Árboles de decisión

Dr. Raimundo Sánchez

raimundo.sanchez@uai.cl

@raimun2

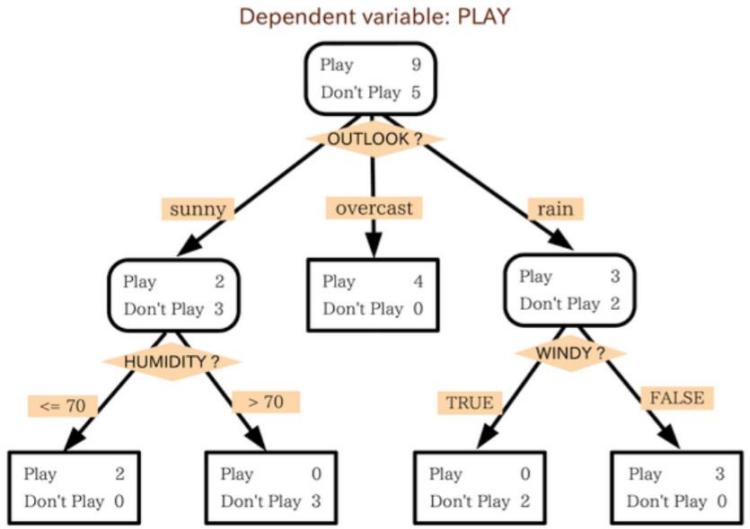
Árbol de decisión

Un árbol de decisión es una estructura similar a un diagrama de flujo en la que:

- cada nodo interno representa una "prueba" en un atributo,
- cada rama representa el resultado de la prueba
- cada nodo hoja representa una etiqueta de clase

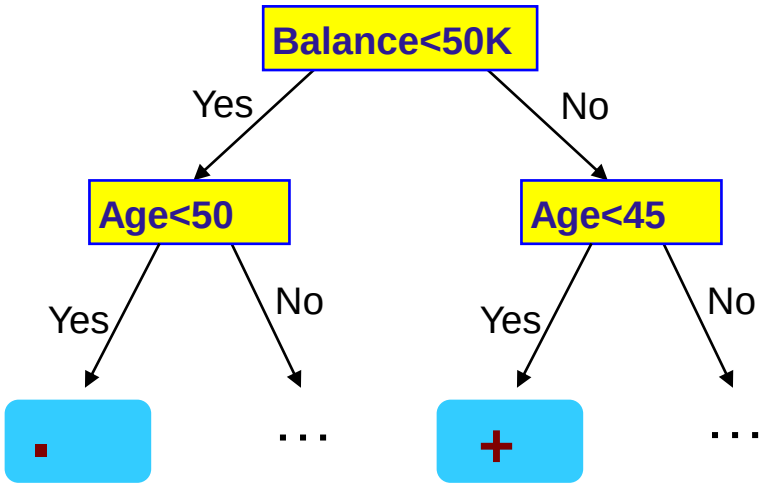
Dado un punto de datos **x**, la salida del modelo es la probabilidad de que **x** pertenezca a una clase específica

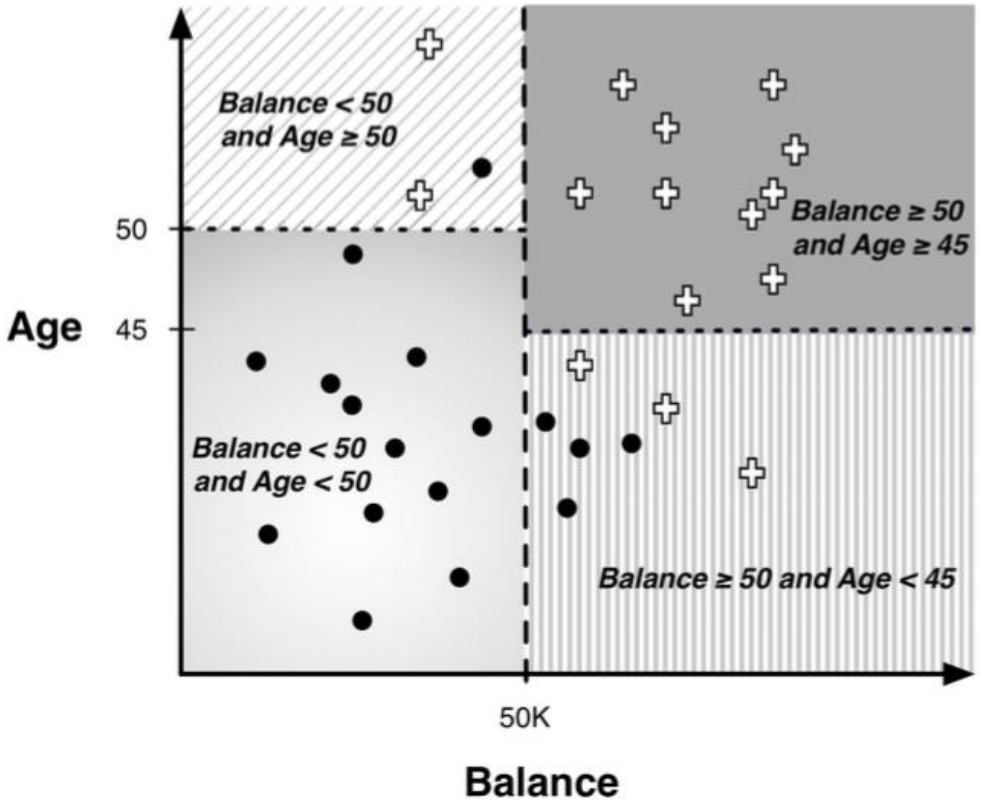
Day	outlook	Temp	Humidity	Windy	Play
1	sunny	85	85	no	**NO**
2	sunny	80	90	yes	**NO**
3	overcast	83	86	no	**YES**
4	rainy	70	96	no	**YES**
5	rainy	68	80	no	**YES**
6	rainy	65	70	yes	**NO**
…	…	…	…	…	…
14	sunny	72	95	no	**NO**



Definición

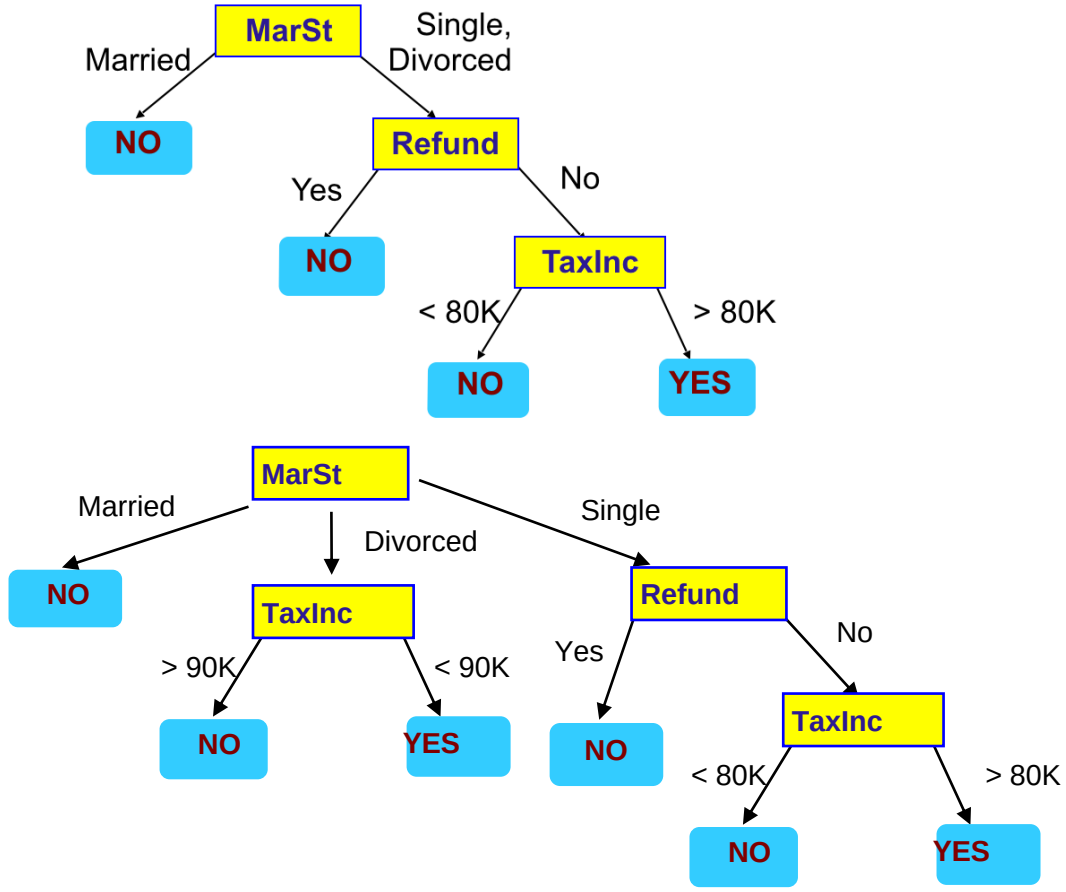
Un árbol de decisión segmenta los datos de entrada utilizando secciones rectangulares del espacio

