Sistemas Inteligentes

Introducción al Aprendizaje Automático

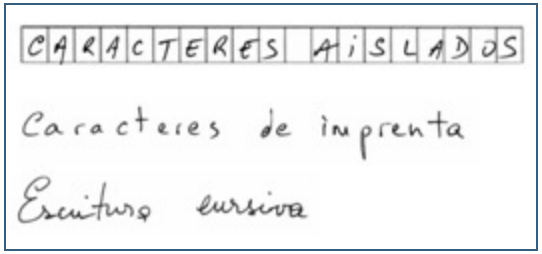
Tema 8

Objetivos

- Conocer en qué consiste el Aprendizaje Automático y el papel que juega en el contexto de los Sistemas Inteligentes
- Conocer las principales técnicas básicas de aprendizaje a partir de ejemplos y los algoritmos asociados
- Saber utilizar algunas herramientas para resolver problemas de aprendizaje y ser capaz de evaluar los resultados

Introducción

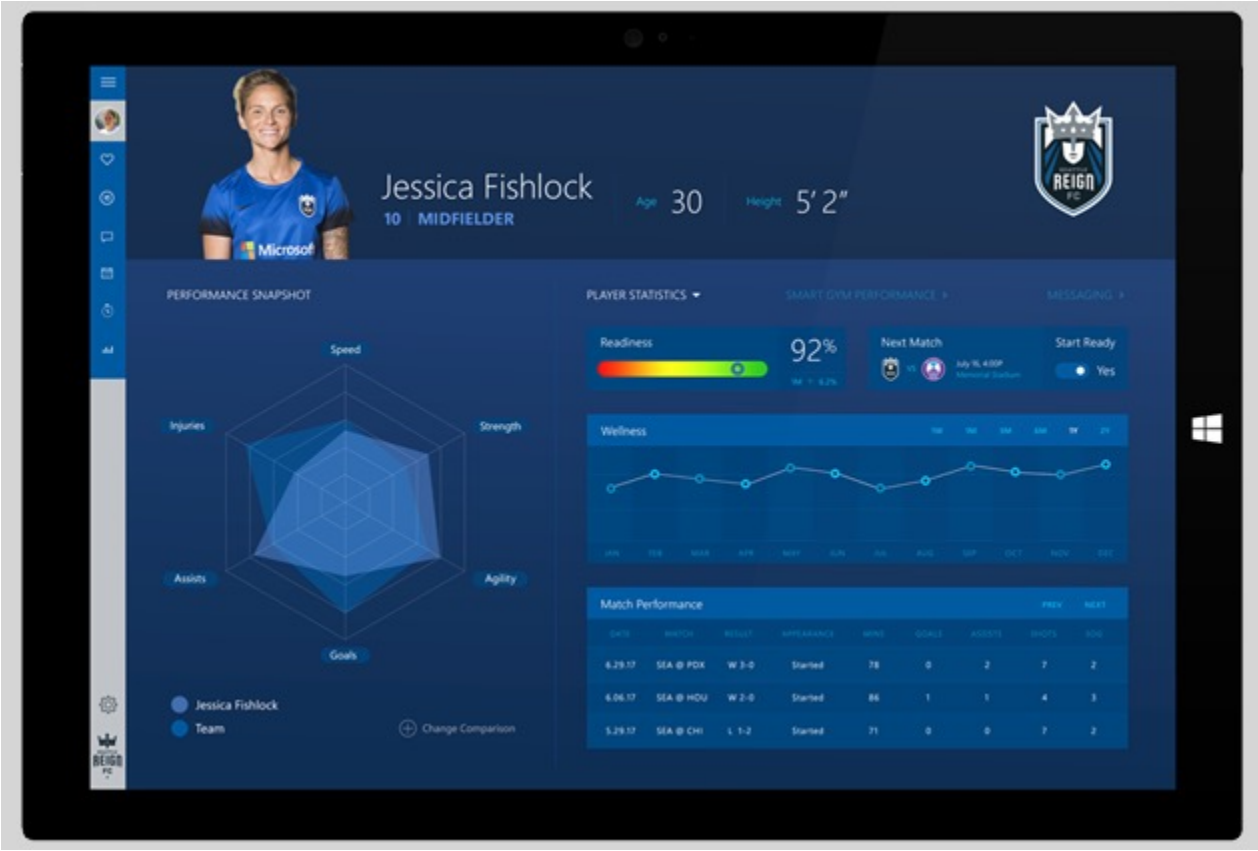
- En el diseño de SI, el Aprendizaje Automático es importante por varias razones
- No siempre es posible programar un sistema para que realice una tarea que requiere inteligencia.
- Puede ser más eficiente extraer el conocimiento de datos.



