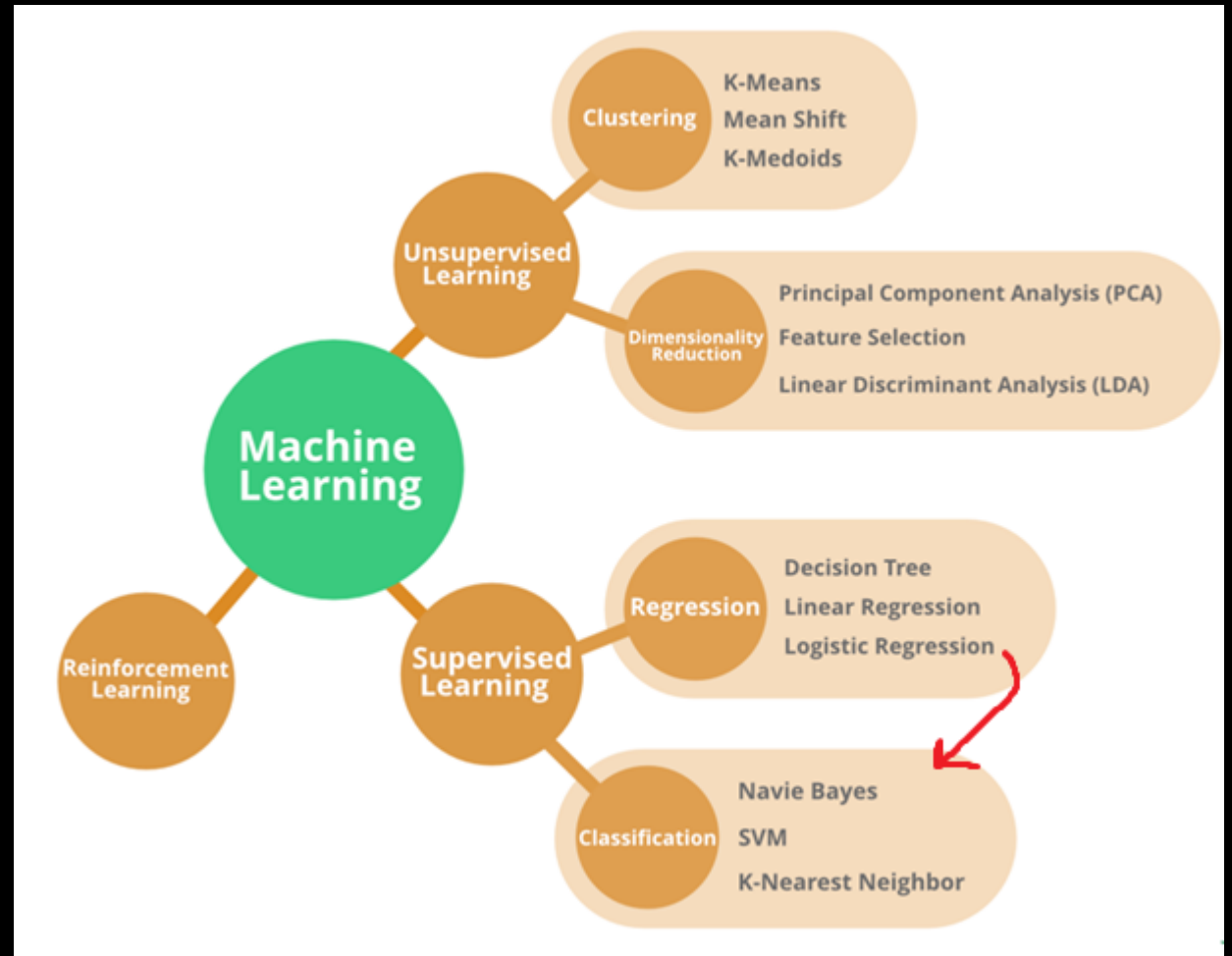


Machine Learning – Árboles de decisión

Algoritmos de machine learning

- Aprendizaje supervisado:
 - Regresión
 - Clasificación
- Aprendizaje no supervisado:
 - Clusterización
 - Reducción de dimensionalidad
- Aprendizaje por refuerzo