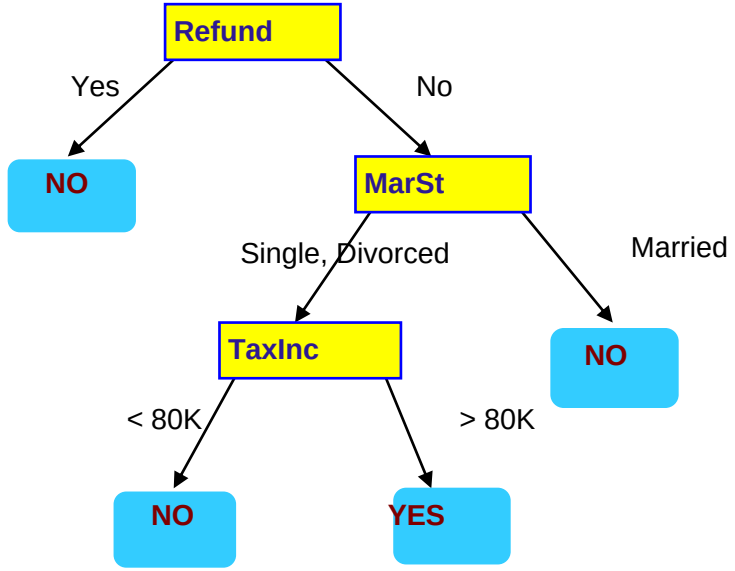




Redundancias

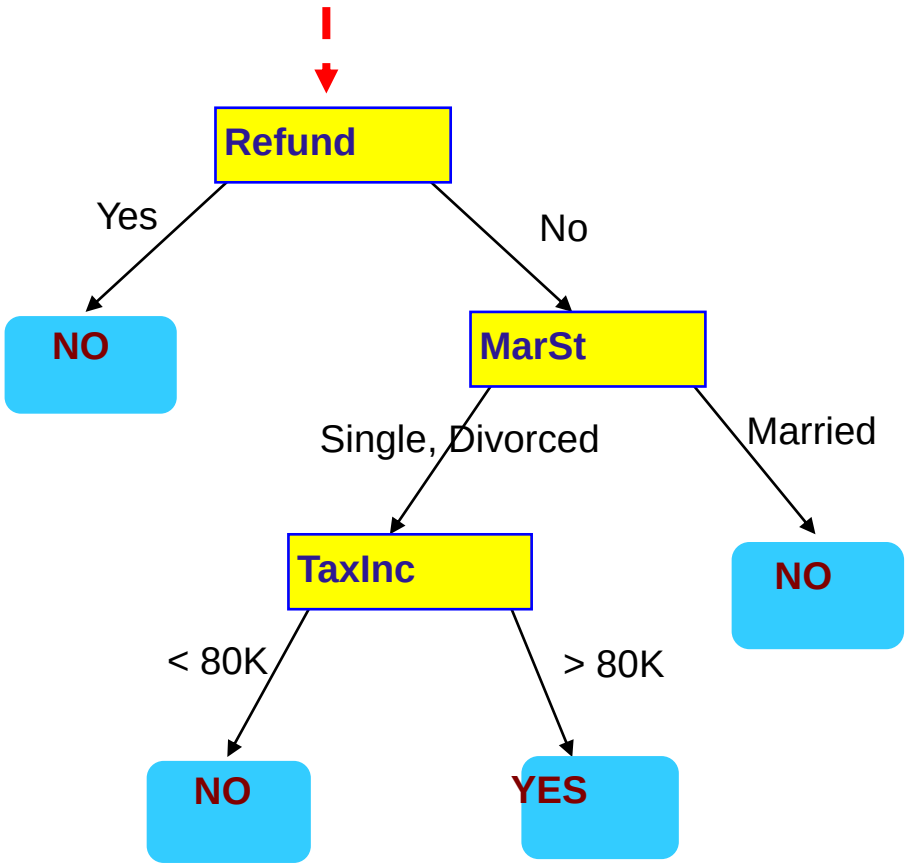
Dado un conjunto de datos, hay varios árboles de decisión que pueden clasificar los datos.





Predicciones

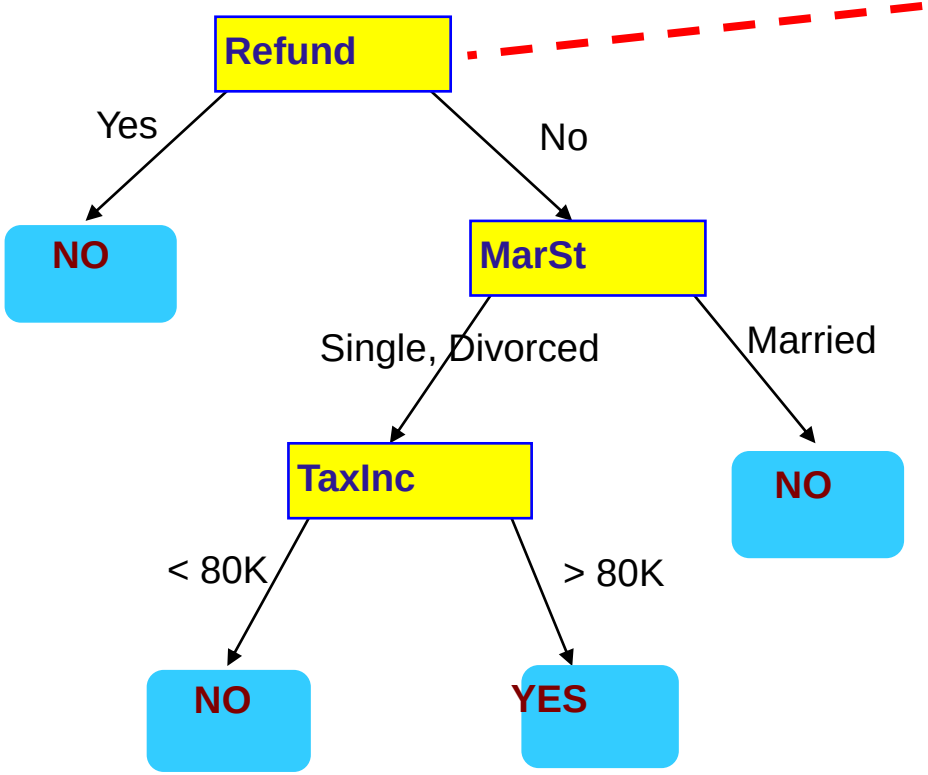
Comience desde la raíz del árbol.



