Árboles de decisión

Aprendizaje Automático

Juan David Martínez
jdmartinev@eafit.edu.co

Agenda

- Árboles de decisión
- Medidas de impureza
- Propiedades de los árboles de decisión
- Regularización de árboles de decisión
- Hands on

Árboles de decisión

¿Por qué árboles de decisión?

A pesar de que las redes neuronales han hecho avanzar áreas como visión por computador, procesamiento de lenguaje natural, procesamiento de audio, entre muchas otras, cuando tenemos datos tabulares, los modelos basados en árboles son la mejor opción.

En estos casos, los modelos basados en árboles son:

- Más rápidos
- Más fáciles de ajustar y de evitar el sobre-entrenamiento (usando ensambles)
- Más fáciles de entender

Árboles de decisión

UNIVERSITY OF LIÈGE
Faculty of Applied Sciences
Department of Electrical Engineering & Computer Science

PhD dissertation

UNDERSTANDING RANDOM FORESTS

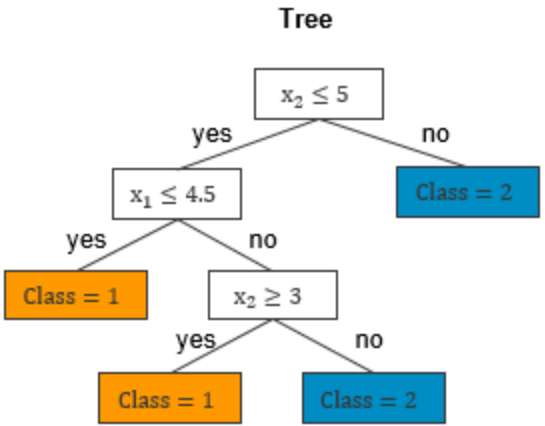
FROM THEORY TO PRACTICE

---

by GILLES LOUPPE

¿Qué es un árbol de decisión?

Un **árbol de decisión** es una serio de preguntas si/no que se hacen de forma secuencial en los datos

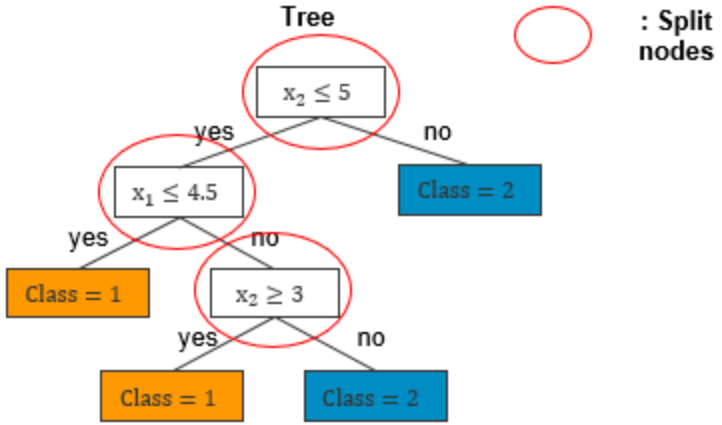


Dataset

x_1	x_2	y
3.5	2	1
5	2.5	2
1	3	1
2	4	1
4	2	1
6	6	2
2	9	2
4	9	2
5	4	1
3	8	2

¿Qué es un árbol de decisión?

Un **árbol de decisión** es una serio de preguntas si/no que se hacen de forma secuencial en los datos

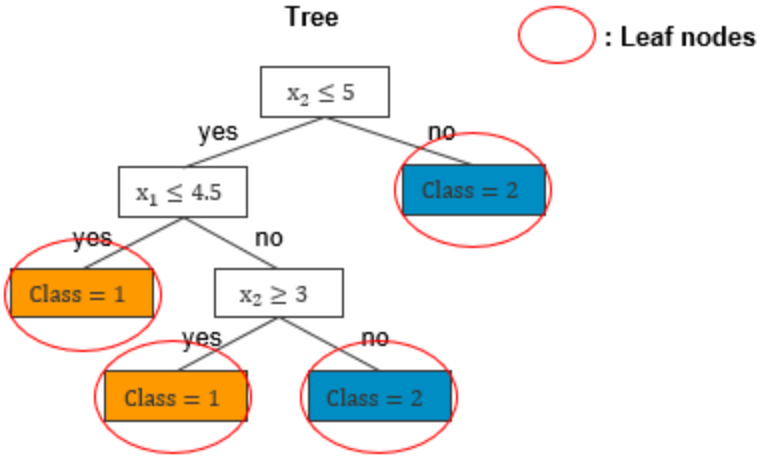


Dataset

x_1	x_2	y
3.5	2	1
5	2.5	2
1	3	1
2	4	1
4	2	1
6	6	2
2	9	2
4	9	2
5	4	1
3	8	2

¿Qué es un árbol de decisión?

