# Árboles de Regresión y Clasificación

MÁSTER EN BIG DATA,

DATA SCIENCE E INTELIGENCIA ARTIFICIAL

# 1. Introducción.

- INTRODUCCIÓN.
- MEDIDAS Y PROCESO PARA LA CONSTRUCCIÓN DE UN ÁRBOL.
- CRITERIOS PARA  LA ELECCIÓN DE UN PUNTO DE CORTE ÓPTIMO.
- MANEJO DE VALORES PERDIDOS.
- Árbol final, subárboles y pruning.
- VENTAJAS Y DESVENTAJAS FRENTE A OTROS ALGORITMOS.

# 1. Introducción.

## ¿Qué es un ÁRBOL de decisión?

Cuando las personas debemos elegir entre varias opciones, “siempre” sopesamos los pros y los contras de cada una de ellas para llevar a cabo una decisión.

Los árboles de decisión llevan a cabo este paradigma a través de una estructura de árbol.

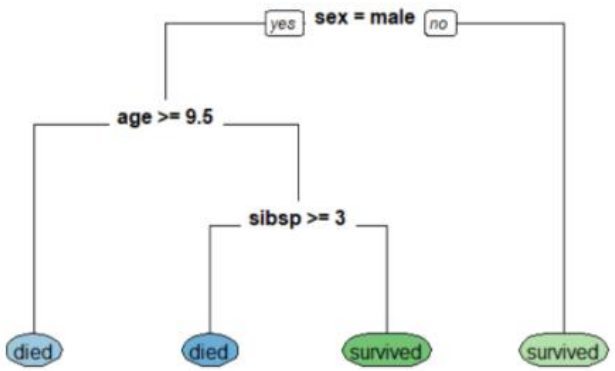
- **¿Qué son?** Métodos que proporcionan modelos que satisfacen objetivos tanto predictivos como explicativos.
- **¿Cuál es el objetivo?** Predecir una variable respuesta, *Y,* en función de un conjunto de variables independientes, $X_1,\ldots X_n$

# 1. Introducción.

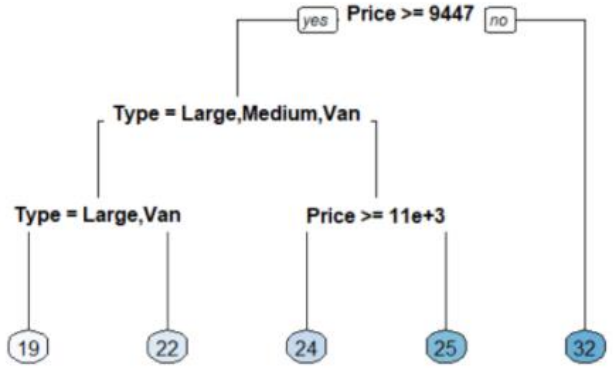
## ¿Qué es un ÁRBOL de decisión?

- **Tipos:**
  - **Árboles de clasificación** en los cuales la variable respuesta *Y* es cualitativa
  - **Árboles de regresión** en los cuales la variable respuesta *Y* es cuantitativa.

Árbol de clasificación




Árbol de regresión



# 1. Introducción.

## ¿Qué es un ÁRBOL de decisión?

- Un árbol de clasificación/regresión es el resultado de preguntar una secuencia ordenada de cuestiones. Las cuestiones que se plantean en cada etapa dependen de las respuestas a las cuestiones previas de la secuencia.

- Estas cuestiones se ordenan de más importante a menos.

- El punto único de inicio del árbol se llama ***nodo raíz*** y contiene el conjunto total de datos a clasificar en la parte superior del árbol.

- Un ***nodo*** es un subconjunto del conjunto de variables, y puede ser terminal o no terminal. Un ***nodo no terminal*** o ***padre*** es un nodo que se divide en *'k'* nodos descendientes.

# 1. Introducción.

**¿Qué es un ÁRBOL de decisión?**

- Un tipo muy generalizado de árbol son arboles de decisión binarios en los que cada *nodo padre* se divide en **dos *nodos descendientes***.

- Una división binaria queda determinada mediante una condición booleana sobre los valores de una única variable, siendo satisfecha la condición (‘si’) o no satisfecha (‘no’) por el valor observado de dicha variable.

# 1. Introducción.

**¿Qué es un ÁRBOL de decisión?**

- Todas las observaciones que alcanzan un nodo (padre) particular y satisfacen la condición para dicha variable van a parar a uno de los nodos descendientes; el resto de las observaciones de dicho nodo (padre) que no satisfacen la condición van a parar al otro nodo descendiente.
- Un nodo que no se divide se llama ***nodo terminal*** y se le asigna un valor específico (ya sea numérico o categórico).
- Cuando una observación de clase desconocida transita a través del árbol y va a parar a un nodo terminal, se le asigna la clase correspondiente a la etiqueta de clase adjunta a dicho nodo.
- Puede haber más de un nodo terminal con la misma etiqueta de clase.

# 1. Introducción.

**¿Qué es un ÁRBOL de decisión?**

- En cada nodo, el algoritmo de generación del árbol tiene que decidir sobre qué variable es ‘óptima’ la partición.

- **Hay que considerar cada posible división sobre todas las variables presentes en dicho nodo, enumerar después todas las posibles divisiones, evaluar cada una, y decidir cuál es la mejor siguiendo algún criterio**.

# 1. Introducción.

## ¿Qué es un ÁRBOL de decisión?

Debido a su estructura y planteamiento, este algoritmo es especialmente indicado cuando la **explicabilidad de un modelo es fundamental**.

Desde un nodo raíz, pasando por las ramas, hasta las hojas, se puede determinar los criterios de decisión empleados para llegar hasta una clase predicha.

Algunas de sus aplicaciones más extendidas son:

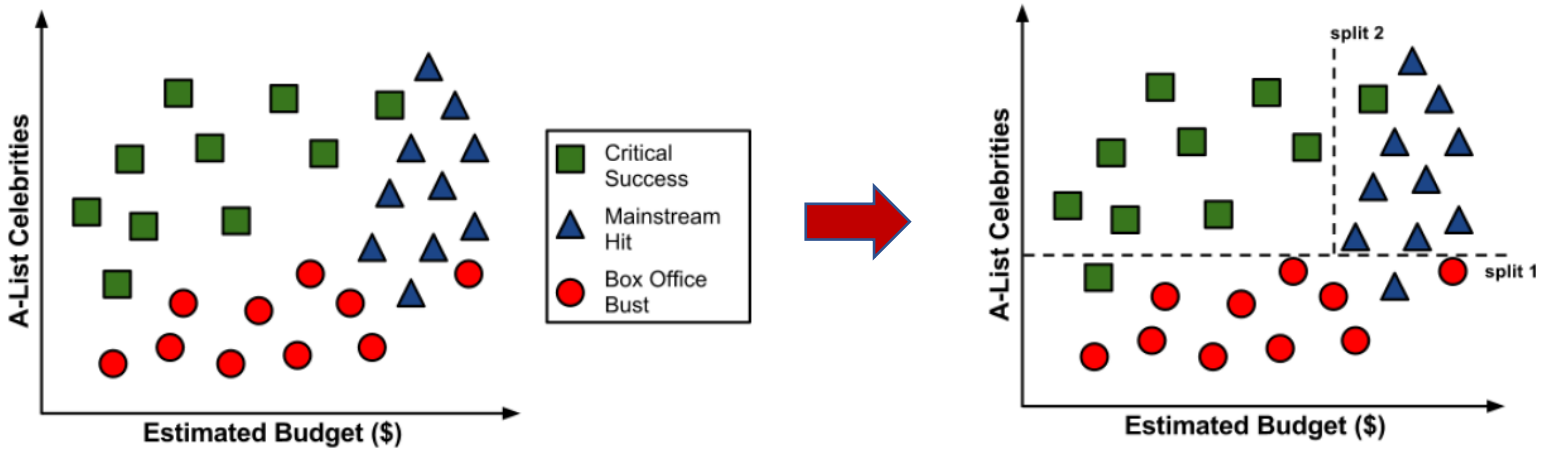
- Concesión de créditos para determinar, y por qué, si un determinado solicitante es rechazado o no.
- Estudios de marketing para conocer la satisfacción de los clientes.
- Diagnóstico médico.

# 1. Introducción.

## ¿Qué es un ÁRBOL de decisión?

Véase el ejemplo de Lantz (2013):

**DIVIDE Y VENCERÁS.**

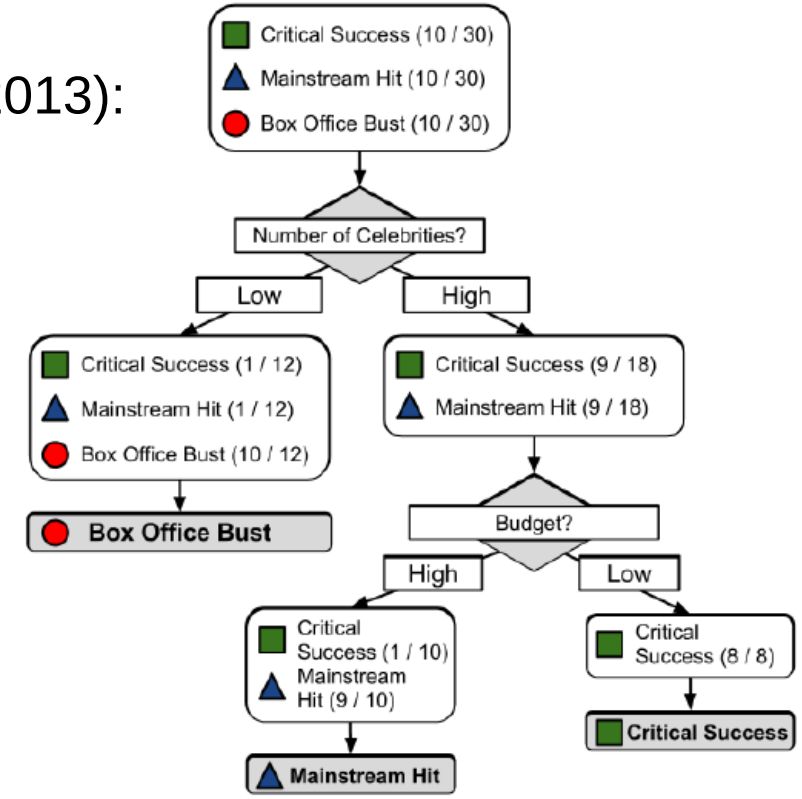


# 1. Introducción.

## ¿Qué es un ÁRBOL de decisión?

Véase el ejemplo de Lantz (2013):

**DIVIDE Y VENCERÁS.**



# 1. Introducción.

**¿Qué es un ÁRBOL de decisión?**

Existen algunos casos en los que estos modelos pueden **NO ser adecuados** a pesar de su capacidad explicativa.

¿Cuáles podrían ser?

# 1. Introducción.

**¿Qué es un ÁRBOL de decisión?**

Existen algunos casos en los que estos modelos pueden **NO ser adecuados** a pesar de su capacidad explicativa.

- Cuando se tiene una gran cantidad de variables nominales con muchos niveles.
- Gran cantidad de variables numéricas.

**El árbol presentaría MUCHAS ramas, resultando en un árbol muy complejo y difícil de interpretar.**

# 1. Introducción.

**¿Qué es un ÁRBOL de decisión?**

- Si *Y* es la **variable dependiente**, y $X_1,\ldots X_p$ son las variables fijas independientes o **predictoras**, se trata de resolver el problema estadístico de establecer una relación entre *Y* y $X_1,\ldots X_p$, de forma que sea posible **predecir el valor de *Y* en base a los valores de $X_1,\ldots X_p$.**

- Matemáticamente, se quiere estudiar la probabilidad condicional de la variable aleatoria *Y*, $P[Y = y|X_1 \ldots X_p]$, o una función de probabilidad tal como la esperanza condicional, $\mathrm{E}[Y|X_1 \ldots X_p]$, dependiendo de si se trata de un árbol de clasificación o de regresión, respectivamente.

# 1. Introducción.

**Criterios para la división de nodos.**

El objetivo es encontrar en los datos el punto de corte (y el criterio, o variable) que divida estos en dos grupos lo más “puros” posibles. Esto consiste en asegurar que un nodo padre produzca nodos hijos en cuya composición de clases sea lo más pura posible. **El objetivo es que cada nodo padre produzca un nodo hijo que únicamente contiene una clase.**

- Este grado de pureza se suele medir con alguno de los siguientes criterios:
  - Entropía.
  - Índice de Gini.
  - Mínima probabilidad.

# 1. Introducción.

**Criterios para la división de nodos.**

- Entropía.

La entropía es una medida de la incertidumbre o aleatoriedad en un conjunto de datos. En un nodo, se calcula la entropía de la distribución de las clases objetivo (etiquetas de los datos) presentes en ese nodo. Un valor de entropía más alto indica mayor mezcla o impureza en los datos.

# 1. Introducción.

**Criterios para la división de nodos.**

- Índice de Gini.

El índice de Gini es una medida de desigualdad, la cual ofrece un valor de homogeneidad/heterogeneidad de los datos. Los valores mínimos indican perfecta igualdad: Todos los valores son el mismo. Por su parte, valores altos indican que todos los datos son distintos entre sí.

# 1. Introducción.

**Criterios para la división de nodos.**

- ❖ Mínima Probabilidad.

- Fijar un umbral de probabilidad mínima para considerar una partición en el árbol. Es útil cuando se quiere reducir la impureza en base a la proporción de clases y no a su "mezcla". Por ejemplo, cuando hay clases desequilibradas y se quiere garantizar que las divisiones se realicen solo cuando la probabilidad de una clase en particular sea superior a un valor mínimo.

# 1. Introducción.

**Nodos terminales.**

El proceso de segmentación recursiva continúa hasta que el árbol se **satura**, en el sentido de que los sujetos en los **nodos descendientes no se pueden partir en una división adicional**.

El número total de **divisiones permitidas** para un nodo **disminuye** cuando **aumentan los niveles del** árbol. Cualquier nodo que no pueda o no sea divido es un nodo terminal.

# 1. Introducción.

## Ejemplo del proceso de construcción de un árbol.

Se considera un ejemplo cuya variable respuesta busca determinar si la persona tiene una enfermedad coronaria (*chd=1 SI, chd=0 NO*), a partir de un conjunto de variables independientes.

❖**PROBLEMA MÁS HABITUAL**: variable de interés dicotómicas, con variables numéricas (*obesity* y *tobacco*) y categóricas para predecirla (*famhist*).

| y | x1 | x2 | A | … |
|---|---|---|---|---|
| Chd | Obesity | Tobacco | Famhist | … |
| 1 | 25.3 | 12 | Present | … |
| 1 | 28.8 | 0.01 | Absent | … |
| 0 | 29.14 | 0.08 | Present | … |
| … | … | … | … | … |

**DISCRETIZACIÓN DE VARIABLES CONTINUAS**: los árboles tratan de hallar **puntos de corte** en las variables independientes que lleven a grupos de individuos con comportamiento homogéneo (% de 1's, % de 0's) respecto a la variable respuesta y diferente entre los grupos.