Test Data

refund	Marital status	Taxablei ncome	Tax evasion
no	Married	80K	?

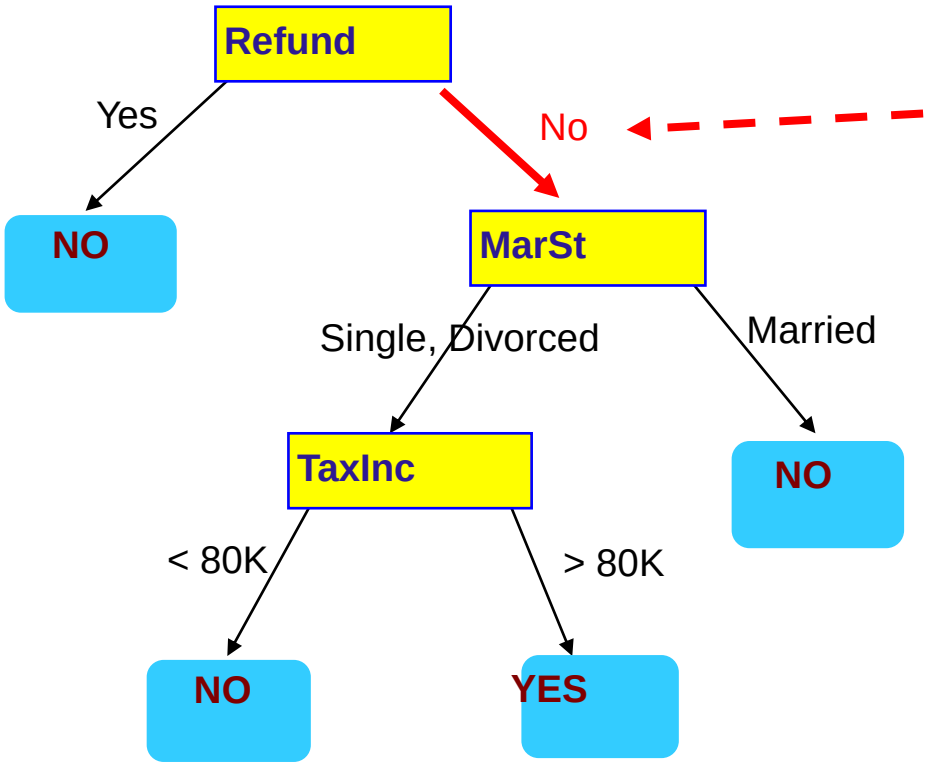
Predicciones



Test Data

refund	Marital status	Taxablei ncome	Tax evasion
no	Married	80K	?

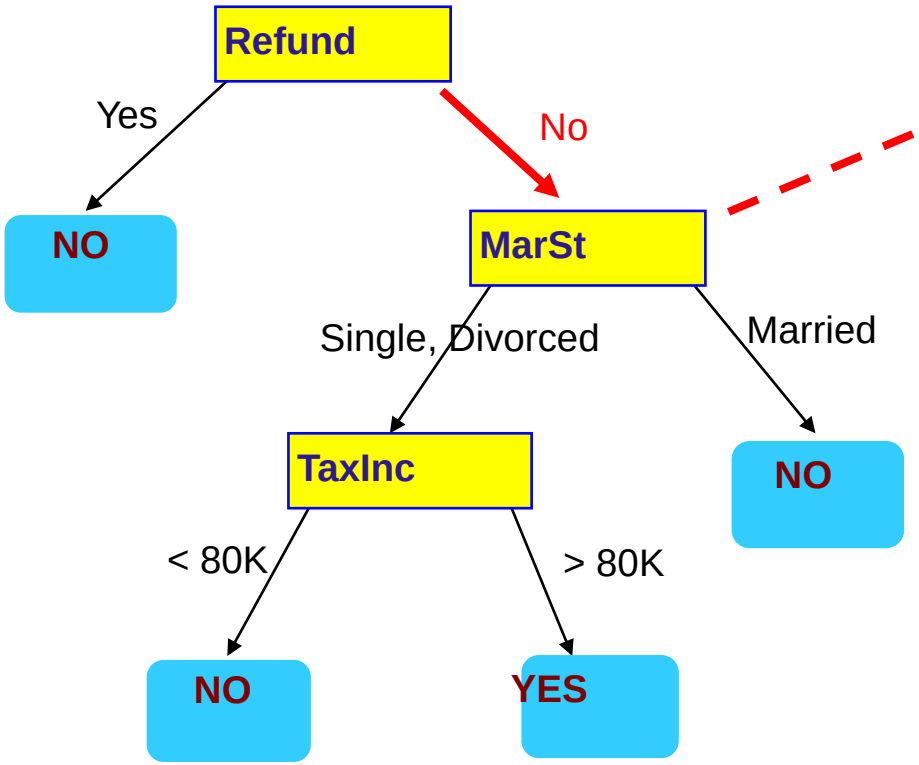
Predicciones



Test Data

refund	Marital status	Taxablei ncome	Tax evasion
no	Married	80K	?

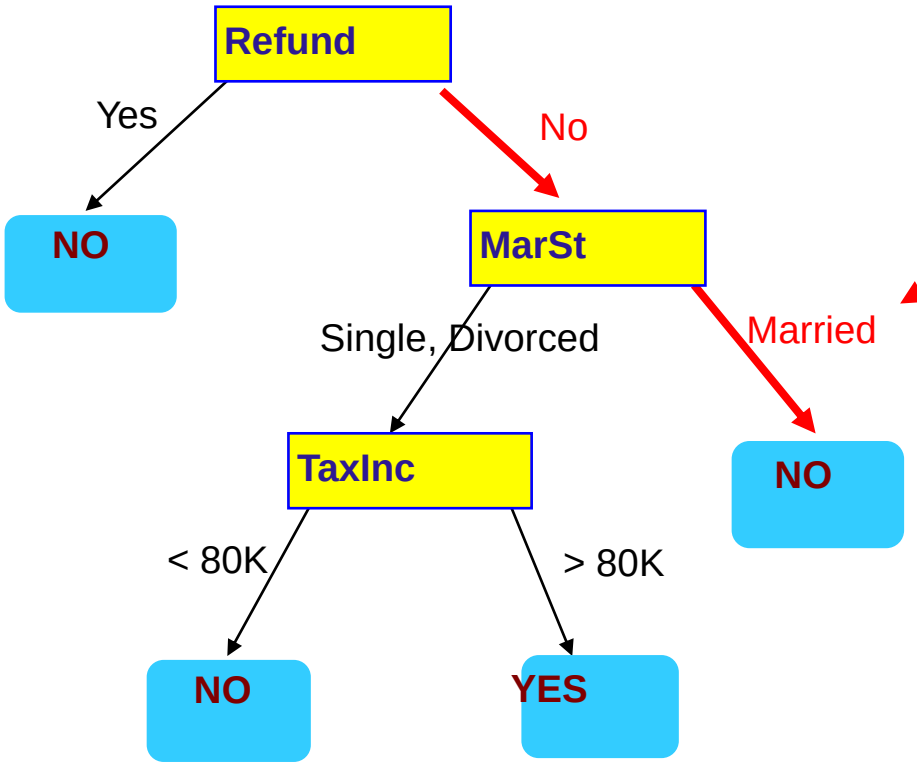
Predicciones



Test Data

refund	Marital status	Taxablei ncome	Tax evasion
no	Married	80K	?

