

# Clasificación

Tratamiento Inteligente de Datos  
Máster Universitario en Ingeniería Informática



UNIVERSIDAD  
DE GRANADA

Gabriel Navarro ([gnavarro@ugr.es](mailto:gnavarro@ugr.es), [gnavarro@decsai.ugr.es](mailto:gnavarro@decsai.ugr.es))

# Objetivos

- ❑ Entender qué es un problema de clasificación y un sistema de clasificación
- ❑ Conocer las fases de cualquier proceso de clasificación
- ❑ Conocer los criterios que se utilizan para evaluar un clasificador
- ❑ Entender el modo de funcionamiento de algunos clasificadores sencillos
- ❑ Conocer distintas metodologías para evaluar un sistema de clasificación

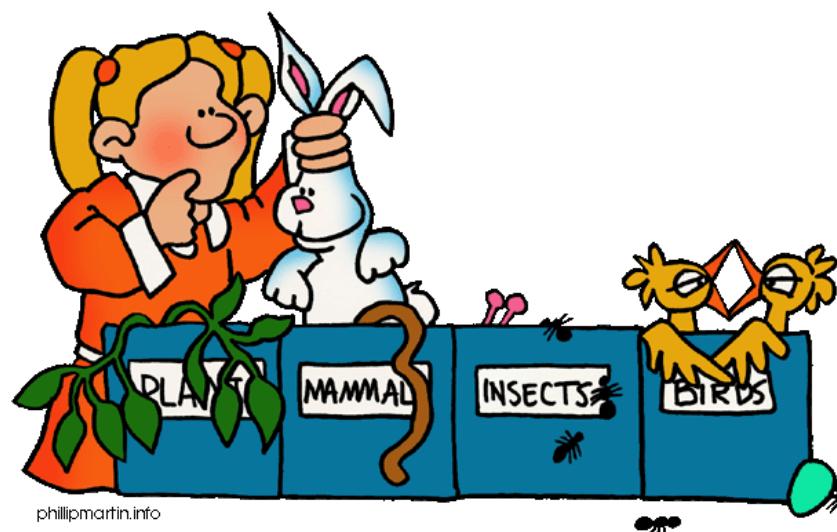
# Índice

- El problema de la clasificación**
  - Concepto de clasificación
  - Fases del proceso de clasificación
  - Ejemplos de clasificadores
  - Evaluación
- Clasificación con árboles de decisión
- Clasificación con reglas
- Clasificación bayesiana
- Otros clasificadores

# El problema de la clasificación

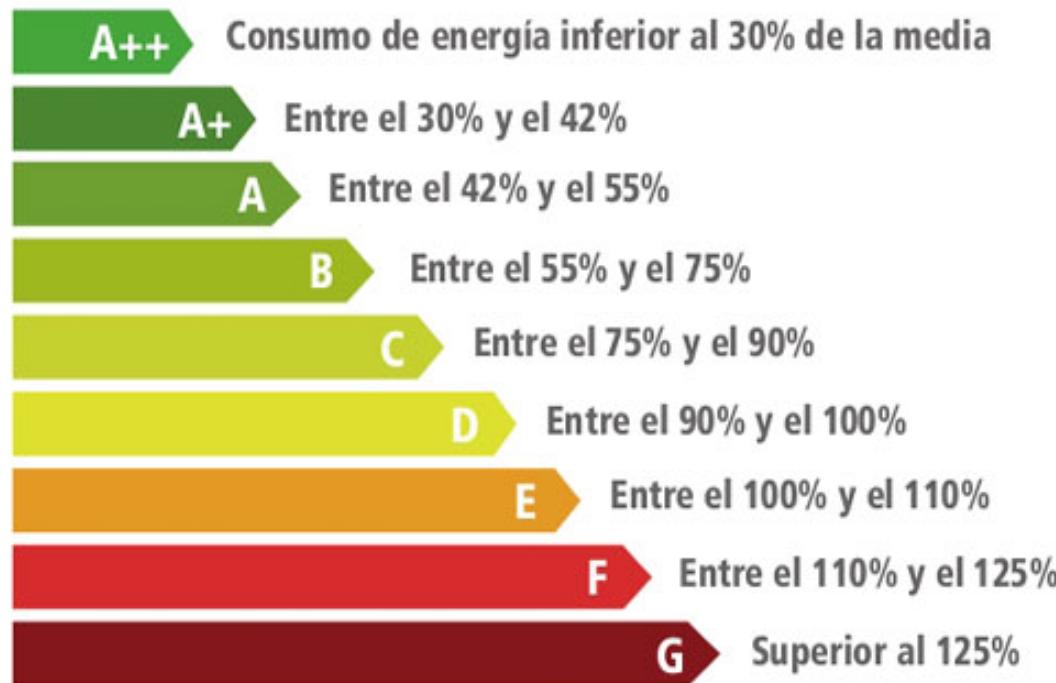
La clasificación es una forma de análisis de datos que consiste en extraer modelos describiendo clases (categóricas) a partir de un conjunto de instancias para predecir la clase de una nueva instancia

- Es la tarea más común en Minería de Datos
- Al modelo obtenido se le llama **clasificador**



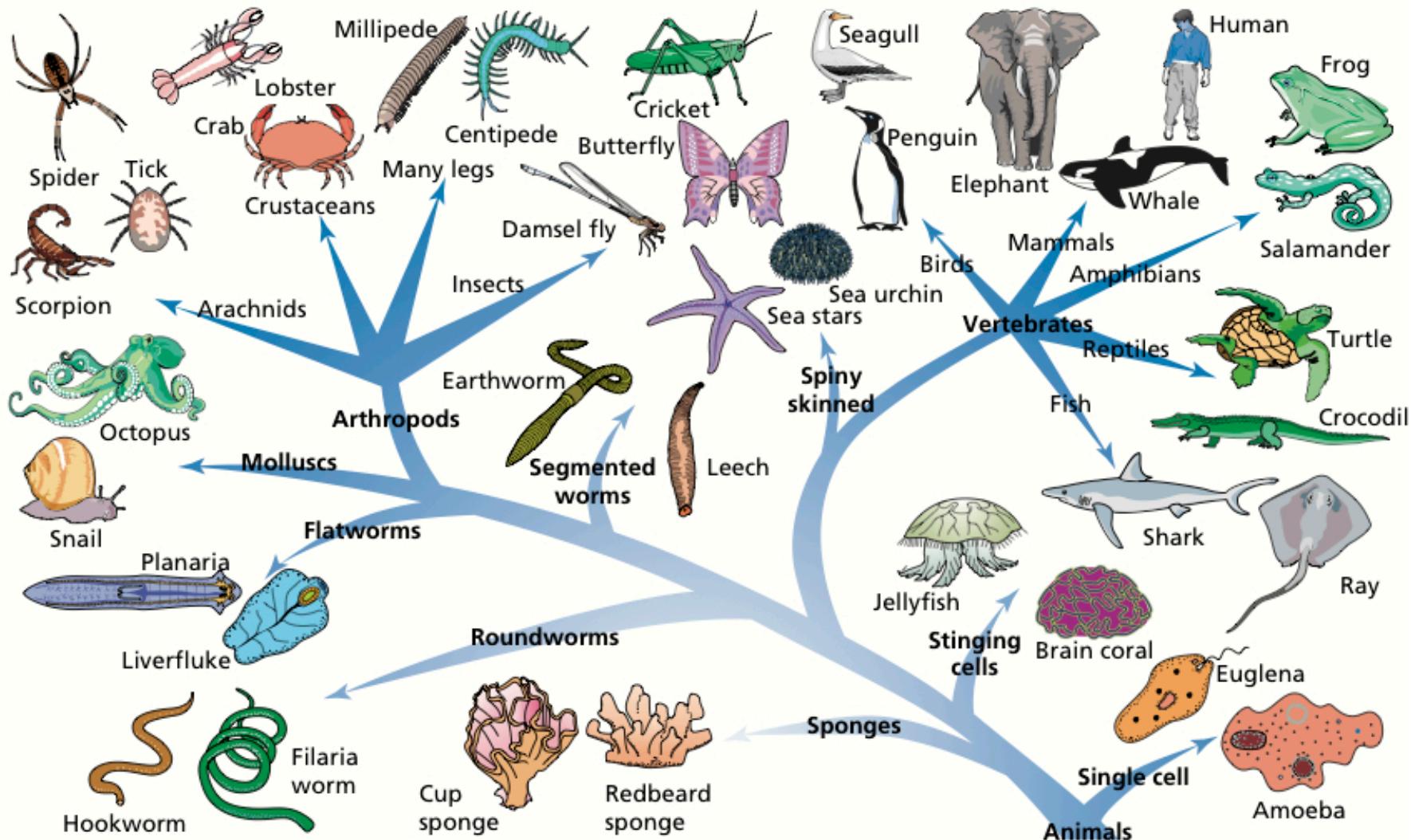
# El problema de la clasificación

MÁS EFICIENTE



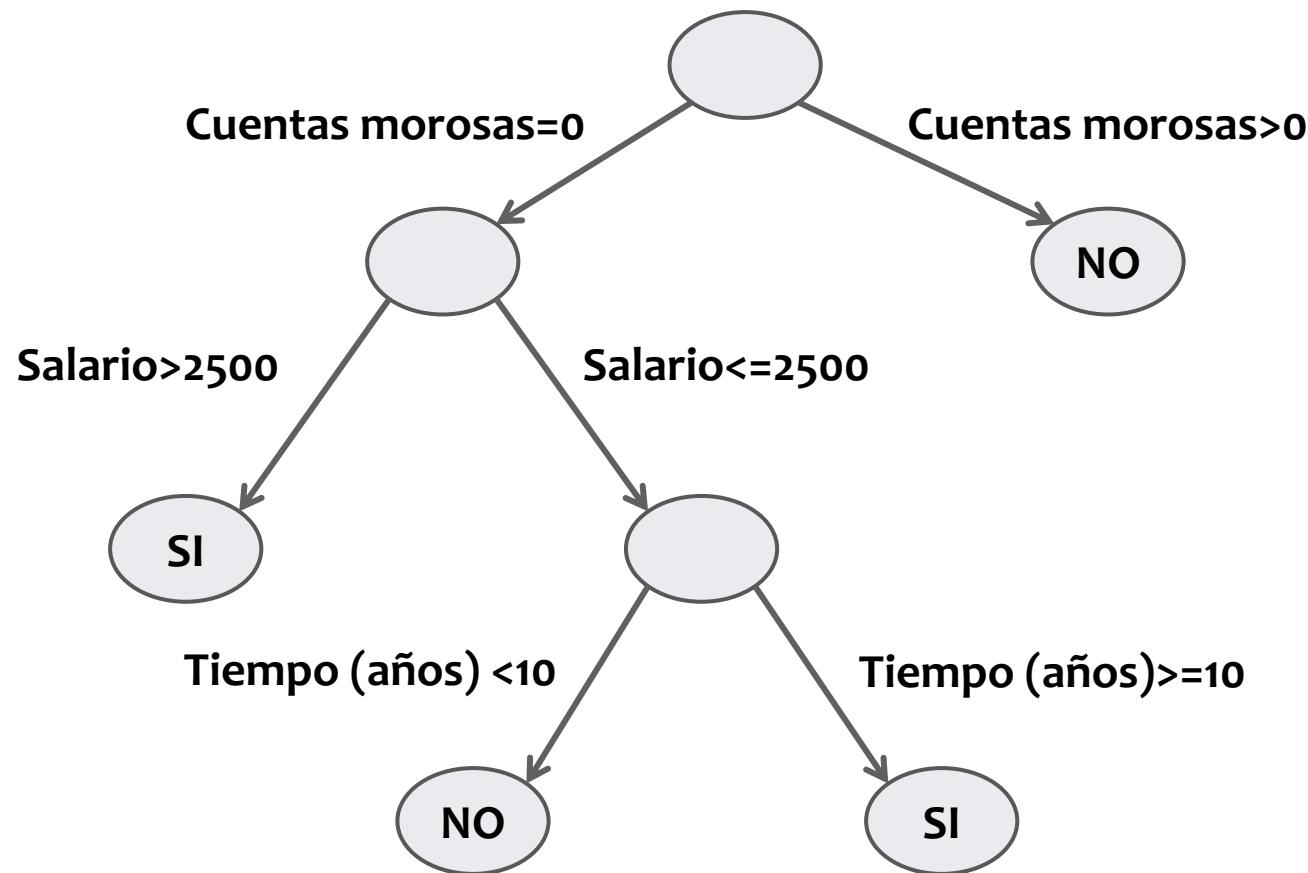
MENOS EFICIENTE

# El problema de la clasificación



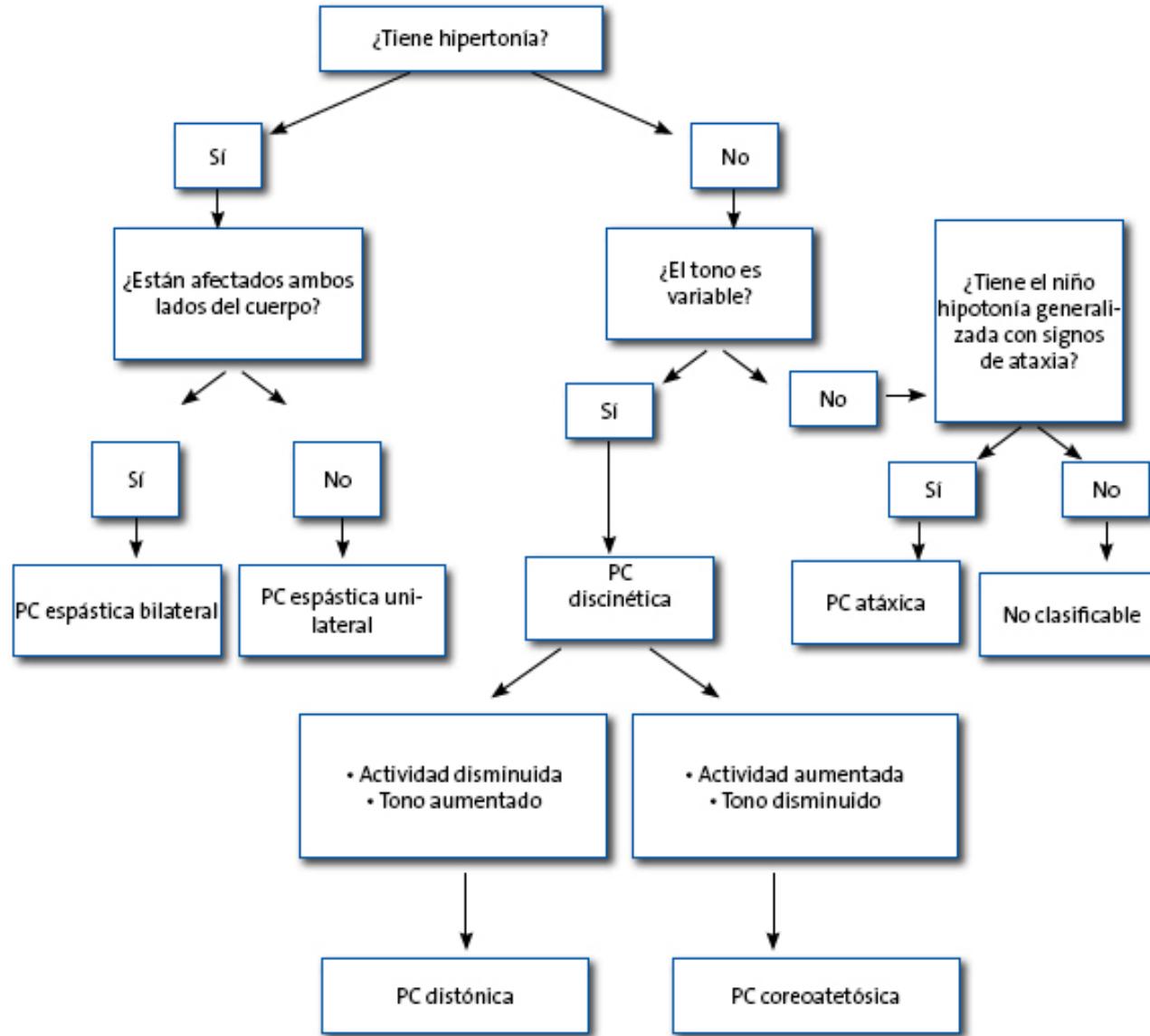
# El problema de la clasificación

Concesión de créditos



# El problema de la clasificación

## Diagnósticos médicos



PC: parálisis cerebral.

# El problema de la clasificación

Otros:

- Detección de fraudes
- Marketing dirigido
- Detección de clientes potencialmente caros (seguros)
- Detección de abandono escolar
- Detección de correo spam
- Selección de empleados
- ...

# El problema de la clasificación

Formalmente, se trata de **aprender** una función

$$f : X_1 \times X_2 \times \cdots \times X_n \longrightarrow \{c_1, \dots, c_t\}$$

donde  $X_1, X_2, \dots, X_n$  son los conjuntos de valores posibles para un conjunto de atributos y  $c_1, \dots, c_t$  son un conjunto de clases posibles

Esta función debe aprenderse a partir de un subconjunto

$$E \subset X_1 \times X_2 \times \cdots \times X_n$$

del cual conocemos su imagen por la función

# El problema de la clasificación

La clasificación es un tipo de aprendizaje supervisado:

- Aprendizaje supervisado

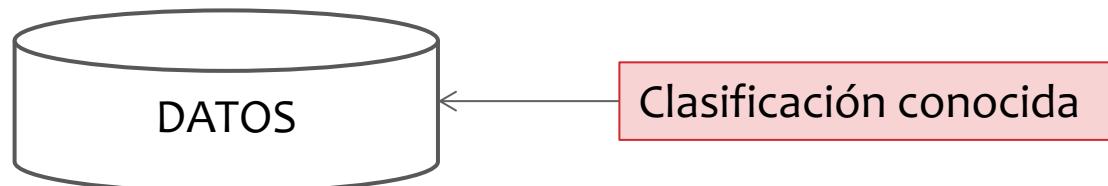
Los casos del conjunto inicial aparecen etiquetados con la clase a la que corresponden

- Aprendizaje no supervisado

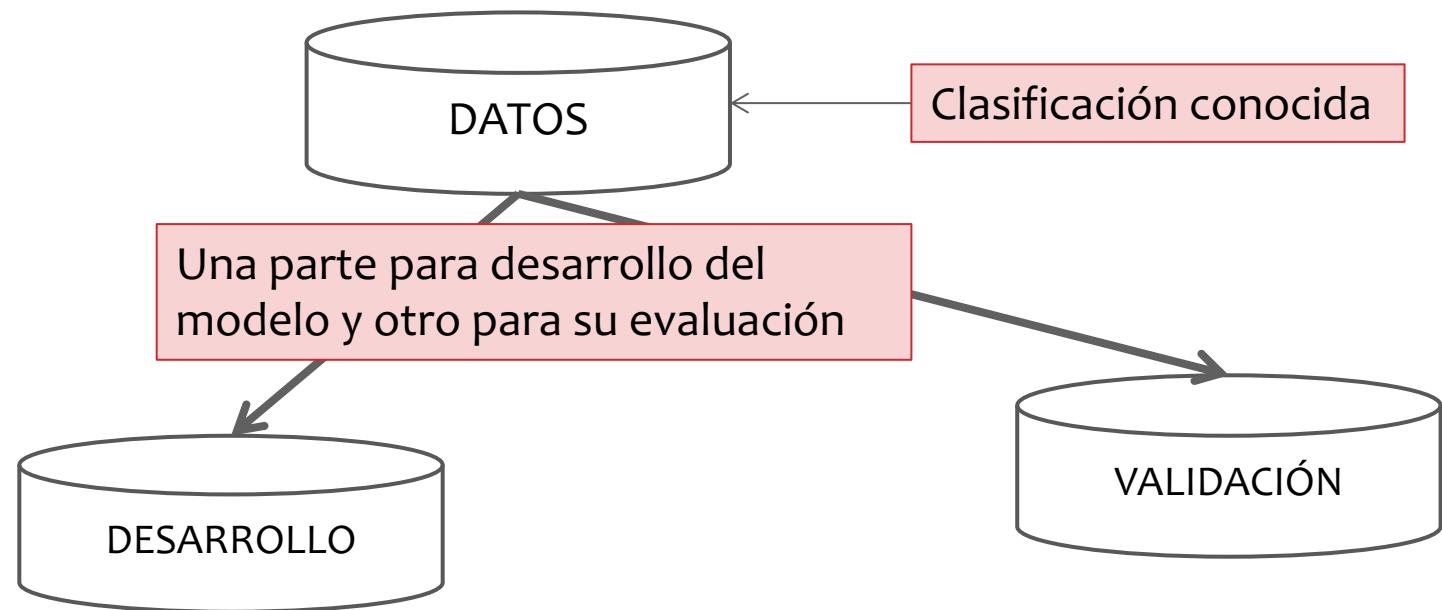
No se conocen las clases de los casos del conjunto inicial (ni siquiera su existencia)

# Fases del proceso de clasificación

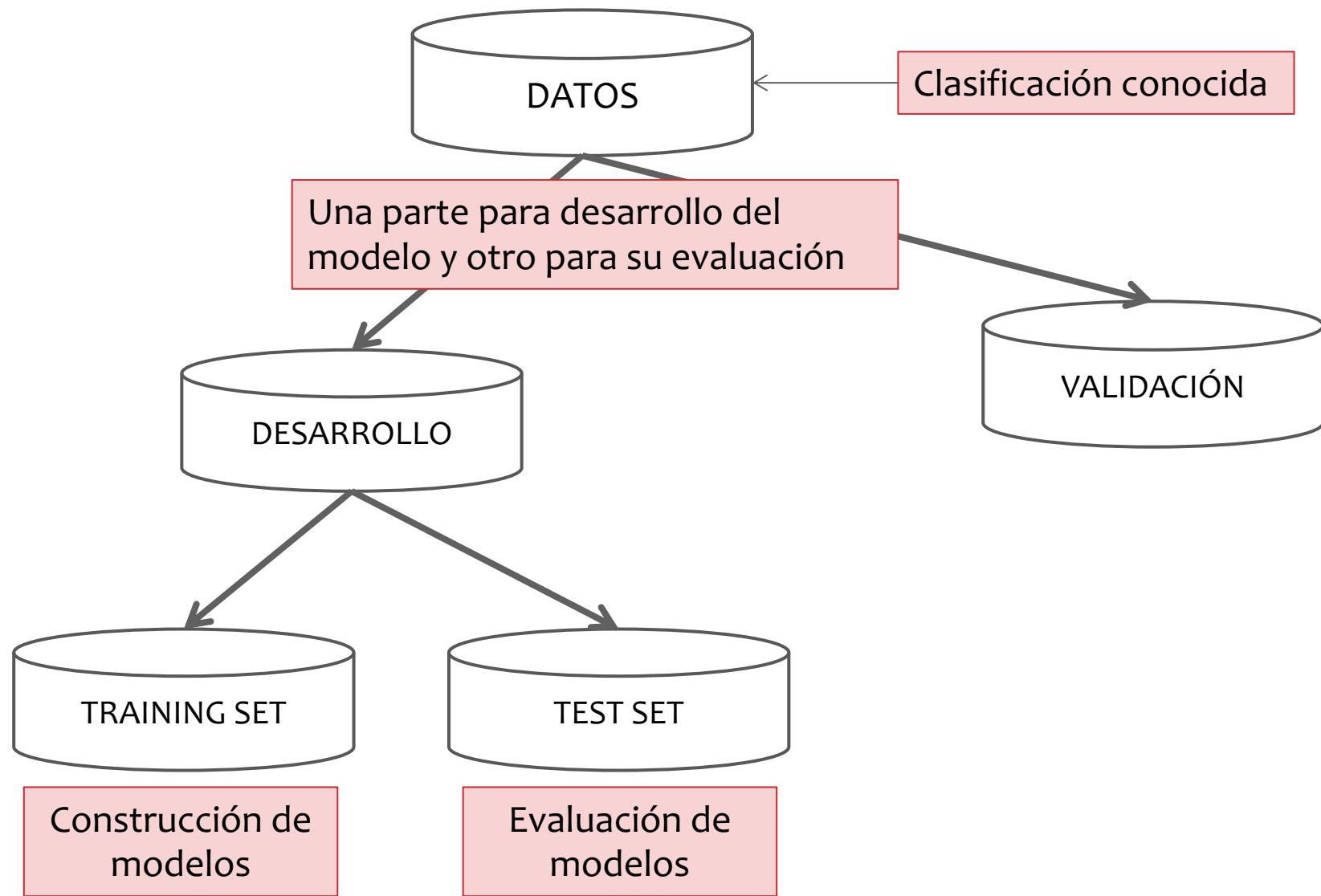
Idealmente supongamos que tenemos muchos datos



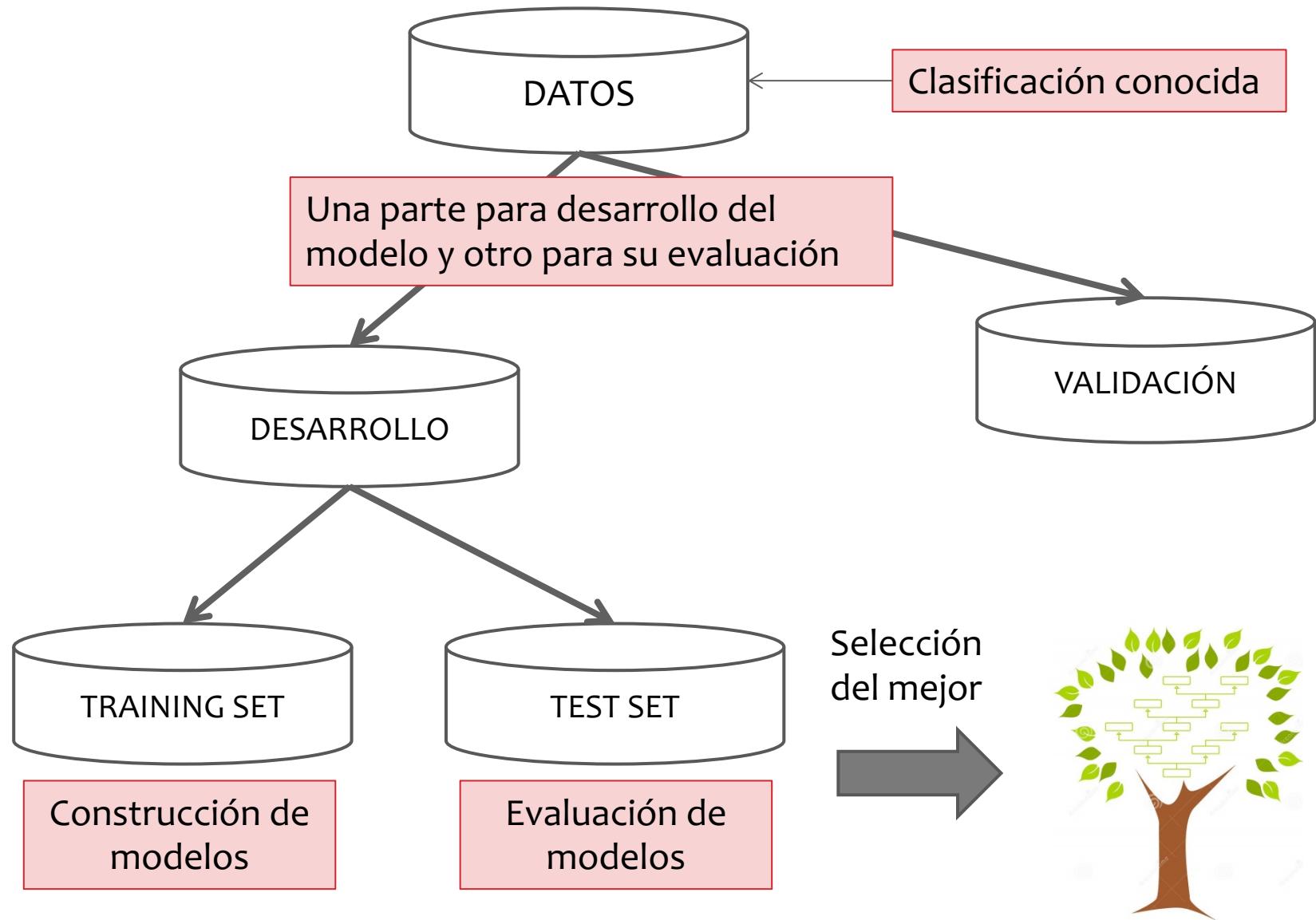
# Fases del proceso de clasificación



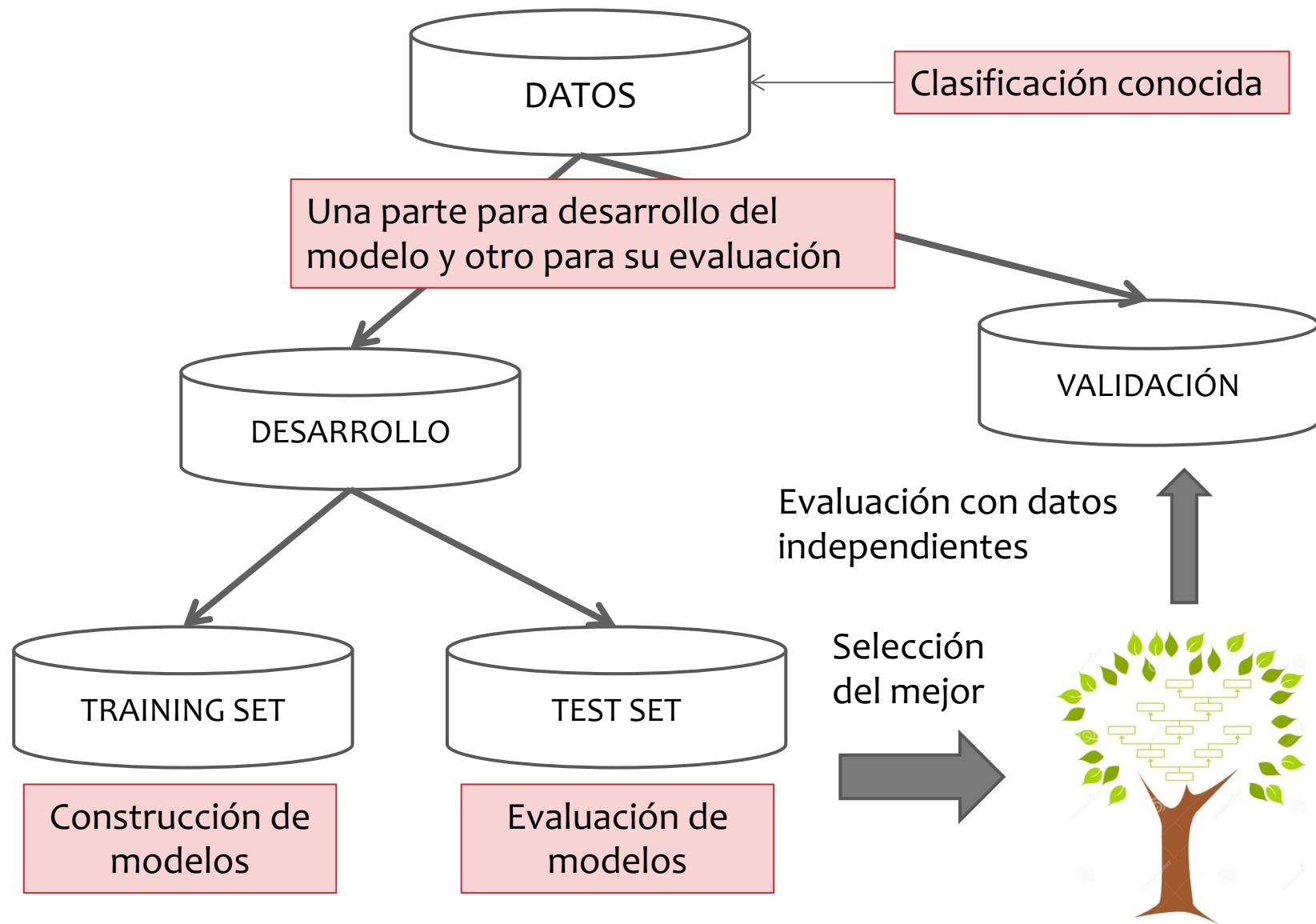
# Fases del proceso de clasificación



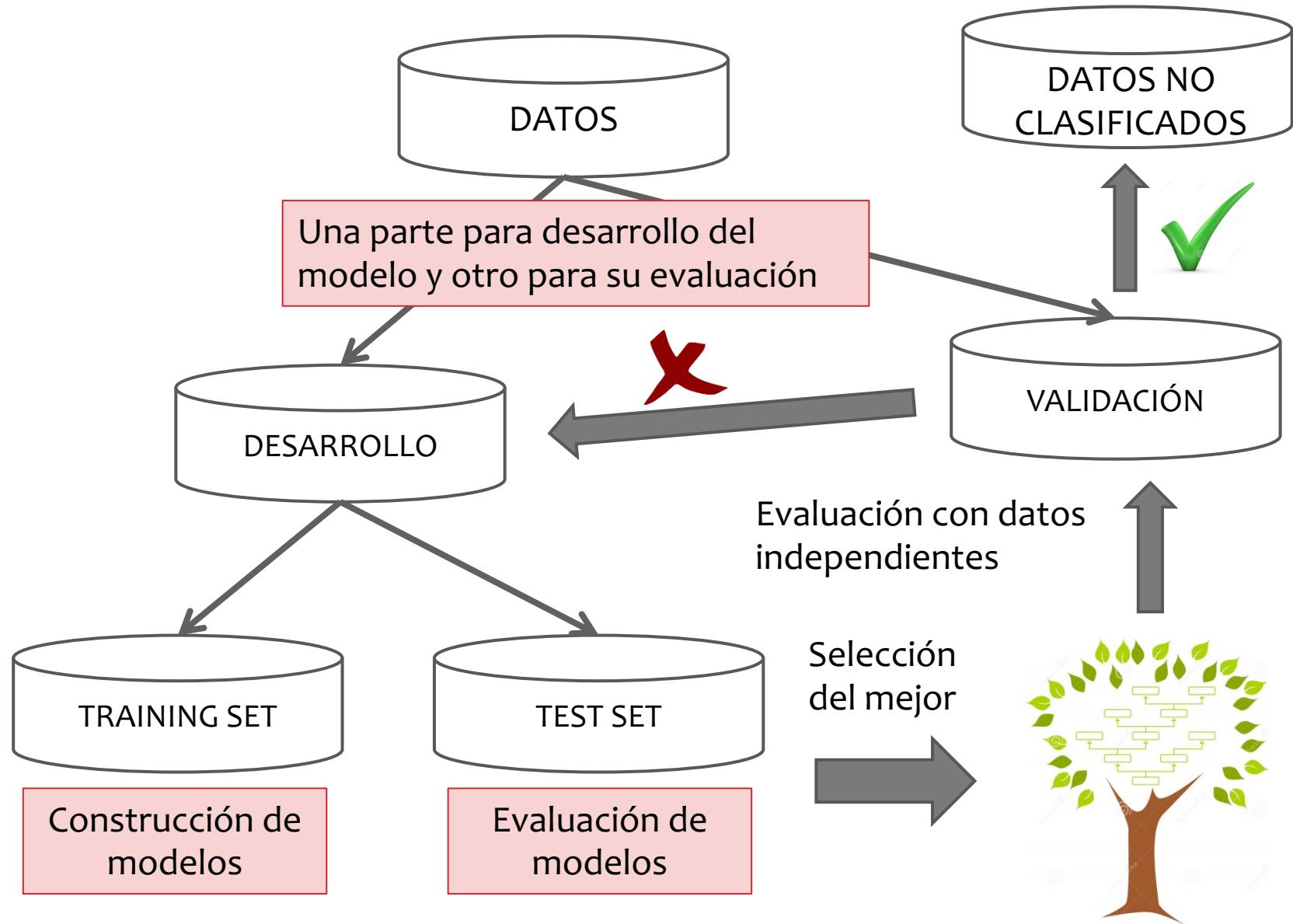
# Fases del proceso de clasificación



# Fases del proceso de clasificación



# Fases del proceso de clasificación



# Construcción del modelo (ejemplos)

## Clasificador zeroR

Se clasifica siempre por la clase mayoritaria

- El más sencillo
- Cualquier algoritmo debería mejorar su rendimiento

Edad	Color pelo
10	Rubio
12	Moreno
15	Rubio
15	Rubio
16	Rubio
18	Pelirrojo
20	Moreno

Color pelo=Rubio es 4/7

Color pelo=Moreno es 2/7

Color pelo=Pelirrojo es 1/7



Clasifica nuevos individuos a **Rubio**

# Construcción del modelo (ejemplos)

## Clasificador oneR

Crea reglas del tipo

**Si variable=valor entonces instancia en clase**

- Algoritmo
  - Para cada atributo AT y cada posible valor VA de dicho atributo, considera la clase mayoritaria CL y crea la regla