# 1. Introducción.

## Ejemplo del proceso de construcción de un árbol.

- De forma genérica, se habla de árboles **binarios** si estos presentan dos ramas: en cada nodo hay que elegir hacia qué lado se bifurca. Para ello se establece el **punto de corte:** hasta un valor del punto de corte se bifurca a un lado, y desde dicho punto de corte, hacia el otro.

- **¿Cómo se eligen los puntos de corte?**

  **VARIABLE CUALITATIVA:** categorías de la variable. En cada categoría se analiza si tiene que ir hacia un lado (0's) u otro (1's).

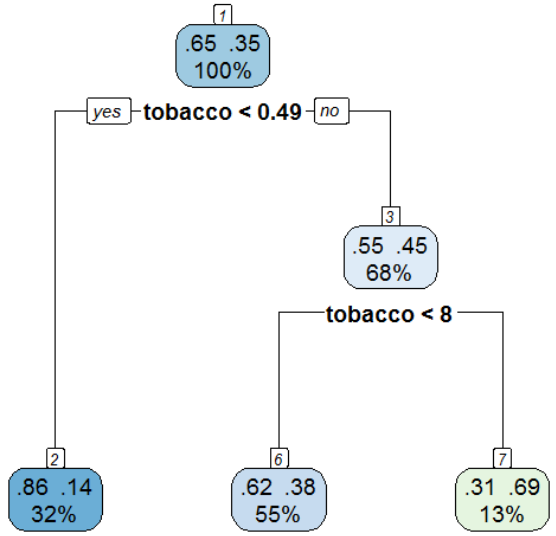
  **VARIABLE CONTINUA:** se van probando distintos cortes (normalmente por deciles de la variable), y en cada uno de estos puntos decide si el corte es bueno o no.

- Los árboles actúan en cascada, tomando **una decisión para cada variable.**

# 1. Introducción.

**Ejemplo del proceso de construcción de un árbol.**

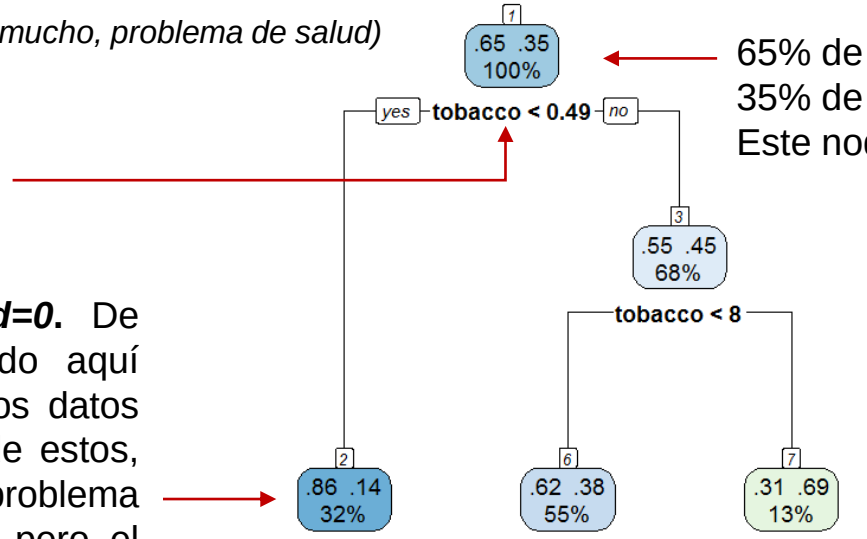
**Tratar de interpretar**



# 1. Introducción.

## Ejemplo del proceso de construcción de un árbol.

- **Tobacco<0.49** (mal clasificado 1: 14% , acierto para 0: 86%), contiene el 32% de todas las observaciones, el 86% bien clasificadas.
- *(nodo propenso a chd=0,fuma poco, no problema de salud)*
- **0.49<=Tobacco<8** (mal clasificado 1: 38% , acierto para 0: 62%), contiene el 55% de todas las observaciones
- **Tobacco>=8** (acierto para 1: 69% , 0: 31%), contiene el 13% de todas las observaciones
- *(nodo más propenso a chd=1, fuma mucho, problema de salud)*



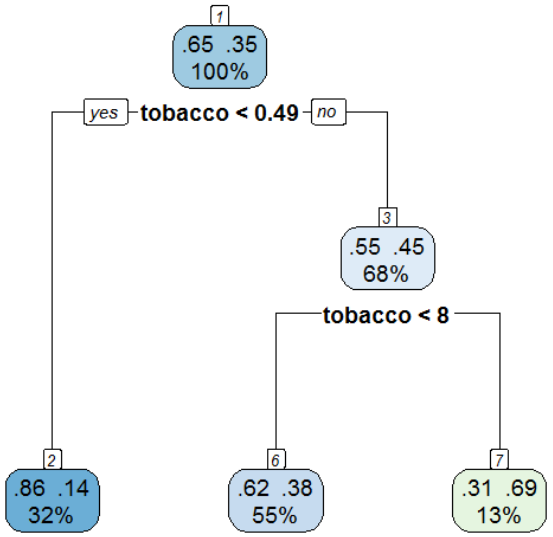

65% de la población clasificada en un grupo *(chd=no);*
35% de la población clasificada en otro (*chd=yes)*
Este nodo contiene el 100% de las observaciones

Si fuma poco, se considera que no tiene problema de salud

Este nodo clasifica ***chd=0*.** De todos los que he metido aquí (32% de la población), los datos reales son que el 86% de estos, en efecto no han tenido problema (bien clasificado *chd=0*); pero el 14% sí ha tenido (mal clasificados, debería ser *chd=1*).

# 1. Introducción.

**Ejemplo del proceso de construcción de un árbol.**




**Cuando los costes de equivocarme en un sentido u otro de la bifurcación son iguales, se clasifica en la categoría mayoritaria.**

Ejemplo: si es 5 veces más grave equivocarme al clasificar a una persona como sana cuando no lo está que al revés, hay que ponderar el coste de equivocarse en la clasificación.

Cuando se hace un modelo de árboles por una parte, se obtiene el *scoring* y por otro la clasificación, con lo que se puede jugar.

Los cortes pueden ser usando varias veces la misma variable (distintos cortes), o intercalando variables distintas.

Además, no se pierde generalidad por considerar árboles binarios en cualquier situación.

# 1. Introducción.

**Ejemplo del proceso de construcción de un árbol.**

❖El árbol se comporta como un **análisis cluster** (creación de grupos homogéneos y diferentes entre sí) y a la vez como un **método predictivo** (a los individuos con tobacco>8 les asignamos *chd=1* ya los demás *chd=0*).

❖Por ello este tipo de técnicas cuando se orientan a crear grupos se denominan **técnicas de segmentación**, siendo utilizados cuando se quiere crear grupos poblacionales con respecto a una variable dependiente.

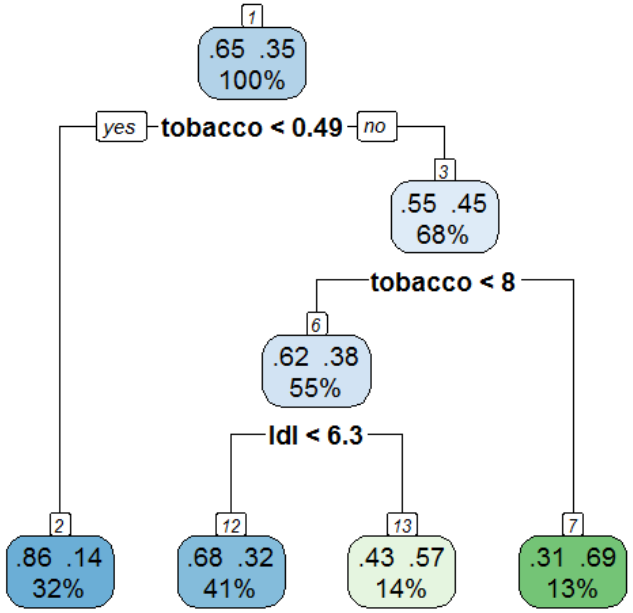
# 1. Introducción.

**Ejemplo del proceso de construcción de un árbol.**

Si se añade la variable idl (Low density lipoprotein cholesterol), se puede observar cómo afecta al árbol:

**IMPORTANTE**: elegir la profundidad del árbol. A menos profundidad menor capacidad predictiva, a mayor profundidad, más riesgo de sobreajuste. En este ejemplo, incluir la variable *Idl* ayuda a clasificar mejor los casos “dudosos”.

Véase como el grupo que acuñaba el 55% de la muestra ahora se ha dividido en dos más.



# 2. Medidas y proceso para la construcción de un árbol.

El proceso de creación del árbol es laborioso por la **complicada casuística** y **combinatoria** que se va generando cada vez que es necesario dividir un nodo.

Históricamente la construcción de árboles ha ido mejorando, creándose nuevos algoritmos:

- **CRT** o **CART** (Classification and Regression Trees)
- **CHAID** (Chi square Automatic Interaction Detector)
- **ID3** (Interactive Dichotomizer)
- **C4.5**
- etc.

❖ Estos algoritmos aportan algunos trucos y opciones para solucionar los problemas de árboles.

❖ Las dos técnicas más usadas y contrastadas son CART y CHAID. En cualquier cosa, hay muy poca diferencia entre unas y otras. Dependiendo del paquete se usa una u otra.

# 2. Medidas y proceso para la construcción de un árbol.

❖**CART (Classification and Regression Trees).**

Esta metodología consiste en construir un **árbol grande** e ir **podando** (quedarse con un subárbol) hasta dar con el tamaño correcto. Consta de tres pasos:

1. Construcción del árbol saturado (hasta que no se “puede” hacer más divisiones).
2. Elección del tamaño correcto.
3. Clasificación de nuevos datos a partir del árbol construido.

# 2. Medidas y proceso para la construcción de un árbol.

❖**CHAID.**

Explora los datos de forma rápida y eficaz. Permite la detección automática de interacciones mediante ji-cuadrado.

- CHAID examina las tablas cruzadas entre los campos de entrada y los resultados para, a continuación, comprobar la significación mediante una comprobación de independencia de chi-cuadrado.
- Si varias de estas relaciones son estadísticamente importantes, CHAID seleccionará el campo de entrada con el valor de significación menor.
- Si una entrada cuenta con más de dos categorías, se compararán estas categorías y se contraen las que no presenten diferencias en los resultados, uniéndose el par de categorías con menor diferencia, y así sucesivamente.
- Este proceso se detiene cuando todas las categorías restantes difieren entre sí en el nivel de comprobación especificado. En el caso de campos de entrada nominales, pueden fundirse todas las categorías. En los conjuntos ordinales, únicamente podrán fundirse las categorías contiguas.

# 2. Medidas y proceso para la construcción de un árbol.

❖**ID3.**

La idea básica del ID3 es determinar, para un conjunto de ejemplos dados, el atributo con mayor poder discriminatorio. **Se basa en la medición de la entropía.**

Después de haber hecho la primera prueba de atributo, el nuevo resultado se convierte en el siguiente problema de aprendizaje de árbol de decisión, con la diferencia que cuenta con menos ejemplos y un atributo menos, hasta finalizar con todos los atributos.

Se dan algunos casos especiales cuando:

- ❖ Existen objetos con clases diferentes: se selecciona el que tenga la mayor información.
- ❖ Los objetos restantes son todos de una sola clase, entonces los objetos se clasifican con esa clase.
- ❖ Si no quedan atributos, pero sí objetos con sus respectivas clases, hay ambigüedad en los datos. También sucede cuando los atributos no proporcionan la suficiente información para describir completamente la situación.

# 2. Medidas y proceso para la construcción de un árbol.

- **C4.5.**

El algoritmo considera todas las pruebas posibles que pueden dividir el conjunto de datos y **selecciona la prueba que proporciona la mayor ganancia de información**. Para cada atributo discreto, se considera una prueba con *n* resultados, siendo *n* el número de valores posibles que puede tomar el atributo.

- En cada nodo, el sistema debe decidir cuál prueba escoge para dividir los datos.
- La prueba "estándar" para las variables discretas, ofrece un resultado y una rama para cada valor posible de la variable.
- Una prueba más compleja, basada en una variable discreta, en donde los valores posibles son asignados a un número variable de grupos con un resultado posible para cada grupo, en lugar de para cada valor.
- Si una variable A tiene valores numéricos continuos, se realiza una prueba binaria con resultados A <= Z y A > Z, para lo cual debe determinarse el valor límite Z.

# 2. Medidas y proceso para la construcción de un árbol.

❖**C4.5.**

Todas estas pruebas se evalúan de la misma manera, mirando el resultado de la proporción de ganancia, o alternativamente, el de la ganancia resultante de la división que producen.

Es útil agregar una restricción adicional: para cualquier división, al menos dos de los subconjuntos $C_i$ deben contener un número razonable de casos. Esta restricción, que evita las subdivisiones casi triviales, es tenida en cuenta solamente cuando el conjunto C es pequeño.

# 2. Medidas y proceso para la construcción de un árbol.

❖**Para una correcta construcción de árboles, es importante tener en cuenta diversos criterios, entre los que caben destacar:**

- Criterios para elegir qué variable independiente y nodo (si el árbol ya está avanzado) va a ser la base para la siguiente división.
- Criterios para elegir el(los) punto(s) de corte óptimo dentro de cada variable independiente o nodo.
- Criterios para elegir grupos de corte para variables independientes nominales con más de una categoría.
- Número de observaciones mínimas para construir un nodo.

# 2. Medidas y proceso para la construcción de un árbol.

❖**Para una correcta construcción de árboles, es importante tener en cuenta diversos criterios, entre los que caben destacar:**

- Criterios de parada-fin del algoritmo.
- Límites al número de nodos, divisiones, etc.
- Criterios de tratamiento de valores perdidos.
- Si se van a utilizar datos de validación para controlar el proceso de construcción o no.

# 3. Criterios para la elección de un punto de corte óptimo.

Por simplicidad, se considerará el *Binary Split* en cada paso (aunque la misma metodología se extiende a > 2.

Dependiendo de la naturaleza de las variables dependiente/independientes, hay varios criterios para elegir el punto de corte.

- Ji-Cuadrado.
- Índice de Gini.
- Entropía.
- F-Snedecor.
- Max. Varianza.

# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Categórica; VI: Categórica/ordinal.

## ❖Ji-Cuadrado.

Se analiza la independencia de una tabla de frecuencias cruzando la VD con la VI, seleccionando el caso que presente un mayor nivel de significación.