Toyota and Audi will demonstrate autonomous-driving

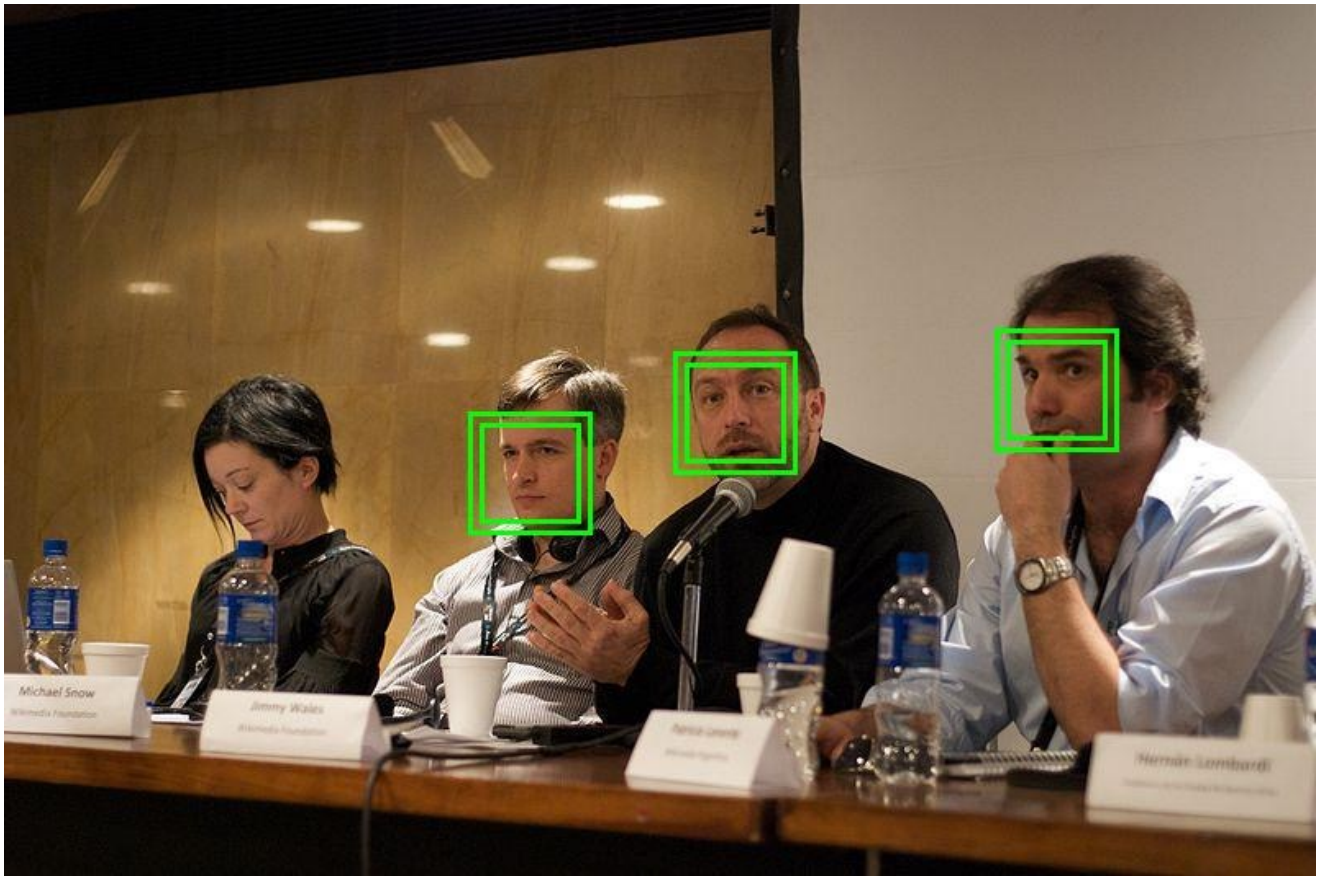
Algunos usos

- Sports Performance Platform de Microsoft



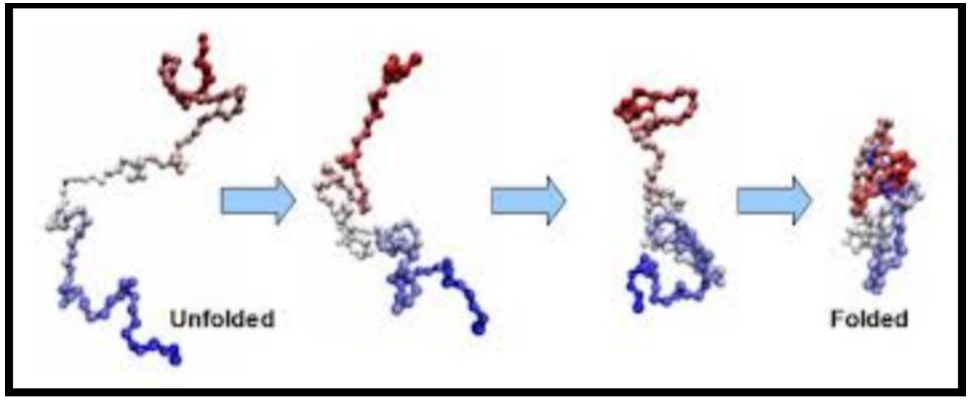
Algunos usos

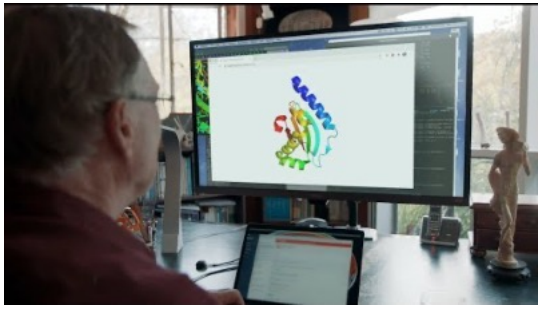
- Sistemas de identificación



Algunos usos

Algunos usos



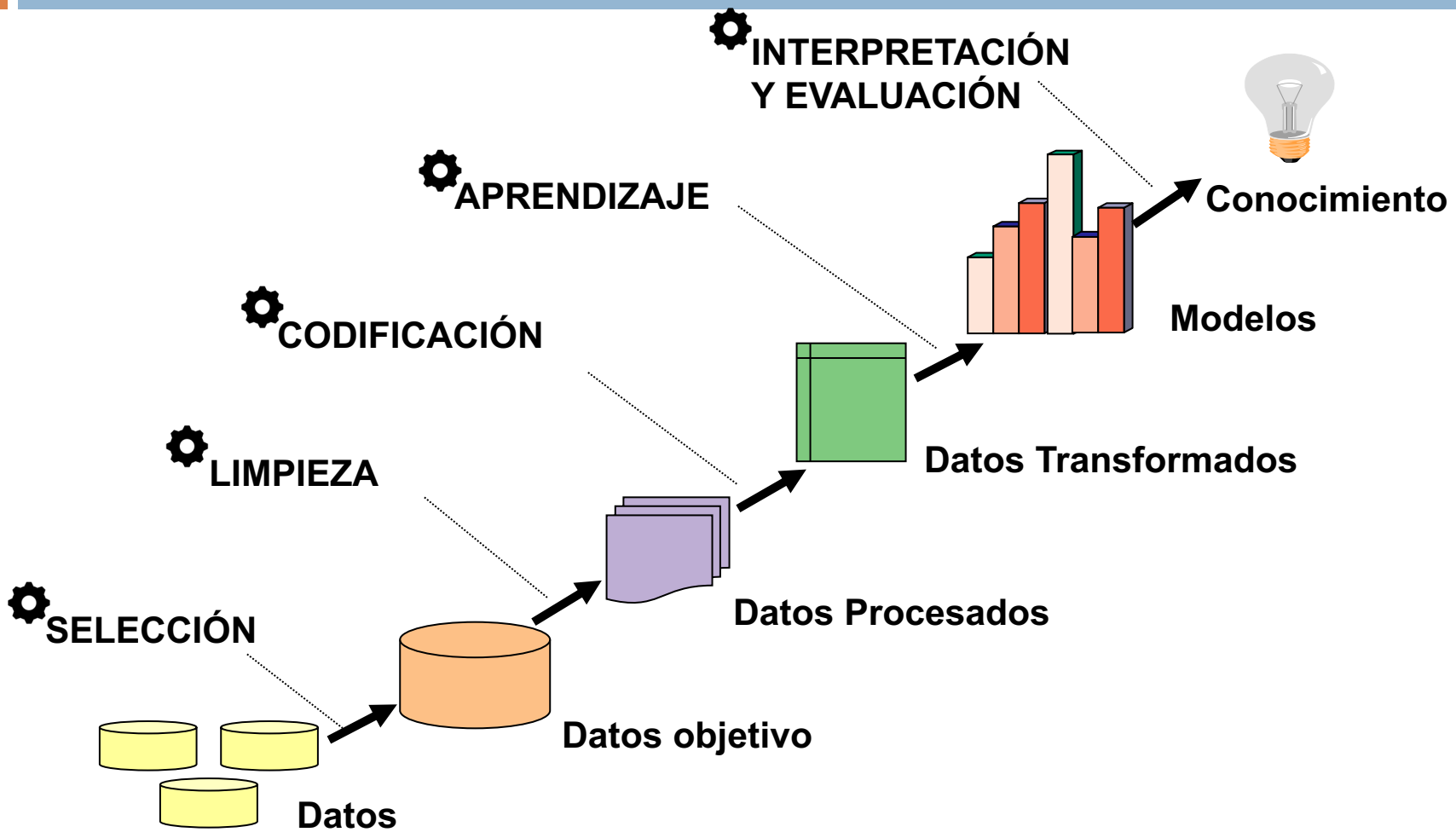


Actualidad

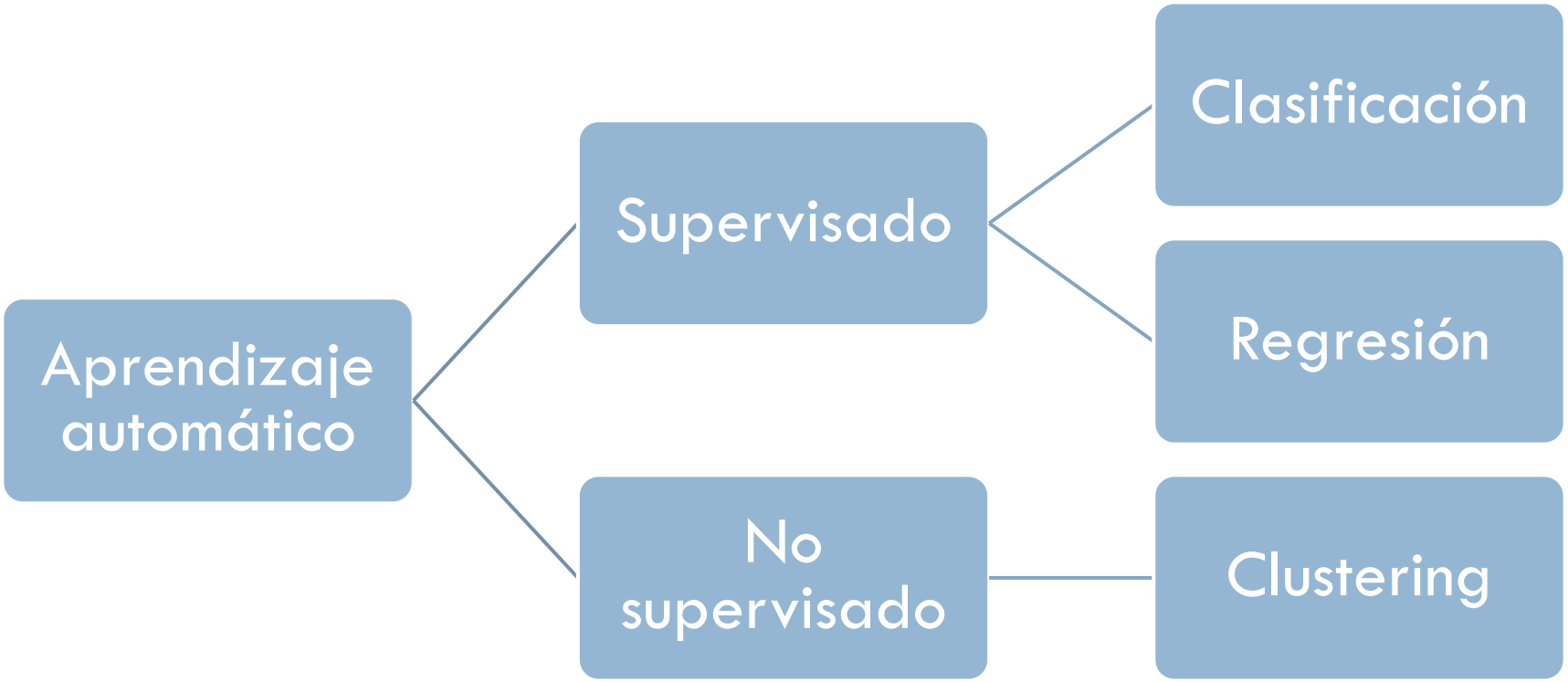
Algunos usos

- Otros ejemplos:
  - Sistema de recomendación
  - Detección de fraude
  - Diagnóstico de enfermedades
  - Categorización de imágenes
  - Predicción del consumo de electricidad
- Pero con datos e imaginación/necesidad cualquier otra aplicación

Proceso de extracción del conocimiento de datos

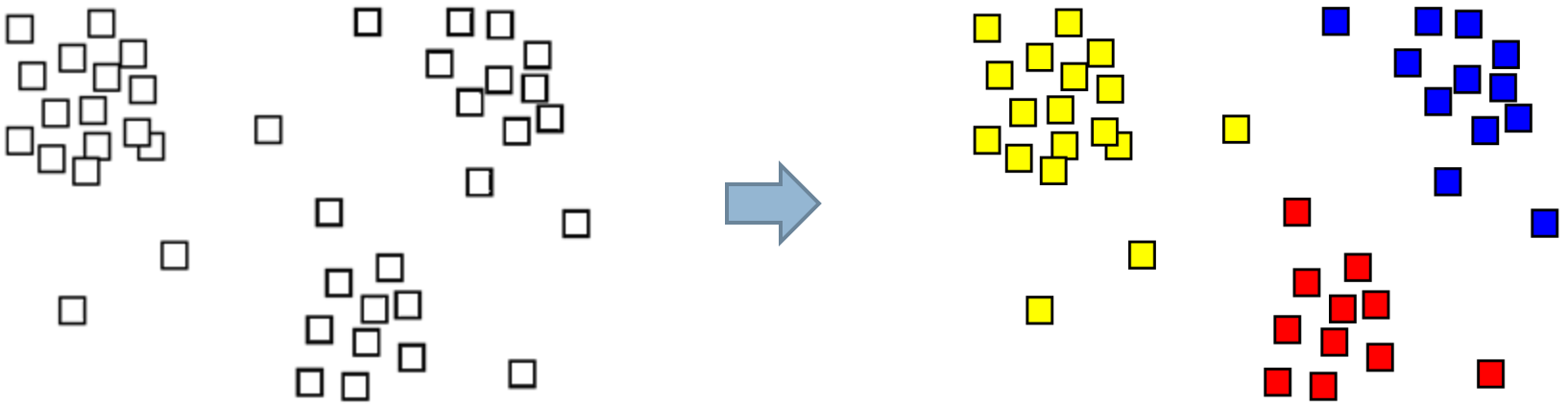


Taxonomía


Aprendizaje automático
