Introducción al Aprendizaje Automático y a la Minería de Datos con Weka

HERRAMIENTAS DE LA INTELIGENCIA ARTIFICIAL

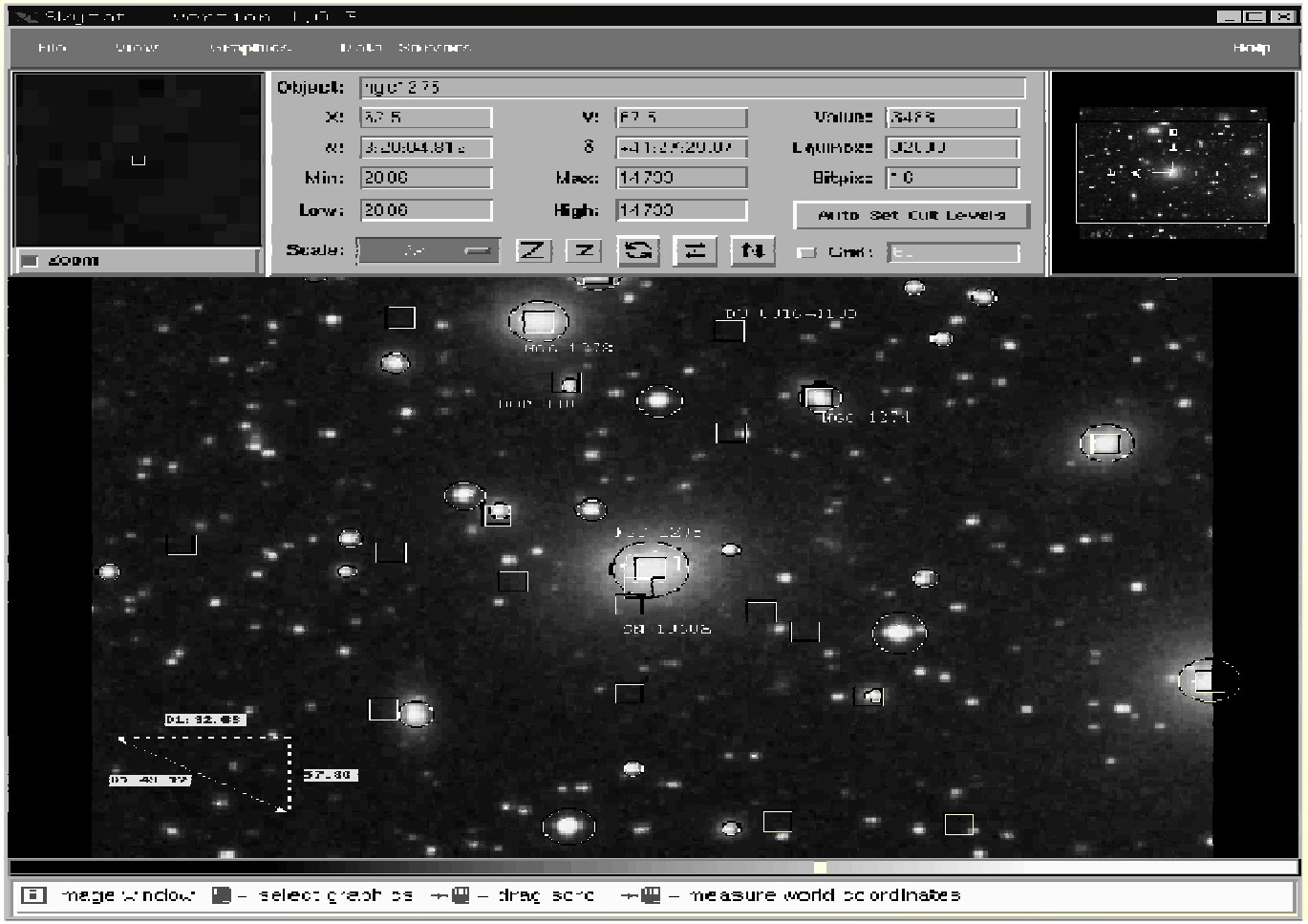
INGENIERÍA INFORMÁTICA

Índice

- MINERÍA DE DATOS
  - INTRODUCCIÓN A LA MINERÍA DE DATOS
  - TAREAS EN MINERÍA DE DATOS
  - FASES EN MINERÍA DE DATOS
  - TIPOS DE ALGORITMOS PARA PREDICCIÓN (CLASIFICACIÓN Y REGRESIÓN)
  - EVALUACIÓN DEL CONOCIMIENTO MINADO
  - SELECCIÓN DE ATRIBUTOS
  - MINERÍA DE TEXTOS

INTRODUCCIÓN A LA MINERÍA DE DATOS

SKYCAT: Clasificación automática de objetos del firmamento



Minería de Datos. Justificación

- **Nuevas posibilidades:** disponibilidad de grandes cantidades de datos (bancos, la web, tarjetas fidelización, ...), potencia de cómputo
- **Nuevas necesidades:** Es complicado analizar los datos de manera manual. Necesidad de técnicas automáticas: resúmenes (BBDD), inferencias (estadística, aprendizaje automático)

Minería de Datos. Objetivos

- Convertir datos en conocimiento para tomar decisiones
- Es importante la inteligibilidad del conocimiento obtenido (los modelos estadísticos no son siempre sencillos de entender)
- MD = BBDD + estadística + aprendizaje automático

TAREAS EN MINERÍA DE DATOS

Minería de Datos. Tareas

- Predicción:
  - Clasificación
  - Regresión
- Asociación
- Agrupación (clustering)

*Ejemplo1. Créditos bancarios (**clasificación**)*

- Un banco por Internet desea obtener reglas para predecir qué personas de las que solicitan un crédito no van a devolverlo.
- La entidad bancaria cuenta con una gran base de datos correspondientes a los créditos concedidos (o no) a otros clientes con anterioridad.

Ejemplo1. Datos (instancias, patrones, …)

IDC	Años	Euros	Salario	Casa propia	Cuentas morosas	…	Devuelve el crédito
101	15	60000	2200	Si	2	…	No
102	2	30000	3500	Si	0	…	Si
103	9	9000	1700	Si	1	…	No
104	15	18000	1900	No	0	…	Si
105	10	24000	2100	No	0	…	No
…	…	…	…	…	…		…

Ejemplo 1. Conocimiento obtenido

- **SI** (cuentas-morosas > 0) **ENTONCES** Devuelve-crédito = no
- **SI** (cuentas-morosas = 0) **Y** ((salario > 2500) **O** (años > 10)) **ENTONCES** devuelve-crédito = si

*Ejemplo 2. Determinar las ventas de un producto (**Regresión**)*

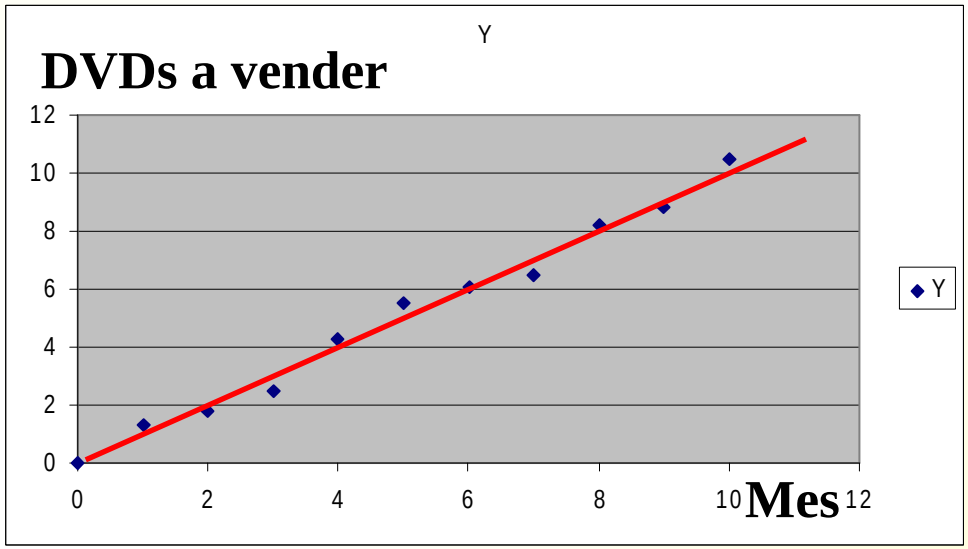
- Una gran cadena de tiendas de electrodomésticos desea optimizar el funcionamiento de su almacén manteniendo un stock de cada producto suficiente para poder servir rápidamente el material adquirido por sus clientes.

Ejemplo 2. Datos

Producto	Mes-12	...	Mes-4	Mes-3	Mes-2	Mes-1
Televisor plano	20	...	52	14	139	74
Video	11	...	43	32	26	59
Nevera	50	...	61	14	5	28
Microondas	3	...	21	27	1	49
Discman	14	...	27	2	25	12
...	...	...	...	...	...	...

Ejemplo 2. Conocimiento obtenido

- Modelo que prediga lo que se va a vender cada mes a partir de lo que se vendió en los meses anteriores (serie temporal)



*Ejemplo 3. Análisis de la cesta de la compra (**Asociación**)*

- Un supermercado quiere obtener información sobre el comportamiento de compra de sus clientes.
- Se piensa que de esta manera se puede mejorar el servicio, colocando ciertos productos juntos, etc.

Ejemplo 3. Datos de las cestas

Id	Huevos	Aceite	Pañales	Vino	Leche	Mantequilla	Salmón	Lechugas	...
1	Si	No	No	Si	No	Si	Si	Si	...
2	No	Si	No	No	Si	No	No	Si	...
3	No	No	Si	No	Si	No	No	No	...
4	No	Si	Si	No	Si	No	No	No	...
5	Si	Si	No	No	No	Si	No	Si	...
6	Si	No	No	Si	Si	Si	Si	No	...
7	No	No	No	No	No	No	No	No	...
8	Si	Si	Si	Si	Si	Si	Si	No	...
...	...	...	...	...	...	...	...	...	...

Ejemplo 3. Conocimiento obtenido

- Reglas **Si** At_1=a y At_2=b y ... **Entonces** At_n=c
  - **Si** pañales=si, **entonces** leche=si (100%, 37%)
- Las reglas también pueden ser:
  - **Si** At_1=a y At_2=b **Entonces** At_n=c, **At_4=D**
- (a,b) = (precisión, cobertura)
  - Precisión ("confidence"): veces que la regla es correcta
  - Cobertura ("support"): frecuencia de ocurrencia de la regla en los datos

Ejemplo 3. Precisión ("confidence")

- Sea la regla:
  - **Si** At_1=a y At_2=b **Entonces** At_n=c
- Precisión: Número de veces que At_n=c supuesto que At_1=a y At_2=b
- **Ejemplo:**
  - **Si** huevos=si, **entonces** aceite=si
  - Número de veces que huevos=si y aceite=si: 2
  - Número de veces que huevos=si: 4
  - Precisión (aciertos) de la regla: 2/4 = 50%

Ejemplo 3. Cobertura ("support")

- Una regla puede ser muy precisa (100%), pero ocurrir muy poco y ser poco relevante
- Ejemplo:
  - **Si** huevos=si y aceite=si y pañales=si **entonces** salmón=si
  - Precisión: 1/1 = 100%
  - Pero sólo ocurre para un cliente (el octavo): 1/8 = 12%
- Sea la regla:
  - **Si** At_1=a y At_2=b **Entonces** At_n=c
- **Cobertura**: número de datos que cumplen las dos condiciones At_1=a, At_2=b, dividido por número total de datos
- Representa el porcentaje de veces que ocurre la regla, lo útil que es

*Ejemplo 4. Agrupación de empleados (“**clustering**”)*

- El departamento de RRHH de una empresa desea categorizar a sus empleados en distintos grupos con el objetivo de entender mejor su comportamiento y tratarlos de manera adecuada

Ejemplo 4. Datos

Id	Sueldo	Casado	Coche	Hijos	Alq/Prop	Sindicado	Bajas	Antigüedad	Sexo
1	1000	Si	No	0	Alq	No	7	15	H
2	2000	No	Si	1	Alq	Si	3	3	M
3	1500	Si	Si	2	Prop	Si	5	10	H
4	3000	Si	Si	1	Alq	No	15	7	M
5	1000	Si	Si	0	Prop	Si	1	6	H
...	...	...	...	...		...	...	...	...

Ejemplo 4. Conocimiento obtenido

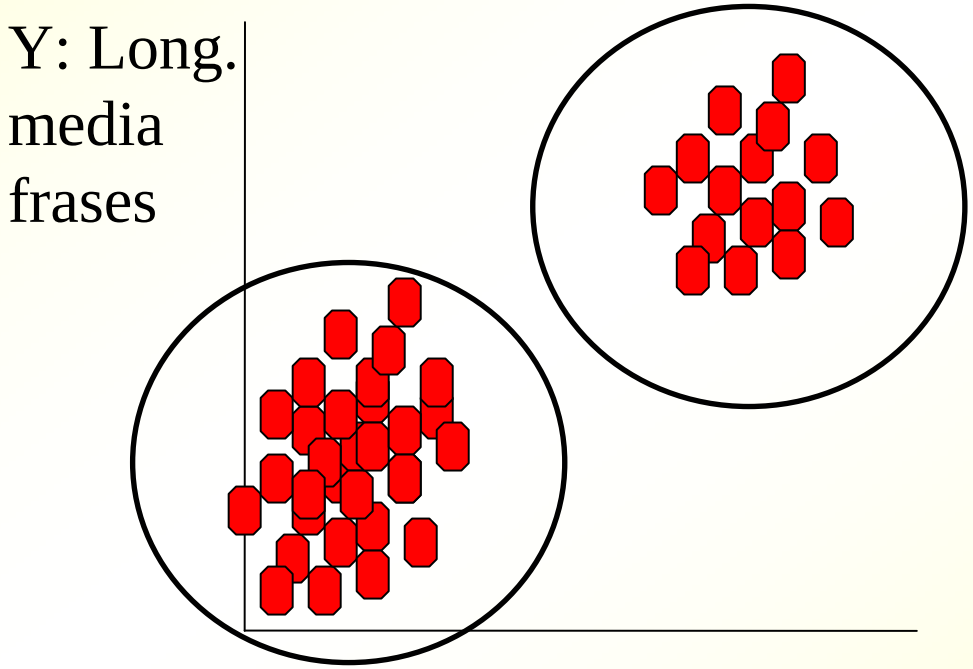
	GRUPO 1	GRUPO 2	GRUPO 3
Sueldo	1535	1428	1233
Casado (No/Si)	77%/22%	98%/2%	0%/100%
Coche	82%/18%	1%/99%	5%/95%
Hijos	0.05	0.3	2.3
Alq/Prop	99%/1%	75%/25%	17%/83%
Sindicado	80%/20%	0%/100%	67%/33%
Bajas	8.3	2.3	5.1
Antigüedad	8.7	8	8.1
Sexo (H/M)	61%/39%	25%/75%	83%/17%

Ejemplo 4. Conocimiento obtenido

- Grupo 1: sin hijos y con vivienda de alquiler. Poco sindicados. Muchas bajas
- Grupo 2: sin hijos y con coche. Muy sindicados. Pocas bajas. Normalmente son mujeres y viven en alquiler
- Grupo 3: con hijos, casados y con coche. Mayoritariamente hombres propietarios. Poco sindicados.

Idea general de agrupación

- Detectar agrupaciones naturales en los datos
- Agrupación (o “clustering”) = aprendizaje no supervisado: se parte de una tabla, como en clasificación, pero sin la clase



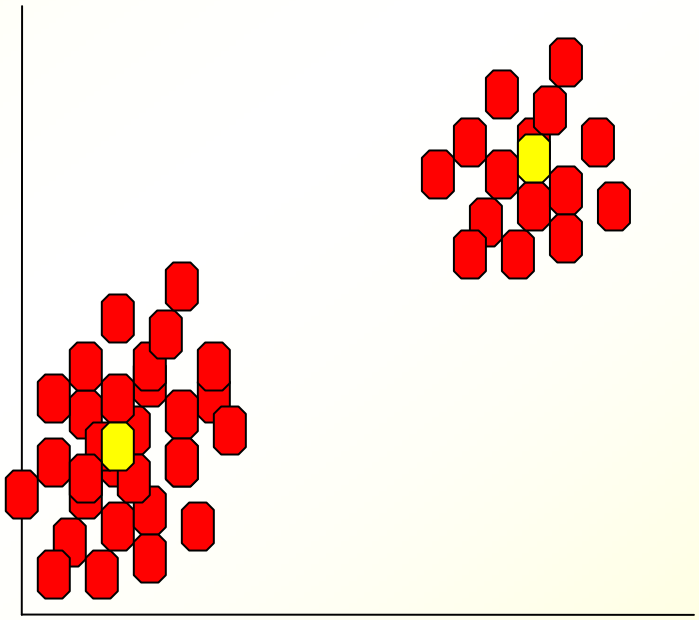
Ejemplo: clustering de libros. 2 grupos:

* Palabras y frases largas (¿filosofía?)

* Palabras y frases cortas (¿novela?)

