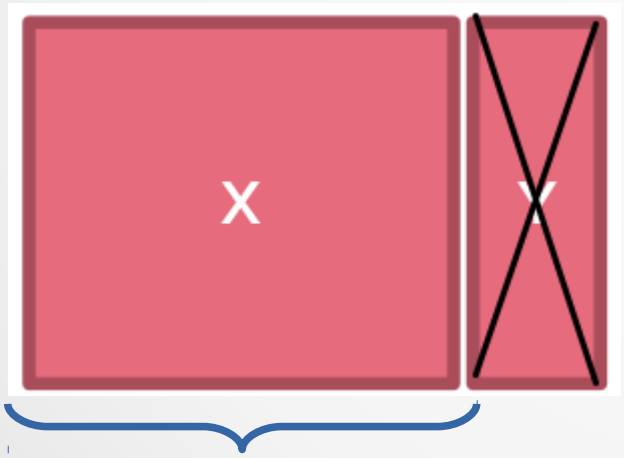
MINERÍA DE DATOS MODELADO

Bases de Datos Masivas

Aprendizaje No Supervisado

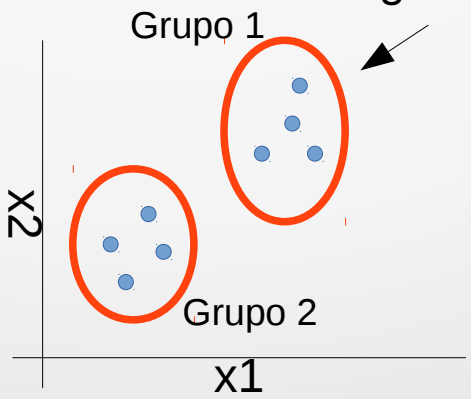
- No hay una variable objetivo identificada en el conjunto de datos.
- En su lugar, el algoritmo de minería de datos busca patrones y estructuras entre todas las variables.



Solo tenemos el vector de características X

- La etiqueta (o clase) de cada tupla de training no se conoce.
- El número clases (o conjuntos) no puede ser conocido de antemano para ser aprendidas.

¿Existe alguna estructura en los datos?



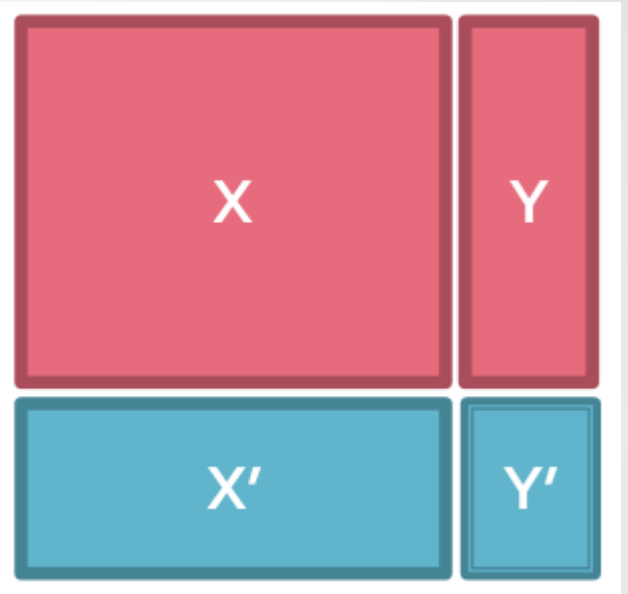
El algoritmo No supervisado podría decidir que los datos están en dos grupos diferentes

Aprendizaje Supervisado

Queremos realizar una **predicción**:
estimar una función **f(x)** de forma que **y = f(x)**

Donde Y puede ser:
- **Número real** de regresión
- **Categorías**: Clasificación
- Objeto complejo:
  - Ranking de los artículos
  - árbol de análisis, etc.

Los datos se **etiquetan**:
Tenemos muchos pares {(x, y)}
- **x ...** vector de variables binarias, categóricas, características expresadas en valores reales
- **Y ...** la clase ({-1, 1}, o un número real)



Conjunto de **Training** y **test**

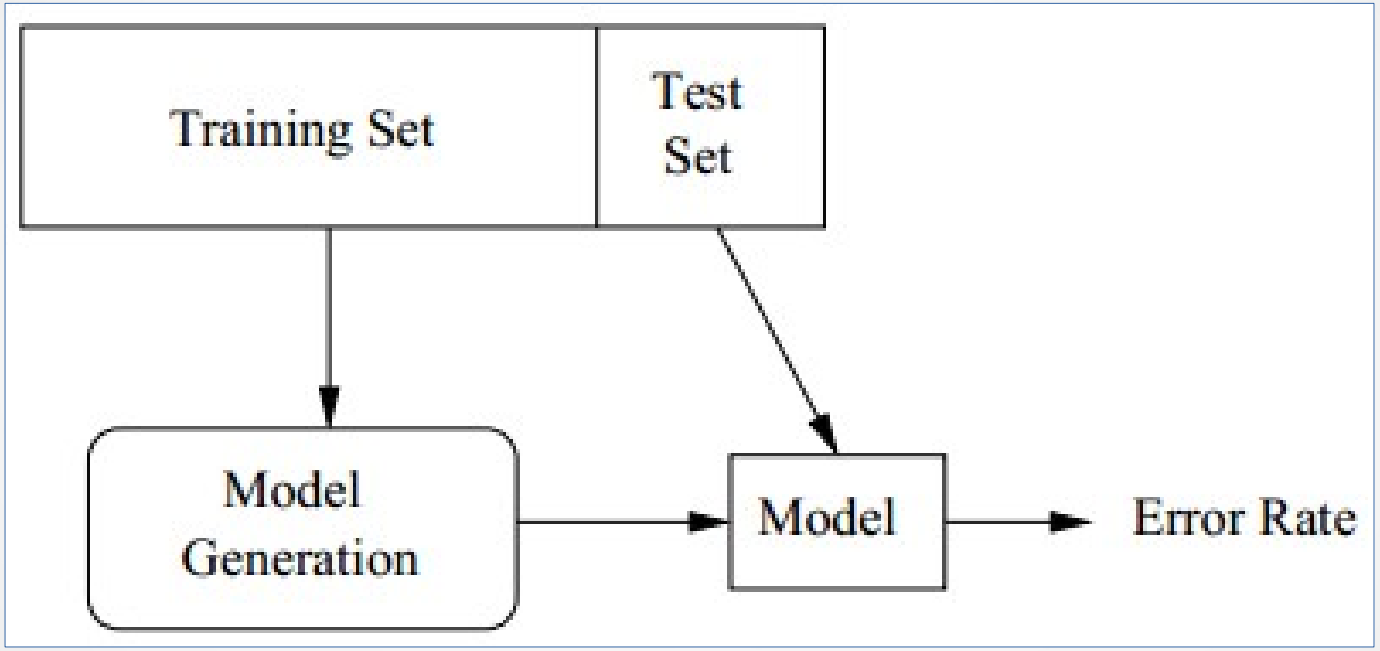
Estimar **y = f(x)** en **X, Y**.
Espero que la misma **f(x)** también funcione **en lo no visto X ', Y'**

Estadística vs Data Mining

- La aplicación de la inferencia estadística utilizando los enormes tamaños de las muestras encontradas en la minería de datos tiende a dar lugar a la significación estadística, incluso cuando los resultados no son de importancia práctica.

- En la metodología estadística, el analista de datos tiene una hipótesis a priori en mente. Los procedimientos de minería de datos, por lo general no tienen una hipótesis a priori.

Arquitectura de Machine-Learning



Holdout vs Subsampling

Random sampling: es una variación de ***Holdout***

Se repite **holdout** k veces, accuracy = promedio de los accuracy obtenidos en cada K

Cross-validation

Consiste en particionar al azar el conjunto de entrenamiento en K subconjuntos mutuamente excluyentes y cada unos aproximadamente de igual tamaño.