Supervisado
No supervisado
Clasificación
Regresión
Clustering

Aprendizaje no supervisado

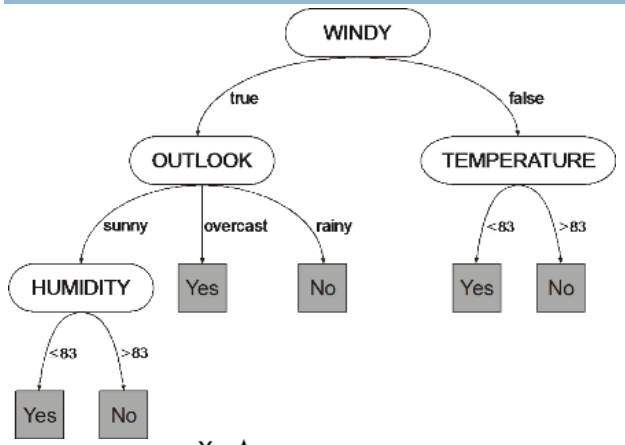
- Tenemos información que *a priori* no sabemos si está estructurada o no
- En esta categoría una de las tareas más comunes es el clustering o agrupamiento
- Se trata de dividir los datos en grupos

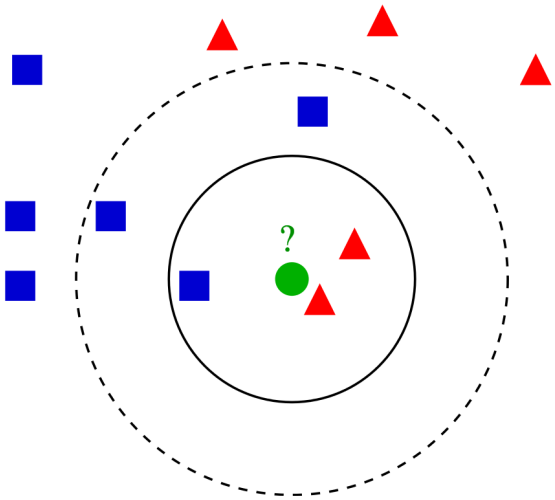


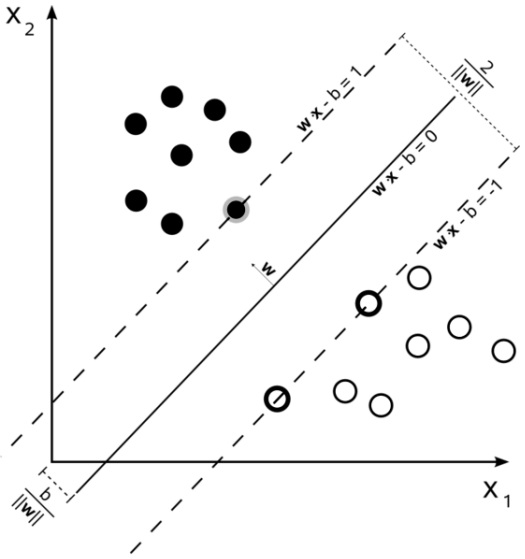
Aprendizaje supervisado

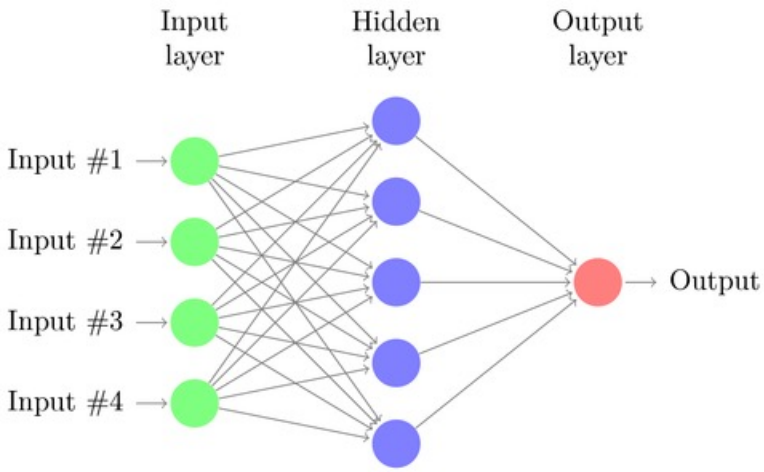
- En este caso la información de entrada tiene asociada una **variable objetivo**
- La tarea consiste en generar un modelo para predecir esta variable de salida
- Si la variable objetivo es **categórica: clasificación**
- Si la variable objetivo es **numérica: regresión**
- Nos centraremos en clasificación