Valores altos indican que la tabla de frecuencias es poco independiente, poco aleatoria: el corte planteado crea **grupos muy claros y poco aleatorios**.

$$\chi^2 = \sum_{i=1}^{m}\sum_{j=1}^{k} \frac{(n_{ij} - \frac{n_i n_j}{N})^2}{\frac{n_i n_j}{N}}$$

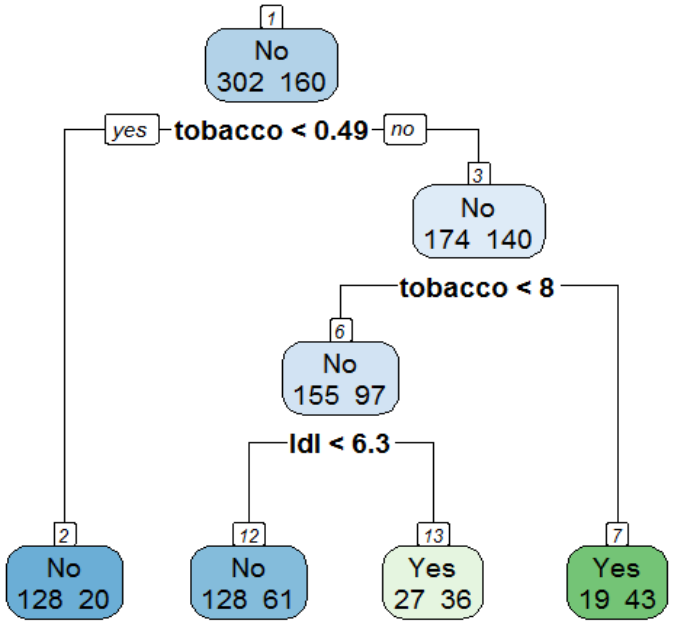
# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Categórica; VI: Categórica/ordinal.

## ❖Ji-Cuadrado.

- $n_{ij}$ número de individuos que han caído tras hacer la clasificación con un corte concreto
- $n_i$, $n_j$ son las contribuciones marginales de la tabla
- N número total de datos

$$\chi^2 = \sum_{i=1}^{m}\sum_{j=1}^{k} \frac{(n_{ij} - \frac{n_i n_j}{N})^2}{\frac{n_i n_j}{N}}$$




Marginal $n_{12}$

$n_1$

| | chd=0 | chd=1 | |
|---|---|---|---|
| Nodo 2 | 128 | 20 | 148 |
| Nodo 3 | 174 | 140 | 314 |
| | 302 | 160 | **462** |

$n_2$

**PLANTEAR LA PRIMERA Y ÚLTIMA ITERACIÓN**

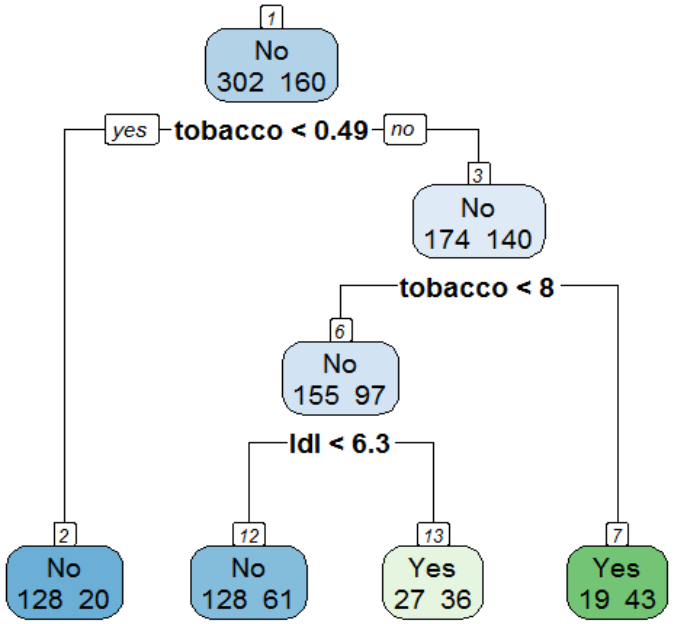
# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Categórica; VI: Categórica/ordinal.

## ❖Ji-Cuadrado.

- $n_{ij}$ número de individuos que han caído tras hacer la clasificación con un corte concreto
- $n_i$, $n_j$ son las contribuciones marginales de la tabla
- N número total de datos

$$\chi^2 = \sum_{i=1}^{m}\sum_{j=1}^{k} \frac{(n_{ij} - \frac{n_i n_j}{N})^2}{\frac{n_i n_j}{N}}$$




Marginal $n_{12}$

$n_1$

| | chd=0 | chd=1 | |
|---|---|---|---|
| Nodo 2 | 128 | 20 | 148 |
| Nodo 3 | 174 | 140 | 314 |
| | 302 | 160 | **462** |

$n_2$

$$\chi^2 = \sum_{i=1}^{m}\sum_{j=1}^{k} \frac{\left(128 - \frac{(148 * 302)}{462}\right)^2}{\frac{(148 * 302)}{462}}, \ldots, \frac{\left(140 - \frac{(314 * 160)}{462}\right)^2}{\frac{(314 * 160)}{462}}$$

# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Categórica; VI: Categórica/ordinal.

## ❖Ji-Cuadrado.

- Lo ideal es escoger el corte que **MAXIMICE** el valor de $\chi^2$
- **Proceso**: para todas las variables y todos los puntos de corte, calcular $\chi^2$ para obtener el máximo
- **PROBLEMA**: dependiendo de las variables que haya y de la complejidad de los datos, este proceso puede tener demasiado coste computacional.
- **IMPORTANTE**: el test de Chaid usando el criterio de $\chi^2$ se basa en la hipótesis de normalidad. La mayoría de los datos reales NO están distribuidos normalmente, lo que dificulta obtener resultados coherentes con este método.

Una alternativa puede ser el Índice de Gini.

# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Categórica; VI: Categórica/ordinal.

## ❖Índice de Gini.

El índice de Gini es una medida de desigualdad que refleja el grado que de homogeneidad o heterogeneidad de valores que presenta un conjunto de datos. Puede reflejar la "impureza" de valores.

$$I(Gini) = 1 - \sum_{i=1}^{k} \left(\frac{n_i}{n}\right)^2$$

Los valores mínimos indican perfecta igualdad: Todos los valores son el mismo. Por su parte, valores altos indican que todos los datos son distintos entre sí.

$$Max\left(I(Gini_{nodo\ padre}) - \sum_{b} I(Gini_{ramas})p(b)\right)$$

Probabilidad en cada uno de los nodos hijos

# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Categórica; VI: Categórica/ordinal.

## ❖Índice de Gini.

**Gini nodo 1 (padre):**

$$I(Gini) = 1 - \sum_{i=1}^{k} \left(\frac{n_i}{n}\right)^2$$

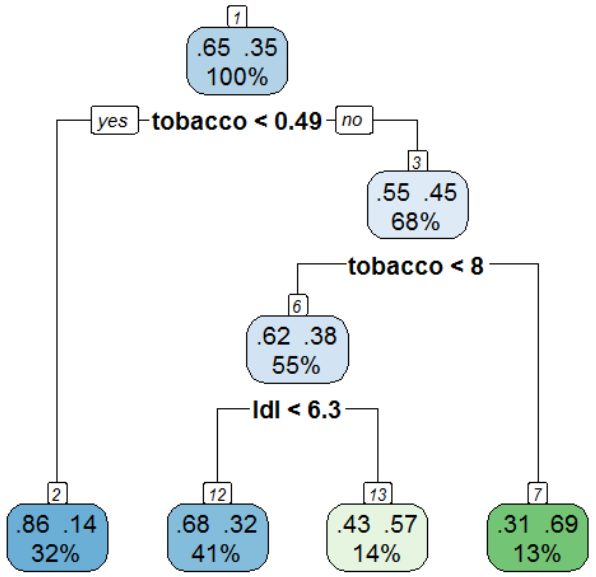
$$Max\left(I\left(Gini_{nodo\ padre}\right) - \sum_{b} I(Gini_{ramas})p(b)\right)$$

**Gini nodo 2:**

**Gini nodo 3:**

**Cálculo de la mejora:**

(cuanto más bajo Gini mejor):



# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Categórica; VI: Categórica/ordinal.

## ❖Índice de Gini.

**Gini nodo 1 (padre):**

$1 - (0{,}65^2 + 0{,}35^2) = 0.455.$

**Gini nodo 2:**

$1 - (0{,}86^2 + 0{,}14^2) = 0.2408.$

**Gini nodo 3:**

$1 - (0{,}55^2 + 0{,}45^2) = 0.495.$

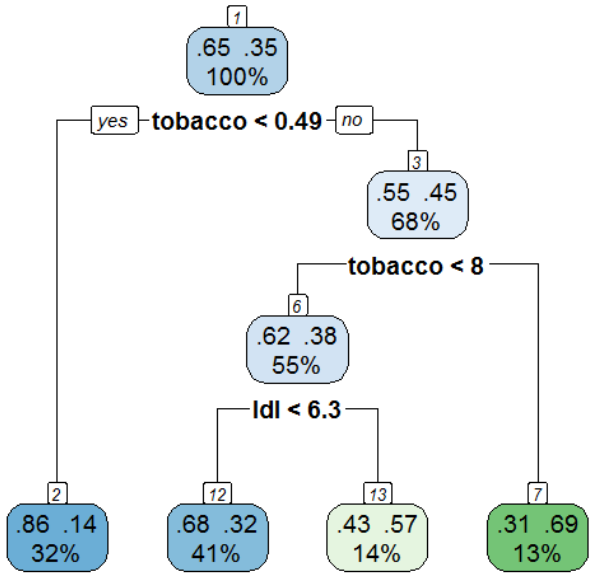
**Cálculo de la mejora:**

(cuanto más bajo Gini mejor):

$0{,}455 - (0{,}2408 * 0{,}32 + 0{,}495 * 0{,}68) = 0{,}041344.$

$$I(Gini) = 1 - \sum_{i=1}^{k} \left(\frac{n_i}{n}\right)^2$$

$$Max\left(I\left(Gini_{nodo\ padre}\right) - \sum_{b} I(Gini_{ramas})p(b)\right)$$



# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Categórica; VI: Categórica/ordinal.

## ❖Entropía.

Consiste en asegurar que un nodo padre produzca nodos hijos en cuya composición de clases sea lo más pura posible. **El objetivo es que cada nodo padre produzca un nodo hijo que únicamente contiene una clase.**

La medición de **la entropía refleja la impureza** de la distribución de clases, o cómo de mezcladas están las clases.

Un **valor de 0** indica el nivel **máximo de homogeneidad**.

Un **valor de 1** indica el **máximo nivel de heterogeneidad**.

# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Categórica; VI: Categórica/ordinal.

## ❖Entropía.

$$Entropía(S_i) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

Donde $S_i$ es un subconjunto de datos, $c$ indica el número de clases diferentes y $p$ la proporción de valores agrupados en la clase de nivel $i$.

Se selecciona la **división que mejora el índice de entropía** respecto de la entropía base del nodo padre:

$$Max\left(Entropía(Nodo\ padre) - \sum_{b} Entropía(S_i)p(b)\right)$$

# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Categórica; VI: Categórica/ordinal.

$$Entropía(S_i) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

## ❖Entropía.

$$Max\left(Entropía(Nodo\ padre) - \sum_{b} Entropía(S_i)p(b)\right)$$

**Entropía nodo 1 (padre):**

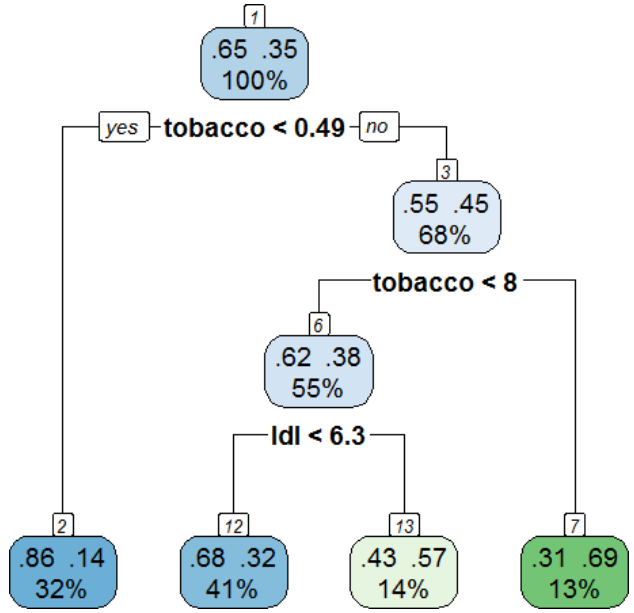
**Entropía nodo 2:**

**Entropía nodo 3:**

**Cálculo de la mejora :**

(cuanto más baja la entropía mejor):

La entropía padre es 0.072 mayor que realizando la división, por lo tanto “compensa” realizarla.



# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Categórica; VI: Categórica/ordinal.

$$Entropía(S_i) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

$$Max\left(Entropía(Nodo\ padre) - \sum_{b} Entropía(S_i)p(b)\right)$$

## ❖Entropía.

**Entropía nodo 1 (padre):**

$-0{,}65 * \log_2(0{,}65) - 0{,}35 * \log_2(0{,}35) = 0{,}934$**.**

**Entropía nodo 2:**

$-0{,}86 * \log_2(0{,}86) - 0{,}14 - \log_2(0{,}14) = 0{,}5842$**.**

**Entropía nodo 3:**

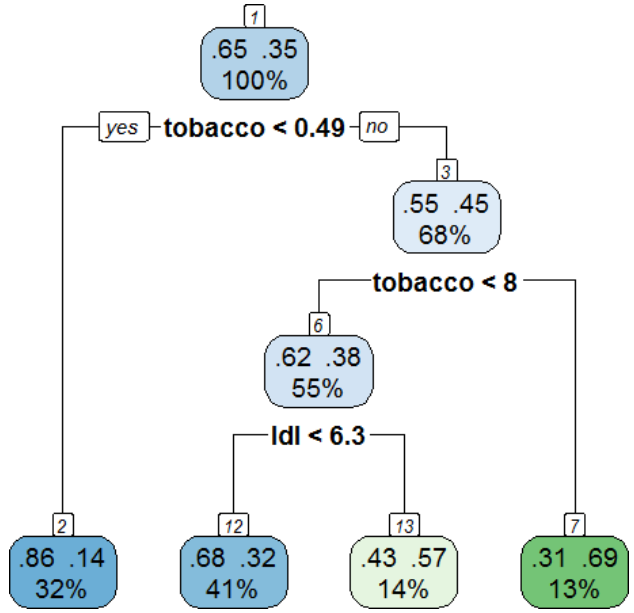
$0{,}55 * \log_2(0{,}55) - 0{,}45 - \log_2(0{,}45) = 0{,}992$**.**

**Cálculo de la mejora :**

(cuanto más baja la entropía mejor):

$0{,}934 - (0{,}5842 * 0{,}32 + 0{,}992 * 0{,}68) = 0{,}072$**.**

La entropía padre es 0.072 mayor que realizando la división, por lo tanto “compensa” realizarla.



# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Contínua; VI: Categórica/ordinal.

## ❖F de Snedecor.

Es el estadístico de contraste utilizado en ANOVA.

Se asume que, si los elementos de los distintos grupos pertenecen a la misma población, la varianza intragrupal debe de ser la misma que la varianza intergrupal. Así, se compara cómo de grande es la varianza entre grupos en comparación con la varianza intragrupal.

**Estimador de la varianza intragrupal** se construye como un promedio de las distancias medias elevadas al cuadrado. Donde se tienen *J* grupos, y $S_j^2$ es la varianza muestral de un grupo *J*, se hace una media ponderada de las *J* varianzas muestrales:

$$MC_E = \frac{\sum(n_j - 1)S_j^2}{N - J}$$

# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Contínua; VI: Categórica/ordinal.