Las versiones más conocidas son:

- K-Fold CV
- Leave-One-Out CV

Holdout vs Subsampling

Random sampling: es una variación de ***Holdout***

Se repite **holdout** k veces, accuracy = promedio de los accuracy obtenidos en cada K

Cross-validation

Consiste en particionar al azar el conjunto de entrenamiento en K subconjuntos mutuamente excluyentes y cada unos aproximadamente de igual tamaño.

Las versiones más conocidas son:

- K-Fold CV
- Leave-One-Out CV

Validación Cruzada

- El analista realiza acciones sobre los datos: filtra, transforma, normaliza, etc, etc
- La minería de datos puede llegar a resultados falsos debido a la variación aleatoria en lugar de efectos reales.
- Es crucial evitar la “filtración” de información.
- La validación cruzada (*Cross Validation*) nos va a permitir evitar algunos de esos problemas.

La validación cruzada es una técnica para asegurar que los resultados descubiertos en un análisis son generalizables a un conjunto de datos independientes, que no han sido vistos.

En DM las validaciones cruzadas más comunes son:

- **2-Folds CV**: los datos se particionan, utilizando la asignación al azar, en un conjunto de **datos de entrenamiento** y **un conjunto de datos de prueba.** Esa separación debe ser validada.

- **K-Folds CV**: los datos originales se dividen en k subconjuntos independientes y similares. El modelo se construye utilizando k – 1 conjuntos. Y se utiliza el k-ésimo como testing.

Validación Cruzada

En K-Folds CV.

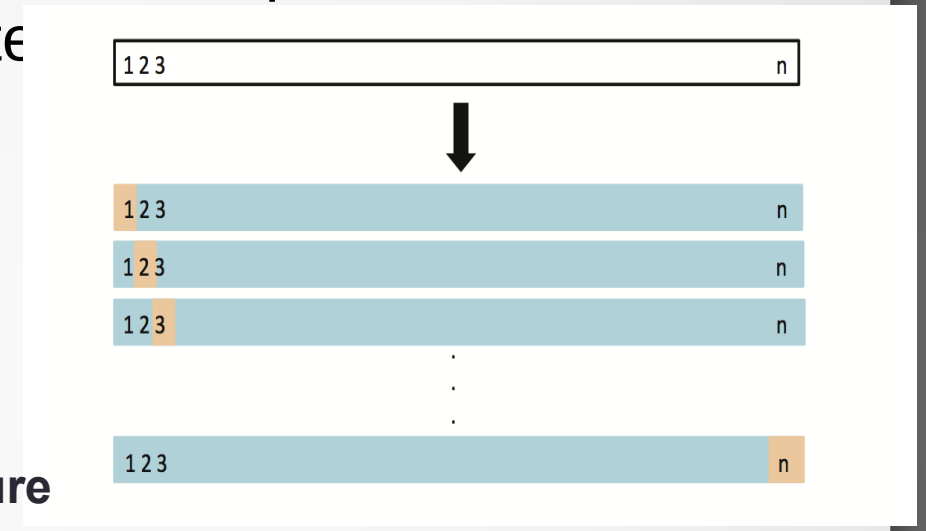
- El proceso de entrenamiento se realiza de forma iterativa hasta que tengamos k diferentes modelos.
- Los resultados de los k **modelos se combinan** (o se elige el mejor) entonces utilizando el promedio o la votación.
- El valor (empírico) generalmente utilizado para k es 10.
- Una **ventaja** de utilizar k veces validación cruzada es que cada registro aparece en el conjunto de prueba exactamente una vez.
- Una **desventaja** es que se complica la tarea de validación.

Leave-One-Out

Este método es similar al método de Holdout, pero trata de hacer frente a las desventajas de este

Para cada modelo:

- Split the data set of size n into
  - Training data set (blue) size: **n -1**
  - Validation data set (beige) size: **1**
- Fit the model using the training data
- Validate model using the validation data, and compute the corresponding **accuracy measure**
- Repeat this process n times

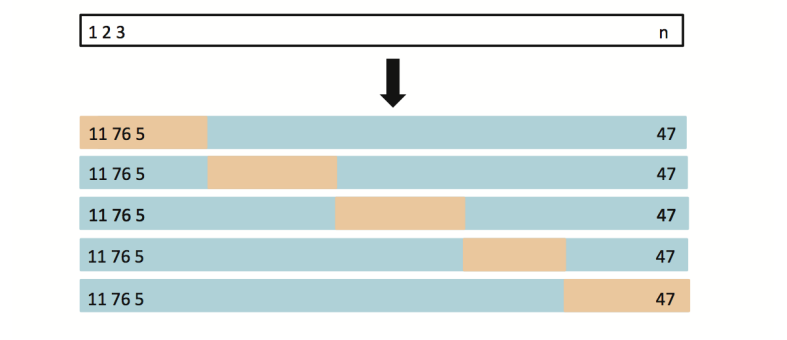


- LOO tiene menos sesgo. El modelo ve todos los casos durante el ajuste.
- LOO reduce la varianza del **accuracy measure** dado que usa siempre casi todo el dataset. Por lo tanto, va a dar siempre lo mismo.
-
- Desventaja: LOO tiene es costoso computacionalmente

K-Fold

- Entonces, si LOO es computacionalmente costoso podemos utilizar K-Fold
- Con K-Fold, Dividimos el dataset en K diferentes partes.
  K = 5 ó K = 10 ← Son los K mágicos

- Entonces, quitamos la primera parte, ajustamos el modelo en el resto de **K - 1 partes**, y evaluamos con la parte que dejamos afuera.
- Repetimos el proceso de entrenamiento K veces dejando afuera una en cada corrida.
- Promediando los K de diferentes medidas de rendimiento obtenemos una validación.