Distintos paradigmas









Definición del problema

- Tenemos una tabla de datos con D ejemplos y N variables predictoras

$$(x_{11}, x_{12}, \ldots, x_{1n}, y_1)$$
$$(x_{21}, x_{22}, \ldots, x_{2n}, y_2)$$
$$\ldots$$
$$(x_{d1}, x_{d2}, \ldots, x_{dn}, y_d)$$

- **Hipótesis**: Existe una función f (desconocida)

$$y_j = f(\boldsymbol{x}_j)$$

  - x_j es un vector de valores $(x_{j1},\ldots, x_{jn})$ para los atributos de entrada $(X_{j1},\ldots, X_{jn})$, y_j es el valor del objetivo

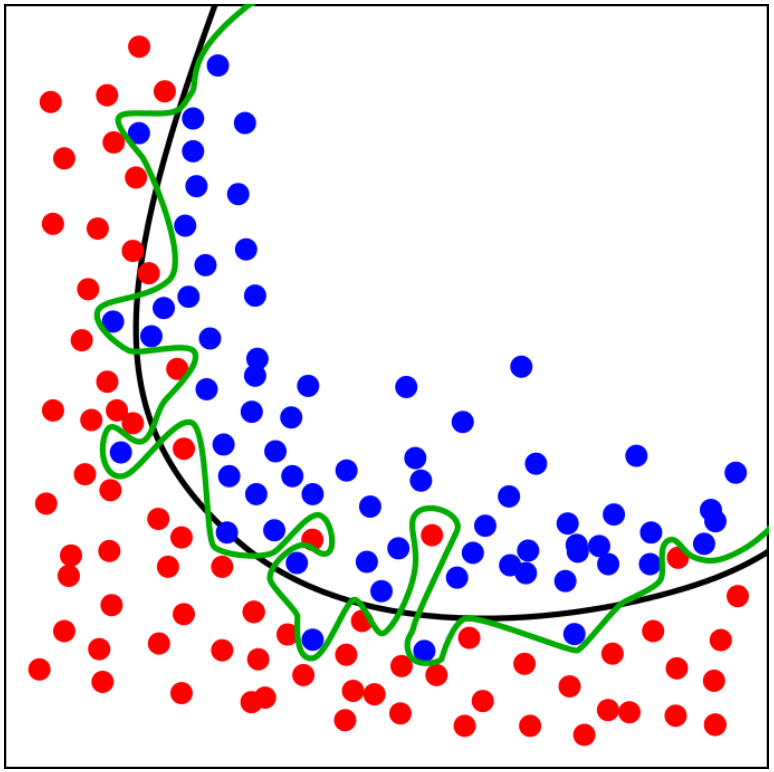
- Se trata de encontrar una función h que aproxime de la mejor forma posible a f

¿Qué solución es buena?

- Se trata de encontrar
$$h^* = argmax_{\{h \in H\}}[P(h|Datos)]$$
- Pero puede haber múltiples funciones consistentes con los datos
- En este caso se optará por la **solución de menos complejidad** (navaja de Ockham)

Sobreajuste

- Cuando el modelo ajusta perfectamente (o casi) los datos usados, pero falla con datos nuevos
- Puede ser debido a **ruido**
- La mayoría de los algoritmos tienen estrategias para evitarlo
- El problema es detectarlo en una etapa temprana



A continuación

- Evaluación de clasificadores
- Paradigmas
  - Algoritmos perezosos
  - Árboles de decisión
  - Redes neuronales artificiales
  - ~~Máquinas de vector soporte~~

Sistemas Inteligentes

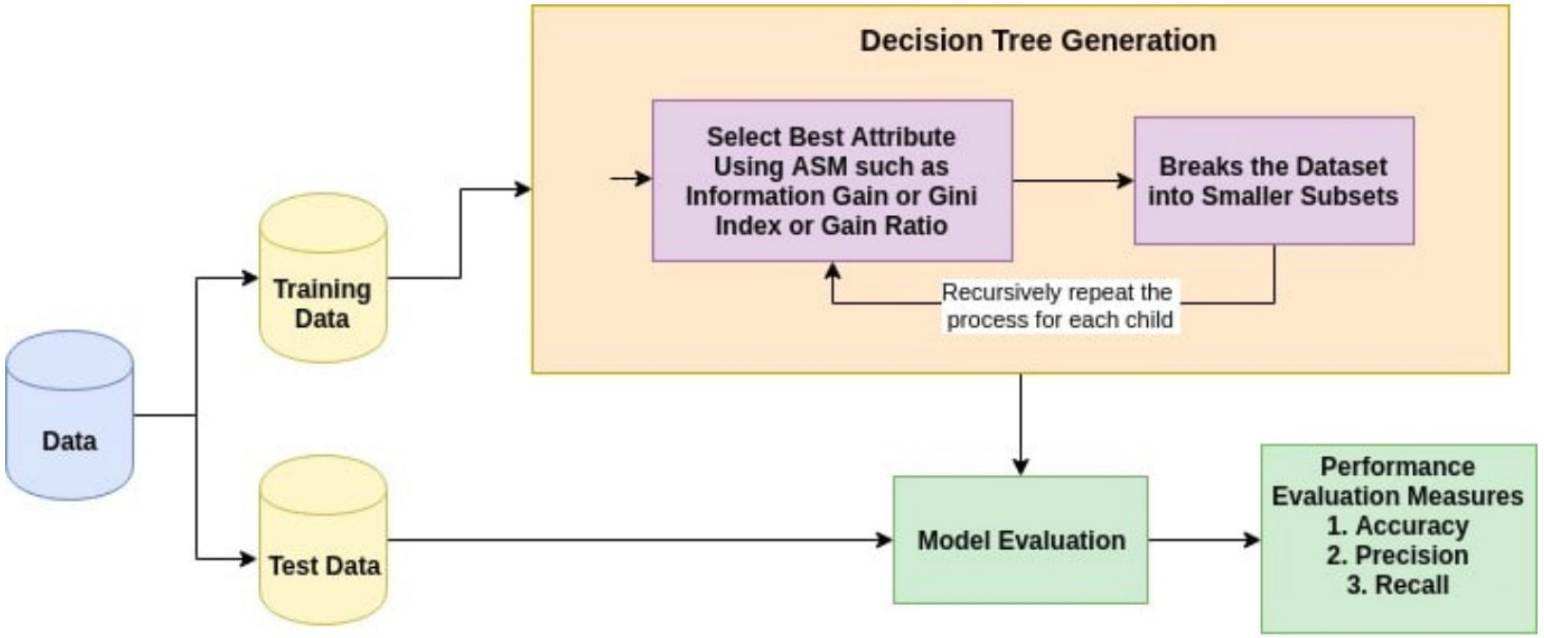
Evaluación y Comparación de Clasificadores

