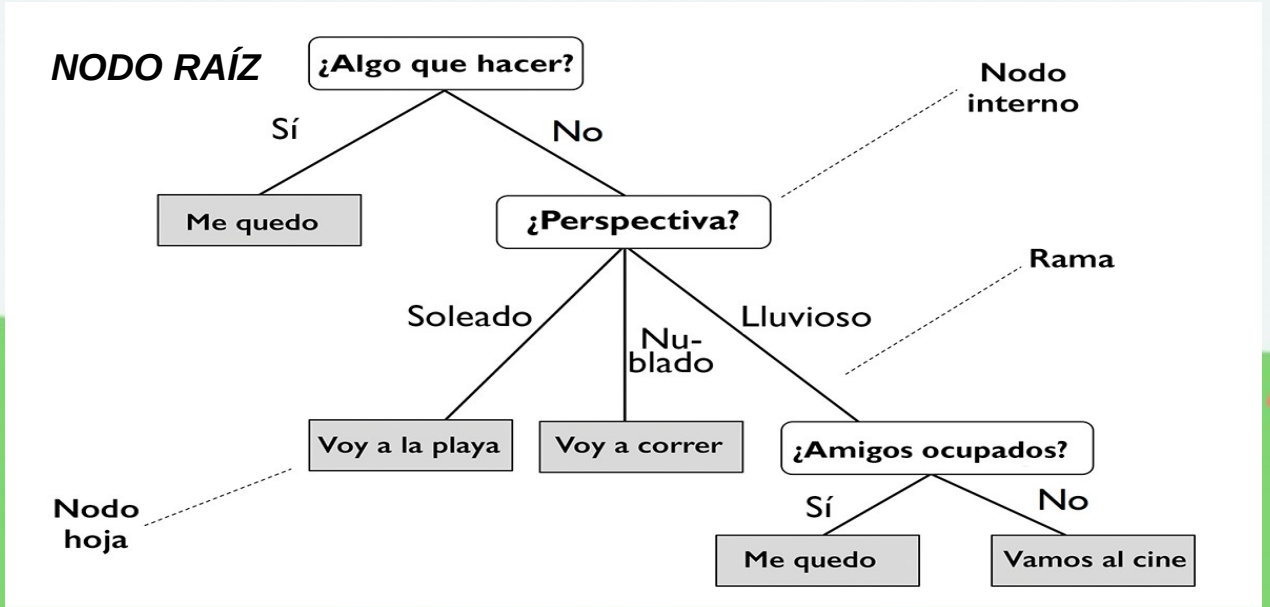
ÁRBOLES DE DECISIÓN

QUÉ SON LOS ÁRBOLES DE DECISIÓN?

- CONJUNTO DE MÉTODOS NO PARAMÉTRICOS DE CLASIFICACIÓN Y REGRESIÓN PARA APRENDIZAJE SUPERVISADO
- HACEMOS UNA DESCOMPOSICIÓN DE LOS DATOS MEDIANTE LA TOMA DE DECISIONES BASADAS EN LA FORMULACIÓN DE UNA SERIE DE PREGUNTAS
- SE PUEDEN APLICAR A VARIABLES CONTINUAS O CATEGÓRICAS YA QUE SON DECISIONES TIPO

'SI' -» 'ENTONCES' (IF / ELSE)



CÓMO CLASIFICA EL ÁRBOL?

- Desde la raíz, se empieza con la característica que produce la mayor Ganancia de Información y se hace la primera división
- De manera iterativa, el proceso se repite en cada NODO HIJO hasta que las hojas sean PURAS, es decir, que las muestras en cada hoja pertenezcan a la misma clase
- Esto puede producir un árbol muy profundo, que SOBREAJUSTE -» PODA

Qué es la Ganancia de Información?