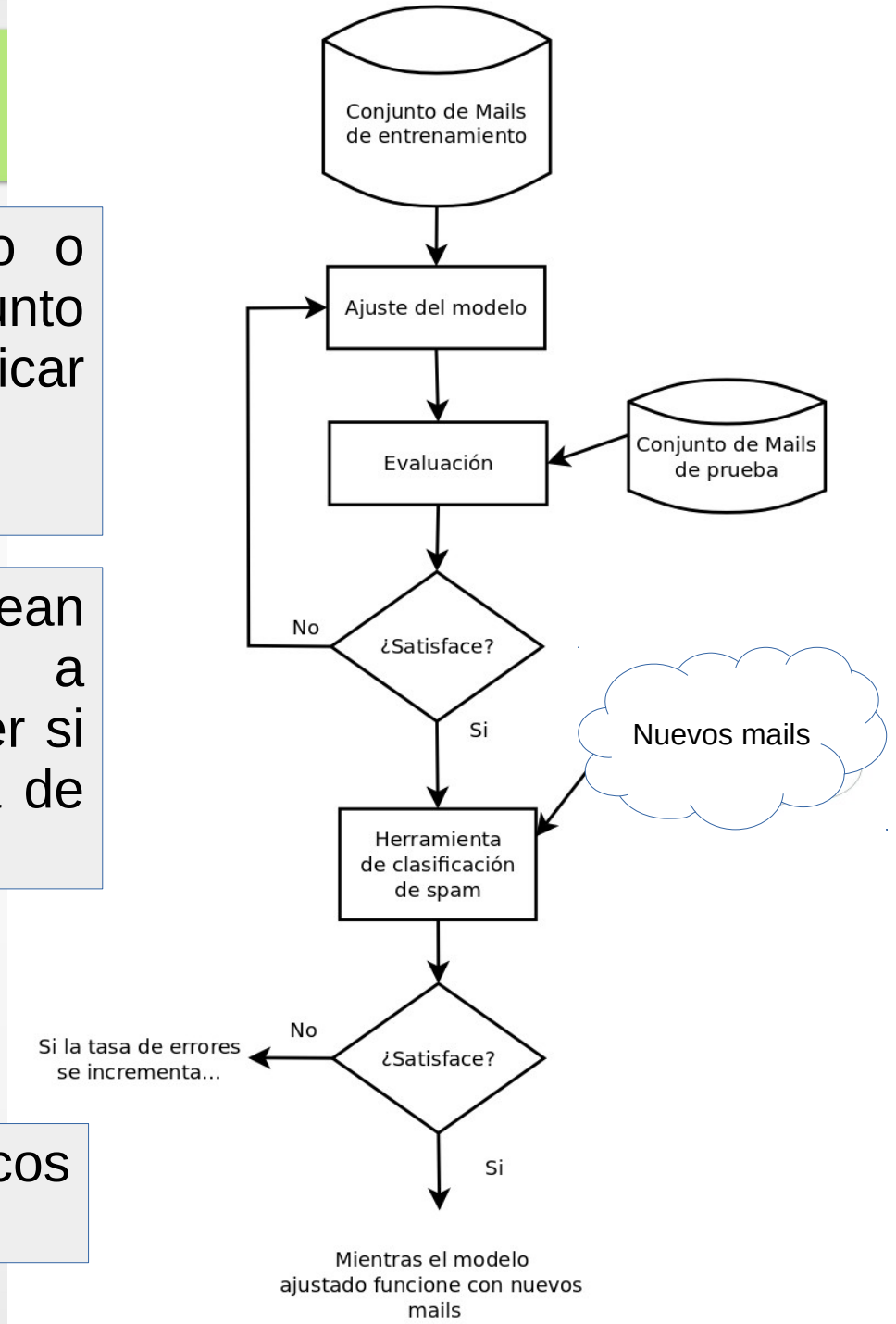
Está demostrado empíricamente que se obtienen estimaciones de la tasa de error en testing que no sufren de sesgo excesivamente alto ni varianza alta

Generalización

- El propósito de crear un modelo o clasificador no es clasificar el conjunto de entrenamiento, sino para clasificar los datos cuya clase no sabemos.

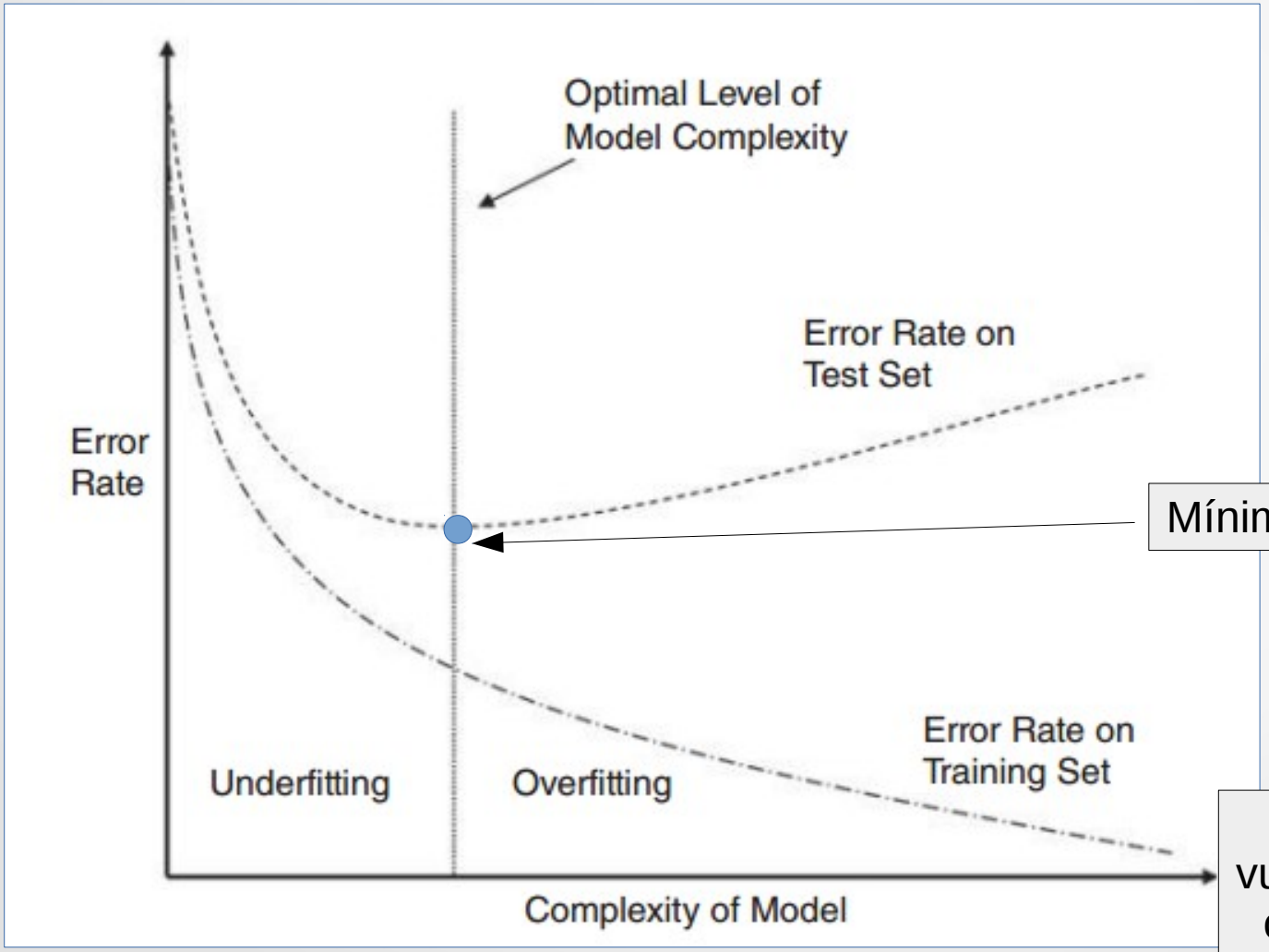
- Queremos que los datos sean clasificados correctamente, pero a menudo no tenemos forma de saber si el modelo lo hace. Si la naturaleza de los datos cambian con el tiempo.

- Ejemplo: detectar correos electrónicos no deseados



Sobreajuste

- Cuando **la exactitud del modelo** no es tan alta en el conjunto de testing como lo es en el de entrenamiento, a menudo es debido a que el modelo **sobreajusta** el conjunto de entrenamiento.