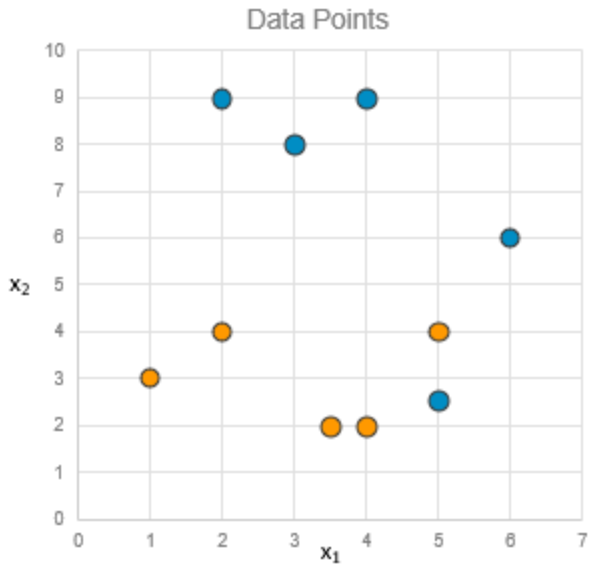
Un **árbol de decisión** es una serio de preguntas si/no que se hacen de forma secuencial en los datos



Dataset

x_1	x_2	y
3.5	2	1
5	2.5	2
1	3	1
2	4	1
4	2	1
6	6	2
2	9	2
4	9	2
5	4	1
3	8	2

¿Qué es un árbol de decisión?

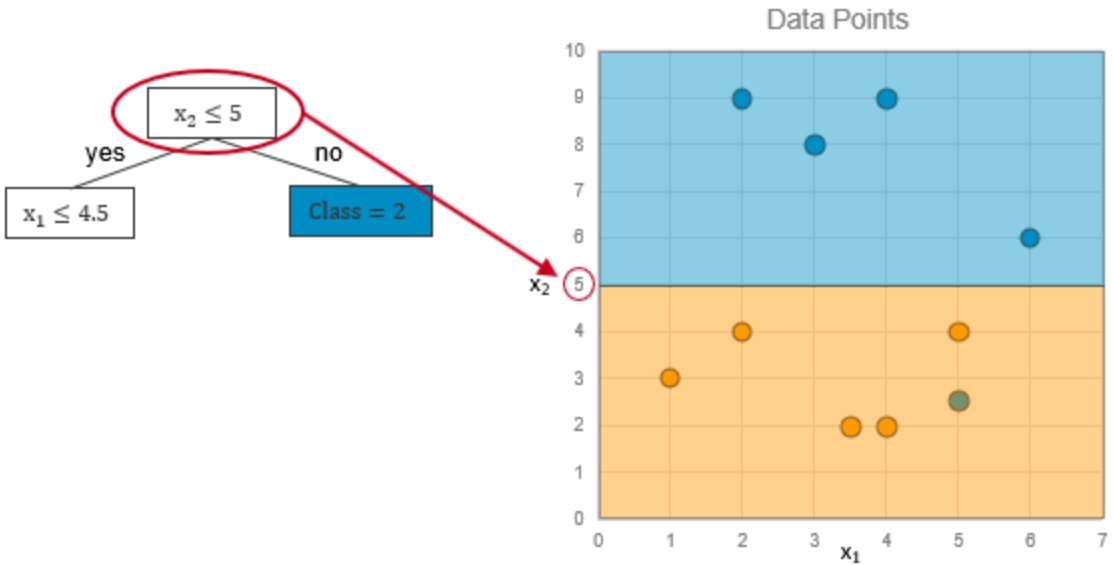


Dataset

x_1	x_2	y
3.5	2	1
5	2.5	2
1	3	1
2	4	1
4	2	1
6	6	2
2	9	2
4	9	2
5	4	1
3	8	2

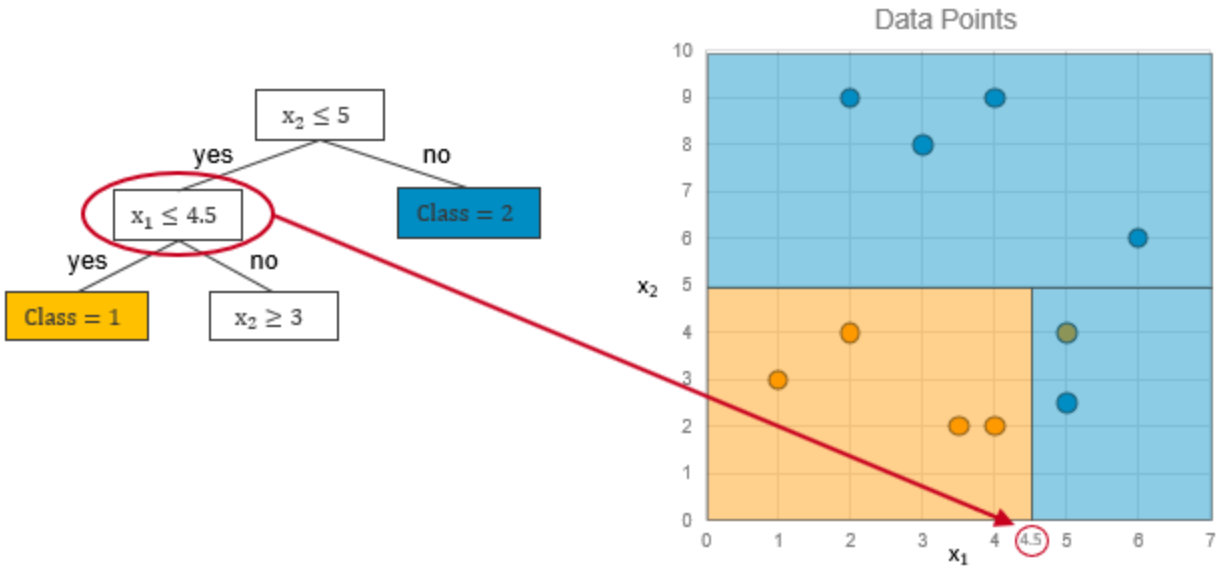
¿Qué es un árbol de decisión?