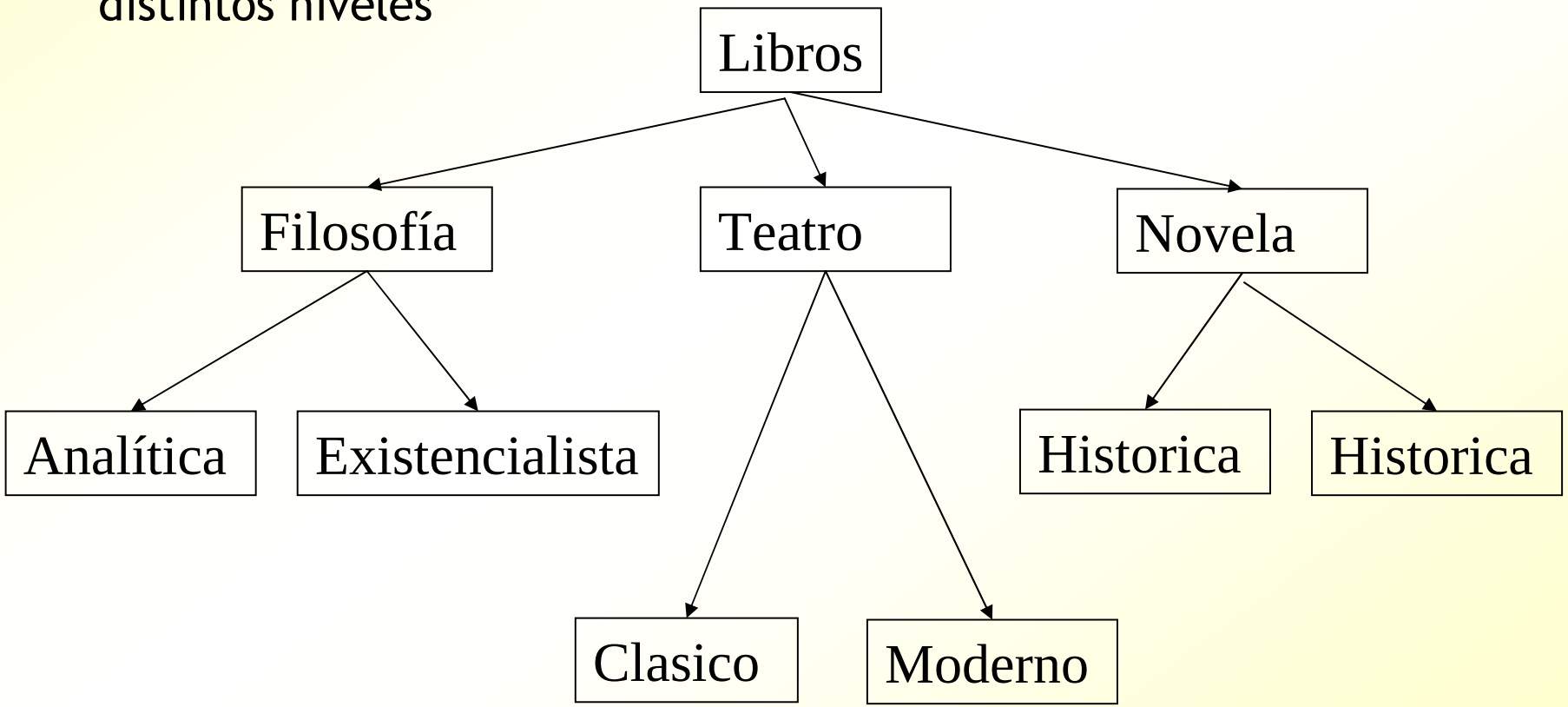
Representación de clusters

- Por sus centroides (ej: algoritmo k-medias)
- La pertenencia a un cluster puede ser probabilística (ej: algoritmo EM)



Representación de clusters

- Jerárquica (ej: algoritmo cobweb)
- Nota: las etiquetas “filosofía”, “clásico”, etc. aparecen sólo a título indicativo. El sistema simplemente detectaría distintos grupos a distintos niveles



Aplicaciones de Minería de Datos (técnica de carácter horizontal)

- Financieras y banca
  - Obtención de patrones de uso fraudulento de tarjetas de crédito
  - Predicción de morosidad (préstamos)
- Análisis de mercado:
  - Análisis de cesta de la compra
  - Segmentación de mercado
- Seguros y salud privada: determinación de clientes potencialmente caros
- Educación: detección de abandonos

Aplicaciones II

- Medicina: diagnóstico de enfermedades (ej: diagnóstico de dolor abdominal)
- Ciencia:
  - Predecir si un compuesto químico causa cáncer
  - Predecir si una persona puede tener potencialmente una enfermedad a partir de su DNA
  - Clasificación de cuerpos celestes (SKYCAT)

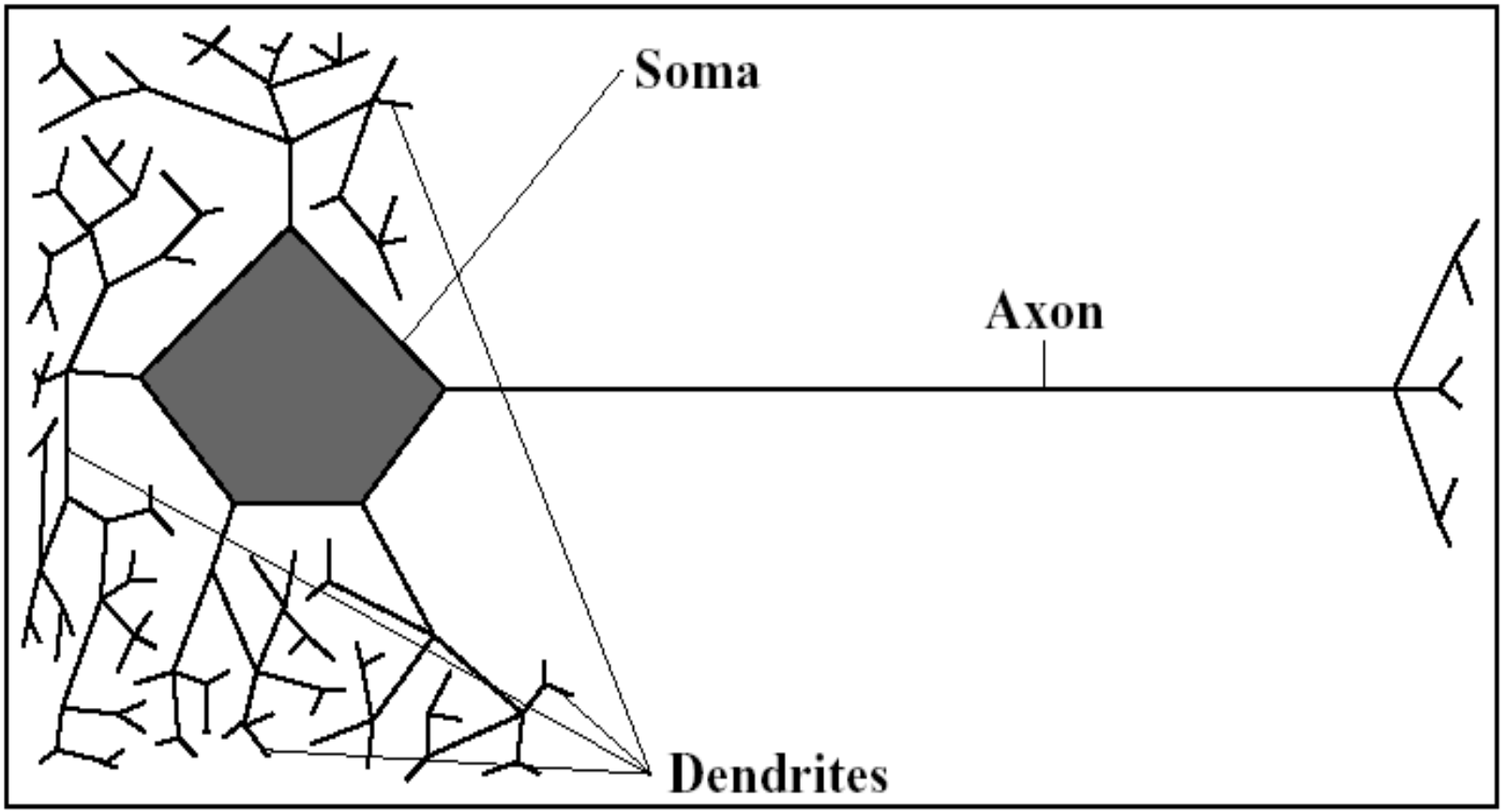
Aplicaciones III

- Detección de spam (SpamAssassin, bayesiano)
- Web: asociar libros que compran usuarios en e-tiendas (amazon.com)
- Web: clasificación automática de páginas web para directorios
- Reconocimiento de caracteres, de voz. etc.
- Predicción de la demanda eléctrica, de gas, etc.

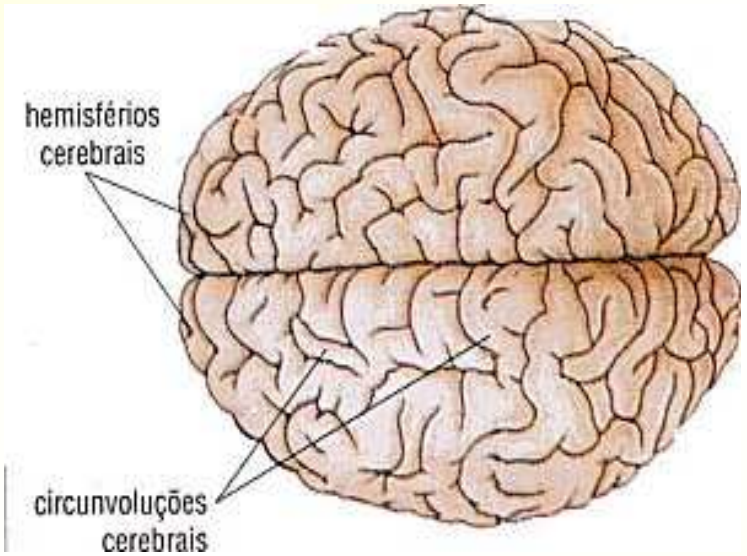
El Brain Computer Interface *(BCI)*

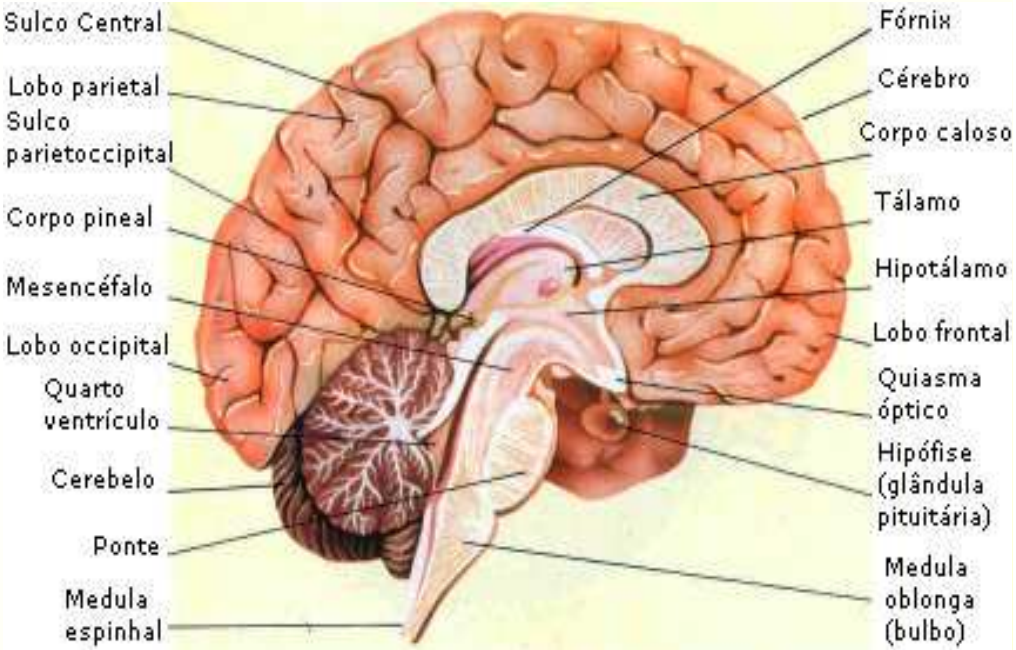
- Objetivo: comunicar personas con ordenadores mediante el pensamiento
- Ayudar a personas inmovilizadas
- Existen otros métodos (movimiento de los ojos, nervios, etc.)

Neuronas (interruptor 0/1)

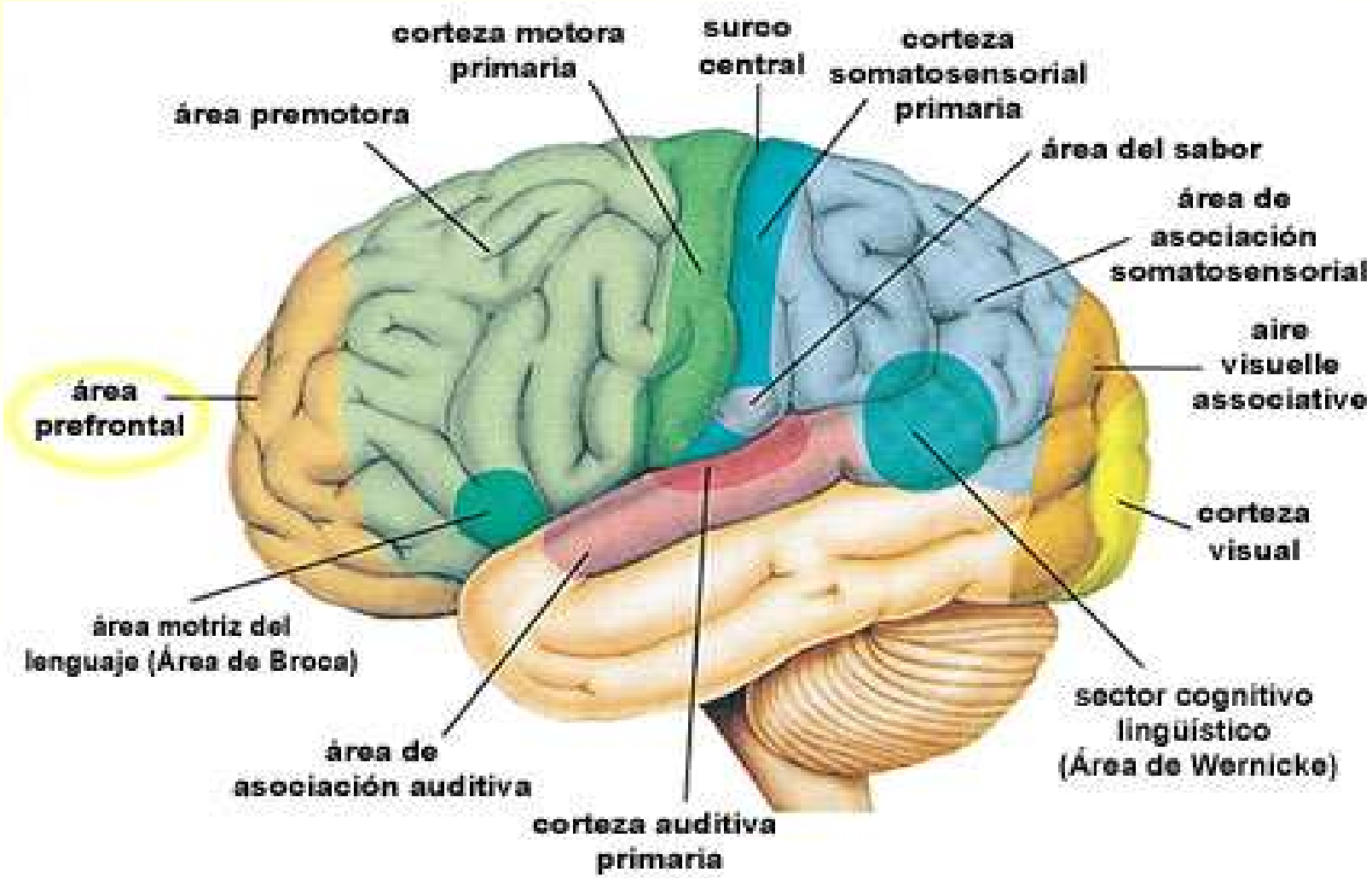


El cerebro (red de billones de neuronas)