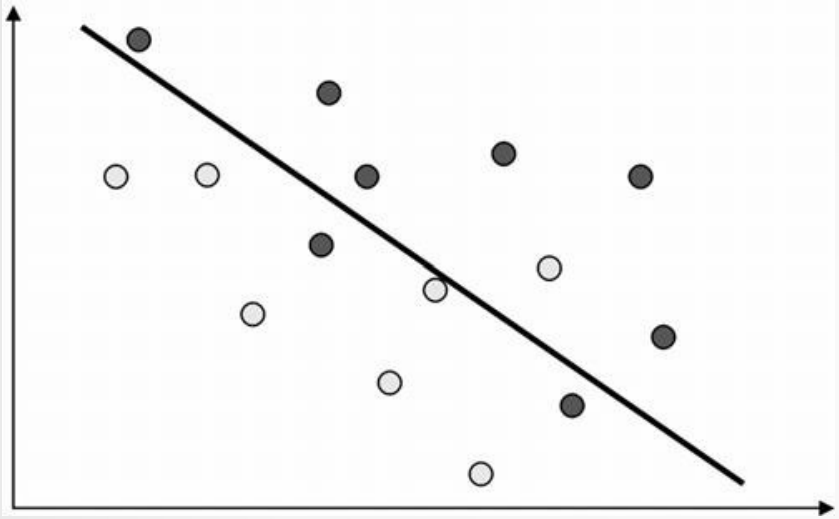


Mínimo error en test

Conforme el modelo se vuelve más complejo pierde capacidad de generalizar

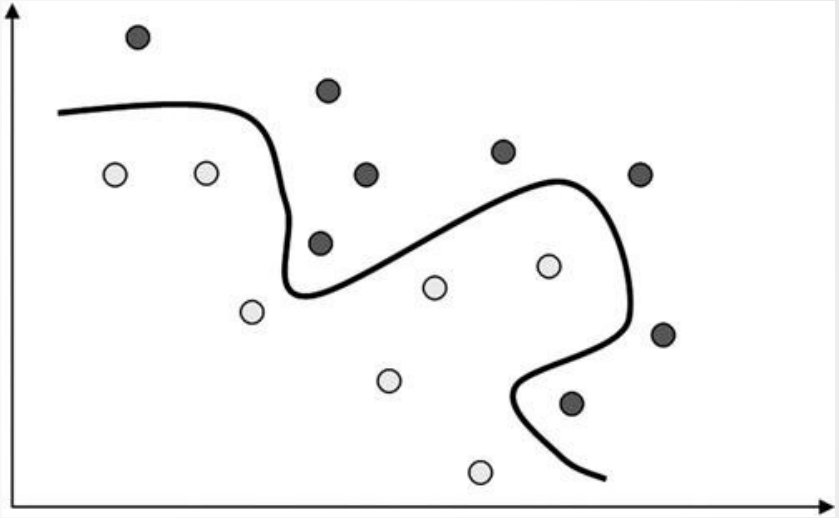
Sesgo–Varianza Trade-Off

Modelo 1: **Linea recta**



La línea recta que tiene el beneficio de **baja complejidad** pero **adolece de algunos errores** de clasificación.

Modelo 2: **Curvilínea**



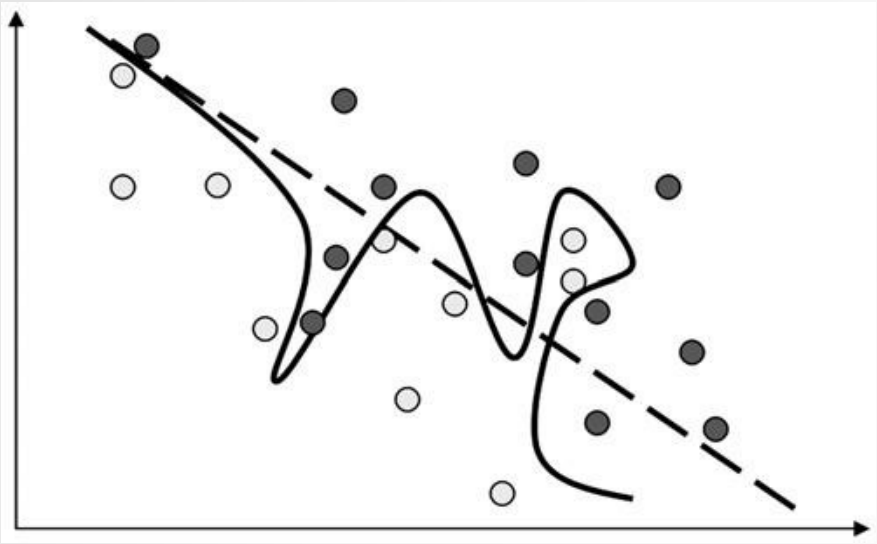
Se **reduce el error** de clasificación a cero, pero a costa de un modelo **más complejo**.

Uno podría estar tentado a adoptar la mayor complejidad con el fin de reducir la tasa de error.

Ojo con la idiosincrasia del conjunto de entrenamiento

Sesgo–Varianza Trade-Off

¿Qué pasa si agrego datos?



- El modelo de baja complejidad (la línea recta) no necesita cambiar mucho para acomodar los nuevos puntos de datos. Esto significa que este separador de baja complejidad tiene una **baja varianza**.

- El modelo de alta complejidad, debe alterar considerablemente si se trata de mantener su tasa de error original. Este alto grado de cambio indica que el separador de alta complejidad tiene una **alta varianza**.

En términos de la tasa de error del conjunto de entrenamiento:

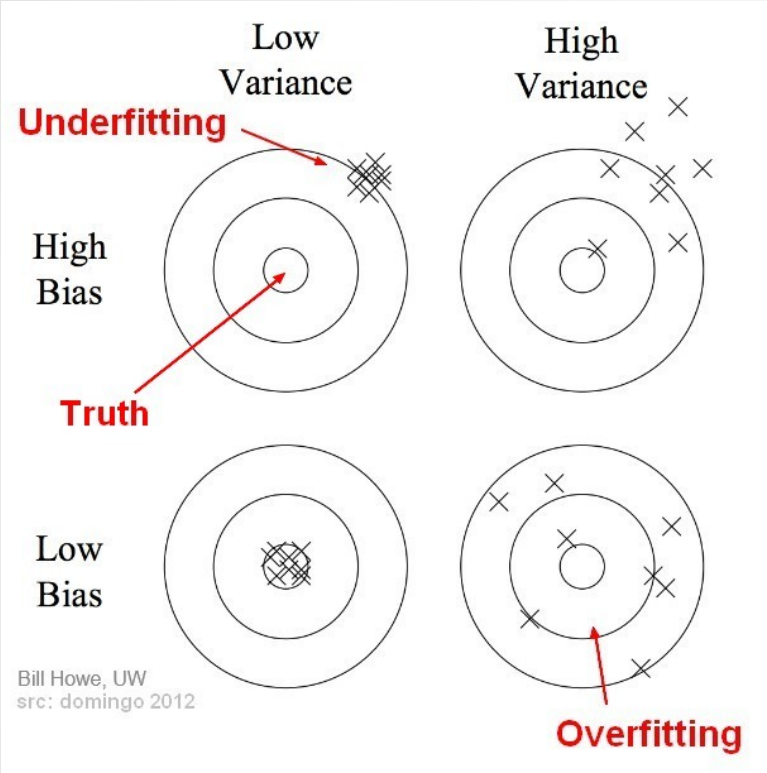
- El modelo de alta complejidad tiene un **sesgo bajo** pero **alta varianza**
- El modelo de baja complejidad tiene un **sesgo alto** pero **baja varianza**

Este es el dilema Sesgo–Varianza

Ejemplo:

$$MSE = variance + bias^2$$

Conceptos de Sesgo y Varianza



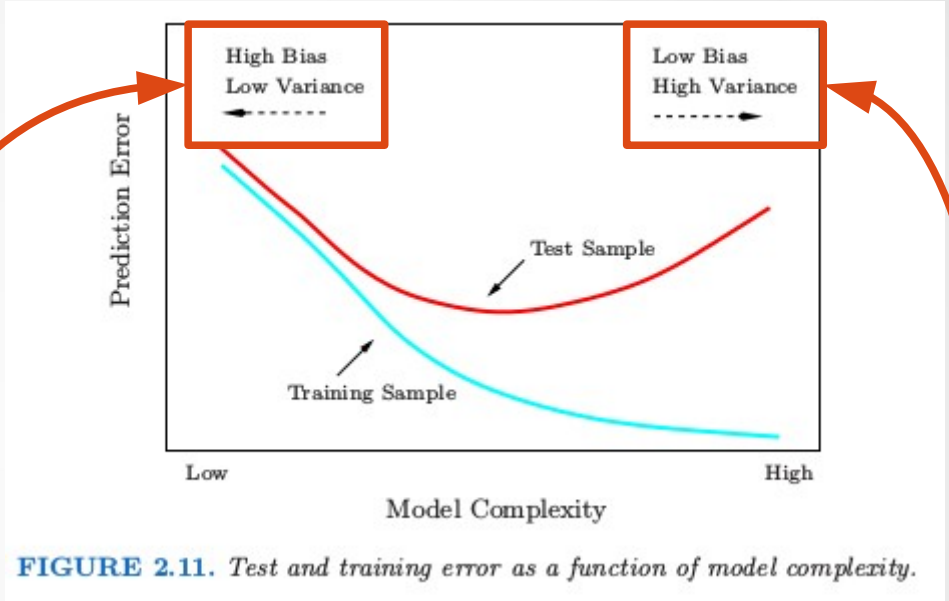


FIGURE 2.11. *Test and training error as a function of model complexity.*

¿Qué funciones utilizar?