- La Ganancia de Información es la función objetivo a maximizar mediante el algoritmo de aprendizaje del árbol, para dividir los nodos en las características más significativas.
- Se define a la Ganancia de Información (IG) como la diferencia entre la Impureza del nodo padre y la suma de las Impurezas de los nodos hijos.
- $IG(Dp,f) = I(Dp) - \sum Nj/Np * I(Dj)$
- Donde 'f' es la característica a considerar,
- 'Dp' y 'Dj' son el conjunto de datos del nodo padre 'p' y del nodo hijo 'j',
- 'I' es la medida de la Impureza,
- 'Np' es el total de muestras en el nodo padre 'p',
- 'Nj' es el número de muestras en el nodo hijo 'j'
- Por simplicidad (y para reducir el costo computacional), la mayoría de las librerías como Sklearn implementan Árboles de Decisión binarios, donde cada nodo padre se divide en dos nodos hijos, Dleft y Dright, y la IG queda:
- $IG(Dp,f) = I(Dp) - Nleft/Np * I(Dleft) - Nright/Np * I(Dright)$

Qué es la Impureza?

- La Impureza es la inhomogeneidad presente en los nodos luego de la división
- Un nodo u hoja son puros si todas las muestras son de la misma clase
- Hay varias maneras de medir la impureza:
- * ***IMPUREZA de GINI***: se puede entender como un criterio para minimizar la probabilidad de clasificación errónea

Se define como $Ig(t) = \Sigma\ p(i|t) * (1- p(i|t)) = 1 - \Sigma\ p(i|t)**2$

(suma sobre i hasta el número de clases 'c' en el nodo 't')

La Impureza de Gini es máxima cuando las clases están perfectamente mezcladas

- * ***ENTROPÍA***: el concepto viene de la Termodinámica y está relacionada con el órden de un sistema físico, cuanto más ordenado el sistema la entropía es menor

Se define sólo para las clases no vacías como:

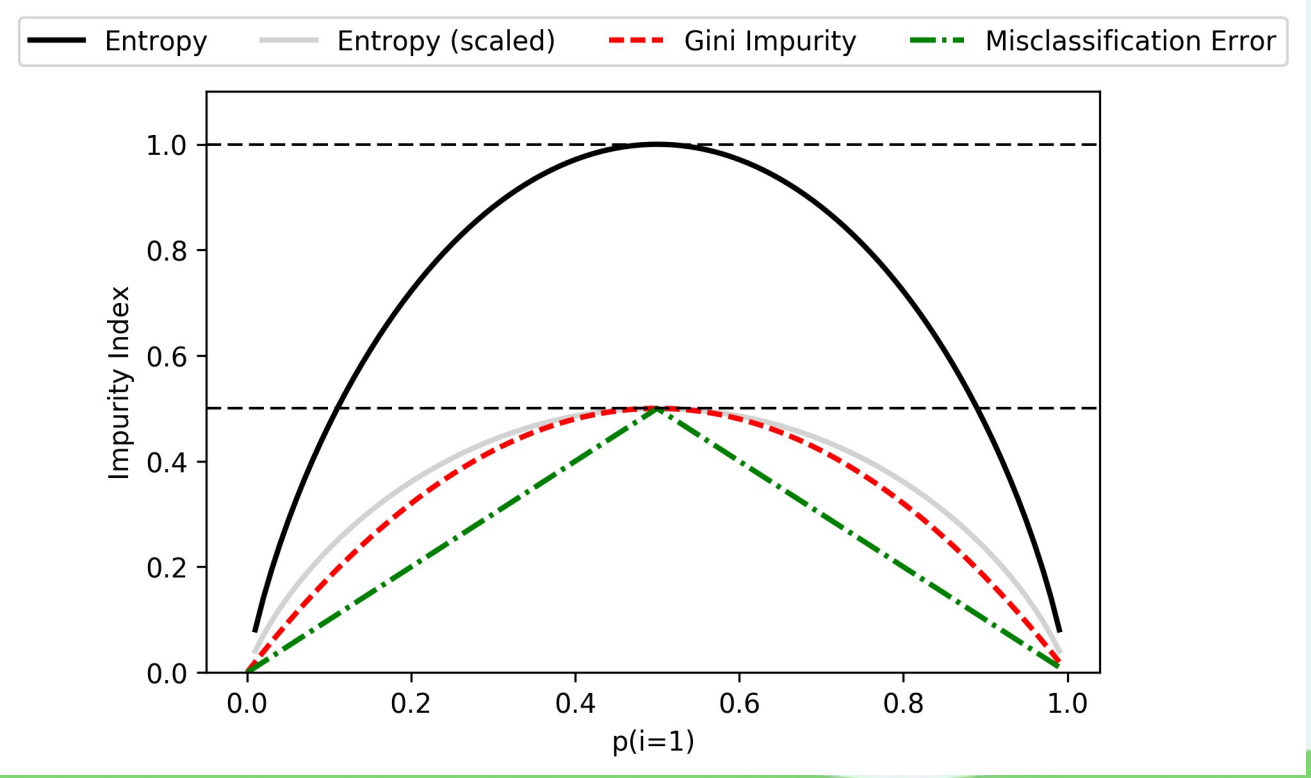
$$Ih(t) = -\Sigma\ p(i|t) * log2(p(i|t))$$