¿Qué característica (x_1 o x_2) debemos dividir de forma que las clases se separen lo mejor posible?



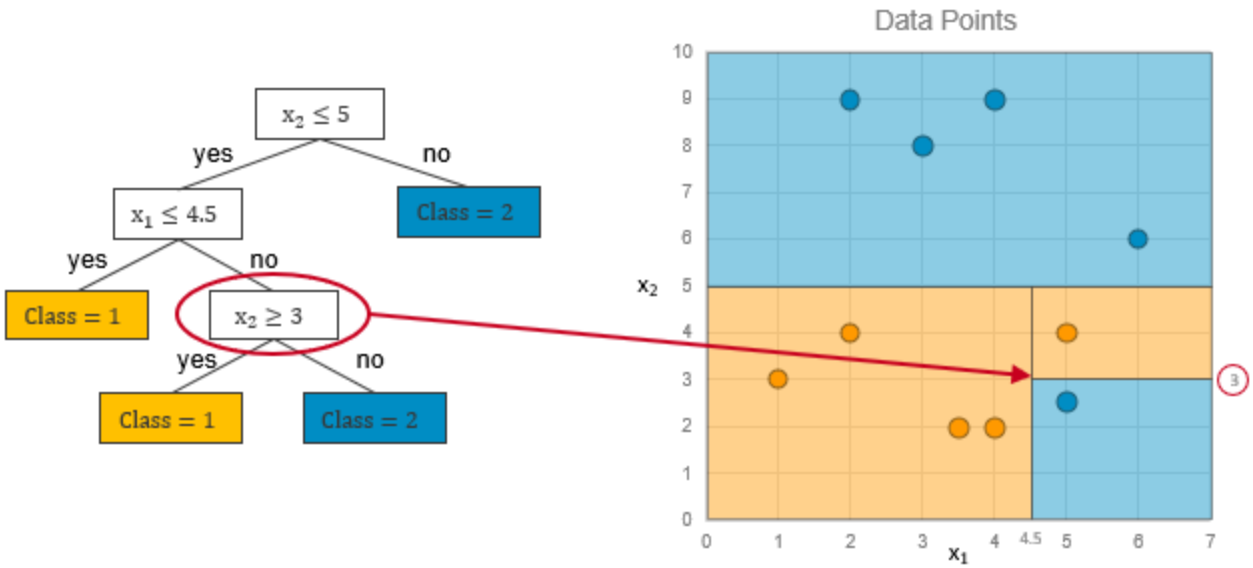
¿Qué es un árbol de decisión?

¿Qué característica (x_1 o x_2) debemos dividir de forma que las clases se separen lo mejor posible?



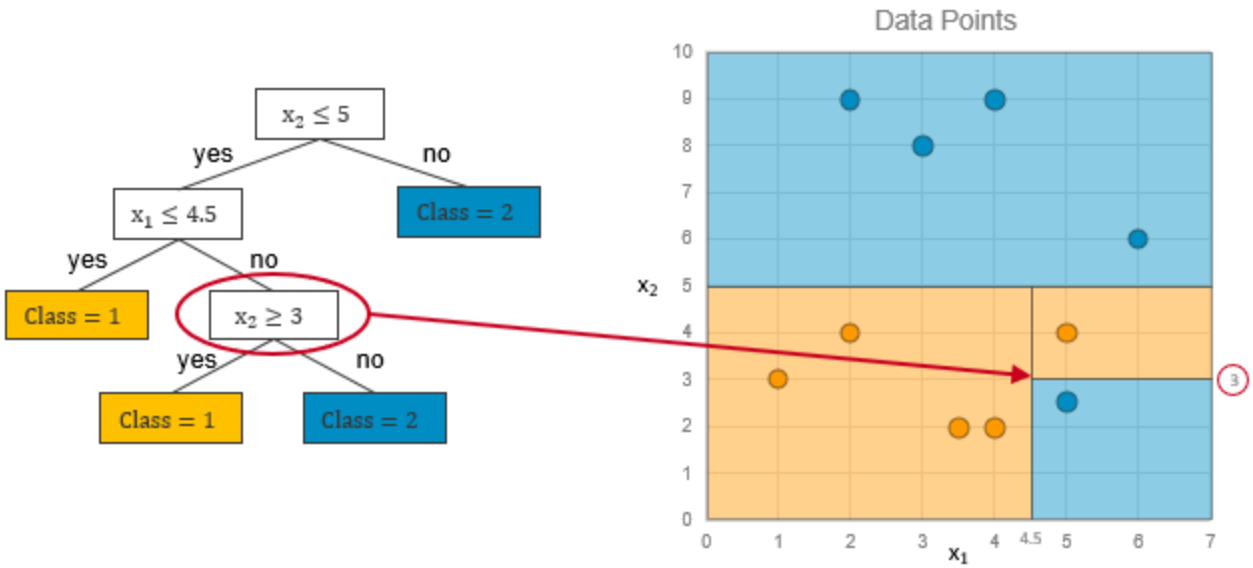
¿Qué es un árbol de decisión?