## ❖F de Snedecor.

Así, si se asume que, además de la misma varianza, deben de tener las mismas medias, se puede partir de la idea de que todas las muestras pertenecen a la misma población, calculando la **Media Cuadrática Intergrupos**:

$$MC_A = \frac{\sum n_j \left(\bar{Y}_J - \bar{Y}\right)^2}{J-1}$$

Premisa: si $MC_E$ y $MC_A$ será iguales o muy parecidas si se han calculado con muestras pertenecientes a la misma población y con medias muy similares. Por el contrario, si las muestras presentan medias muy distintas, ambas medias cuadráticas serán muy distintas.

# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Contínua; VI: Categórica/ordinal.

## ❖F de Snedecor.

**El estadístico F** se calcula, con J-1 y N-1 grados de libertad, como:

$$F = \frac{MC_A}{MC_E}$$

$$MC_E = \frac{\sum(n_j - 1)S_J^2}{N-J}; \; MC_A = \frac{\sum n_j\left(\bar{Y}_J - \bar{Y}\right)^2}{J-1}$$

Se seleccionará aquél punto de corte que maximice el nivel de significación asociado a F.

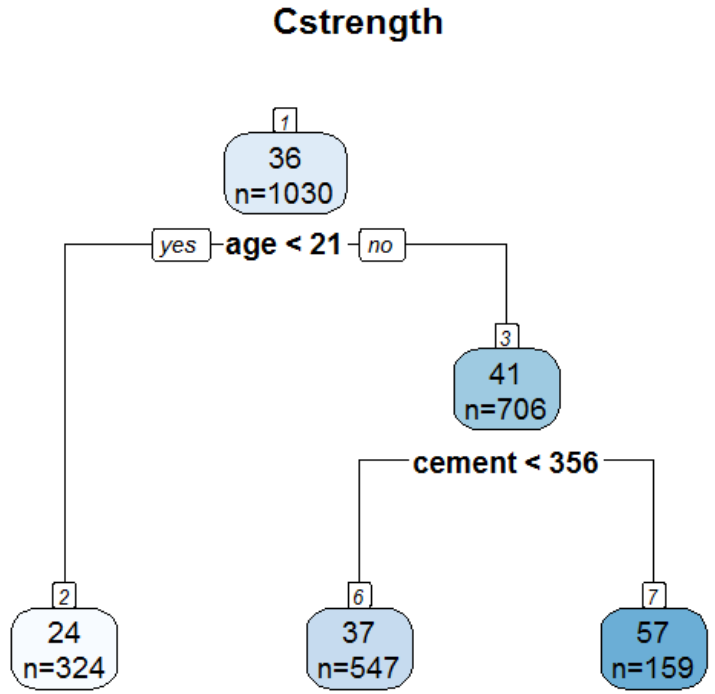
# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Contínua; VI: Categórica/ordinal.

## ❖F de Snedecor.

$$F = \frac{\frac{\sum n_j \left(\bar{Y}_J - \bar{Y}\right)^2}{J-1}}{\frac{\sum (n_j - 1) S_J^2}{N-J}}$$

Numerador F = $(24-36)^2*324+(41-36)^2*706=71243$
Denominador F = 210 (para calcularlo se calcula la varianza de y en cada nodo 2 y 3)
F=71243/210=**339**



# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Contínua; VI: Categórica/ordinal.

## ❖Varianza.

Se calcula la variabilidad de la variable dependiente en cada grupo y se suma (literalmente la varianza estadística sería dividiendo por n, pero no se hace así) la diferencia entre los datos con la media, y se busca MAXIMIZAR.

$$Varianza(nodo) \approx \sum_{j=1}^{n_{nodo}} \left( y_j - \bar{y}_{(nodo)} \right)^2$$

Se elige la división que construye grupos más **homogéneos internamente** y **diferentes entre sí**.

$Max(Var(nodo\ padre) - \sum_b Var(rama_b))$

SST=varianza nodo padre=287091

SSnodo2=varianza nodo 2=49896

# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Contínua; VI: Categórica/ordinal.

## ❖Varianza.

$$Max(Var(nodo\ padre) - \sum_b Var(rama_b))$$

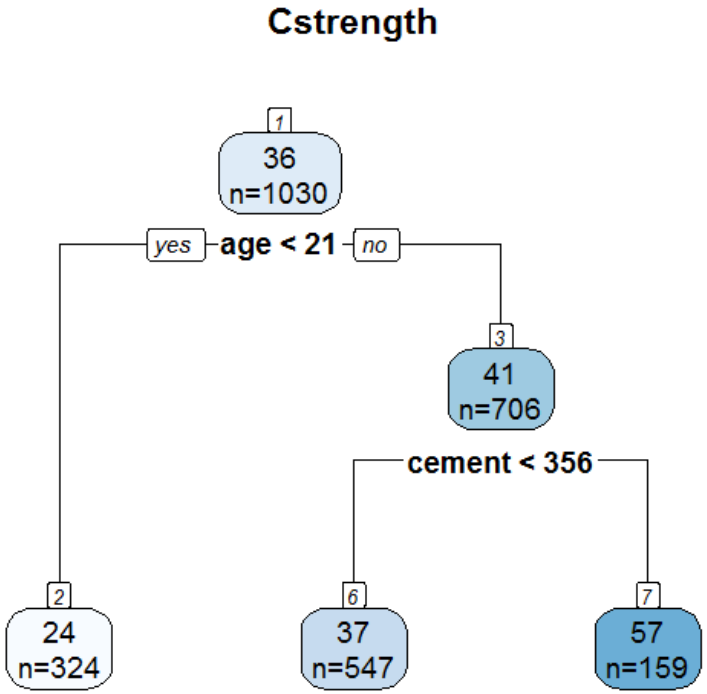
$$Varianza(nodo) \approx \sum_{j=1}^{n_{nodo}} \left(y_j - \bar{y}_{(nodo)}\right)^2$$

SST=varianza nodo padre=287091

SSnodo2=varianza nodo 2=49896

SSnodo3=varianza nodo 3=165910

SST-SSnodo2-SSnodo3=71285



# 3. Criterios para la elección de un punto de corte óptimo.

Caso: VD: Contínua; VI: Continua.

Los nodos continuos **han de ser tratados como categóricos para realizar la división**, por lo tanto el punto de corte se buscará entre un conjunto de puntos de corte candidatos seleccionados a través de métodos iterativos (en ocasiones, deciles) y se calcularán las mismas medidas Prob.F o Varianza vistas anteriormente, donde la variable independiente continua, dividida en dos grupos, hace el papel de categórica.

Caso: VD: Categórica; VI: Continua.

Se categoriza la variable continua (haciendo cortes) por métodos iterativos y se utilizan los criterios Chi cuadrado, Gini o Entropía.

# 4. Criterios para seleccionar la V.I. que va a crear la siguiente división.

- **IMPORTANTE**: no todas las variables de la base de datos son útiles para hacer modelos de machine learning.

- **OJO:** dependiendo del modelo, se usan unas u otras variables

- Se deben eliminar las variables de la base de datos que no se puedan tener en el momento de realizar las predicciones.

- Se pueden eliminar las variables de la base de datos que no tengan relacción con la variable independiente. (altura en metros para altura en cm, o IMC).

- No es necesario eliminar las variables de la base de datos que tengan multicolinealidad entre ellas.

# 5. Criterios para el manejo de valores perdidos.

❖Hay varias posibilidades:

a) Asignar las observaciones con **missing a la rama más grande.**

b) Asignar los missings a la rama con **menor error calculado en las observaciones sin missings**

c) Imputar los valores perdidos.

❖**IMPORTANTE**: el tratamiendo de missings puede enturbiar la información disponible y desequilibrarla. Es imprescindible conocer bien la base de datos para dar un tratamiento correcto. A veces, es conveniente usar los valores missing como un valor más de una variable categórica.

# 5. Criterios para el manejo de valores perdidos.

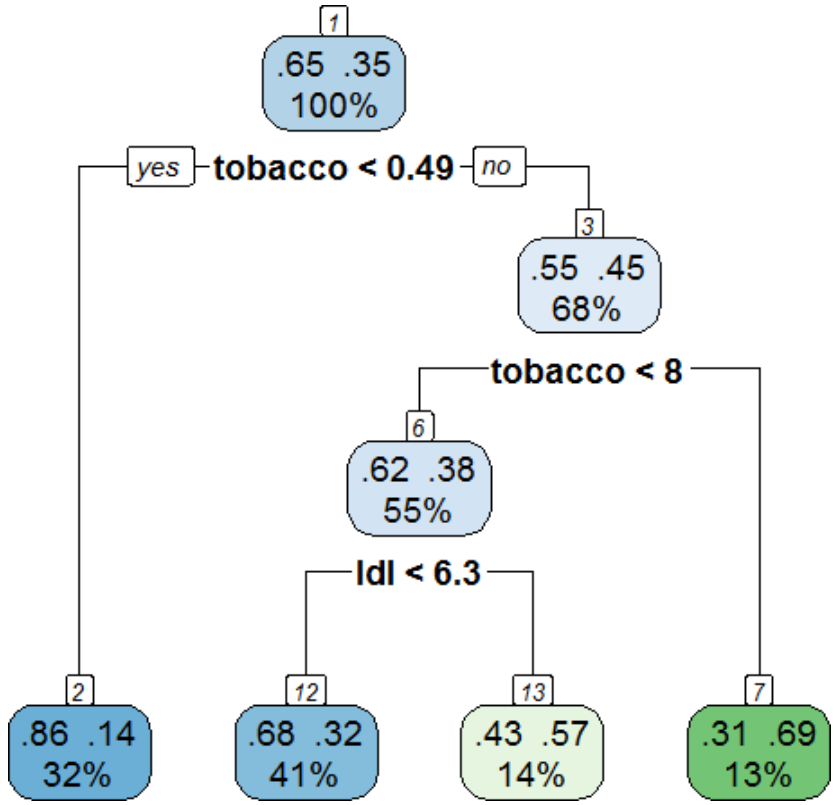
❖ Supongamos que en la primera división del gráfico (nodos 2 y 3) hay una observación missing en tobacco.

a) La observación iría al nodo 3, que es el más numeroso (contiene el 68% de la población total)

b) La observación probablemente iría al nodo 2 (es el que menos falla, 14%)

c) Se considerarían los nodos 2 y 3 como una variable binaria *A* dependiente.

Se buscaría la mejor variable (diferente de Tobacco), llamada “surrogate” para predecir esa variable *A* y se aplicaría la predicción con un nodo simple binario con esa variable. Se predeciría a qué nodo (2 o 3) iría la observación missing, y a ese nodo se asignaría.



# 6. Árbol final, subárboles y pruning.

❖El algoritmo finaliza cuando se cumple **alguno** de los criterios de parada:

- **No hay suficientes observaciones** en las hojas finales para considerar su división
- La **profundidad máxima** (parámetro prefijado) ha sido alcanzada
- **En ningún nodo se puede mejorar** el criterio de división, por ejemplo índice de Gini (o F para variables dependientes continuas)

El modelo final de árbol puede estar sobreajustado si los datos son complejos. Tras obtener el árbol final, puede actuarse como en los métodos cluster: escoger la solución=subárbol que parezca más estable. Pruning significa podar y es el acto de quedarse con un subárbol.

# 6. Árbol final, subárboles y pruning.

## ❖ PODA o prunning:

Un árbol de decisión puede crecer indefinidamente en particiones más y más pequeñas hasta encontrar la solución perfecta.

Las soluciones serían tan específicas que resultaría en un modelo con *overfitting*.

**El proceso de poda** asegura la generalización de los resultados a través de la reducción del tamaño del árbol.

- ❖ Pre-poda: Determinar un número específico de decisiones o bien establecer un mínimo de casos por nodo.
- ❖ Post-poda: Permitir el alto crecimiento de un árbol para después podar de acuerdo a la reducción de los ratios de error encontrados.

# 6. Árbol final, subárboles y pruning.

## ❖ PODA o prunning con CART:

- Hay que usar el parámetro de control de la complejidad *cp*, que establece "penalizaciones" si se producen muchas divisiones. El valor predeterminado de *cp* es 0.01. Cuanto **más alto** sea el valor de ***cp***, **más pequeño** será el **árbol**.

- Un valor de *cp* demasiado pequeño provoca sobreajuste, y un valor de *cp* demasiado grande resultará en un árbol demasiado pequeño. **Ambos casos disminuyen el rendimiento predictivo del modelo.**

- Se puede estimar un valor de *cp* óptimo probando diferentes valores y utilizando validación cruzada para determinar la precisión de predicción correspondiente del modelo. El mejor *cp* se define entonces como el que maximiza la precisión de la validación cruzada.

# 7. Ejemplo. VD: Categórica.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier, export_text, DecisionTreeRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, roc_curve, auc
from sklearn.metrics import make_scorer, mean_absolute_error, mean_squared_error, r2_score
```

# 7. Ejemplo. VD: Categórica.

file_path = …/arboles.csv'  # Reemplaza con la ruta correcta de tu archivo

df = pd.read_csv(file_path)

df.head()

# Se observa si se tiene algún valor perdido.

df.isna().sum()

| | sbp | tobacco | ldl | adiposity | famhist | typea | obesity | alcohol | age | chd |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 160 | 12.00 | 5.73 | 23.11 | Present | 49 | 25.30 | 97.20 | 52 | 1 |
| **1** | 144 | 0.01 | 4.41 | 28.61 | Absent | 55 | 28.87 | 2.06 | 63 | 1 |
| **2** | 118 | 0.08 | 3.48 | 32.28 | Present | 52 | 29.14 | 3.81 | 46 | 0 |
| **3** | 170 | 7.50 | 6.41 | 38.03 | Present | 51 | 31.99 | 24.26 | 58 | 1 |
| **4** | 134 | 13.60 | 3.50 | 27.78 | Present | 60 | 25.99 | 57.34 | 49 | 1 |

```
sbp          0
tobacco      0
ldl          0
adiposity    0
famhist      0
typea        0
obesity      0
alcohol      0
age          0
chd          0
dtype: int64
```

# 7. Ejemplo. VD: Categórica.

```
# Categorizar la variable de respuesta
df['chd'] = df['chd'].apply(lambda x: 'Yes' if x == 1 else 'No')
print(df.head())
```

| | sbp | tobacco | ldl | adiposity | famhist | typea | obesity | alcohol | age | chd |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 160 | 12.00 | 5.73 | 23.11 | Present | 49 | 25.30 | 97.20 | 52 | Yes |
| 1 | 144 | 0.01 | 4.41 | 28.61 | Absent | 55 | 28.87 | 2.06 | 63 | Yes |
| 2 | 118 | 0.08 | 3.48 | 32.28 | Present | 52 | 29.14 | 3.81 | 46 | No |
| 3 | 170 | 7.50 | 6.41 | 38.03 | Present | 51 | 31.99 | 24.26 | 58 | Yes |
| 4 | 134 | 13.60 | 3.50 | 27.78 | Present | 60 | 25.99 | 57.34 | 49 | Yes |

# 7. Ejemplo. VD: Categórica.