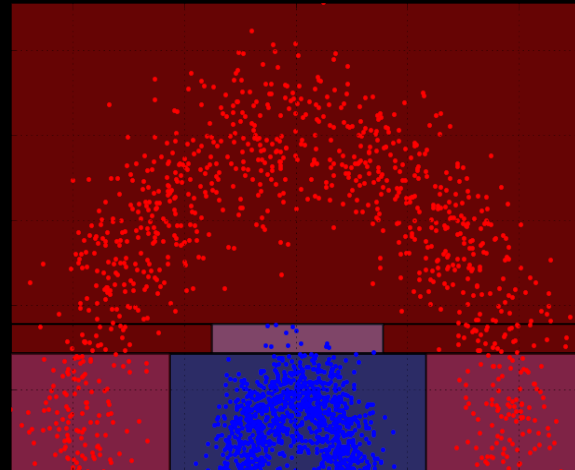
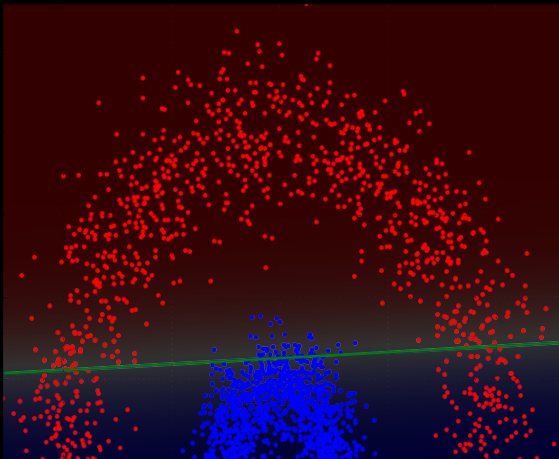


Definición

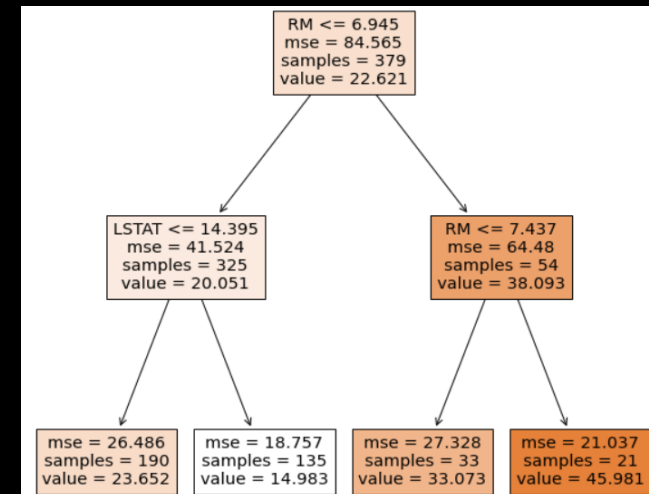
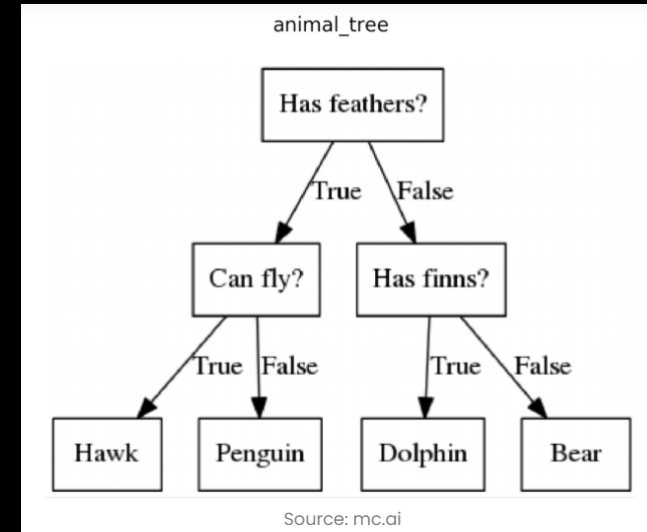
Definición

Algoritmo supervisado
White box

No lineal



Clasificación y Regresión



Algunos términos importantes

CART algorithm

Algoritmo utilizado por Sklearn. Produce árboles binarios y se puede utilizar para regresión y para clasificación.

Root node

Representa toda la muestra, que luego se subdivide.

Decision node

Nodo que se divide a su vez en otros subnodos.