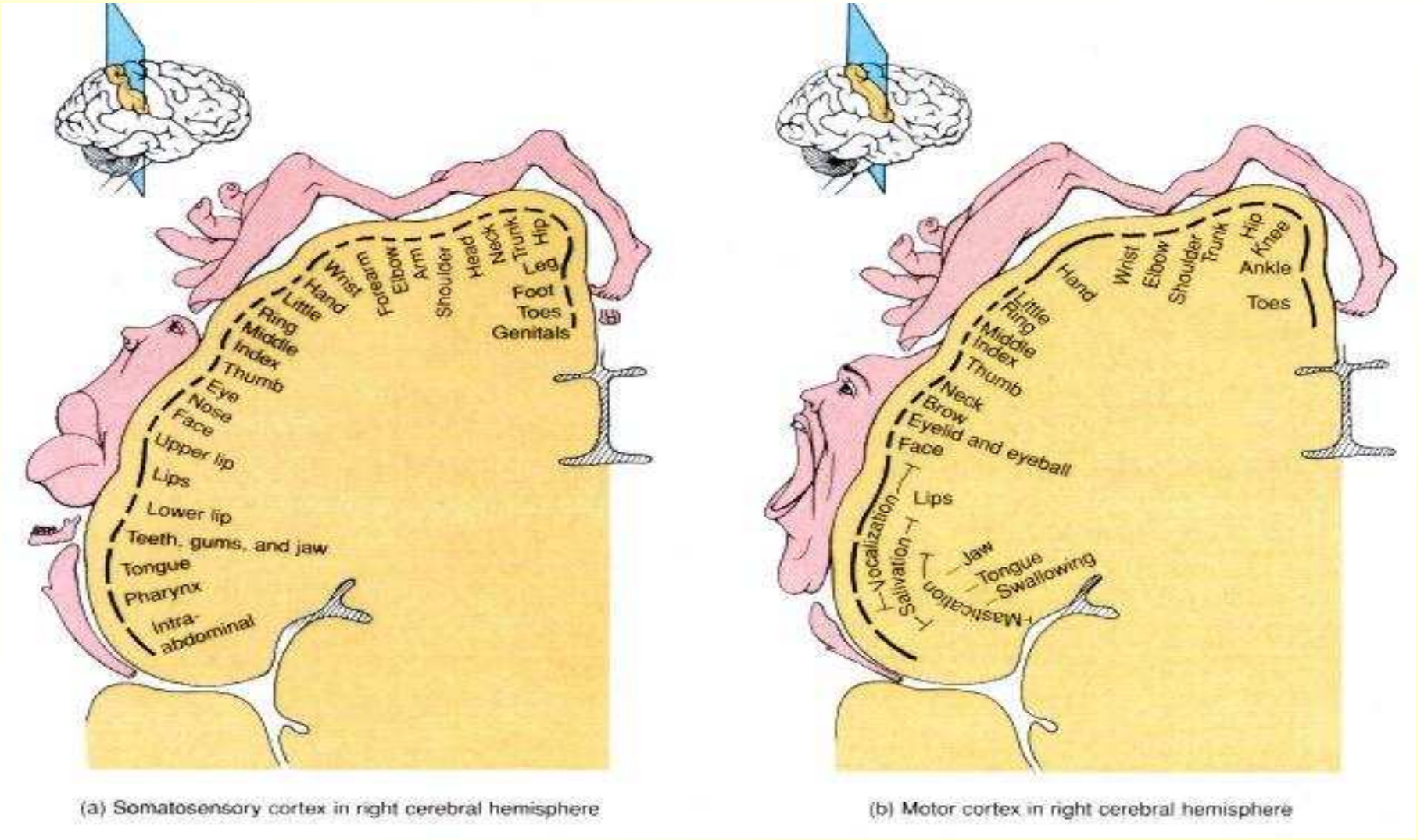




Áreas funcionales del cerebro



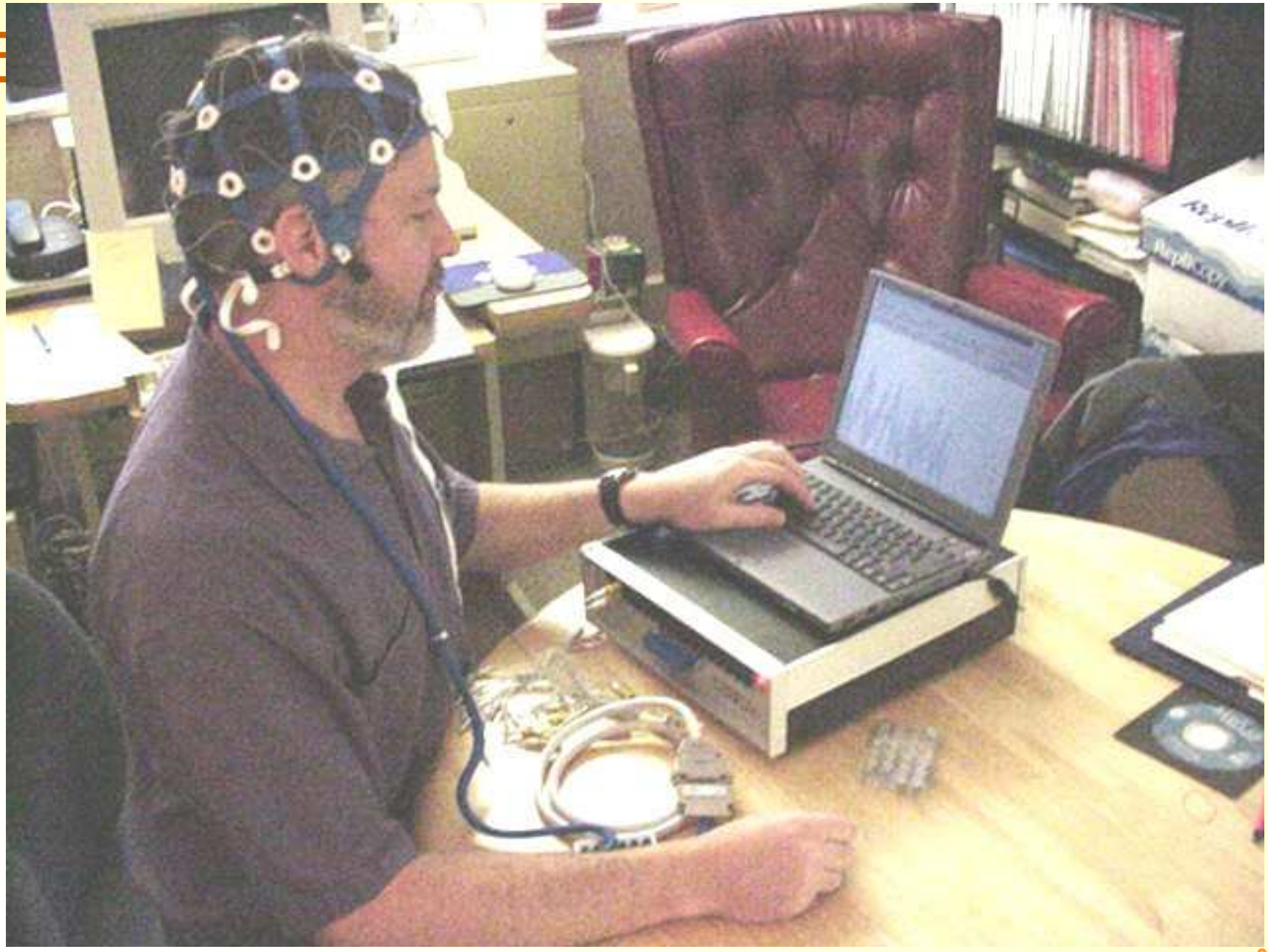
Sensory Homunculus



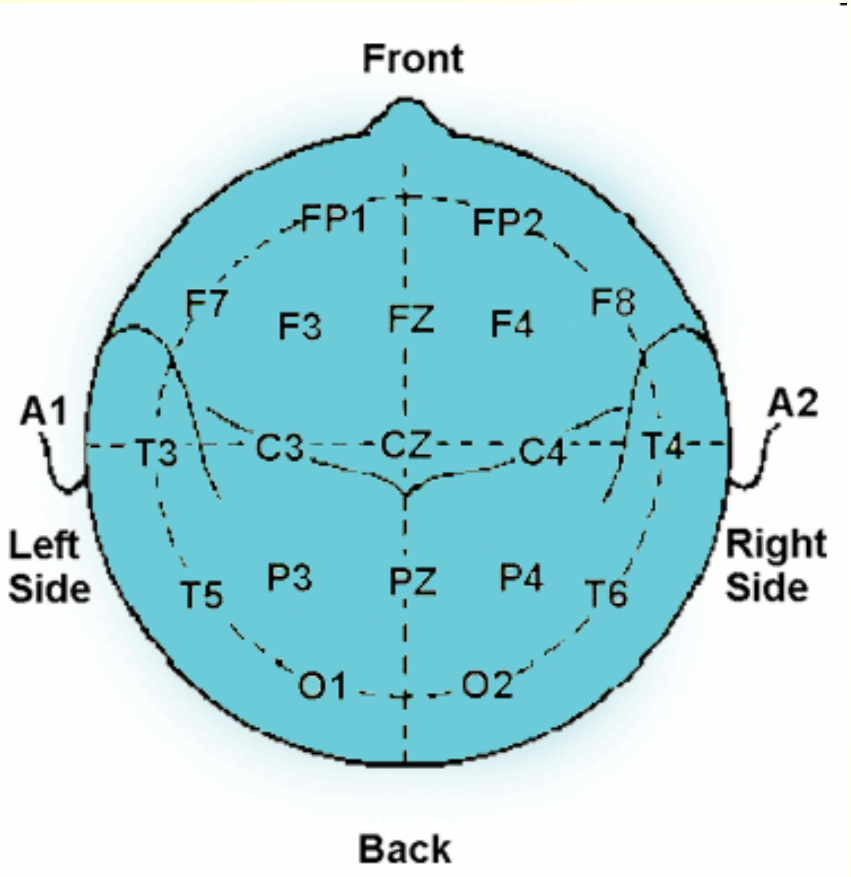
(a) Somatosensory cortex in right cerebral hemisphere

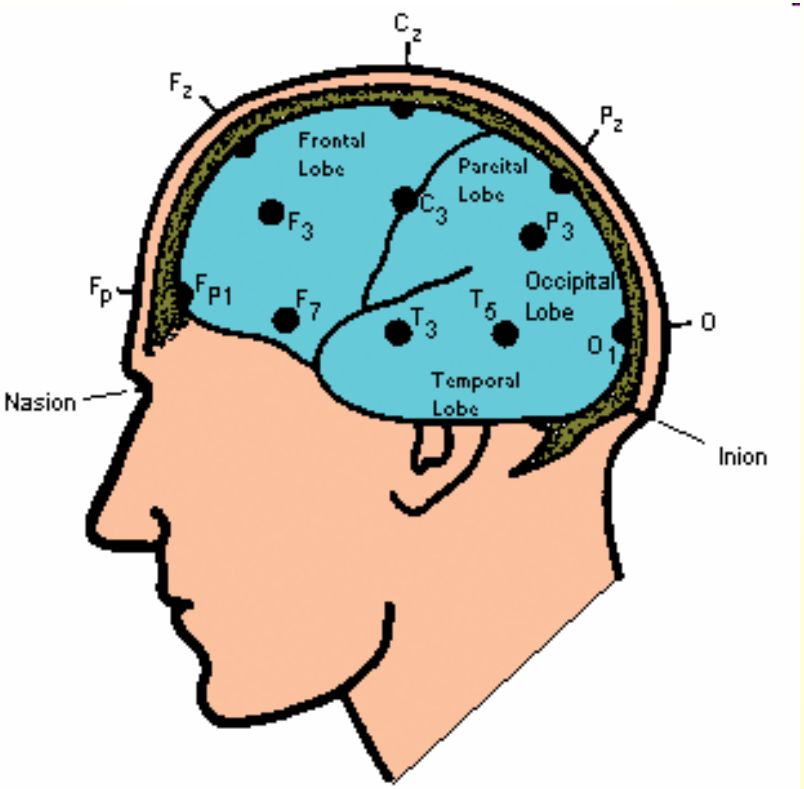
(b) Motor cortex in right cerebral hemisphere

E



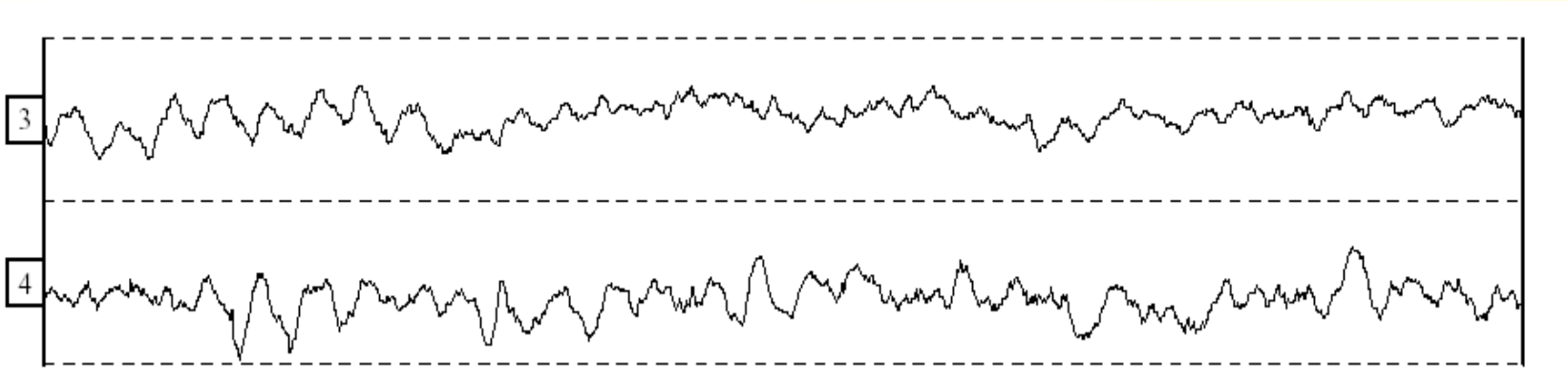
Sistema 10-20





El electro-encefalograma (EEG)

- Cambios de potencial -> ondas electromagnéticas (muy débiles)
- Medición: invasiva o no invasiva
- El aprendizaje se utiliza para decodificar las ondas (cada individuo tiene sus peculiaridades)
- Para hacer aprendizaje automático es necesario convertir cada forma de onda en un conjunto de atributos que la caracterize (transformada de Fourier, PSD)
- Es útil la banda de frecuencias entre 8Hz y 30Hz