(suma sobre i hasta el número total de clases 'c' en el nodo 't')

La entropía es cero cuando todas las muestras pertenecen a la misma clase y máxima cuando las distribución de clases es uniforme.

Puede pensarse que maximiza la información mutua en el árbol

- ***ERROR de CLASIFICACIÓN:*** criterio muy útil para podar un árbol, no tanto para hacerlo crecer pues es menos sensible a los cambios de las probabilidades de clase de los nodos
  - Se define como ***Ie = 1 – max{p(i|t)}***
- ***Gini vs Entropía:*** * Gini tiende a poner cada clase de un lado del árbol, Entropía produce árboles más balanceados
- * Aunque Gini es más rápido, ambos criterios producen árboles similares, por lo que a veces hay que optar por hacer poda
  - * Gini produce sesgo cuando hay muchas clases y atributos multivaluados

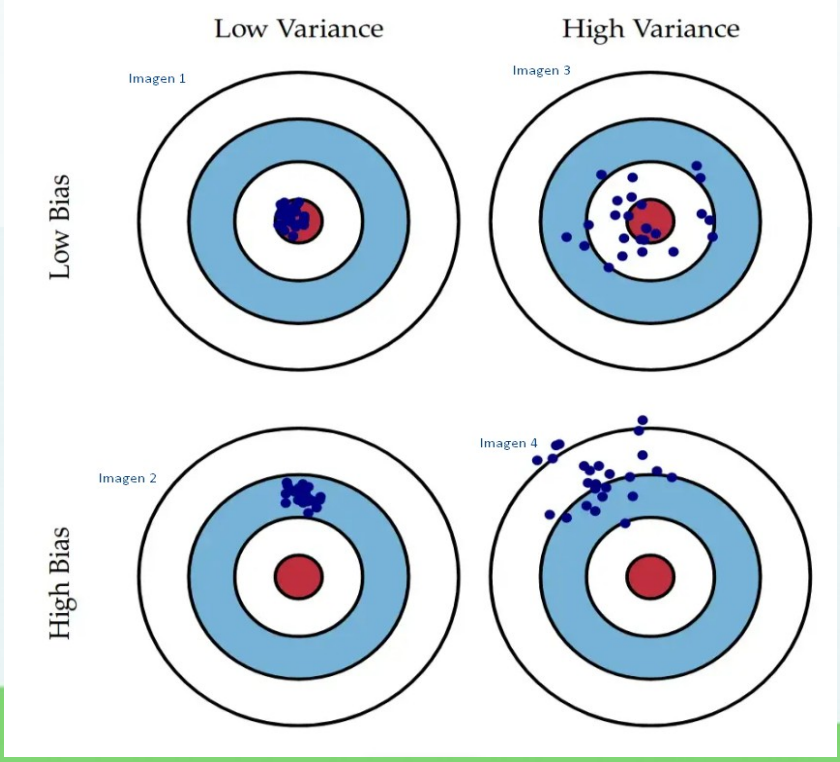


VENTAJAS

- Algoritmo fácil de comprender, interpretar y visualizar
- Se puede usar como REGRESOR y como CLASIFICADOR
- Puede manejar variables continuas y categóricas
- No requiere escalado de características
- Maneja eficientemente problemas no lineales
- Es robusto frente a outliers y valores ausentes
- Tiempo de entrenamiento menor que el empleado por otros algoritmos más complejos
- No hace suposiciones sobre los datos (método no paramétrico)