Terminal node o leaf node

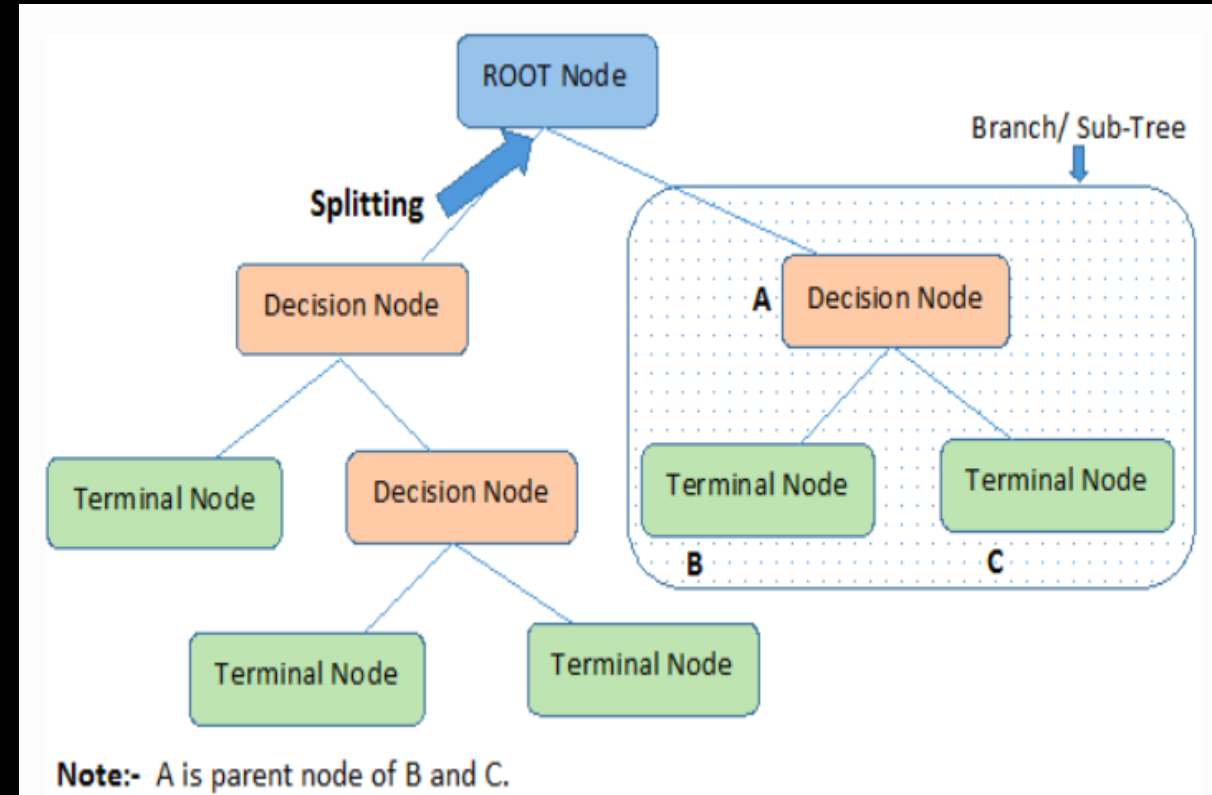
Nodo que no se puede subdividir en otros subnodos.

Splitting

División de un nodo en dos ramas basada en condiciones if-else.

Profundidad del árbol (Depth)

Cuántos niveles tiene el árbol. En este ejmplo serían tres.

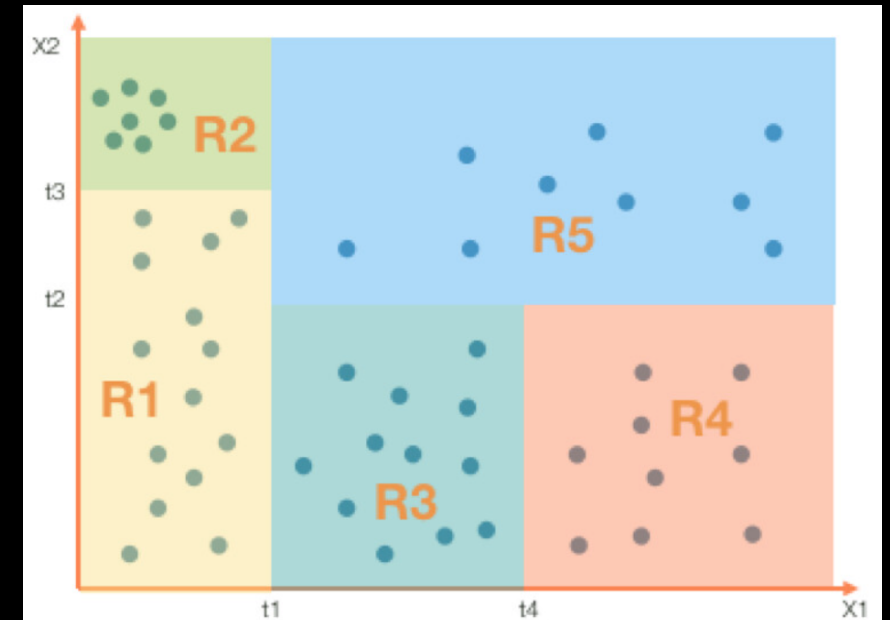
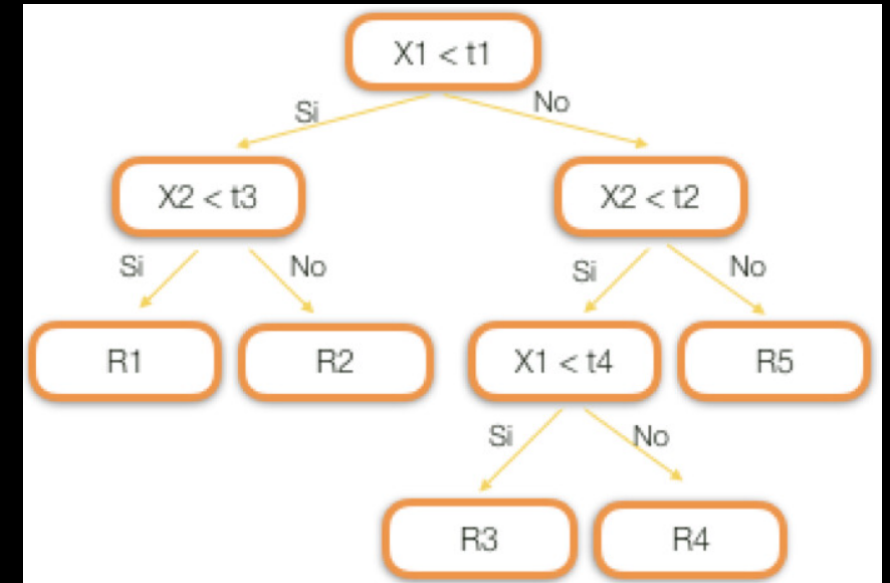