**Si AT=VA entonces clase=CL**
  - Calcular el error cometido por las reglas de cada variable
  - Nos quedamos con las reglas de la variable con menor error
- Cualquier algoritmo debería mejorar su rendimiento

# Construcción del modelo (ejemplos)

## Clasificador oneR

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

# Construcción del modelo (ejemplos)

## Clasificador oneR

★		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3

IF Outlook=Sunny THEN Golf=Yes

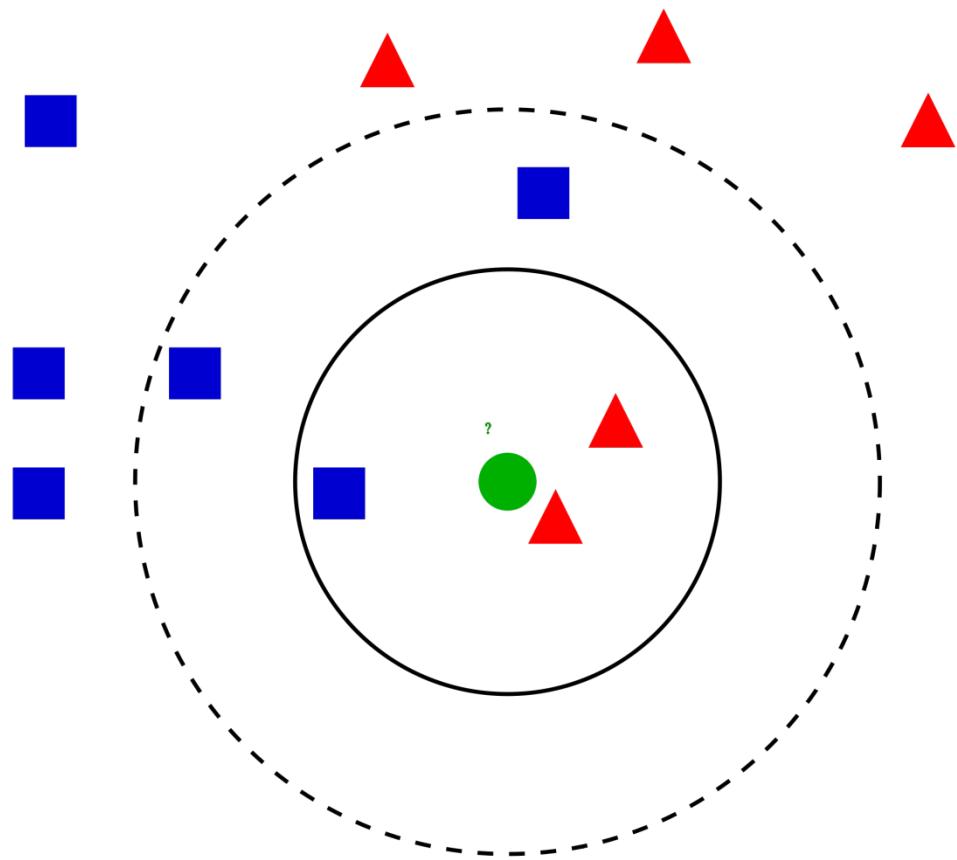
IF Outlook=Overc. THEN Golf=Yes

IF Outlook=Rainy THEN Golf=NO

# Construcción del modelo (ejemplos)

## Clasificador k-NN (k-nearest neighbor)

Utilizando una distancia (normalmente, euclídea) clasifica según la clase más repetida entre los k casos más cercanos



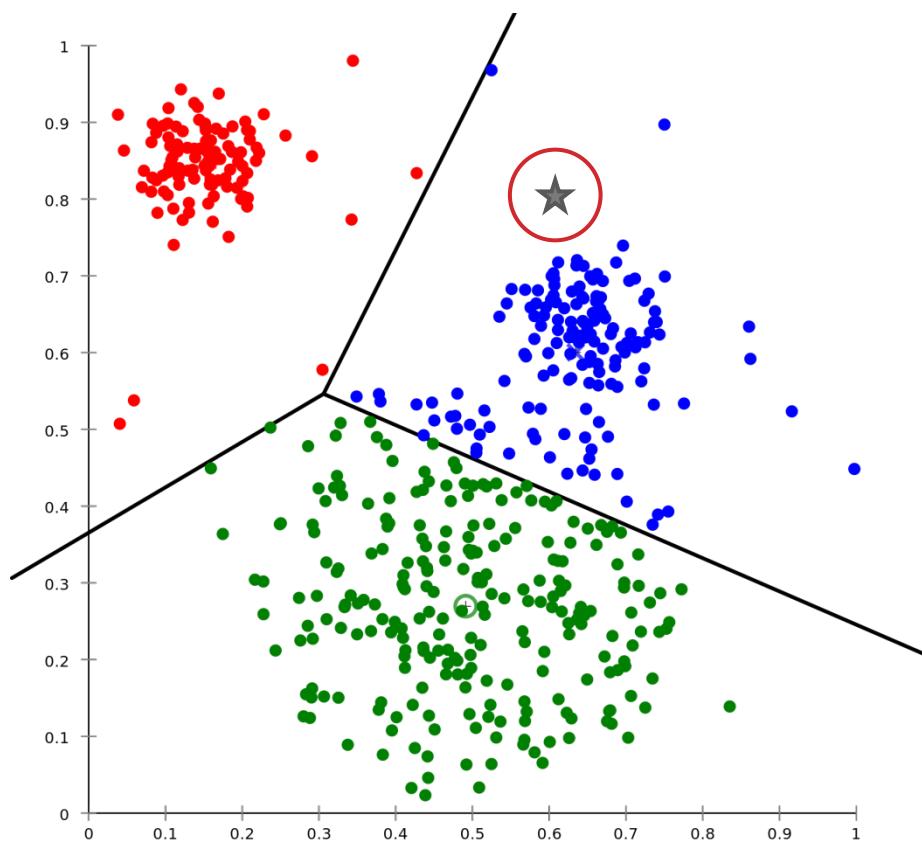
K=3, clasifica como triángulo

K=5, clasifica como cuadrado

# Construcción del modelo (ejemplos)

## Clasificador basado en clustering

Se realiza un proceso de agrupamiento o clustering y se clasifica al grupo perteneciente



# Evaluación

- ❑ **Métodos (metodología)**

Que proceso sigo para, de forma fiable, evaluar la calidad de un modelo

- ❑ **Medidas**

Cómo medir la “calidad” de un modelo de clasificación

- ❑ **Comparación**

Cómo comparar el rendimiento relativo de dos modelos de clasificación alternativos

# Métodos de evaluación

Una vez construido el modelo a partir del conjunto de desarrollo, se usa dicho modelo para clasificar los datos del conjunto de validación:

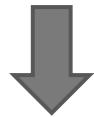
- Comparando los casos etiquetados del conjunto de prueba con el resultado de aplicar el modelo, se obtiene un **porcentaje de clasificación**.
- Si la precisión del clasificador es aceptable, podremos utilizar el modelo para **clasificar nuevos casos** (de los que desconocemos realmente su clase).

# Métodos de evaluación

¿Por qué utilizar el conjunto de validación?

¿Por qué dividir en varios conjuntos?

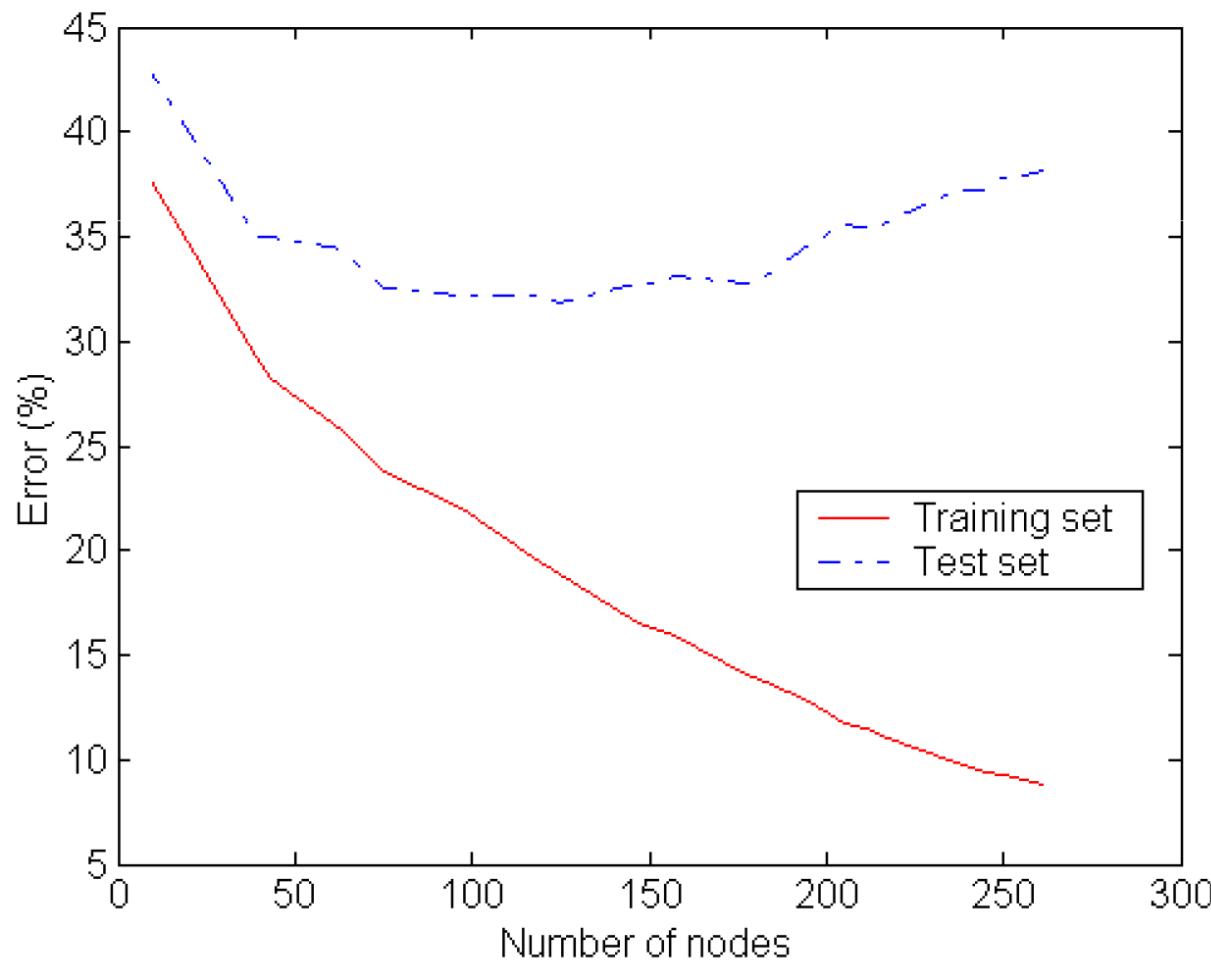
Cuanto mayor sea su complejidad, los modelos de clasificación tienden a ajustarse más al conjunto de entrenamiento/desarrollo utilizado en su construcción (sobreaprendizaje), lo que los hace menos útiles para clasificar nuevos datos



- El error de clasificación en el conjunto de desarrollo **NO** es un buen estimador de la precisión del clasificador
- El conjunto de validación debe ser siempre independiente del conjunto de desarrollo

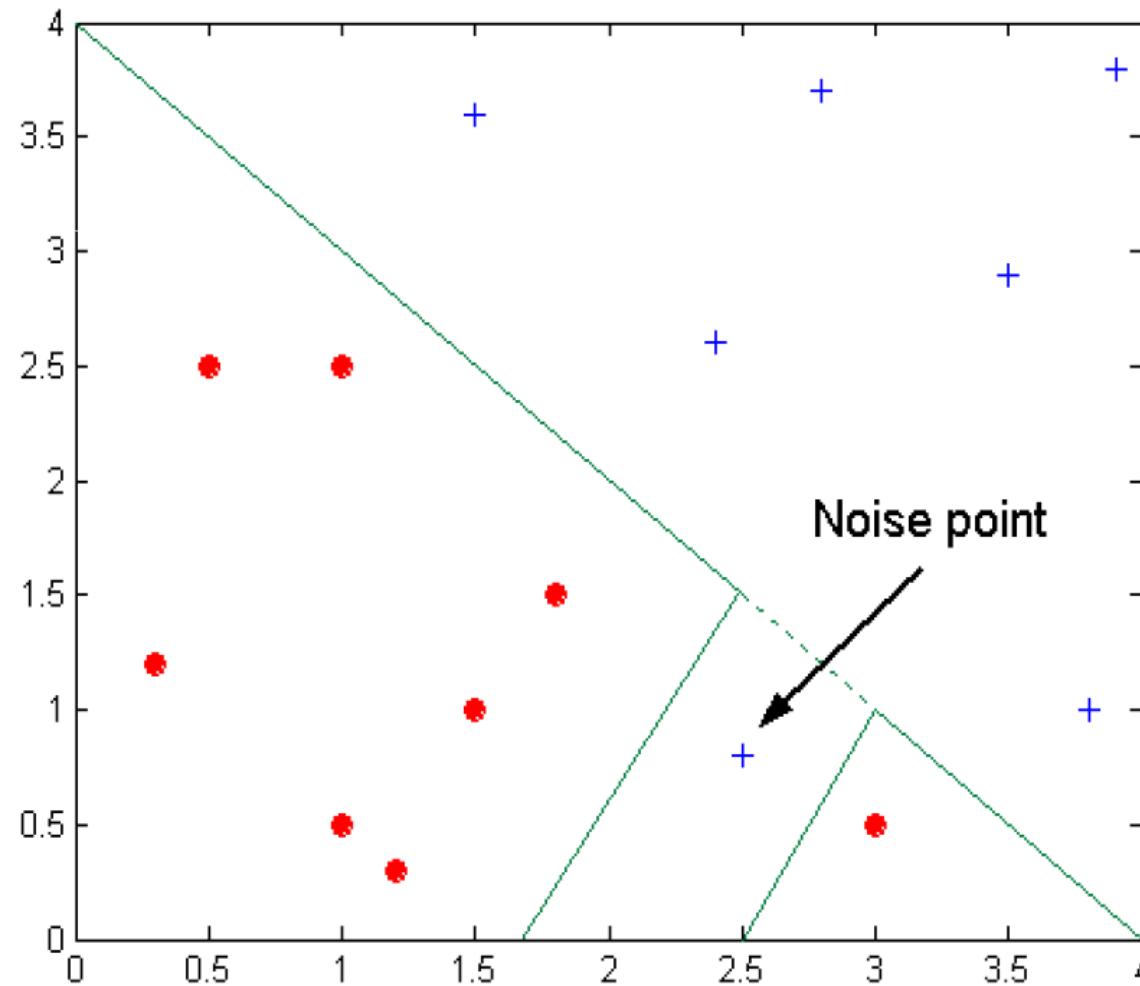
# Métodos de evaluación

Sobreaprendizaje debido a la complejidad del clasificador



# Métodos de evaluación

Sobreaprendizaje debido a la presencia de ruido en datos

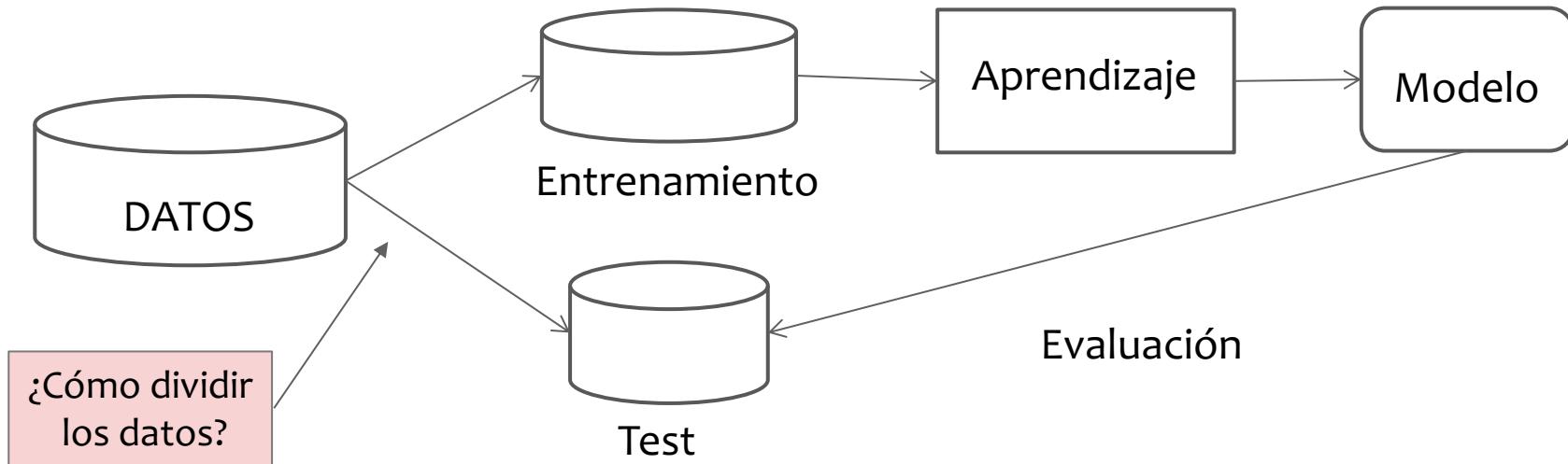


# Métodos de evaluación

¿Cómo evalúo el modelo?

Un problema. Normalmente, no tengo tantos datos como para dividir en conjunto de entrenamiento, conjunto de test y conjunto de validación

Puesto que he utilizado un conjunto test, tomo la estimación obtenida en ese conjunto. Es decir, no considero el conjunto de validación

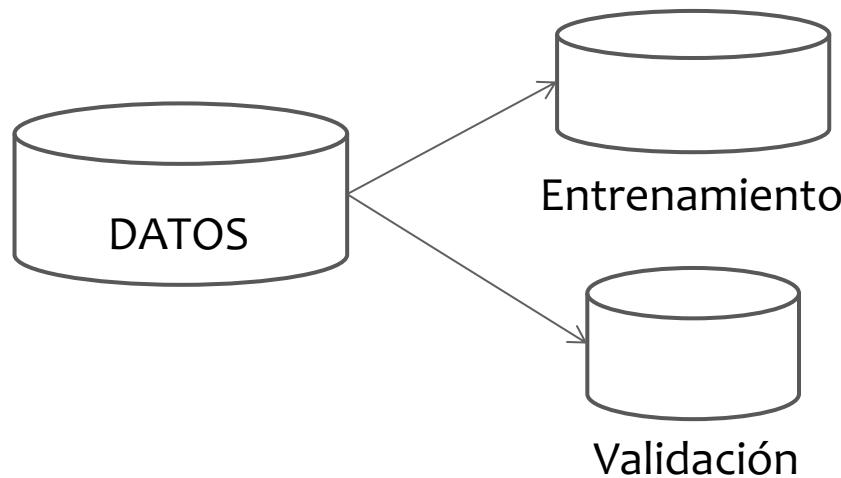


En cualquier caso, hay **distintas metodologías**

# Métodos de evaluación

## Hold-out

Es la partición más básica. Consiste en separar el conjunto de datos en **dos subconjuntos disjuntos**, uno para entrenamiento y otro para test



# Métodos de evaluación

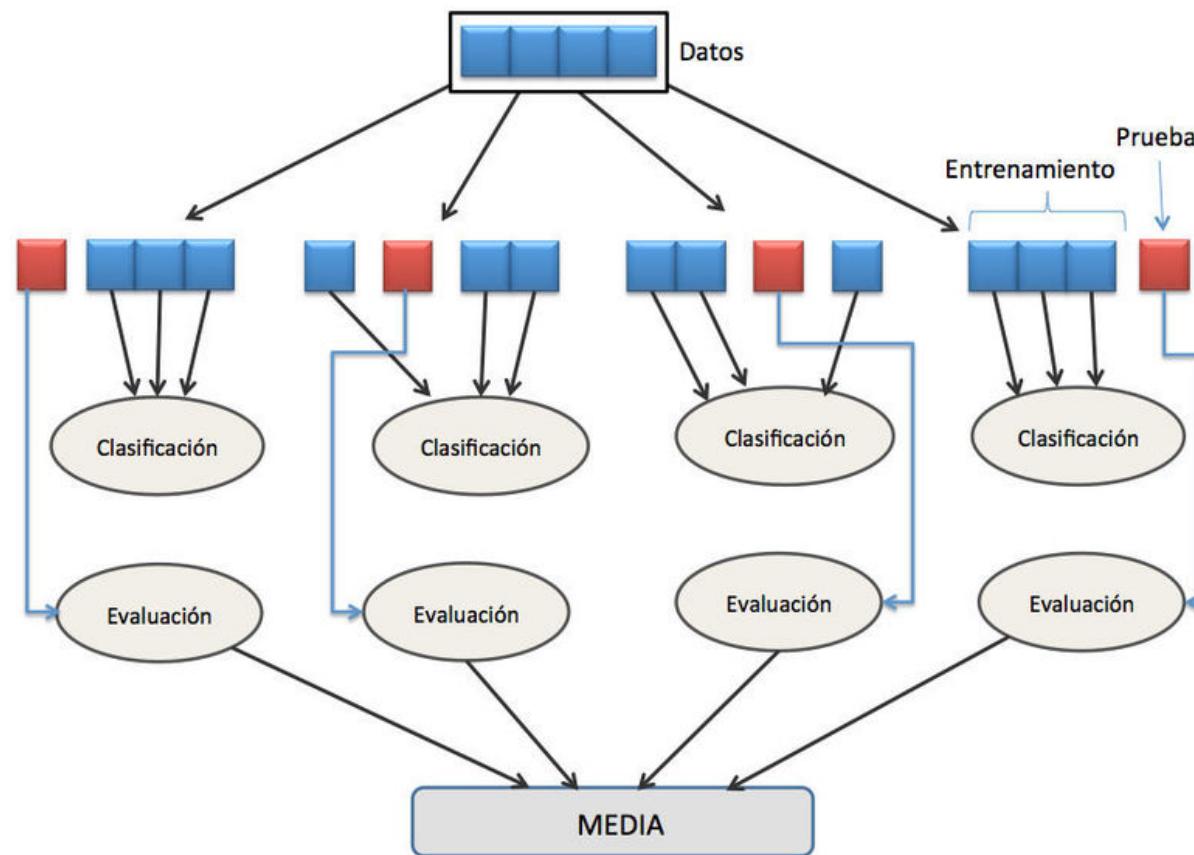
## Hold-out

- ❑ Normalmente, el conjunto para entrenamiento es más grande que el de test
  - Porcentajes: 50-50, 75-25, 80-20, 90-10, ...
  - Problemas de sobreentramiento y subajuste!
- ❑ La elección del conjunto test se puede realizar con muestreo aleatorio sin reemplazamiento (por ejemplo)
  - Problemas de sesgo!
- ❑ Para grandes cantidades de datos es lo más razonable

# Métodos de evaluación

## Validación cruzada (k-fold cross-validation)

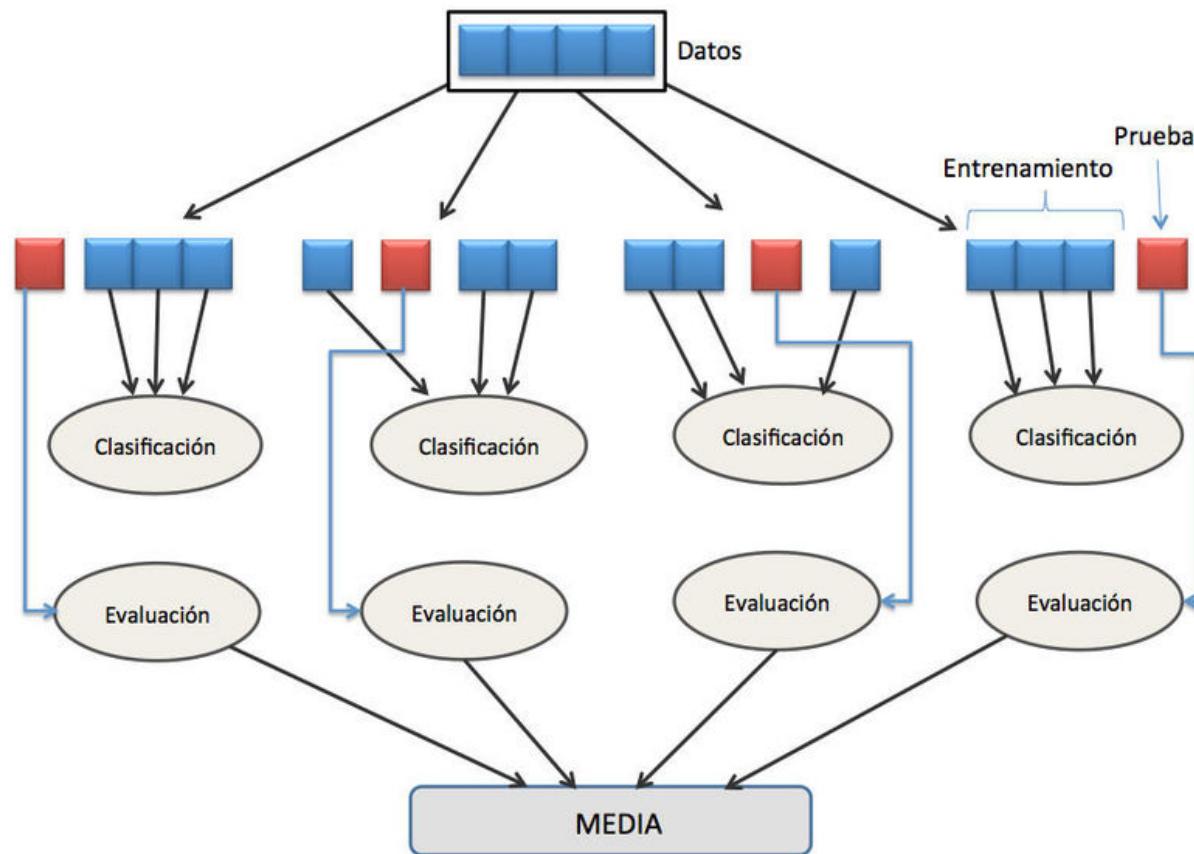
1. Se dividen los datos en k conjuntos, de aprox. el mismo tamaño



# Métodos de evaluación

## Validación cruzada (k-fold cross-validation)

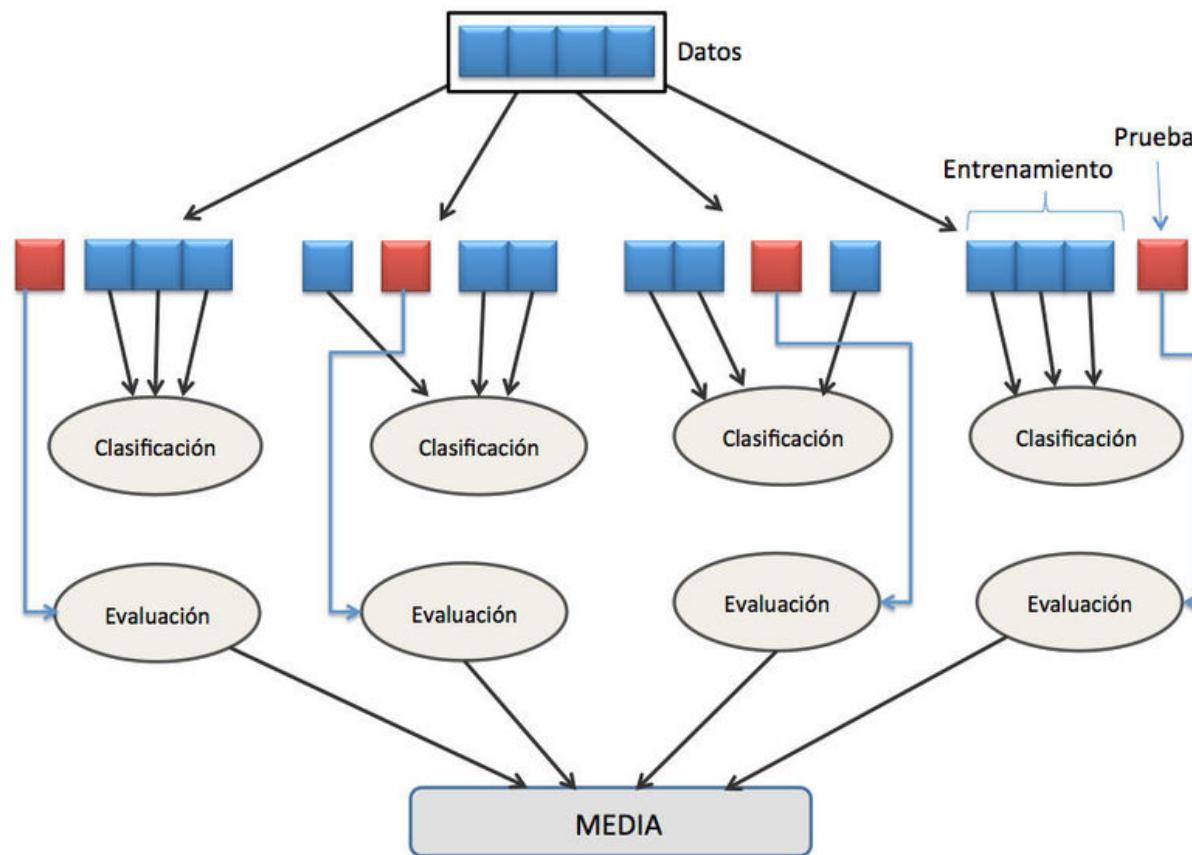
2. Para cada conjunto, se entrena con los  $k-1$  restantes y se evalúa con el subconjunto elegido



# Métodos de evaluación

## Validación cruzada (k-fold cross-validation)

3. La evaluación global es la media de las k evaluaciones obtenidas anteriormente



# Métodos de evaluación

## Validación cruzada (k-fold cross-validation)

- ❑ Si existen clases dentro de los datos puede existir sesgo
  - Solución: **validación cruzada estratificada**. Los subconjuntos se estratifican en función de la clase
- ❑ Valores normales para k son 5 ó 10
- ❑ Tiene un costo computacional bastante alto
- ❑ Para conjuntos de datos de tamaño “mediano”

# Métodos de evaluación

## Leaving-one-out

Un caso especial de validación cruzada en el que  $k$  es igual al número de registros y se evalúa un sólo un registro



# Métodos de evaluación

## Leaving-one-out

- Tiene la ventaja de que el proceso es determinista y de que en todo momento se utiliza el máximo posible de datos para la inducción del clasificador
- Tiene un alto costo computacional
- Se utiliza:
  - Cuando el conjunto de datos es pequeño
  - No hay muchos datos para evaluar

# Métodos de evaluación

## Bootstrap

Basado en el proceso de muestreo con reemplazo. Consiste en seleccionar, para un conjunto de tamaño  $n$ ,  $n$  casos que formarán el conjunto de entrenamiento

- ❑ Como es con reemplazo, algunos casos se elegirán más de una vez...
- ❑ ¿Cuántos casos habrá en el conjunto de entrenamiento?
  - Para un caso particular, **en una selección**

$$P(\text{elegido}) = \frac{1}{n} \quad P(\text{no elegido}) = 1 - \frac{1}{n}$$

luego

$$P(\text{no select. entrenamiento}) = \left(1 - \frac{1}{n}\right)^n \rightarrow e^{-1} \approx 0.368$$

# Métodos de evaluación

## Bootstrap

- ❑ ¿Cuántos casos habrá en el conjunto de entrenamiento?
  - El conjunto de entrenamiento tiene el 63.2% aprox.
  - El conjunto de test tiene el 36.8% aprox.
- ❑ Normalmente, se evalúan ambos conjuntos y se calcula una media ponderada
  - Si se realizan k iteraciones, se calcula la media
$$Ev = 0.368 \cdot Ev_{\text{Entrenamiento}} + 0.632 \cdot Ev_{\text{Test}}$$
- ❑ Para conjuntos de datos pequeños

# Medidas de evaluación

Primero, ¿qué criterio para evaluar?

- Precisión**, porcentaje de casos correctamente evaluados
- Coste** de una clasificación incorrecta
- Velocidad**, para construir el modelo o para usarlo
- Robustez**, capacidad para tratar ruido o valores nulos
- Escalabilidad**, utilidad con grandes cantidades de datos
- Interpretabilidad**, comprensibilidad del modelo obtenido
- Complejidad** del modelo (Navaja de Occam)

# Medidas de evaluación

## Matriz de confusión (confusion matrix)

	Predictión positiva	Predictión negativa
Real positiva	True Positive (TP)	False Negative (FN)
Real negativa	False Positive (FP)	True Negative (TN)

### Exactitud (accuracy)

Número de casos clasificados correctamente entre el número total de casos

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FN + FP}$$

# Medidas de evaluación

	Predictión positiva	Predictión negativa
Real positiva	True Positive (TP)	False Negative (FN)
Real negativa	False Positive (FP)	True Negative (TN)

## Exactitud (accuracy)

- No es un buen estimador si las clases están desequilibradas
- Por ejemplo,
  - 99% son de un tipo
  - 1 % son de otro
- Si el estimador predice siempre que la clase es la del primer tipo... acierta un 99%
- No se detectan clases del segundo tipo

# Medidas de evaluación

	Predicción positiva	Predicción negativa
Real positiva	True Positive (TP)	False Negative (FN)
Real negativa	False Positive (FP)	True Negative (TN)

## □ Otras medidas

- Precisión

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Exhaustividad

$$\text{Exhaustividad} = \frac{TP}{TP + FN}$$

- Media armónica

$$F = \frac{2}{\frac{1}{\text{Exhaustividad}} + \frac{1}{\text{Precision}}} = \frac{2 \text{ Exh. Prec.}}{\text{Exh.} + \text{Prec.}}$$

# Medidas de evaluación

## Matriz de confusión multidimensional

Matriz  $n \times n$ , donde  $n$  es el número de clases, que en la casilla  $ij$  tiene el número de casos que se clasifican en la clase  $i$  cuando realmente pertenecen a la clase  $j$

		Real		
		Salida	Observación	UCI
Predicción	Salida	71	3	1
	Observación	8	7	1
	UCI	4	2	3

# Medidas de evaluación

## Matriz de confusión multidimensional

Predichos \ ciertos	$y_1$	$y_2$	.....	$y_k$	$tot_{pred}$
$\hat{y}_1$	$n_{11}$	$n_{12}$	.....	$n_{1k}$	$m_1$
$\vdots$	$\vdots$	$\vdots$	.....	$\vdots$	$\vdots$
$\hat{y}_k$	$n_{k1}$	$n_{k2}$	.....	$n_{kk}$	$m_k$
$tot_{cier}$	$n_1$	$n_2$	.....	$n_k$	$n$

- Precisión a una clase  $P_i = \frac{n_{ii}}{m_i} \quad \forall i \in \{1, \dots, k\}$
- Recall/Exhaustividad a una clase  $R_i = \frac{n_{ii}}{n_i} \quad \forall i \in \{1, \dots, k\}$
- F a una clase  $F^i = \frac{2n_{ii}}{n_i + m_i} \quad \forall i \in \{1, \dots, k\}$
- F global  $F = \frac{1}{k} \sum_{i=1}^k F_1^i$

Precisión y recall global, también se puede calcular la media

# Medidas de evaluación

## Matriz de confusión multidimensional

Predichos \ ciertos	$y_1$	$y_2$	.....	$y_k$	$tot_{pred}$
$\hat{y}_1$	$n_{11}$	$n_{12}$	.....	$n_{1k}$	$m_1$
$\vdots$	$\vdots$	$\vdots$	.....	$\vdots$	$\vdots$
$\hat{y}_k$	$n_{k1}$	$n_{k2}$	.....	$n_{kk}$	$m_k$
$tot_{cier}$	$n_1$	$n_2$	.....	$n_k$	$n$

- Kappa de Cohen, compara secuencia de predichos y ciertos

$$\kappa = \frac{P_0 - P_e}{1 - P_e}$$

$$P_0 = \frac{1}{n} \sum_{i=1}^k n_{ii} \quad P_e = \frac{1}{n^2} \sum_{i=1}^k n_i m_i$$

Valor	Interpretación
$k < 0.2$	No acuerdo
$0.2 < k < 0.4$	Bajo
$0.4 < k < 0.6$	Medio
$0.6 < k < 0.8$	Bueno
$0.8 < k < 1$	Muy bueno

# Medidas de evaluación

## Ejemplo

Datos del iris utilizando *sepal length* y *sepal width*. Clasificador Naïve Bayes. 120 ejemplos entrenamiento y 30 datos test.