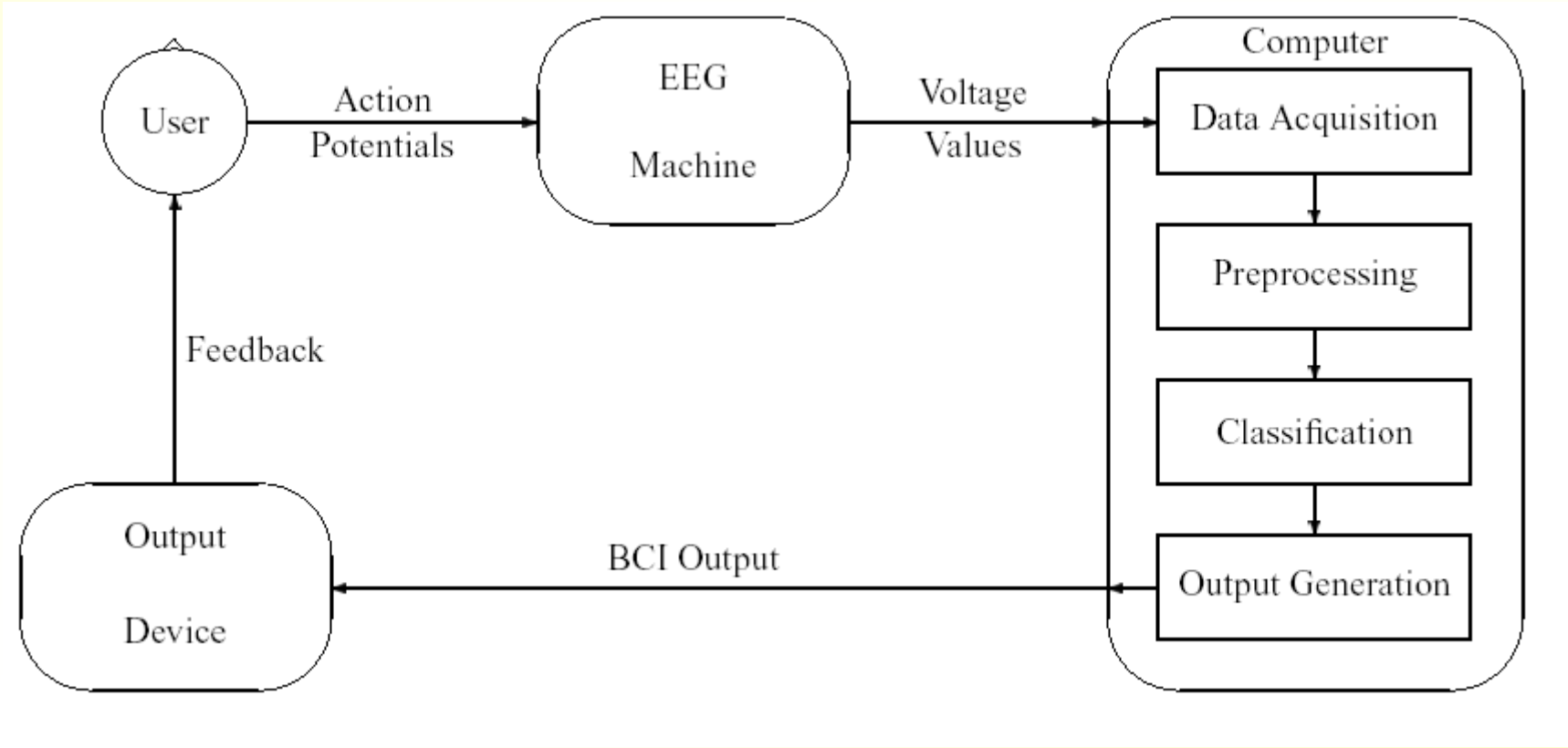


Aplicaciones del EEG

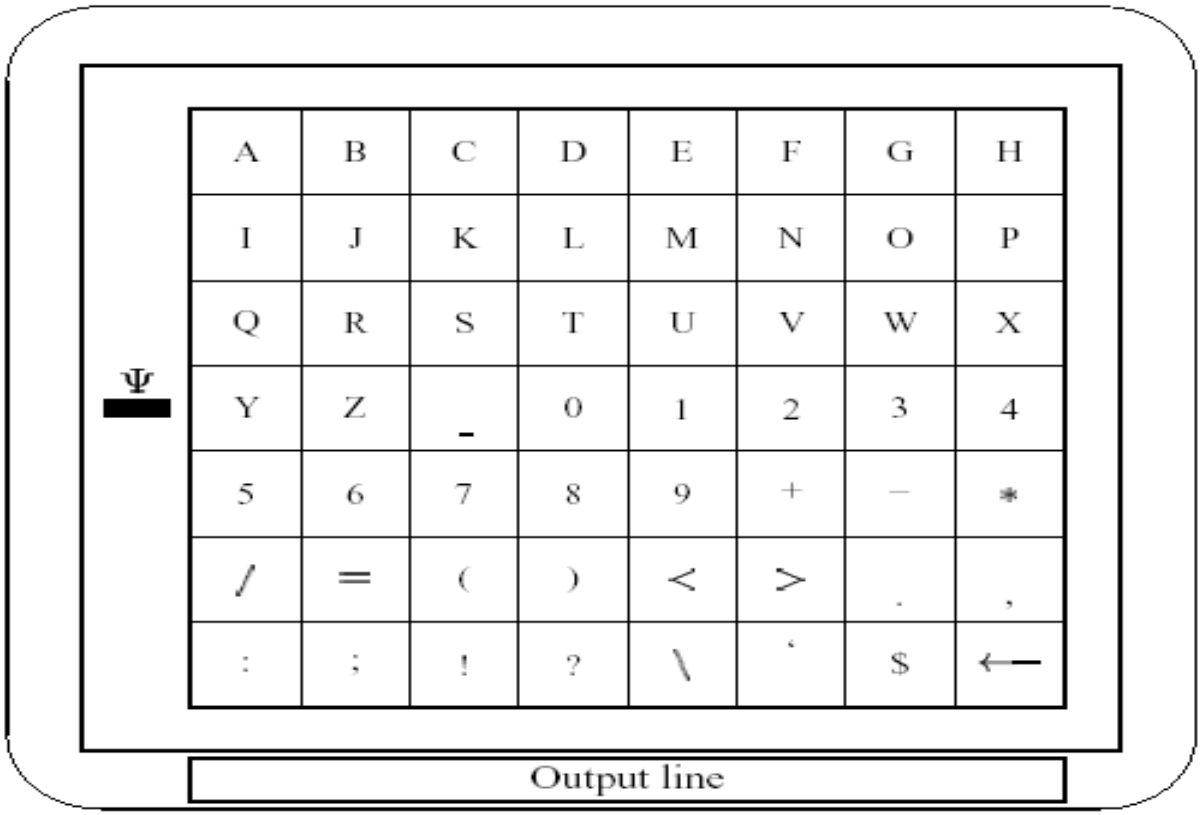
- Diagnóstico de enfermedades (epilepsia)
- Biofeedback
- El Interfaz cerebro-máquina

Frequency Band	Range
Alpha (α)	$8 - 13$ Hz
Beta (β)	$14 - 30$ Hz
Theta (θ)	$4 - 7$ Hz
Delta (δ)	$0.5 \quad 3$ Hz

Esquema del BCI



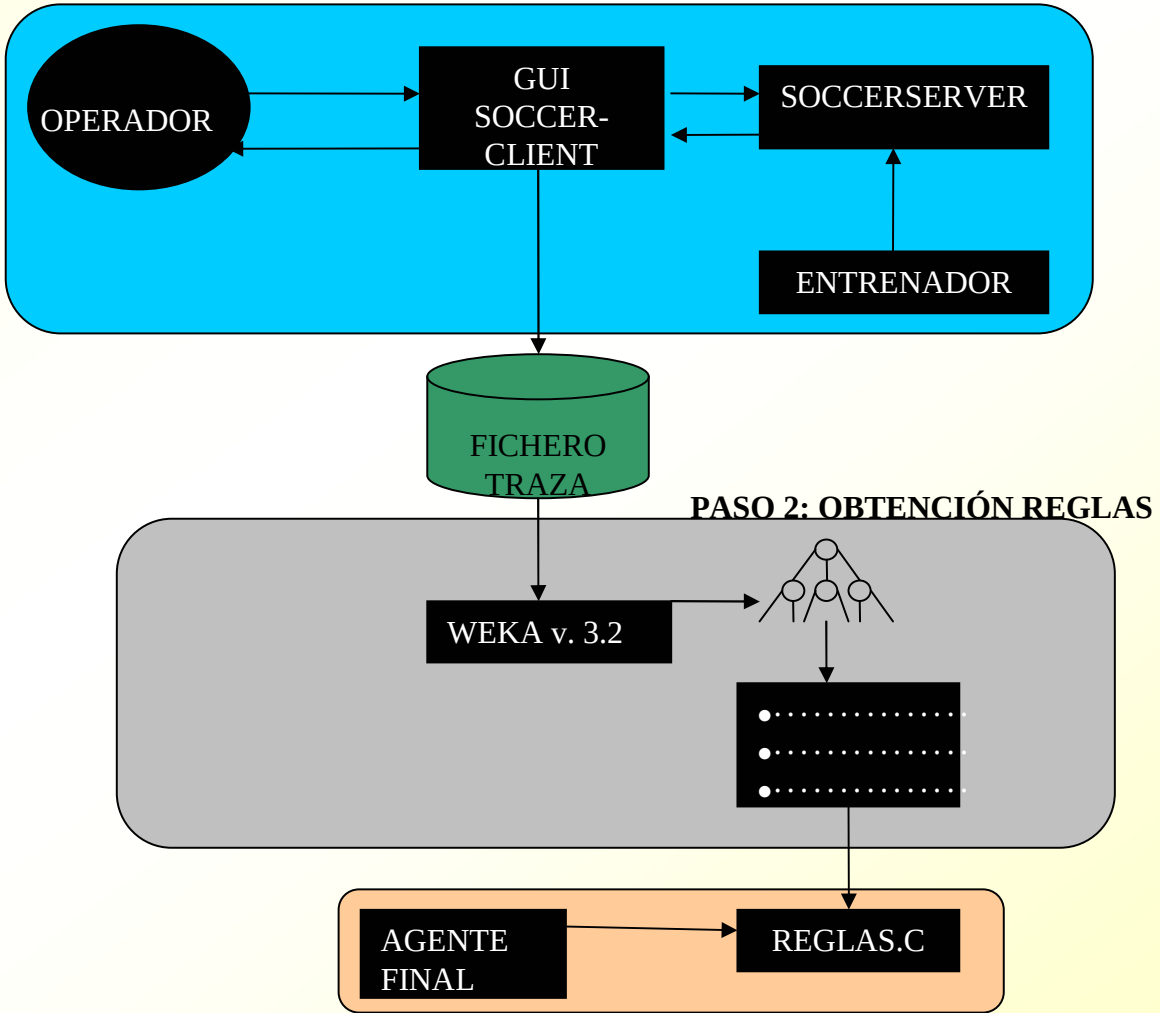
El spellboard



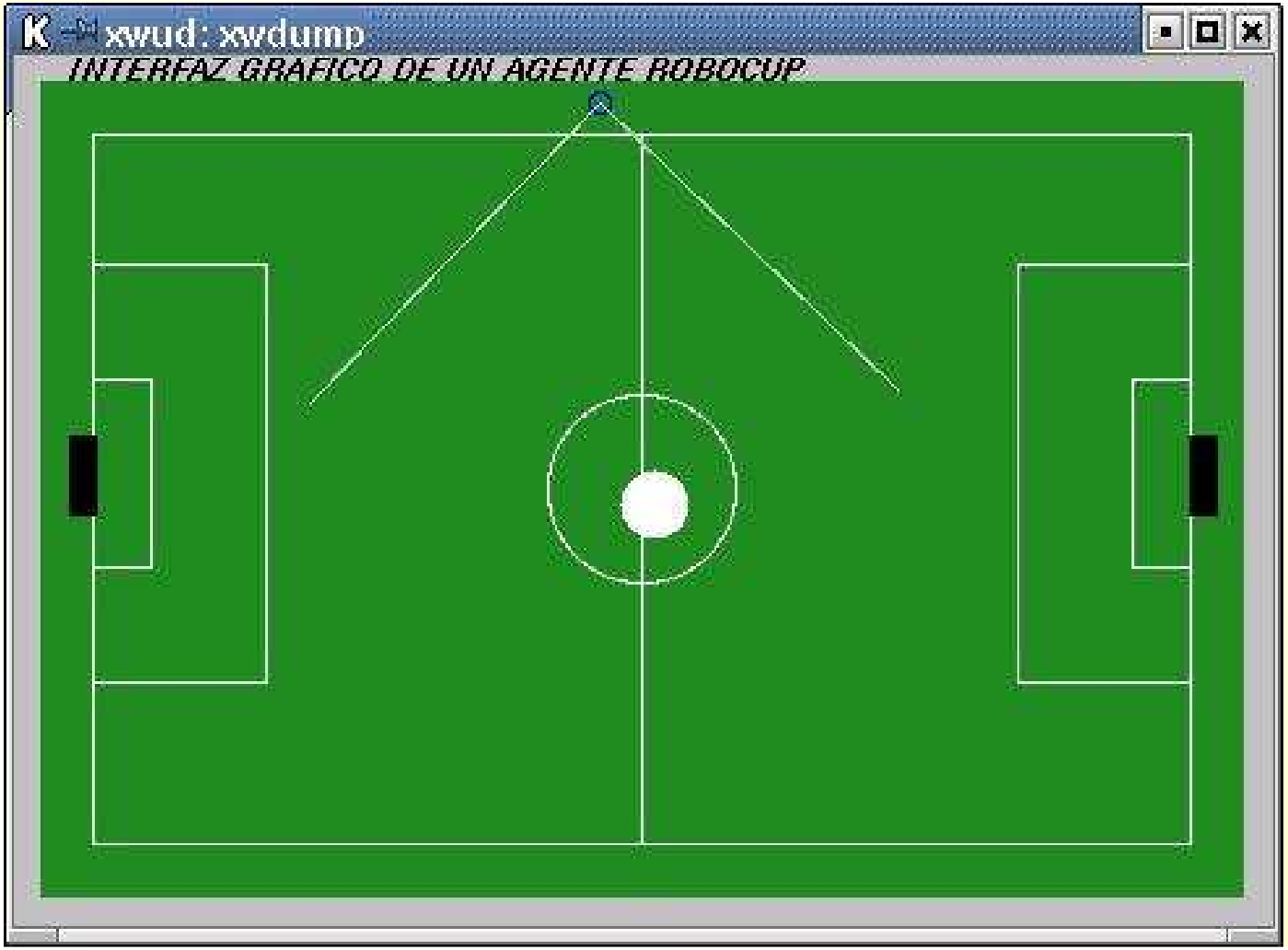
Modelización de Jugadores Humanos para la Robocup

- Se trata de obtener un modelo de una persona jugando a Robosoccer, para después programar a un agente que juegue de forma similar
- Datos para aprender: (Sensores, Acción) [lo que ve el agente en el momento *t*, lo que hace la persona en *t*]
- PFC Alberto López Cilleros

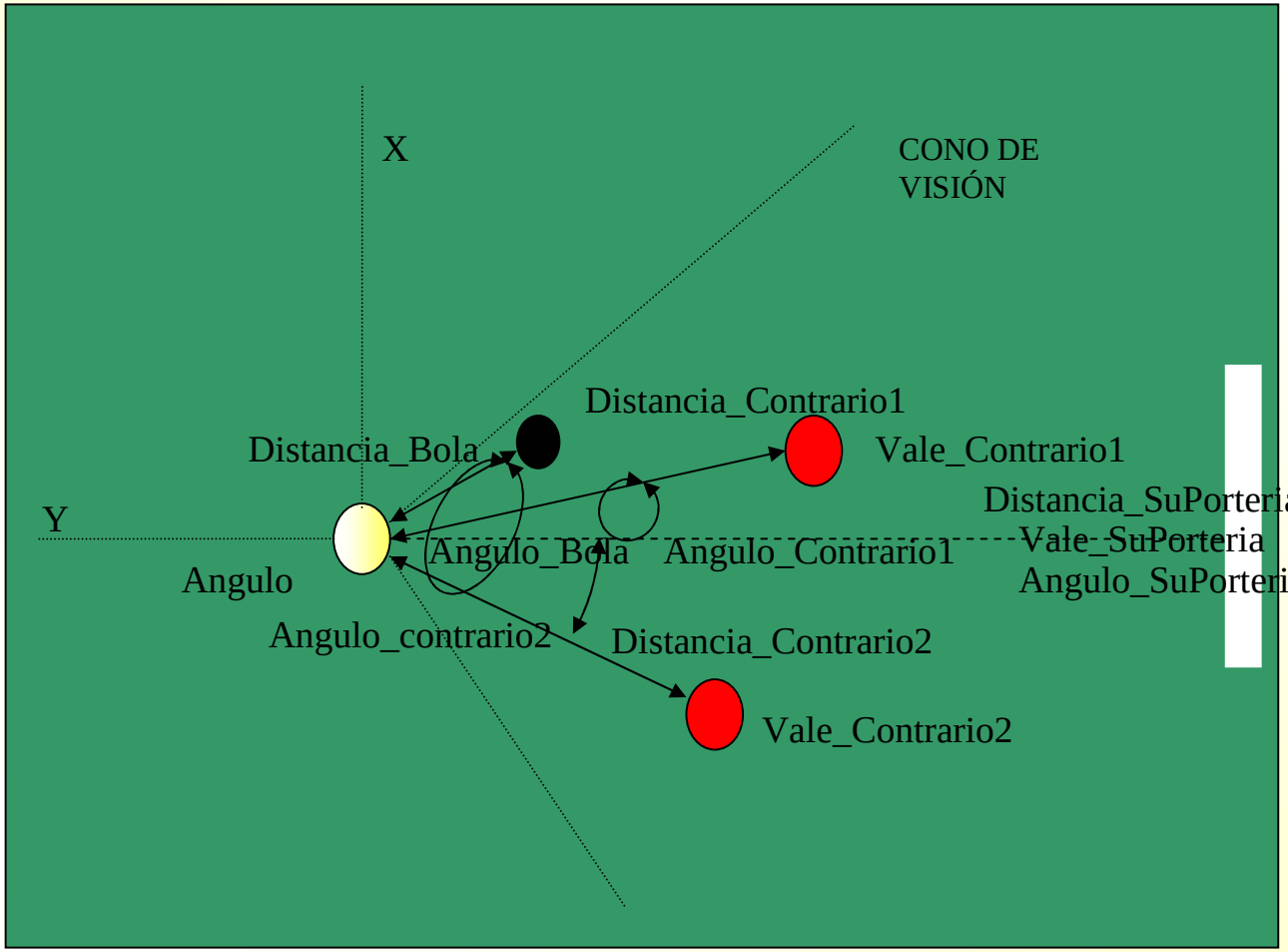
Esquema de aprendizaje



GUI Soccerclient



Atributos a utilizar



Acciones

Acciones
Avanzar rápido: **dash99**
Avanzar lento: **dash 60**
Girar 10º Derecha: **turn-right-10**
Girar 10º Izquierda: **turn-left-10**
Tirar a puerta: **kick99**
Tiro corto: **kick60**

Atributos con los que aprender

@attribute Distancia_Bola real,

@attribute Angulo_Bola real,

@attribute X_Bola real,

@attribute Y_Bola real

@attribute Angulo real,

@attribute Distancia_Contrario1 real,

@attribute Angulo_Contrario1 real

@attribute Vale_Contrario1 real,

@attribute Distancia_Contrario2 real,

@attribute Angulo_Contrario2 real

@attribute Vale_Contrario2 { 0, 1 },

@attribute Distancia_SuPorteria real,

@attribute Angulo_SuPorteria real

@attribute Vale_SuPorteria { 0, 1 },

@attribute Accion { dash99, dash60, turnmenos45, turn45, kick99, kick60 }

Datos de los que aprender

36.6,-4,3.26785,36.9995,-91,1000,1000,0,1000,1000,0,1000,1000,0,dash99

36.6,-4,3.26785,36.9995,-91,1000,1000,0,1000,1000,0,1000,1000,0,dash99

36.6,-4,3.12456,36.1997,-91,1000,1000,0,1000,1000,0,1000,1000,0,dash99

13.5,-13,6.88539,12.4929,-91,1000,1000,0,1000,1000,0,1000,1000,0,turn10

33.1,-5,2.92243,33.3897,-91,1000,1000,0,1000,1000,0,1000,1000,0,dash99

33.1,-5,2.92243,33.3897,-91,1000,1000,0,1000,1000,0,1000,1000,0,dash99

33.1,-5,3.63682,33.3897,-91,1000,1000,0,1000,1000,0,1000,1000,0,dash99

Ejemplo de conocimiento obtenido

```
if ((Angulo_Bola > -37 )&&(Distancia_Bola > 1.2 )
&&(Angulo_Bola <= 24)) {dash99(memoria,puerto); break;}

if ((Angulo_Bola > 19 )&&(Angulo_Bola <= 42 )&&(X <=
33.9477)) {dash99(memoria,puerto);break;}

if ((Angulo_Bola > 11)) {turn10(memoria,puerto);break;}

if ((Distancia_Bola <= 0.4 )&&(Angulo_Bola <= -20))
{turn10(memoria,puerto);break;}
```