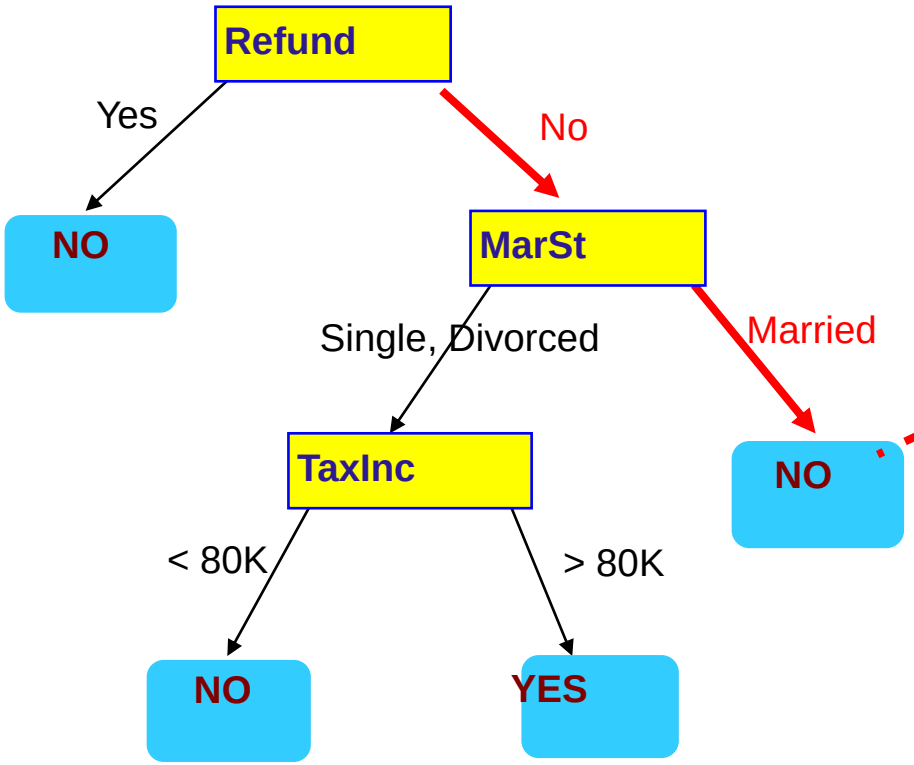
Predicciones



Test Data

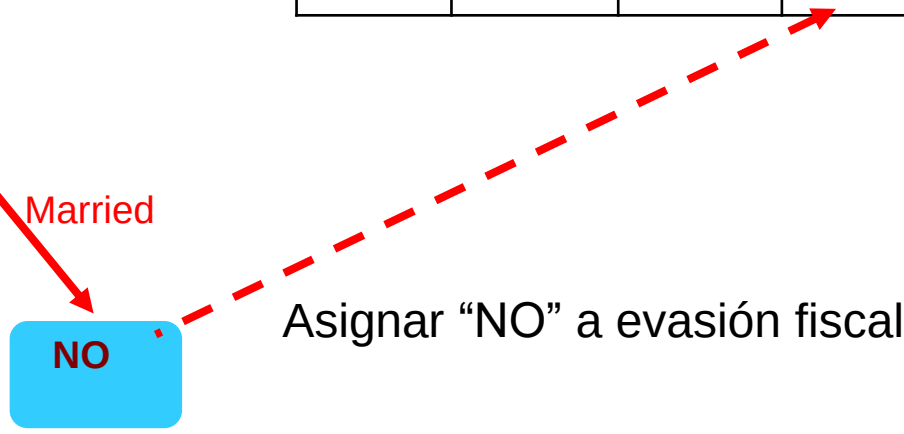
refund	Marital status	Taxablei ncome	Tax evasion
no	Married	80K	?

Predicciones



Test Data

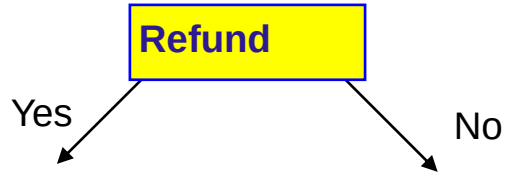
refund	Marital status	Taxablei ncome	Tax evasion
no	Married	80K	?



Asignar “NO” a evasión fiscal

Algoritmo de divide y vencerás

- Comienza con todos los ejemplos en la raíz
- Seleccione el mejor atributo
- Particiona por atributo seleccionado
- Vuelve y repite

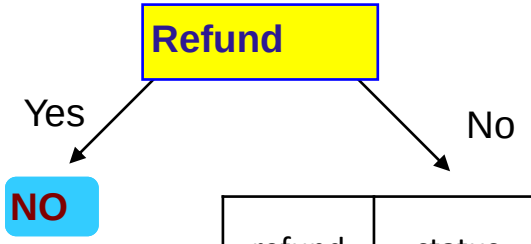


refund	status	income	evasion
yes	single	125K	**no**
yes	married	120K	**no**
yes	divorced	220K	**no**

refund	status	income	evasion
no	married	100K	**no**
no	single	70K	**no**
no	divorced	95K	**yes**
no	married	60K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**

Algoritmo de divide y vencerás

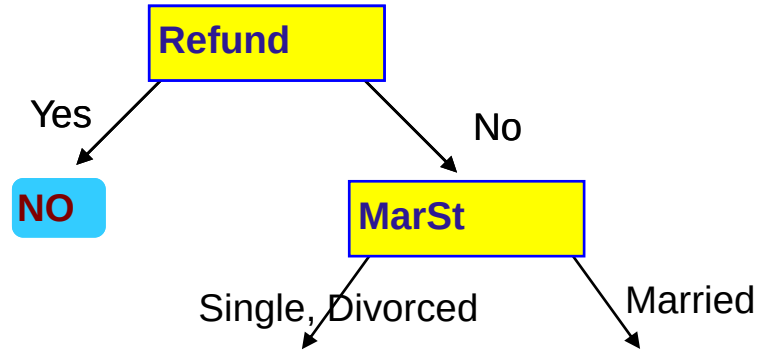
- Comienza con todos los ejemplos en la raíz
- Seleccione el mejor atributo
- Particiona por atributo seleccionado
- Vuelve y repite



refund	status	income	evasion
no	married	100K	**no**
no	single	70K	**no**
no	divorced	95K	**yes**
no	married	60K	**no**
no	single	85K	**yes**
no	married	75K	**no**
no	single	90K	**yes**

Algoritmo de divide y vencerás

- Comienza con todos los ejemplos en la raíz
- Seleccione el mejor atributo
- Particiona por atributo seleccionado
- Vuelve y repite

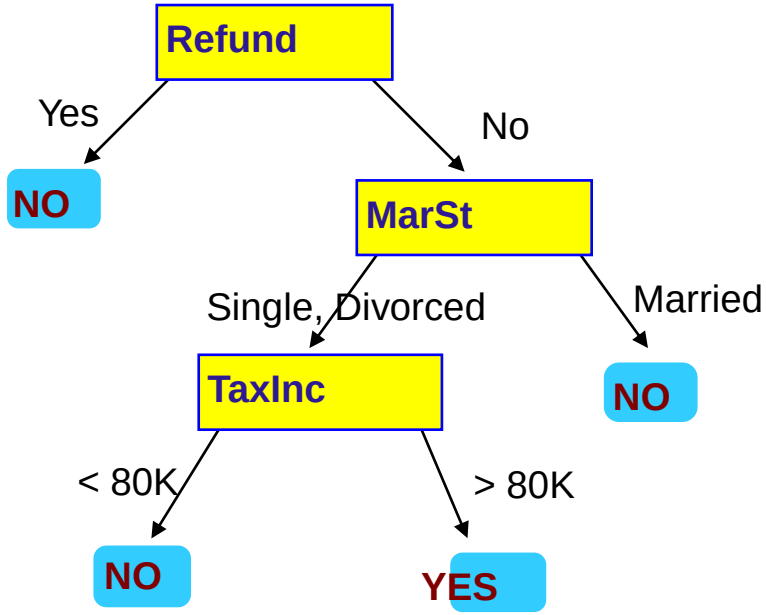


refund	status	income	evasionn
no	single	70K	**no**
no	divorced	95K	**yes**
no	single	85K	**yes**
no	single	90K	**yes**

refund	status	income	evasionf air
no	married	100K	**no**
no	married	60K	**no**
no	married	75K	**no**