```
# Separar las variables predictoras y la variable de respuesta.
X = df[['tobacco']]
y = df['chd']
# min_sample_split: el número mínimo de casos que contiene una hoja para que pueda ser creada.
# criterion: Criterio de división: “gini”, “entropy”, “log_loss”.
# max_depth = Profundidad máxima del árbol. En caso de no especificar, el clasificador sigue segmentando hasta que las hojas son puras, o se alcanza el min_sample_split. Con caracter ilustrativo, se selecciona bajo.
arbol1 = DecisionTreeClassifier(min_samples_split=30, criterion='gini', max_depth = 2)
# Crear un conjunto de entrenamiento y uno de prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

# 7. Ejemplo. VD: Categórica.

```
# ES IMPORTANTE QUE LA DISTRIBUCIÓN DE LAS CLASES SEA 'SIMILAR' EN TRAIN Y TEST.
print(f'La frecuencia de cada clase en train es: \n{y_train.value_counts(normalize=True)}')
print(f'\nLa frecuencia de cada clase en test es: \n{y_test.value_counts(normalize=True)}')
```

```
La frecuencia de cada clase en train es:
0    0.658537
1    0.341463
Name: chd, dtype: float64

La frecuencia de cada clase en test es:
0    0.634409
1    0.365591
Name: chd, dtype: float64
```

```
# Construir el modelo de árbol de decisiones
arbol1.fit(X_train, y_train)
```

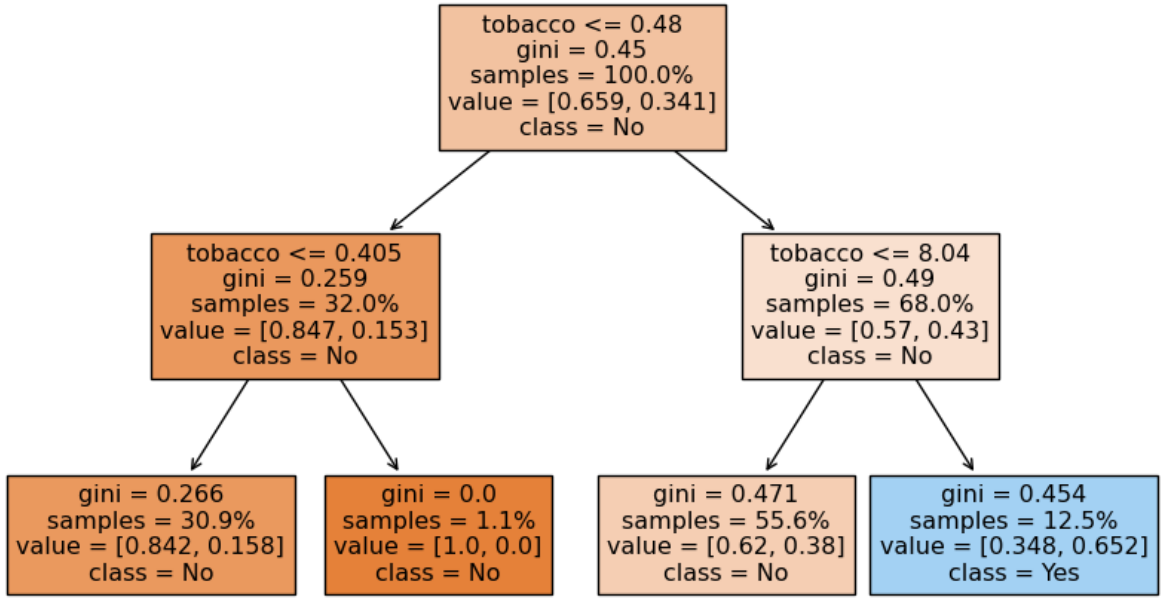
# 7. Ejemplo. VD: Categórica.

```
# Conocer los niveles de la variable a predecir
print(arbol1.classes_)
# Conocer el nombre de las variables predictoras
print(arbol1.feature_names_in_)
# Obtener información detallada de cada nodo y las reglas de decisión
tree_rules = export_text(arbol1, feature_names=list(X.columns),show_weights=True)
print(tree_rules)
```

```
['No' 'Yes']
['tobacco']
|--- tobacco <= 0.48
|   |--- tobacco <= 0.41
|   |   |--- weights: [96.00, 18.00] class: No
|   |--- tobacco >  0.41
|   |   |--- weights: [4.00, 0.00] class: No
|--- tobacco >  0.48
|   |--- tobacco <= 8.04
|   |   |--- weights: [127.00, 78.00] class: No
|   |--- tobacco >  8.04
|   |   |--- weights: [16.00, 30.00] class: Yes
```

# 7. Ejemplo. VD: Categórica.

```
plt.figure(figsize=(10, 6))
plot_tree(arbol1, feature_names=X.columns.tolist(), class_names=['No', 'Yes'], filled=True,
        proportion = True)
plt.show()
```

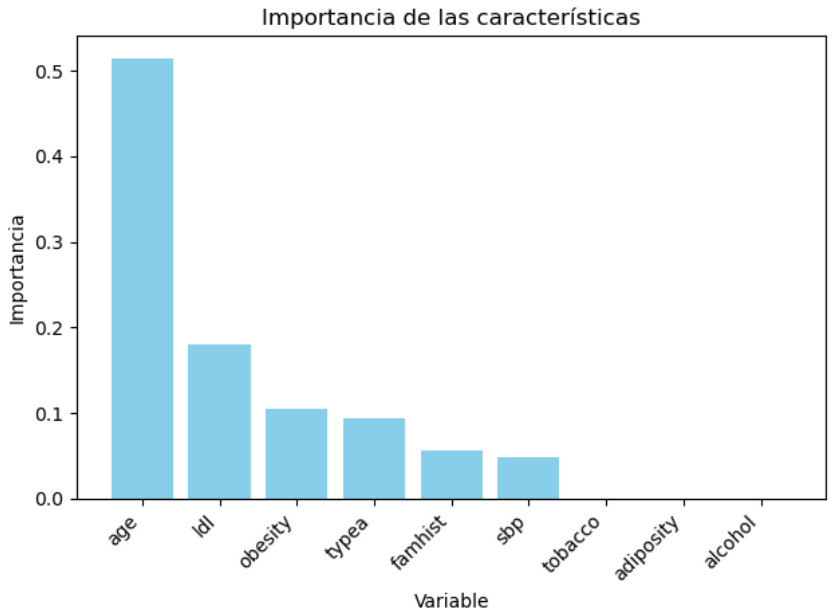


# 7. Ejemplo. VD: Categórica.

```
# Se vuelve a entrenar el árbol con más variables. No necesariamente tiene que utilizar todas, por lo que es importante
# conocer la importancia predictiva de cada variable en el modelo.


#es importante tratar de forma adecuada las variables categóricas. Se convierten en numéricas con la regla: one hot encoding.
df[['famhist']] = pd.get_dummies(df[['famhist']],drop_first=True)
# Separar las variables predictoras y la variable de respuesta.
X = df.drop('chd', axis=1)
y = df['chd']
#Se selecciona profundidad 4 sólo con caracter ilustrativo, al simplificar el árbol.
arbol2 = DecisionTreeClassifier(min_samples_split=30, criterion='gini', max_depth = 4)
# Crear un conjunto de entrenamiento y uno de prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Construir el modelo de árbol de decisiones
arbol2.fit(X_train, y_train)
```

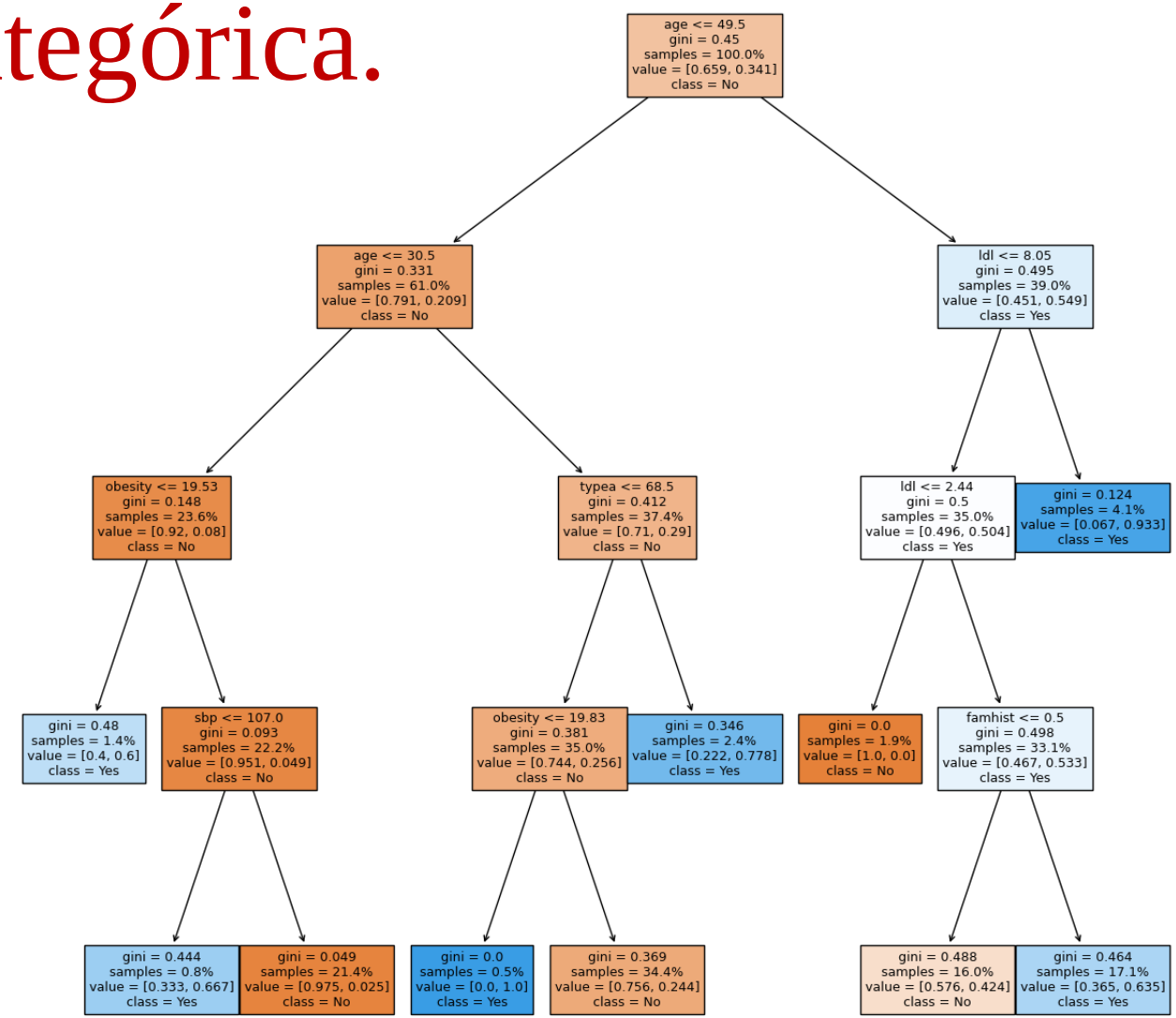
# 7. Ejemplo. VD: Categórica.

```
# Se estudia la importancia - o valor predictivo - de cada variable en el modelo.
print(pd.DataFrame({'nombre': arbol2.feature_names_in_, 'importancia': arbol2.feature_importances_}))
# Ordenar el DataFrame por importancia en orden descendente
df_importancia = pd.DataFrame({'Variable': arbol2.feature_names_in_, 'Importancia': arbol2.feature_importances_}).sort_values(by='Importancia', ascending=False)
# Crear un gráfico de barras
plt.bar(df_importancia['Variable'], df_importancia['Importancia'], color='skyblue')
plt.xlabel('Variable')
plt.ylabel('Importancia')
plt.title('Importancia de las características')
plt.xticks(rotation=45, ha='right')  # Rotar los nombres en el eje x para mayor legibilidad
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```



# 7. Ejemplo. VD: Categórica.

```
plt.figure(figsize=(15, 15))
plot_tree(arbol2, feature_names=X.columns.tolist(),
class_names=['No', 'Yes'], filled=True,
        proportion = True)
plt.show()
```



# 7. Ejemplo. VD: Continua.

```
file_path = …/Árboles/compress.csv'  # Reemplaza con la ruta correcta de tu archivo
compress = pd.read_csv(file_path)
compress.head()
```

```
compress.isna().sum()
```

| | cstrength | cement | blast | ash | water | plasti | aggreg | fineagg | age |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 79.99 | 540.0 | 0.0 | 0.0 | 162.0 | 2.5 | 1040.0 | 676.0 | 28 |
| **1** | 61.89 | 540.0 | 0.0 | 0.0 | 162.0 | 2.5 | 1055.0 | 676.0 | 28 |
| **2** | 40.27 | 332.5 | 142.5 | 0.0 | 228.0 | 0.0 | 932.0 | 594.0 | 270 |
| **3** | 41.05 | 332.5 | 142.5 | 0.0 | 228.0 | 0.0 | 932.0 | 594.0 | 365 |
| **4** | 44.30 | 198.6 | 132.4 | 0.0 | 192.0 | 0.0 | 978.4 | 825.5 | 360 |

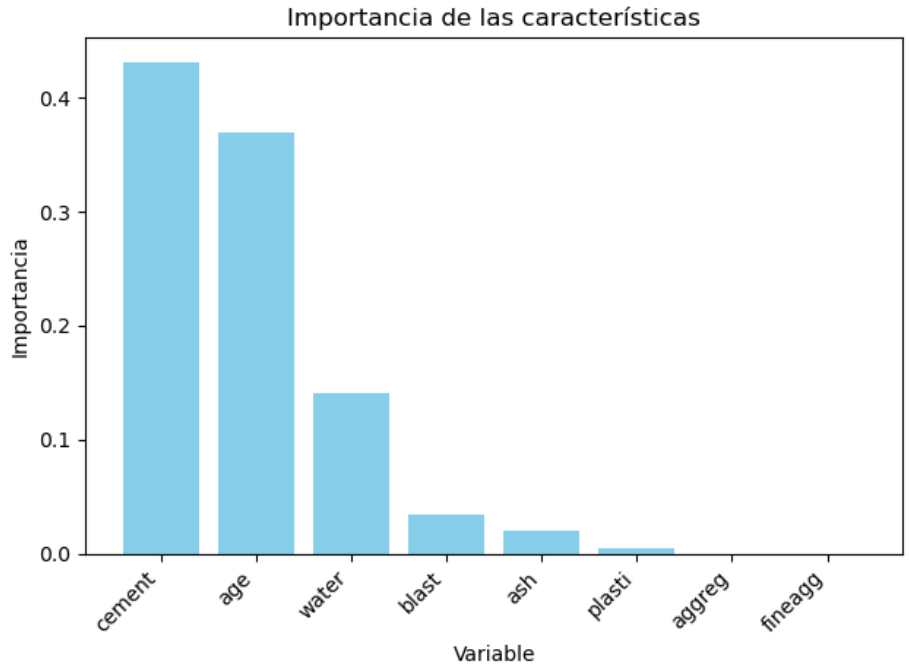
```
cstrength    0
cement       0
blast        0
ash          0
water        0
plasti       0
aggreg       0
fineagg      0
age          0
dtype: int64
```

# 7. Ejemplo. VD: Continua.

