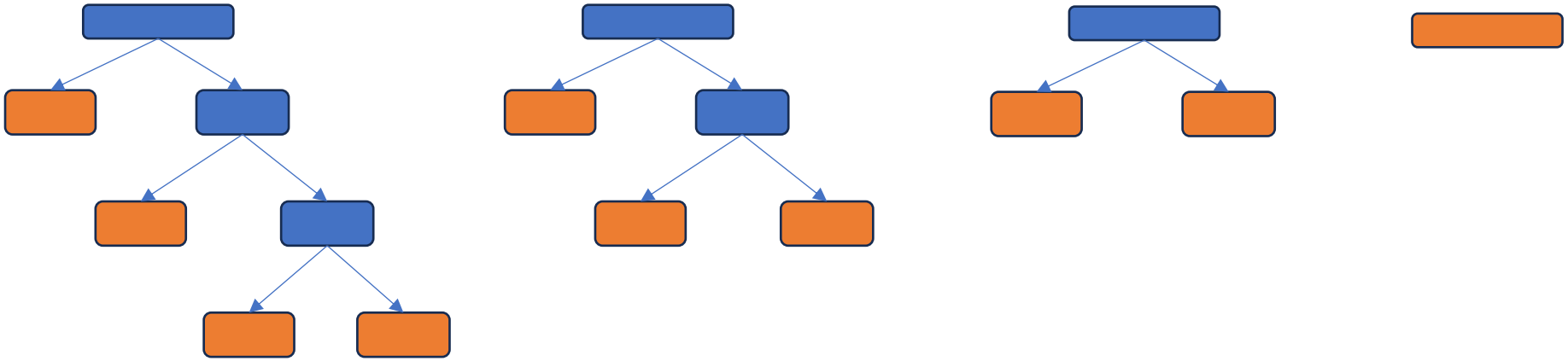
- Vamos a comentar una técnica llamada **poda de complejidad de coste** (*cost complexity pruning*)

Poda

Esta técnica califica cada árbol creado en función de algúna métrica de error y un término de complejidad, que penaliza con valores más grandes a árboles con mayor número de hojas, y por lo tanto más complejos. La función de complejidad es la siguiente:

$$C_\alpha = Error + \alpha T$$

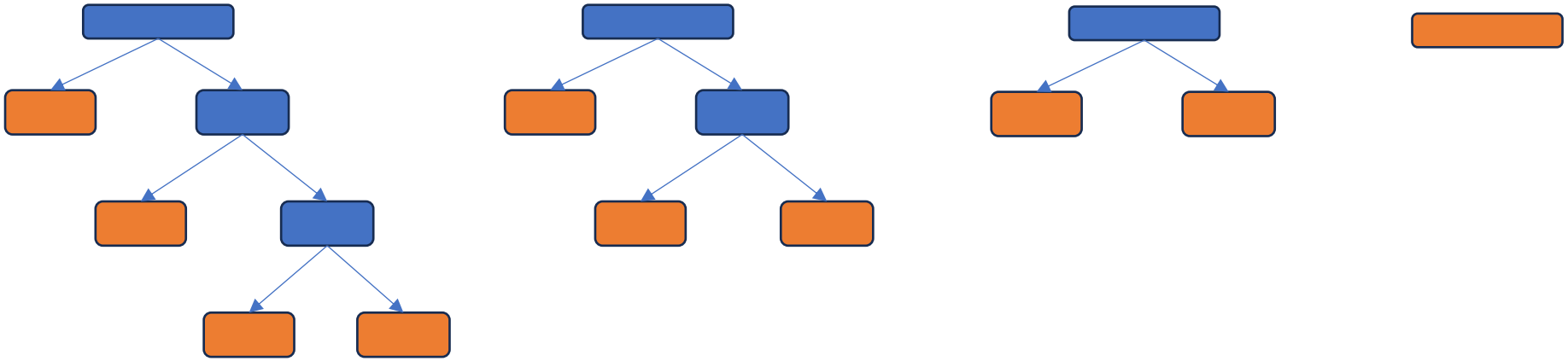
Donde T es el número de hojas y alfa es un parámetro de complejidad. La función de error puede ser el MSE.