Tema 8

Validación

- Queremos modelos que ajusten los datos, pero con capacidad de **generalización**
- ¿Cómo podemos asegurar que un modelo se comportará bien ante datos futuros (no vistos)?
- Vamos a ocultar parte de los datos de entrada
- El algoritmo solo verá la parte que llamaremos de entrenamiento
- El resto será el conjunto de validación

Validación

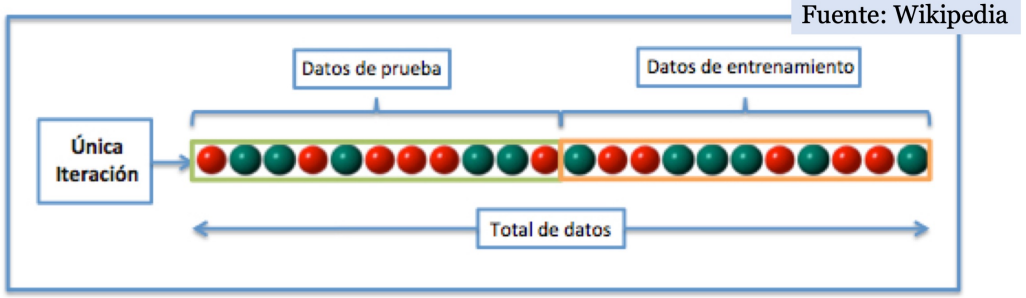


Tomado de https://rafneta.github.io/CienciaDatosPythonCIDE/Laboratorios/Lab6/Arboles.html

División entrenamiento/test

- Hacer una única división de los datos
- Por ejemplo 80% para entrenar y 20% para validar
- Está bien para **grandes cantidades de datos**
- Con cantidades pequeñas habrá un sesgo en la estimación

Fuente: Wikipedia

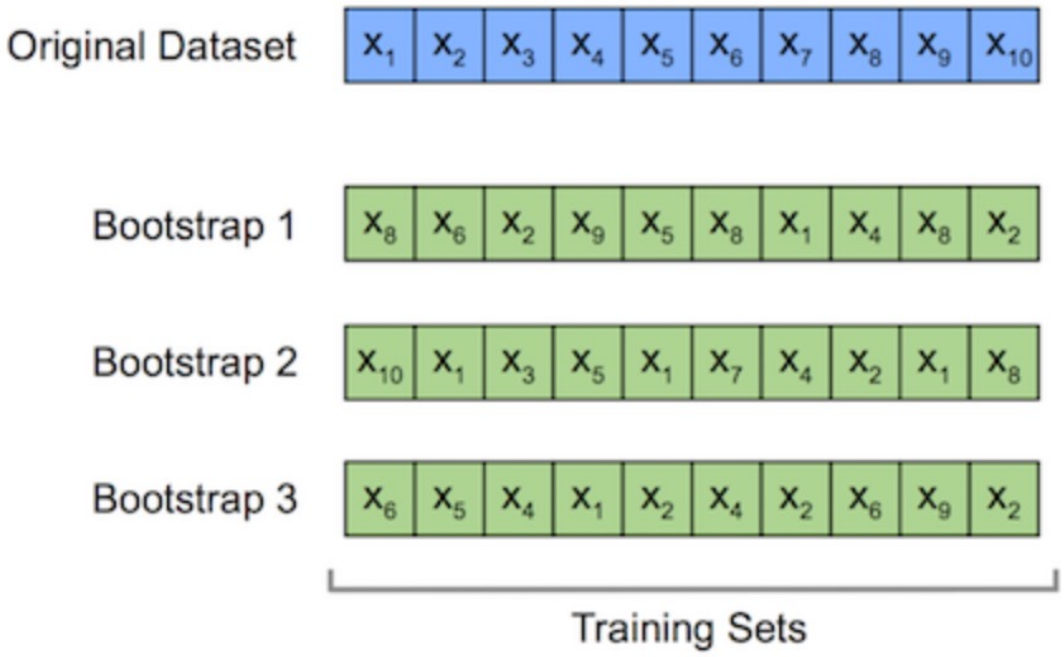


Bootstrap

- Como antes, hay dos conjuntos
- El de entrenamiento se obtiene muestreando con reemplazo *N* instancias (≈63,2%)
- Para validación se usan todos los datos
- Da una estimación optimista, pero es muy adecuado para **conjuntos de datos pequeños**
- Se puede reducir el sesgo con repeticiones, validando solo con las instancias no muestreadas o ponderando las instancias según el grupo

Bootstrap

- El de entrenamiento se obtiene muestreando con reemplazo *N* instancias (≈63,2%)



$$p(\text{no ser elegido}) = \left(1 - \frac{1}{n}\right)^n$$

0.368

Validación cruzada

Consiste en dividir las instancias en *k* grupos

El aprendizaje se hará *k* veces con *k*-1 de los grupos, validando con el grupo que queda

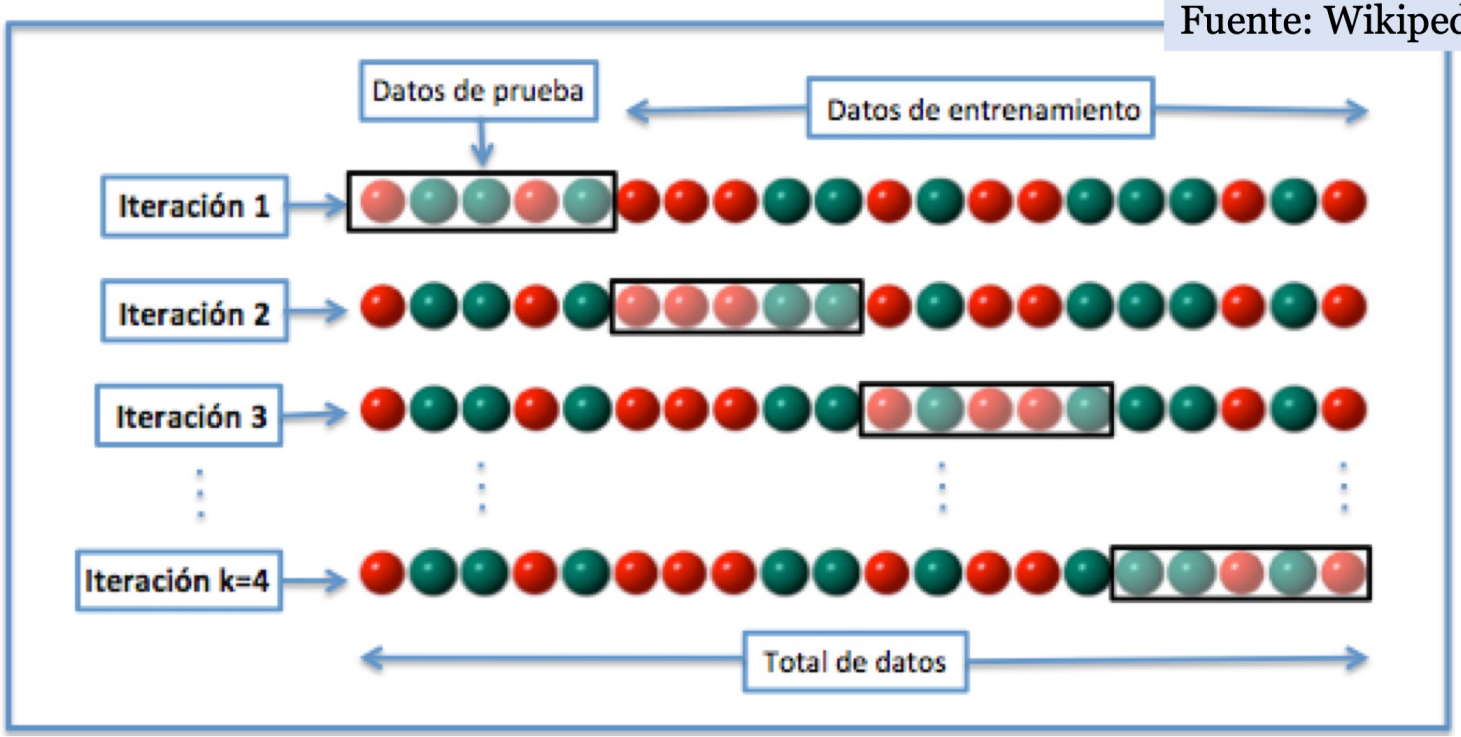
Se puede usar para conjuntos de datos

- Grandes: *k* será pequeño (suele ser 10)
- Pequeños: *k* será grande (hasta *k*=*D*-1, leave-one-out)