¿Qué característica (x_1 o x_2) debemos dividir de forma que las clases se separen lo mejor posible?



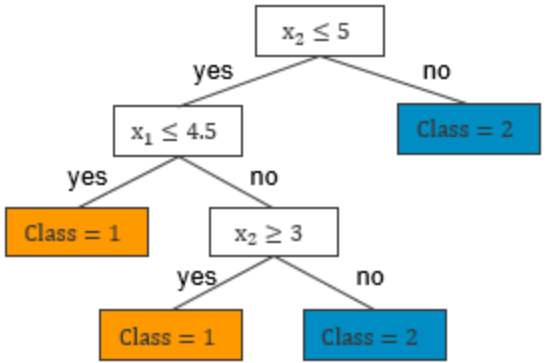
¿Qué es un árbol de decisión?

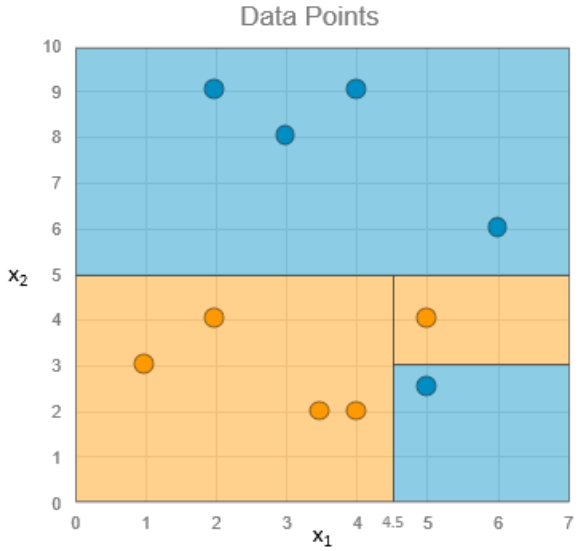
¿Qué característica (x_1 o x_2) debemos dividir de forma que las clases se separen lo mejor posible?



¿Qué es un árbol de decisión?

$$f(x_1 = 7.5, x_2 = 5) = 1$$





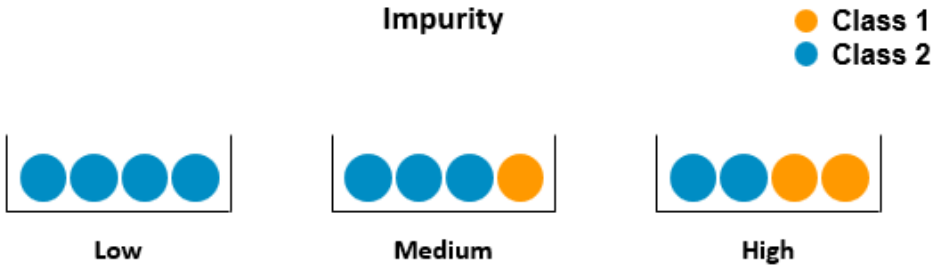
¿Qué es un árbol de decisión?

- Secuencia de preguntas si/no en una sola característica (x_1>3)
- Modelos aprenden escogiendo de forma recurrente, en cada nodo, la partición que maximiza el incremento de “pureza”
- Se termina el proceso con un criterio de parada:
  - Máxima profundidad
  - Máximo número de hojas
  - Mínima cantidad de muestras en un nodo hoja
  - Nivel de pureza deseado en una hoja

Funciones de impureza

La **impureza** es el concepto principal para entender el proceso de los árboles

Sirven para responder la pregunta: ¿cuál es la mejor partición?



Funciones de impureza para clasificación

Existen principalmente dos opciones:

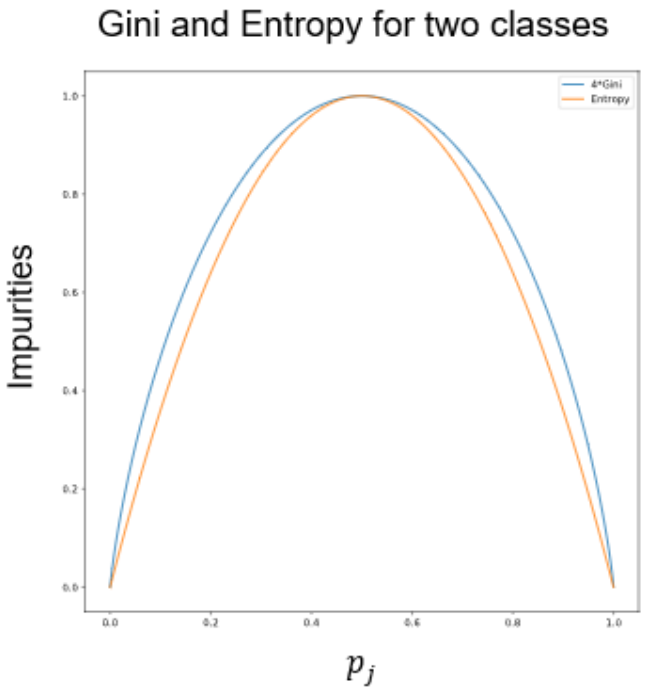
- Entropía:

$$i(p_1, \ldots, p_k) = -\sum_{j=1}^{k} p_j \log_2(p_j)$$

- Gini:

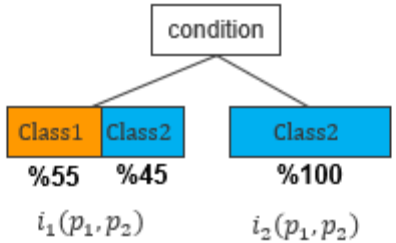
$$i(p_1, \ldots, p_k) = \sum_{j=1}^{k} p_j(1 - p_j)$$

Son extremadamente similares, no presentan diferencia en el rendimiento del modelo final. Dado que el logaritmo es más complejo de calcular, generalmente se utiliza **Gini index**



Funciones de impureza para clasificación

$$i(p_1, \ldots, p_k) = \sum_{j=1}^{k} p_j(1 - p_j)$$



$$i_1(p_1, p_2) = 0.55 * (1 - 0.55) + 0.45 * (1 - 0.45)$$
$$= 0.495$$
$$i_2(p_1, p_2) = 0.0 * (1 - 0.0) + 1.0 * (1 - 1.0)$$
$$= 0.0$$

El Gini score resultante es la suma ponderada de los scores de cada nodo

$$i_{nodo}(p_1, p_2) = \frac{n_1}{n_1+n_2} i_1 + \frac{n_2}{n_1+n_2} i_2$$

Funciones de impureza para regresión

Para **regresión**, la medida de impureza más común es la varianza:

$$i(\mathbf{y}) = var(\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})2$$

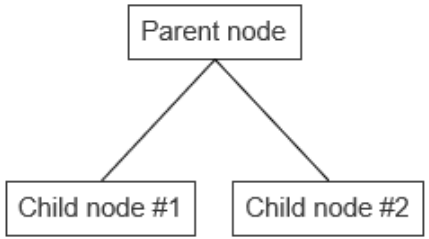
Otra medida común es el error absoluto promedio (MAE):

$$i(\mathbf{y}) = mae(\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}|y_i - \bar{y}|$$

Nota: No es necesario que las funciones sean diferenciables, así que podemos utilizar cualquier función.

Ganancia de información (IG)

Information Gain (IG) es la diferencia entre la impureza del nodo padre y la suma ponderada de las impurezas de los nodos hijos



$$IG(\mathbf{y}) = i(\mathbf{y}) - \frac{N_L i(\mathbf{y}_L) + N_R i(\mathbf{y}_R)}{N_L + N_R}$$

Algoritmo CART

Partimos del concepto de impureza de nodo.

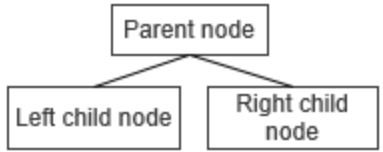
CART: **Classification** and **Regression Trees.** Construye un **árbol binario** (particiones si/no).

- Asuma que tenemos una matriz de datos **X** = ($x_{i,j}$), donde i se refiere a una muestra en particular y j a una característica en particular.
- Además, tenemos nuestra variable objetivo **y** que puede ser categórica o numérica.
- Finalmente, tenemos una forma de medir la impureza i(**y**) de una colección de "etiquetas".

Algoritmo CART

Iterar sobre cada característica j **y hacer una partición en cada valor** $x_{i,j}$

- Dividir la base de datos en dos partes (Left & Righr):
  - $(\boldsymbol{X}_L, \boldsymbol{y}_L)$ para puntos $\boldsymbol{x}_k$ donde $x_{k,j} < x_{i,j} \forall\, k = 1, \ldots, N$
  - $(\boldsymbol{X}_R, \boldsymbol{y}_R)$ para puntos $\boldsymbol{x}_k$ donde $x_{k,j} \geq x_{i,j} \forall\, k = 1, \ldots, N$
  - Cada conjunto tendrá N_L y N_R muestras, respectivamente.



- Seleccionamos la partición que maximiza la ganacia de información **IG**

$$IG(\mathbf{y}) = i(\mathbf{y}) - \frac{N_L i(\mathbf{y}_L) + N_R i(\mathbf{y}_R)}{N_L + N_R}$$

- Pasar de forma recursiva las bases de datos izquierda y derecha a los nodos hijos
- Si se cumple algún criterio, identificar el valor para devolver, usualmente, la moda o media.

Diferentes tipos de variables

El algoritmo **CART** usa un atributo numérico $x_{k,j}$. ¿Cuántas particiones diferentes tenemos que hacer para diferentes tipos de variables?

- **Variables categóricas:** Con **C** categorías, existen $2^{c-1} - 1$ particiones diferentes.