Algoritmo de divide y vencerás

- Comienza con todos los ejemplos en la raíz
- Seleccione el mejor atributo
- Particiona por atributo seleccionado
- Vuelve y repite

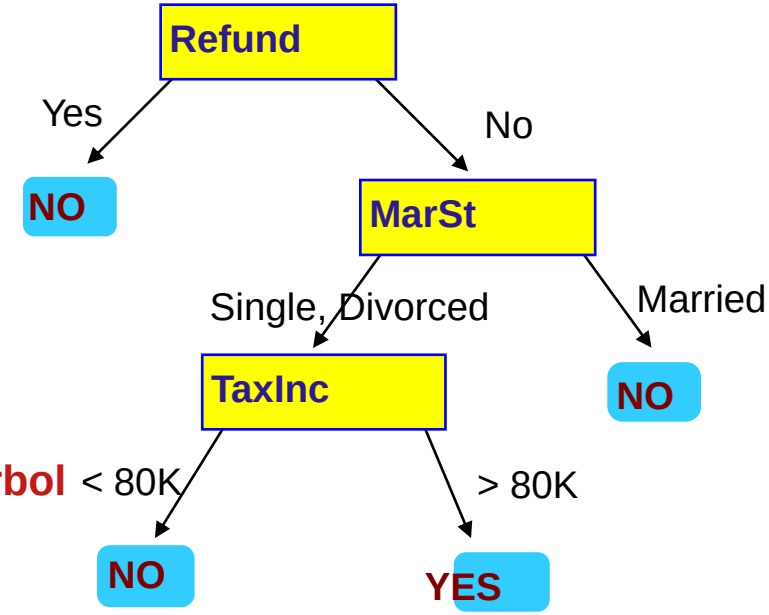


Algoritmo de divide y vencerás

- Comienza con todos los ejemplos en la raíz
- Seleccione el mejor atributo
- Particiona por atributo seleccionado
- Vuelve y repite

Problemas

- **Construcción de ramas**
- **Cuándo dejar de crecer**
- **Poda de partes irrelevantes del árbol**



Hay varios métodos para construir rmas, los más populares son:

- Hunt’s Algorithm (uno de los primeros)
- CART
- ID3, C4.5
- SLIQ, SPRINT

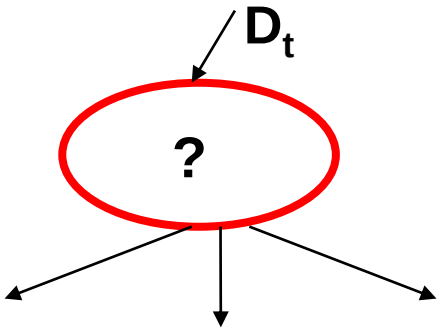
Algoritmo de Hunt

Sea D_t el conjunto de registros de entrenamiento que llegan a un nodo T

Procedimiento General:

- si D_t contiene registros que pertenecen a la misma clase yt, entonces t es un nodo hoja etiquetado como yt
- si D_t es un conjunto vacío, entonces t es un nodo hoja etiquetado por la clase predeterminada, yd
- Si D_t contiene registros que pertenecen a más de una clase, utilice una prueba de atributo para dividir los datos en subconjuntos más pequeños. Aplicar el procedimiento de forma recursiva a cada subconjunto.

refund	Marital status	Taxablein come	Tax evasion
yes	single	125K	no
no	married	100K	no
no	single	70K	no
yes	married	120K	no
no	divorced	95K	yes
no	married	60K	no
yes	divorced	220K	no
no	single	85K	yes
no	married	75K	no
no	single	90K	yes

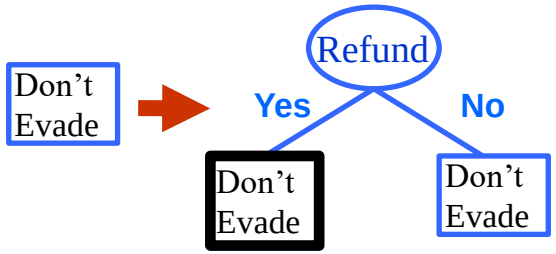


Algoritmo de Hunt

Default class label: NO

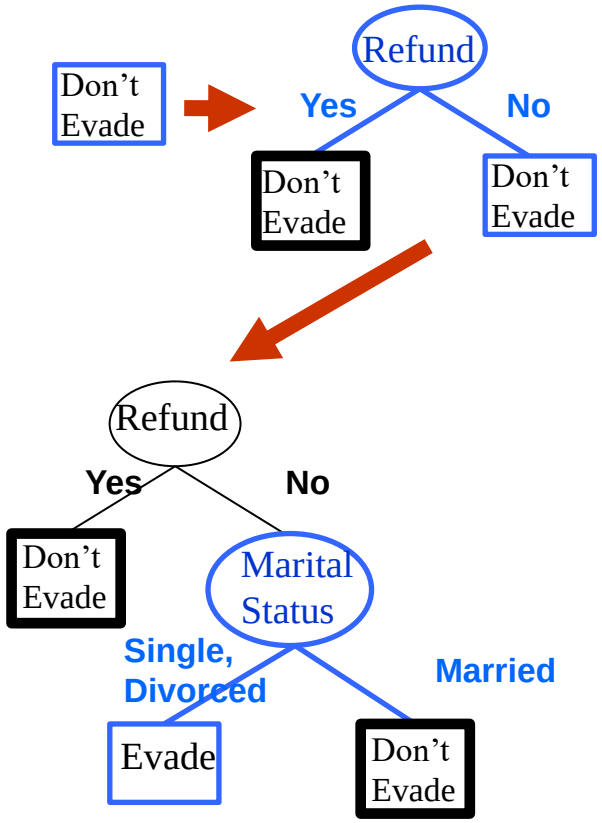
refund	Marital status	Taxablein come	Tax evasion
yes	single	125K	no
no	married	100K	no
no	single	70K	no
yes	married	120K	no
no	divorced	95K	yes
no	married	60K	no
yes	divorced	220K	no
no	single	85K	yes
no	married	75K	no
no	single	90K	yes