Predichos \ ciertos	Setosa	Versicolor	Viginica	
Setosa	10	0	0	10
Versicolor	0	7	5	12
Virginica	0	3	5	8
	10	10	10	30

	Accuracy	Recall	F-measure
Setosa	1	1	1
Versicolor	0.583	0.7	0.636
Virginica	0.625	0.5	0.556

## Datos Globales

$$P = 0.73 \quad R = 0.73 \quad F = 0.731 \quad \kappa = 0.6$$

# Medidas de evaluación

## Alternativa: Matriz de Costes

Matriz  $n \times n$ , donde  $n$  es el número de clases, que en la casilla  $ij$  tiene el coste de clasificar en la clase  $i$  cuando realmente pertenece a la clase  $j$

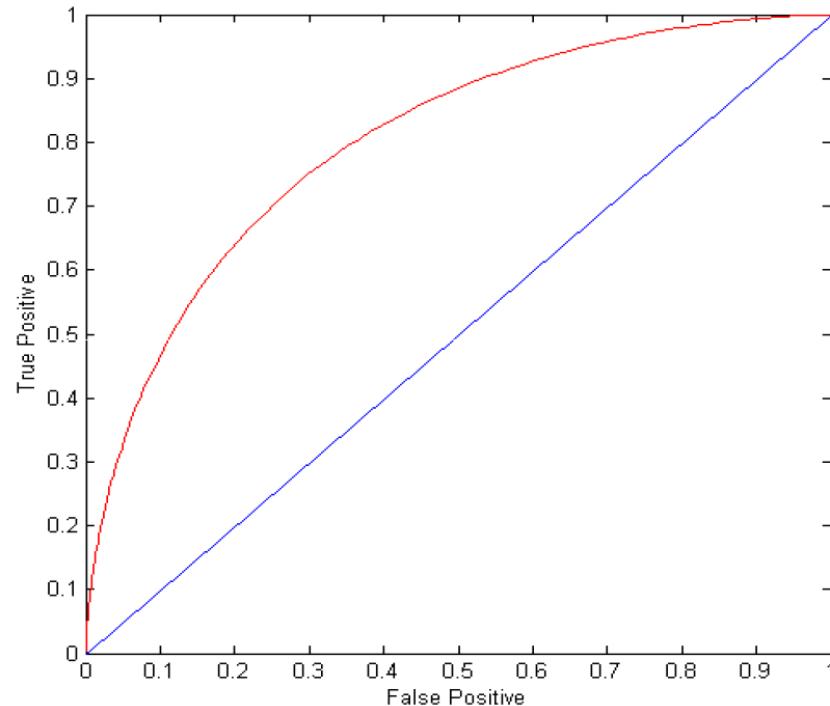
		C		
		Real		
Predicción	Salida	Observación	UCI	
	Salida	0€	5000€	500000€
	Observación	300€	0€	50000€
	UCI	800€	500€	0€

$$\text{Coste} = \sum_{i=1}^n \sum_{j=1}^n M_{ij} C_{ij}$$

# Comparación entre modelos

## Análisis/Curvas ROC (Receiver Operating Characteristics)

- ❑ Para problemas con dos clases
- ❑ Desarrolladas en los años 50 para analizar señales con ruido: caracterizar el compromiso entre aciertos y falsas alarmas
- ❑ Permiten comparar visualmente distintos modelos de clasificación



# Comparación entre modelos

Construcción de la función/curva ROC

1. Necesitamos una medida S para cada objeto. Por ejemplo, la probabilidad de que un objeto sea de una clase (Naïve Bayes)
2. Ordenamos según probabilidad

<i>Tuple #</i>	<i>Class</i>	<i>Prob.</i>
1	<i>P</i>	0.90
2	<i>P</i>	0.80
3	<i>N</i>	0.70
4	<i>P</i>	0.60
5	<i>P</i>	0.55
6	<i>N</i>	0.54
7	<i>N</i>	0.53
8	<i>N</i>	0.51
9	<i>P</i>	0.50
10	<i>N</i>	0.40

# Comparación entre modelos

Construcción de la función/curva ROC

3. Para cada valor  $t$ , calculamos su matriz de confusión prediciendo como positivos aquellos casos con medida S mayor que  $t$

<i>Tuple #</i>	<i>Class</i>	<i>Prob.</i>	<i>TP</i>	<i>FP</i>	<i>TN</i>	<i>FN</i>
1	<i>P</i>	0.90	1	0	5	4
2	<i>P</i>	0.80	2	0	5	3
3	<i>N</i>	0.70	2	1	4	3
4	<i>P</i>	0.60	3	1	4	2
5	<i>P</i>	0.55	4	1	4	1
6	<i>N</i>	0.54	4	2	3	1
7	<i>N</i>	0.53	4	3	2	1
8	<i>N</i>	0.51	4	4	1	1
9	<i>P</i>	0.50	5	4	1	0
10	<i>N</i>	0.40	5	5	0	0

# Comparación entre modelos

Construcción de la función/curva ROC

## 3. Calcular la matriz de confusión

	Predictión positiva	Predictión negativa
Real positiva	True Positive (TP)	False Negative (FN)
Real negativa	False Positive (FP)	True Negative (TN)

Normalizamos las medidas

- True Positive Rate  $TPR = TP/(TP + FN)$
- False Negative Rate
- False Positive Rate
- True Negative Rate

# Comparación entre modelos

Construcción de la función/curva ROC

## 3. Calcular la matriz de confusión

	Predictión positiva	Predictión negativa
Real positiva	True Positive (TP)	False Negative (FN)
Real negativa	False Positive (FP)	True Negative (TN)

Normalizamos las medidas

- True Positive Rate  $TPR = TP/(TP + FN)$
- False Negative Rate  $FNR = FN/(TP + FN)$
- False Positive Rate
- True Negative Rate

# Comparación entre modelos

Construcción de la función/curva ROC

## 3. Calcular la matriz de confusión

	Predictión positiva	Predictión negativa
Real positiva	True Positive (TP)	False Negative (FN)
Real negativa	False Positive (FP)	True Negative (TN)

Normalizamos las medidas

- True Positive Rate  $TPR = TP/(TP + FN)$
- False Negative Rate  $FNR = FN/(TP + FN)$
- False Positive Rate  $FPR = FP/(FP + TN)$
- True Negative Rate

# Comparación entre modelos

Construcción de la función/curva ROC

## 3. Calcular la matriz de confusión

	Predictión positiva	Predictión negativa
Real positiva	True Positive (TP)	False Negative (FN)
Real negativa	False Positive (FP)	True Negative (TN)

Normalizamos las medidas

- True Positive Rate  $TPR = TP/(TP + FN)$
- False Negative Rate  $FNR = FN/(TP + FN)$
- False Positive Rate  $FPR = FP/(FP + TN)$
- True Negative Rate  $TNR = TN/(FP + TN)$

# Comparación entre modelos

Construcción de la función/curva ROC

## 3. Calcular la matriz de confusión

	Predictión positiva	Predictión negativa
Real positiva	True Positive (TP)	False Negative (FN)
Real negativa	False Positive (FP)	True Negative (TN)

Normalizamos las medidas

- True Positive Rate  $TPR = TP/(TP + FN)$
  - False Negative Rate  $FNR = FN/(TP + FN)$
  - False Positive Rate  $FPR = FP/(FP + TN)$
  - True Negative Rate  $TNR = TN/(FP + TN)$
- ]} Suman 1                  ]} Suman 1

La información se recoge considerando sólo TPR y FPR

# Comparación entre modelos

## Construcción de la función/curva ROC

4. Para cada valor de probabilidad  $t$ , calculamos TPR y FPR

<i>Tuple #</i>	<i>Class</i>	<i>Prob.</i>	<i>TP</i>	<i>FP</i>	<i>TN</i>	<i>FN</i>	<i>TPR</i>	<i>FPR</i>
1	$P$	0.90	1	0	5	4	0.2	0
2	$P$	0.80	2	0	5	3	0.4	0
3	$N$	0.70	2	1	4	3	0.4	0.2
4	$P$	0.60	3	1	4	2	0.6	0.2
5	$P$	0.55	4	1	4	1	0.8	0.2
6	$N$	0.54	4	2	3	1	0.8	0.4
7	$N$	0.53	4	3	2	1	0.8	0.6
8	$N$	0.51	4	4	1	1	0.8	0.8
9	$P$	0.50	5	4	1	0	1.0	0.8
10	$N$	0.40	5	5	0	0	1.0	1.0

# Comparación entre modelos

Construcción de la función/curva ROC

- Dibujamos TPR en función de FPR. Si para un valor hay varias opciones, tomamos el máximo

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	<i>P</i>	0.90	1	0	5	4	0.2	0
2	<i>P</i>	0.80	2	0	5	3	0.4	0
3	<i>N</i>	0.70	2	1	4	3	0.4	0.2
4	<i>P</i>	0.60	3	1	4	2	0.6	0.2
5	<i>P</i>	0.55	4	1	4	1	0.8	0.2
6	<i>N</i>	0.54	4	2	3	1	0.8	0.4
7	<i>N</i>	0.53	4	3	2	1	0.8	0.6
8	<i>N</i>	0.51	4	4	1	1	0.8	0.8
9	<i>P</i>	0.50	5	4	1	0	1.0	0.8
10	<i>N</i>	0.40	5	5	0	0	1.0	1.0

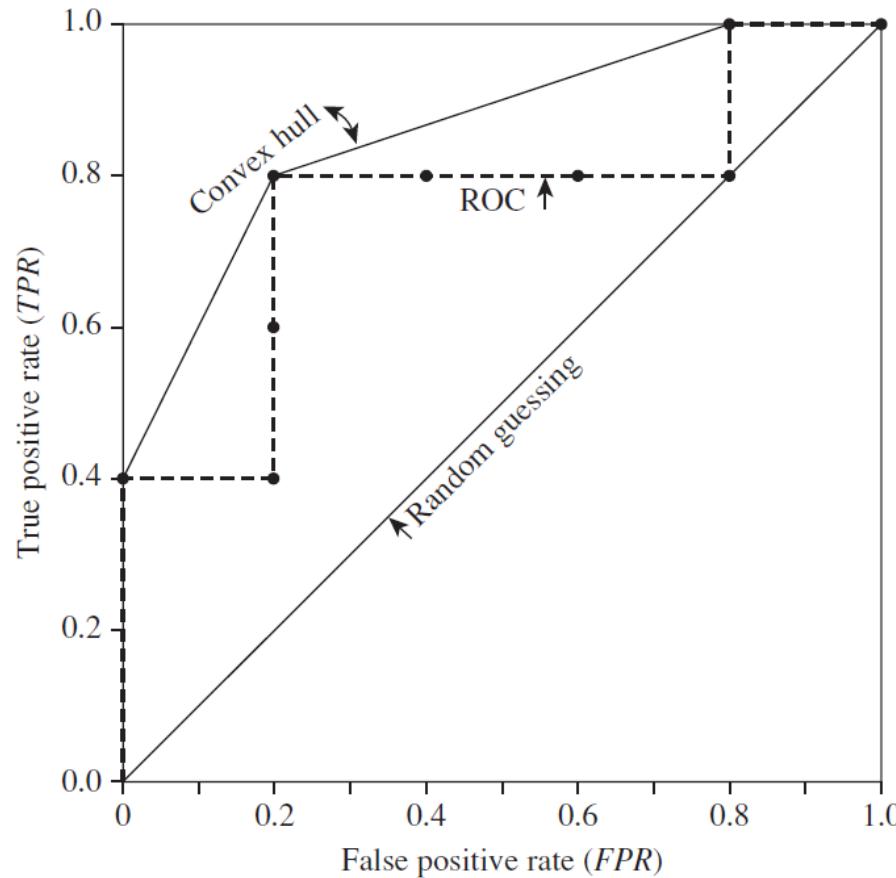
Al o le corresponde 0.4

# Comparación entre modelos

## Construcción de la función/curva ROC

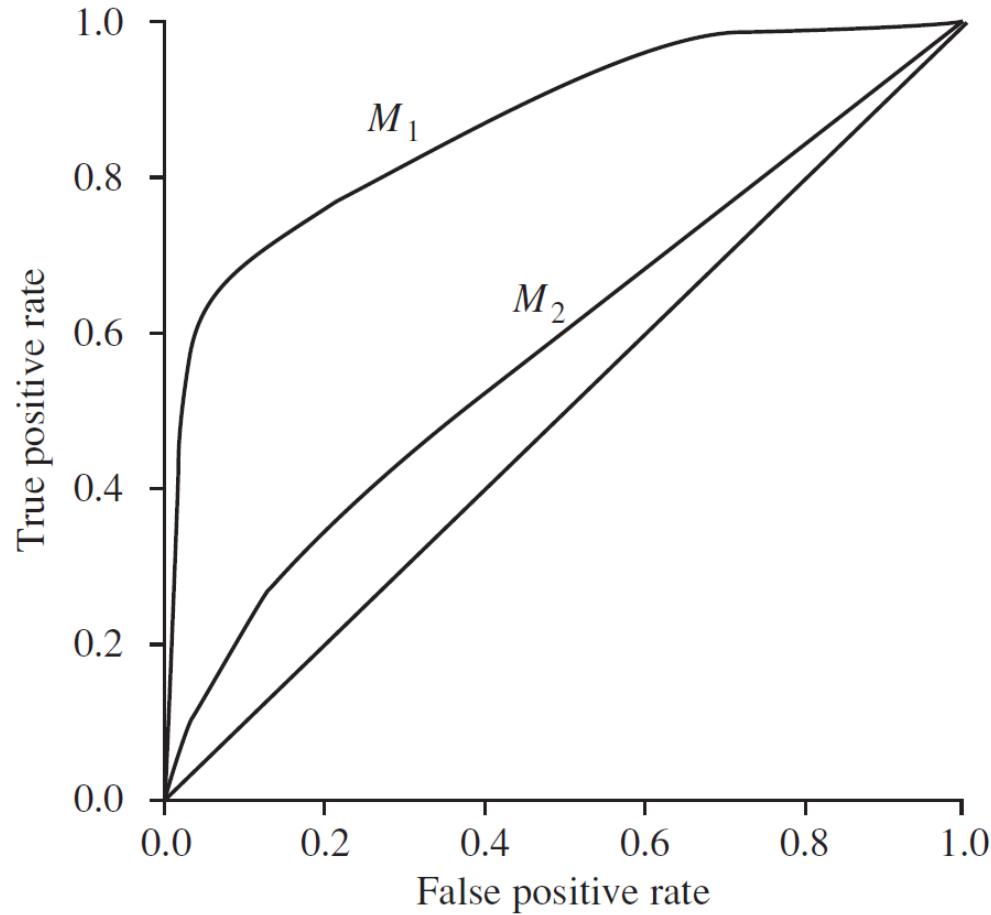
5. Dibujamos TPR en función de FPR

También se considera la envolvente convexa



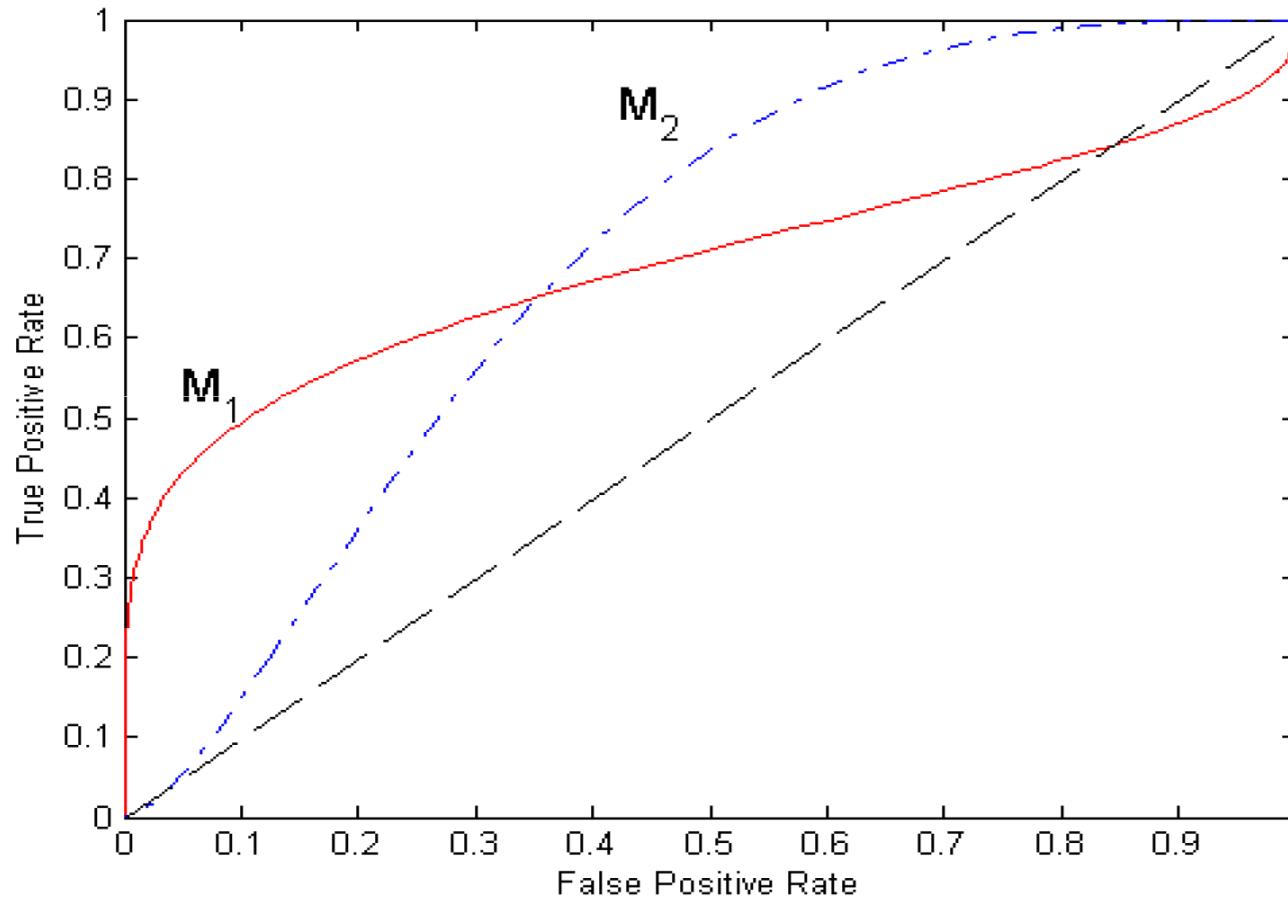
# Comparación entre modelos

Comparamos los modelos viendo qué función es mayor



# Comparación entre modelos

Normalmente, ningún modelo es consistentemente mejor que el otro. Por eso, se considera como medida de exactitud el área bajo la curva



# Comparación de modelos

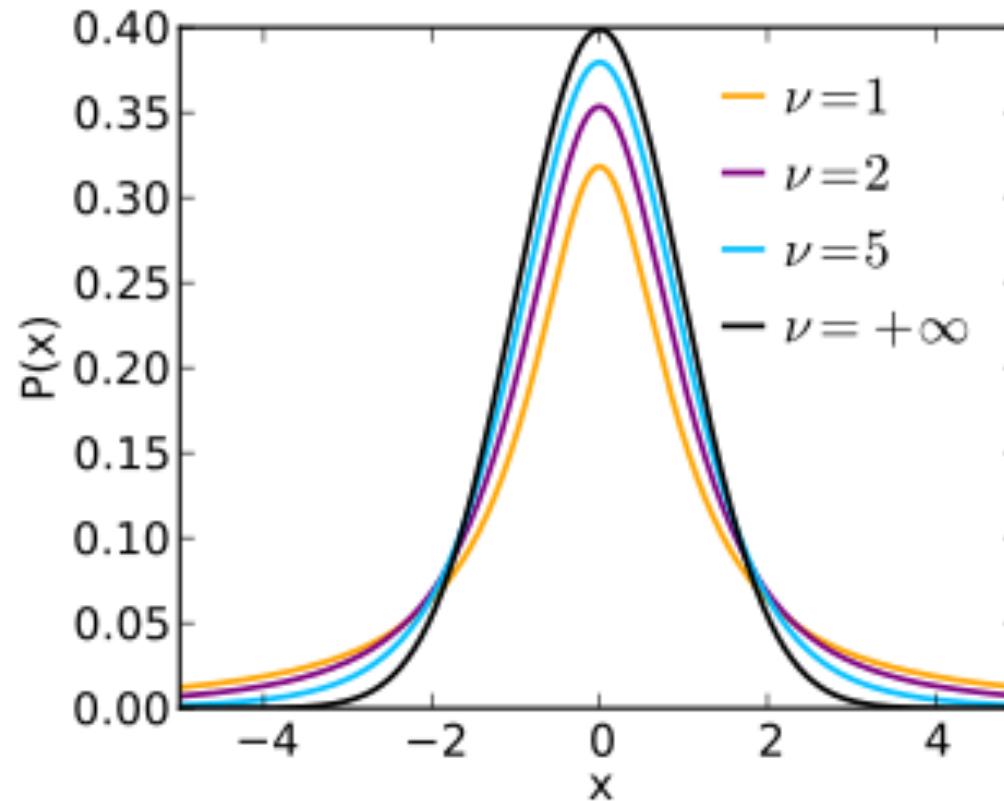
Supongamos dos modelos  $M_1$  y  $M_2$  aprendidos para un conjunto de datos

- Evaluamos el primer método** mediante validación cruzada obteniendo una evaluación media (error)  $e_1$
- Evaluamos el segundo método** mediante validación cruzada obteniendo una evaluación media (error)  $e_2$
- $e_1$  es menor que  $e_2$**

¿Es realmente  $M_1$  mejor que  $M_2$ ? ¿La diferencia entre métodos es estadísticamente significativa?

# Comparación de modelos

Para comprobar si los resultados de los modelos son diferentes podemos utilizar la **distribución t de Student**



# Comparación de modelos

Para comprobar si los resultados de los modelos son diferentes podemos utilizar la **distribución t de Student**

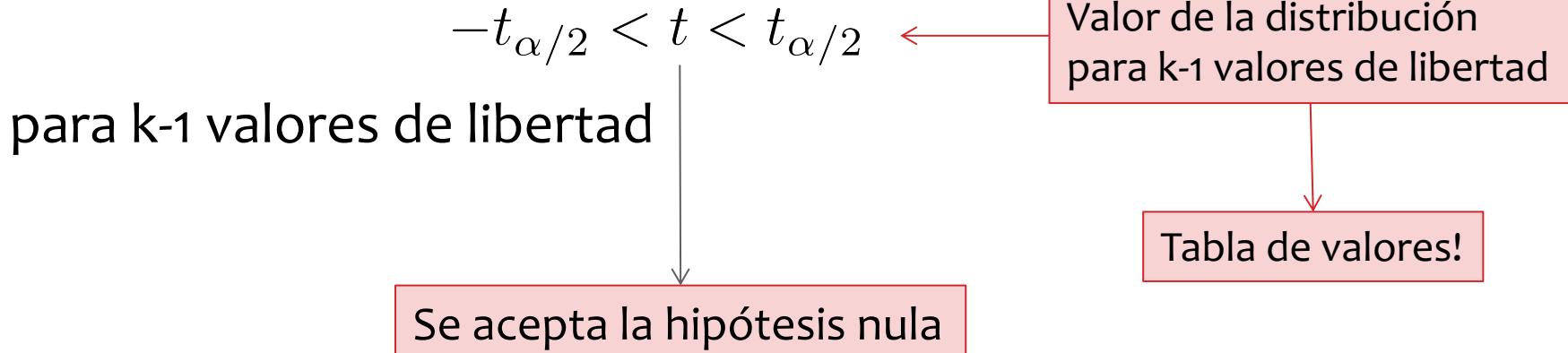
- Sean  $e_{11}, e_{12}, e_{13}, \dots$  la evaluación de cada modelo asociado a la k-validación cruzada del primer modelo
- Sean  $e_{21}, e_{22}, e_{23}, \dots$  la evaluación de cada modelo asociado a la k-validación cruzada del segundo modelo

$$t = \frac{\bar{e}_1 - \bar{e}_2}{\sqrt{Var(e_1 - e_2)/k}}$$

# Comparación de modelos

- Hipótesis nula,  $H_0 : \bar{e}_1 - \bar{e}_2 = 0$
- Hipótesis no nula,  $H_1 : \bar{e}_1 - \bar{e}_2 \neq 0$

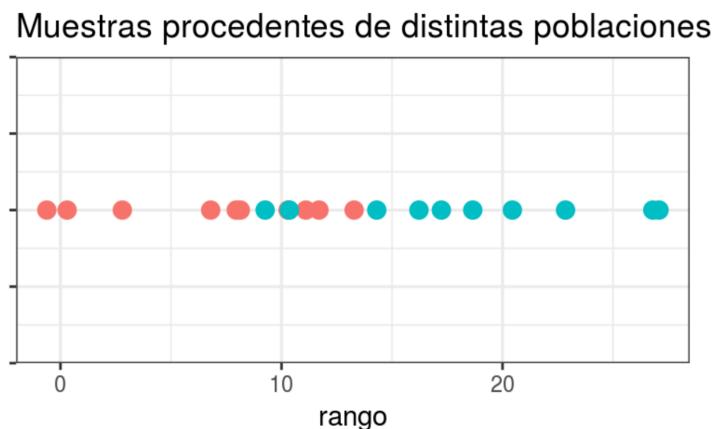
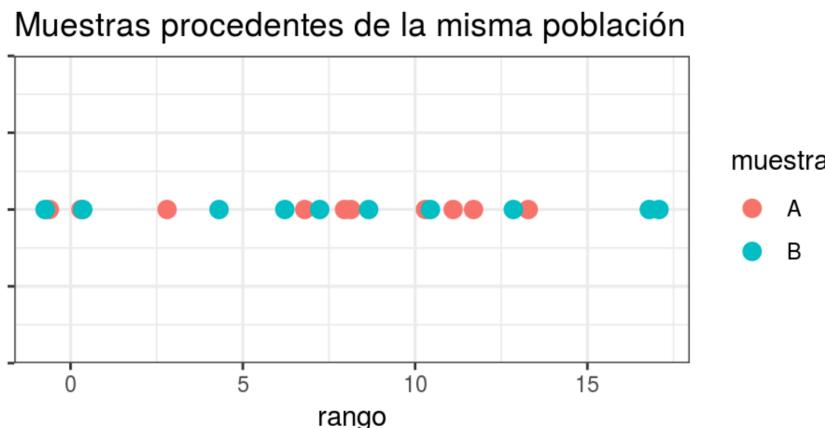
Para un nivel de significancia  $\alpha$  (normalmente 0.01 ó 0.05), comprobamos si



# Comparación de modelos

## Test de Wilcoxon

Basado en que si dos muestras proceden de la misma población, al juntar todas las observaciones y ordenarlas, deberían estar intercaladas aleatoriamente



# Comparación de modelos

Para una **comparación genérica** entre clasificadores

- Se elige un conjunto de problemas
- Se elige una medida de calidad. Habitualmente precisión.
- Se evaluán y se realiza algún test estadístico (diferencia de medias, ANOVA, etc. ) que permita comparar resultados

No es fácil comparar clasificadores en general, lo más probable es que unos trabajen mejor que otros según la clase de problemas

# Índice

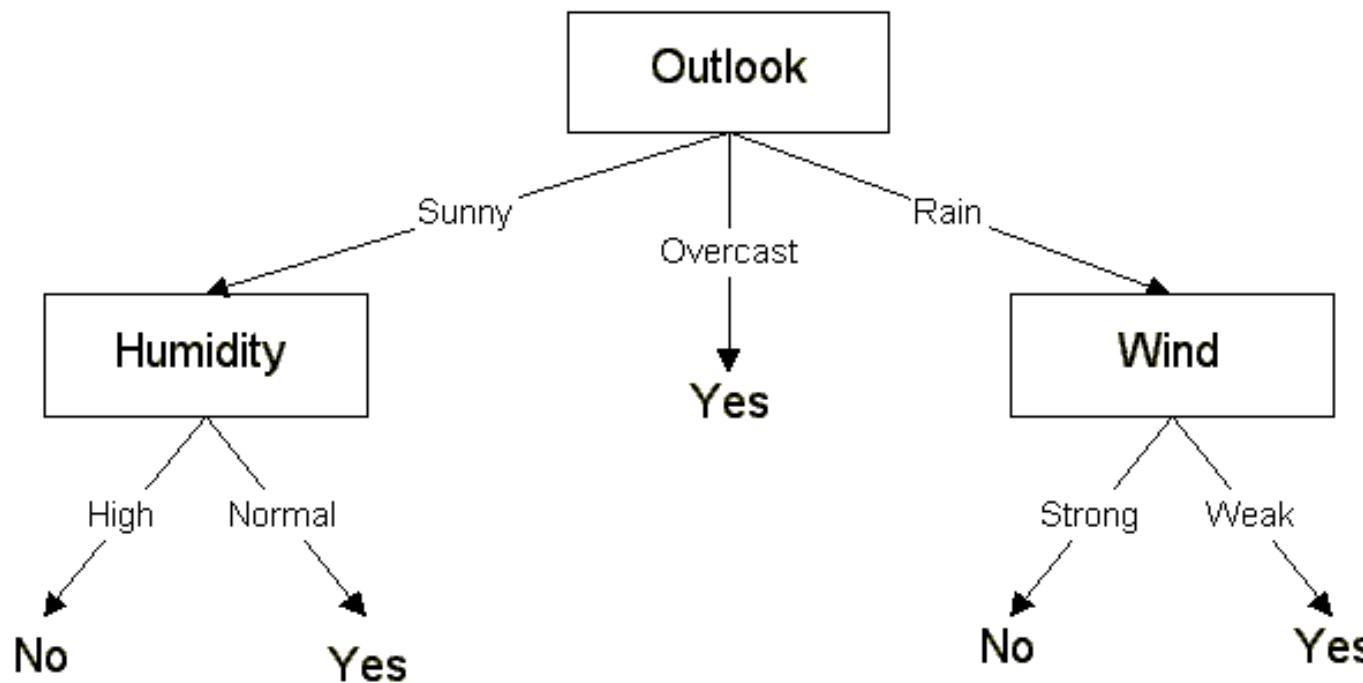
- ❑ El problema de la clasificación
- ❑ **Clasificación con árboles de decisión**
  - Concepto de árbol de decisión
  - Construcción de árboles de decisión
  - Algoritmos destacados
- ❑ Clasificación con reglas
- ❑ Clasificación con métodos Bayesianos
- ❑ Otros clasificadores

# Objetivos

- ❑ Entender qué es un árbol de decisión y cómo se utiliza en las tareas de clasificación
- ❑ Conocer los criterios y métricas que se utilizan para construir un árbol de decisión
- ❑ Conocer algunos de los algoritmos más famosos de aprendizaje para árboles de decisión
- ❑ Entender el concepto de regla de clasificación como generalización de árbol de decisión
- ❑ Conocer las formas usuales de obtención de reglas: mediante árboles de decisión y mediante cobertura

# Árboles de decisión

Un árbol de decisión es un conjunto de condiciones organizadas en una estructura jerárquica (árbol), de tal manera que la decisión final a tomar se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguna de sus hojas

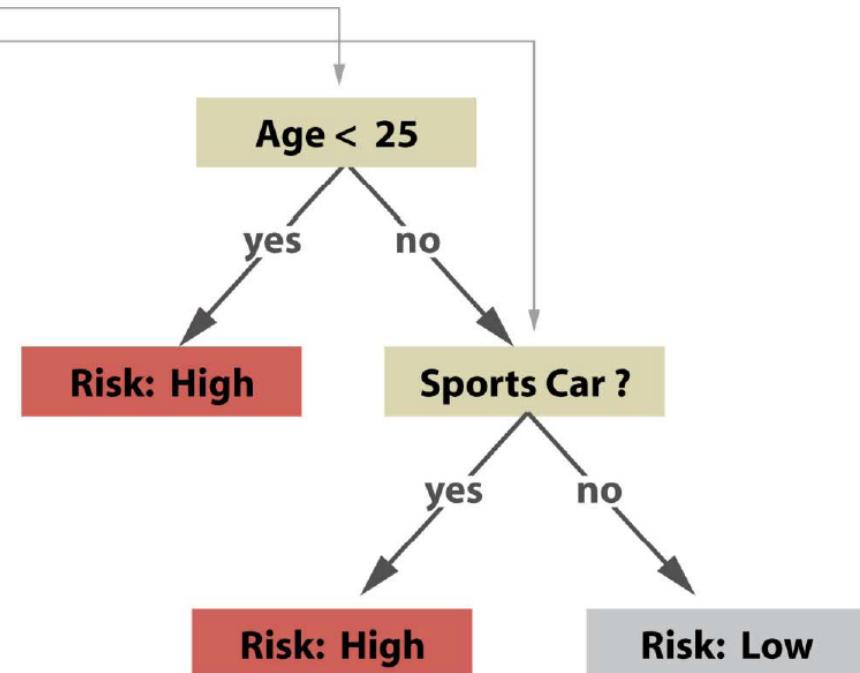


# Árboles de decisión

## Estructura

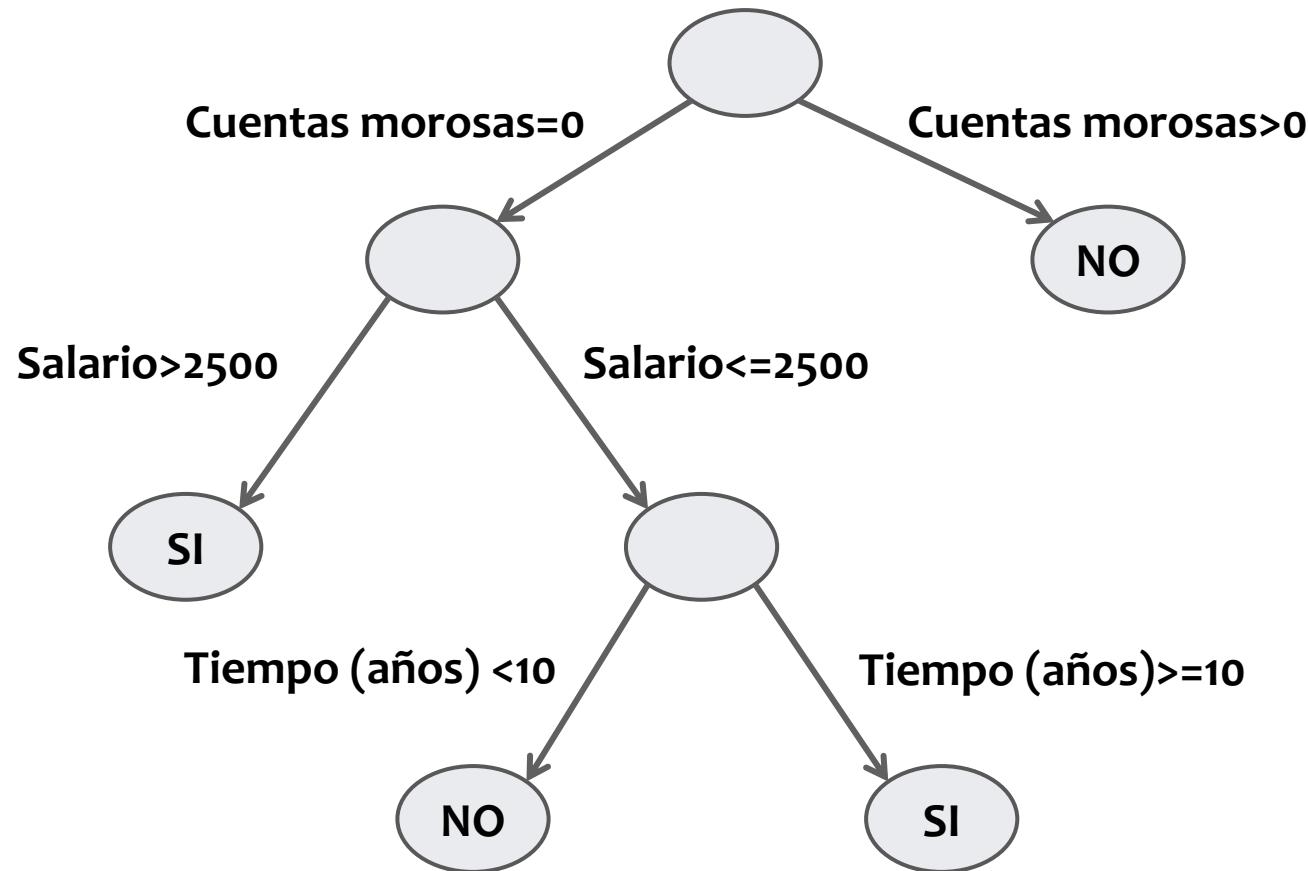
- Cada hoja es una clase de la variable objeto de la clasificación
- Cada nodo especifica una prueba simple a comprobar
- Los descendientes de cada nodo son los posibles resultados de la prueba del nodo (excluyentes)

Age	Car Type	Risk
23	family	High
17	sports	High
43	sports	High
68	family	Low
32	truck	Low
20	family	High