¿Cómo creamos un árbol de
decision?

Decision tree classifier

Funcionamiento

1. Dividimos el espacio muestral con la variable más predictiva (la que mejor separa los datos)
2. Tras esta división, el árbol se vuelve a dividir en el siguiente nivel con la siguiente variable que mejor separa los datos del nodo.
3. Y así, hasta que alcanzamos un criterio de parada:
 1. Están todos los elementos de la clase clasificados perfectamente (**WARNING**)
 2. No puede encontrar una división que reduzca la impureza del nodo.
 3. El árbol alcanza un tamaño predefinido.
4. La clase asignada al nodo es el **modo** de las clases de la instancias que caen en esa región. Esa será la predicción para nuevas instancias.



¿Cuál es la feature más
predictiva? ¿Cómo elegimos los
splits?

Mejores splits

El algoritmo CART divide cada nodo de la manera que **minimiza la suma ponderada de la impureza sus hijos**.

El algoritmo comienza dividiendo los datos de train en dos subconjuntos, utilizando una única variable k y un umbral t_k (e.g., “petal length ≤ 2.45 cm”). Lo que busca es el par (k, t_k) que produzca los subconjuntos más puros (ponderados por su tamaño).

Esta es la **función de coste** que el algoritmo trata de minimizar:

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

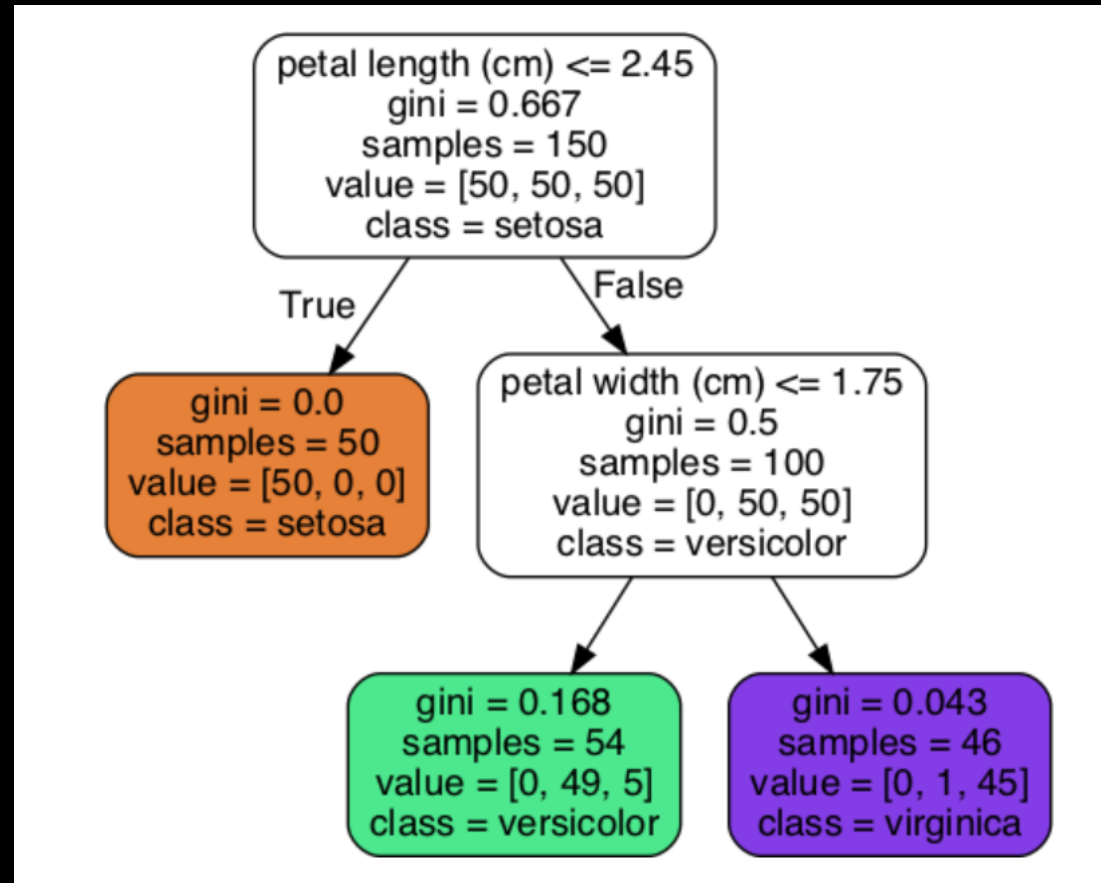
where $\begin{cases} G_{\text{left/right}} & \text{measures the impurity of the left/right subset,} \\ m_{\text{left/right}} & \text{is the number of instances in the left/right subset.} \end{cases}$