Algoritmo de Hunt



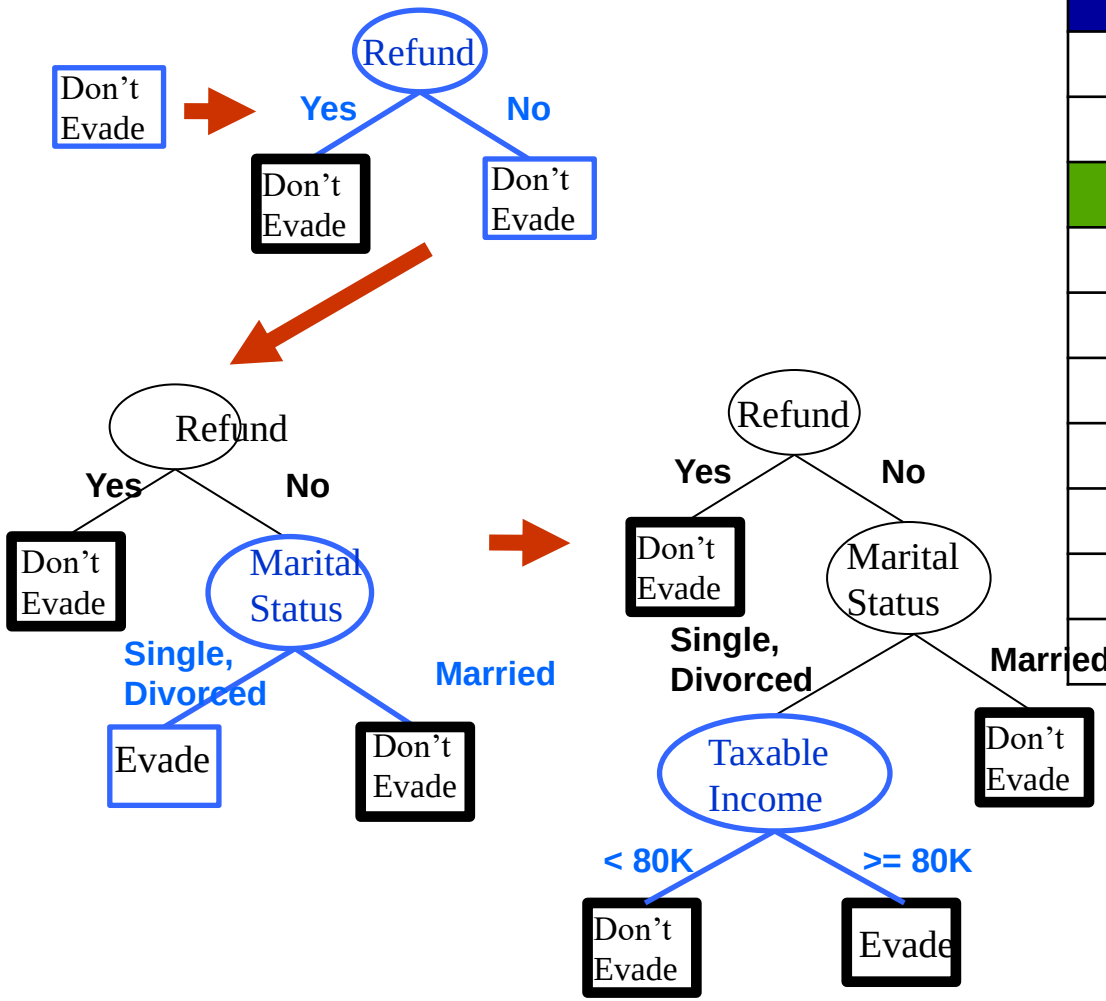
refund	Marital status	Taxablei ncome	Tax Evasion
yes	single	125K	no
no	married	100K	no
no	single	70K	no
yes	married	120K	no
no	divorced	95K	yes
no	married	60K	no
yes	divorced	220K	no
no	single	85K	yes
no	married	75K	no
no	single	90K	yes

Algoritmo de Hunt



refund	Marital status	Taxablei ncome	Tax[SEP]eva sion
no	married	100K	no
no	single	70K	no
no	divorced	95K	yes
no	married	60K	no
no	single	85K	yes
no	married	75K	no
no	single	90K	yes

Algoritmo de Hunt



refund	Marital status	Taxablei ncome	Tax eva sion
no	single	70K	no
no	divorced	95K	yes
no	single	85K	yes
no	single	90K	yes

Cómo especificar los puntos de corte de atributos

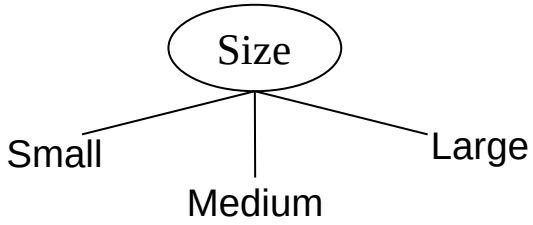
Número de formas de dividir:

- División de 2 vías
- División multi vías

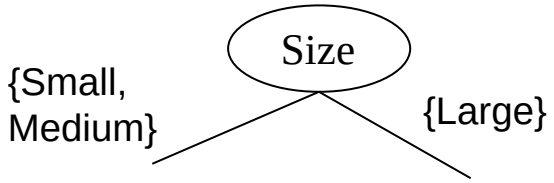
Tipos de atributos

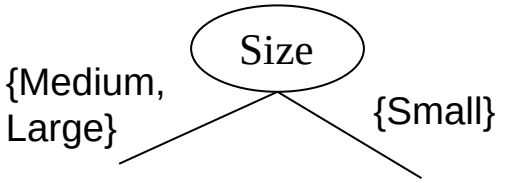
- Nominal
- Ordinal
- Continuo

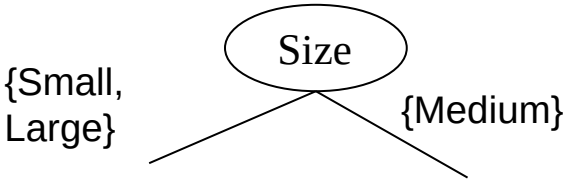
División multi vía : Utiliza tantas particiones como valores distintos.



División de 2 vías (binaria): Divide los valores en dos subconjuntos. Requiere encontrar un particionamiento óptimo.





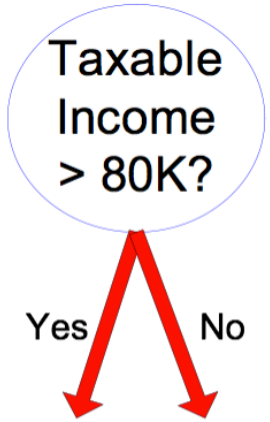


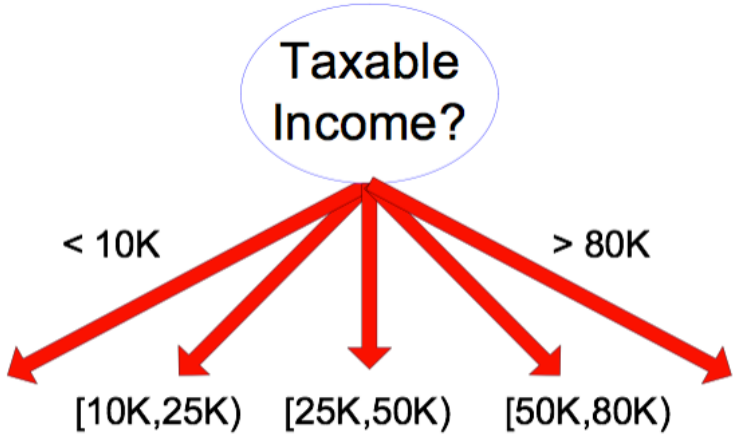
Particionamiento de atributo continuo

Partición binaria: Regla única que divide el atributo en dos subconjuntos (Xj>v)
Podría ser computacionalmente costoso, porque debe considerar todas las divisiones posibles y encontrar el mejor corte.