También se puede repetir múltiples veces con divisiones distintas en cada caso

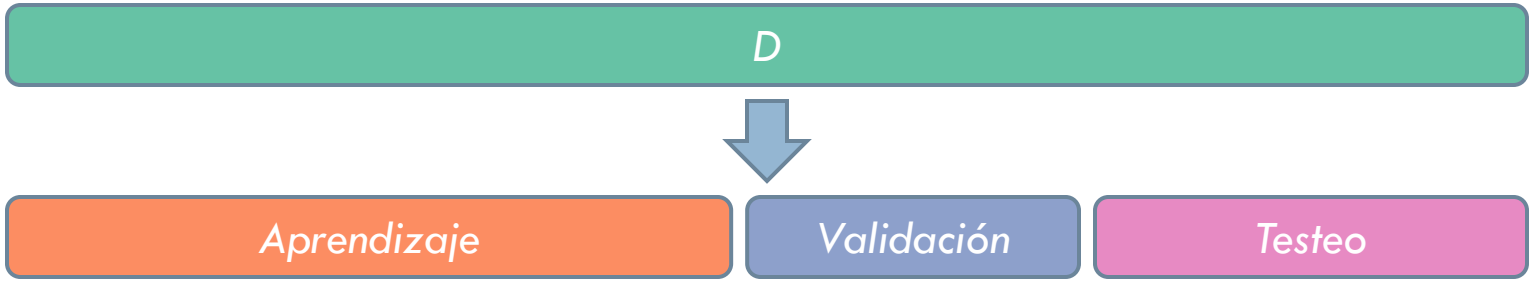
Validación cruzada

Fuente: Wikipedia



Meta-validación

- ¿Que hacer si las funciones *h* que estamos valorando requieren algún parámetro?
- Probamos varios valores y nos quedamos con el mejor
- Pero ¿será ese valor el más general?
- Para responder necesitamos todavía más datos no vistos durante el aprendizaje



Medidas de evaluación

- ¿Cómo cuantificar la bondad de un modelo?
- Matriz de confusión (caso binario)
  - $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
  - $Cobertura/Recall = Sensitivity = TPR = \frac{TP}{TP+FN}$
  - Especificidad$/Specificity = \frac{TN}{TN+FP}$
  - $Precision = \frac{TP}{TP+FP}$
  - $FPR = \frac{FP}{TN+FP}$

		Real	
		C_P	**C_N**
Predicción	**C_P**	**TP**: True positive	**FP**: False positive
	C_N	**FN**: False negative	**TN**: True negative

Problema de balanceado

- Predicción de una enfermedad (100000 personas)

		Real	
Pred	c_1	Pos	Neg
	Pos	300	500
	Neg	200	99000

		Real	
	c_2	Pos	Neg
	Pos	0	0
	Neg	500	99500

		Real	
	c_3	Pos	Neg
	Pos	400	5400
	Neg	100	94100

Acc.	0.993	0.995	0.945
Sens.	0.600	0.000	0.800
Spec.	0.995	1.000	0.946
Prec	0.375	0.000	0.068

- ¿Qué clasificador escogemos?

El valor kappa

- Indica si el clasificador es mejor que una predicción basada en la frecuencia de las clases

$$\kappa = \frac{p_o - p_e}{1 - p_e}, p_o = Acc, p_e = \frac{1}{D^2}\sum_k n_{k_{real}} n_{k_{pred}}$$

		Real	
Pred	c_1	Pos	Neg
	Pos	300	500
	Neg	200	99000

		Real	
	c_2	Pos	Neg
	Pos	0	0
	Neg	500	99500

		Real	
	c_3	Pos	Neg
	Pos	400	5400
	Neg	100	94100

Kappa	0.458	0.000	0.119

$$p_e = \frac{1}{100000^2}(500 * 800 + 99500 * 99200) = 0.98708$$

$$k = \frac{0.993 - 0.987}{1 - 0.987} = \frac{0.00592}{0.01292} = 0.458$$

El valor kappa

- Indica si el clasificador es mejor que una predicción basada en la frecuencia de las clases

$$\kappa = \frac{p_o - p_e}{1 - p_e}, p_o = Acc, p_e = \frac{1}{D^2}\sum_k n_{k_{real}} n_{k_{pred}}$$

		Real	
Pred	c_1	Pos	Neg
	Pos	300	500
	Neg	200	99000

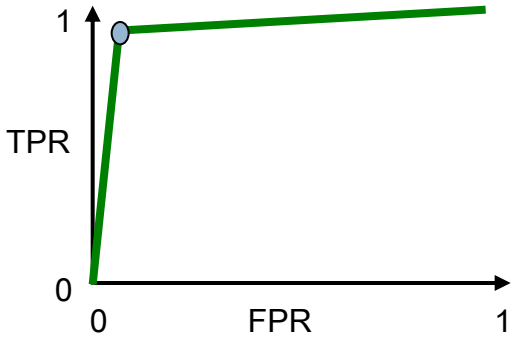
		Real	
	c_2	Pos	Neg
	Pos	0	0
	Neg	500	99500

		Real	
	c_3	Pos	Neg
	Pos	400	5400
	Neg	100	94100

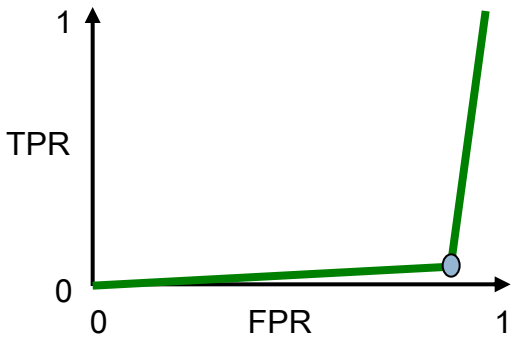
Acc.	0.993	0.995	0.945
Sens.	0.600	0.000	0.800
Spec.	0.995	1.000	0.946
Kappa	0.458	0.000	0.119

Curvas ROC

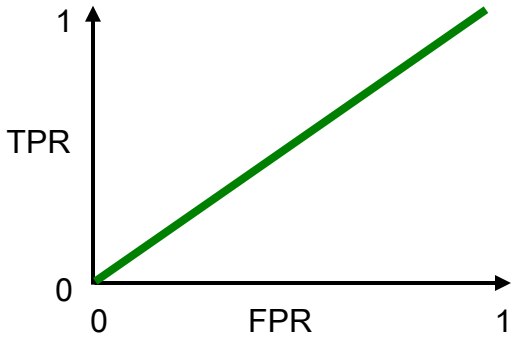
- Expresión visual de la calidad de un clasificador
- Compromiso entre sensitivity (TPR) y 1- specificity (FPR)
- Lo ideal es maximizar lo primero y minimizar lo segundo



Buen clasificador



Mal clasificador



Mal clasificador (aleatorio)

Algoritmo para la curva ROC

GenerateROCPoints(D, $score$)