Una vez que el algoritmo CART ha dividido con éxito el conjunto de entrenamiento en dos, divide los subconjuntos usando la misma lógica, luego los sub-subconjuntos, y así sucesivamente, de forma recursiva, hasta que alcanza la profundidad máxima.

De forma predeterminada Sklearn utiliza la medida de impureza de **Gini**, pero es posible seleccionar la medida de impureza de **Entropía**, con el parámetro “criterion”.

Impurity	Task	Formula	Description
Gini impurity	Classification	$\sum_{i=1}^C f_i(1 - f_i)$	f_i is the frequency of label i at a node and C is the number of unique labels.
Entropy	Classification	$\sum_{i=1}^C -f_i \log(f_i)$	f_i is the frequency of label i at a node and C is the number of unique labels.

Ejemplo



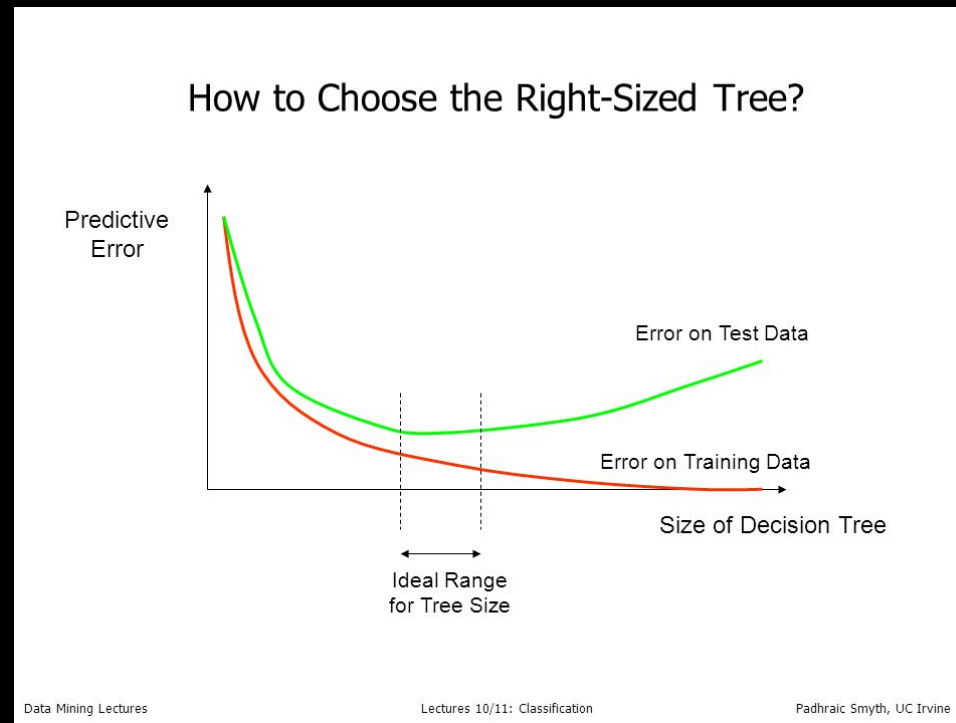
Profundidad del árbol

Riesgo de overfitting

Modelo no paramétrico : Los árboles de decisión hacen muy pocas suposiciones sobre los datos de entrenamiento, a diferencia de los modelos lineales, que asumen que los datos tienen una relación lineal, por ejemplo.

Por lo tanto, la estructura del modelo es libre para ajustarse todo lo posible a los datos de entrenamiento.

Si no se limita la dimensión del árbol, se producirá muy probablemente overfitting.