FASES EN MINERÍA DE DATOS

Fases del proceso de extracción de conocimiento

- Integración y recopilación de datos
- Selección, limpieza y transformación -> Datos
- **Aprendizaje Automático -> Patrones**
- Evaluación e interpretación -> Conocimiento
- Difusión y uso -> Decisiones

Integración y recopilación

- Almacenes de datos (data warehousing): repositorio de información obtenido de diversas fuentes (heterogéneas), almacenada bajo un esquema unificado

Selección, limpieza, transformación

- Datos:
  - Valores que no se ajustan al comportamiento general (*outliers*): eliminar o dejar
  - Muestreo de datos
- Atributos:
  - Eliminar atributos redundantes o irrelevantes
  - Reducción/aumento dimensionalidad
  - Calcular nuevos atributos que sean más relevantes (area, población -> densidad de población, para predecir cantidad de daños en terremotos)
  - Valores faltantes (*missing values*): rellenarlos
  - Discretización, numerización, normalización, …

Datos (instancias, patrones, …)

Atributos

Clase

Datos

IDC	Años	Euros	Salario	Casa propia	Cuentas morosas	…	Devuelve el crédito
101	15	60000	2200	Si	2	…	No
102	2	30000	3500	Si	0	…	Si
103	9	9000	1700	Si	1	…	No
104	15	18000	1900	No	0	…	Si
105	10	24000	2100	No	0	…	No
…	…	…	…	…	…		…

Selección, limpieza, transformación

- Objetivos:
  - Mejorar la eficiencia de la herramienta de minería de datos
  - Mejorar la calidad (precisión) del conocimiento obtenido
- Posibilidades:
  - Hacerlo a mano
  - Utilizar herramientas de preproceso (ej: selección de atributos)
  - Dejar que lo haga el algoritmo de minería de datos (peor solución)

- **TIPOS DE ALGORITMOS PARA PREDICCIÓN (CLASIFICACIÓN Y REGRESIÓN)**

Datos de entrada (ej: clasificación)

Cielo	Temperatura	Humedad	Viento	Tenis
Sol	85	85	No	**No**
Sol	80	90	Si	**No**
Nublado	83	86	No	**Si**
Lluvia	70	96	No	**No**
Lluvia	68	80	No	**Si**
Nublado	64	65	Si	**Si**
Sol	72	95	No	**No**
Sol	69	70	No	**Si**
Lluvia	75	80	No	**Si**
Sol	75	70	Si	**Si**
Nublado	72	90	Si	**Si**
Nublado	81	75	No	**Si**
Lluvia	71	91	Si	**No**

Esquema general en predicción

Datos

Cielo	Temperatura	Humedad	Viento	Tenis
Sol	85	85	No	No
Sol	80	90	Si	No
Nubes	83	86	No	Si
Lluvia	70	96	No	So
Lluvia	68	80	No	Si
Nublado	64	65	Si	Si
Sol	72	95	No	No
Sol	69	70	No	Si
Lluvia	75	80	No	Si
Sol	75	70	Si	Si
Nublado	72	90	Si	Si
Nublado	81	75	No	Si
Lluvia	71	91	Si	No

Dato a clasificar

Cielo	Temperatura	Humedad	Viento	Tenis
Sol	60	65	No	?????

Algoritmo MD

Clasificador

IF Cielo = Sol **Y** Humedad <= 75

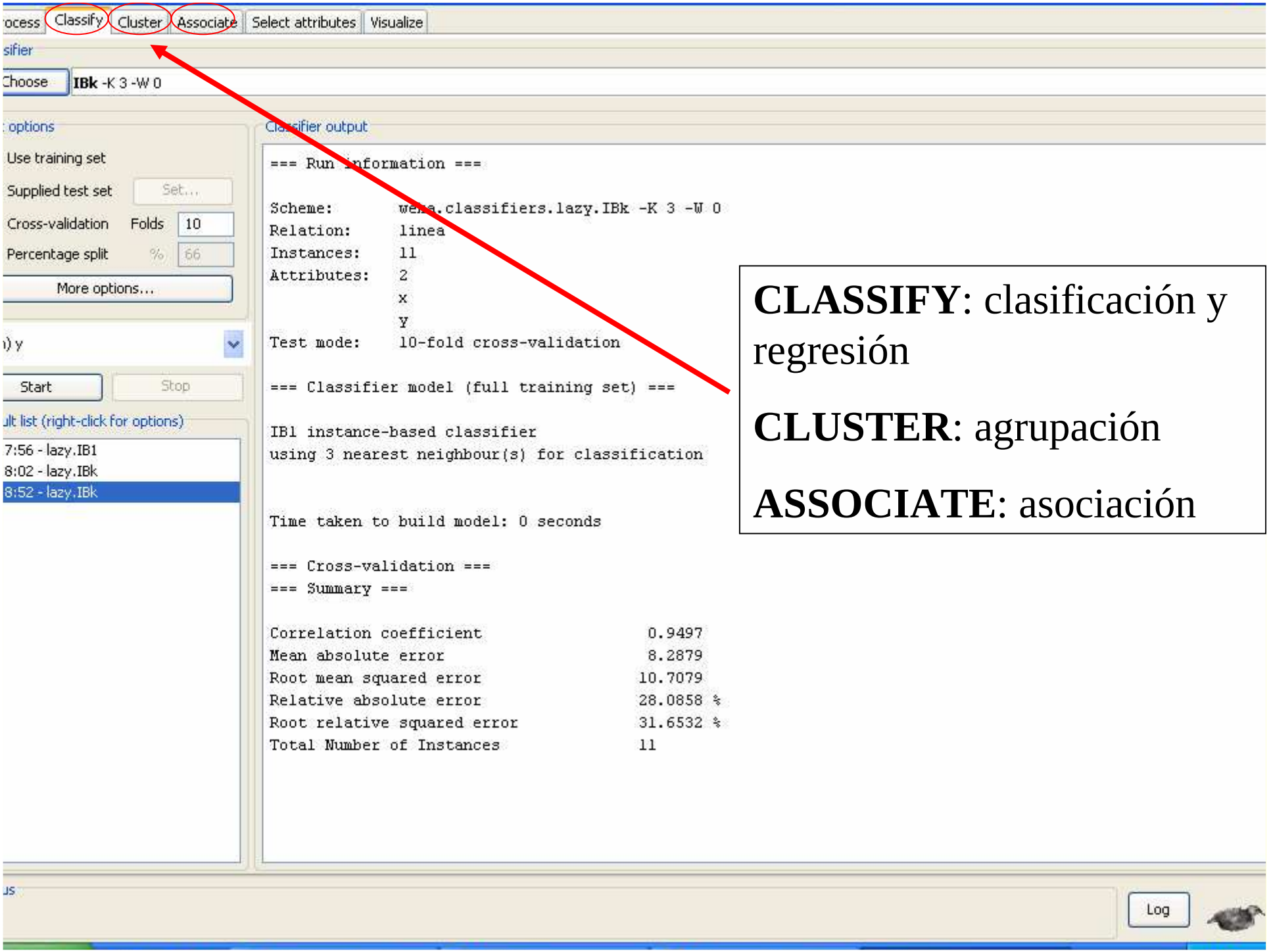
THEN Tenis = Si ...

Predicción

Clase = Si

Idea general en predicción

- Utilizar un conjunto de datos disponibles, en forma de tabla de atributos, para aprender un *predictor (clasificador o regresor)*, que sea capaz de predecir la clase de datos **NO VISTOS TODAVÍA**. Hay **GENERALIZACIÓN** a partir de los datos
- El *predictor* puede tomar diversas formas, según el algoritmo (árbol de decisión, reglas, función, red de neuronas, probabilidades, centroides, ..)
- Pero en último término, un *predictor* es una estructura que toma una entrada (los distintos valores de los atributos que representan al dato) y devuelve una salida (la clase o cantidad predicha para ese dato)



CLASSIFY: clasificación y regresión

CLUSTER: agrupación

ASSOCIATE: asociación

Tipos de atributos

- **Nominales** (discretos, categóricos): cielo, viento
- **Numéricos**: temperatura, humedad
- Hay atributos numéricos que son realmente nominales (ej: DNI)
- Hay atributos nominales que son realmente numéricos (ej: edad con valores “niño”, “joven”, “adulto”, “mayor”).

Formato arff. Definición de atributos

% Comentarios precedidos de %

@relation tiempo

@attribute cielo {sol, nubes, lluvia}

@attribute temperatura numeric

@attribute humedad numeric

@attribute viento {si, no}

@attribute tenis {si, no}

Formato arff. Definición de datos

@data

Sol, 85, 85, no, no

Sol, 80, 90, si, no

Nublado, 81, 86, no, si

Lluvia, 70, 96, no, si

...

Formato Arff

- @relation tiempo
- @attribute cielo {sol, nubes, lluvia}
- @attribute temperatura numeric
- @attribute humedad numeric
- @attribute viento {si, no}
- @attribute tenis {si, no}
- @data
- Sol, 85, 85, no, no
- Sol, 80, 90, si, no
- Nublado, 81, 86, no, si
- Lluvia, 70, 96, no, si

Algoritmos de clasificación / regresión (predicción)

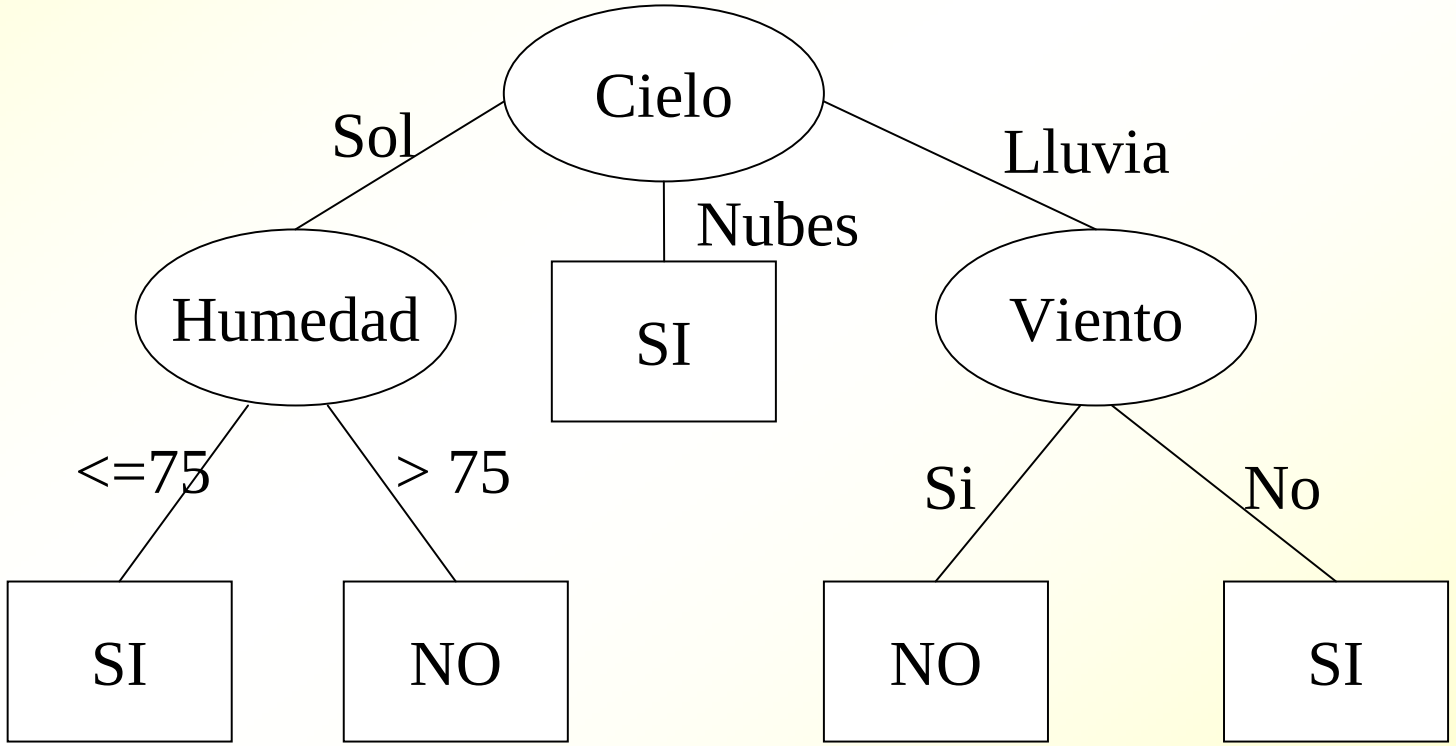
- Árboles de decisión y reglas. Para clasificación
  - Árboles de decisión: ID3, C4.5 (J48), ...
  - Reglas: PART, CN2, AQ, ...
- Funciones:
  - Para regresión: linear regressíon, neural networks
  - Para clasificación: simple logistics, support vector machines (SMO)
- Árboles de regresión: LMT (M5), ...
- Técnicas perezosas. Para clasificación y regresión
  - IB1, IBK, ...
- Técnicas Bayesianas. Para clasificación:
  - Naive Bayes
- Metatécnicas. Para clasificación y regresión:
  - Boosting, Bagging, Stacking, Random Forests

Tipos de clasificadores (y regresores)

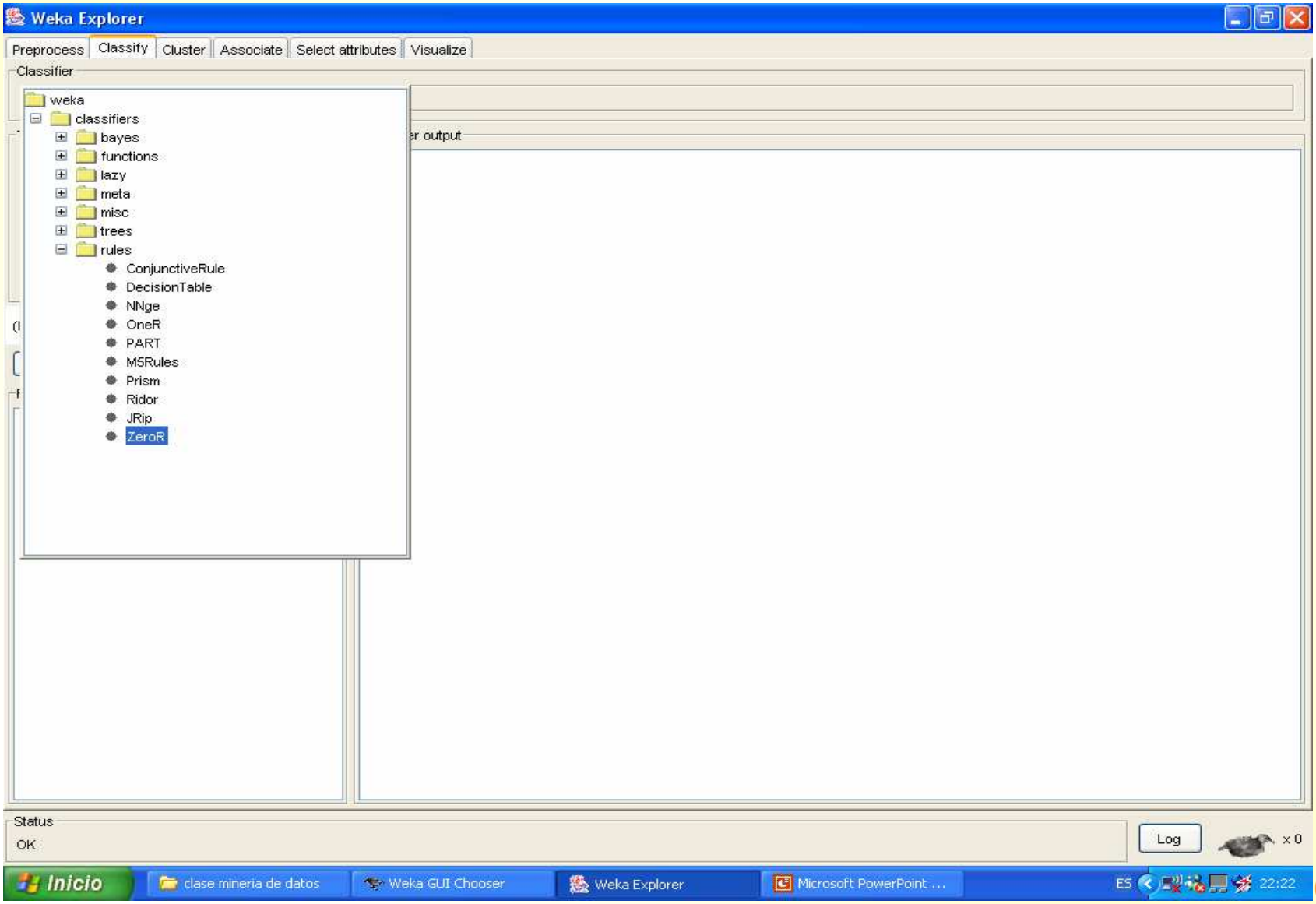
- En el fondo, la clasificación consiste en encontrar fronteras de separación entre las clases)
- Tipos:
  - Lineales: la frontera es una línea (en dos dimensiones) o un hiperplano (en N-dimensiones)
  - No lineales: cualquier otro tipo de frontera
- Caso de que hablemos de regresores, el objetivo en este caso es aprender una función y también encontramos los tipos lineal y no lineal

- Árboles de decisión y reglas. Para clasificación
  - Árboles de decisión: ID3, C4.5 (J48), ...
  - Reglas: PART, CN2, AQ, ...