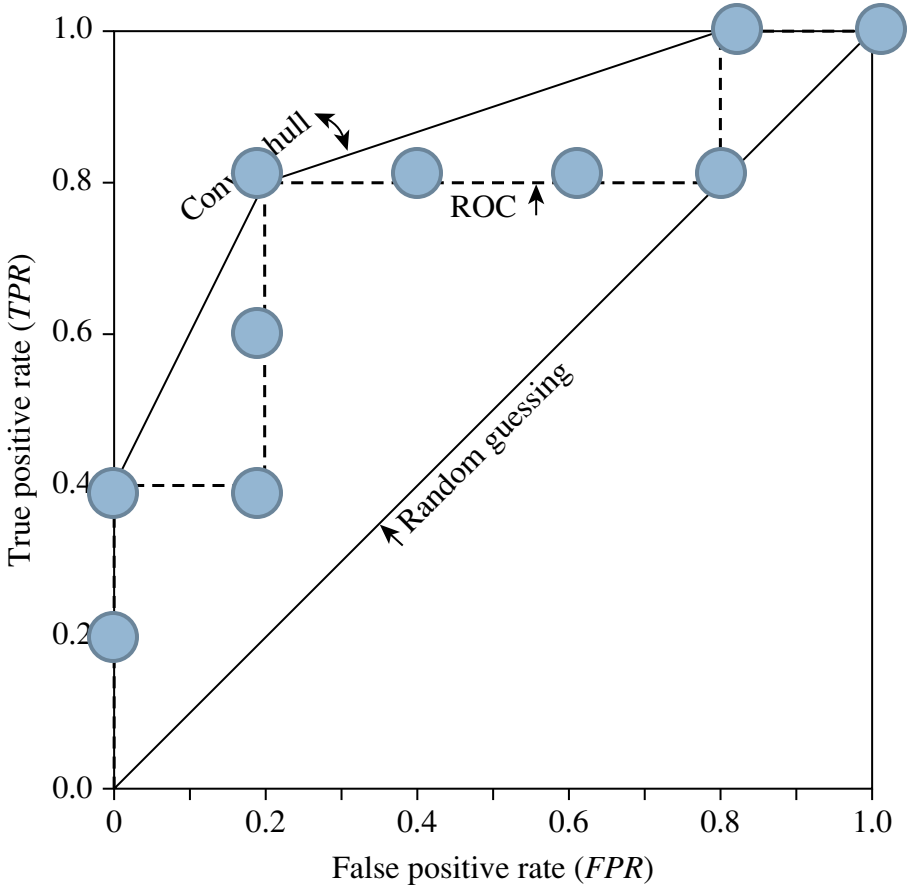
1. $D_{sorted} \leftarrow D$ ordenado decrecientemente por $score$
2. $FP, TP \leftarrow 0$
3. $score_{prev} \leftarrow -\infty$
4. $R \leftarrow \emptyset$
5. Para cada instancia i en D_{sorted}
6. Si $score[i] \neq score_{prev}$ entonces
7. Añade $\left(\frac{FP}{N}, \frac{TP}{P}\right)$ a R
8. $score_{prev} \leftarrow score[i]$
9. Si $D_{sorted}[i]$ es una instancia positiva entonces
10. $\mathrm{TP} \leftarrow TP + 1$
11. sino
12. $\mathrm{FP} \leftarrow FP + 1$
13. Añade $\left(\frac{FP}{N}, \frac{TP}{P}\right)$ a R

N=FP+TN
P=TP+FN

Area Under the Curve (AUC)

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
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	0	1	1.0	0.8
10	*N*	0.40	5	5	0	0	1.0	1.0

$$\left(\frac{FP}{FP+TN}, \frac{TP}{TP+FN}\right)$$

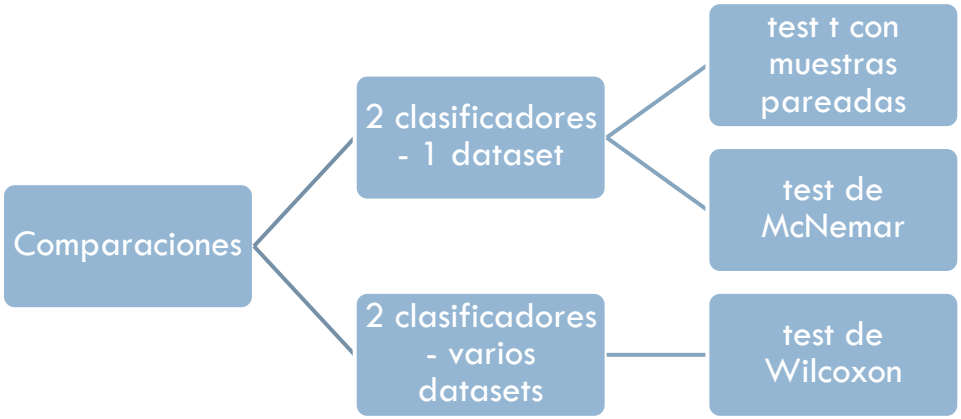


Teorema "No Free Lunch"

- Básicamente dice que, a priori, no hay un algoritmo que sea el mejor para cualquier problema
- Podemos ver que ciertos algoritmos suelen ser mejores para ciertos problemas
- En general, sin información/experiencia, suele ser bueno hacer una comparación entre distintas opciones para un problema dado

Comparación

- Tenemos varios clasificadores y su porcentaje de acierto u otra métrica
- ¿Elegimos simplemente el de mayor porcentaje?
- Por ejemplo:
  - Clasificador 1: 0.87
  - Clasificador 2: 0.91
  - Clasificador 3: 0.82



Contraste de hipótesis estadísticas

- Vamos a usar distintos tests estadísticos para comparar clasificadores
- Nos dan una valoración sobre la hipótesis de que los resultados son similares, el **p-valor**
- Además es necesario establecer un umbral para el que interpretaremos que hay diferencias significativas
- Normalmente este umbral es 0.05 (asumimos un 5% de fallo al señalar diferencias cuando los clasificadores son similares)

Test de la *t* pareado

- Usado cuando tenemos un **esquema de validación con repetición, por ejemplo validación cruzada**
- Explotar el hecho de tener varios resultados para cada clasificador (mismas condiciones)
- Este test nos dirá si el comportamiento de cada par de clasificadores muestra una tendencia similar
- Por ejemplo

Clasificador A	0.90	0.88	0.91	0.93	0.90	0.89	0.91	0.90	0.94	0.92	**0.91**
Clasificador B	0.86	0.90	0.87	0.88	0.95	0.90	0.89	0.92	0.88	0.93	**0.90**

- p-valor = 0.413

Test de la *t* pareado

HIPÓTESIS

H_0: $\mu_A - \mu_B = 0$ (La diferencia entre las medias pareadas de ambos grupos es igual a 0).

H_1: $\mu_A - \mu_B \neq 0$ (La diferencia entre las medias pareadas de ambos grupos es distinta de 0).

- Condiciones
  - Mismas muestras para cada grupo.
  - Aleatoriedad de la muestra
  - Normalidad o más de 30 muestras. En el caso de aplicar validación cruzada con k cajas, necesitaríamos un conjunto con k x 30. Comprobar la normalidad con:
    - Kolmogorov-Smirnov
    - Shapiro Wilk

Test de McNemar

- Se trata de construir una tabla 2x2 resumiendo los casos de acuerdo o desacuerdo entre clasificadores

Casos acertados por ambos	Acertados por 1 y no por 2
Acertados por 2 y no por 1	Fallados por ambos

- Se espera que los aciertos propios de cada uno sean similares
- Solo se puede utilizar cuando la suma de desacuerdos (b+c) es >25

a	b
c	d

62	20
10	8

310	100
50	40

H_0: $p_b = p_c$
H_1: $p_b != p_c$

p-valor=0.100

p-valor<10^{-4}

Ejemplo

- Índice kappa de C5.0 y RF en validación cruzada

```
metricas <- data.frame(C50 = c(0.9073259, 0.6822547, 0.6925111, 0.2719983,
                               0.9073259, 0.0925111, 0.0925111, 0.8572170,
                               0.9106929, 0.0925111),
                       Rf = c(0.6541556, 0.7521948, 0.8056707, 0.2816065,
                              0.9874889, -0.1236222, 0.3208222, 0.7521948,
                              0.8056707, 0.1236222))

metricas
```

```
##          C50         Rf
## 1  0.9073259  0.6541556
## 2  0.6822547  0.7521948
## 3  0.6925111  0.8056707
## 4  0.2719983  0.2816065
## 5  0.9073259  0.9874889
## 6  0.0925111 -0.1236222
## 7  0.0925111  0.3208222
## 8  0.8572170  0.7521948
## 9  0.9106929  0.8056707
## 10 0.0925111  0.1236222
```