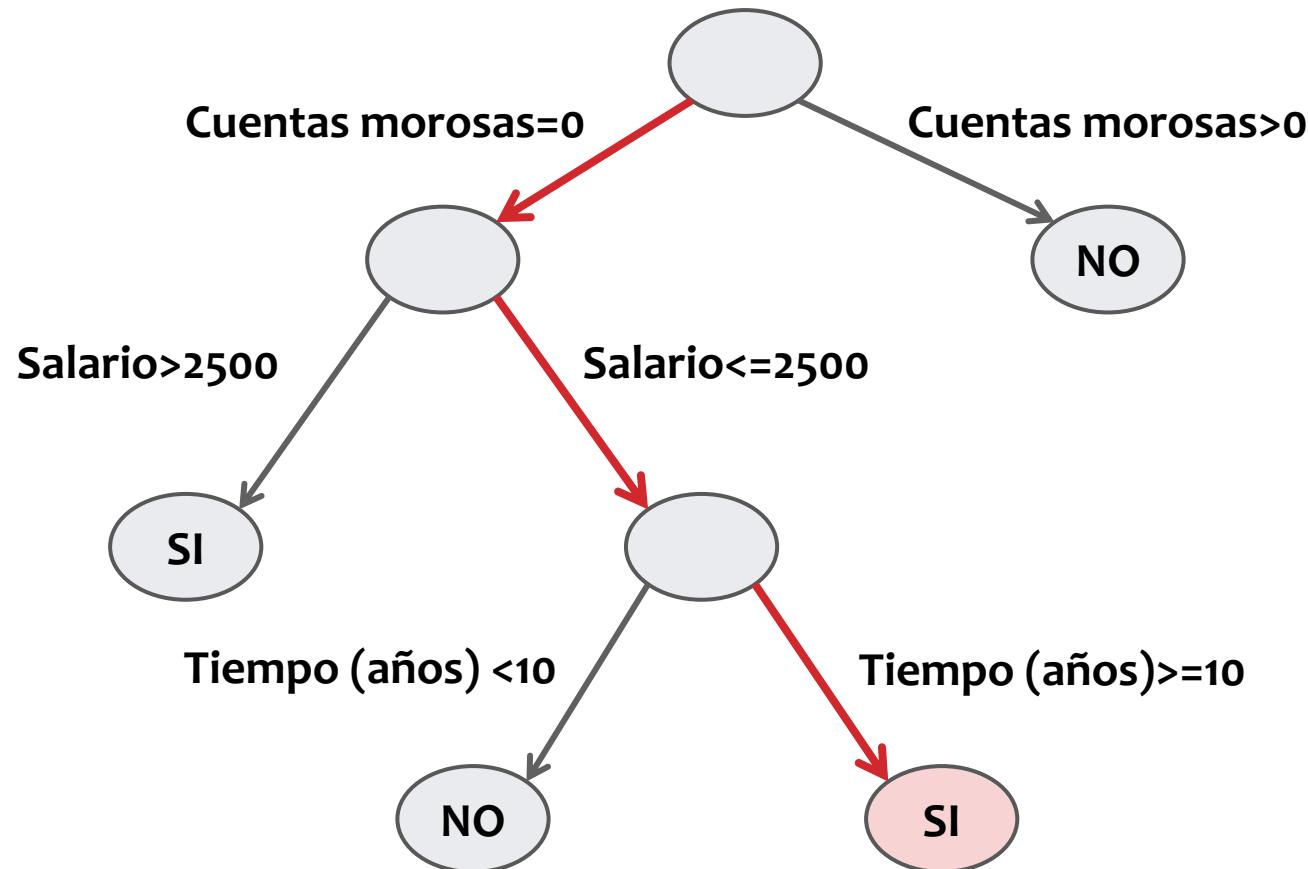
# Árboles de decisión

Concesión de créditos

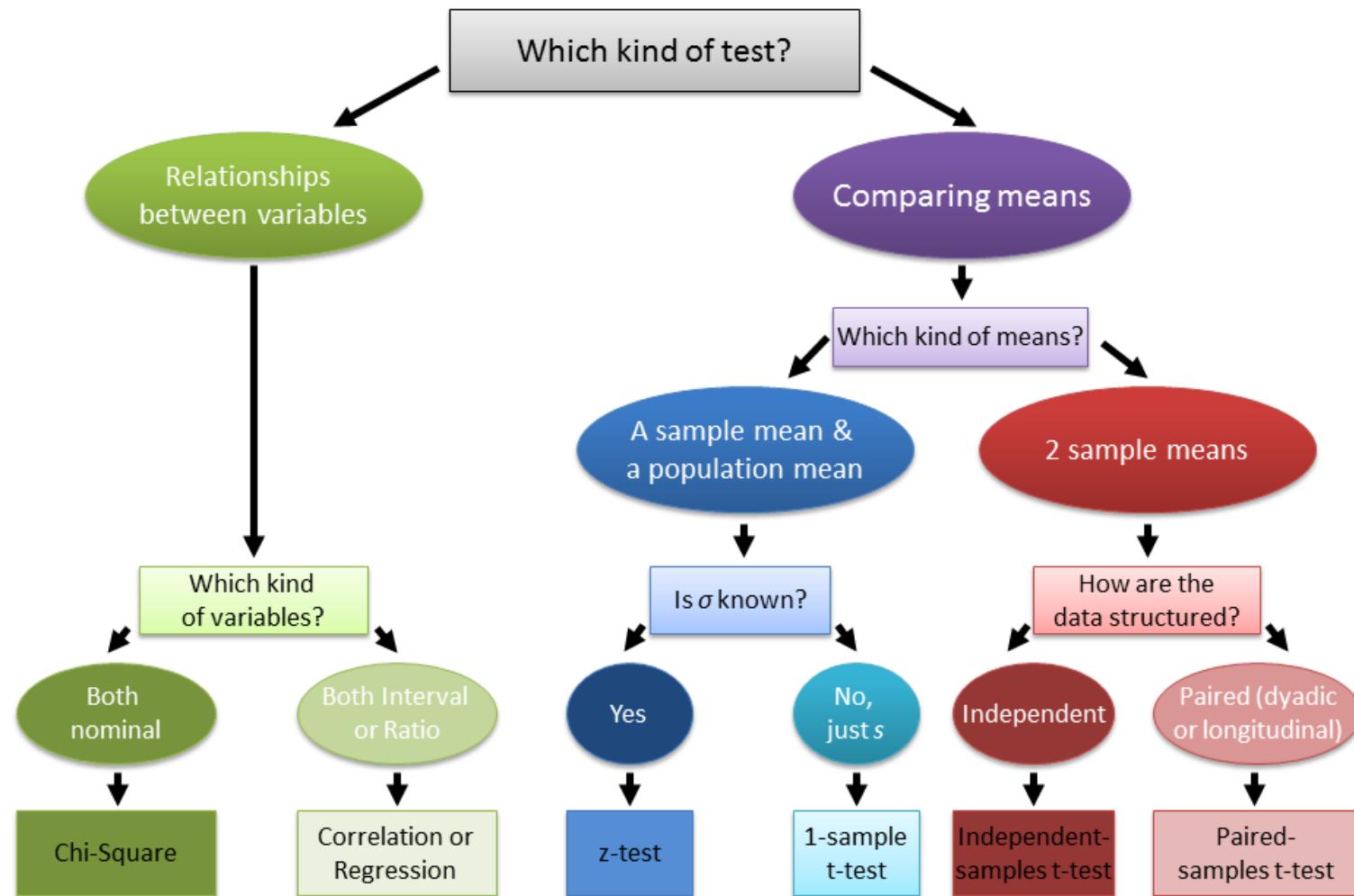


# Árboles de decisión

ID	Euros	Tiempo	Salario	Interés	Cuentas morosas
210	150000€	30	2000	1.5	0



# Árboles de decisión



# Construcción de árboles de decisión

- ❑ Estrategia greedy (problema NP)
- ❑ Algoritmo *divide y vencerás*:
  - Comenzamos con todos los datos de entrenamiento en la raíz
  - Se van dividiendo en función del atributo que se seleccione para ramificar el árbol en cada nodo
  - Los atributos se eligen en función de una heurística
- ❑ ¿Cuándo se detiene la construcción del árbol de decisión?
  - Todos los ejemplos que quedan pertenecen a la misma clase (se añade una hoja al árbol con la etiqueta de la clase)
  - No quedan datos que clasificar (se toma la clase mayoritaria en el nodo padre... o cualquier valor)

# Construcción de árboles de decisión

## Algoritmo genérico de aprendizaje

---

**Algorithm** Algoritmo genérico de aprendizaje

---

**Input:**  $E$  ejemplos,  $X$  atributos,  $C$  clases posibles y  $N$  nodo raíz

**Output:**  $A$  árbol de decisión

- 1: **if** los elementos de  $E$  pertenecen a la misma clase  $c$  **then**
  - 2:     Asignar  $c$  a  $N$
  - 3:     **return**  $N$
  - 4:  $T \leftarrow \text{SELEC\_ATRIBUTO}(X)$
  - 5:  $X \leftarrow X - \{T\}$
  - 6: **for** cada valor  $a_i$  de  $T$  **do**
  - 7:     Crear un nodo hijo  $H_i$  de  $N$
  - 8:     A la rama añadirle el test  $T = a_i$
  - 9:     Aplicar el algoritmo a  $E, X, C$  y  $H_i$
  - 10: **return**  $N$
-

# Construcción de árboles de decisión

## Algoritmo genérico de aprendizaje

---

**Algorithm** Algoritmo genérico de aprendizaje

---

**Input:**  $E$  ejemplos,  $X$  atributos,  $C$  clases posibles y  $N$  nodo raíz

**Output:**  $A$  árbol de decisión

- 1: **if** los elementos de  $E$  pertenecen a la misma clase  $c$  **then**
  - 2:     Asignar  $c$  a  $N$
  - 3:     **return**  $N$
  - 4:      $T \leftarrow \text{SELEC\_ATRIBUTO}(X)$  ← ¿Cómo seleccionar?
  - 5:      $X \leftarrow X - \{T\}$
  - 6:     **for** cada valor  $a_i$  de  $T$  **do**
  - 7:         Crear un nodo hijo  $H_i$  de  $N$
  - 8:         A la rama añadirle el test  $T = a_i$
  - 9:         Aplicar el algoritmo a  $E, X, C$  y  $H_i$
  - 10:     **return**  $N$
-

# Construcción de árboles de decisión

Ejemplo, asignación de créditos

crédito	ingresos	propietario	Gastos-mensuales
N	Bajos	N	Altos
N	Bajos	S	Altos
N	Medios	S	Altos
N	Medios	N	Altos
N	Altos	N	Altos
S	Altos	S	Altos
N	Bajos	N	Bajos
N	Medios	N	Bajos
N	Altos	N	Bajos
S	Medios	S	Bajos

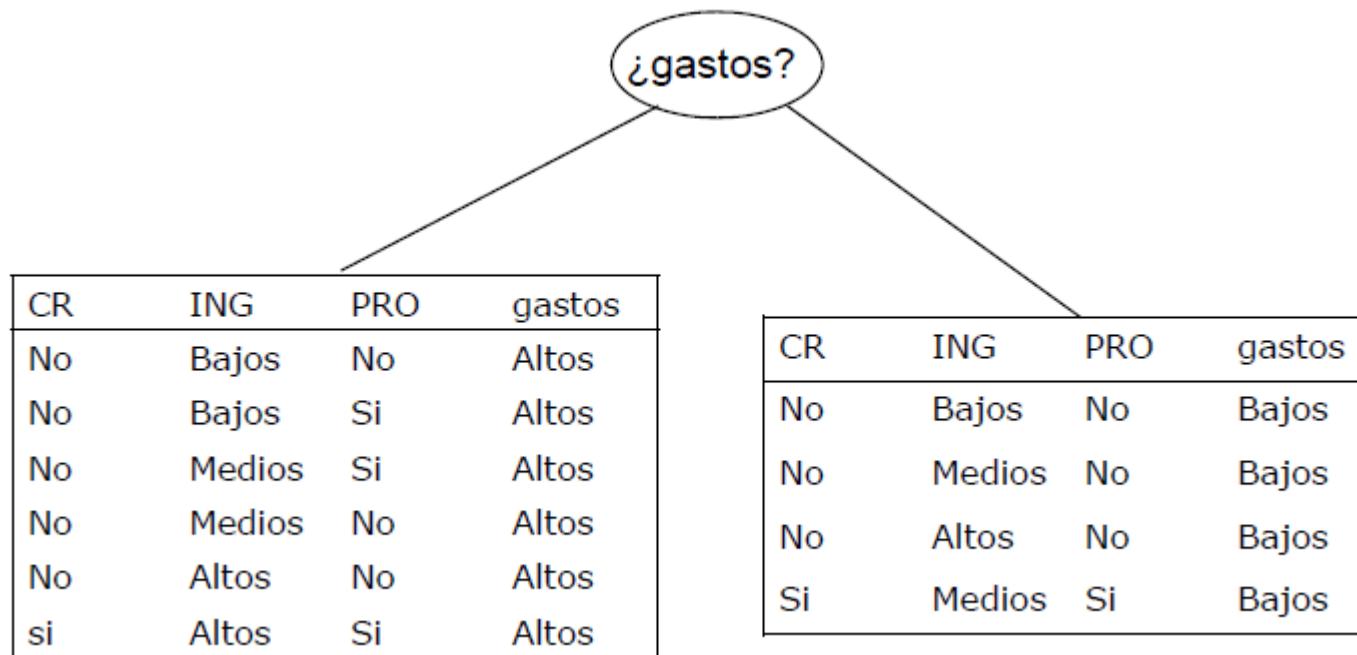
# Construcción de árboles de decisión

## 1. Inicio del algoritmo sobre nodo raíz

Crédito	ingresos	propietario	Gastos-mensuales
N	Bajos	N	Altos
N	Bajos	S	Altos
N	Medios	S	Altos
N	Medios	N	Altos
N	Altos	N	Altos
S	Altos	S	Altos
N	Bajos	N	Bajos
N	Medios	N	Bajos
N	Altos	N	Bajos
S	Medios	S	Bajos

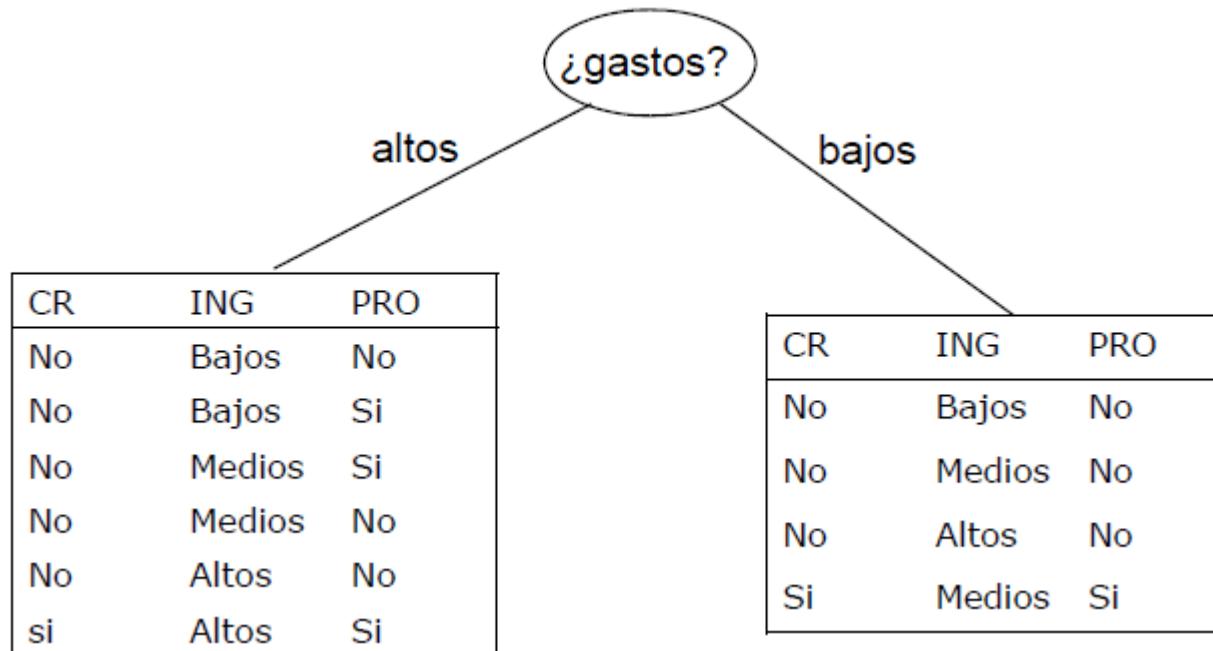
# Construcción de árboles de decisión

2. Seleccionamos *gastos-mensuales* como atributo



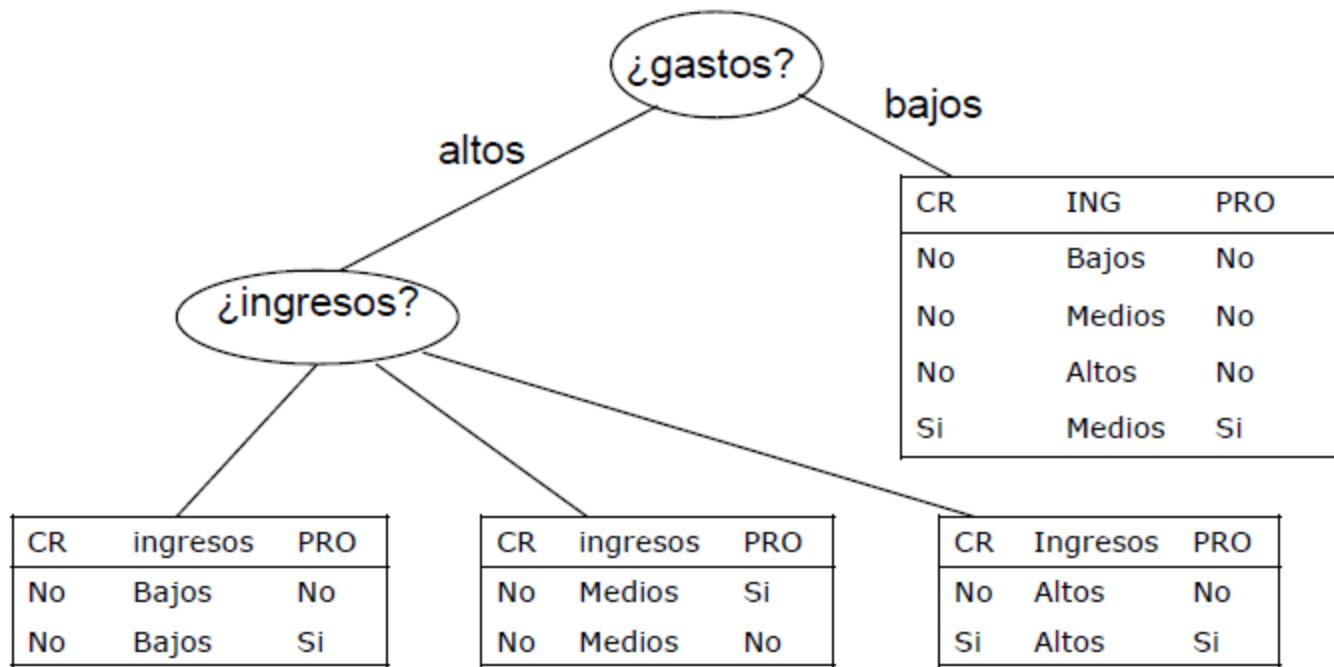
# Construcción de árboles de decisión

## 3. Asignamos test a las aristas



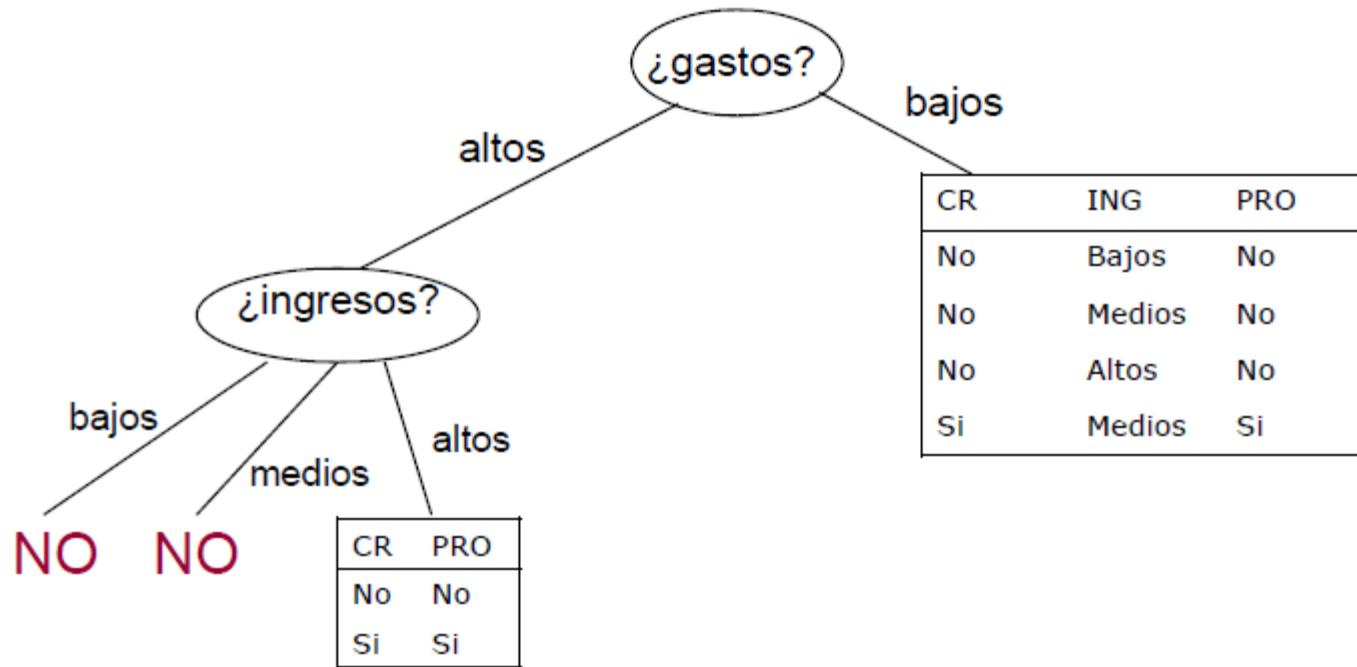
# Construcción de árboles de decisión

4. Seleccionamos *ingresos* para *gastos=altos*



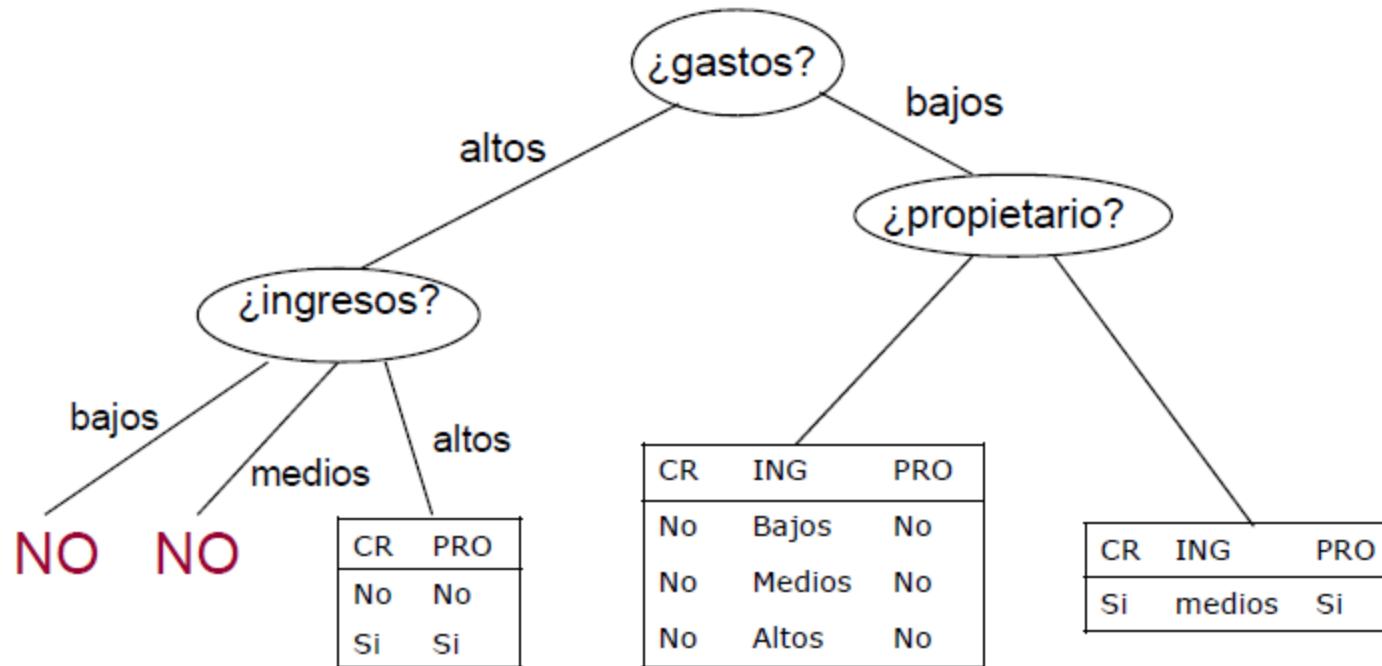
# Construcción de árboles de decisión

5. Se crean nodos hoja para *ingresos=bajos* y *ingresos=medios*



# Construcción de árboles de decisión

6. Seleccionamos propietario para  $gastos=bajos$



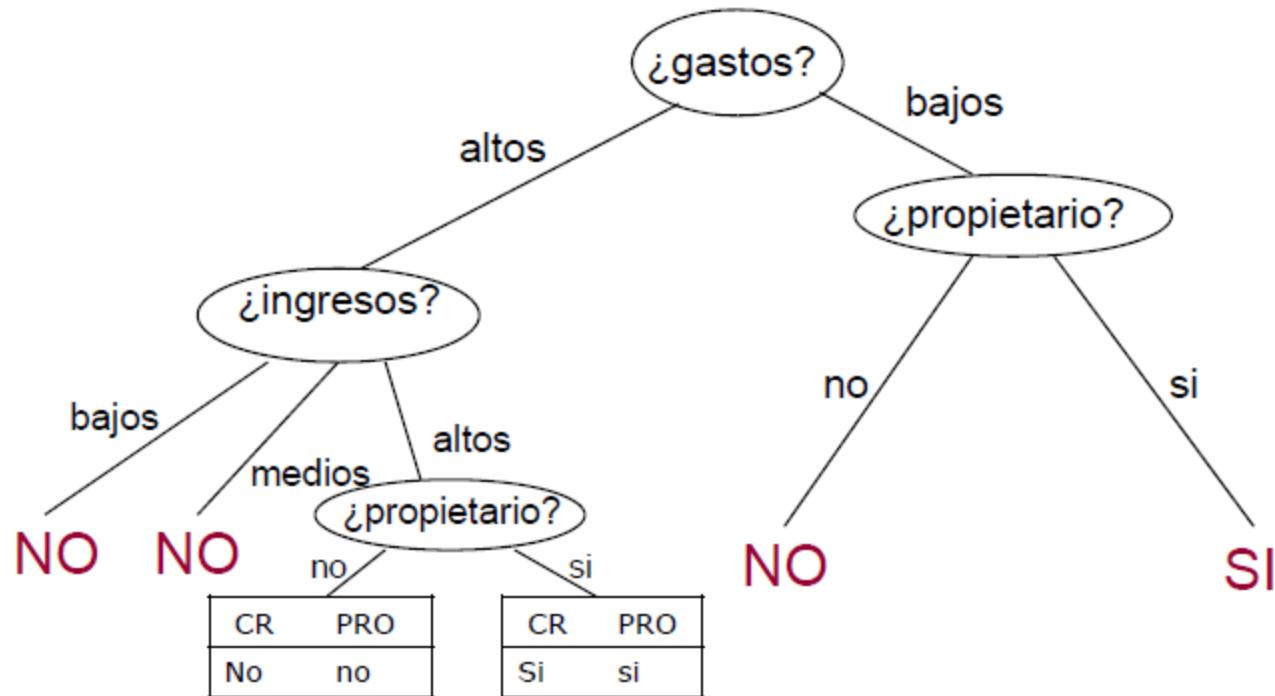
# Construcción de árboles de decisión

## 7. Creamos nodos hoja para propietario



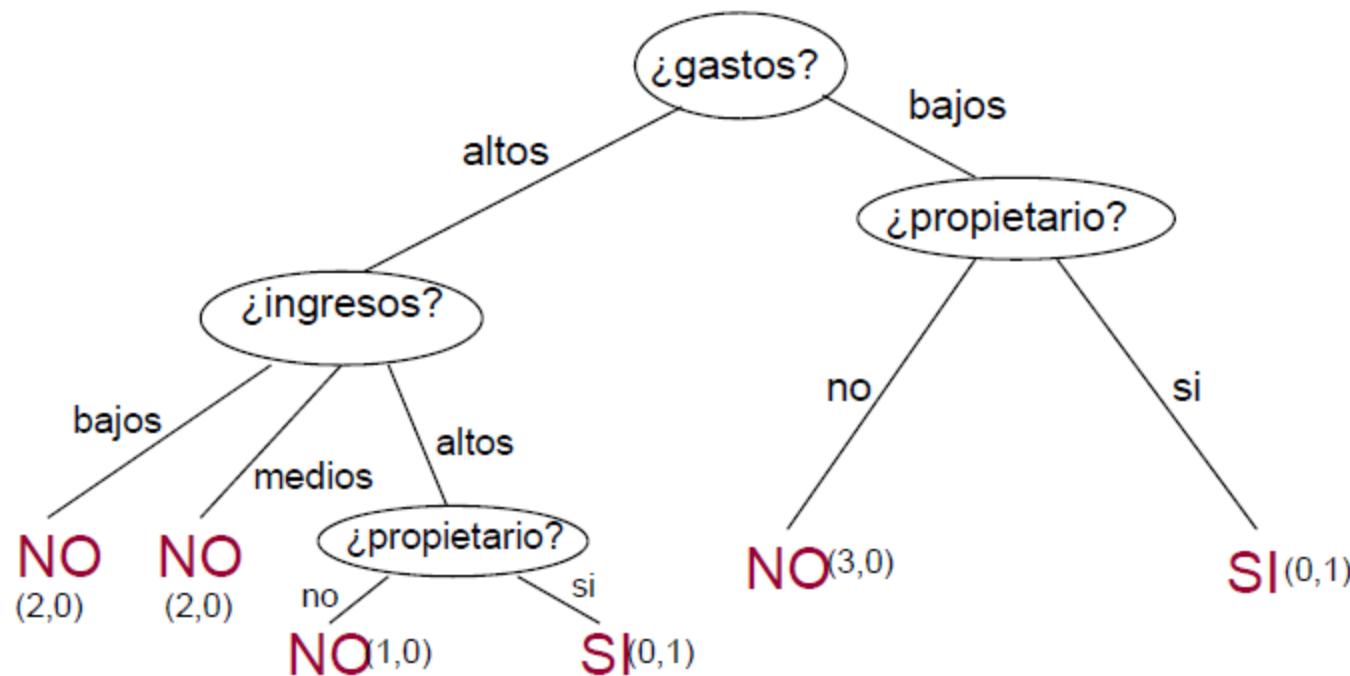
# Construcción de árboles de decisión

8. Seleccionamos propietario para  $gastos=altos$   $ingresos=altos$



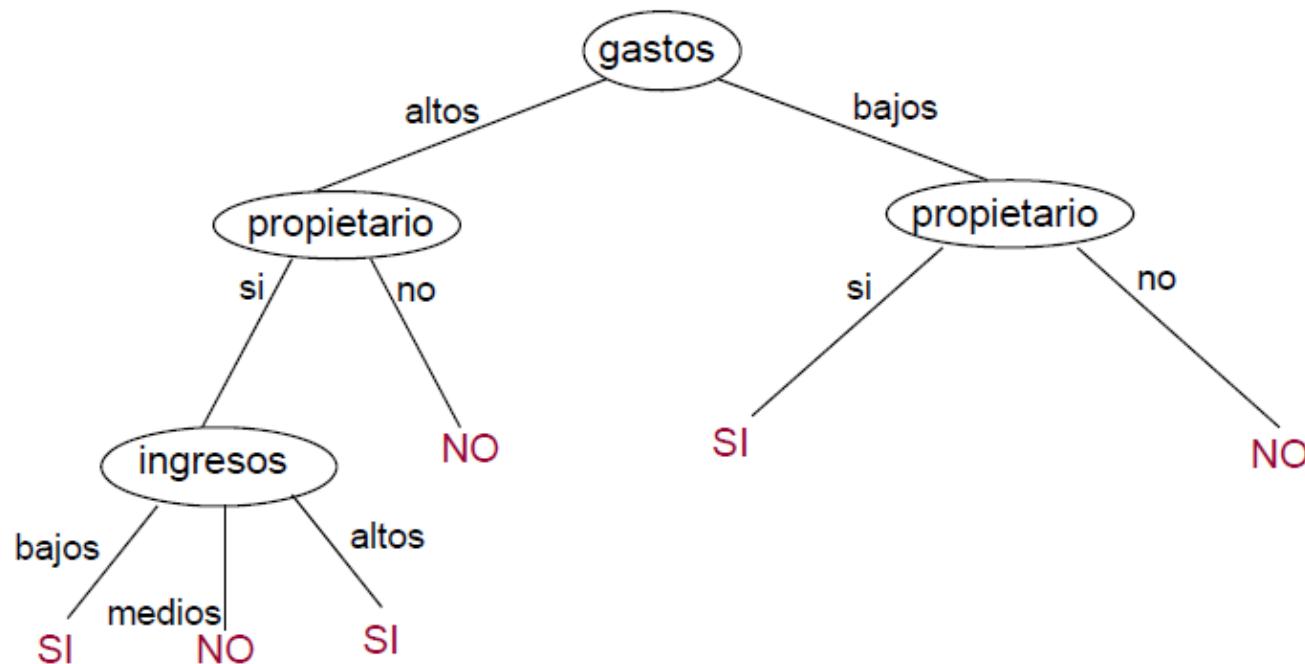
# Construcción de árboles de decisión

9. Creamos nodos hoja y terminamos árbol



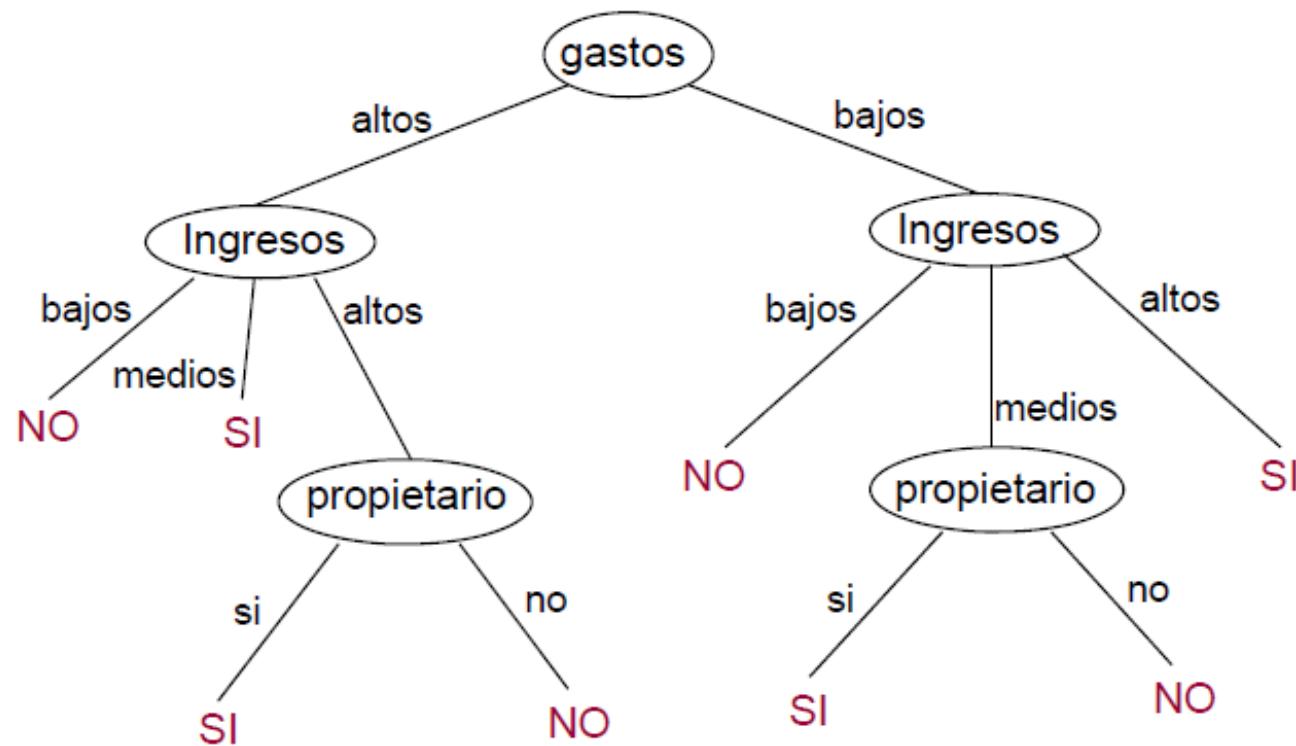
# Construcción de árboles de decisión

Dependiendo del criterio de selección construimos diferentes árboles



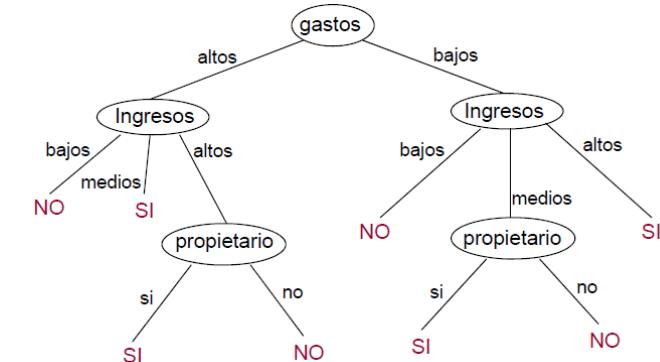
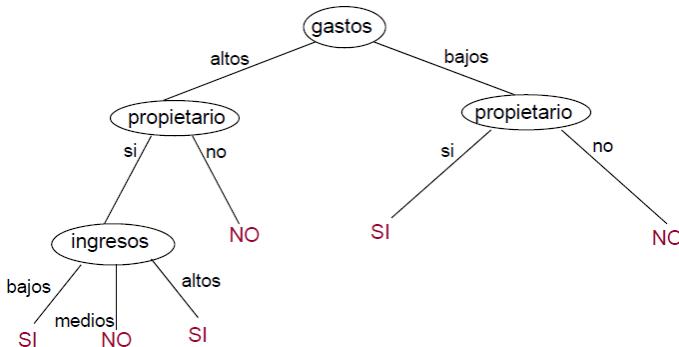
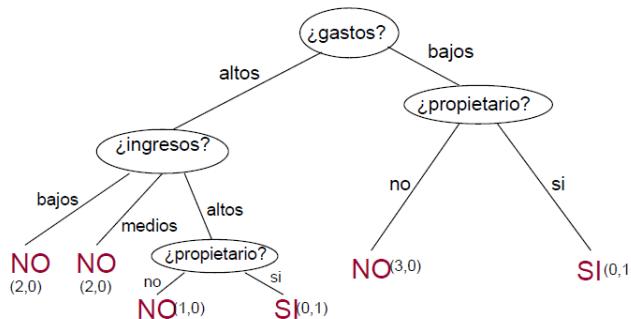
# Construcción de árboles de decisión

Dependiendo del criterio de selección construimos diferentes árboles



# Construcción de árboles de decisión

¿Cuál es mejor?