Decisión de discretización: Formar un atributo categórico ordinal

- **Estático – discretizar una vez al principio (edad<30, 30<edad<40, 40<edad)**
- **Dinámico: los rangos pueden variar dependiendo de la rama del árbol**

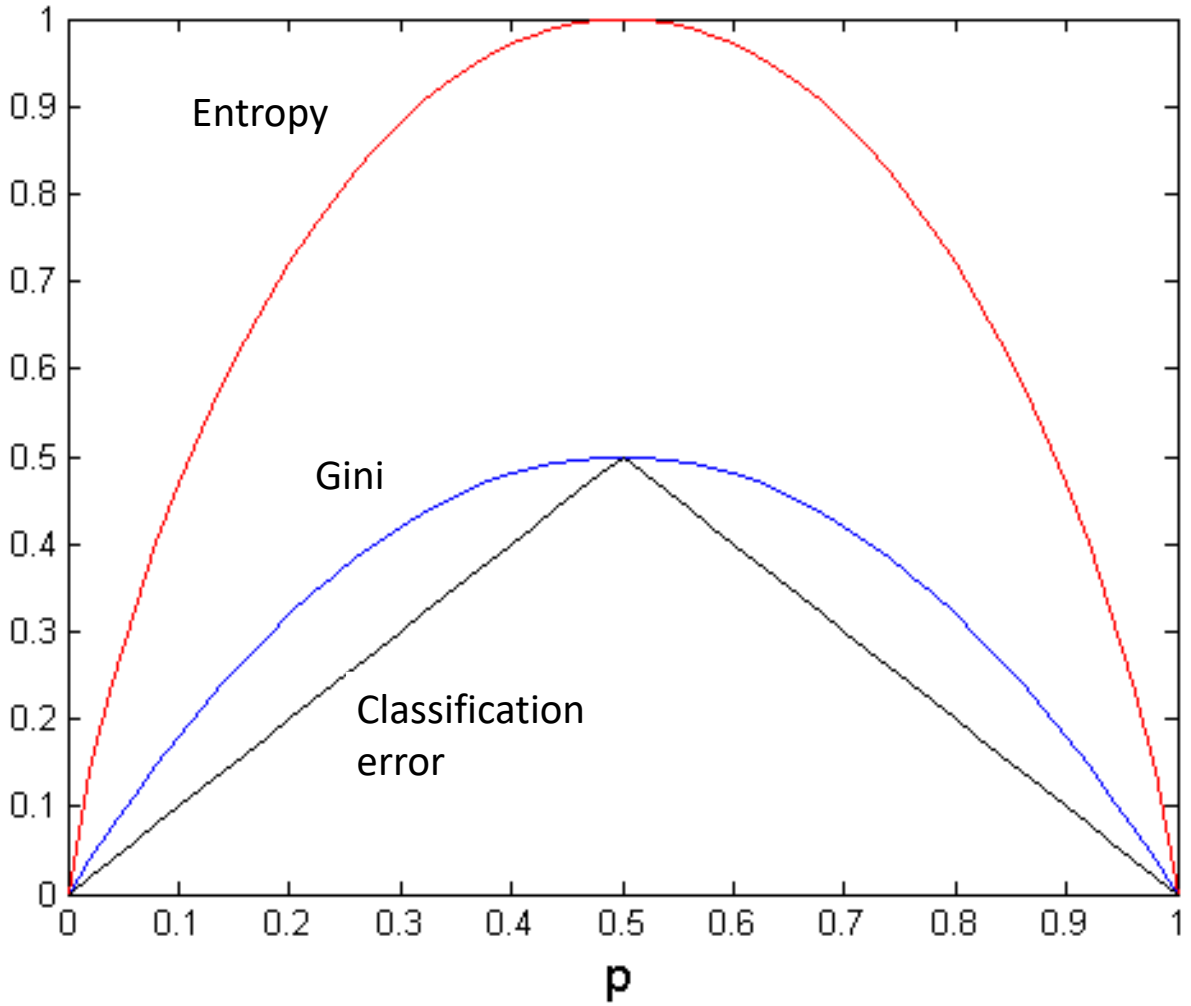




Función de puntuación

- Gini: $$Gini(X) = 1 - \sum_x p(x)^2$$
- Entropía: $$H(X) = -\sum_x p(x) \log_2 p(x)$$
- Classification error: $$Error(X) = 1 - \max_i P(x)$$
- χ^2 score: $$\chi^2(X) = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$

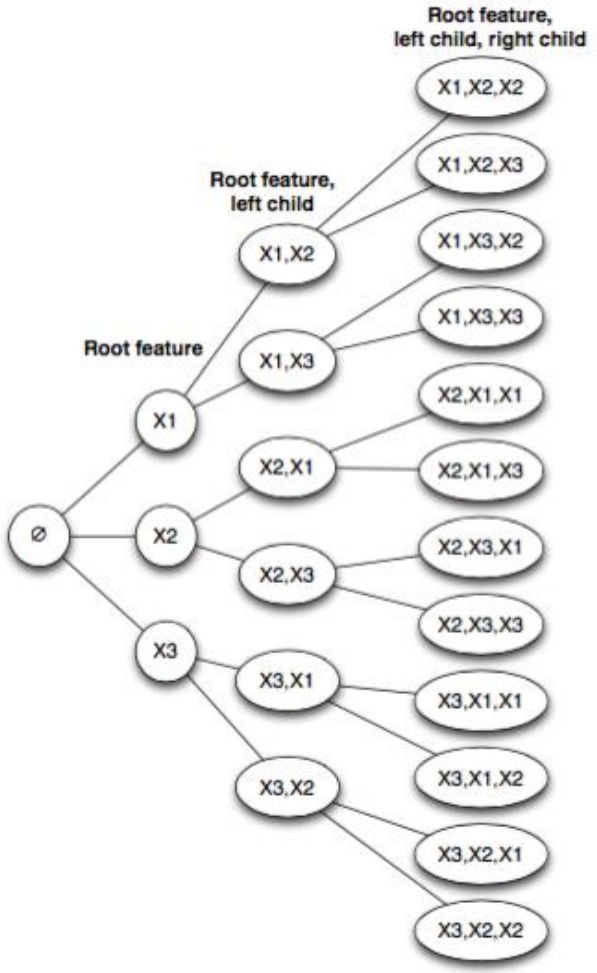
Función de puntuación



Búsqueda

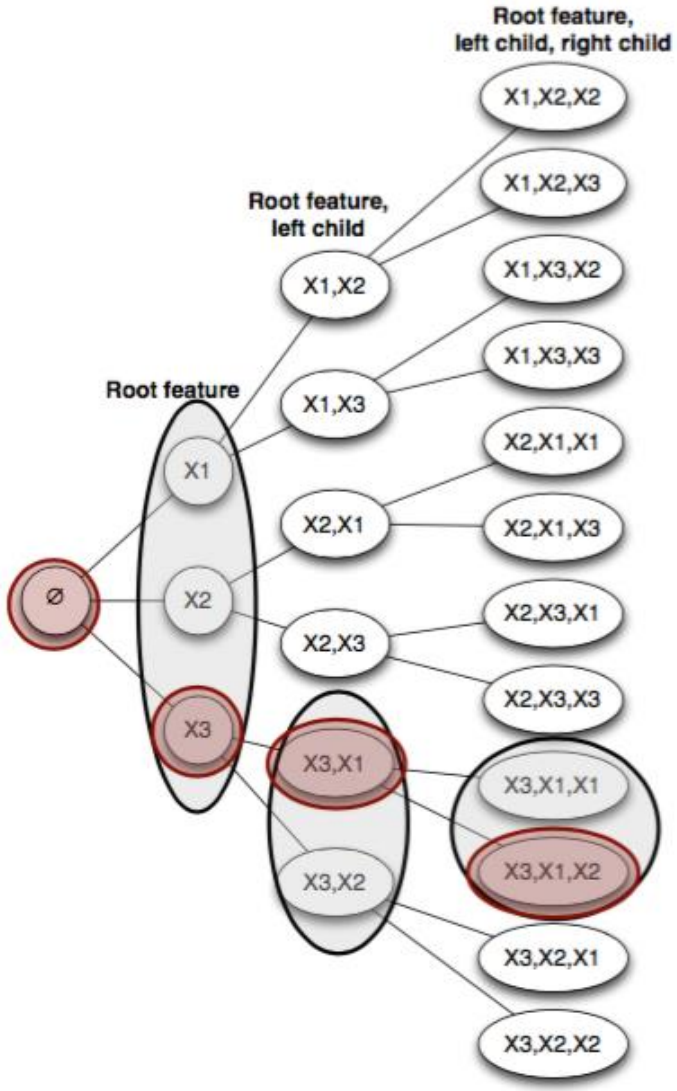
Considere un espacio de posibles modelos M={M1, M2, ..., Mk} dado por la estructura del árbol.
¿Cuál es la estructura de árbol que minimiza el error en los datos de entrenamiento?