Cuidado con los tests

Estos tests son una buena forma para ayudarnos a decidir con que modelo nos quedamos

Pero no son infalibles

No están diseñados específicamente para esta tarea

Existen otras alternativas más complejas

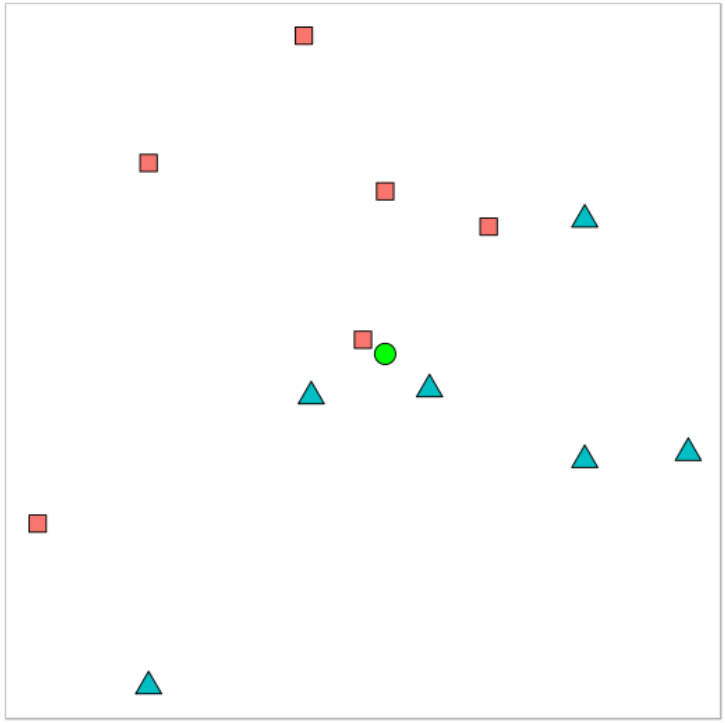
Sobre todo, cuidado cuando se hacen **muchas comparaciones** => sube la probabilidad de una decisión errónea

Sistemas Inteligentes

Algoritmos perezosos

kNN (k Nearest Neighbours)

- Es un método basado en instancias
- También llamado aprendizaje "perezoso"
- No hay modelo
- Se compara el ejemplo a predecir con los datos iniciales
- Requiere:
  - Datos de entrenamiento
  - Una medida de distancia
  - Parámetro *k*, número de vecinos



kNN

- El coste de aprendizaje es nulo
- Para clasificar una instancia
  - Se calcula su distancia a cada instancia de entrenamiento
  - Se eligen los *k* elementos más cercanos
  - Se utiliza el valor de la clase de los *k* vecinos para hacer la predicción
- Para este último paso hay varias estrategias
  - Asignar la clase mayoritaria
  - Ponderar por la distancia de cada vecino
    - Por ejemplo $w = \frac{1}{d^2}$

Cálculo de distancia

- Lo más común es usar la distancia euclídea

$$dist(\vec{x}, \vec{y}) = \sqrt{\sum_i (x_i - y_i)^2}$$

- Sin embargo esta distancia tiene un problema cuando cada componente tiene escalas distintas
  - Sueldo: 10K € a 1M €
  - Edad: 18 años a 100 años
  - Altura: 1.2, a 2.2m

Distancia euclídea

- Una variable con valores mucho mayores toma el mando

Edad	Sueldo	Préstamo	Distancia
25	40,000 €	N	50000
35	60,000 €	N	30000
45	80,000 €	N	10000
20	20,000 €	N	70000
35	120,000 €	N	30000
52	18,000 €	N	72000
23	95,000 €	Y	5000
40	62,000 €	Y	28000
60	100,000 €	Y	10000
48	220,000 €	Y	130000
23	150,000 €	Y	60000
48	90,000 €	?	

Distancia euclídea

- La solución a este problema es el escalado
- $x_{std} = \frac{x-\mu}{\sigma}$
- $x_{norm} = \frac{x-x_{min}}{x_{max}-x_{min}}$

Edad	Sueldo	Préstamo	Distancia
-0.89	-0.80	N	1.92
-0.14	-0.46	N	1.10
0.61	-0.13	N	0.28
-1.27	-1.13	N	2.40
-0.14	0.54	N	1.10
1.13	-1.16	N	1.24
-1.04	0.12	Y	1.88
0.23	-0.43	Y	0.76
1.73	0.21	Y	0.91
0.83	2.21	Y	2.17
-1.04	1.04	Y	2.12
0.83	0.04		

Otras distancias

- Existen otras formas de calcular distancias
- Distancia de Minkowski

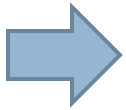
$$dist(\vec{x}, \vec{y}) = \left(\sum_i |x_i - y_i|^p \right)^{1/p}$$

- Para $p = 2$ tenemos la distancia euclídea
- Para $p = 1$ tenemos la distancia de Manhattan

Variables discretas

- Es posible usar la distancia de Hamming
- Cuenta el número de variables distintas
- También se pude convertir cada una en variables binarias (0, 1), tantas como valores menos 1
- Variable temperatura con estados Hot, Mild, Cool

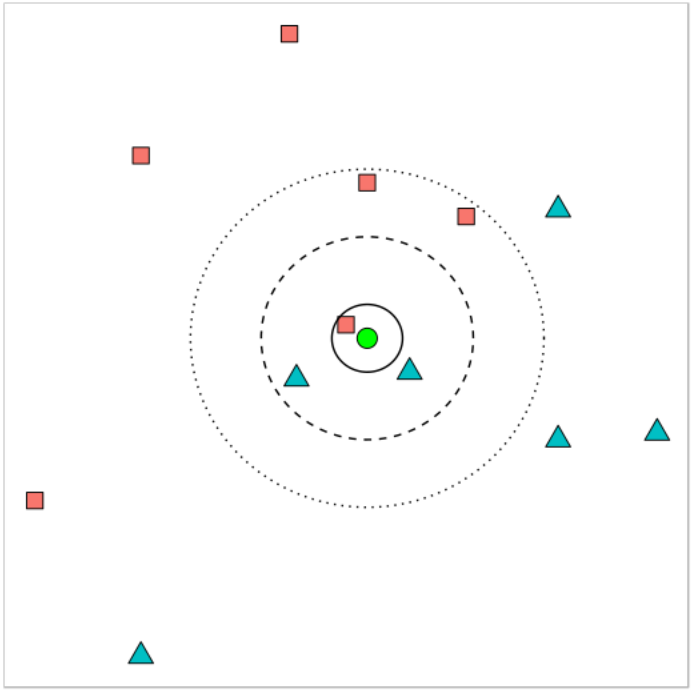
Temp.
Hot
Mild
Cool



Temp.Mild	Temp.Cool
0	0
1	0
0	1

Selección de parámetros

- Esencialmente hay que seleccionar el valor de *k*
- A mayor valor menos afecta el ruido
- También la función de distancia y de pesos
- Utilizaremos un proceso de validación para seleccionarlos



Ventajas y desventajas

Fortalezas	Debilidades
Aprenden funciones muy complejas	La clasificación puede ser lenta
No se pierde información	Los atributos irrelevantes son perjudiciales
	No es adecuado con muchos atributos

Bibliografía

- Tom Mitchell. *Machine Learning*. McGraw-Hill
- Ethem Alpaydin. *Introduction to machine learning*. The MIT Press
- Christopher Bishop. *Pattern recognition and machine learning*. Springer
- Stuart Russell, Peter Norvig. *Inteligencia Artificial: Un Enfoque Moderno*. Prentice Hall