DESVENTAJAS

- Tiene fuerte tendencia al sobreajuste si el árbol crece por completo
- Padece de alta varianza (debido a la tendencia al overfitting)
- Inestable frente a nuevas muestras agregadas al dataset, ya que el árbol se recalcula
- No recomendable para grandes datasets por el costo computacional, ya que el árbol se vuelve muy complejo y lento para crecer



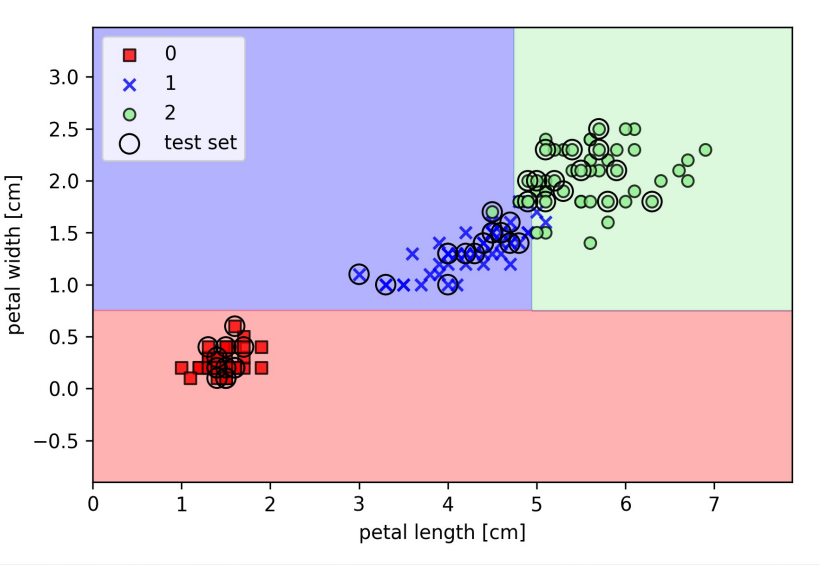
Algoritmos de Árbol

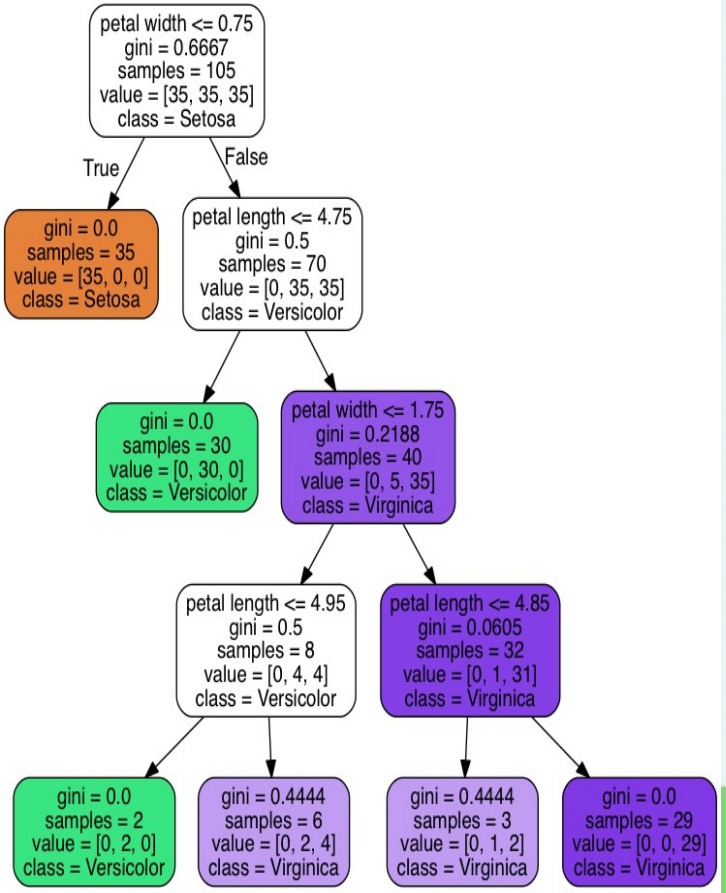
- Entre los algoritmos de árbol (tienen usos variados), se destacan:
- ID3 (Iterative Dichotomiser, 1986): sólo características categóricas, usa Ganancia de Información para decidir. Puede manejar más de 2 nodos hijos. Produce árboles poco profundos, pues ejerce poda. Los atributos con alta Ganancia de Información son situados cerca de la raíz
- C4.5 (sucesor de ID3): acepta también variables numéricas. Convierte la salida del árbol en un cjto de reglas IF/THEN. También puede podar para evitar sobreajuste
- CART (Classification And Regression Trees): usa Índice de Gini como medida de selección de atributos. Construye sólo árboles binarios. Soporta regresión (predice valores continuos)