Búsqueda exhaustiva: Cree y evalúe todos los modelos posibles y seleccione el mejor modelo.
Normalmente, hay un número exponencial de modelos en el espacio de búsqueda (discreto), lo que hace que sea intratable buscar exhaustivamente en el espacio.
Garantiza encontrar el mejor modelo entre todos los modelos posibles.



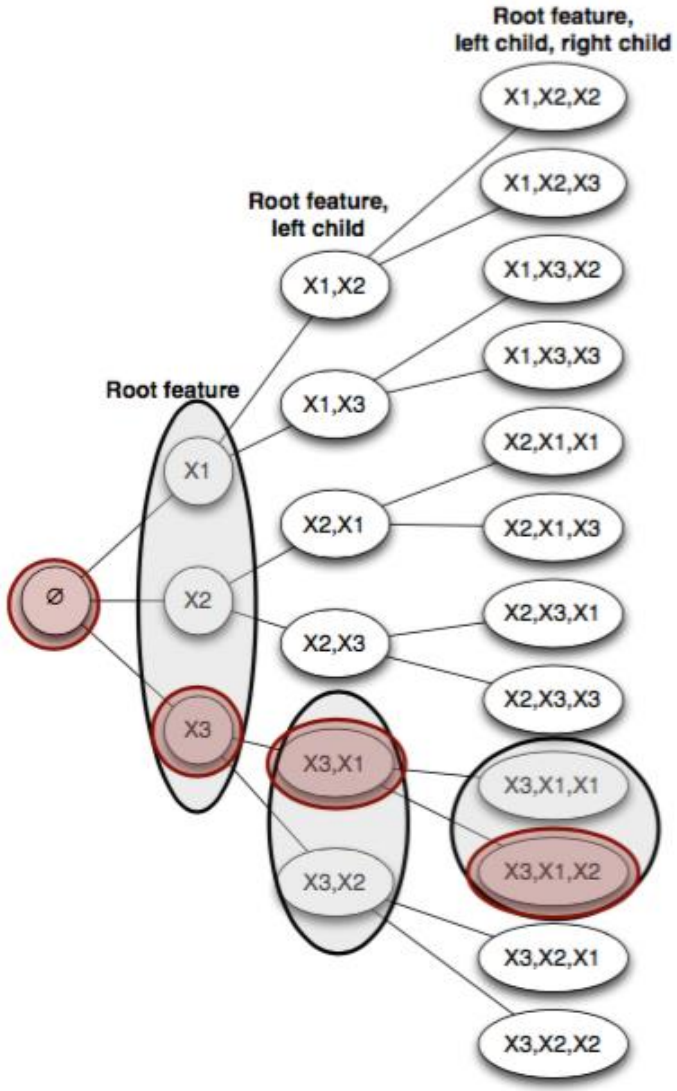
Búsqueda heurística

En cada paso de bifurcación evalúa las alternativas directas en base a la información disponible y toma una decisión.
Una búsqueda heurística no realiza una búsqueda exhaustiva del espacio modelo.
El modelo seleccionado es un óptimo local.



Búsqueda codiciosa (Greedy search)

Seleccione el mejor modelo en cada paso de bifurcación.



Métodos de crecimiento completo

- Todos los ejemplos de un nodo pertenecen a la misma clase
- No quedan atributos para divisiones posteriores
- No quedan muestras

¿Qué impacto tiene esto en la calidad de los árboles aprendidos?

- **Los árboles sobreajustan los datos y la precisión de las pruebas disminuye.**
- **El modelo aprende los datos de entrenamiento, pero no se generaliza a los nuevos datos.**
- **La poda se utiliza para evitar el sobreajuste.**

Poda

Pre poda

- Aplicar una prueba estadística para decidir si se debe expandir un nodo
- Utilice una medida explícita de complejidad para penalizar los árboles grandes (por ejemplo, longitud mínima de la descripción)

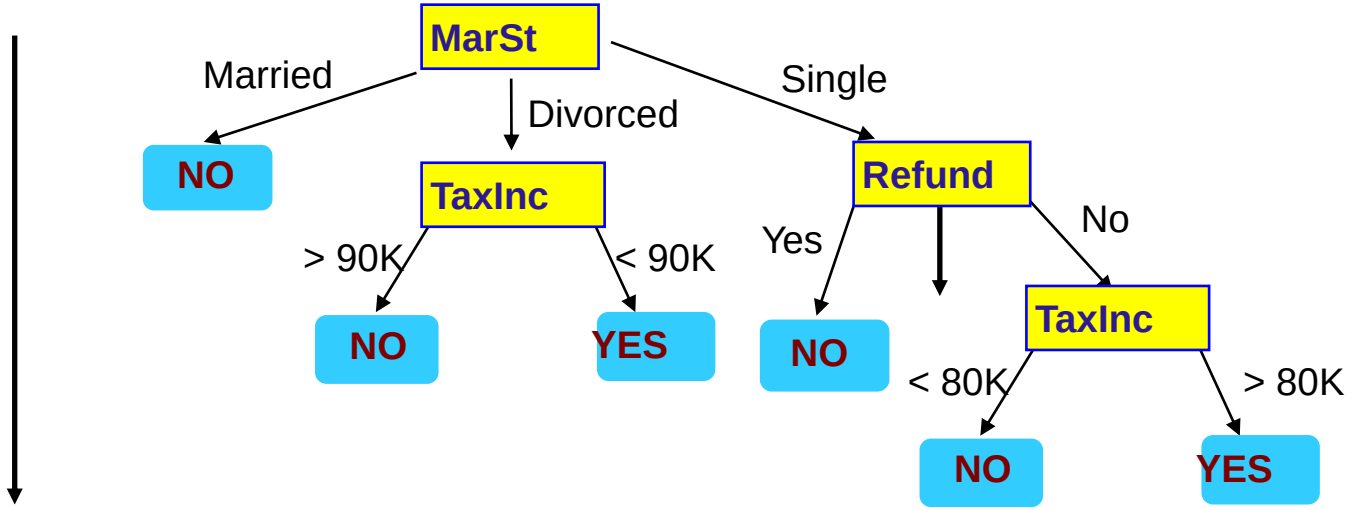
Post poda

- Utilice un conjunto de ejemplos para evaluar la utilidad de podar nodos del árbol (después de que el árbol esté completamente crecido)

Pre poda

Detener el algoritmo antes de que se convierta en un árbol completamente crecido.

- **Detener si el número de instancias de un nodo es menor que algún umbral especificado por el usuario (por ejemplo, un porcentaje específico como el 21% de los datos) (min_samples_leaf).**
- **Detener si la expansión del nodo actual no mejora las medidas de impureza (por ejemplo, entropía) (min_impurity_decrease).**
- **Establecer una profundidad máxima del árbol (max_depth)**



Post poda

Una vez que el árbol esté completamente crecido, recorte los nodos del árbol de decisión de abajo hacia arriba

- En primer lugar, antes de la formación, separe un conjunto de ejemplos $\mathbf{X_{prune}}$ para evaluar la utilidad de podar nodos del árbol.
- En segundo lugar, evaluar $\mathbf{X_{prune}}$ utilizando todo el árbol de clasificación.
- En tercer lugar, recorte un nodo y reemplace el subárbol por un nodo hoja (la etiqueta de clase del nodo hoja se determina a partir de la clase mayoritaria de instancias en el subárbol).
- En cuarto lugar, reevaluar $\mathbf{X_{prune}}$, Si el rendimiento mejora, acepte la poda pasa.

Árboles de decisión

Dr. Raimundo Sánchez

raimundo.sanchez@uai.cl

@raimun2