```
# Separar las variables predictoras y la variable de respuesta.
X_c = compress.drop('cstrength', axis=1)
y_c = compress['cstrength']
# criterion: Criterio de división: “squared_error”, “friedman_mse”, “absolute_error”, “poisson”}, default=”squared_error”.
# Se selecciona squared error con motivos ilustrativos. Es equivalente a la reducción de varianza.
# A priori, con motivos ilustrativos, se mantiene min_sample_split y max_depth en estos valores para facilitar la visualización del árbol.
# Recordar que estos son parámetros a modificar para encontrar el modelo óptimo.
# cpp_alpha es el parámetro de complejidad, el cual establece “penalizaciones” si se producen muchas divisiones. Cuanto más alto
# más pequeño será el árbol.
arbol3 = DecisionTreeRegressor(min_samples_split=30, criterion='squared_error', max_depth = 4, ccp_alpha = 0.01)
# Crear un conjunto de entrenamiento y uno de prueba
X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(X_c, y_c, test_size=0.2, random_state=42)
# Construir el modelo de árbol de decisiones
arbol3.fit(X_train_c, y_train_c)
```

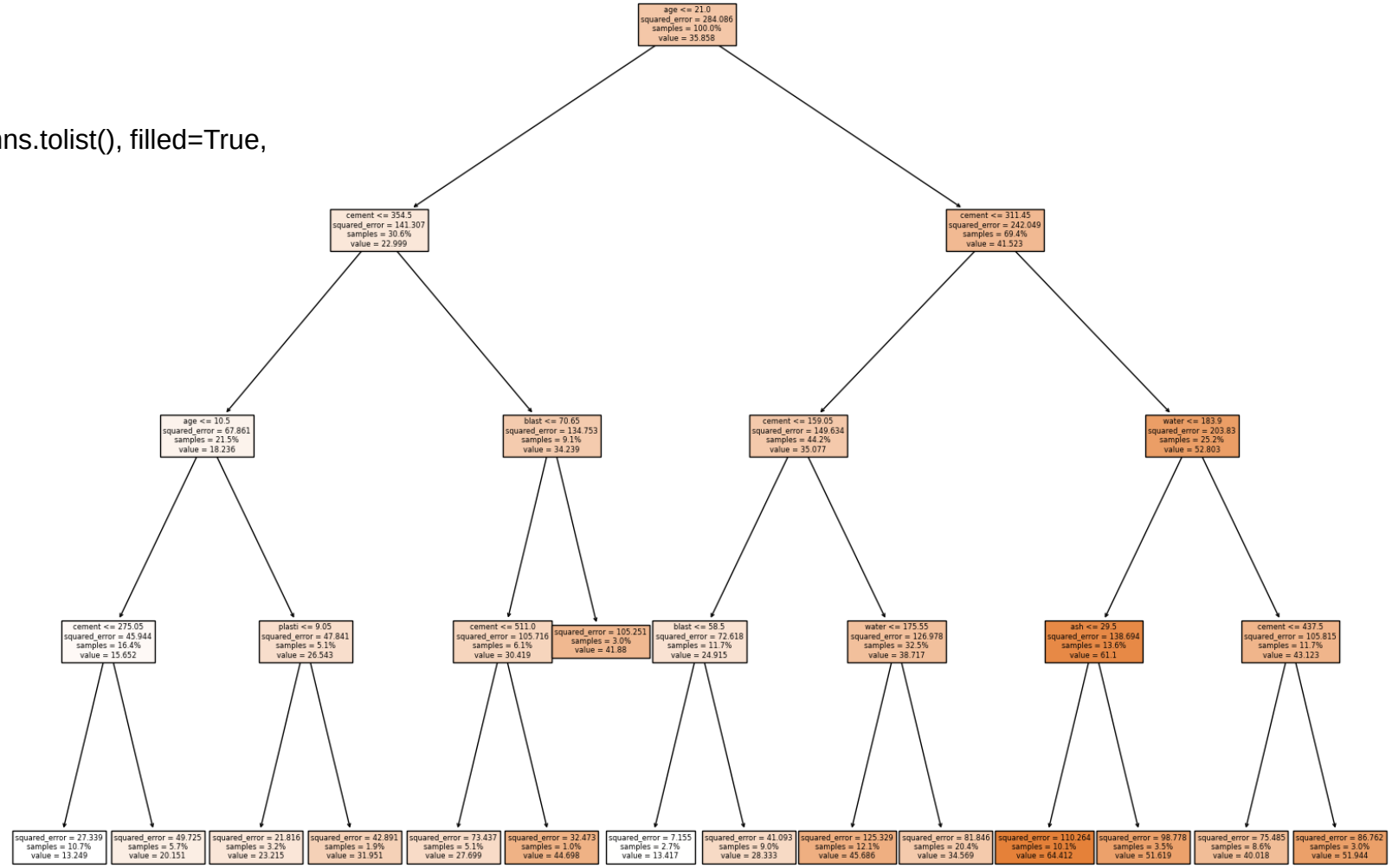
# 7. Ejemplo. VD: Continua.

```
print(pd.DataFrame({'nombre': arbol3.feature_names_in_, 'importancia': arbol3.feature_importances_}))
# Ordenar el DataFrame por importancia en orden descendente

df_importancia_c = pd.DataFrame({'Variable': arbol3.feature_names_in_, 'Importancia': arbol3.feature_importances_}).sort_values(by='Importancia', ascending=False)

# Crear un gráfico de barras
plt.bar(df_importancia_c['Variable'], df_importancia_c['Importancia'], color='skyblue')
plt.xlabel('Variable')
plt.ylabel('Importancia')
plt.title('Importancia de las características')
plt.xticks(rotation=45, ha='right')  # Rotar los nombres en el eje x para mayor legibilidad
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```



# 7. Ejemplo. VD: Continua.

```
plt.figure(figsize=(20, 15))
plot_tree(arbol3, feature_names=X_c.columns.tolist(), filled=True,
        proportion = True)
plt.show()
```



# 7. Ejemplo. VD: Categórica. Tuneo y evaluación predictiva

```
## tuneo y evaluación predictiva del modelo para variable dependiente categórica.


params = {
    'max_depth': [2, 3, 5, 10, 20],
    'min_samples_split': [5, 10, 20, 50, 100],
    'criterion': ["gini", "entropy"]
}
scoring_metrics = ['accuracy', 'precision_macro', 'recall_macro', 'f1_macro']
#recordar que arbol2 es el árbol cuyas VI son todas las variables.
# cv = crossvalidation
grid_search = GridSearchCV(estimator=arbol2,
                           param_grid=params,
                           cv=4, scoring = scoring_metrics, refit='accuracy')
grid_search.fit(X_train, y_train)
```

# 7. Ejemplo. VD: Categórica. Tuneo y evaluación predictiva

```
# Obtener resultados del grid search
results = pd.DataFrame(grid_search.cv_results_)

# Mostrar resultados
print("Resultados de Grid Search:")
print(results[['params', 'mean_test_accuracy',
'mean_test_precision_macro',
'mean_test_recall_macro', 'mean_test_f1_macro']])

# Obtener el mejor modelo
best_model = grid_search.best_estimator_
print(grid_search.best_estimator_)
```

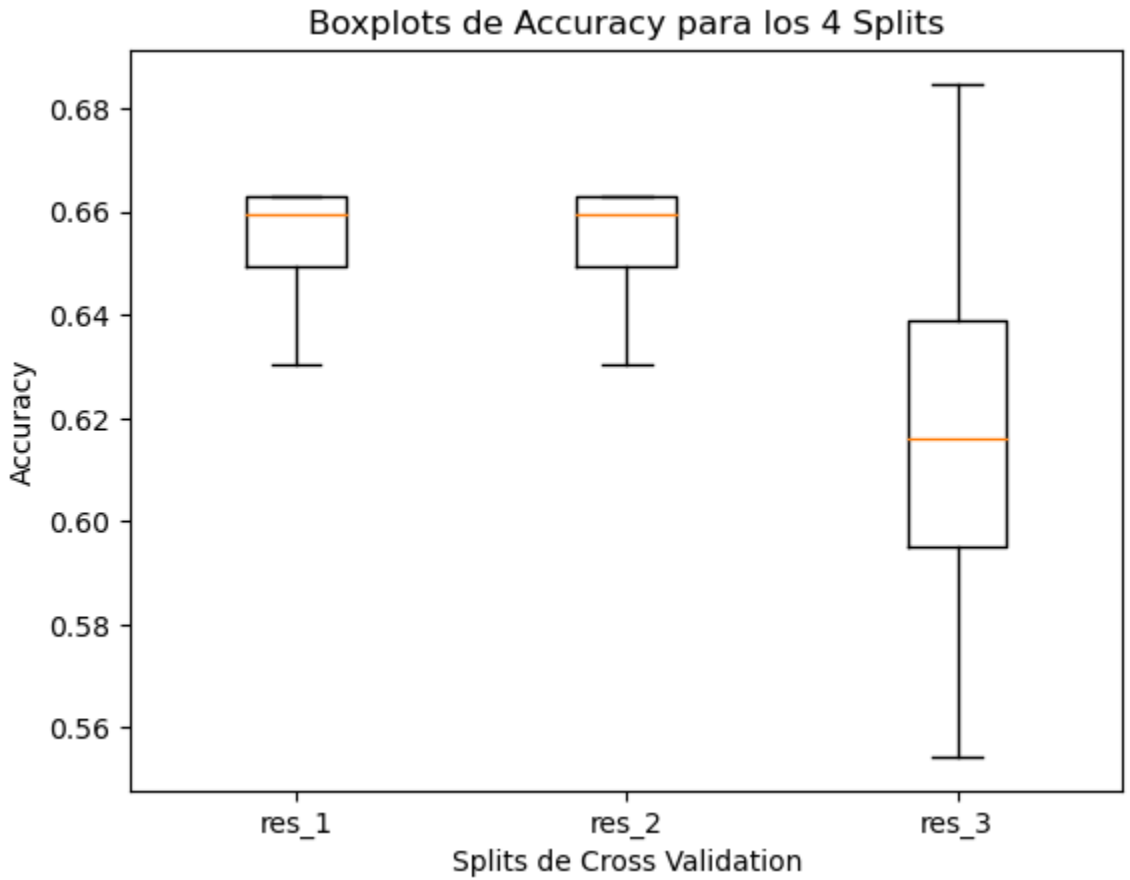
```
Resultados de Grid Search:
                                                params  mean_test_accuracy  \
0   {'criterion': 'gini', 'max_depth': 2, 'min_sam...            0.650421
1   {'criterion': 'gini', 'max_depth': 2, 'min_sam...            0.650421
2   {'criterion': 'gini', 'max_depth': 2, 'min_sam...            0.653109
3   {'criterion': 'gini', 'max_depth': 2, 'min_sam...            0.653109
4   {'criterion': 'gini', 'max_depth': 2, 'min_sam...            0.653109
5   {'criterion': 'gini', 'max_depth': 3, 'min_sam...            0.631399
6   {'criterion': 'gini', 'max_depth': 3, 'min_sam...            0.620588
7   {'criterion': 'gini', 'max_depth': 3, 'min_sam...            0.626023
8   {'criterion': 'gini', 'max_depth': 3, 'min_sam...            0.625964
9   {'criterion': 'gini', 'max_depth': 3, 'min_sam...            0.626023
10  {'criterion': 'gini', 'max_depth': 5, 'min_sam...            0.634087
11  {'criterion': 'gini', 'max_depth': 5, 'min_sam...            0.639522
12  {'criterion': 'gini', 'max_depth': 5, 'min_sam...            0.639493
13  {'criterion': 'gini', 'max_depth': 5, 'min_sam...            0.631428
DecisionTreeClassifier(criterion='entropy', max_depth=3, min_samples_split=100)
```

# 7. Ejemplo. VD: Categórica. Tuneo y evaluación predictiva

```
# Para seleccionar una parametrización específica y la mejor de acuerdo con el criterio
# de GridSearch, acceder a esta y conocer su combinación.
results.iloc[8].params


# (En este caso, son ejemplos de selección aleatorios para ilustrar el script de selección y representación),.
# se selecciona el modelo candidato, y se procede a analizar su robustez a lo largo de cross validation.
res_1 = results[['split0_test_accuracy', 'split1_test_accuracy','split2_test_accuracy', 'split3_test_accuracy']].iloc[2]
res_2 = results[['split0_test_accuracy', 'split1_test_accuracy','split2_test_accuracy', 'split3_test_accuracy']].iloc[4]
res_3 = results[['split0_test_accuracy', 'split1_test_accuracy','split2_test_accuracy', 'split3_test_accuracy']].iloc[18]
```

# 7. Ejemplo. VD: Categórica. Tuneo y evaluación predictiva

```
# Crear un boxplot para los cuatro valores de accuracy
plt.boxplot([res_1.values,res_2.values,res_3.values], labels = [‘res_1','res_2’,’res_3'])
plt.title('Boxplots de Accuracy para los 4 Splits')
plt.xlabel('Splits de Cross Validation')
plt.ylabel('Accuracy')
plt.show()
# Nótese en la solución que boxplots con gran amplitud no son deseables,
#  ya que se caracterizan por poca robustez de la solución
```



# 7. Ejemplo. VD: Categórica. Tuneo y evaluación predictiva

```
# Obtener el mejor modelo (best estimador, o el seleccionado dado los pasos anteriores).
best_model = grid_search.best_estimator_

# Ajustar el mejor modelo con todo el conjunto de entrenamiento
best_model.fit(X_train, y_train)

# Predicciones en conjunto de entrenamiento y prueba
y_train_pred = best_model.predict(X_train)
y_test_pred = best_model.predict(X_test)
```