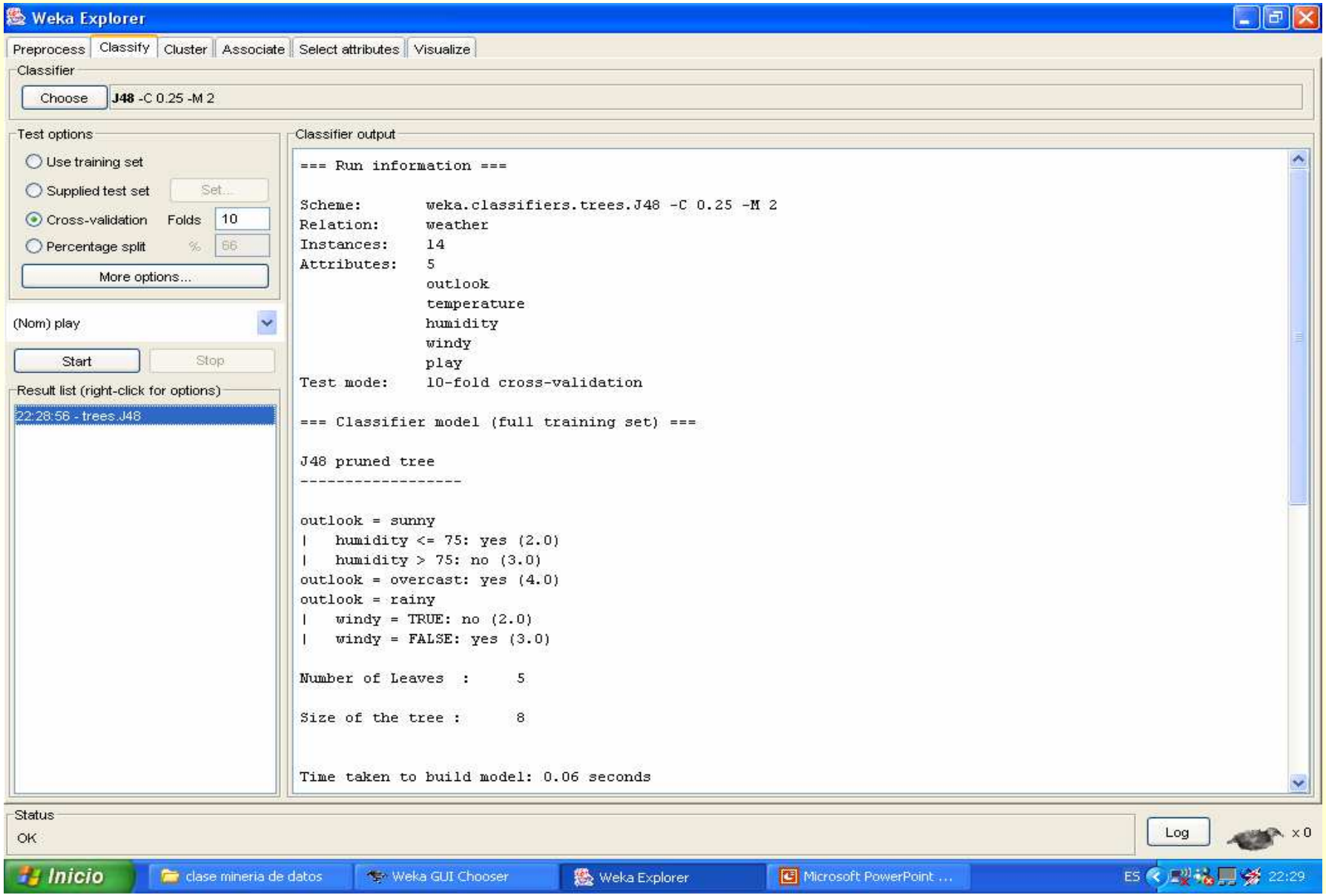
Árboles de decisión (para clasificación)



Algoritmos de predicción (**classifiers**) en Weka



Arbol de decisión J48



Algoritmos de construcción de árboles de decisión

- El más básico es el ID3: construye árboles de decisión de manera recursiva, de la raíz hacia las hojas, seleccionando en cada momento el mejor nodo para poner en el árbol
- El C4.5 (o J48), trata con valores continuos y utiliza criterios estadísticos para impedir que el árbol se sobreadapte (que “crezca demasiado”, que se aprenda los datos en lugar de generalizar)

Algoritmo ID3 simplificado

1. Detener la construcción del árbol si:
    1. Todos los ejemplos pertenecen a la misma clase
    2. Si no quedan ejemplos o atributos
2. Si no, elegir el mejor atributo para poner en ese nodo (el que minimice la entropía media)
3. Crear de manera recursiva tantos subárboles como posibles valores tenga el atributo seleccionado

Algoritmo ID3 detallado

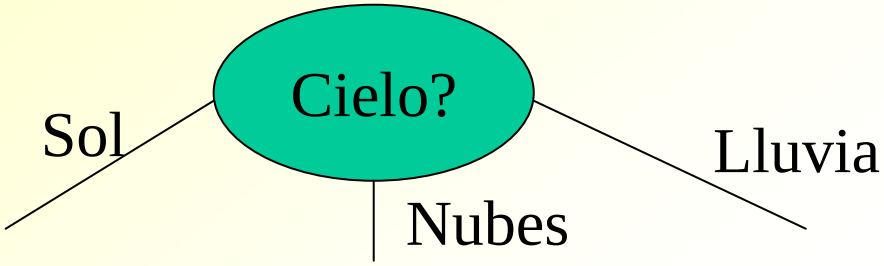
- **ID3(Ejemplos, Atributo-objetivo, Atributos)**

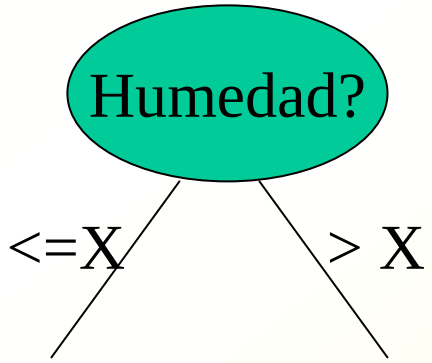
```
1. Si todos los Ejemplos son positivos, devolver un nodo etiquetado con +
2. Si todos los Ejemplos son negativos, devolver un nodo etiquetado con -
3. Si Atributos está vacío, devolver un nodo etiquetado con el valor más
   frecuente de Atributo-objetivo en Ejemplos.
4. En otro caso:
   4.1. Sea A el atributo de Atributos que MEJOR clasifica Ejemplos
   4.2. Crear Árbol, con un nodo etiquetado con A.
   4.3. Para cada posible valor v de A, hacer:
     * Añadir un arco a Árbol, etiquetado con v.
     * Sea Ejemplos(v) el subconjunto de Ejemplos con valor
       del atributo A igual a v.
     * Si Ejemplos(v) es vacío:
       - Entonces colocar debajo del arco anterior un nodo etiquetado con
         el valor más frecuente de Atributo-objetivo en Ejemplos.
       - Si no,  colocar debajo del arco anterior el subárbol
         ID3(Ejemplos(v), Atributo-objetivo, Atributos-{A}).
   4.4  Devolver Árbol
```

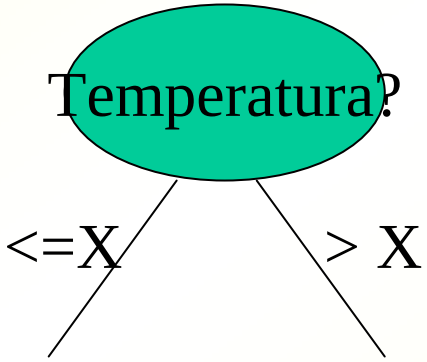
Algoritmo C4.5 simplificado

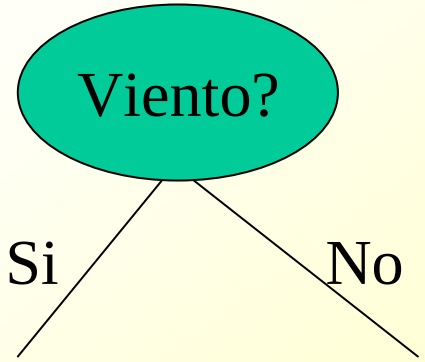
1. Detener la construcción del árbol si:
    1. Todos los ejemplos pertenecen a la misma clase
    2. Si no quedan ejemplos o atributos
    3. Si no se espera que se produzcan mejoras continuando la subdivisión
2. Si no, elegir el mejor atributo para poner en ese nodo (el que minimice la entropía media)
3. Crear de manera recursiva tantos subárboles como posibles valores tenga el atributo seleccionado

¿Qué nodo es el mejor para poner en la raíz del árbol?



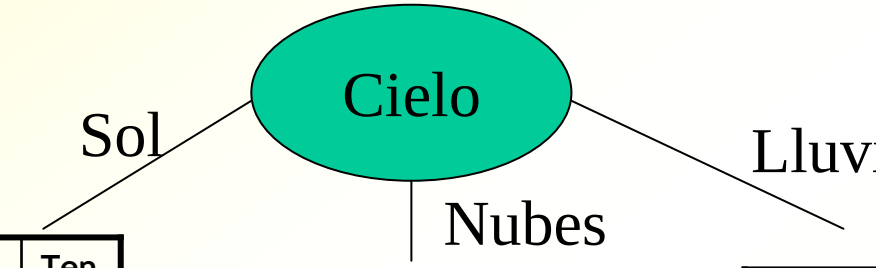






Supongamos que usamos Cielo

Cielo nos genera tres particiones de los datos, tantas como valores posibles tiene



C	T	H	V	Ten
Sol	85	85	No	No
Sol	80	90	Si	No
Sol	72	95	No	No
Sol	69	70	No	Si
Sol	75	70	Si	Si

C	T	H	V	Ten
Nubes	83	86	No	Si
Nubes	64	65	Si	Si
Nubes	72	90	Si	Si
Nubes	81	75	No	Si

C	T	H	V	Ten
Lluvia	70	96	No	No
Lluvia	68	80	No	Si
Lluvia	75	80	No	Si
Lluvia	65	70	Si	No
Lluvia	71	91	Si	No

“3 No, 2 Si”

Tendencia al “no”

“0 No, 4 Si”

Partición perfecta

“3 No, 2 Si”

Tendencia al “no”

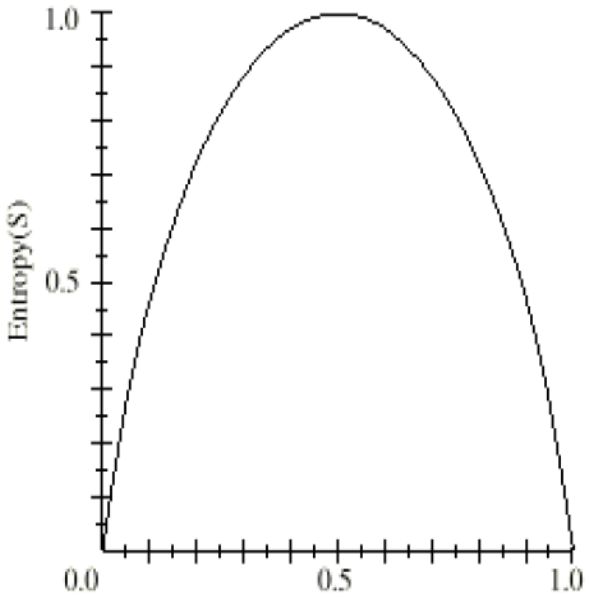
¿Cómo medimos lo bueno que es Cielo como atributo para clasificar?

- Usaremos una medida que obtenga el mejor valor cuando el atributo me obtenga particiones lo mas homogéneas posible, en media
  - Homogénea: “0 No, todo Si”; o bien “todo No, 0 Si”
  - Indecisión: “50% No, 50% Si”
- Una medida que me dice lo lejana que está una partición de la perfección es la entropía
- A mayor entropía, peor es la partición

$$H(P) = -\sum_{Ci} p_{Ci} \log_2(p_{Ci})$$

$$H(P) = -(p_{si} \log_2(p_{si}) + p_{no} \log_2(p_{no}))$$

$$p_{no} = (1 - p_{si})$$



Entropía media de Cielo

- Cielo genera tres particiones cuya entropía es:
  1. "3 No, 2 Si": $H=-((3/5)*\log_2(3/5)+(2/5)*\log_2(2/5) = 0.97$
  2. "0 No, 4 Si": $H=-((0/4)*\log_2(0/4)+1*\log_2(1))=0$
  3. "3 No, 2 Si": $H=-((3/5)*\log_2(3/5)+(2/5)*\log_2(2/5))= 0.97$

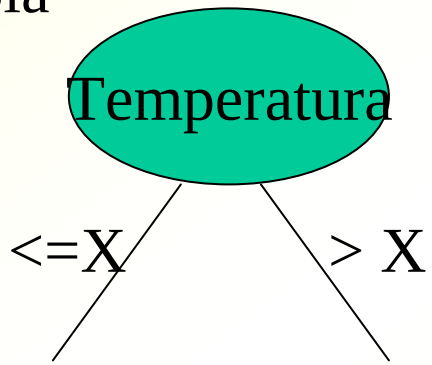
La entropía media ponderada de Cielo será:

- $HP=(5/14)*0.97+(4/14)*0+(5/14)*0.97 = \mathbf{0.69}$
- Nota: hay 14 datos en total

¿Y si el atributo es contínuo?

Hay que partir por el valor X, donde sea mas conveniente, minimizando la entropía

Nota: solo hemos probado algunas de las posibles particiones, entre las que se encuentra la mejor



X<=70

64 – Si, 65 – No, 68 – Si, 69 – Si, 70 – Si, | 71 – No, 72 – No Si, 75 – Si Si, 80 – No, 81 – Si, 83 – Si, 85 - No

1 No, 4 Si

4 No, 5 Si

HP = 0.89

64 – Si, 65 – No, 68 – Si, 69 – Si, 70 – Si, 71 – No, 72 – No Si | 75 – Si Si, 80 – No, 81 – Si, 83 – Si, 85 - No

3 No, 5 Si

2 No, 4 Si

HP = 0.93

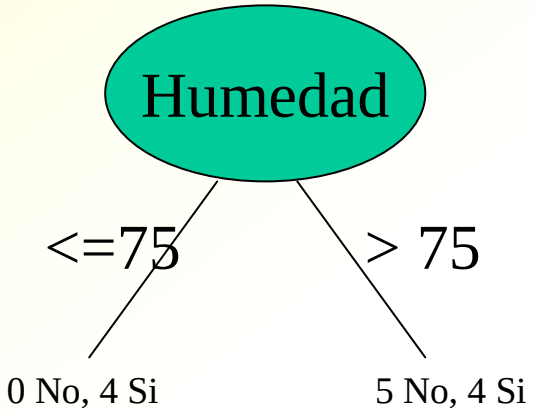
64 – Si, 65 – No, 68 – Si, 69 – Si, 70 – Si, 71 – No, 72 – No Si, 75 – Si Si, | 80 – No, 81 – Si, 83 – Si, 85 - No

3 No, 7 Si

2 No, 2 Si

HP = 0.91

Caso de humedad



65-Si, 70-No Si Si, 75–Si, 80-Si Si, | 85-No, 86-Si, 90-No Si, 91-No, 95-No, 96-Si,