Funciones rígidas:

Buena estimación de los parámetros óptimos – poca flexibilidad

Funciones flexibles:
Buen ajuste – mala estimación de los parámetros óptimos

Machine Learning A Cappella - Overfitting Thriller!

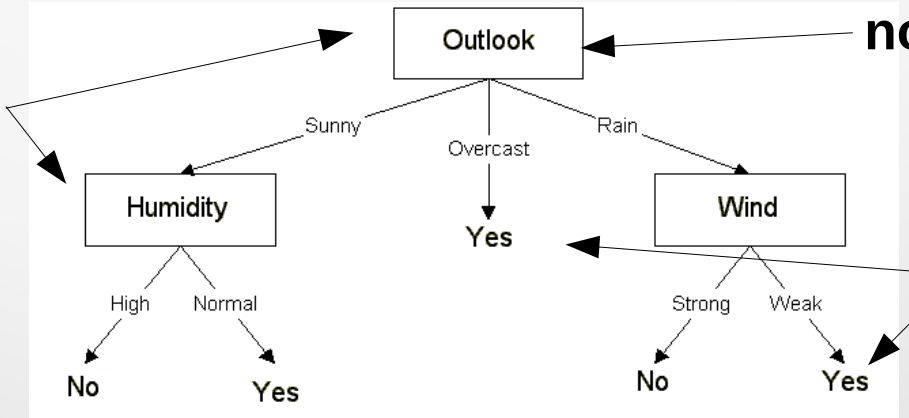
Árboles de Decisiones

- ¿Qué es un árbol de decisión?
- ¿Cuando se puede aplicar?
- Algoritmos:
  - IDE3
  - C4.5
  - CART

¿Qué es un árbol de decisión?

- Es un **método de clasificación**
- Formado por una **colección de nodos de decisión** que se conectan por **ramas** que se extiende hacia abajo desde el **nodo raíz** hasta que termina en **nodos hoja**. **Es un DAG**
- El **nodo raíz**, por convención se coloca en la parte superior del diagrama de decisión
- Los atributos se evalúan en los nodos de decisión, con cada resultado posible lo que resulta es una rama.
- Cada rama conduce bien a otro **nodo de decisión** o de un **nodo hoja**.

nodos de decisión



nodo raíz

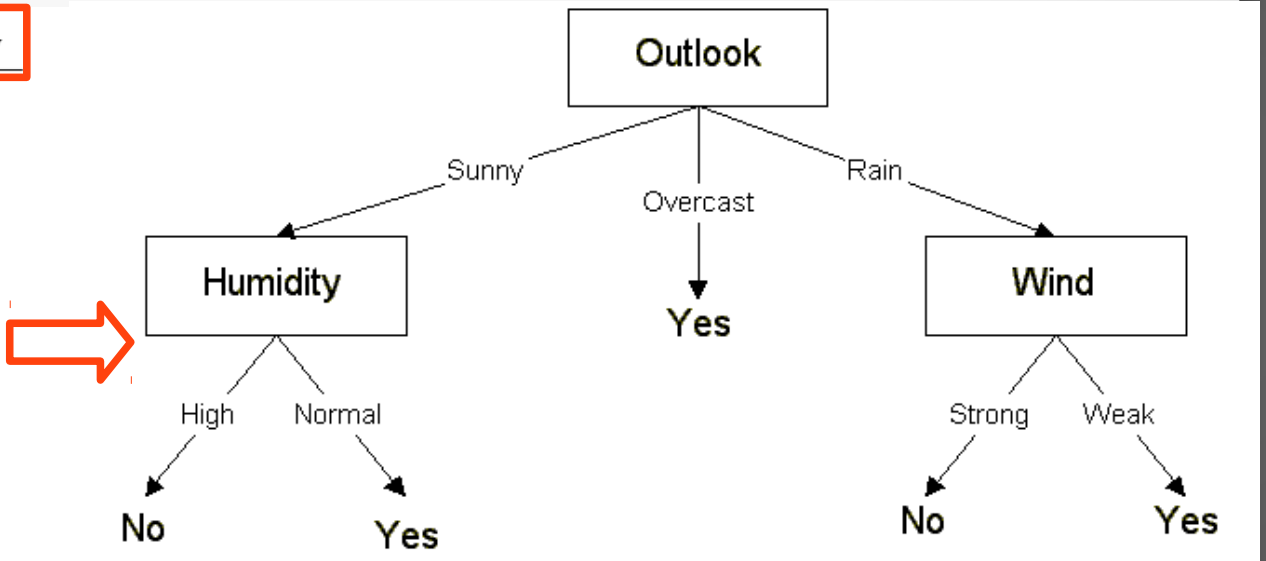
nodo hoja

¿Qué es un árbol de decisión?

Clase

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Table 1: The weather data (Witten and Frank; 1999, p. 9).



¿Cuando se puede aplicar?

1) Los algoritmos de búsqueda de reglas para la construcción de árboles de decisiones utilizan **aprendizaje supervisado**.

   En consecuencia, se va a requerir un **conjunto de datos de entrenamiento** con los valores de la variable objetivo.

2) Este conjunto de datos de entrenamiento debe tener una **alta variedad** y debe proporcionar al algoritmo diferentes registros de ejemplo que permitan la clasificación en el futuro de ejemplos nos vistos.

   Los árboles de decisión aprenden con ejemplos – y si faltan ejemplos – solo serán útiles para un subconjunto reducido de registros.

3) **Las clases o variable objetivo debe ser discreta**. Es decir, no se puede aplicar el análisis de árboles de decisiones a una variable destino continua. Por el contrario, la variable de destino debe tomar valores que están claramente demarcados.

Algoritmos: ID3

- ID3 es un simple algoritmo de aprendizaje árbol de decisión desarrollado por Ross **Quinlan** (1983).
- La idea básica del algoritmo ID3 es construir el árbol de decisión usando un enfoque Top-Down de búsqueda greedy a través de los conjuntos dados para poner a prueba cada atributo en cada nodo del árbol.
- ID3 construye un árbol de decisiones a partir de un conjunto de ejemplos.
- El árbol resultante se utiliza para clasificar las muestras futuras.
- Cada ejemplo tiene varios atributos que pertenecen a una clase (sí/no).
- Los nodos hoja del árbol de decisión contienen el nombre de la clase, mientras que un nodo no hoja es un nodo de decisión.
- El nodo de decisión es una prueba de atributo con cada rama (a otro árbol de decisión) con cada posible valor del atributo.
- ID3 utiliza **ganancia de información** para ayudarle a decidir qué atributo entra en un nodo de decisión.

Algoritmo ID3: Entropía

- Necesitamos una función que pueda medir qué preguntas proporcionan la división más equilibrada.
  - Entropía (o medida de desorden)
  - Ganancia de información.