- 6 reglas
- 2.33 premisas por regla

- 6 reglas
- 2.5 premisas por regla

- 8 reglas
- 2.5 premisas por regla

Dependiendo de la selección de atributos obtenemos diferentes complejidades

# Construcción de árboles de decisión

Los dos puntos para el diseño de un algoritmo concreto:

- ❑ Cómo realizar la **partición en los posibles valores** de un atributo
  - En cuántos valores posibles
  - Qué valores. Si es nominal, en principio, parece sencillo...
  - ... si es numérico, no está tan claro
- ❑ **Criterio de selección de atributos**
  - En cada nodo, qué atributo elijo para ramificar el árbol

# Partición de los atributos

Cuestiones a considerar:

## Número de particiones

### Muchas

- Árboles más precisos
- Árboles más expresivos

### Pocas

- Algoritmos más eficientes
- Árboles que tienden menos al sobreaprendizaje

Se busca un equilibrio!

# Partición de los atributos

Cuestiones a considerar:

## Tipo del atributo

### Nominal o numérico finito

- Si tiene valores  $\{a_1, a_2, \dots, a_t\}$  lo normal es considerar la partición como  $\{X = a_1; X = a_2; \dots; X = a_t\}$
- Si se buscan árboles binarios, entonces consideramos condiciones

$$\{X = a_1; X \neq a_1\}, \{X = a_2; X \neq a_2\}, \dots$$

En realidad, son árboles equivalentes

- Se puede buscar el agrupamiento de varios valores

# Partición de los atributos

Cuestiones a considerar:

## Tipo del atributo

### Numérico infinito

- Se busca discretizar con particiones del tipo  $\{X > a; X \leq a\}$
- Para elegir los valores se puede buscar el valor intermedio entre los posibles valores encontrados en los ejemplos



# Selección de atributos

Heurísticas/métricas para selección de atributos

- Ganancia de información (Information Gain)
  - ID3, C4.5, C5.0
- Índice de Gini
  - CART, SLIQ, SPRINT
- Otros
  - MDL (Mínimum Description Length), Chi-cuadrado

# Selección de atributos

Basado en el clasificador oneR

En cada nodo seleccionamos el atributo con menor error

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

# Selección de atributos

Basado en el clasificador oneR

En cada nodo seleccionamos el atributo con menor error



		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

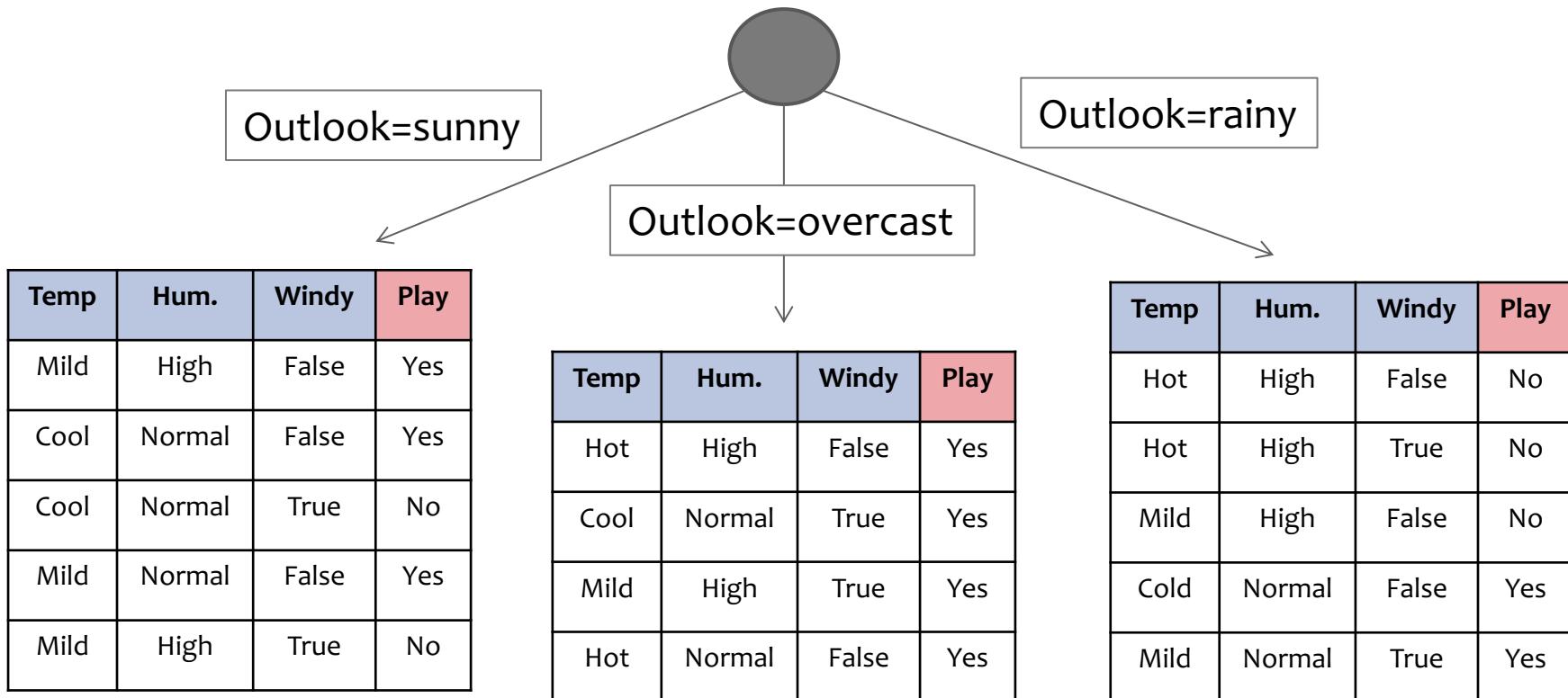
		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3

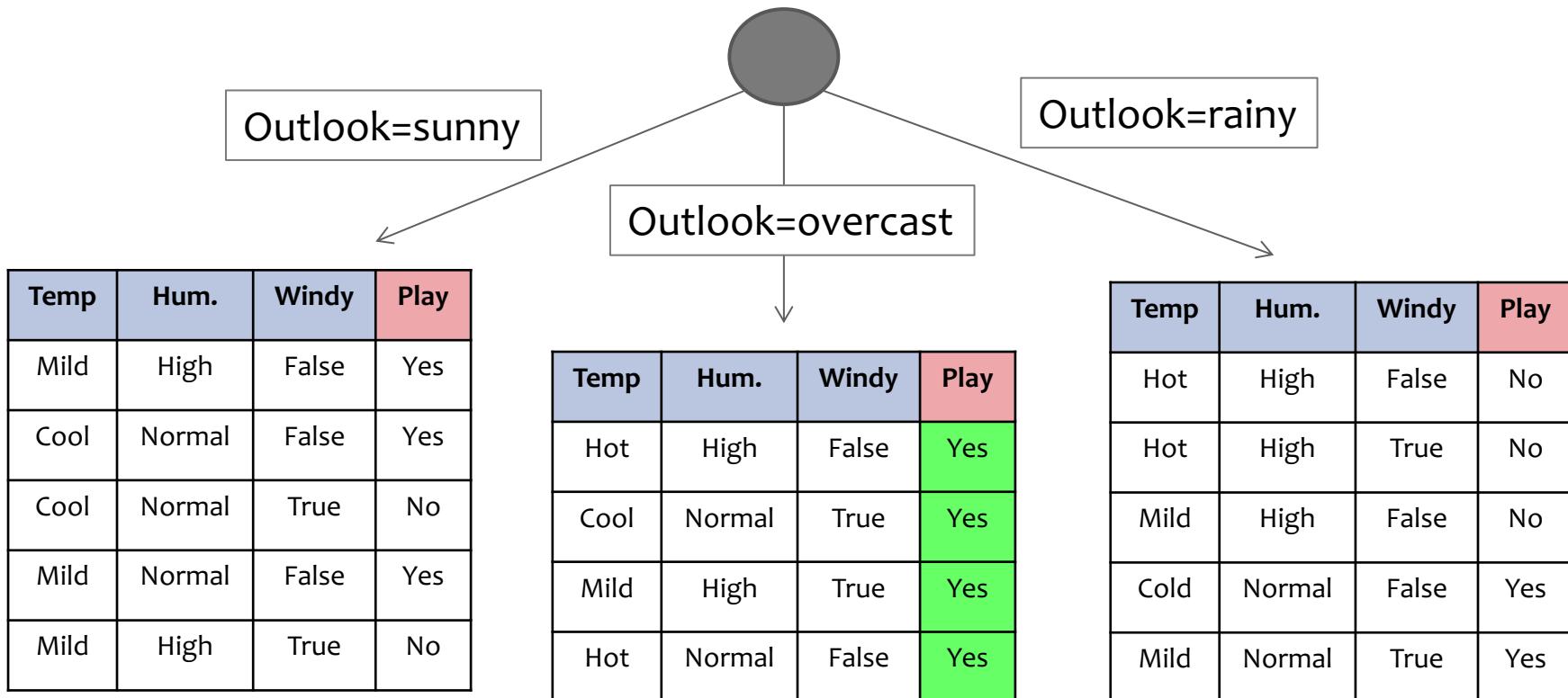
# Selección de atributos

Basado en el clasificador oneR



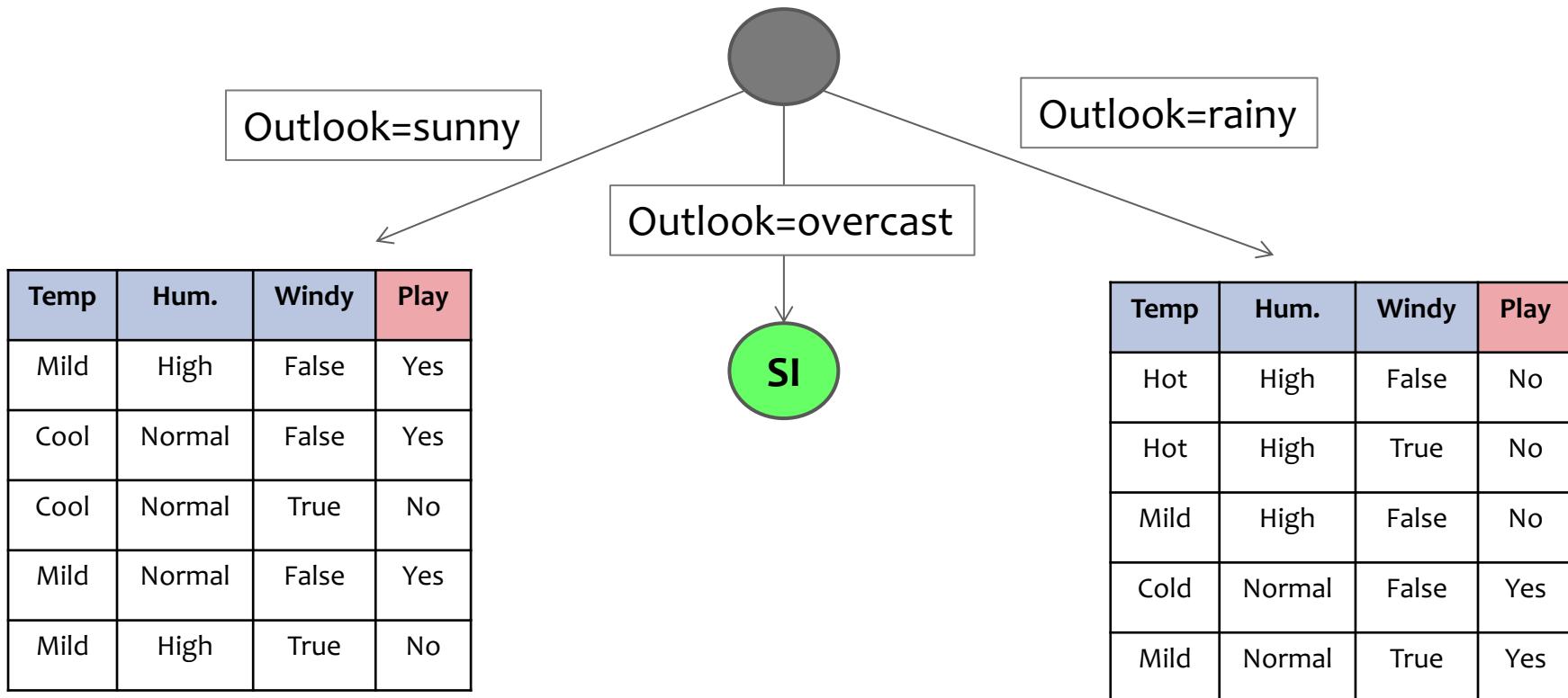
# Selección de atributos

Basado en el clasificador oneR



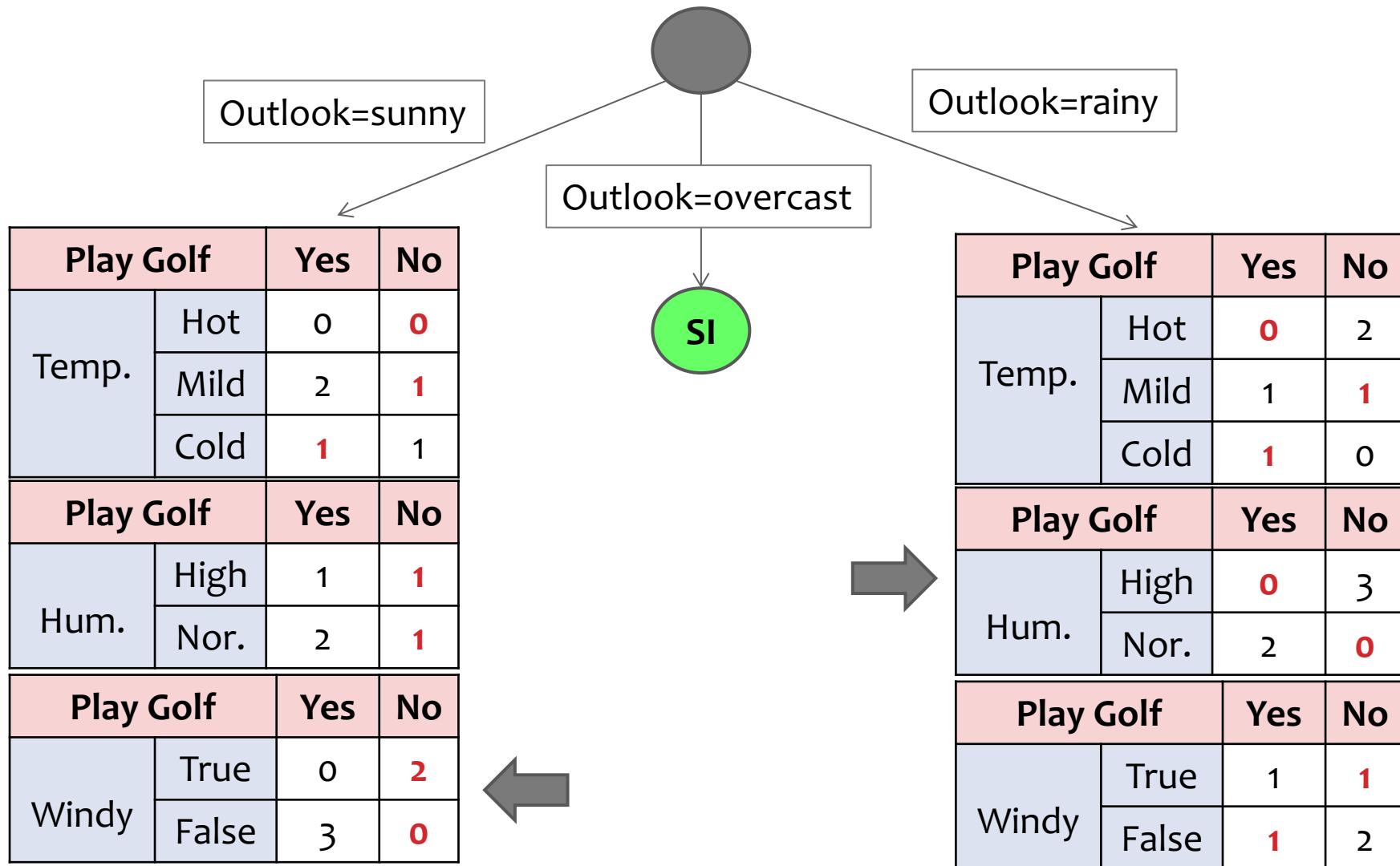
# Selección de atributos

Basado en el clasificador oneR



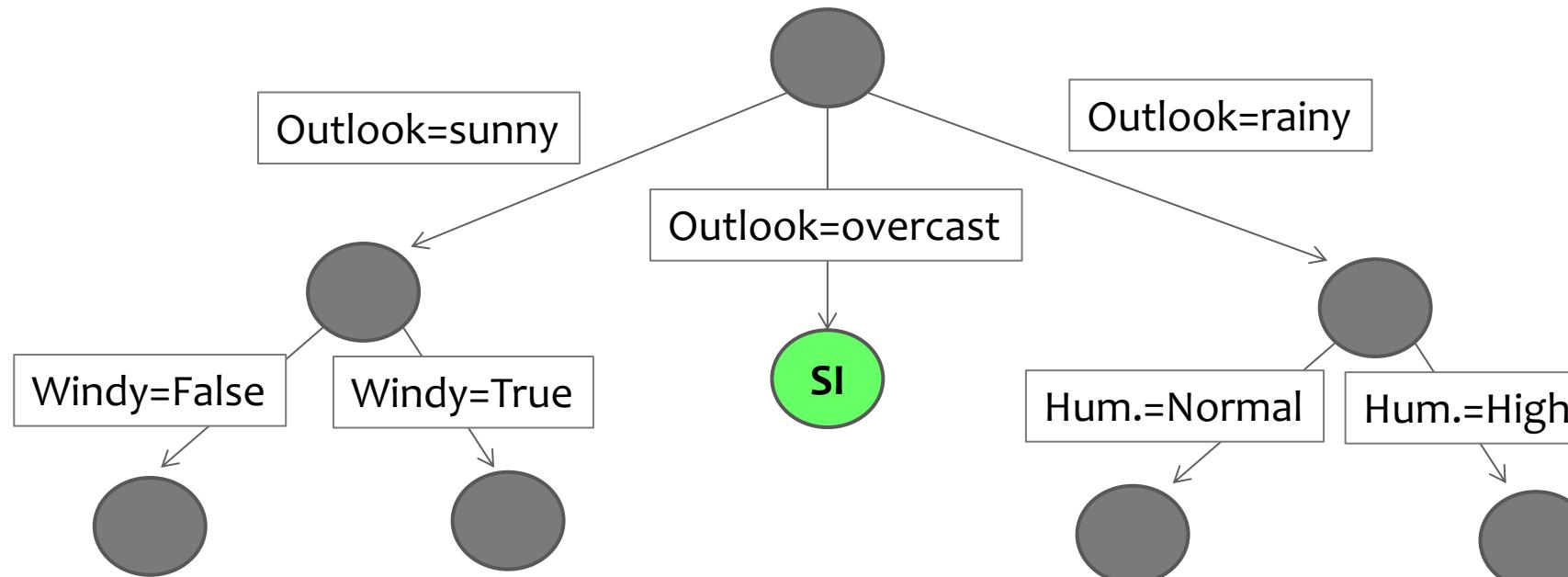
# Selección de atributos

Basado en el clasificador oneR



# Selección de atributos

Basado en el clasificador oneR



Temp	Hum.	Play
Mild	High	Yes
Cool	Normal	Yes
Mild	Normal	Yes

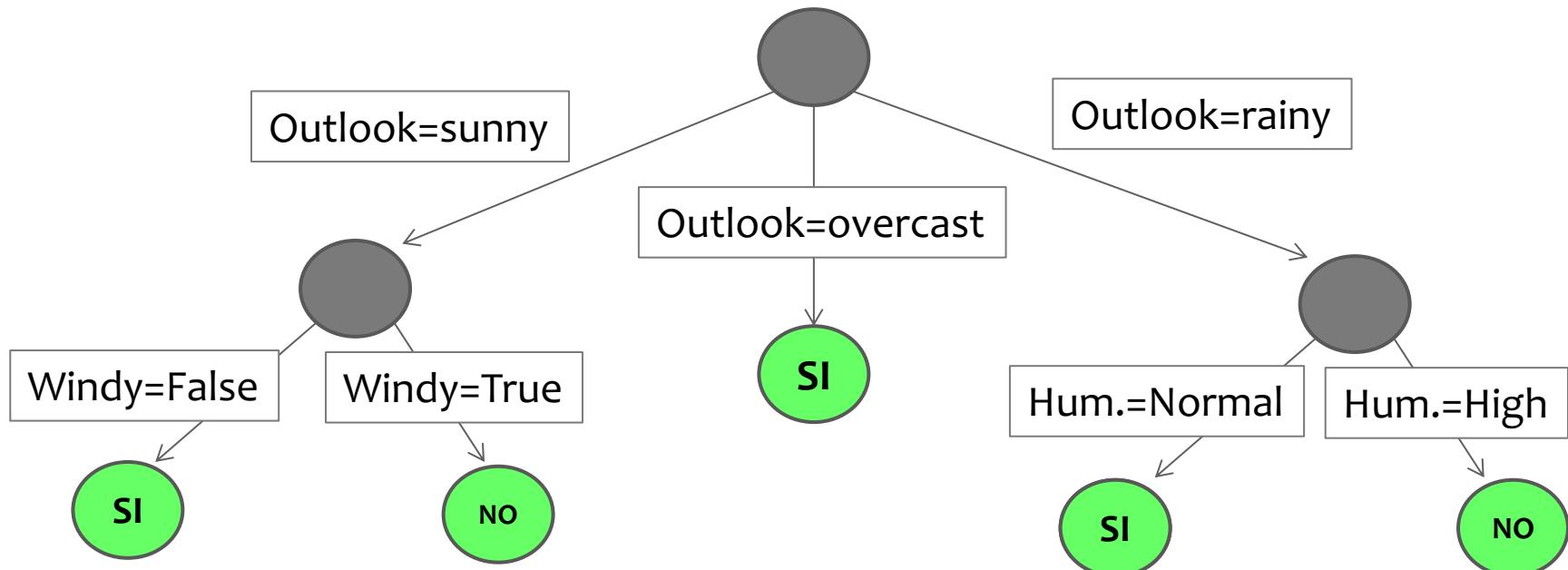
Temp	Hum.	Play
Cool	Normal	No
Mild	High	No

Temp	Windy	Play
Cold	False	Yes
Mild	True	Yes

Temp	Windy	Play
Hot	False	No
Hot	True	No
Mild	False	No

# Selección de atributos

Basado en el clasificador oneR



# Selección de atributos

Ganancia de información-Entropía (Information Gain)

Selecciona el atributo con la mayor ganancia de información

Para un atributo  $X$  que toma valores  $x_1, x_2, \dots, x_m$  se definen los coeficientes

$$f_i = \frac{\text{num. instancias con } X = x_i}{\text{Total de instancias}} \quad \text{con } i = 1, \dots, m$$

Entonces, la cantidad de información (bits) necesaria para clasificar una instancia en alguna de las clases es

$$\text{Info}(X) = - \sum_{i=1}^m f_i \log_2 f_i$$

# Selección de atributos

Tabla para facilitar el calculo de la entropía (dos clases)

$E(+,-)$	0-	1-	2-	3-	4-	5-
0+		0,000	0,000	0,000	0,000	0,000
1+	0,000	1,000	0,918	0,811	0,722	0,650
2+	0,000	0,918	1,000	0,971	0,918	0,863
3+	0,000	0,811	0,971	1,000	0,985	0,954
4+	0,000	0,722	0,918	0,985	1,000	0,991
5+	0,000	0,650	0,863	0,954	0,991	1,000
6+	0,000	0,592	0,811	0,918	0,971	0,994
7+	0,000	0,544	0,764	0,881	0,946	0,980
8+	0,000	0,503	0,722	0,845	0,918	0,961
9+	0,000	0,469	0,684	0,811	0,890	0,940

# Selección de atributos

Ganancia de información-Entropía (Information Gain)

Selecciona el atributo con la mayor ganancia de información

Sea ahora un atributo  $A$  que toma valores  $a_1, a_2, \dots, a_n$ , entonces la información esperada para clasificar una vez que hemos particionado el árbol con los valores del atributo es

$$\text{Info}_A(X) = \sum_{i=1}^n \frac{\text{num. instancias con } A = a_i}{\text{Total de instancias}} \text{Info}(X; A = a_i)$$

Entonces, la ganancia de información es

$$\text{Gain}(A) = \text{Info}(X) - \text{Info}_A(X)$$

# Selección de atributos

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Play	No	5
	Yes	9

$$\text{Info(Play)} = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$$

# Selección de atributos

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Outlook	Sunny	5 (3,2)
	Rainy	5 (2,3)
	Overcast	4 (4,0)

Temp.	Hot	4 (2,2)
	Mild	6 (4,2)
	Cool	4 (3,1)

Humidity	High	7 (3,3)
	Normal	7 (6,1)

Windy	True	6 (3,3)
	False	8 (6,2)

Play	No	5
	Yes	9

# Selección de atributos

Outlook	Sunny	5 (3,2)
	Rainy	5 (2,3)
	Overcast	4 (4,0)

$$\begin{aligned}\text{Info}_{\text{Outlook}}(\text{Play}) &= (5/14)\text{Info}(\text{Play}; \text{Outlook}=\text{Sunny}) \\ &\quad + (5/14)\text{Info}(\text{Play}; \text{Outlook}=\text{Rainy}) \\ &\quad + (4/14)\text{Info}(\text{Play}; \text{Outlook}=\text{Overcast}) \\ &= (5/14) \cdot 0.971 + (5/14) \cdot 0.971 + (4/14) \cdot 0 \\ &= 0.693 \text{ bits}\end{aligned}$$

$$\text{Gain}(\text{Outlook}) = \text{Info}(\text{Play}) - \text{Info}_{\text{Outlook}}(\text{Play}) = 0.247 \text{ bits}$$

# Selección de atributos

$$\text{Gain(Outlook)} = \text{Info(Play)} - \text{Info}_{\text{Outlook}}(\text{Play}) = 0.247 \text{ bits}$$

$$\text{Gain(Temp.)} = \text{Info(Play)} - \text{Info}_{\text{Temp.}}(\text{Play}) = 0.029 \text{ bits}$$

$$\text{Gain(Humidity)} = \text{Info(Play)} - \text{Info}_{\text{Humidity}}(\text{Play}) = 0.151 \text{ bits}$$

$$\text{Gain(Windy)} = \text{Info(Play)} - \text{Info}_{\text{Windy}}(\text{Play}) = 0.048 \text{ bits}$$

# Selección de atributos

$$\text{Gain(Outlook)} = \text{Info(Play)} - \text{Info}_{\text{Outlook}}(\text{Play}) = 0.247 \text{ bits}$$

$$\text{Gain(Temp.)} = \text{Info(Play)} - \text{Info}_{\text{Temp.}}(\text{Play}) = 0.029 \text{ bits}$$

$$\text{Gain(Humidity)} = \text{Info(Play)} - \text{Info}_{\text{Humidity}}(\text{Play}) = 0.151 \text{ bits}$$

$$\text{Gain(Windy)} = \text{Info(Play)} - \text{Info}_{\text{Windy}}(\text{Play}) = 0.048 \text{ bits}$$

Ramificamos usando el atributo *Outlook*

# Selección de atributos

## Gain Ratio

La ganancia de información tiende a favorecer atributos que toman muchos valores diferentes.

Para mejorarlo, dividimos por un factor

$$\text{Info}(A) = - \sum_{i=0}^n f_i \log_2 f_i$$

con

$$f_i = \frac{\text{num. instancias con } A = a_i}{\text{Total de instancias}}$$

Entonces

$$\text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{Info}(A)}$$

# Selección de atributos

## Índice de Gini

- ❑ El índice de Gini es una medida para determinar el grado en el que una población comparte un recurso
- ❑ El índice de Gini básicamente nos indica la equidad en la distribución de un recurso
- ❑ Los valores del índice de Gini van de 0 a 1, siendo 0 la mayor equidad en la distribución y 1 representa el mayor grado de desigualdad posible

# Selección de atributos

## Índice de Gini

Nosotros lo vamos a utilizar para medir la impureza y utilizamos la partición que más reduce esa impureza

- Un atributo  $X$  que toma valores  $x_1, x_2, \dots, x_m$  tiene índice de Gini

$$\text{Gini}(X) = 1 - \sum_{i=1}^m p_i^2$$

donde

$$p_i = \frac{\text{num. instancias con } X = x_i}{\text{Total instancias}}$$

# Selección de atributos

## Índice de Gini

Nosotros lo vamos a utilizar para medir la impureza y utilizamos la partición que más reduce esa impureza

- Sea ahora  $A$  un atributo que toma valores  $a_1, a_2, \dots, a_n$  el índice de Gini de los datos separados por el atributo es

$$\text{Gini}_{\text{Split}}(X; A) = \sum_{i=1}^n f_i \text{Gini}(X; A = a_i)$$

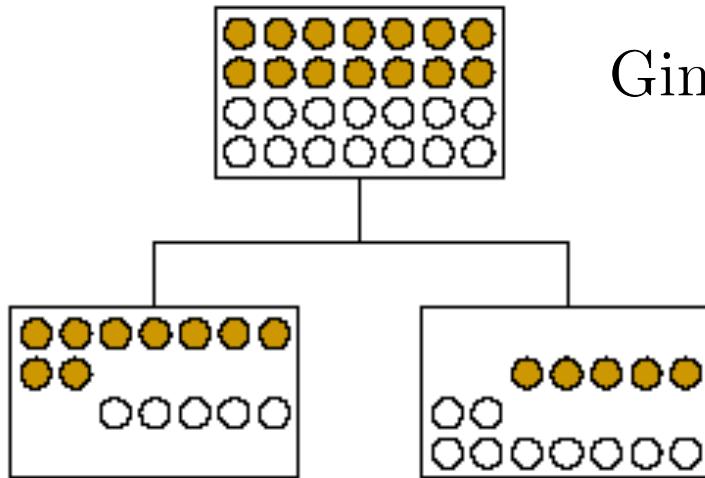
donde

$$f_i = \frac{\text{num. instancias con } A = a_i}{\text{Total instancias}}$$

Elegimos el atributo que minimice el índice de los datos separados

# Selección de atributos

Índice de Gini para la partición



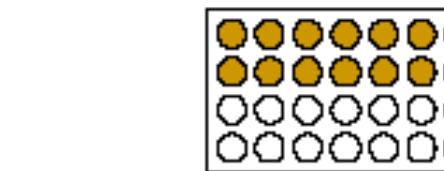
$$\text{Gini} = 1 - 0.5^2 - 0.5^2 = 0.5$$

$$\text{Gini}_{\text{Split}} = \frac{14}{28} \left( 1 - \frac{9^2}{14} - \frac{5^2}{14} \right) + \frac{14}{28} \left( 1 - \frac{5^2}{14} - \frac{9^2}{14} \right) = 0.459$$

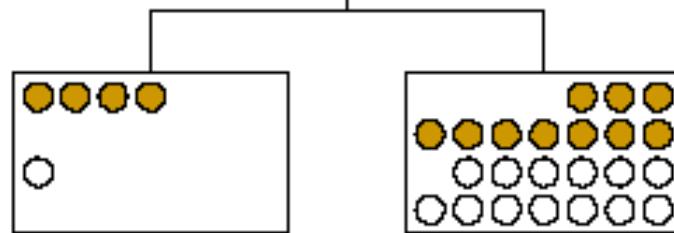
$$\Delta_{\text{Gini}} = \text{Gini} - \text{Gini}_{\text{Split}} = 0.5 - 0.459 = 0.041$$

# Selección de atributos

Índice de Gini para la partición



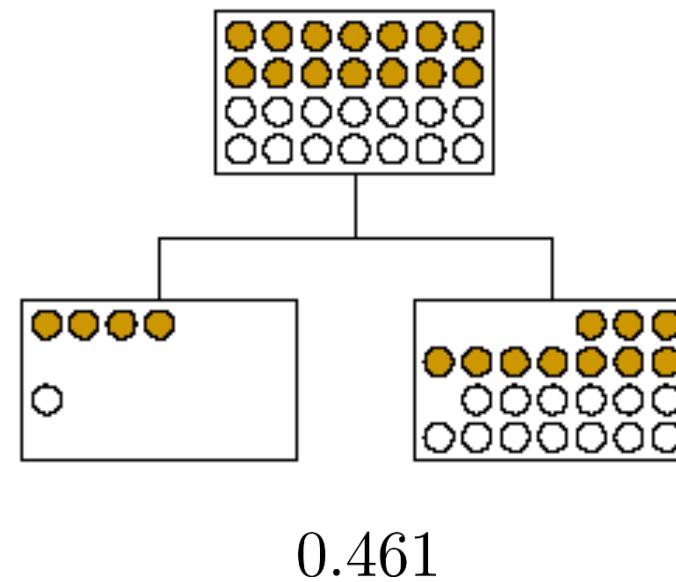
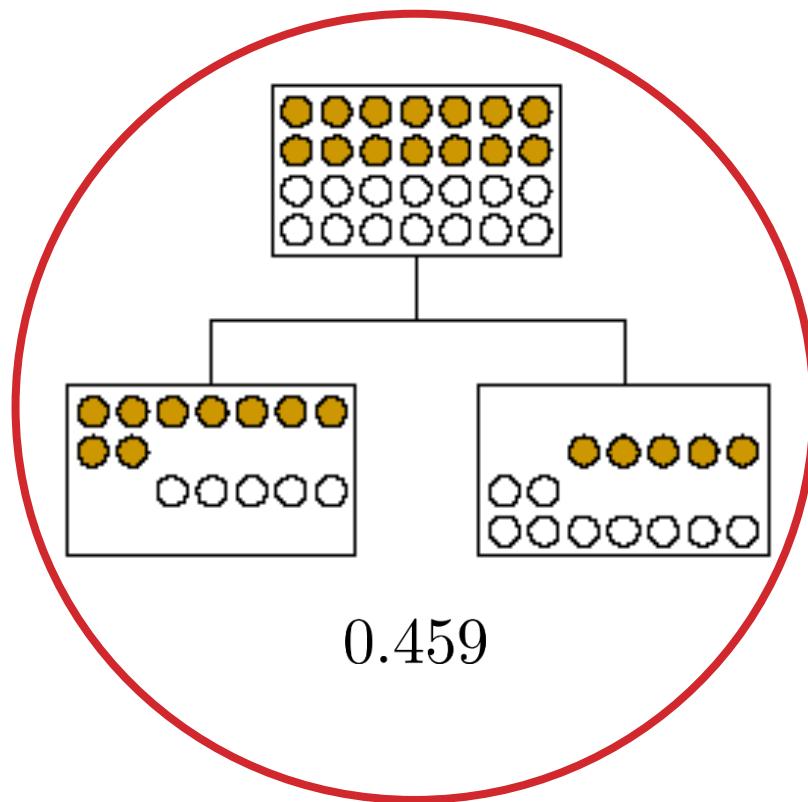
$$\text{Gini} = 1 - 0.5^2 - 0.5^2 = 0.5$$



$$\text{Gini}_{\text{Split}} = \frac{5}{28} \left( 1 - \frac{4^2}{5} - \frac{1^2}{5} \right) + \frac{23}{28} \left( 1 - \frac{10^2}{23} - \frac{13^2}{23} \right) = 0.461$$

$$\Delta_{\text{Gini}} = \text{Gini} - \text{Gini}_{\text{Split}} = 0.5 - 0.461 = 0.039$$