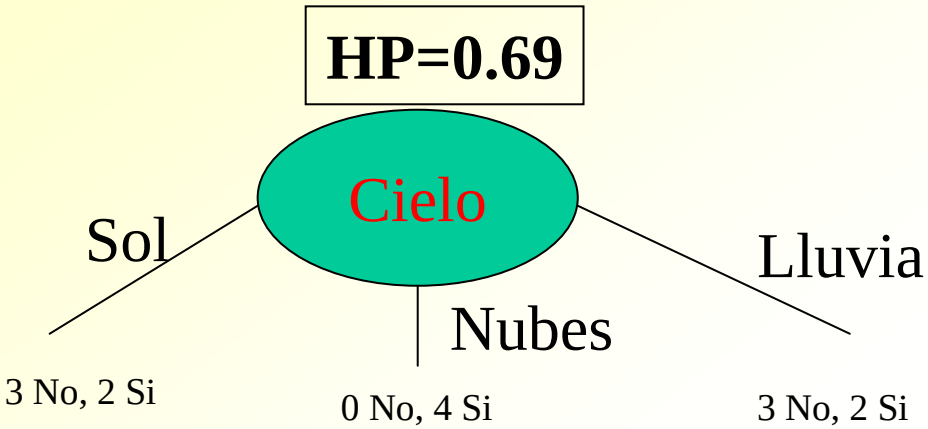
1 No, 6 Si

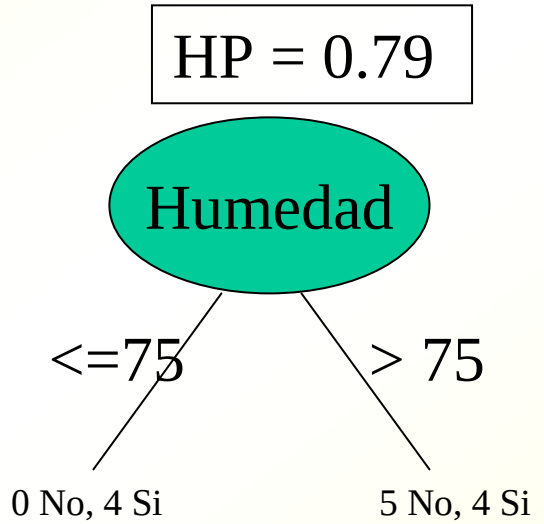
4 No, 3 Si

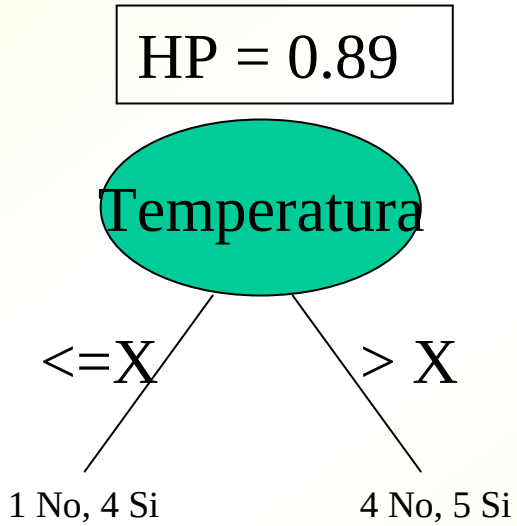
HP = 0.79

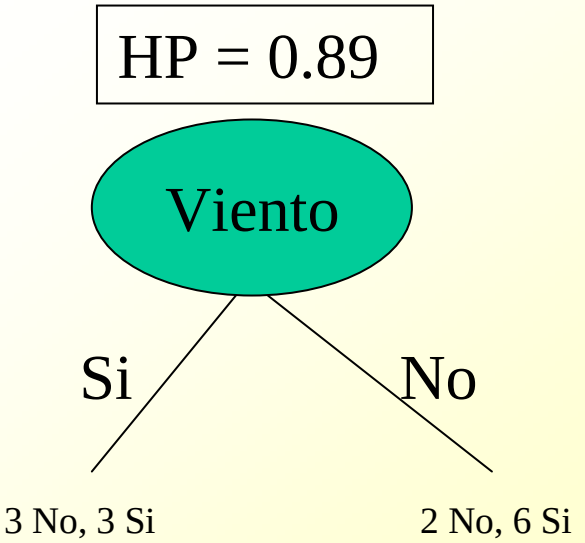
Nota: hay otras posibilidades de particiones, pero esta es la mejor

¿Qué nodo es el mejor para poner en la raíz?



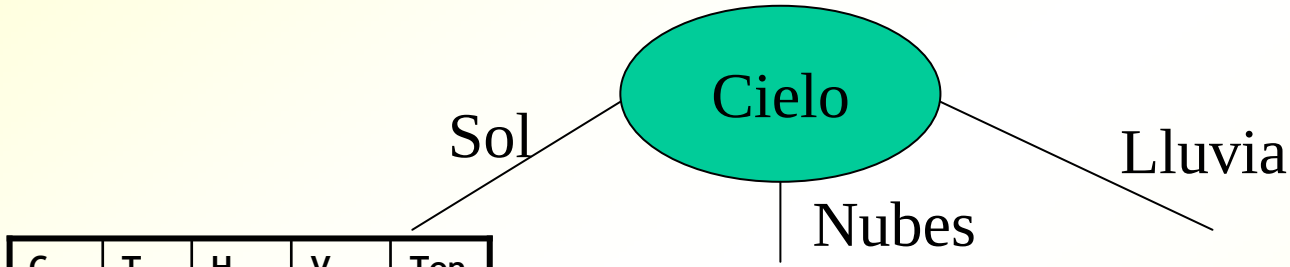






Construcción recursiva del árbol

Ahora que ya tenemos el nodo raíz, el proceso continua recursivamente: hay que construir tres subárboles con los datos que se muestran en cada rama



C	T	H	V	Ten
Sol	85	85	No	No
Sol	80	90	Si	No
Sol	72	95	No	No
Sol	69	70	No	Si
Sol	75	70	Si	Si

C	T	H	V	Ten
Nubes	83	86	No	Si
Nubes	64	65	Si	Si
Nubes	72	90	Si	Si
Nubes	81	75	No	Si

C	T	H	V	Ten
Lluvia	70	96	No	No
Lluvia	68	80	No	Si
Lluvia	75	80	No	Si
Lluvia	65	70	Si	No
Lluvia	71	91	Si	No

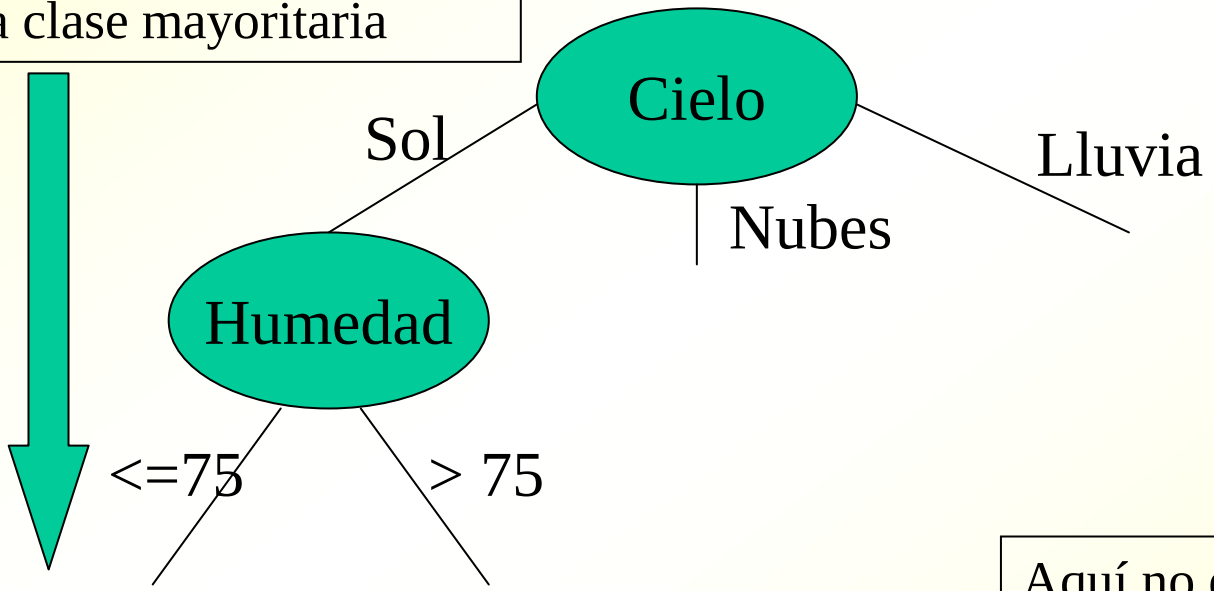
“3 No, 2 Si”

“0 No, 4 Si”

“3 No, 2 Si”

Construcción recursiva del árbol

Aquí un criterio estadístico determina que no merece la pena seguir subdividiendo y se asigna la clase mayoritaria

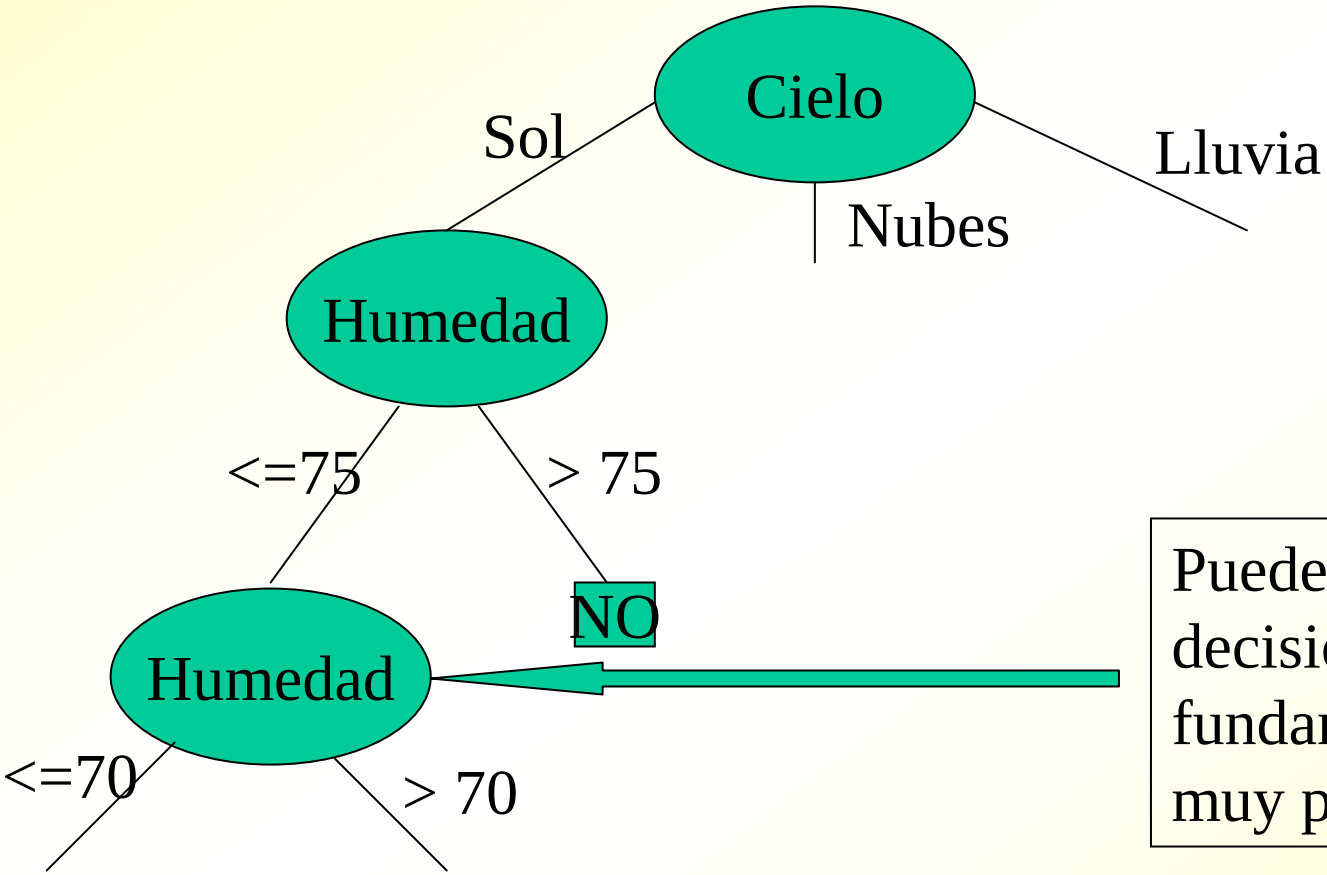


T	H	V	Ten
72	95	No	No
69	70	No	Si
75	70	Si	Si

T	H	V	Ten
85	85	No	No
80	90	Si	No

Aquí no es necesario seguir subdividiendo porque todos los datos son de la misma clase

¿Porqué no seguir subdividiendo?



Puede que esta decisión esté fundamentada en muy pocos datos

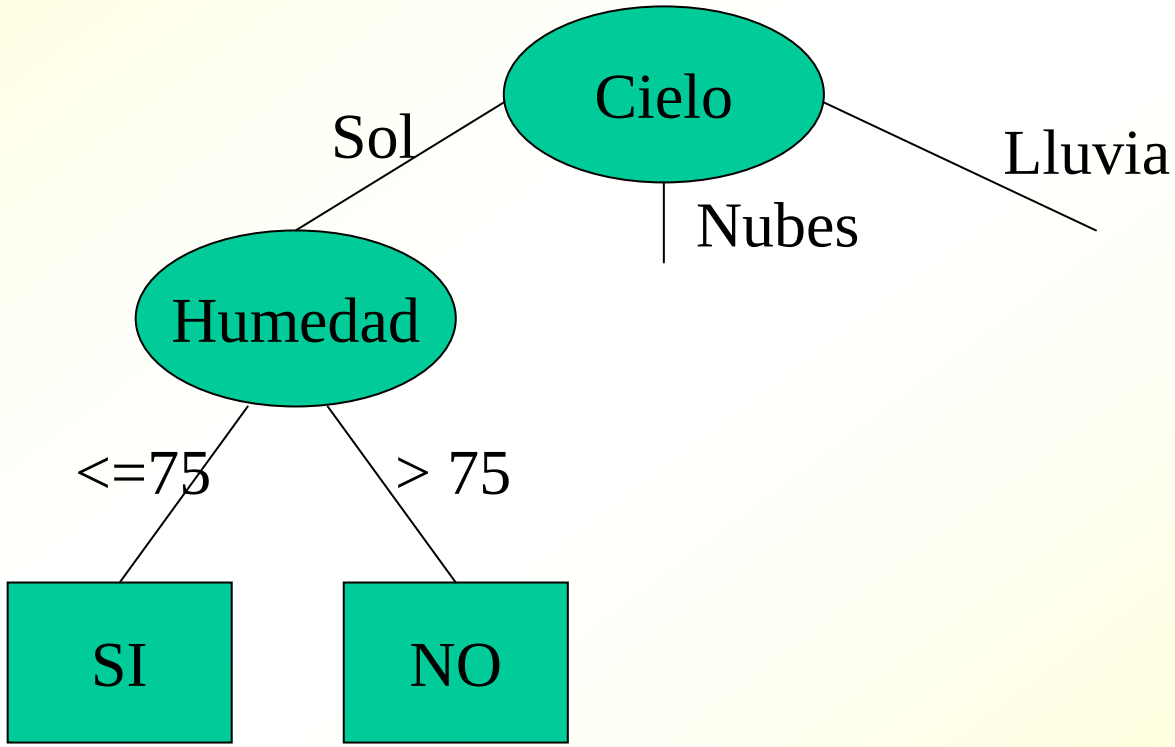
T	H	V	Ten
69	70	No	Si
75	70	Si	Si

T	H	V	Ten
72	95	No	No

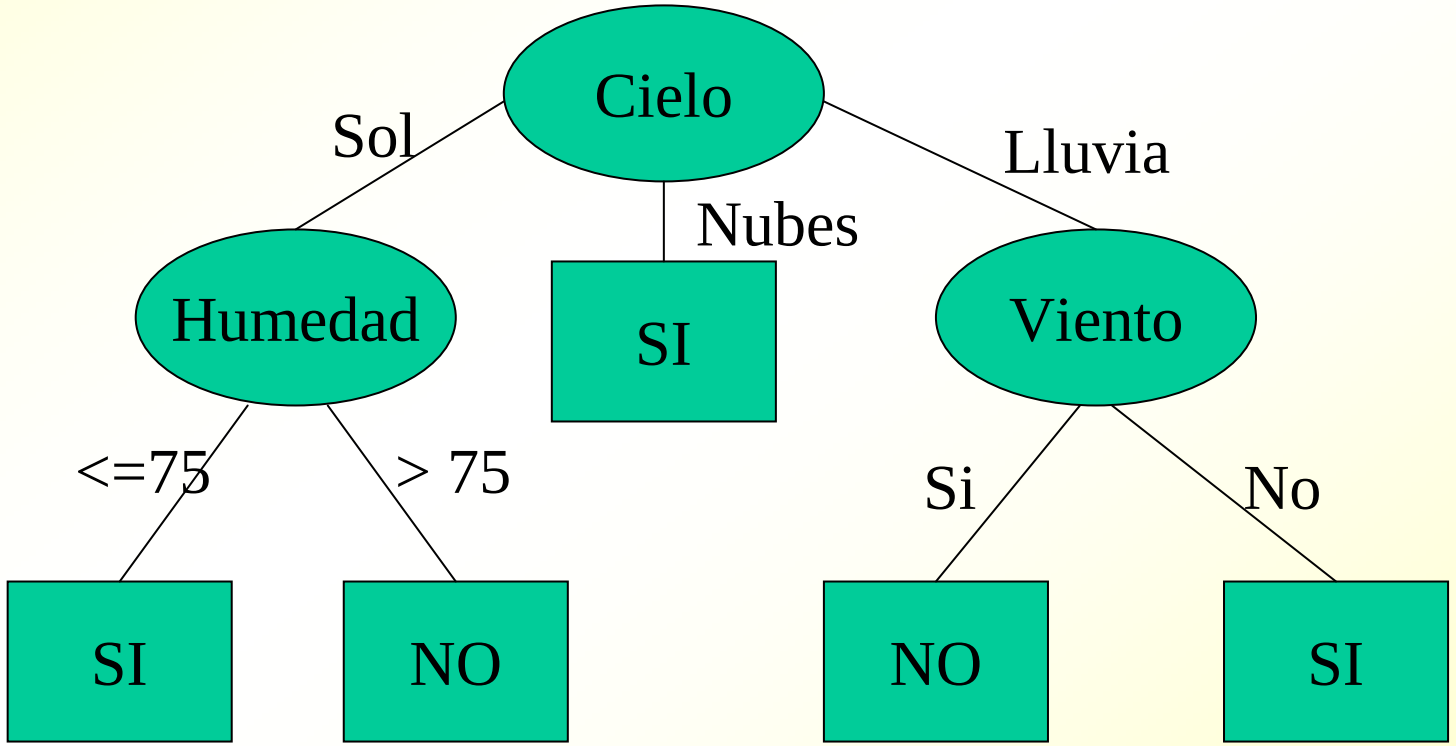
¿Porqué no seguir subdividiendo?