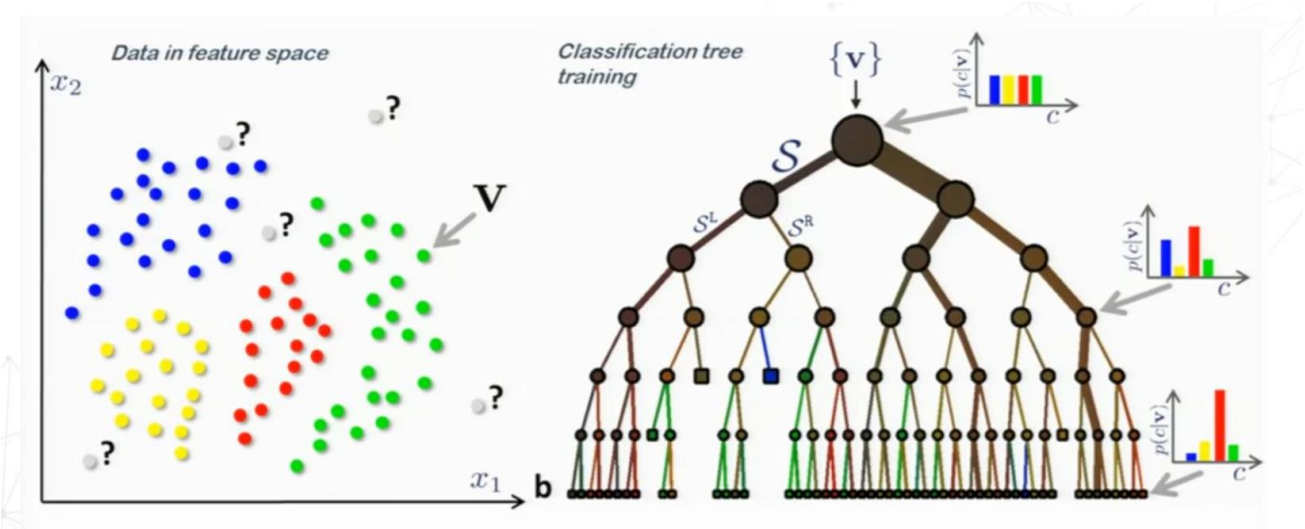
  [“Rojo”,”Verde”,”Azul”] genera tres posibilidades:

  - “Rojo”,[“Verde”,”Azul”]
  - “Verde”,[“Rojo”,”Azul”]
  - “Azul”,[“Verde”,”Rojo”]}
- **Variables ordinales o continuas:** Con **K** valores diferentes, existen **K-1** posibles particiones:
  - [12,15,18,20,23] genera 4 posibles particiones: 13.5, 16.5, 19.5, 21.5 (puntos medios)

Algoritmo CART

Generalmente se ordenan los datos antes de generar las particiones dado que actualizar los valores de impureza será más sencillo cuando se mueve un solo dato de $\boldsymbol{X}_R$ a $\boldsymbol{X}_L$

p50(Understanding Random Forest) .



Propiedades de los árboles de decisión

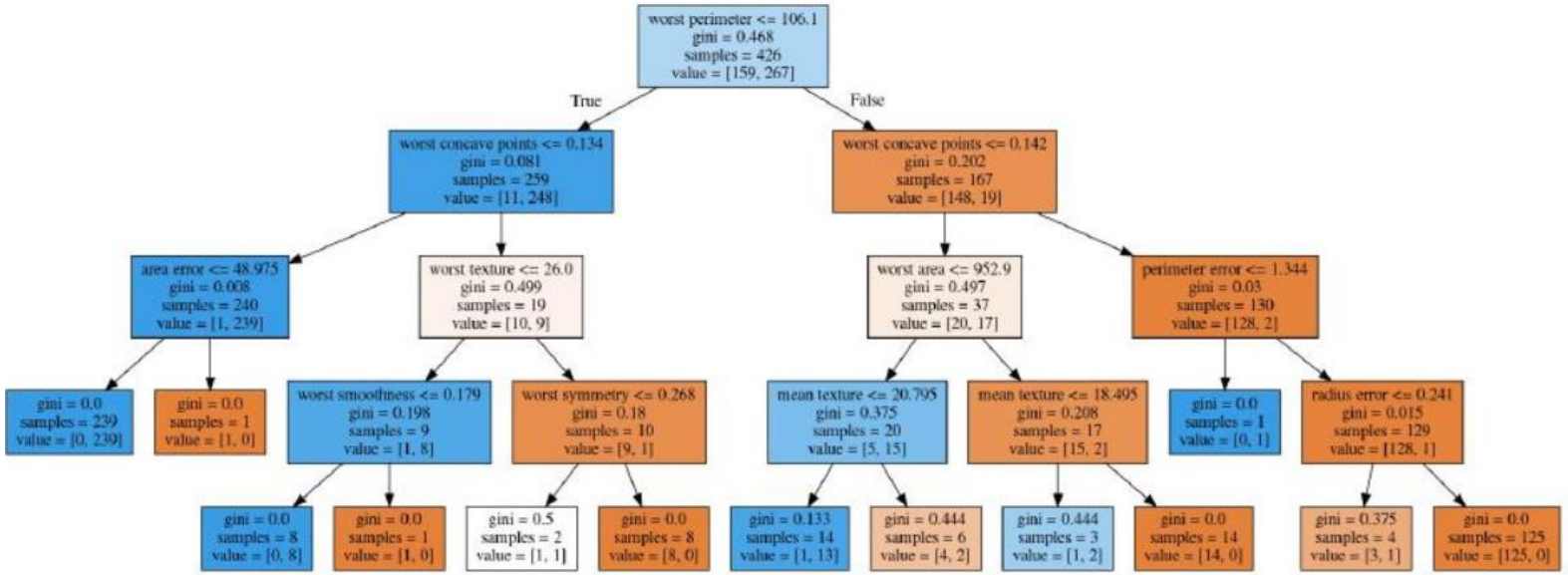
- No es necesario estandarizar las características.
- Suponga que $f(x)$ es una función incremental ($x > y \rightarrow f(x) > f(y)$), entonces la transformación de los datos con la función f no cambiará el árbol aprendido.