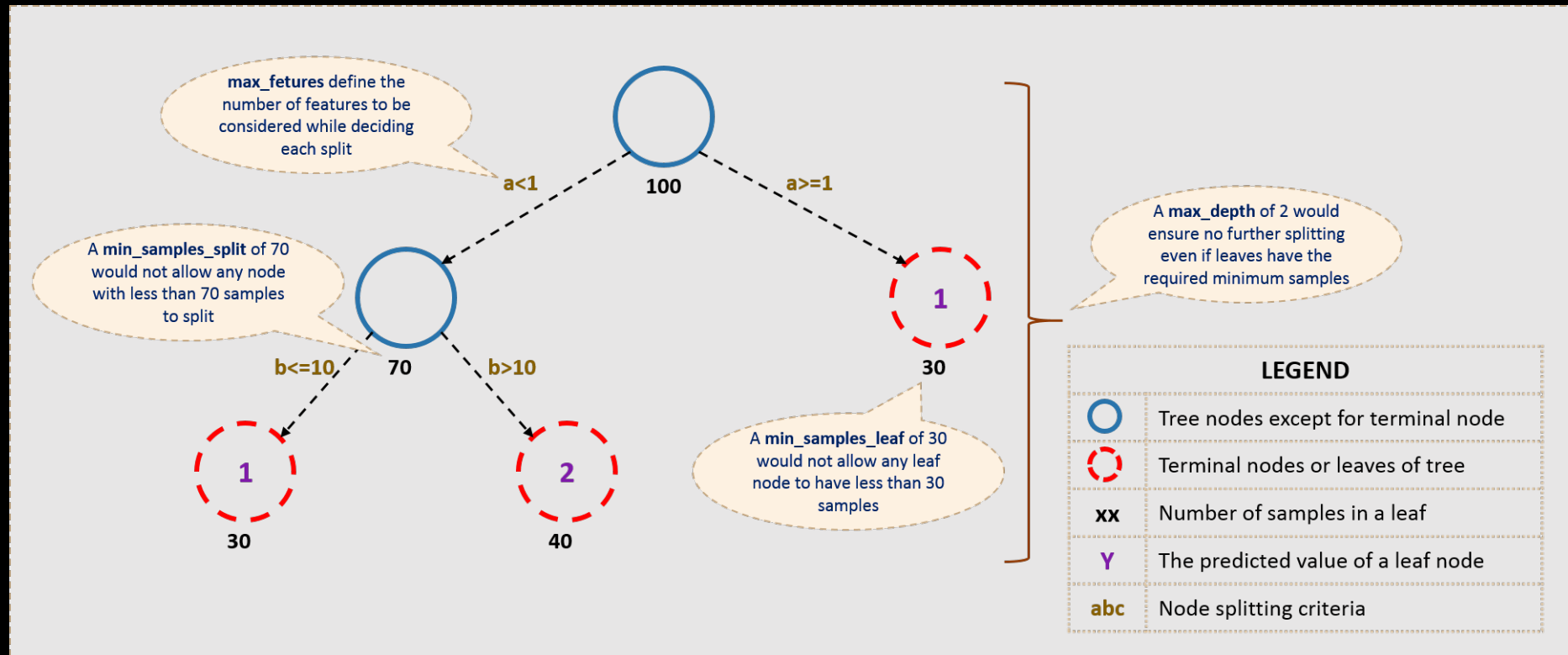


¿Cómo solucionamos el overfitting?
Prunning

Pruning

Sencilla técnica que consiste en “podar” el árbol. Lo único que tenemos que hacer es reducir la dimensión del árbol de decisión con un número menor de niveles de profundidad (parámetro `max_depth`).

Se pueden establecer otras limitaciones al tamaño del árbol para evitar el overfitting:



Decision Tree Regression

Árboles regresores

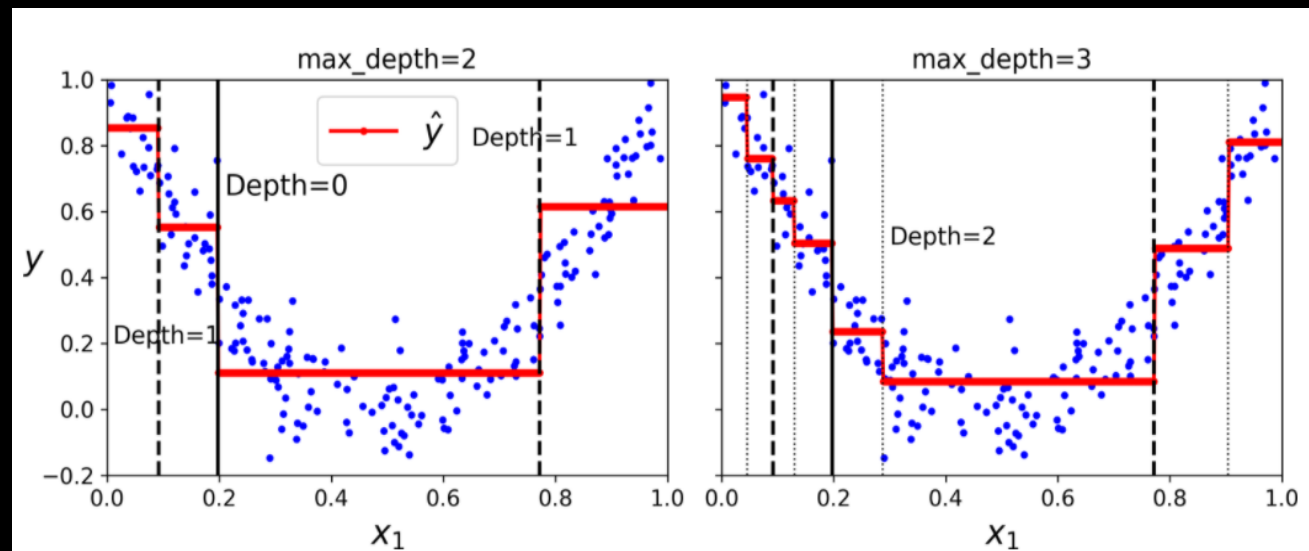
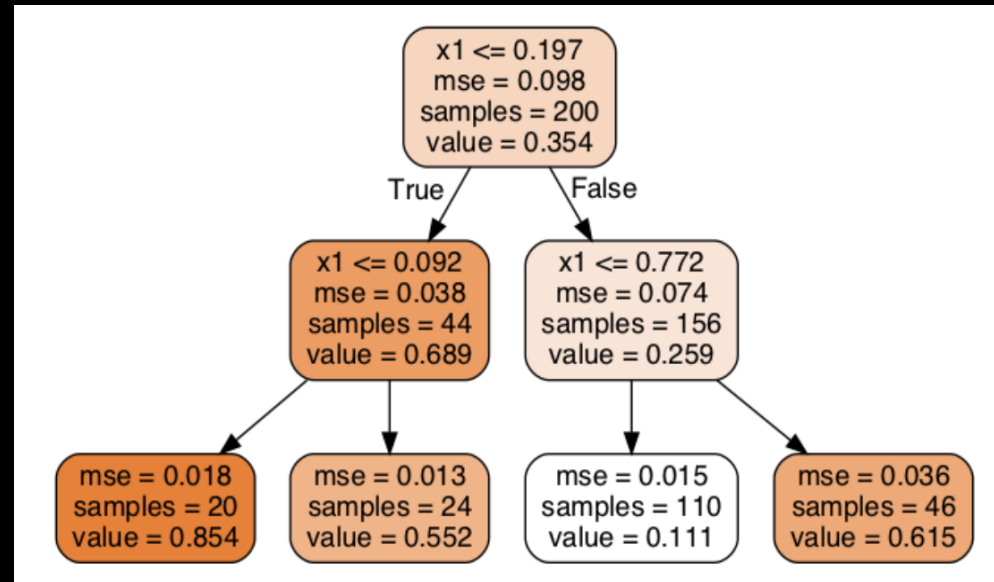
Se diferencian de los árboles de decisión en que, en lugar de predecir una clase en cada nodo, predicen un valor.

Esta predicción es el **valor objetivo promedio** de las instancias de entrenamiento asociadas con este nodo.

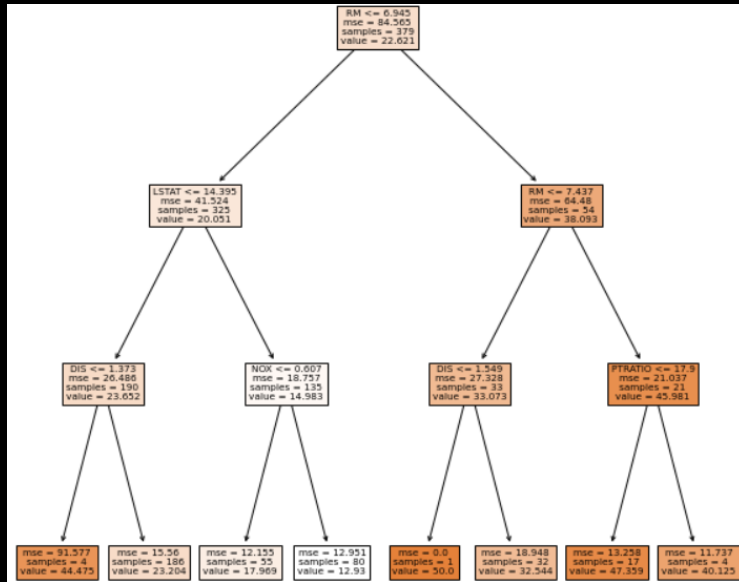
El algoritmo funciona casi de la misma manera, excepto que en lugar de intentar dividir el conjunto de entrenamiento de una manera que minimice la impureza, ahora intenta **minimizar la variación de los datos (MSE)**.

$$J(k, t_k) = \frac{m_{\text{left}}}{m} \text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m} \text{MSE}_{\text{right}} \quad \text{where} \quad \begin{cases} \text{MSE}_{\text{node}} = \sum_{i \in \text{node}} (\hat{y}_{\text{node}} - y^{(i)})^2 \\ \hat{y}_{\text{node}} = \frac{1}{m_{\text{node}}} \sum_{i \in \text{node}} y^{(i)} \end{cases}$$