Poda

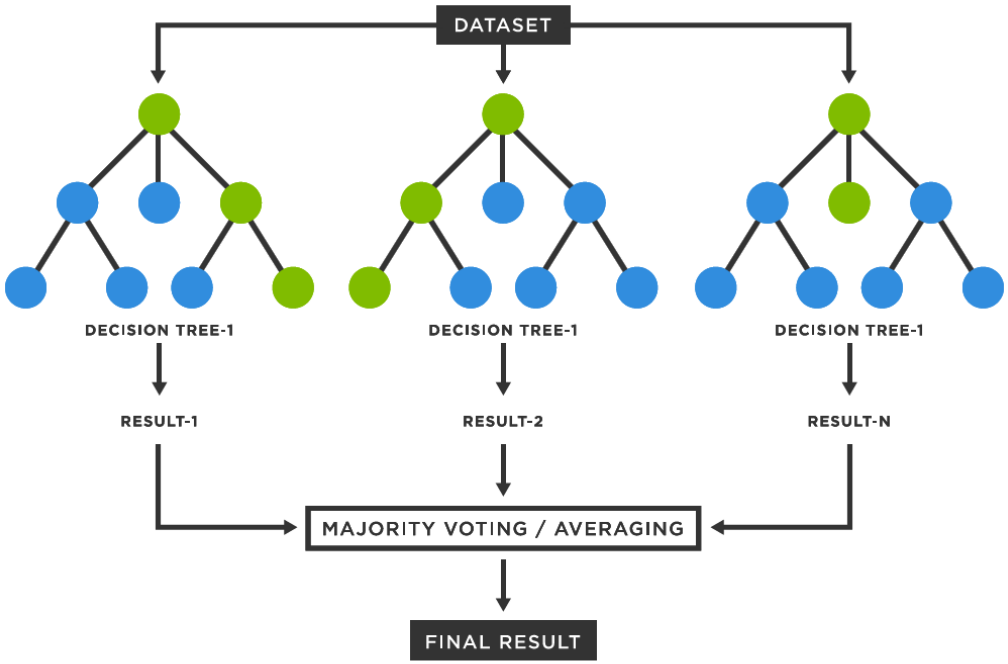
El valor óptimo de alfa puede determinarse primero encontrando un conjunto de valores que progresivamente minimicen el error en versiones cada vez más recortadas del árbol. Utilizando validación cruzada y evaluando sobre distintas versiones del árbol, se encuentra un alfa óptimo. Alfa varía entre 0 y 1, y a medida que crece, más nodos se "podan" resultando árboles más sencillos.

$$C_\alpha = Error + \alpha T$$



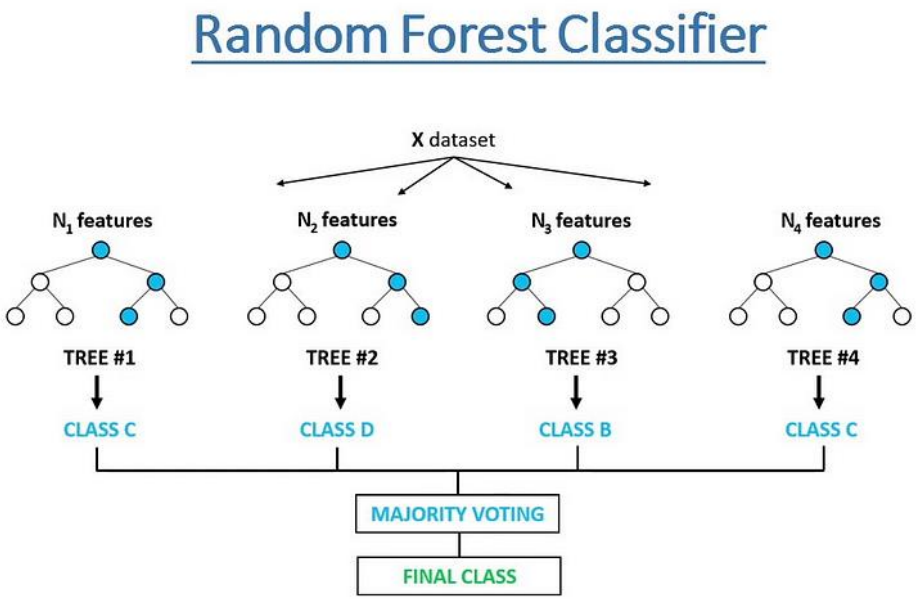
Random Forest

La mejor robustez y precisión en modelos basados en árboles de decisión se encuentra con los bosques aleatorios o **random forest**. Al agregar una medida de aleatoriedad, los árboles son más independientes y diversos. Es una técnica de ensamble, lo que significa que combina los resultados de varios modelos (en este caso, árboles de decisión) para obtener una predicción más precisa y confiable.



Random Forest

Se entrena un número determinado de AD, cada uno con subconjuntos aleatorios de los datos. El resultado final del algoritmo toma en cuenta los resultados de los distintos árboles y entonces promedia (regresión) o toma la moda (clasificación) de los mismos.



Random Forest

En la creación de cada árbol:

- Se toma una muestra aleatoria del dataset. Se permite que una muestra esté incluida varias veces y que algunas muestras no aparezcan en absoluto.
- Se crea el árbol teniendo en cuenta que, en cada paso de ramificación, se tienen en cuenta solo un numero determinado de features elegidos al azar. Se suele tomar dicho número con un valor cercano a la raíz cuadrada del número de features.

Se repite un número (por lo general 100) de veces, y en cada caso se obtuvieron árboles muy distintos.

Random Forest

Ventajas

- Mejor Generalización: Debido a la combinación de múltiples árboles, Random Forest tiende a generalizar mejor que un solo árbol de decisión, lo que reduce el riesgo de sobreajuste.
- Robustez: Random Forest es menos sensible a los ruidos y a las variables irrelevantes en el conjunto de datos.
- Versatilidad: Puede ser utilizado tanto para problemas de clasificación como de regresión.

Desventajas

- Entrenar múltiples árboles puede ser más costoso en términos de tiempo y recursos computacionales.
- Aunque los árboles de decisión individuales son fáciles de interpretar, un bosque de muchos árboles puede ser más difícil de analizar y entender.