# Selección de atributos



# Selección de atributos

## Comparación de reglas de división

- Ganancia de información

Sesgado hacia atributos con muchos valores diferentes

- Criterio de proporción de ganancia

Tiende a preferir particiones poco balanceadas (con una partición mucho más grande que las otras)

- Índice de Gini

Funciona peor cuando hay muchas clases y tiende a favorecer particiones de tamaño y pureza similares

Ninguna regla de división es significativamente mejor que los demás

# Ventajas e inconvenientes

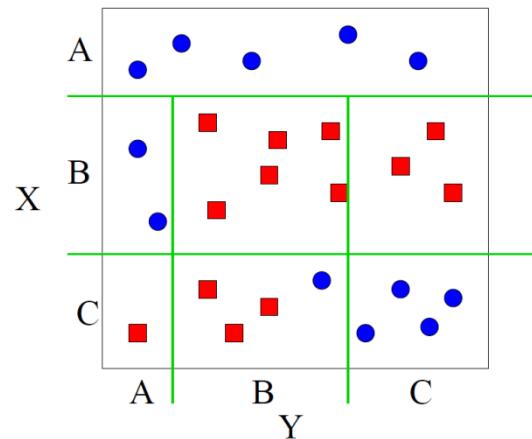
## □ Ventajas

- Los árboles de decisión son fáciles de utilizar
- Son muy eficientes
- Las reglas que generan son fáciles de interpretar
- Escalan mejor que otros tipos de técnicas
- Manejan bien el ruido
- Tienen una representación simbólica y se pueden desplegar de mayor a menor detalle
- Están implementados en multitud de sistemas, muchos de ellos gratuitos

# Ventajas e inconvenientes

## Inconvenientes

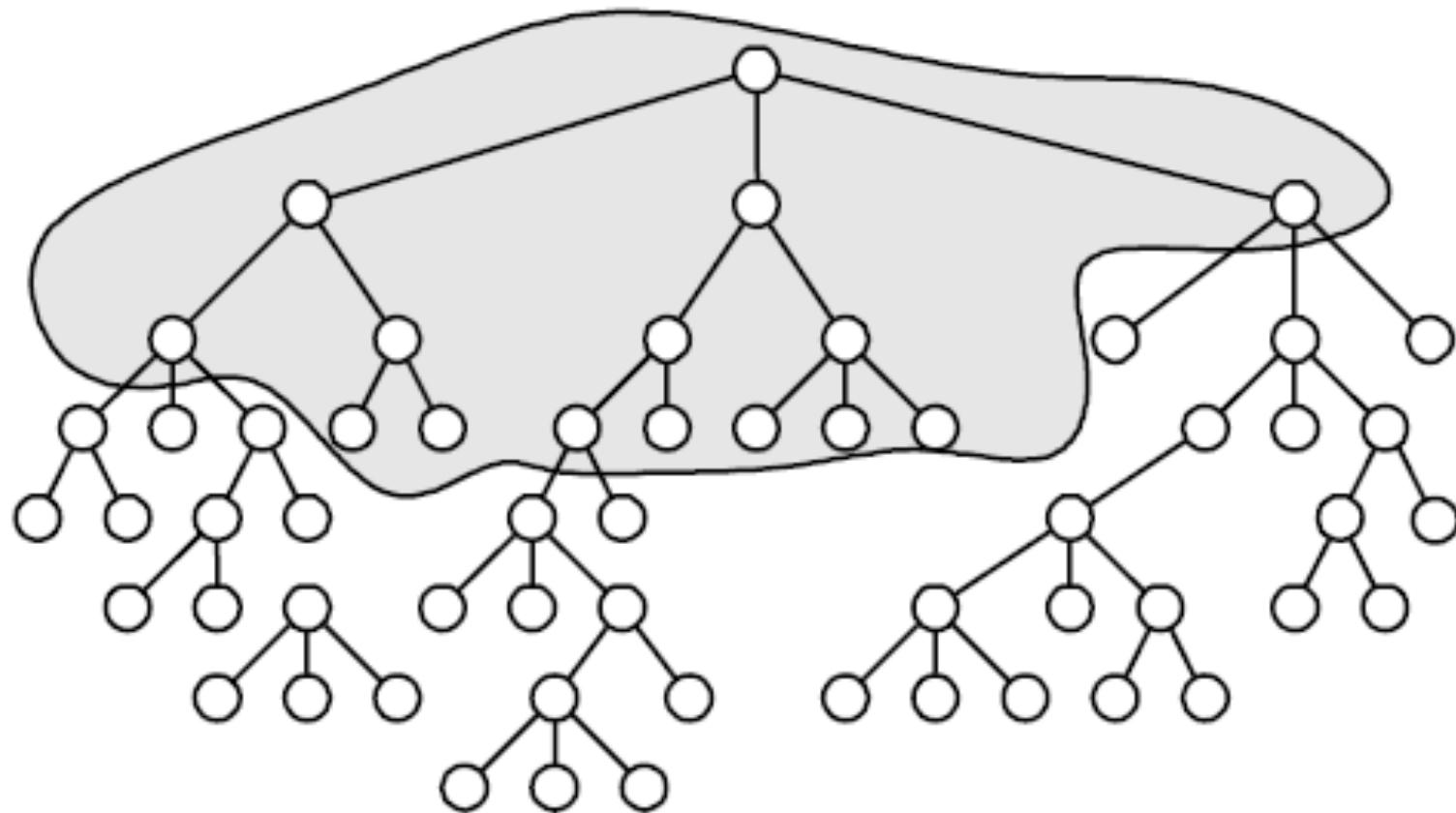
- No manejan de forma fácil los atributos continuos
- Tratan de dividir el dominio de los atributos en regiones rectangulares y no todos los problemas son de ese tipo



- Tienen dificultad para trabajar con valores perdidos. Pueden crear sobreaprendizaje
- No detectan correlaciones entre atributos

# Poda (pruning)

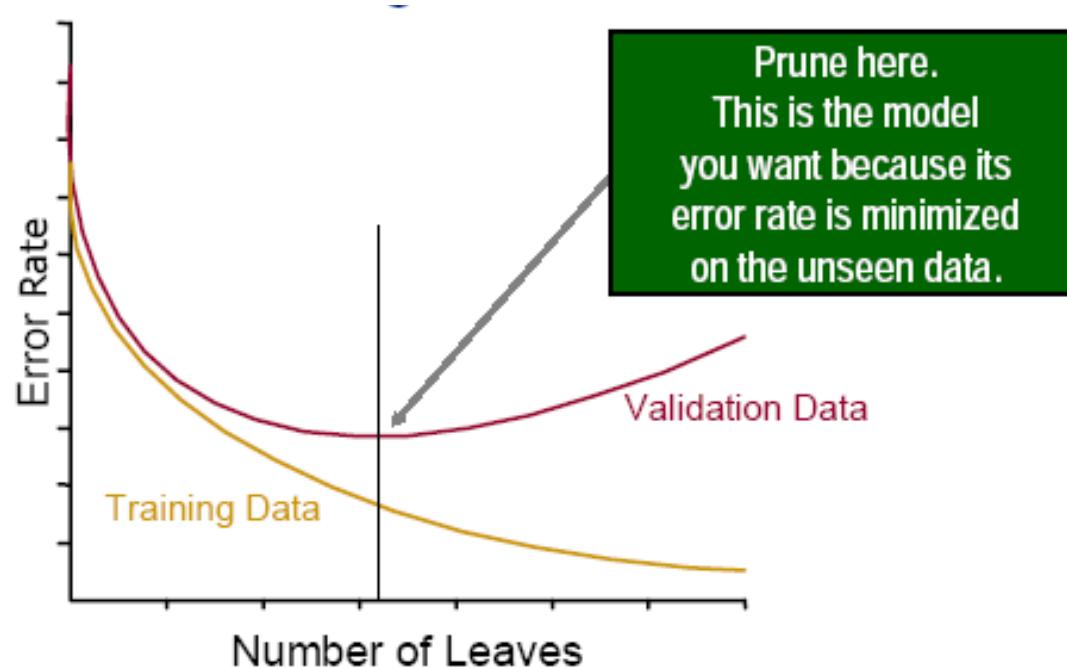
Incrementa la estabilidad del modelo al reducir su complejidad



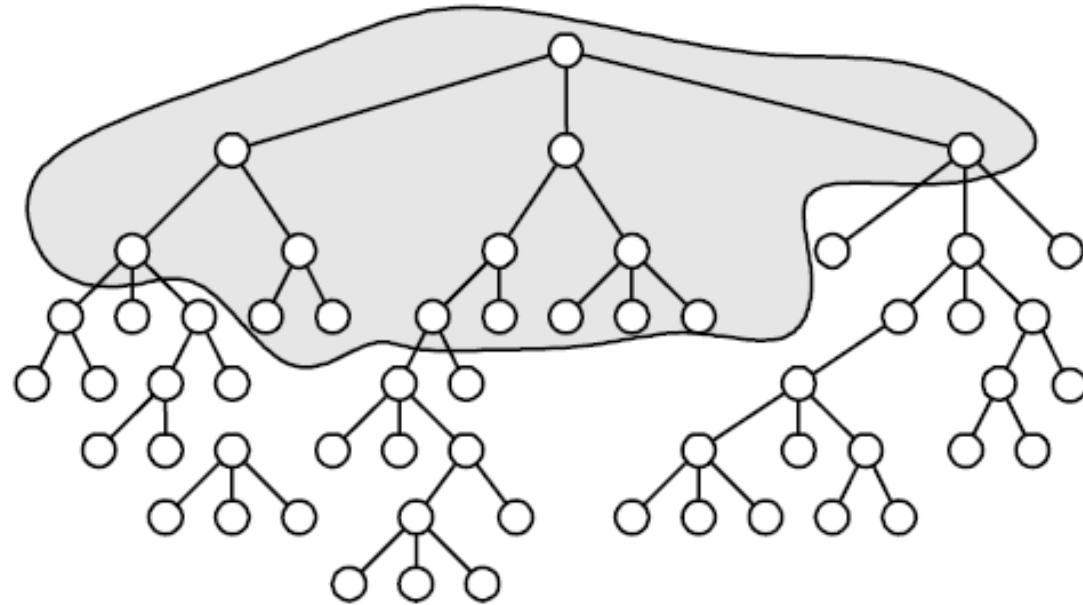
# Poda (pruning)

- ❑ Cada división del árbol reduce el error cometido con los datos de entrenamiento
- ❑ Pero a mayor complejidad, mayor sobreentrenamiento

Se trata en encontrar el lugar donde la complejidad del modelo empieza a aumentar el error en la clasificación



# Poda (pruning)



Los nodos por debajo de dicho límite/altura se podan porque son demasiado específicos (ad-hoc)

# Poda (pruning)

Dos posibilidades:

- ❑ **Pre-poda**, se añaden condiciones al criterio de parada en la ramificación
  - Más eficiente, menos exacto
- ❑ **Post-poda**, se poda el árbol una vez construido
  - Más ineficiente, más exacto

Al podar, los nodos hoja no son puros y se suelen etiquetar con la clase mayoritaria

# Poda (pruning)

## ❑ Pre-poda

- Si hay poco ejemplos (clase mayoritaria)
- Si no hay diferencia significativa entre las clases (test chi-cuadrado)
- ...

## ❑ Post-poda (desde los nodos hojas hasta la raíz)

- Si el error con el conjunto test es mayor dividiendo que sin dividir
- ...

# Algoritmo ID3

J.R. Quinlan, Induction of Decision Trees. Machine Learning 1 (1986), 81-106



- ID3 elige el atributo con máxima ganancia de información
- Intenta reducir el número de comparaciones
- Permite ruido en los datos de entrenamiento
- Por la ganancia de información, tiene tendencia a elegir atributos con muchos valores

# Algoritmo C4.5

J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Pub., 1993



- ❑ Evolución de ID3
- ❑ En C4.5 se utiliza como criterio de selección el ratio de ganancia de información
- ❑ Como consecuencia, tiende a preferir particiones poco balanceadas (una partición mucho más grande que las otras)
- ❑ Utiliza técnicas de poda

# Otros

- ❑ C4.8
- ❑ C5.0
- ❑ PUBLIC (Rastogi & Shim, VLDB'1998)  
integra la poda en el proceso de construcción del árbol
- ❑ RainForest (Gehrke et al., VLDB'1998)  
separa lo que determina la escalabilidad del algoritmo
- ❑ BOAT (Gehrke et al., PODS'1999)  
sólo necesita recorrer 2 veces el conjunto de datos
- ❑ ...

# Ejercicio evaluable

(3 personas, 30 minutos) **Técnicas de construcción de árboles de decisión.** Realiza una presentación explicando los algoritmos de construcción de árboles de decisión nombrados en la transparencia anterior

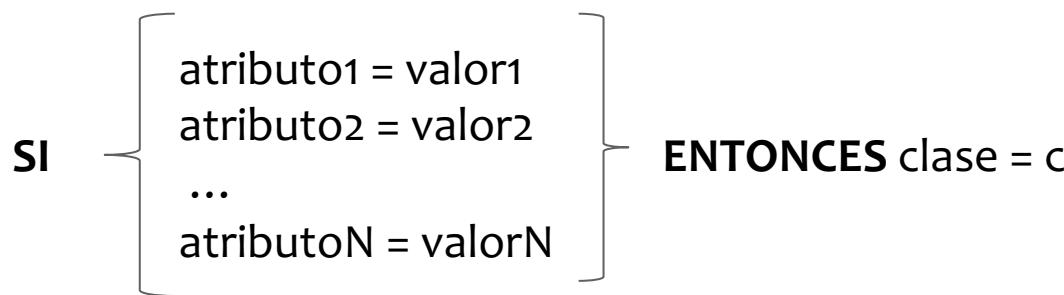
# Índice

- El problema de la clasificación
- Clasificación con árboles de decisión
- **Clasificación con reglas**
  - El concepto de regla de clasificación
  - Generación de reglas
- Clasificación estadística
- Otros clasificadores

# Concepto de regla

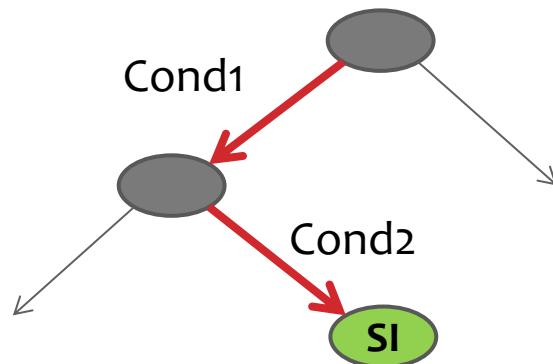
Una regla de clasificación es una regla IF-THEN formada por

- ❑ Un **antecedente**, que contiene un predicado que se evaluará como verdadero o falso con respecto a cada ejemplo de la base de ejemplos
- ❑ Un **consecuente**, que contiene una etiqueta de clase



# Generación de reglas (árbol de decisión)

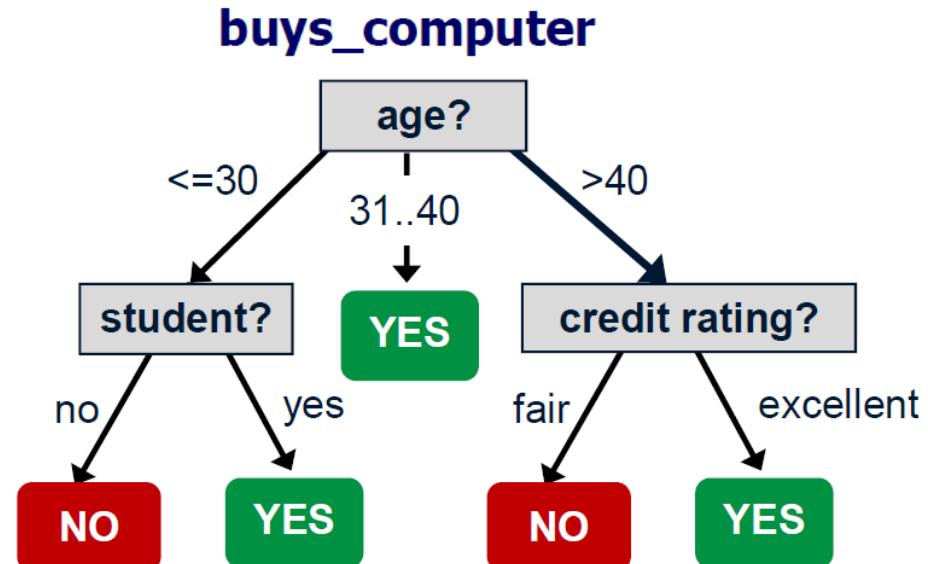
- Podemos extraer las reglas de un árbol de decisión y usarlas como un sistema basado en reglas
  - **Consecuencia:** Un sistema basado en reglas es un modelo más general que un árbol de decisión
  - **¿Por qué reglas?** Las reglas son más fáciles de interpretar que un árbol de decisión complejo
- El algoritmo para obtener el sistema basado en reglas: para cada camino de la raíz del árbol a un nodo hoja, genera una regla y la añade al conjunto final



**Si Cond1 Y Cond2 ENTONCES SI**

# Generación de reglas (árbol de decisión)

A partir de un árbol de decisión



**IF (age<=30) AND (student=no)  
THEN buys\_computer = NO**

**IF (age<=30) AND (student=yes)  
THEN buys\_computer = YES**

**IF (30<age<=40)  
THEN buys\_computer = YES**

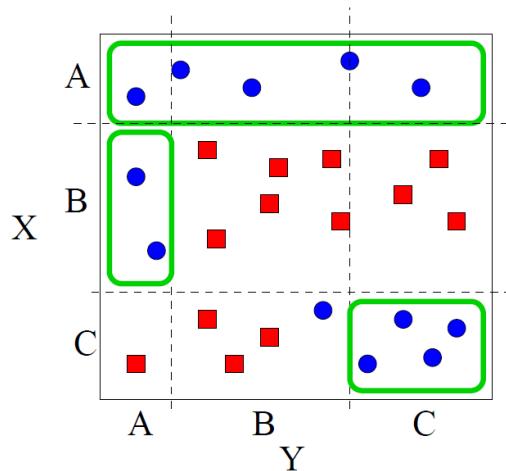
**IF (age>40) AND (credit\_rating=excellent)  
THEN buys\_computer = YES**

**IF (age>40) AND (credit\_rating=fair)  
THEN buys\_computer = NO**

# Generación de reglas (árbol de decisión)

Las reglas resultantes obtenidas de un árbol de decisión son **mutuamente excluyentes** y **exhaustivas** pero en un sistema de reglas no tiene que ser así

- En estos clasificadores lo que se intenta es buscar las reglas que cubran el mayor número de ejemplos positivos (o el menor de ejemplos negativos)
- No se realiza un particionamiento exhaustivo del espacio de soluciones



Si  $X=A$  entonces círculo  
Si  $X=B$  e  $Y=A$  entonces círculo  
Si  $X=C$  e  $Y=C$  entonces círculo  
En otro caso cuadrado

# Generación de reglas (árbol de decisión)

Por ejemplo, **simplificando las reglas** obtenidas del árbol:

- ❑ Dejan de ser mutuamente excluyentes
  - Varias reglas pueden ser válidas para un mismo ejemplo (hay que establecer un orden entre las reglas [**lista de decisión**] o realizar una votación)
- ❑ Dejan de ser exhaustivas
  - Puede que ninguna regla sea aplicable a un ejemplo concreto (hace falta incluir una clase por defecto)

# Generación de reglas (cobertura)

Otra forma es **generarlas directamente del conjunto de entrenamiento**

- Las reglas se aprenden de una en una
- Cada vez que se escoge una regla, se eliminan del conjunto de entrenamiento todos los casos cubiertos por la regla seleccionada
- El proceso se repite iterativamente hasta que se cumpla alguna condición de parada

# Generación de reglas (cobertura)

---

**Algorithm** Algoritmo de aprendizaje por cobertura

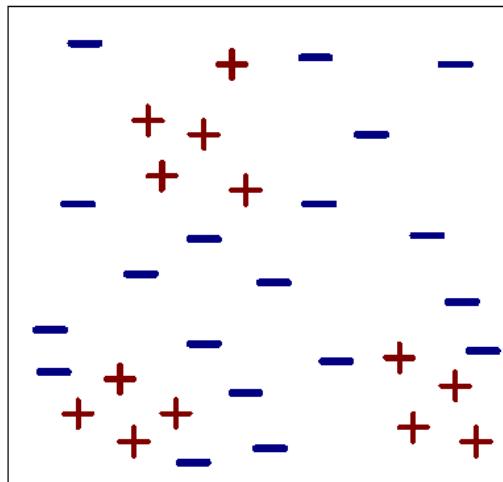
---

**Input:**  $P$  ejemplos positivos,  $N$  ejemplos negativos

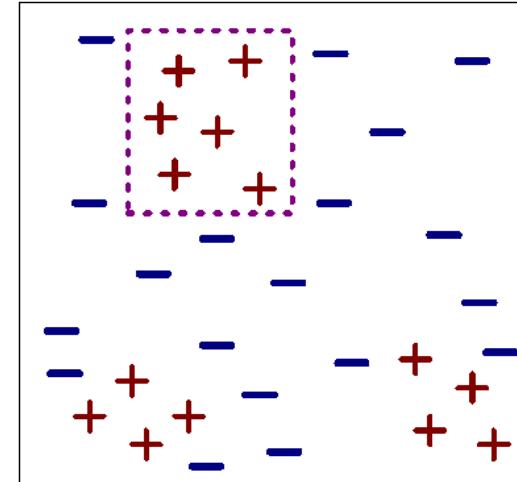
**Output:**  $R$  reglas de clasificación

- 1:  $R \leftarrow \emptyset$
  - 2: **while**  $P \neq \emptyset$  y ParadaReglas=False **do**
  - 3:     Regla  $\leftarrow \emptyset$
  - 4:      $N_{aux} \leftarrow N$
  - 5:     **while**  $N_{aux} \neq \emptyset$  y ParadaCondiciones=False **do**
  - 6:         Cond = SeleccionarCondicion
  - 7:         Regla = Regla  $\cup \{Cond\}$
  - 8:          $N_{aux} =$  Ejemplos negativos consistentes con Regla
  - 9:      $R \leftarrow R \cup \{Regla\}$
  - 10:     $P \leftarrow P - \{\text{Ejemplos cubiertos por Regla}\}$
  - 11:     $R \leftarrow R \cup \{\text{En otro caso, negativo}\}$
  - 12: **return**  $R$
-

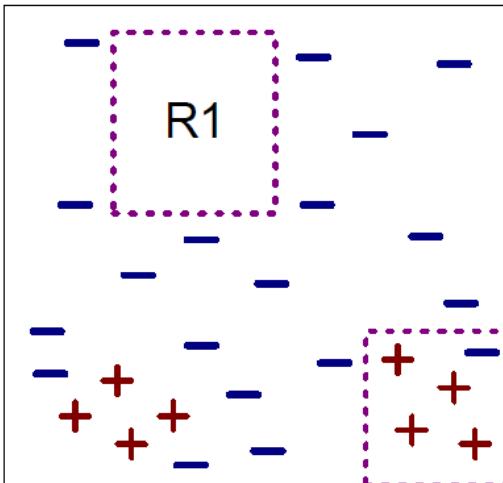
# Generación de reglas (cobertura)



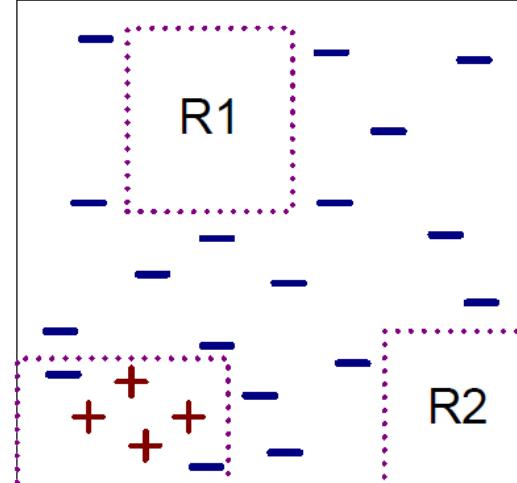
(i) Original Data



(ii) Step 1



(iii) Step 2



(iv) Step 3

# Generación de reglas (cobertura)

- Para seleccionar la mejor regla utilizaremos el tanto por ciento de acierto (para el SI)

Crédito	Ingresos	Propietario	Gastos-mensuales
NO	Bajos	No	Altos
NO	Altos	No	Altos
NO	Bajos	No	Bajos
NO	Medios	No	Bajos
NO	Altos	No	Bajos
NO	Medios	No	Altos
NO	Bajos	Si	Altos
NO	Medios	Si	Altos
SI	Medios	Si	Bajos
SI	Altos	Si	Bajos

# Generación de reglas (cobertura)

- Para seleccionar la mejor regla utilizaremos el tanto por ciento de acierto

Crédito	Ingresos	Propietario	Gastos-mensuales
NO	Bajos	No	Altos
NO	Altos	No	Altos
NO	Bajos	No	Bajos
NO	Medios	No	Bajos
NO	Altos	No	Bajos
NO	Medios	No	Altos
NO	Bajos	Si	Altos
NO	Medios	Si	Altos
SI	Medios	Si	Bajos
SI	Altos	Si	Bajos

antecedente	consecuente	s
Ingresos=altos	Si	1/3
Ingresos=medios	Si	1/4
Ingresos=bajos	Si	0/3
Propietario=si	Si	2/4
Propietario=no	Si	0/6
Gastos=altos	Si	0/5
Gastos=bajos	Si	2/5

# Generación de reglas (cobertura)

- Para seleccionar la mejor regla utilizaremos el tanto por ciento de acierto

Crédito	Ingresos	Propietario	Gastos-mensuales
NO	Bajos	No	Altos
NO	Altos	No	Altos
NO	Bajos	No	Bajos
NO	Medios	No	Bajos
NO	Altos	No	Bajos
NO	Medios	No	Altos
NO	Bajos	Si	Altos
NO	Medios	Si	Altos
SI	Medios	Si	Bajos
SI	Altos	Si	Bajos



antecedente	consecuente	s
Ingresos=altos	Si	1/3
Ingresos=medios	Si	1/4
Ingresos=bajos	Si	0/3
Propietario=si	Si	2/4
Propietario=no	Si	0/6
Gastos=altos	Si	0/5
Gastos=bajos	Si	2/5

Regla parcial: **Si Propietario = Si entonces Crédito = SI**

Existen casos negativos, por lo que debemos continuar

# Generación de reglas (cobertura)

Crédito	Ingresos	Propietario	Gastos-mensuales
NO	Bajos	No	Altos
NO	Altos	No	Altos
NO	Bajos	No	Bajos
NO	Medios	No	Bajos
NO	Altos	No	Bajos
NO	Medios	No	Altos
NO	Bajos	Si	Altos
NO	Medios	Si	Altos
SI	Medios	Si	Bajos
SI	Altos	Si	Bajos

antecedente	consecuente	s
propietario=si Y ingresos=altos	Si	1/1
propietario=si Y ingresos=medios	Si	1/2
propietario=si Y ingresos=bajos	Si	0/1
propietario=si Y gastos=altos	Si	0/2
propietario=si Y gastos=bajos	Si	2/2

# Generación de reglas (cobertura)

Crédito	Ingresos	Propietario	Gastos-mensuales
NO	Bajos	No	Altos
NO	Altos	No	Altos
NO	Bajos	No	Bajos
NO	Medios	No	Bajos
NO	Altos	No	Bajos
NO	Medios	No	Altos
NO	Bajos	Si	Altos
NO	Medios	Si	Altos
SI	Medios	Si	Bajos
SI	Altos	Si	Bajos

antecedente	consecuente	s
propietario=si Y ingresos=altos	Si	1/1
propietario=si Y ingresos=medios	Si	1/2
propietario=si Y ingresos=bajos	Si	0/1
propietario=si Y gastos=altos	Si	0/2
propietario=si Y gastos=bajos	Si	2/2



Regla parcial: **Si Propietario = SI y Gastos = bajos entonces Crédito = SI**

La regla es perfecta, por lo que la añadimos

# Generación de reglas (cobertura)

Crédito	Ingresos	Propietario	Gastos-mensuales
NO	Bajos	No	Altos
NO	Altos	No	Altos
NO	Bajos	No	Bajos
NO	Medios	No	Bajos
NO	Altos	No	Bajos
NO	Medios	No	Altos
NO	Bajos	Si	Altos
NO	Medios	Si	Altos
SI	Medios	Si	Bajos
SI	Altos	Si	Bajos

antecedente	consecuente	s
propietario=si Y ingresos=altos	Si	1/1
propietario=si Y ingresos=medios	Si	1/2
propietario=si Y ingresos=bajos	Si	0/1
propietario=si Y gastos=altos	Si	0/2
propietario=si Y gastos=bajos	Si	2/2

Regla parcial: **Si Propietario = SI y Gastos = bajos entonces Crédito = SI**

La regla es perfecta, por lo que la añadimos

No existen casos positivos fuera de esta regla

# Generación de reglas (cobertura)

Crédito	Ingresos	Propietario	Gastos-mensuales
NO	Bajos	No	Altos
NO	Altos	No	Altos
NO	Bajos	No	Bajos
NO	Medios	No	Bajos
NO	Altos	No	Bajos
NO	Medios	No	Altos
NO	Bajos	Si	Altos
NO	Medios	Si	Altos
SI	Medios	Si	Bajos
SI	Altos	Si	Bajos

Sistema de reglas obtenido

**Si Propietario = SI y Gastos = bajos entonces Crédito = SI  
En otro caso, Crédito = NO**

# Generación de reglas (cobertura)

- ❑ El conjunto óptimo de reglas puede no obtenerse, debido a que actúa de forma voraz incluyendo una condición en cada paso
  - Podíamos haber seleccionado Ingresos = altos
- ❑ Consigue una clasificación perfecta, pero sobreajusta a los datos
  - Regla 1 con  $s=1/1$
  - Regla 2 con  $s=19/20$
  - El algoritmo elegiría R1 cuando R2 es mucho más general
- ❑ Alternativa

$$F_1 = \frac{2}{\frac{1}{\text{Exhaustividad}} + \frac{1}{\text{Precision}}} = \frac{2 \text{ Exh. Prec.}}{\text{Exh.} + \text{Prec.}}$$

# Algoritmos

## Algoritmos de inducción de reglas

- FOIL (Quinlan, Machine Learning, 1990)
- CN2 (Clark & Boswell, EWSL'1991)
- RIPPER (Cohen, ICML'1995)
- PNrule (Joshi, Agarwal & Kumar, SIGMOD'2001)

Ejercicio evaluable. (3 personas, 30 minutos) Algoritmos de inducción de reglas. Realiza una presentación explicando los algoritmos de inducción de reglas nombrados

# Índice

- El problema de la clasificación
- Clasificación con árboles de decisión
- Clasificación con reglas
- **Clasificación con métodos bayesianos**
  - Teorema de Bayes
  - Clasificador Naïve Bayes
  - Redes Bayesianas
- Otros clasificadores

# Objetivos

- ❑ Entender el interés de utilizar técnicas Bayesianas en el problema de clasificación
- ❑ Conocer el clasificador Naïve Bayes, modo de aprendizaje y utilización
- ❑ Conocer el concepto de red Bayesiana y algunos clasificadores basados en redes Bayesianas
- ❑ Conocer clasificadores basados en redes neuronales, instancias y vecindad

# Incertidumbre en la clasificación

Incertidumbre debido a:

- Conocimiento incompleto del dominio
- Entendimiento/comprendión incorrecta del dominio
- Relaciones en el dominio de naturaleza no determinística (por ejemplo, enfermedades y síntomas)
- Términos involucrados muy vagos (grande, hermosa, dolor)
- Realización de algún tipo de abstracción
- Naturaleza aleatoria de los mecanismos que rigen el comportamiento del dominio (por ejemplo, modelado de usuarios)

# Incertidumbre en la clasificación

Los métodos considerados hasta ahora no tratan de forma directa **la incertidumbre** natural asociada al proceso de clasificación

- La clasificación es booleana, o pertenece a una clase o no
- Pero quizás es interesante dar una probabilidad

Parece razonable introducir técnicas relacionadas con **teoría de la probabilidad**, ya que es el lenguaje adecuado para cuantificar la incertidumbre

# Incertidumbre en la clasificación

- **Invertir en bolsa.** Clasificación de dos acciones A y B, para decidir si comprar o no:
  - Método sin tratar incertidumbre, A->SI y B->SI
  - Método tratando incertidumbre
    - A->SI con probabilidad 0.9
    - B->SI con probabilidad 0.6
- No parece razonable invertir a partes iguales

# Teorema de Bayes

Relaciona probabilidad condicional y marginal

Probabilidad  
a posteriori

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Probabilidad a priori

- $P(A|B)$  es la probabilidad de que ocurra  $A$  supuesto  $B$
- $P(B|A)$  es la probabilidad de que ocurra  $B$  supuesto  $A$
- $P(B)$  es la probabilidad de que ocurra  $B$
- $P(A)$  es la probabilidad de que ocurra  $A$

# Teorema de Bayes

**Ejercicio.** Se considera el siguiente proceso:

Tenemos dos urnas, una con 5 bolas rojas y 3 bolas blancas, y otra con 4 bolas rojas y 5 bolas blancas. Se tira una moneda, si sale cara, se toma una bola de la primera urna. Si sale cruz, se toma una bola de la segunda urna.

¿Cuál es la probabilidad de haber obtenido cara, sabiendo que hemos tomado una bola blanca?

# Hipótesis MAP

Supongamos un problema de clasificación:

- ❑ Para clasificar sobre un conjunto  $C$  de clases
- ❑ Los datos se presentan con atributos  $\{A_1, \dots, A_n\}$

En vista de lo anterior, clasificaremos un dato como la clase que maximice la probabilidad condicionada (hipótesis MAP, *maximum a posteriori*)

$$c_{\text{MAP}} = \arg \max_{c \in C} P(c | A_1 \cdots A_n)$$



Aplicamos teorema de Bayes

# Hipótesis MAP

Supongamos un problema de clasificación:

- ❑ Para clasificar sobre un conjunto  $C$  de clases
- ❑ Los datos se presentan con atributos  $\{A_1, \dots, A_n\}$

En vista de lo anterior, clasificaremos un dato como la clase que maximice la probabilidad condicionada (hipótesis MAP, *maximum a posteriori*)

$$c_{\text{MAP}} = \arg \max_{c \in C} \frac{P(A_1 \cdots A_n | c) P(c)}{P(A_1 \cdots A_n)}$$

# Hipótesis MAP

Supongamos un problema de clasificación:

- ❑ Para clasificar sobre un conjunto  $C$  de clases
- ❑ Los datos se presentan con atributos  $\{A_1, \dots, A_n\}$

En vista de lo anterior, clasificaremos un dato como la clase que maximice la probabilidad condicionada (hipótesis MAP, *maximum a posteriori*)

$$c_{\text{MAP}} = \arg \max_{c \in C} \frac{P(A_1 \cdots A_n | c) P(c)}{P(A_1 \cdots A_n)}$$

Denominador común a todos  
los valores de la clase

# Hipótesis MAP

Supongamos un problema de clasificación:

- ❑ Para clasificar sobre un conjunto  $C$  de clases
- ❑ Los datos se presentan con atributos  $\{A_1, \dots, A_n\}$

En vista de lo anterior, clasificaremos un dato como la clase que maximice la probabilidad condicionada (hipótesis MAP, *maximum a posteriori*)

$$c_{\text{MAP}} = \arg \max_{c \in C} P(A_1 \cdots A_n | c)P(c)$$

# Clasificador Naïve Bayes

Problema principal de MAP

Alto coste computacional para conocer  $P(A_1 \cdots A_n | c)$

Solución sencilla (clasificador Naïve Bayes)

Considerar que los atributos son independientes

- No es una suposición realista...
- ...pero funciona bastante bien

$$c_{\text{MAP}} = \arg \max_{c \in C} \prod_{i=1}^n P(A_i | c) P(c)$$

# Clasificador Naïve Bayes

¿Cómo calcular  $P(A_i|c)$ ?