Entropía

Dado un conjunto **S**, que contiene la variable objetivo con sus valores positivos y negativos, la entropía de **S** en relación con esta clasificación booleana es:

$$Entropy(S) = -P(posi) * \log_2(P(posi)) - P(nega)\log_2(P(nega))$$

Donde:

P(posi): es la proporción de + en S
P(nega): es la proporción de - en S

Ganancia de Información

- Para minimizar la profundidad de árbol de decisión, tenemos que seleccionar el atributo óptimo para dividir el nodo del árbol.
- Este es el atributo con la mayor reducción de la entropía.
- Definimos ganancia de información como la reducción de la entropía esperada relativa a un determinado atributo cuando se divide un nodo del árbol de decisión.

GI

La ganancia de Información, Gain(S,A) de un atributo A,

$$Gain(S, A) = Entropy(S) - \sum_{v=1}^{v=N} \frac{|S_v|}{|S|} * Entropy(S_v)$$

Podemos utilizar esta noción de GI para clasificar los atributos y para construir árboles de decisión, donde en cada nodo se localiza el atributo con mayor GI entre los atributos aún no considerados en el camino desde la raíz.

PROBLEMA: La medida Ganancia favorece aquellas variables con mayor número de posibles valores

```
Algoritmo ID3(Ejemplos, Clase, Atributos)
   Ejemplos: Ejemplos de aprendizaje.
   Clase: Atributo a predecir por el árbol.
   Atributos: Lista de atributos a comprobar por el árbol.
   (1)  Crear una raíz para el árbol.
   (2)  Si todos los ejemplos son positivos    Return(raiz,+)
   (3)  Si todos los ejemplos son negativos Return(raiz,-)
   (4)  Si Atributos=∅ Return(raiz,l)
      ( l es el máximo valor común de Clase en Ejemplos )
   Si ninguna de las anteriores condiciones se cumple
   Begin
      (1) Seleccionar el atributo A con mayor Ganancia(Ejemplos, A)
      (2) El atributo de decisión para raíz es A
      (3) Para cada posible valor v_i de A
         (3.1) Añadir una rama a raíz con el test A=v_i
         (3.2)  Ejemplos_v_i es el subconjunto de Ejemplos con
                valor v_i para A
         (3.3) Si Ejemplos_v_i = ∅  entonces
                  añadir un nodo (n, l) a partir de la rama creada.
               sino añadir a la rama creada al subárbol
                  ID3(Ejemplos_vi, Clase, Atributos-{A})
   End
   Return(raíz)
```

ID3: Características

- Espacio de hipótesis completo
- Hipótesis única en cada momento de tiempo
- No se realiza “backtracking”
- Búsqueda no incremental
- Principio de “la navaja de Occam” (MDL)
- Saturación sobre los datos (“overfitting”)

Occam's razor: Prefer the simplest hypothesis that fits the data.

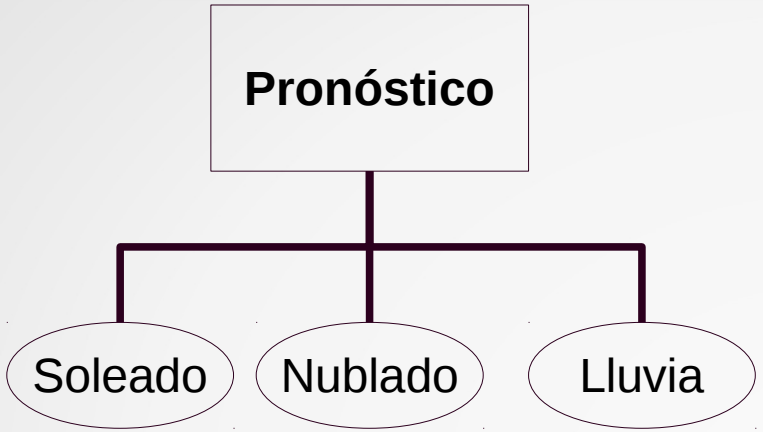
Approximate inductive bias of ID3: Shorter trees are preferred over larger trees.

Ejemplo: ¿Hay *fulbito*?

Clase

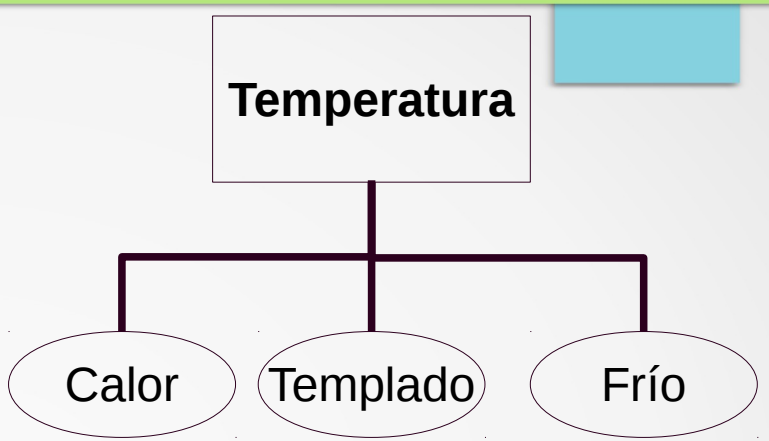
	pronostico	temperatura	humedad	viento	se_juega
1	soleado	calor	alta	debil	no
2	soleado	calor	alta	fuerte	no
3	nublado	calor	alta	debil	si
4	lluvia	templado	alta	debil	si
5	lluvia	frio	normal	debil	si
6	lluvia	frio	normal	fuerte	no
7	nublado	frio	normal	fuerte	si
8	soleado	templado	alta	debil	no
9	soleado	frio	normal	debil	si
10	lluvia	templado	normal	debil	si
11	soleado	templado	normal	fuerte	si
12	nublado	templado	alta	fuerte	si
13	nublado	calor	normal	debil	si
14	lluvia	templado	alta	fuerte	no

Ejemplo: ¿Hay *fulbito*?



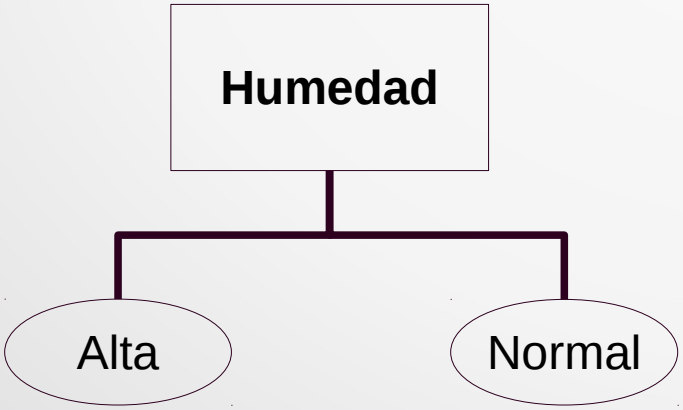
	no	si
lluvia	2	3
nublado	0	4
soleado	3	2