Ejemplo



Feature importance



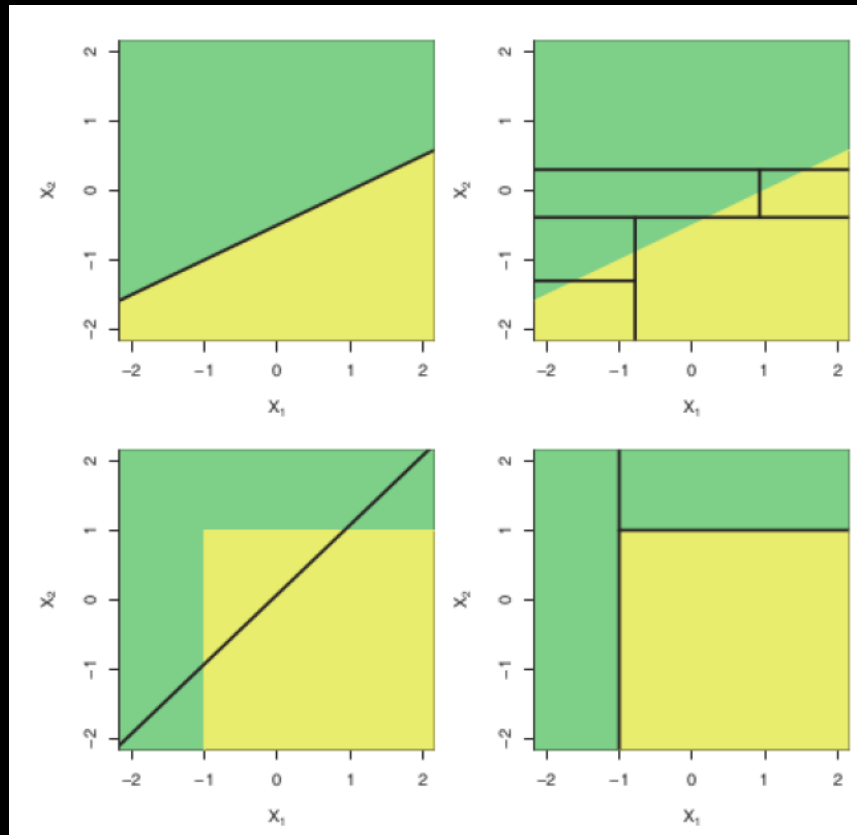
Feature importance	
RM	0.656763
LSTAT	0.226307
DIS	0.078884
NOX	0.031580
PTRATIO	0.006466
CRIM	0.000000
ZN	0.000000
INDUS	0.000000
CHAS	0.000000
AGE	0.000000
RAD	0.000000
TAX	0.000000
B	0.000000

- El árbol de decisión es un modelo intrínsecamente explicable a través del atributo `features_importance_`
- Este atributo devuelve, una vez entrenado el modelo, la importancia relativa (de 0 a 1) de cada variable de entrada a efectos de predecir el target. Cuanto más cerca de 1, mayor es la importancia de la variable a estos efectos.
- La importancia relativa de cada variable se calcula como como la disminución de la impureza del nodo de decisión ponderada por la probabilidad de alcanzar ese nodo.
- Este mismo enfoque, con algún ajuste, se puede utilizar con algoritmos que utilizan conjuntos de árboles de decisión, como el random forest y los algoritmos de gradient boosting.

¿Cuándo usar árboles de decisión?

Trees vs Linear Regression

La importancia del análisis exploratorio



Ventajas e Inconvenientes

Ventajas

- Computacionalmente son eficientes
- Requiere muy poco preprocesado (no hace falta estandarizar ni normalizar valores)
- Son robustos frente a outliers.
- Resistentes a variables irrelevantes.
- Si son cortos, son muy sencillos de explicar e interpretables. Manejan un solo parametro (tree size).
- Se pueden visualizar y de forma muy intuitiva.
- Se puede usar con pocos o muchos datos de entrenamiento.

Inconvenientes

- No es muy preciso.
- **Inestabilidad.** Muy sensible a pequeñas variaciones en los datos de entrenamiento.
- Muy propenso al **overfitting** si no se controla el tamaño del árbol.
- Árboles grandes son difíciles de interpretar.
- Cada split depende del anterior, por lo que **los errores** cometidos se propagan.

A efectos prácticos...

- Algoritmo con una precisión muy mejorable por modelos más complejos como los ensembles o las redes neuronales.
- Sin embargo, es interesante conocerlo y saber interpretarlo por dos razones:
 - Es un **buen modelo para resolver problemas de negocio**, en la medida en que obtenga un buen resultado para el conjunto de datos analizados.
 - Es sencillo de explicar (salvo que sea muy profundo).
 - Se puede visualizar de forma bastante intuitiva, lo que ayuda mucho a su comprensión por los usuarios de negocio.
 - Es un modelo intrínsecamente explicable a través del atributo *features_importance_*.
 - Es **necesario para comprender algoritmos más complejos** consistentes en la combinación de árboles de decisión, como el random forest, gradient boosting o XG boost.

iDemo time!

<http://www.r2d3.us/visual-intro-to-machine-learning-part-1/>