## □ Atributos discretos

- Mediante el estimador de máxima verosimilitud

$$P(a_i|c) = \frac{\text{Casos con } A_i = a_i, C = c}{\text{Casos con } C = c}$$

- Mediante el estimador de Laplace (pocos datos)

$$P(a_i|c) = \frac{(\text{Casos con } A_i = a_i, C = c) + 1}{(\text{Casos con } C = c) + (\text{num. valores de } A_i)}$$

# Clasificador Naïve Bayes

¿Cómo calcular  $P(A_i|c)$ ?

## □ Atributos numéricos

- Las distribuciones se aproximan mediante una normal

$$\mathcal{N}_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

- La media y la varianza se calculan a partir del conjunto de datos

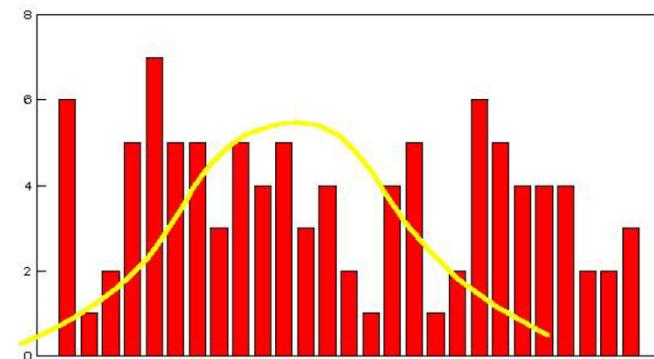
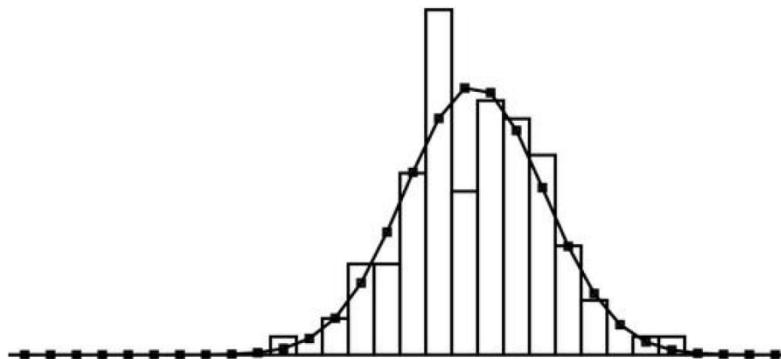
Hay tablas/calculadoras para saber esto!

# Clasificador Naïve Bayes

## □ Atributos numéricos

$$\mathcal{N}_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Evidentemente, en unos casos la aproximación realizada será mejor que en otros



# Clasificador Naïve Bayes

¿Cómo calcular  $P(c)$  ?

Esto es fácil, es un atributo nominal

$$P(c) = \frac{\text{Intancias con } C = c}{\text{Número total de instancias}}$$

# Clasificador Naïve Bayes

## Ejemplo

	A={a,b}	B, numérico	Clase={+,-}
1	a	5	+
2	a	2.2	-
3	a	1.8	-
4	b	4	+
5	b	2	+
6	a	3	-

Clasificar la instancia (b,3.5)

# Clasificador Naïve Bayes

	A={a,b}	B, numérico	Clase={+,-}
1	a	5	+
2	a	2.2	-
3	a	1.8	-
4	b	4	+
5	b	2	+
6	a	3	-

Clasificar la instancia (b,3.5)

$$P(b|+) = 0.67$$

# Clasificador Naïve Bayes

	A={a,b}	B, numérico	Clase={+,-}
1	a	5	+
2	a	2.2	-
3	a	1.8	-
4	b	4	+
5	b	2	+
6	a	3	-

Clasificar la instancia (b,3.5)

$$P(b|+) = 0.67 \quad P(b|-) = 0$$

# Clasificador Naïve Bayes

	A={a,b}	B, numérico	Clase={+,-}
1	a	5	+
2	a	2.2	-
3	a	1.8	-
4	b	4	+
5	b	2	+
6	a	3	-

Clasificar la instancia (b,3.5)

$$P(b|+) = 0.67 \quad P(b|-) = 0$$

$$\mathcal{N}^+ = \mathcal{N}_{3.67, 1.53} \quad P(B = 3.5|+) = 0.26$$

# Clasificador Naïve Bayes

	A={a,b}	B, numérico	Clase={+,-}
1	a	5	+
2	a	2.2	-
3	a	1.8	-
4	b	4	+
5	b	2	+
6	a	3	-

Clasificar la instancia (b,3.5)

$$P(b|+) = 0.67 \quad P(b|-) = 0$$

$$\mathcal{N}^+ = \mathcal{N}_{3.67, 1.53} \quad P(B = 3.5|+) = 0.26$$

$$\mathcal{N}^- = \mathcal{N}_{2.33, 0.61} \quad P(B = 3.5|-) = 0.1$$

# Clasificador Naïve Bayes

	A={a,b}	B, numérico	Clase={+,-}
1	a	5	+
2	a	2.2	-
3	a	1.8	-
4	b	4	+
5	b	2	+
6	a	3	-

Clasificar la instancia (b,3.5)

$$P(b|+) = 0.67 \quad P(b|-) = 0$$

$$\mathcal{N}^+ = \mathcal{N}_{3.67, 1.53} \quad P(B = 3.5|+) = 0.26$$

$$\mathcal{N}^- = \mathcal{N}_{2.33, 0.61} \quad P(B = 3.5|-) = 0.1$$

$$P(+) = P(-) = 0.5$$

# Clasificador Naïve Bayes

Clasificar la instancia (b,3.5)

$$P(b|+) = 0.67 \quad P(b|-) = 0$$

$$\mathcal{N}^+ = \mathcal{N}_{3.67, 1.53} \quad P(B = 3.5|+) = 0.26$$

$$\mathcal{N}^- = \mathcal{N}_{2.33, 0.61} \quad P(B = 3.5|-) = 0.1$$

$$P(+) = P(-) = 0.5$$

$$P(+|b, 3.5) \rightarrow P(b|+)P(3.5|+)P(+) = 0.67 \cdot 0.26 \cdot 0.5 = 0.087$$

# Clasificador Naïve Bayes

Clasificar la instancia (b,3.5)

$$P(b|+) = 0.67 \quad P(b|-) = 0$$

$$\mathcal{N}^+ = \mathcal{N}_{3.67, 1.53} \quad P(B = 3.5|+) = 0.26$$

$$\mathcal{N}^- = \mathcal{N}_{2.33, 0.61} \quad P(B = 3.5|-) = 0.1$$

$$P(+) = P(-) = 0.5$$

$$P(+|b, 3.5) \rightarrow P(b|+)P(3.5|+)P(+) = 0.67 \cdot 0.26 \cdot 0.5 = 0.087$$

$$P(-|b, 3.5) \rightarrow P(b|-)P(3.5|-)P(-) = 0 \cdot 0.1 \cdot 0.5 = 0$$

Clasificamos a +

# Clasificador Naïve Bayes

Clasificar la instancia (b,3.5)

$$P(b|+) = 0.67 \quad P(b|-) = 0$$

$$\mathcal{N}^+ = \mathcal{N}_{3.67, 1.53} \quad P(B = 3.5|+) = 0.26$$

$$\mathcal{N}^- = \mathcal{N}_{2.33, 0.61} \quad P(B = 3.5|-) = 0.1$$

$$P(+) = P(-) = 0.5$$

$$P(+|b, 3.5) \rightarrow P(b|+)P(3.5|+)P(+) = 0.67 \cdot 0.26 \cdot 0.5 = 0.087$$

$$P(-|b, 3.5) \rightarrow P(b|-)P(3.5|-)P(-) = 0 \cdot 0.1 \cdot 0.5 = 0$$

Clasificamos a +

Esto seguramente se  
debe a los pocos  
datos existentes

# Clasificador Naïve Bayes

	A={a,b}	B, numérico	Clase={+,-}
1	a	5	+
2	a	2.2	-
3	a	1.8	-
4	b	4	+
5	b	2	+
6	a	3	-

Clasificar la instancia (b,3.5)

$$P(b|+) = 3/5 = 0.6 \quad P(b|-) = 1/5 = 0.2$$

Estimador  
de Laplace

$$\mathcal{N}^+ = \mathcal{N}_{3.67, 1.53} \quad P(B = 3.5|+) = 0.26$$

$$\mathcal{N}^- = \mathcal{N}_{2.33, 0.61} \quad P(B = 3.5|-) = 0.1$$

$$P(+) = P(-) = 0.5$$

# Clasificador Naïve Bayes

Clasificar la instancia (b,3.5)

$$P(b|+) = 0.6 \quad P(b|-) = 0.2$$

$$\mathcal{N}^+ = \mathcal{N}_{3.67, 1.53} \quad P(B = 3.5|+) = 0.26$$

$$\mathcal{N}^- = \mathcal{N}_{2.33, 0.61} \quad P(B = 3.5|-) = 0.1$$

$$P(+) = P(-) = 0.5$$

$$P(+|b, 3.5) \rightarrow P(b|+)P(3.5|+)P(+) = 0.6 \cdot 0.26 \cdot 0.5 = 0.078$$

$$P(-|b, 3.5) \rightarrow P(b|-)P(3.5|-)P(-) = 0.2 \cdot 0.1 \cdot 0.5 = 0.01$$

Clasificamos a +

# Clasificador Naïve Bayes

## □ Ventajas

- Es fácil de implementar
- Obtiene buenos resultados en gran parte de los casos

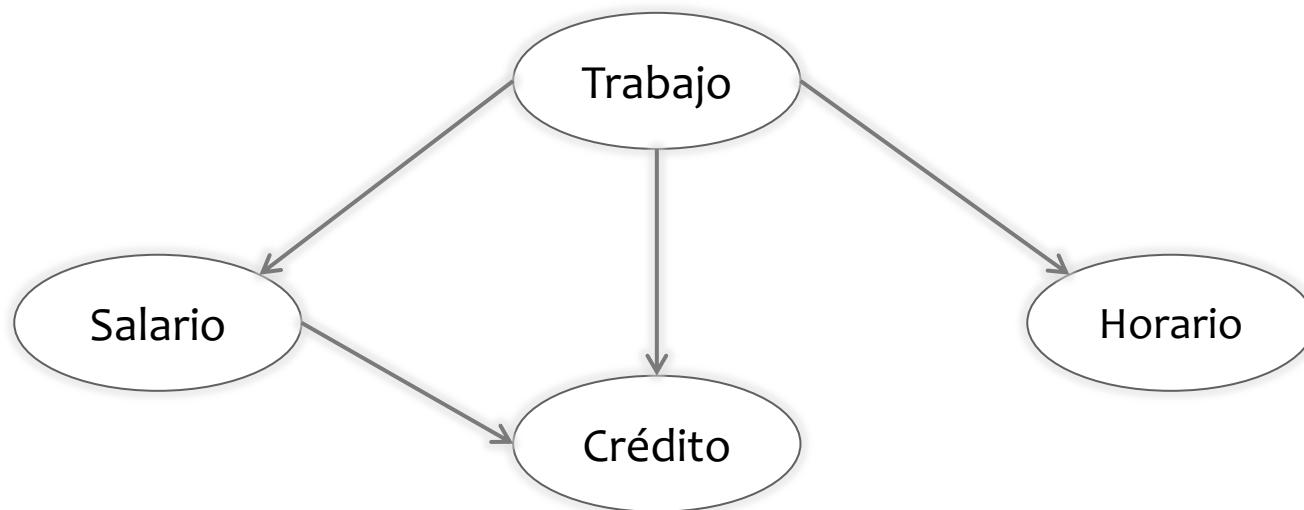
## □ Desventajas

- Asumir que las variables tienen independencia condicional respecto a la clase lleva a una falta de precisión
- En la práctica, existen dependencias entre las variables
  - Perfil: edad, historia familiar, etc.
  - Síntomas: fiebre, tos, etc.
  - Enfermedad: cáncer de pulmón, diabetes, etc.
- Con un clasificador Naïve-Bayes no se pueden modelar estas dependencias
- Solución: Redes de creencia bayesianas, que combinan razonamiento bayesiano con relaciones causales entre los atributos

# Redes de Bayes

¿Qué pasa si no consideramos las variables independientes?

- ❑ Podemos asumir ciertas relaciones entre variables
- ❑ La representación gráfica de esas dependencias es lo que se conoce como una **red de Bayes**



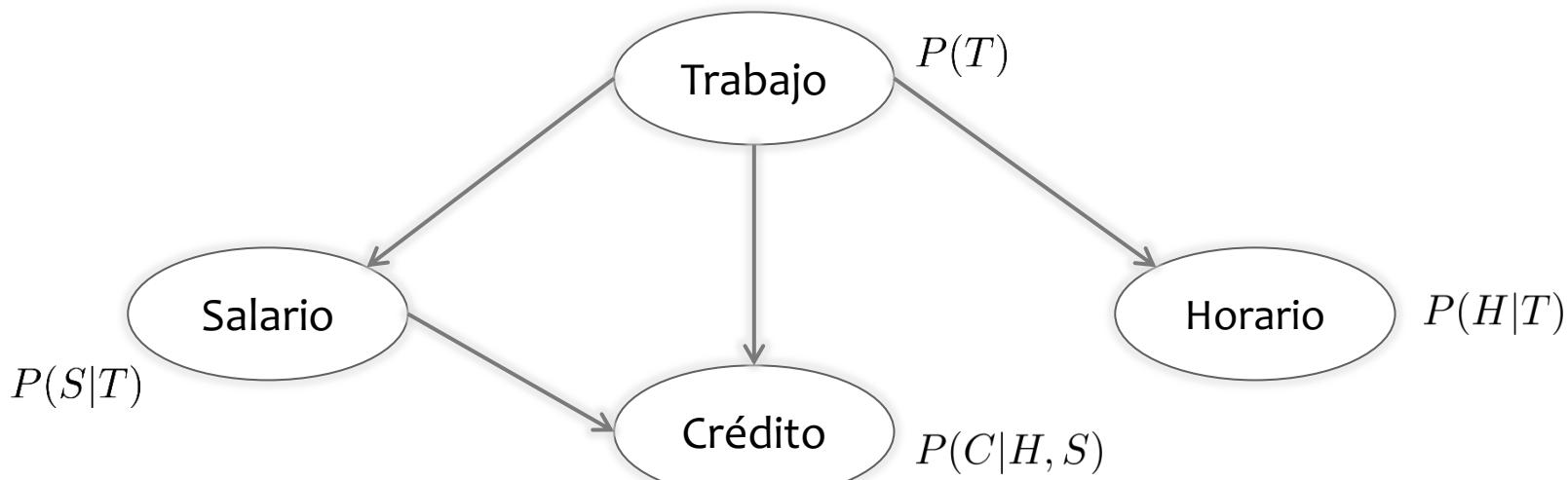
# Redes de Bayes

Formalmente, una red de Bayes es un **grafo dirigido**

- ❑ Cada nodo corresponde a una variable
- ❑ Una arista entre dos nodos significa que una variable es dependiente de la otra

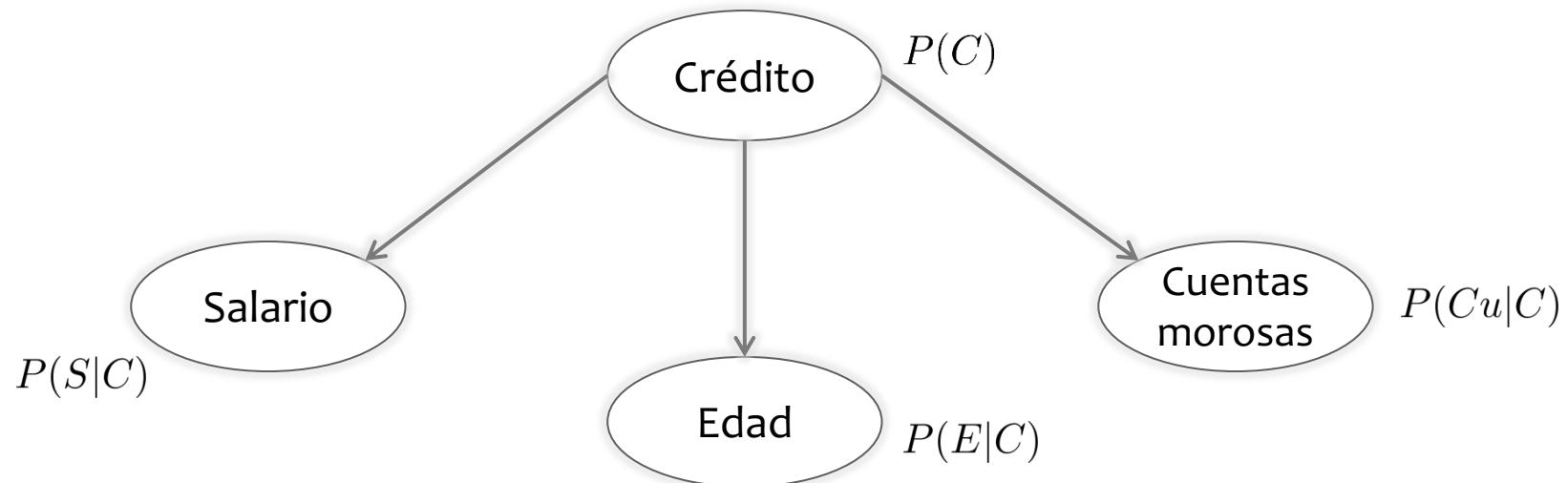
junto con un **conjunto de distribuciones de probabilidad**

- ❑ Para cada nodo, una distribución de probabilidad condicionada a las variables de los nodos padre



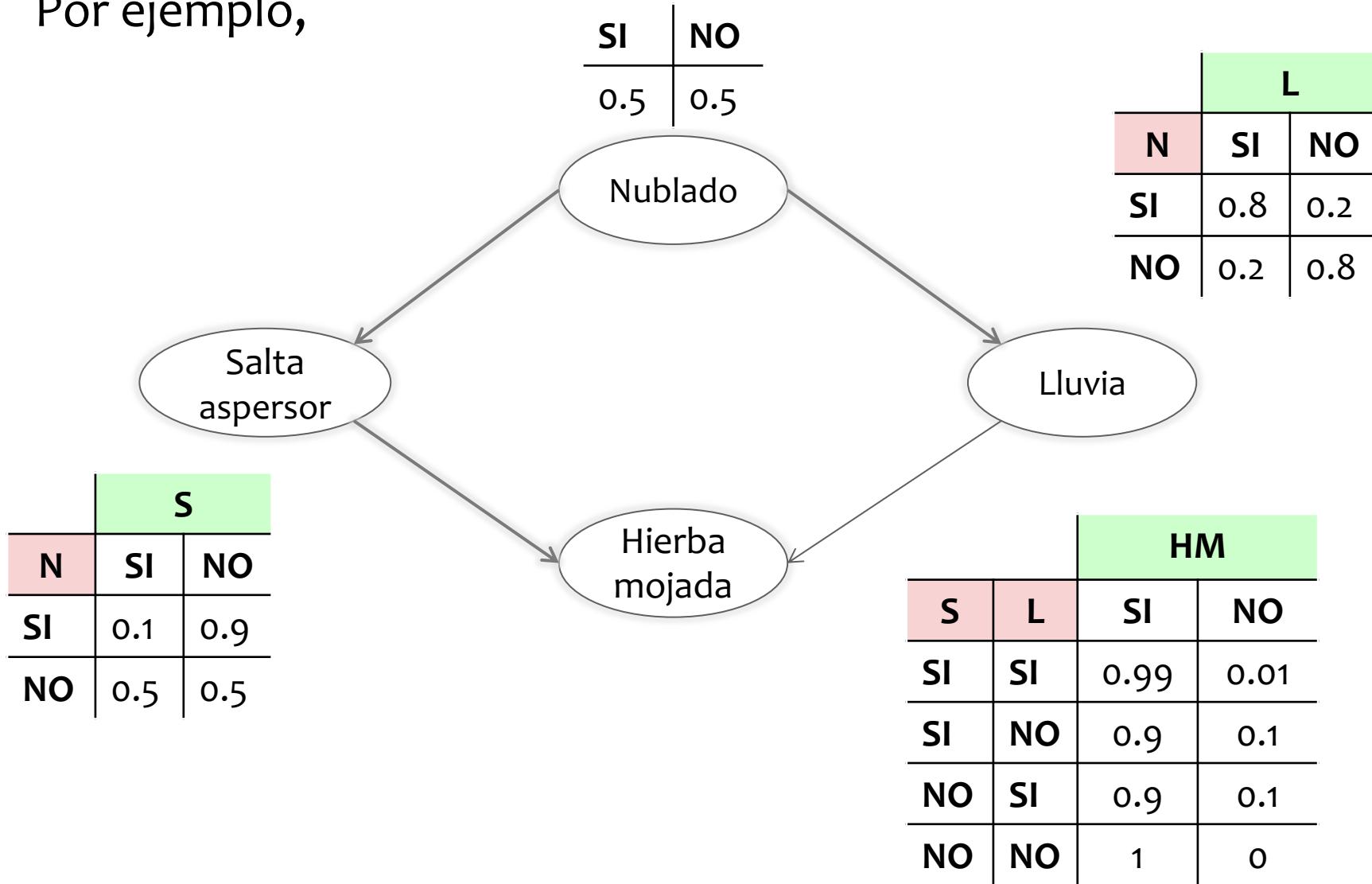
# Redes de Bayes

Por ejemplo, la red más sencilla es la obtenida al considerar el clasificador Naïve-Bayes



# Redes de Bayes

Por ejemplo,



# Redes de Bayes

Por ejemplo,

N	
SI	NO
0.5	0.5

S		
N	SI	NO
SI	0.1	0.9
NO	0.5	0.5

L		
N	SI	NO
SI	0.8	0.2
NO	0.2	0.8

		HM	
S	L	SI	NO
SI	SI	0.99	0.01
SI	NO	0.9	0.1
NO	SI	0.9	0.1
NO	NO	1	0

Clasificar ( $S=SI, L=SI, HM=NO$ )

$$\begin{aligned} P(N|S, L, \neg HM) &\approx P(N)P(S|N)P(L|N)P(\neg HM|S, L) \\ &= 0,5 \cdot 0,1 \cdot 0,8 \cdot 0,01 = 0,0004 \end{aligned}$$

# Redes de Bayes

Por ejemplo,

N	
SI	NO
0.5	0.5

S		
N	SI	NO
SI	0.1	0.9
NO	0.5	0.5

L		
N	SI	NO
SI	0.8	0.2
NO	0.2	0.8

		HM	
S	L	SI	NO
SI	SI	0.99	0.01
SI	NO	0.9	0.1
NO	SI	0.9	0.1
NO	NO	1	0

Clasificar ( $S=SI, L=SI, HM=NO$ )

$$\begin{aligned}P(N|S, L, \neg HM) &\approx P(N)P(S|N)P(L|N)P(\neg HM|S, L) \\&= 0,5 \cdot 0,1 \cdot 0,8 \cdot 0,01 = 0,0004\end{aligned}$$

$$\begin{aligned}P(\neg N|S, L, \neg HM) &\approx P(\neg N)P(S|\neg N)P(L|\neg N)P(\neg HM|S, L) \\&= 0,5 \cdot 0,5 \cdot 0,2 \cdot 0,01 = 0,0005\end{aligned}$$

# Redes de Bayes

Por ejemplo,

N	
SI	NO
0.5	0.5

S		
N	SI	NO
SI	0.1	0.9
NO	0.5	0.5

L		
N	SI	NO
SI	0.8	0.2
NO	0.2	0.8

		HM	
S	L	SI	NO
SI	SI	0.99	0.01
SI	NO	0.9	0.1
NO	SI	0.9	0.1
NO	NO	1	0

Clasificar ( $S=SI, L=SI, HM=NO$ )

$$\begin{aligned}P(N|S, L, \neg HM) &\approx P(N)P(S|N)P(L|N)P(\neg HM|S, L) \\&= 0,5 \cdot 0,1 \cdot 0,8 \cdot 0,01 = 0,0004\end{aligned}$$

$$\begin{aligned}P(\neg N|S, L, \neg HM) &\approx P(\neg N)P(S|\neg N)P(L|\neg N)P(\neg HM|S, L) \\&= 0,5 \cdot 0,5 \cdot 0,2 \cdot 0,01 = 0,0005\end{aligned}$$

Clasificamos a NO

# Construcción de la red

Pasos para construir la red:

- Determinar la **topología de la red** (el grafo orientado)
- Determinar las **distribuciones de probabilidad**

# Algoritmo TAN

El algoritmo más sencillo para aprender la topología de la red es el algoritmo TAN (**Tree Augmented Naïve-Bayes**)

Está basado en el concepto de información mutua

$$I_p(X, Y|C) = \sum_{x,y,c} P(x, y, c) \log \frac{P(x, y, c)}{P(x|c)P(y|c)}$$

Es la información que proporciona la variable Y de la X, supuesto que C es conocida

# Algoritmo TAN

---

## Algorithm 1 Algoritmo TAN

---

**Input:**  $E$  conjunto de instancias sobre variables  $A_1, A_2, \dots, A_n$  y una clase  $C$

**Output:**  $R$  red de Bayes modelizando  $E$

- 1:  $R \leftarrow$  un grafo completo con  $n$  nodos
  - 2: A cada arista  $(i, j)$  añadir el peso  $I_p(A_i, A_j | C)$
  - 3:  $R \leftarrow$  árbol generador maximal de  $R$
  - 4: Seleccionar un nodo y dirigir las flechas a partir de él
  - 5: Añadir un nodo extra  $C$  a  $R$  y flechas desde  $C$  al resto
  - 6: **return**  $R$
- 

**Eficiencia:**  $O(N \cdot n^2)$

# Algoritmo TAN

---

## Algorithm 1 Algoritmo TAN

---

**Input:**  $E$  conjunto de instancias sobre variables  $A_1, A_2, \dots, A_n$  y una clase  $C$

**Output:**  $R$  red de Bayes modelizando  $E$

- 1:  $R \leftarrow$  un grafo completo con  $n$  nodos
- 2: A cada arista  $(i, j)$  añadir el peso  $I_p(A_i, A_j | C)$
- 3:  $R \leftarrow$  árbol generador maximal de  $R$
- 4: Seleccionar un nodo y dirigir las flechas a partir de él
- 5: Añadir un nodo extra  $C$  a  $R$  y flechas desde  $C$  al resto
- 6: **return**  $R$

Eficiencia:  $O(N \cdot n^2)$

Por ejemplo, algoritmo de Kruskal

# Distribuciones de probabilidad

Para calcular las distribuciones de probabilidad podemos seguir el mismo proceso que con el clasificador Naïve-Bayes

## □ Atributos discretos

- Mediante el estimador de máxima verosimilitud

$$P(a_i|a_1 \dots a_m) = \frac{\text{Casos con } A_i = a_i, A_1 = a_1, \dots, A_m = a_m}{\text{Casos con } A_1 = a_1, \dots, A_m = a_m}$$

- Mediante el estimador de Laplace

$$P(a_i|a_1 \dots a_m) = \frac{(\text{Casos con } A_i = a_i, A_1 = a_1, \dots, A_m = a_m) + 1}{(\text{Casos con } A_1 = a_1, \dots, A_m = a_m) + (\text{num. valores de } A_i)}$$

# Distribuciones de probabilidad

Para calcular las distribuciones de probabilidad podemos seguir el mismo proceso que con el clasificador Naïve-Bayes

## □ Atributos continuos

- Aproximamos mediante una normal

$$\mathcal{N}_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

- La media y varianza la calculamos de los datos

# Índice

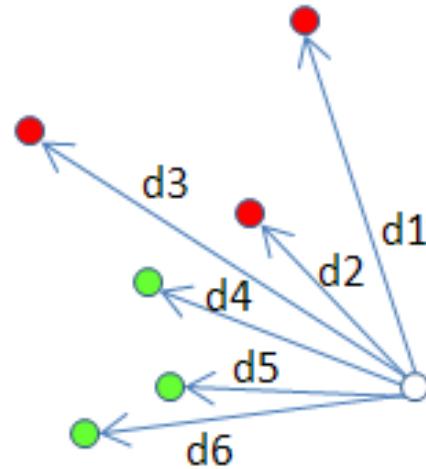
- El problema de la clasificación
- Clasificación con árboles de decisión
- Clasificación con reglas
- Clasificación con métodos bayesianos
- Otros clasificadores
  - Clasificadores basados en instancias y vecindad
  - Redes neuronales

# Clasificadores basados en instancias

- ❑ Están basados en el aprendizaje por analogía
  - Se trabaja cuando llega un nuevo caso a clasificar (paradigma perezoso). Se buscan los casos más parecidos y la clasificación se construye en función de la clase a la que dichos casos pertenecen
  - No se construye ningún modelo, el modelo es el conjunto de entrenamiento
- ❑ Los algoritmos más conocidos están basados en la regla del vecino más próximo

# Nearest neighbour (1-NN)

Entre las instancias del conjunto de entrenamiento se busca la instancia más cercana al caso a clasificar



# Nearest neighbour (1-NN)

---

**Algorithm** Algoritmo 1-NN

---

**Input:**  $\{(E_i, c_i)\}_{i=1,\dots,m}$  ejemplos clasificados,  $A$  sin clasificar,  $d$  distancia

**Output:**  $c$  (clasificación de  $A$ )

```
1:  $q \leftarrow d(E_1, A)$ 
2:  $c \leftarrow c_1$ 
3: for  $2 \leq i \leq m$  do
4:   if  $d(E_i, A) < q$  then
5:      $q \leftarrow d(E_i, A)$ 
6:      $c \leftarrow c_i$ 
7: return  $c$ 
```

---



Hay que decidir una distancia

# Nearest neighbour (1-NN)

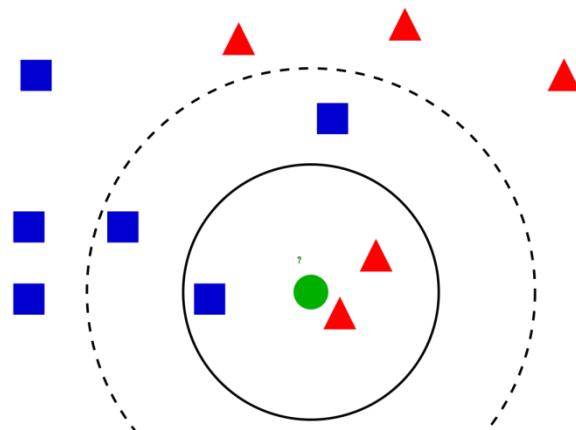
## Distancias usuales

- **Euclidea**     $d_e(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$
- **Manhattan**     $d_M(x, y) = \sum_i |x_i - y_i|$
- **Minkowski**     $d^k(x, y) = \left( \sum_i |x_i - y_i|^k \right)^{\frac{1}{k}}$
- **Hamming (nominales)**     $d_H(x, y) = \#\{i \text{ tales que } x_i \neq y_i\}$

# k-nearest neighbor (k-NN)

Extensión de 1-NN:

- ❑ Considera los k casos más cercanos a la instancia a clasificar
- ❑ Selecciona la clase entre las clases de dichos k casos
  - Voto por la mayoría (clase mayoritaria)
  - Voto con pesos en función de la distancia



# k-nearest neighbor (k-NN)

## Ventajas

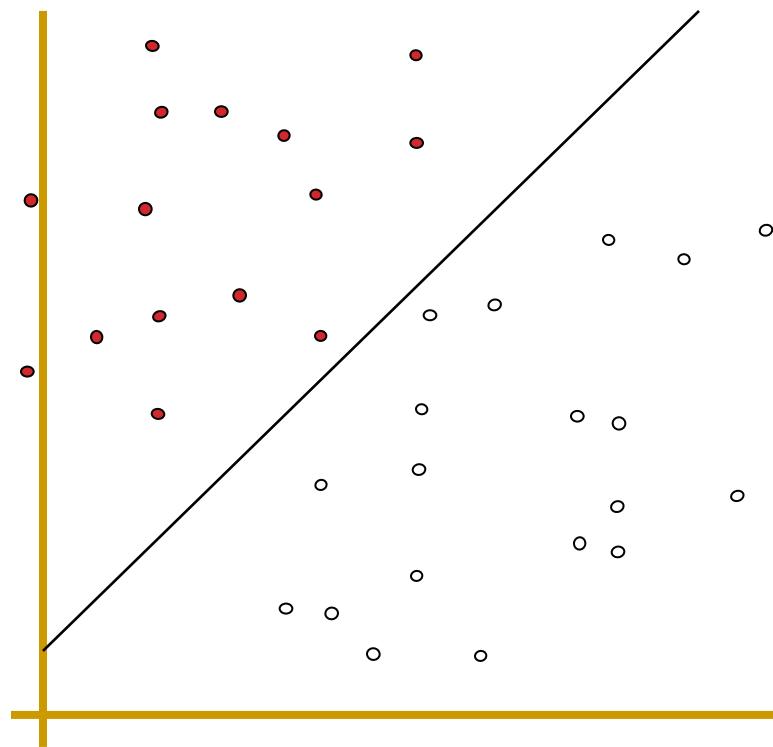
- ❑ El algoritmo k-NN es robusto frente al ruido (k moderado)
- ❑ Es bastante eficaz
- ❑ Es válido para clasificación y para predicción numérica (devolviendo la media o la media ponderada por la distancia)

## Desventajas

- ❑ Es muy ineficiente en memoria (almacenar todos los datos)
- ❑ La distancia entre vecinos podría estar dominada por variables irrelevantes
  - Selección previa de características
- ❑ Evaluar un ejemplo está en  $O(dn^2)$  siendo  $O(d)$  la complejidad temporal de la distancia utilizada
  - Una forma de reducir esta complejidad es usar prototipos

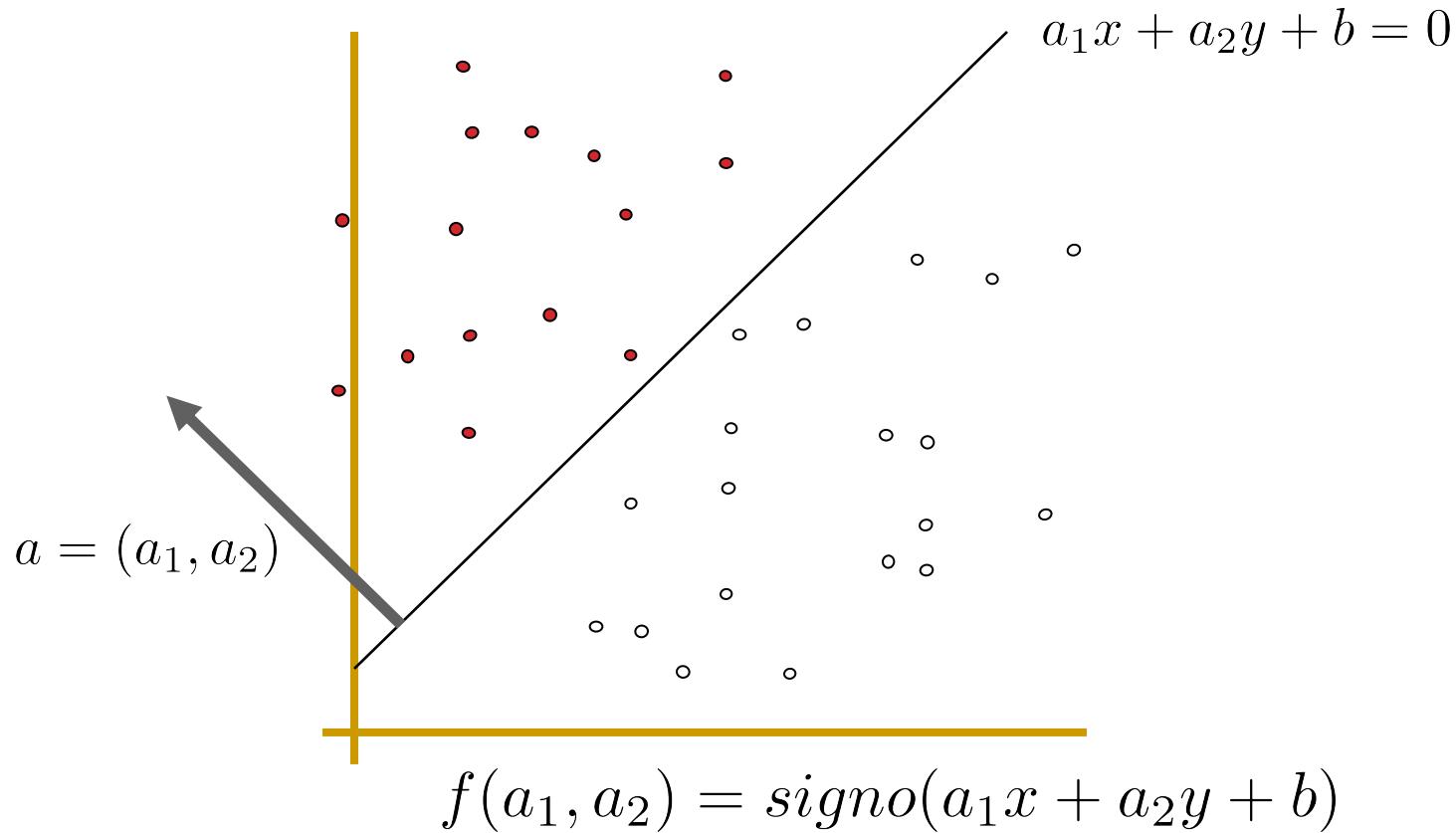
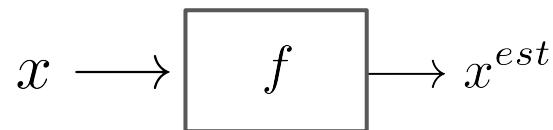
# Máquinas de vector soporte (SVM)