Regularización de los árboles de decisión

- En general, cuando queremos regularizar un modelo de ML, es clave acotar su complejidad.
- En árboles de decisión, la complejidad está relacionada con el número de hojas (nodos de decisión)
- Es posible generar un árbol de decisión que genere particiones para cada uno de los datos de entrenamiento (sobre-entrenamiento)
- Esto se puede controlar directamente (limitando el número de hojas) o imponiendo una profundidad máxima.
- Esta idea de utilizar un criterio de parada se conoce como pre-pruning. Funciona enteramente desde los árboles de decisión.
- ¿Cómo seleccionar los mejores parámetros?

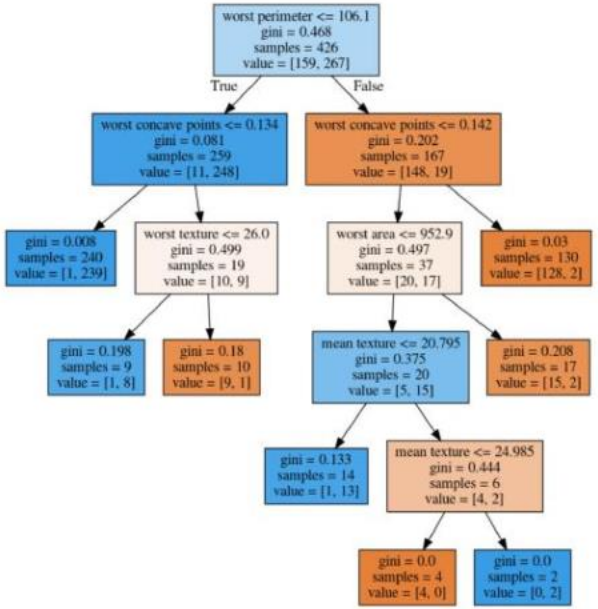
Regularización de los árboles de decisión

max_depth = 4

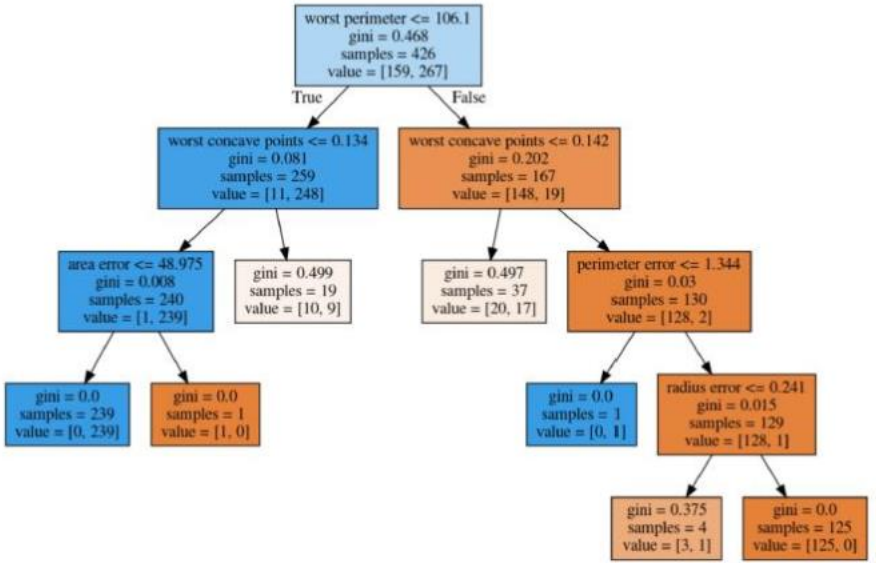


Regularización de los árboles de decisión

max_leaf_nodes = 8



min_samples_split = 50



Regularización de los árboles de decisión

- Otra forma de regularizar los árboles de decisión es haciendo un proceso inverso o **post-pruning.**
- Se comienza con un árbol muy complejo y se recorta.
- La forma más simple es reduciendo el error: iterar sobre nodos hoja removiendo aquellos que se pueden remover sin dañar la tasa de acierto en un conjunto de validación.
- Generalmente no se utiliza dado que los árboles de decisión se utilizan en conjunto con técnicas de ensamble.