GI = 0.25



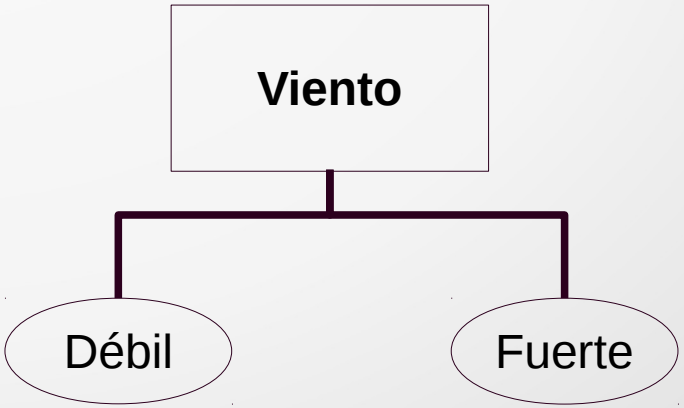
	no	si
calor	2	2
frio	1	3
templado	2	4

GI = 0.03



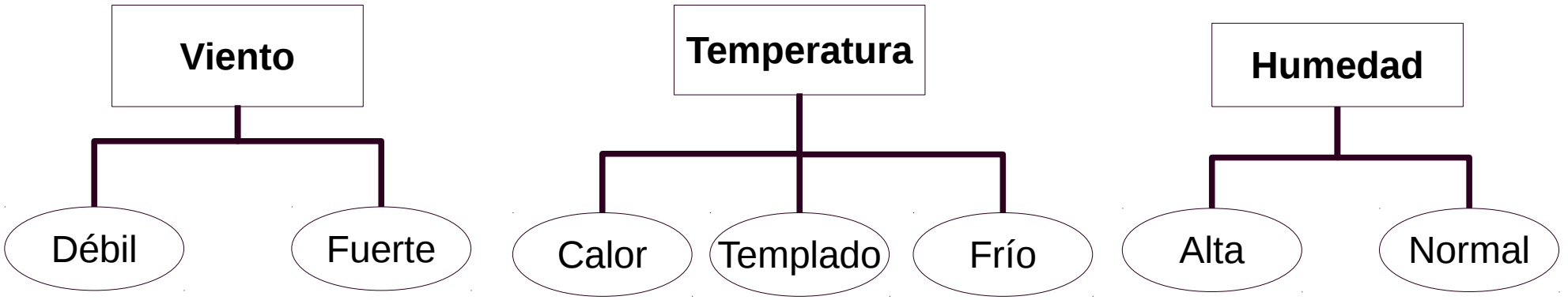
	no	si
alta	4	3
normal	1	6

GI = 0.15



	no	si
débil	2	6
fuerte	3	3

GI = 0.05



Soleado

	no	si
débil	2	1
fuerte	1	1

GI = 0.02

Soleado

	no	si
calor	2	0
frio	0	1
templado	1	1

GI = 0.57

Soleado

	no	si
alta	3	0
normal	0	2

GI = 0.97

Lluvia

	no	si
débil	0	3
fuerte	2	0

GI = 0.97

Lluvia

	no	si
calor	0	0
frio	1	1
templado	1	2

GI = 0.02

Lluvia

	no	si
alta	1	1
normal	1	2

GI = 0.02

Nublado

	no	si
débil	0	2
fuerte	0	2

GI = 0.0

Nublado

	no	si
calor	0	2
frio	0	1
templado	0	1

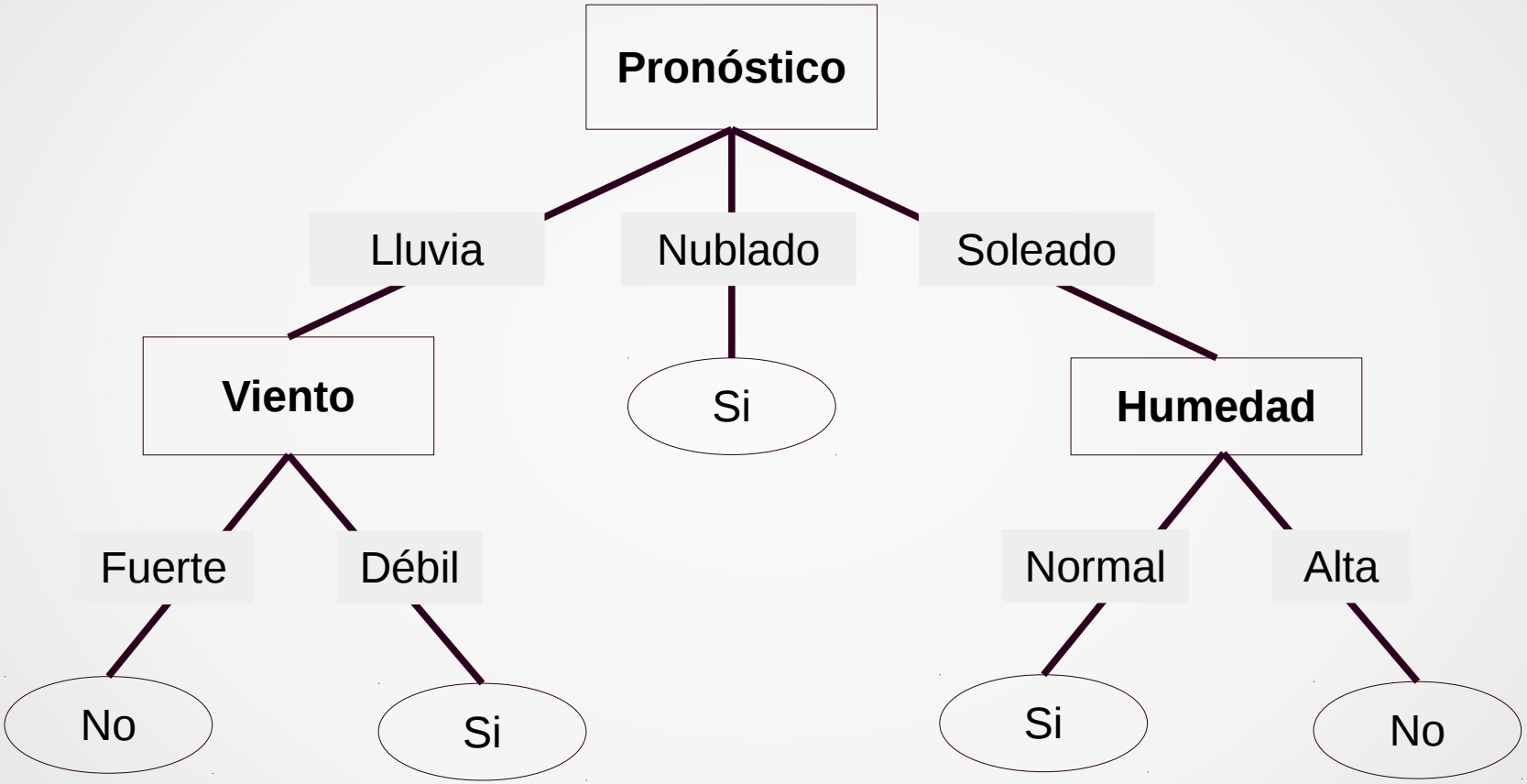
GI = 0.0

Nublado

	no	si
alta	0	2
normal	0	2

GI = 0.0

¿Se juega?


Pronóstico
Lluvia
Nublado
Soleado
Viento
Si
Humedad
Fuerte
Débil
Normal
Alta
No
Si
Si
No

Algoritmo C4.5

Sobreajuste

Dado un espacio de **hipótesis H**, una hipótesis **h ∈ H** diremos que sobreajusta **un conjunto de aprendizaje si existe otra hipótesis h' tal que h tiene menor error que h'** sobre el conjunto de aprendizaje, pero **h'** tiene **menor error que h** sobre la distribución total de instancias de aprendizaje.

C4.5 mejora algunas cuestiones de ID3

Utiliza un **conjunto de aprendizaje A** y un conjunto de **validación V**.

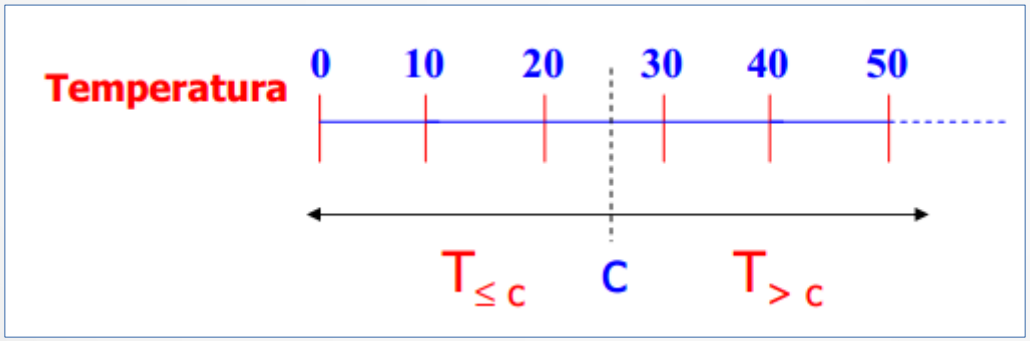
1. Inferir el árbol con el conjunto **A**
2. Establecer todas las posibles **podas del árbol**
3. Para cada poda **medir el error** respecto del conjunto **V**
4. **Ordenar los mejores resultados** y aplicarlos en la fase de test.