# 7. Ejemplo. VD: Categórica. Tuneo y evaluación predictiva

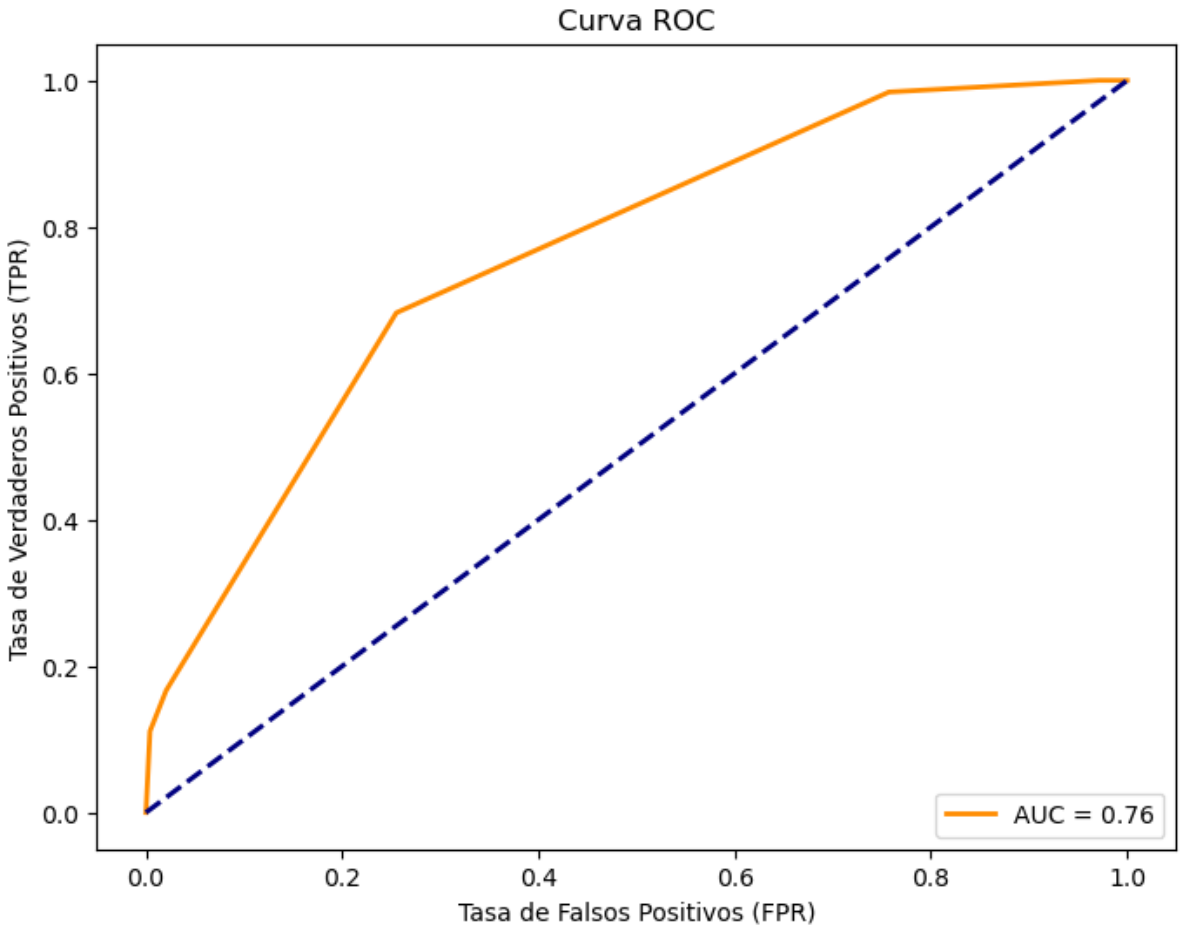
#medidas de bondad de ajuste en **train**

```
conf_matrix = confusion_matrix(y_train, y_train_pred)
print("Matriz de Confusión:")
print(conf_matrix)
print("\nMedidas de Desempeño:")
print(classification_report(y_train, y_train_pred))
```

```
Matriz de Confusión:
[[181  62]
 [ 40  86]]

Medidas de Desempeño:
              precision    recall  f1-score   support

          No       0.82      0.74      0.78       243
         Yes       0.58      0.68      0.63       126

    accuracy                           0.72       369
   macro avg       0.70      0.71      0.70       369
weighted avg       0.74      0.72      0.73       369
```

# 7. Ejemplo. VD: Categórica. Tuneo y evaluación predictiva

```
y_train_auc = pd.get_dummies(y_train,drop_first=True)
# Calcular el área bajo la curva ROC (AUC)
y_prob_train = best_model.predict_proba(X_train)[:, 1]
fpr, tpr, thresholds = roc_curve(y_train_auc, y_prob_train)
roc_auc = auc(fpr, tpr)
print(f"\nÁrea bajo la curva ROC (AUC): {roc_auc:.2f}")
# Graficar la curva ROC
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('Tasa de Falsos Positivos (FPR)')
plt.ylabel('Tasa de Verdaderos Positivos (TPR)')
plt.title('Curva ROC')
plt.legend(loc="lower right")
plt.show()
```



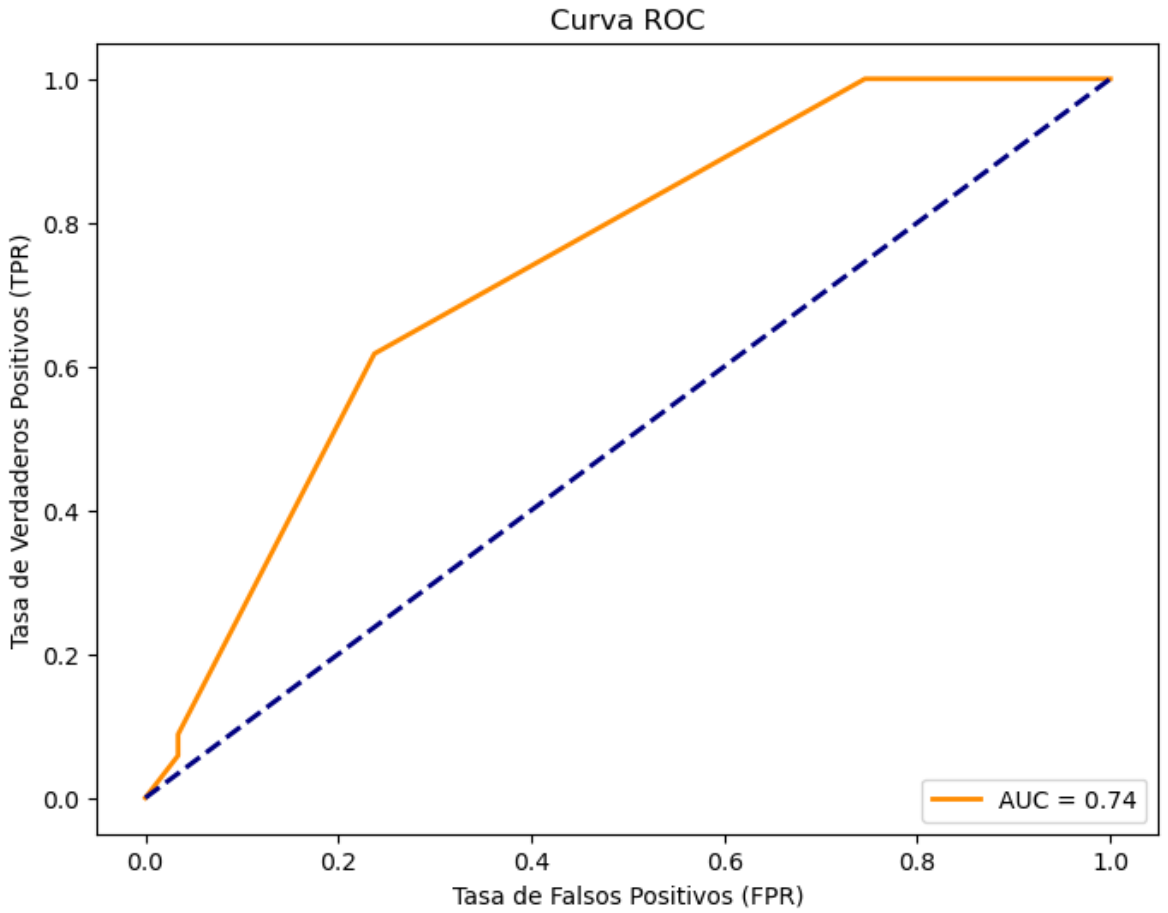
# 7. Ejemplo. VD: Categórica. Tuneo y evaluación predictiva

```
# medidas de bondad de ajuste en test


conf_matrix = confusion_matrix(y_test, y_test_pred)
print("Matriz de Confusión:")
print(conf_matrix)
print("\nMedidas de Desempeño:")
print(classification_report(y_test, y_test_pred))
```

```
Matriz de Confusión:
[[45 14]
 [13 21]]

Medidas de Desempeño:
              precision    recall  f1-score   support

          No       0.78      0.76      0.77        59
         Yes       0.60      0.62      0.61        34

    accuracy                           0.71        93
   macro avg       0.69      0.69      0.69        93
weighted avg       0.71      0.71      0.71        93
```

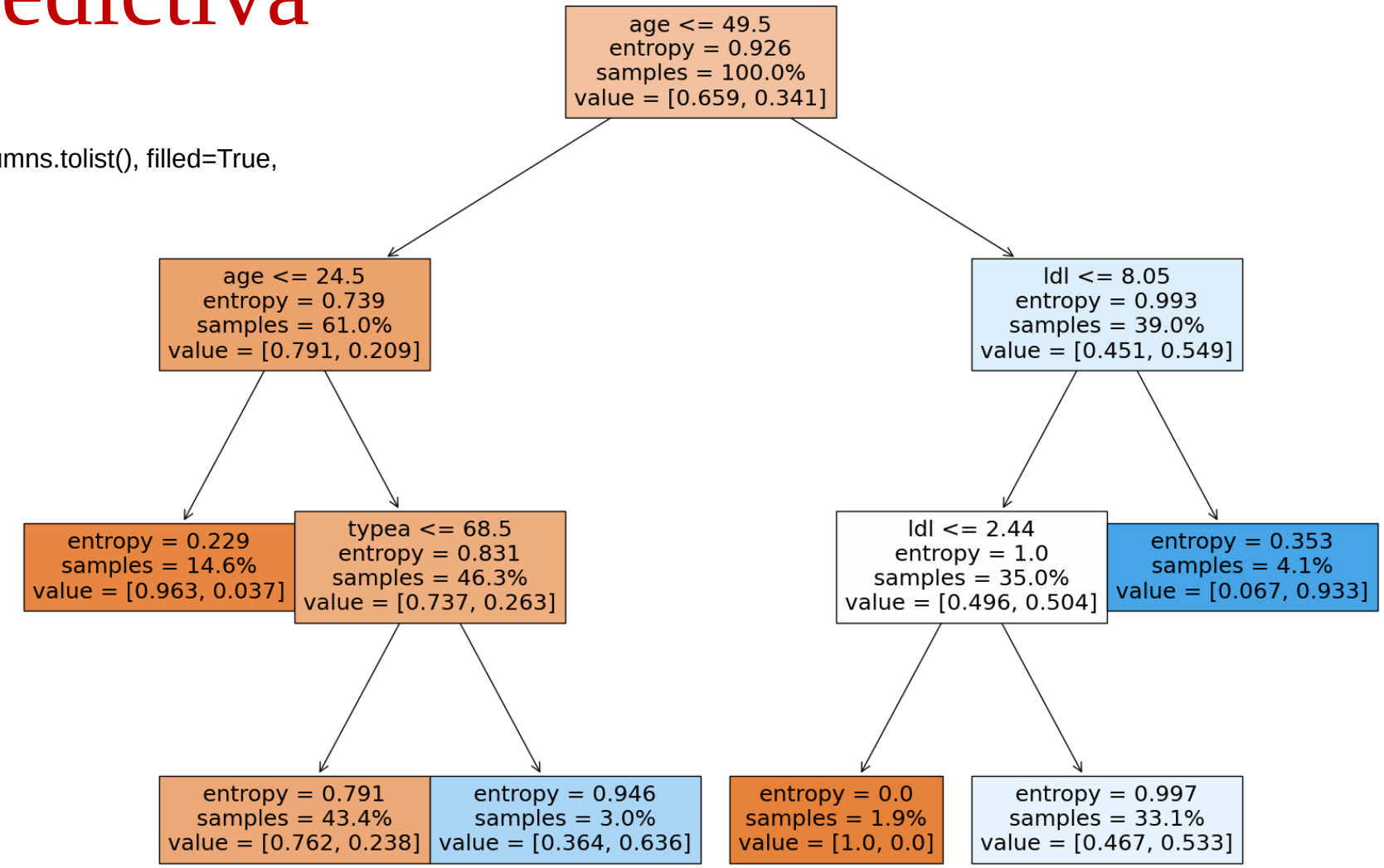
# 7. Ejemplo. VD: Categórica. Tuneo y evaluación predictiva

```
y_test_auc = pd.get_dummies(y_test,drop_first=True)
# Calcular el área bajo la curva ROC (AUC)
y_prob_test = best_model.predict_proba(X_test)[:, 1]


fpr, tpr, thresholds = roc_curve(y_test_auc, y_prob_test)
roc_auc_test = auc(fpr, tpr)
print(f"\nÁrea bajo la curva ROC (AUC): {roc_auc:.2f}")
# Graficar la curva ROC
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('Tasa de Falsos Positivos (FPR)')
plt.ylabel('Tasa de Verdaderos Positivos (TPR)')
plt.title('Curva ROC')
plt.legend(loc="lower right")
plt.show()
```



# 7. Ejemplo. VD: Categórica. Tuneo y evaluación predictiva

```
plt.figure(figsize=(20, 15))
plot_tree(best_model, feature_names=X.columns.tolist(), filled=True,
        proportion = True)
plt.show()
```



# 7. Ejemplo. VD: Continua. Tuneo y evaluación predictiva

```
## tuneo y evaluación predictiva del modelo para variable dependiente numérica.
params_c = {
    'max_depth': [2, 3, 5, 10, 20],
    'min_samples_split': [5, 10, 20, 50, 100],
    'criterion': ['squared_error', 'friedman_mse', 'absolute_error', 'poisson']}
# Definir las métricas de evaluación que deseas utilizar
scoring_metrics_c = {'MAE': make_scorer(mean_absolute_error),
    'MSE': make_scorer(mean_squared_error),
    'RMSE': make_scorer(lambda y_true, y_pred: mean_squared_error(y_true, y_pred, squared=False))}
# cv = crossvalidation
grid_search_c = GridSearchCV(estimator=arbol3,
                             param_grid=params_c,
                             cv=4, scoring = scoring_metrics_c, refit='MSE')
grid_search_c.fit(X_train_c, y_train_c)
```