Es un algoritmo voraz que busca la óptima separación al máximo nivel y repite el mismo procedimiento de manera iterativa para los sgtes niveles, mientras verifica si la separación da la mínima impureza. La solución no siempre es óptima pues requeriría un tiempo del O(N*logN*D) para obtenerla.

class sklearn.tree.**DecisionTreeClassifier**(*, *criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0*)

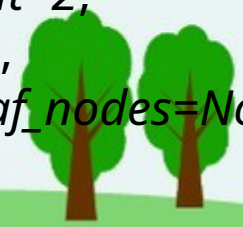
```
>>> from sklearn.datasets import load_iris
>>> from sklearn import tree
>>> iris = load_iris()
>>> X, y = iris.data, iris.target
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, y)
```





Los árboles pueden crear límites de decisión muy complejos dividiendo el espacio de características en rectángulos, pero hay que ir con cuidado pues más profundo el árbol, más complejo el límite y más propenso al sobreajuste.
La estructura del árbol se puede exportar en archivo .dot para ver con GraphViz.
Con .plot_tree podemos graficar el árbol para comprender cómo tomó las decisiones

class sklearn.tree.**DecisionTreeRegressor**(*, *criterion='squared_error',*
splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, ccp_alpha=0.0)



Cómo evitar el sobreajuste?

Al crear un árbol y dejarlo crecer en toda su extensión, se aprecia que en cada nodo hoja hay una alta homogeneidad, pues las muestras quedan bien clasificadas. Esta homogeneidad puede provocar sobreajuste, es decir, que el árbol no generalice la clasificación a muestras que no 'vió' durante el entrenamiento.

Como dijimos, hay varias estrategias para evitar el overfitting que, incluso, se pueden combinar entre sí.

Una es limitar la **profundidad del árbol** poniendo un valor para **max_depth**. El problema puede ser que el árbol haga 'underfitting' si el desarrollo no es suficiente.

Otra estrategia es fijar un **número mínimo de muestras por hoja** (min_samples_leaf) >1. Si éste es muy grande, también puede ir al subajuste.

Las estrategias anteriores pueden ser previas a construir el modelo.

Existe otra, llamada '***poda***' (pruning), que deja crecer el árbol y luego va eliminando algunos nodos, y para ello puede usar los 'errores de clasificación'. Hay dos modos:

- **Post-pruning**: se hace después de la construcción del árbol cuando muestra overfitting
- **Pre-pruning**: antes de construir el árbol, mediante un método llamado cost_complexity_pruning_path(*X*, *y*, *sample_weight=None*), que permite determinar un parámetro ccp_alpha que se ingresa en la instancia del clasificador. Ver
- **https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html**

DUDAS O CONSULTAS? MUCHAS GRACIAS POR ESCUCHAR!

FUENTES:

- **'Python Machine Learning', Sebastian Raschka, Vahid Mirjalili, Marcombo, 2da edición**
- https://scikit-learn.org/stable/modules/tree.html