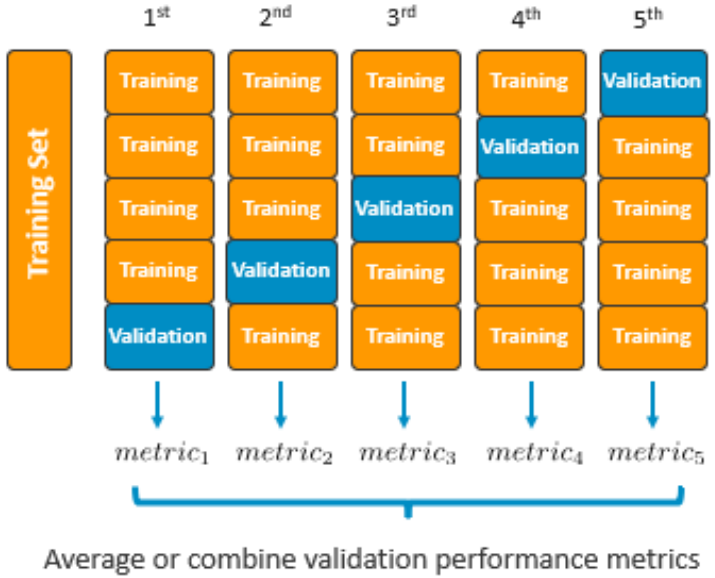
Árboles de decisión en sklearn

```
sklearn.tree.DecisionTreeClassifier(criterion='gini',
                                    max_depth=None,
                                    min_samples_split=2,
                                    min_samples_leaf=1,
                                    max_leaf_nodes=None,
                                    min_impurity_decrease=0.0)
```

```
sklearn.tree.DecisionTreeRegressor(criterion='mse',
                                   max_depth=None,
                                   min_samples_split=2,
                                   min_samples_leaf=1,
                                   max_leaf_nodes=None,
                                   min_impurity_decrease=0.0)
```

Sintonización de hiperparámetros

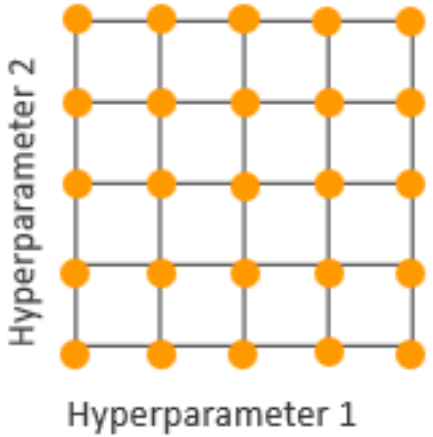
- Los **hiperparámetros** son parámetros de los algoritmos de ML que afectan su estructura y rendimiento.
- **Ejemplos:**
  - **KNN:** n_neighbors, metric
  - **Decision tres:** max_depth, min_simples_leaf, class_weight, criterion
- La sintonización de hiperparámetros busca la mejor combinación que maximiza el rendimiento del modelo.
- Cross-validación:



Búsqueda exhaustiva

- **GridSearchCV:** método de **sklearn** para hacer sintonización de hiperparámetros. Se utilizan los métodos .fit() y .predict()

  GridSearchCV(estimator, param_grid, scoring=None)



- **Ejemplo:** hiperparámetros para un árbol de decisión:

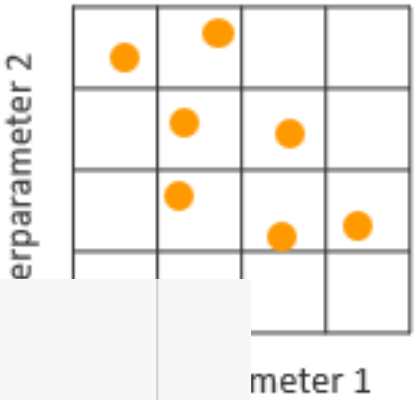
```
param_grid = {
    "criterion": ["gini", "entropy"],
    "max_depth": [None] + list(np.arange(2, 50)),
    "min_samples_split": list(np.arange(2, 50)),
    "min_samples_leaf": list(np.arange(1, 50)),
    "max_features": ["auto", "sqrt", "log2"] + list(np.arange(1, total_features+1)),
}
```

Búsqueda aleatoria

- **RandomizedSearchCV:** método de **sklearn** para hacer búsqueda aleatoria en los hiperparámetros. Se utilizan los métodos .fit() y .predict()

  RandomizedSearchCV(estimator, param_distributions, scoring= None)

- **Ejemplo:** hiperparámetros para un árbol de decisión:




```
param_dist = {
    "criterion": ["gini", "entropy"],
    "max_depth": [None] + list(np.arange(2, 50)),
    "min_samples_split": list(np.arange(2, 50)),
    "min_samples_leaf": list(np.arange(1, 50)),
    "max_features": ["auto", "sqrt", "log2"] + list(np.arange(1, total_features+1)),
}
```

Búsqueda bayesiana

- La búsqueda basada en optimización bayesiana genera modelos probabilísticos basándose en evaluaciones previas de los hiperparámetros.
- Intenta crear un balance entre exploración (hiperparámetros con alta incertidumbre) y explotación (hiperparámetros con alta probabilidad de ser óptimos).
- Optuna, AWS SageMaker usan este tipo de optimización.

https://optuna.org/#code_examples