# 7. Ejemplo. VD: Continua. Tuneo y evaluación predictiva

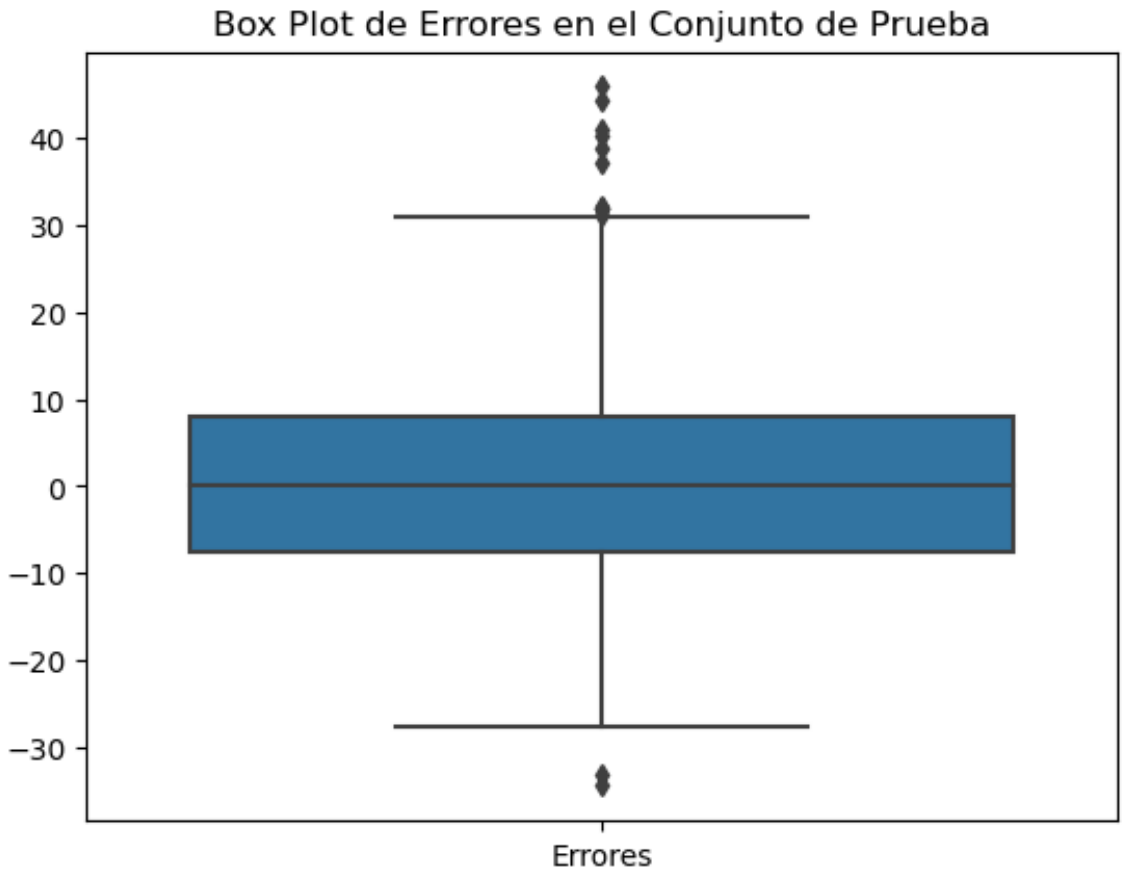
```
# Obtener resultados del grid search
results_c = pd.DataFrame(grid_search_c.cv_results_)

# Mostrar resultados
print("Resultados de Grid Search:")
print(results_c)

# Obtener el mejor modelo
best_model_c = grid_search_c.best_estimator_
print(grid_search_c.best_estimator_)
```

```
Resultados de Grid Search:
    mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0        0.001991      0.000648         0.000811    4.713109e-04
1        0.002959      0.005125         0.000746    1.292440e-03
2        0.001777      0.000451         0.001000    6.078505e-07
3        0.002201      0.000473         0.001704    4.174331e-04
4        0.001524      0.000476         0.000999    3.520208e-06
..            ...           ...              ...             ...
95       0.004276      0.007000         0.003779    6.544894e-03
96       0.000000      0.000000         0.003778    6.543964e-03
97       0.003934      0.006813         0.000000    0.000000e+00
98       0.004091      0.007086         0.000000    0.000000e+00
99       0.000000      0.000000         0.000000    0.000000e+00

   param_criterion param_max_depth param_min_samples_split  \
0    squared_error               2                       5
1    squared_error               2                      10
2    squared_error               2                      20
3    squared_error               2                      50
4    squared_error               2                     100
```

```
DecisionTreeRegressor(ccp_alpha=0.01, criterion='absolute_error', max_depth=2, min_samples_split=5)
```

# 7. Ejemplo. VD: Continua. Tuneo y evaluación predictiva

```
# Ajustar el mejor modelo con todo el conjunto de entrenamiento
best_model_c.fit(X_train_c, y_train_c)
# Predicciones en conjunto de entrenamiento y prueba
y_train_pred_c = best_model_c.predict(X_train_c)
y_test_pred_c = best_model_c.predict(X_test_c)
# Medidas de bondad de ajuste en train
y_pred_train_c = best_model_c.predict(X_train_c)
# Suponiendo que tienes los valores reales en y_test_c y
#las predicciones en y_pred_test_c
errores = y_train_c - y_pred_train_c
# Convertir los errores a un DataFrame
errores_df = pd.DataFrame({'Errores': errores})
# Box Plot de los errores en el conjunto de prueba
sns.boxplot(errores_df)
plt.title('Box Plot de Errores en el Conjunto de Prueba')
plt.show()
```



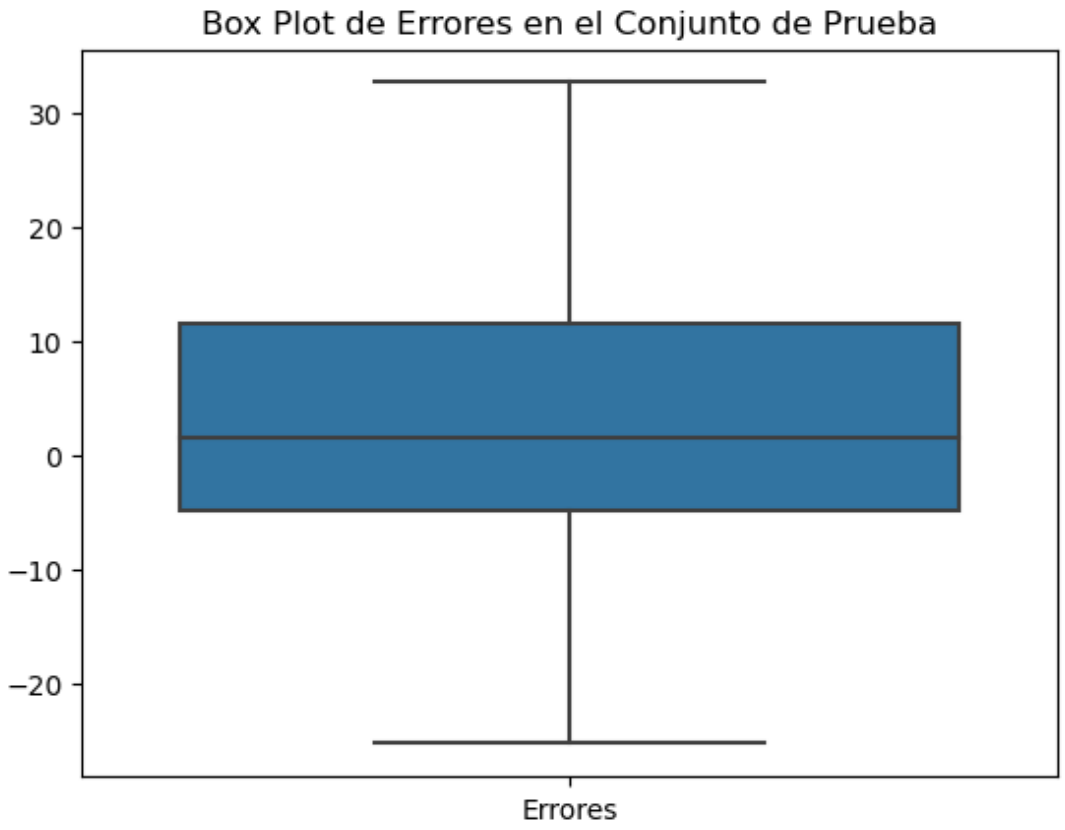
# 7. Ejemplo. VD: Continua. Tuneo y evaluación predictiva

```
# Calcular diferentes medidas de bondad de ajuste
mae = mean_absolute_error(y_train_c, y_pred_train_c)
mse = mean_squared_error(y_train_c, y_pred_train_c)
rmse = np.sqrt(mse)
r2 = r2_score(y_train_c, y_pred_train_c)


# Imprimir las métricas
print(f'MAE (Error Absoluto Medio): {mae:.2f}')
print(f'MSE (Error Cuadrático Medio): {mse:.2f}')
print(f'RMSE (Raíz del Error Cuadrático Medio): {rmse:.2f}')
print(f'R²: {r2:.2f}')
```

```
MAE (Error Absoluto Medio): 9.46
MSE (Error Cuadrático Medio): 146.00
RMSE (Raíz del Error Cuadrático Medio): 12.08
R²: 0.49
```

# 7. Ejemplo. VD: Continua. Tuneo y evaluación predictiva

```
# Medidas de bondad de ajuste en test:


# Medidas de bondad de ajuste en train
y_pred_test_c = best_model_c.predict(X_test_c)
# Suponiendo que tienes los valores reales en y_test_c y
#las predicciones en y_pred_test_c
errores = y_test_c - y_pred_test_c


# Convertir los errores a un DataFrame
errores_df = pd.DataFrame({'Errores': errores})
# Box Plot de los errores en el conjunto de prueba
sns.boxplot(errores_df)
plt.title('Box Plot de Errores en el Conjunto de Prueba')
plt.show()
```



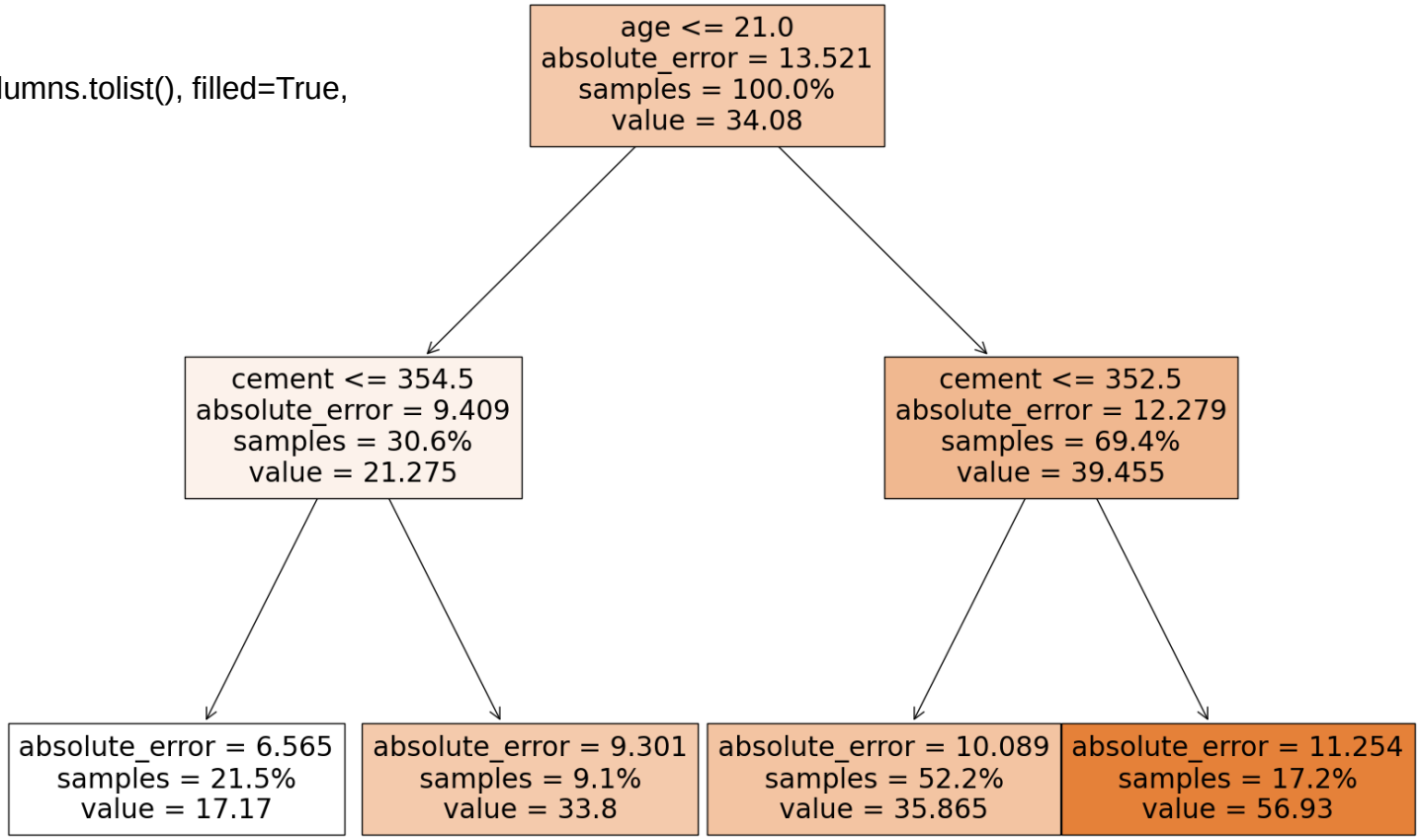
# 7. Ejemplo. VD: Continua. Tuneo y evaluación predictiva

```
mae = mean_absolute_error(y_test_c, y_pred_test_c)
mse = mean_squared_error(y_test_c, y_pred_test_c)
rmse = np.sqrt(mse)
r2 = r2_score(y_test_c, y_pred_test_c)

# Imprimir las métricas
print(f'MAE (Error Absoluto Medio): {mae:.2f}')
print(f'MSE (Error Cuadrático Medio): {mse:.2f}')
print(f'RMSE (Raíz del Error Cuadrático Medio): {rmse:.2f}')
print(f'R²: {r2:.2f}')
```

```
MAE (Error Absoluto Medio): 9.49
MSE (Error Cuadrático Medio): 141.96
RMSE (Raíz del Error Cuadrático Medio): 11.91
R²: 0.45
```

# 7. Ejemplo. VD: Continua. Tuneo y evaluación predictiva

```
plt.figure(figsize=(20, 15))
plot_tree(best_model_c, feature_names=X_c.columns.tolist(), filled=True,
        proportion = True)
plt.show()
```



# 8. Notas finales.

IMPORTANTE: grandes diferencias de los árboles frente a otros métodos predictivos.

- **Las transformaciones monótonas** (crecientes o decrecientes) sobre las variables continuas **no tienen efecto**, el árbol solo tiene en cuenta **el orden** entre observaciones.
- Los **missing** están incorporados al proceso y normalmente no necesitan un tratamiento exhaustivo previo (solo lo básico, borrar observaciones y variables con demasiados missings si el hecho de ser missing no es informativo, en caso de las variables)
- La **selección de variables está incorporada al proceso** (método embedded). Aunque siempre es beneficioso realizar un estudio previo y preselección
- Es el mejor método para descubrir **interacciones entre variables categóricas** (de hecho CHAID, el primer método de árboles, significa Chi-Square Automatic Interaction Detector)

# 8. Notas finales.

## Ventajas de los árboles:

- Gran potencia descriptiva, se comprende muy bien el resultado. Resultados a menudo simples.
- Sobre todo en clasificación, pero también en regresión, se descubren interacciones y reglas muy difíciles de encontrar con otros métodos. Estas reglas se pueden utilizar como variables dummy para utilizar en otros métodos predictivos.
- Las relaciones no lineales no afectan tanto al comportamiento de los árboles como a otros métodos.
- No hay asunciones teóricas sobre los datos
- Aportan medidas de importancia de las variables
- Manera propia y eficiente de tratar los missings, incorporada al proceso
- Incorporación automática de interacciones, detectan relaciones por regiones que ningún otro método puede encontrar

# 8. Notas finales.

Desventajas de los árboles:

- Poca fiabilidad y mala generalización: cada hoja es un parámetro y esto provoca modelos sobreajustados e inestables para la predicción. Añadir una variable nueva o un nuevo conjunto de observaciones puede alterar mucho el árbol.

- Complejidad en la construcción del árbol y casuística: dos plataformas (programas) diferentes dan dos árboles diferentes

- Poca eficacia predictiva, sobre todo en regresión: toscos en los valores de predicción. Por ello raramente son el modelo final. Se suelen usar como apoyo a otros modelos.

# 9. Referencias.


1. Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. 1nd ed. Boca Raton, Florida: Chapman; https://www.crcpress.com/Classification-and-Regression-Trees/Breiman-Friedman-Stone-Olshen/p/book/9780412048418.

2. Hastie, Tibshirani: *The Elements of Statistical Learning* (PDF) (En la web hay más información) http://statweb.stanford.edu/~tibs/ElemStatLearn/

3. https://scikit-learn.org/