Algoritmo C4.5: Selección de atributos

Evaluación de atributos continuos

¿Cómo incorporar atributos continuos en las fases de aprendizaje y de test?

- Dada una variable x de carácter continuo, se establecen los intervalos adecuados en sus valores para proporcionar variables discretas



De igual forma que vimos en discretización supervizada se seleccionan aquellos valores que aportan mayor ganancia de información

Algoritmo C4.5: Selección de atributos

Problema: La medida GI favorece aquellas variables con mayor número de posibles valores.

Posible solución
Dados S (un conjunto de ejemplos de aprendizaje) y A (un atributo de los ejemplos que puede tomar c posibles valores) definimos:

$$SplitInformation(S,A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

SplitInformation(S,A) denota la entropía de S con respecto a los valores de A

$$RatiodeGanancia(S,A) \equiv \frac{Ganancia(S,A)}{SplitInformation(S,A)}$$

El **RatiodeGanacia(S,A)** favorece aquellos atributos que, en igualdad de Ganacia, separen los datos en menos clases.

Algoritmo C4.5: Manejo de ejemplos incompletos

Problema: Dado un conjunto de ejemplos ¿qué hacer cuando algunos atributos no tienen valor?

Posibles soluciones

- Estimar el valor desconocido como el **valor mayoritario** que aparece en el resto de ejemplos

- Asignar a cada posible valor una **probabilidad** (frecuencia relativa) de acuerdo con el resto de ejemplos. A continuación repartir el ejemplo en cada uno de sus valores de acuerdo con la probabilidad y hacer el cálculo de la Ganancia

En el caso de la clasificación, los casos con valores desconocidos se clasifican de acuerdo con **la mayor probabilidad que proporcione el árbol**.

Selección de modelos

En estadística y *machine learning*, la "**selección de modelos**" es el problema de escoger entre diferentes modelos matemáticos que pretenden describir el mismo conjunto de datos.

- Tenemos que seleccionar la “mejor” combinación de parámetros para nuestro algoritmo de aprendizaje.
- El objetivo es optimizar una **medida de desempeño** del algoritmo en un conjunto de datos independiente.

Medidas de *performace*

Las medidas de *performance* nos van a permitir evaluar de manera cuantitativa si uno de los modelos ajustados es mejor que otros.

En aprendizaje supervisado, para métodos de clasificación vamos a ver:

- Accuracy
- Curva ROC (*Receiver operating characteristic*)
- F-score,
- Coeficiente de Kappa

Y para predicciones:

- MSE (*Mean Squared Error*)

Accuracy

Accuracy de un clasificador M, **acc(M)**: Es el porcentaje de tuplas del conjunto de prueba que fueron correctamente clasificadas por el modelo M

- Error rate (tasa de mal clasificados) es: **M = 1 – acc(M)**
- Dadas m clases, $\boldsymbol{CM_{i,j}}$, una entrada en la **matriz de confusión**, indica el # de tuplas en la clase *i* que son etiquetadas por el clasificador como clase *j*

		Predicted class	
		C_1	C_2
Actual class	C_1	true positives	false negatives
	C_2	false positives	true negatives

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

Alternativas *Accuracy*

Existen alternativas o derivaciones de *Accuracy*

$$Sensibilidad = \frac{True\ POS}{POS}$$

$$Especificidad = \frac{True\ \mathrm{NEG}}{\mathrm{NEG}}$$

$$Precisión = \frac{True\ \mathrm{POS}}{(True\ POS + False\ POS)}$$

$$Recall = \frac{True\ \mathrm{POS}}{(True\ \mathrm{POS} + True\ \mathrm{NEG})}$$

Accuracy en términos de Sensibilidad y Especificidad

$$Accuracy = \frac{Sensibilidad * \frac{POS}{POS + \mathrm{NEG}}}{Especificidad * \frac{\mathrm{NEG}}{POS + \mathrm{NEG}}}$$

F-Score

Es una medida de precisión muy utilizada en *Information Retrieval* & clasificación de documentos.

Se define a partir de la matriz de confusión como:

$$F_\beta = (1 + \beta^2) \frac{Precisión * Exhaustividad}{(\beta^2 * Precisión) + Exhaustividad}$$

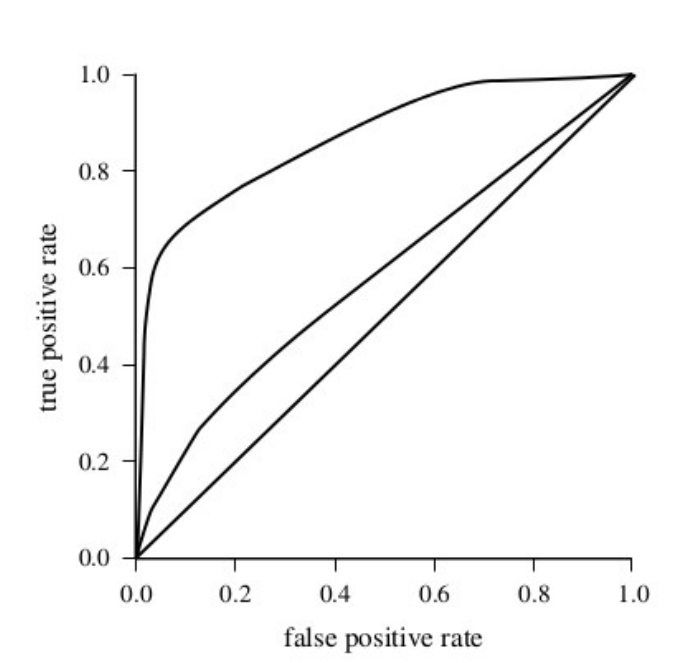
Permite ponderar los conceptos de precisión y exhaustividad presentes en la métrica a través de β.

Si β = 1, tienen no hay misma ponderación.

Curva ROC

- **Curva ROC** (*Receiver Operating Characteristics*): es una herramienta visual para comparar modelos que ajustan clases binarias.
- Se originó a partir de la teoría de detección de señales
- Muestra el ***trade-off*** entre la tasa de verdaderos positivos (**VP**) y la tasa de falsos positivos (**FP**).
- El **área bajo la curva ROC** (**AUC**) es una medida de la precisión del modelo.

- Cuanto más cerca de la línea diagonal (es decir, cuanto más cerca de la zona es 0.5), **menos preciso es el modelo**



http://www.bioestadistica.uma.es/analisis/roc1/

Coeficiente Kappa

Resulta interesante dado que **cuantifica el azar** con respecto a la coincidencia observada (predicción & clase real), el cual simboliza como **Pr(e)**.

$$K = \frac{Accuracy - \Pr(e)}{1 - \Pr(e)}$$

Donde Pr(e):

Pr(SI al azar) = (PosPredicted/Total) $*$ (PosClase/Total)

Pr(NO al azar) = (NegPredicted/Total) $*$ (NegClase/Total)

Pr(e) = Pr(Si al azar) + Pr(No al azar)

Por lo tanto, Pr(e) cuantifica la probabilidad de la coincidencia por azar.

Referencias

- Wiley Series on Methods and Applications in Data Mining. Larose. Cap. 6 PREPARING TO MODEL THE DATA
- Mitchell, T. M. (1997). Machine learning. Artificial Intelligence. Cap 3