La idea básica es dividir el espacio donde se encuentran los datos de entrenamiento. Un nuevo dato se clasificará dependiendo de su posición



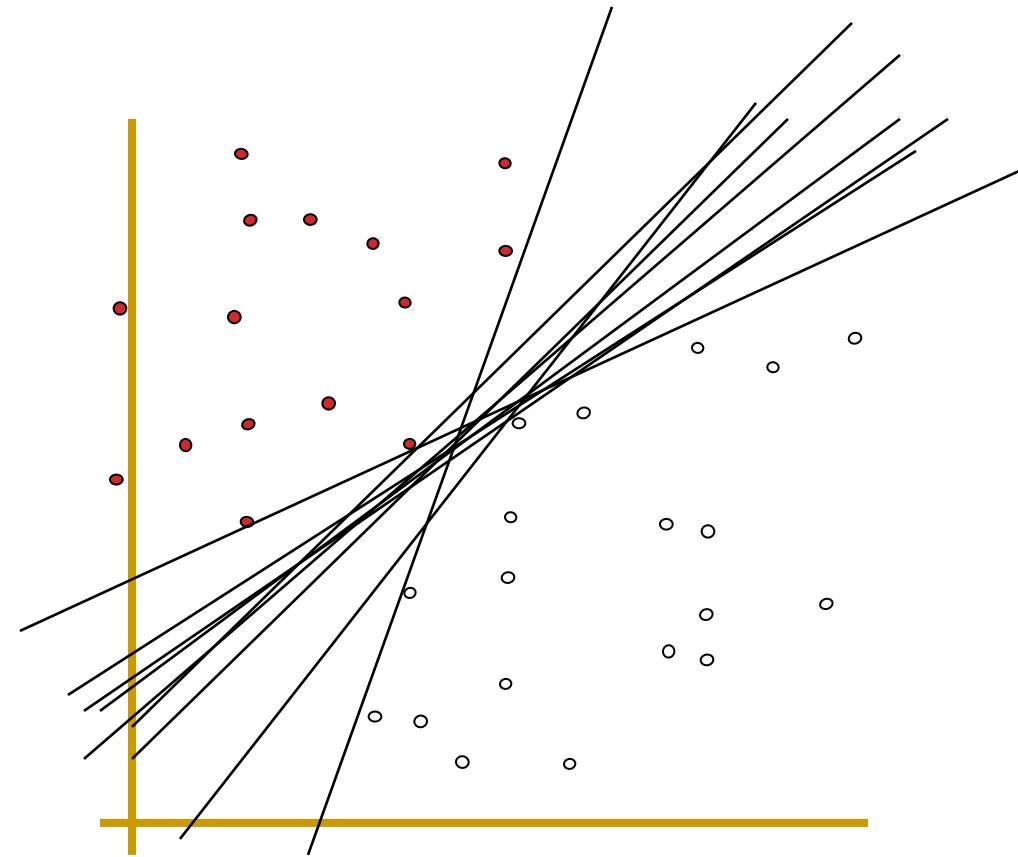
# SVM

Más concretamente, se clasifica con una función



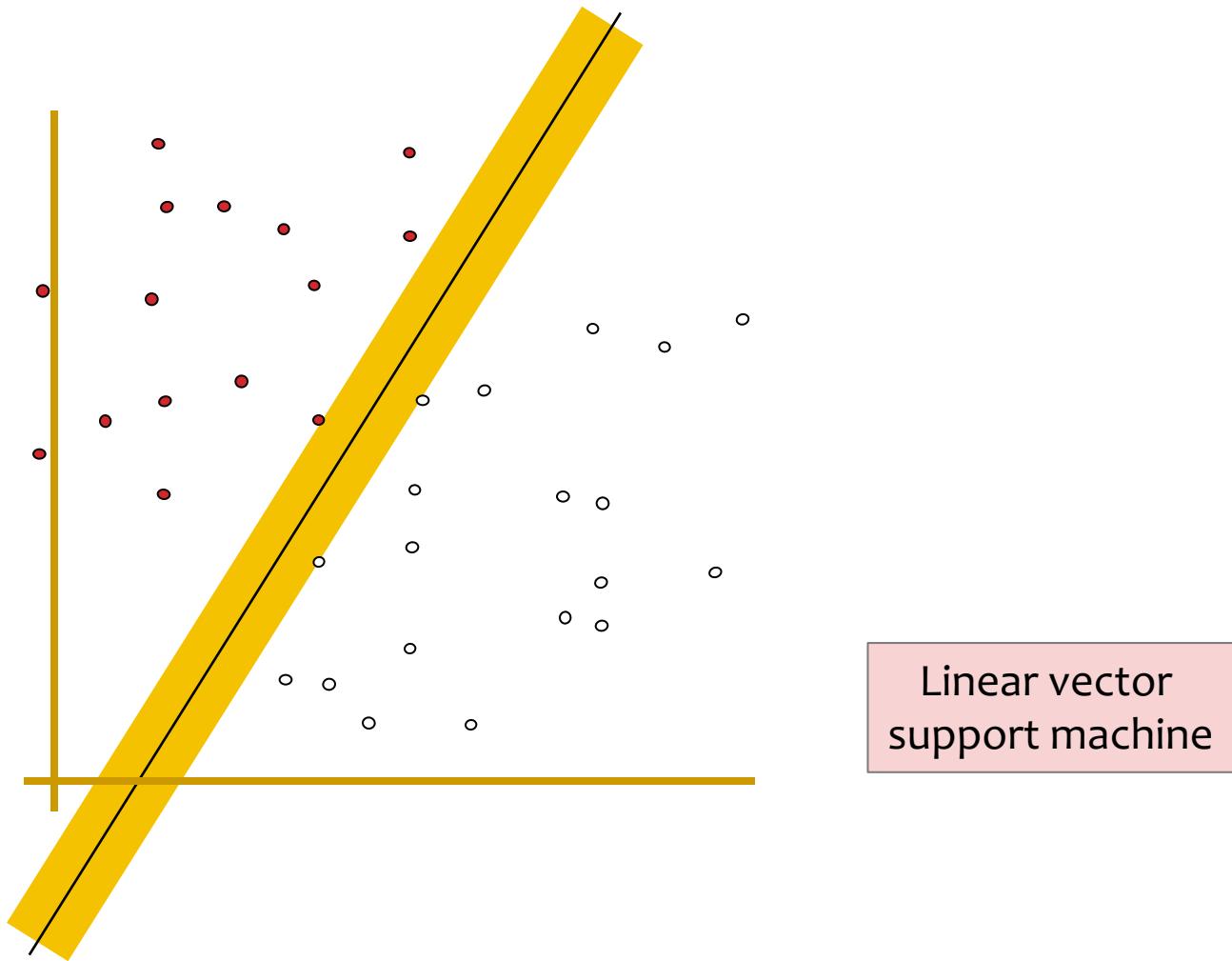
# SVM

Aunque hay muchas rectas que dividen estos puntos, ¿cuál es la mejor?



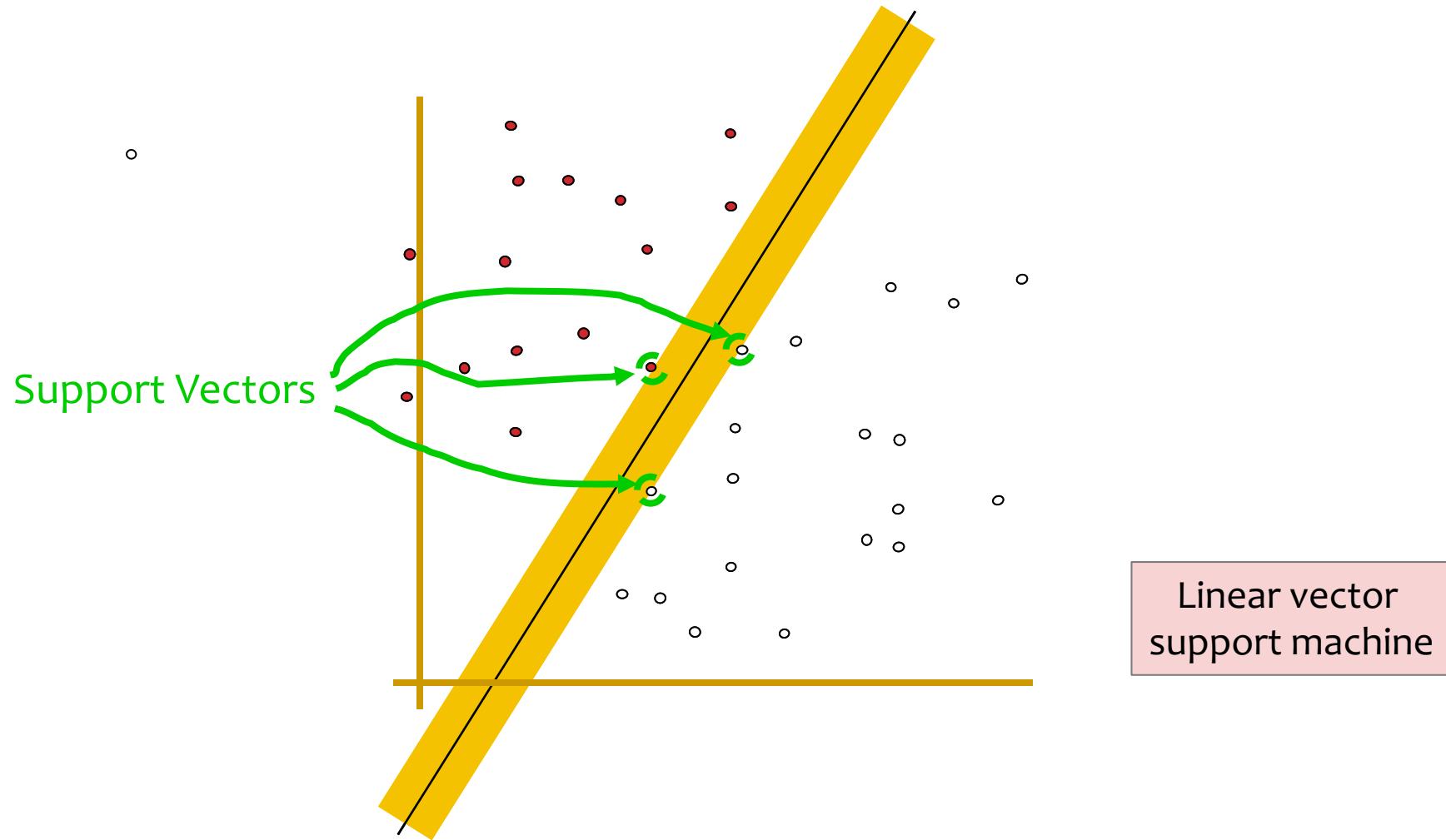
# SVM

Aquella que deja mayor margen entre los puntos



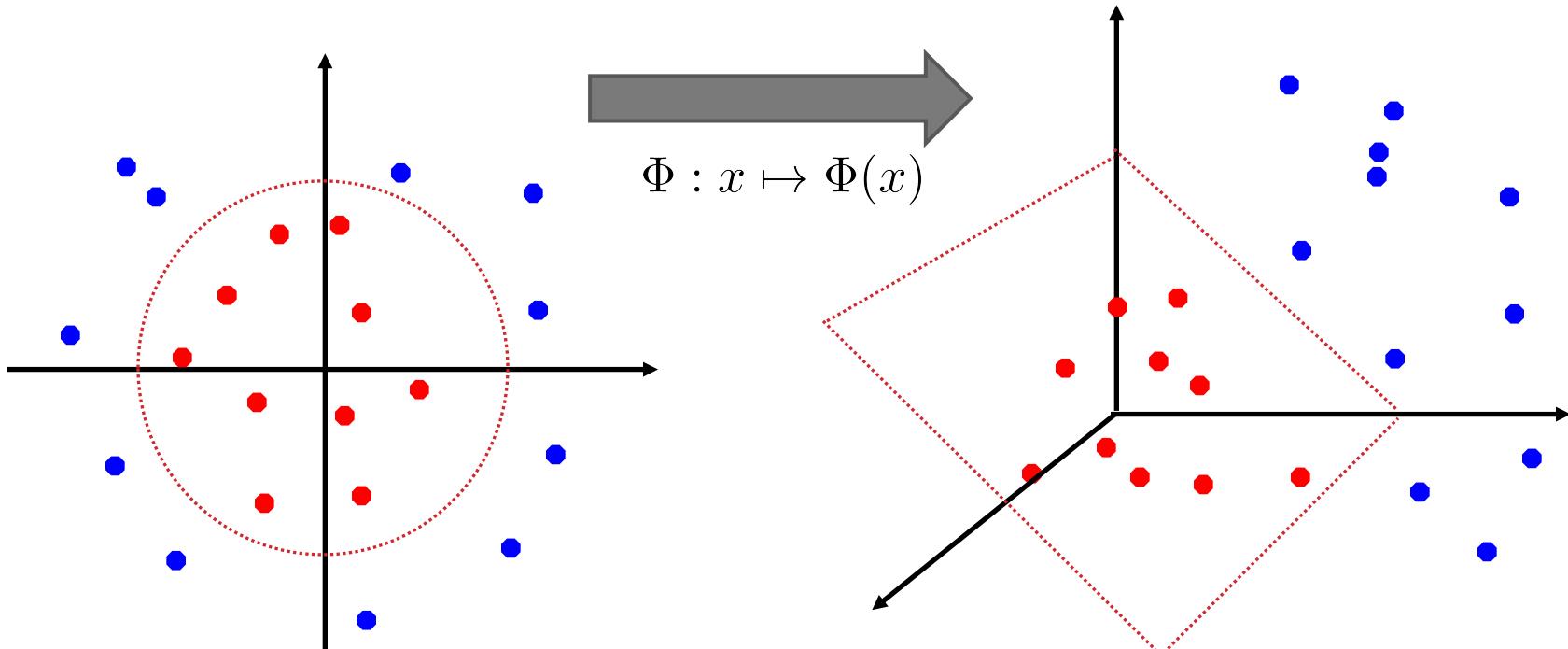
# SVM

Aquella que deja mayor margen entre los puntos



# SVM

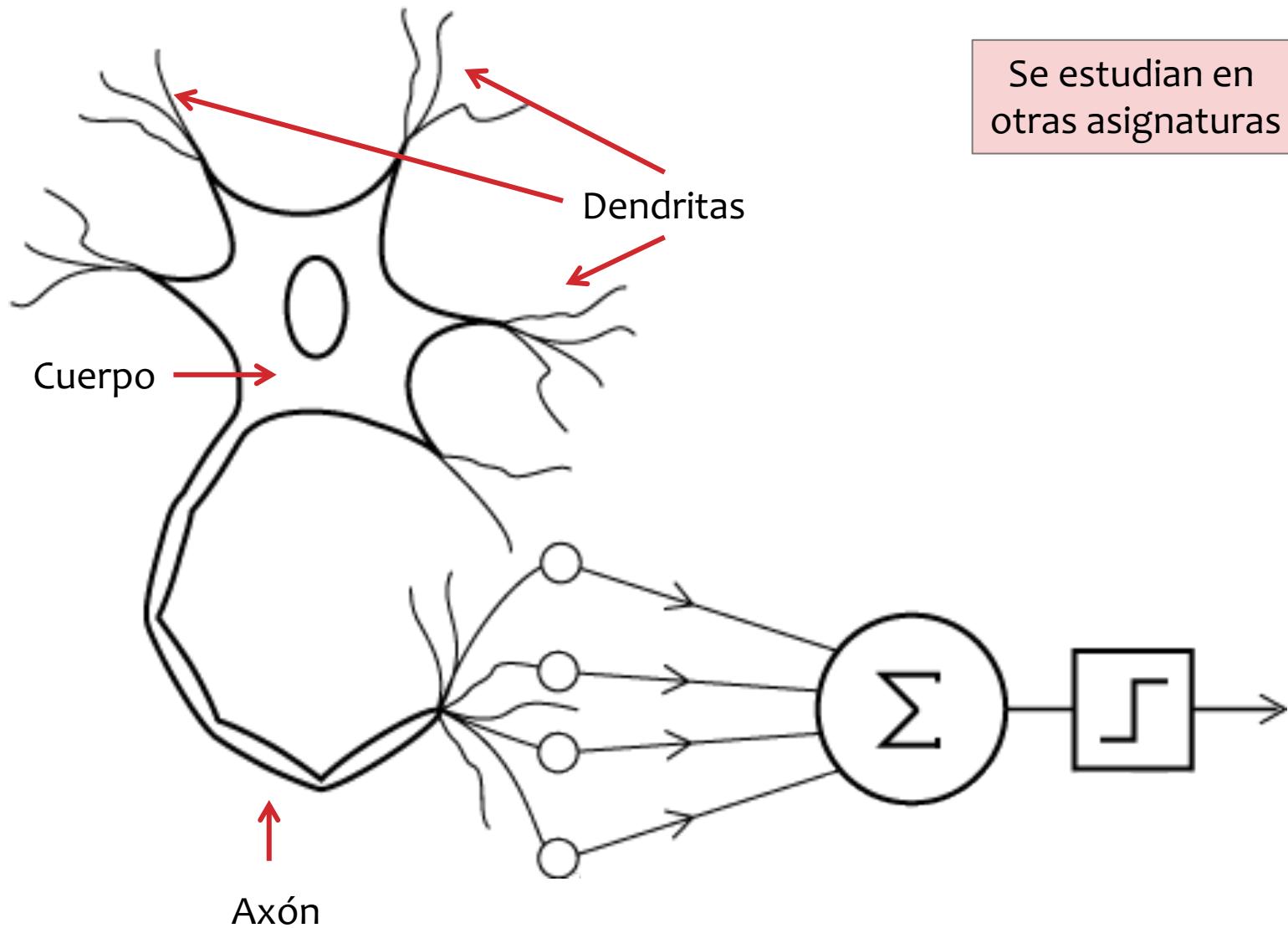
Si la separación no es lineal, transformamos el espacio



Normalmente la transformación es a un espacio con más dimensiones, lo que conlleva un mayor **coste computacional**

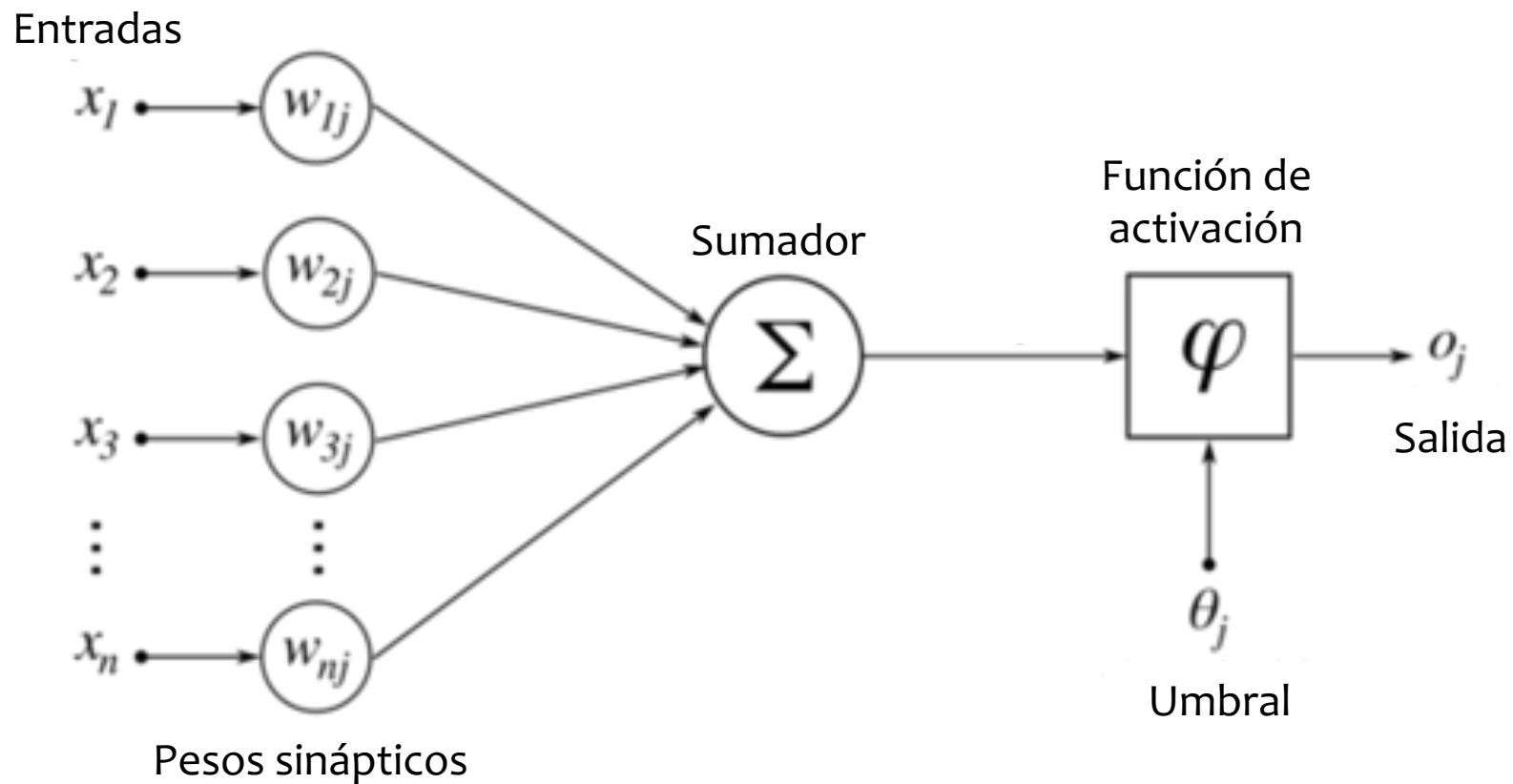
Kernel trick

# Redes neuronales artificiales



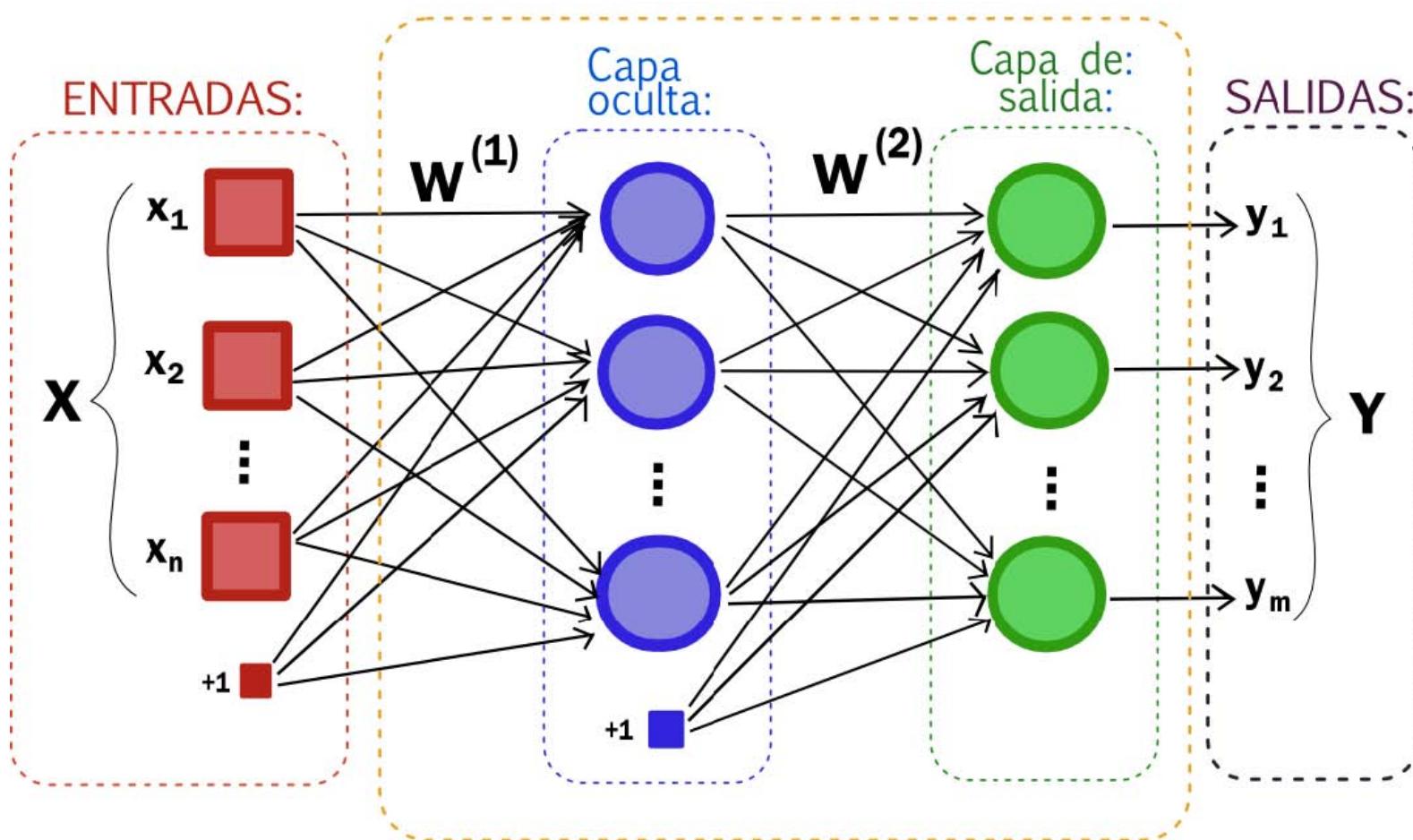
# Redes neuronales artificiales

## Neurona artificial



# Redes neuronales artificiales

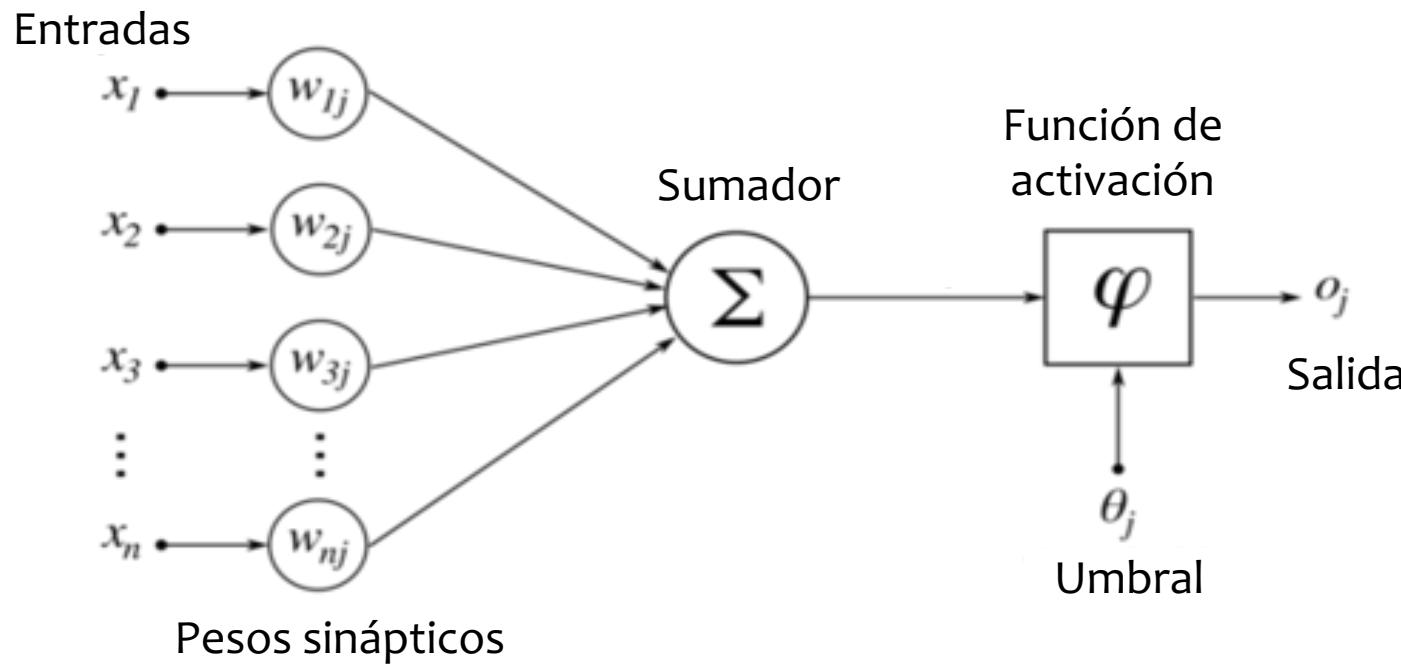
## Red neuronal



# Redes neuronales artificiales

Los componentes básicos de una neurona son:

- **Vector de pesos.** Un peso por cada entrada
- Un **sesgo/umbral** asociado a la neurona
- Una **función de activación** de la neurona



# Redes neuronales artificiales

## Funciones de activación usuales

- Función escalón     $f(x) = \begin{cases} -1 & \text{si } x \leq \theta \\ 1 & \text{si } x > \theta \end{cases}$
- Función lineal     $f(x) = x - \theta$
- Función sigmoidal     $f(x) = \frac{1}{1 + e^{-b(x-\theta)}}$
- Tangente hiperbólica     $f(x) = \tanh(b(x - \theta))$

# Redes neuronales artificiales

Resolver un problema de clasificación, implica determinar:

- Número de neuronas de la red (topología)

Normalmente, determinado por un experto

- Número de nodos de entrada
  - Uno por cada variable numérica
  - Para variables nominales, uno por cada valor de la variable
- Número de nodos de salidas
  - Tantos nodos como número de clases
- Número de capas ocultas

- Pesos y funciones de cada neurona

Mediante entrenamiento, propagando el conjunto de entrenamiento por la red

# Redes neuronales artificiales

## ¿Cómo clasificar?

- Cuando la red neuronal está entrenada, para clasificar una instancia, se introducen los valores de la misma que corresponden a las variables de los nodos de entrada
- La salida de cada nodo de salida indica la probabilidad de que la instancia pertenezca a esa clase
- La instancia se asigna a la **clase con mayor probabilidad**

# Redes neuronales artificiales

Ejemplo: **perceptrón simple** (sin capas ocultas)

1. Elegir pesos aleatorios pequeños, entre [-1,1]
2. Seleccionar una instancia  $x$  del conjunto de datos
3. Obtener salida  $y$
4. Si  $y=t$ , la salida real, volver al paso 2
5. En caso contrario, el vector de pesos  $w$  varía como

$$w_i = \alpha x_i(t_i - y_i)$$

donde  $\alpha$  es un número positivo (coeficiente de aprendizaje).

Volver al paso 2

# Redes neuronales artificiales

## Ventajas

- ❑ Habitualmente gran tasa de acierto en la predicción
- ❑ Son más robustas que los árboles de decisión por los pesos
- ❑ Robustez ante la presencia de errores (ruido, outliers,... )
- ❑ Gran capacidad de salida: nominal, numérica, vectores,...
- ❑ Eficiencia (rapidez) en la evaluación de nuevos casos
- ❑ Mejoran su rendimiento mediante aprendizaje y éste puede continuar después de que se haya aplicado al conjunto de entrenamiento

# Redes neuronales artificiales

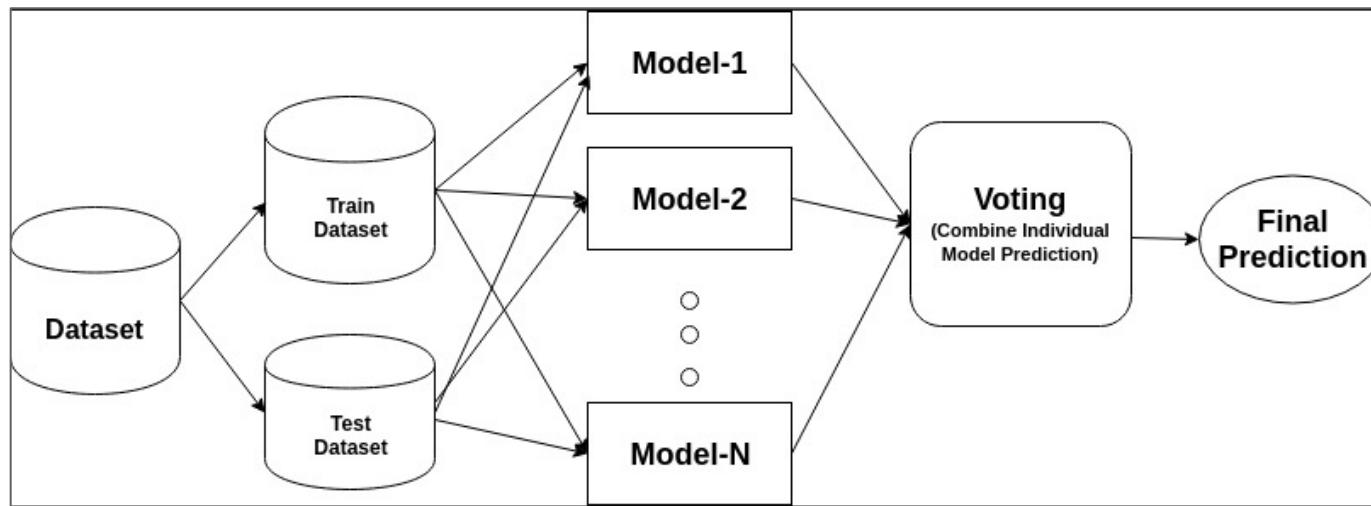
## Desventajas

- ❑ Necesitan mucho tiempo para el entrenamiento
- ❑ Entrenamiento: gran parte es ensayo y error
- ❑ Poca interpretabilidad del modelo (caja negra)
- ❑ Difícil de incorporar conocimiento del dominio
- ❑ Los atributos de entrada deben ser numéricos
- ❑ Generar reglas a partir de redes neuronales no es inmediato
- ❑ Pueden tener problemas de sobreaprendizaje

# Ensembles

Combinar varios modelos para mejorar la precisión del clasificador. Para decidir qué clase se asigna:

- Bagging**, votación por mayoría
- Boosting**, votación ponderada, según calidad del clasificador (AdaBoost)



# Trabajos evaluables

(3 personas, 30 minutos) **Máquinas de vectores soporte.**

Bibliografía:

- Introducción a la Minería de Datos. José Hernández Orallo, M.José Ramírez Quintana, Cèsar Ferri Ramírez. Pearson, 2004. **Capítulo 14.**
- C. Aggarwal, Data Mining: The textbook, Springer, 2015. **Capítulo 10.6.**
- J. Han, M. Kamber and J. Pei. Data Mining, Second Edition: Concepts and Techniques. Morgan Kaufmann, 2006. **Capítulo 9.**
- N. Cristianini and J. Shawe-Taylor, An introduction to support vector machines and other kernel-based learning methods, Cambridge University Press, 2000

(3 personas, 30 minutos) **Redes neuronales.** Bibliografía:

- Introducción a la Minería de Datos. José Hernández Orallo, M.José Ramírez Quintana, Cèsar Ferri Ramírez. Pearson, 2004. **Capítulo 13.**
- <https://www.infor.uva.es/~teodoro/neuro-intro.pdf>

# Bibliografía

- Introducción a la Minería de Datos. José Hernández Orallo, M.José Ramírez Quintana, Cèsar Ferri Ramírez. Pearson, 2004. **Capítulos 6, 10, 11, 13, 14, 16 y 17.**
- J. Han, M. Kamber and J. Pei. Data Mining, Second Edition: Concepts and Techniques. Morgan Kaufmann, 2006. **Capítulos 8 y 9.**
- Mohammed J. Zaki, Wagner Meira, Jr., Data Mining and Analysis: Fundamental Concepts and Algorithms, Cambridge University Press, May 2014. **Capítulo 9**
- C. Aggarwal, Data Mining: The textbook, Springer, 2015.

Algunas transparencias y gráficos tomados de:

- <http://sci2s.ugr.es/docencia/in/>
- <http://elvex.ugr.es/idbis/dm/>