- ¿Hay que detener la construcción cuando tenemos “1 no, 2 si”?
- Tal vez. Cuando hay tan pocos datos (y suponiendo que haya ruido) es posible que el “1 no” haya aparecido por azar, e igualmente podriamos tener “2 no, 2 si”
- Pasamos de una situación en la que hay mayoría de “si” a otra en la que están equiparados con los “no”
- Se puede utilizar algún criterio estadístico para saber si es probable que “1 no, 2 si” se deba al azar
- Cuando se manejan pocos datos (3 en este caso), es bastante probable que las regularidades (humedad<=70 en ete caso) sean sólo aparentes y se deban al azar

Construcción recursiva del árbol

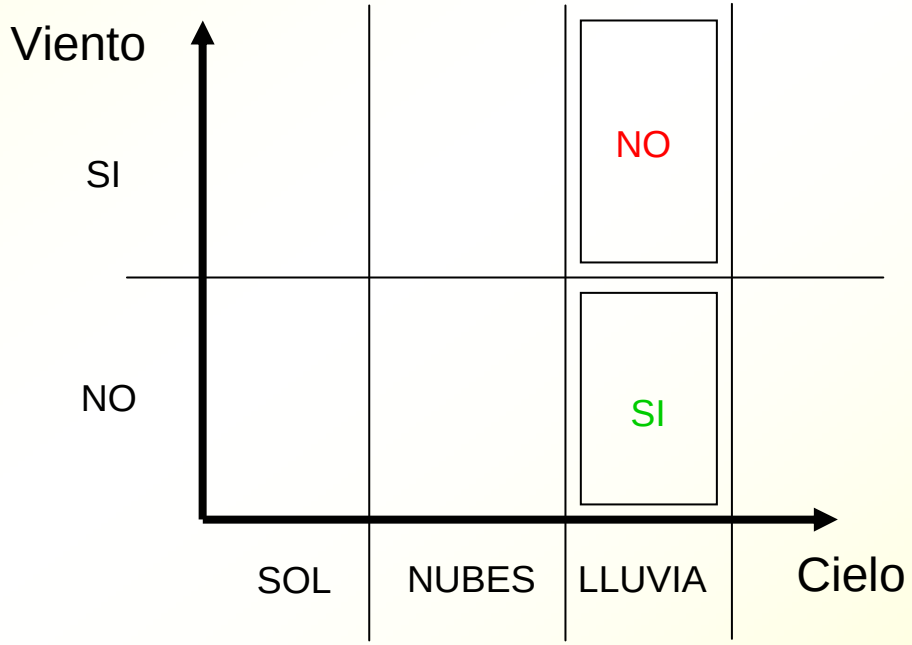


Construcción recursiva del árbol



C4.5 (J48) Tipo de clasificador

- Es no lineal
- Las fronteras de separación entre las clases son rectas paralelas a los ejes



Reglas (para clasificación)

IF Cielo = Sol

Humedad <= 75 **THEN** Tenis = Si

ELSE IF Cielo = Sol

Humedad > 75 **THEN** Tenis = No

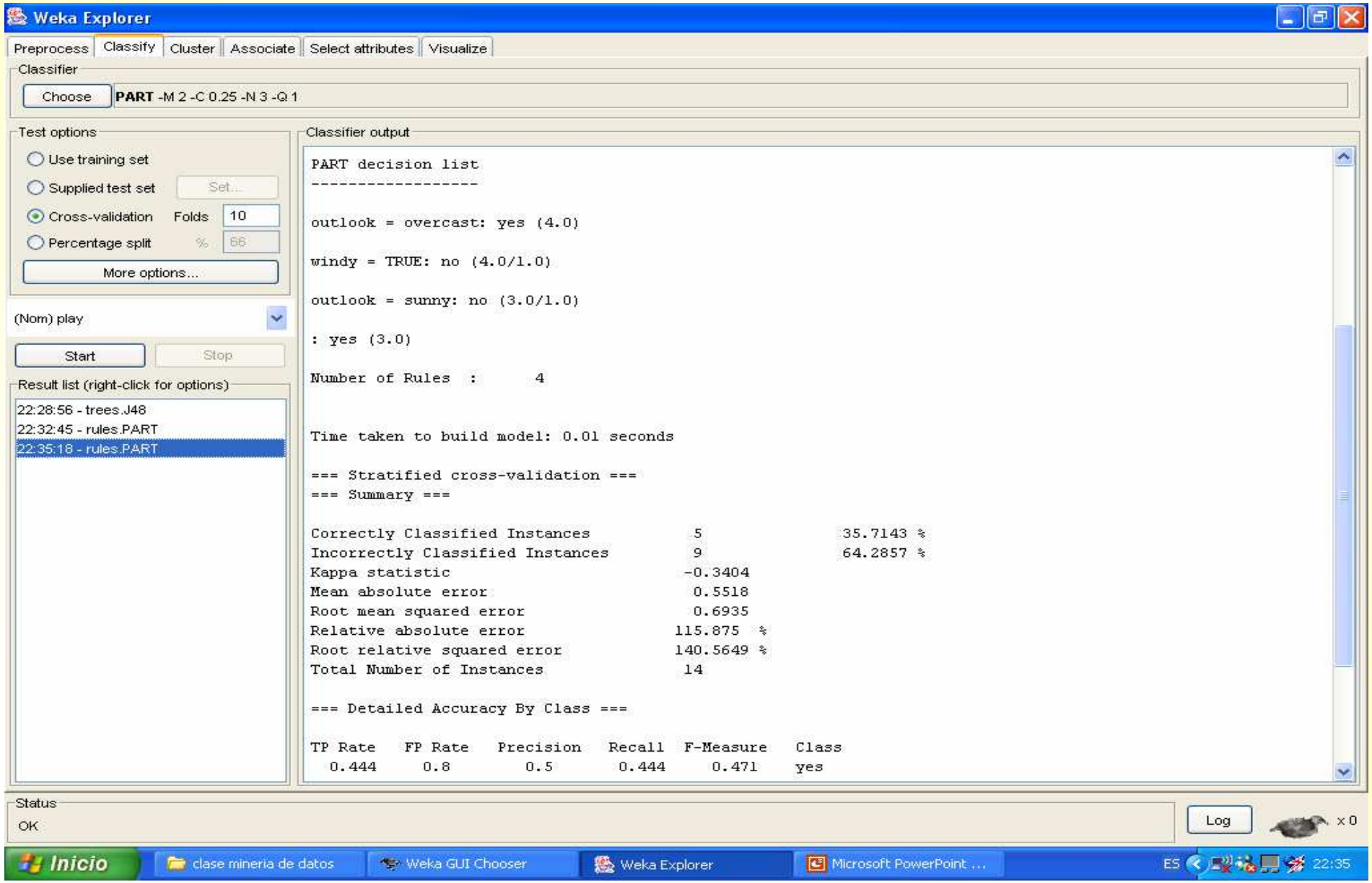
ELSE IF Cielo = Nubes **THEN** Tenis = Si

ELSE IF Cielo = Lluvia

Viento = Si **THEN** Tenis = Si

ELSE Tenis = No

Algoritmo de reglas PART



- Funciones:
  - **Para regresión**: linear regressíon, neural networks
  - Para clasificación: simple logistics, support vector machines (SMO)

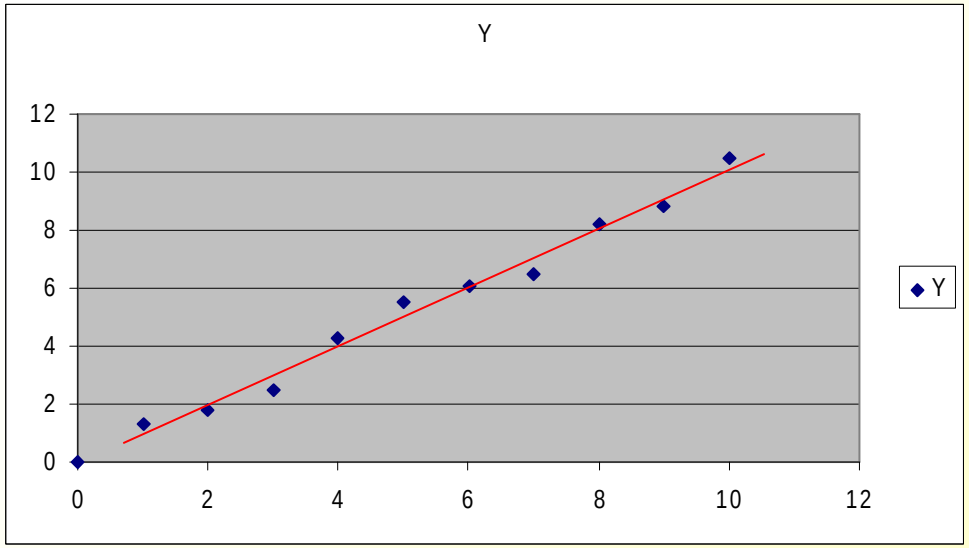
Funciones (para regresión)

X	Y
0	0
1	1,3
2	1,8
3	2,5
4	4,3
5	5,5
6	6,1
7	6,5
8	8,2
9	8,8
10	10,5

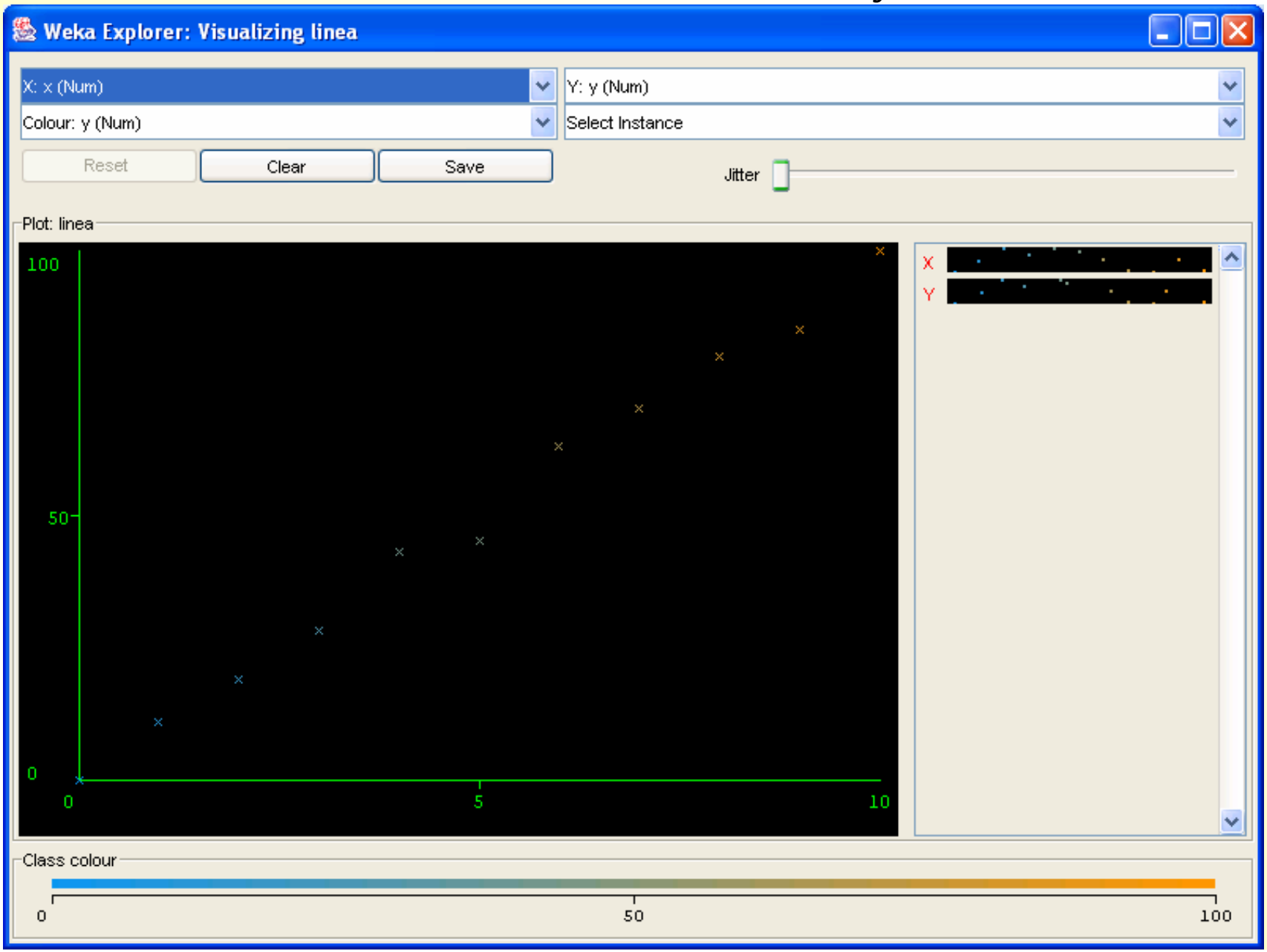
$Y = 1 * X$

Caso general (regresión lineal)

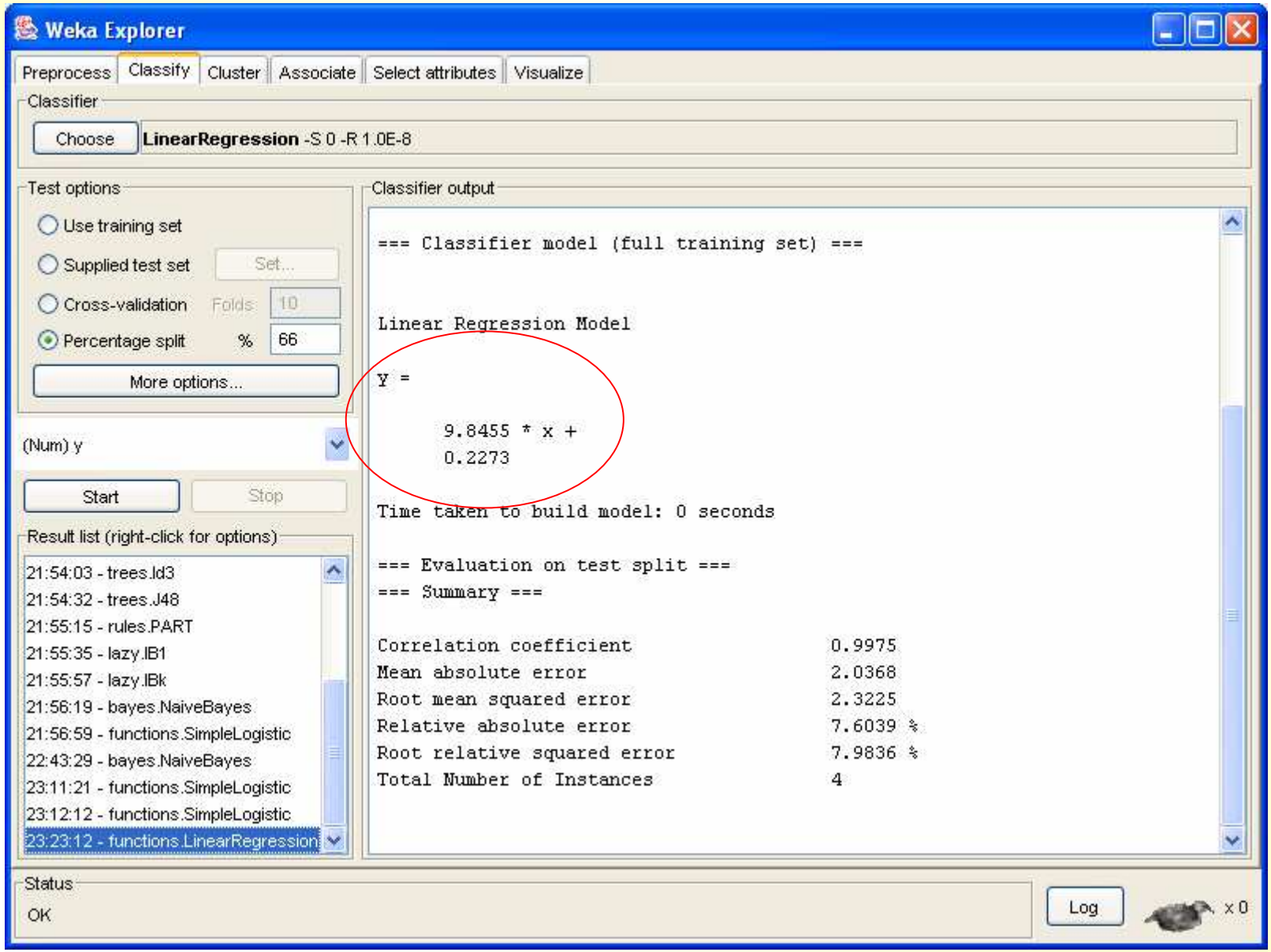
$Y = A1*X1 + A2*X2 + A3*X3 + A4$



Visualización datos linea.arff y=10*x



Resultados regresión lineal



- Funciones:
  - Para regresión: linear regressíon, neural networks
  - **Para clasificación**: simple logistics, support vector machines (SMO)

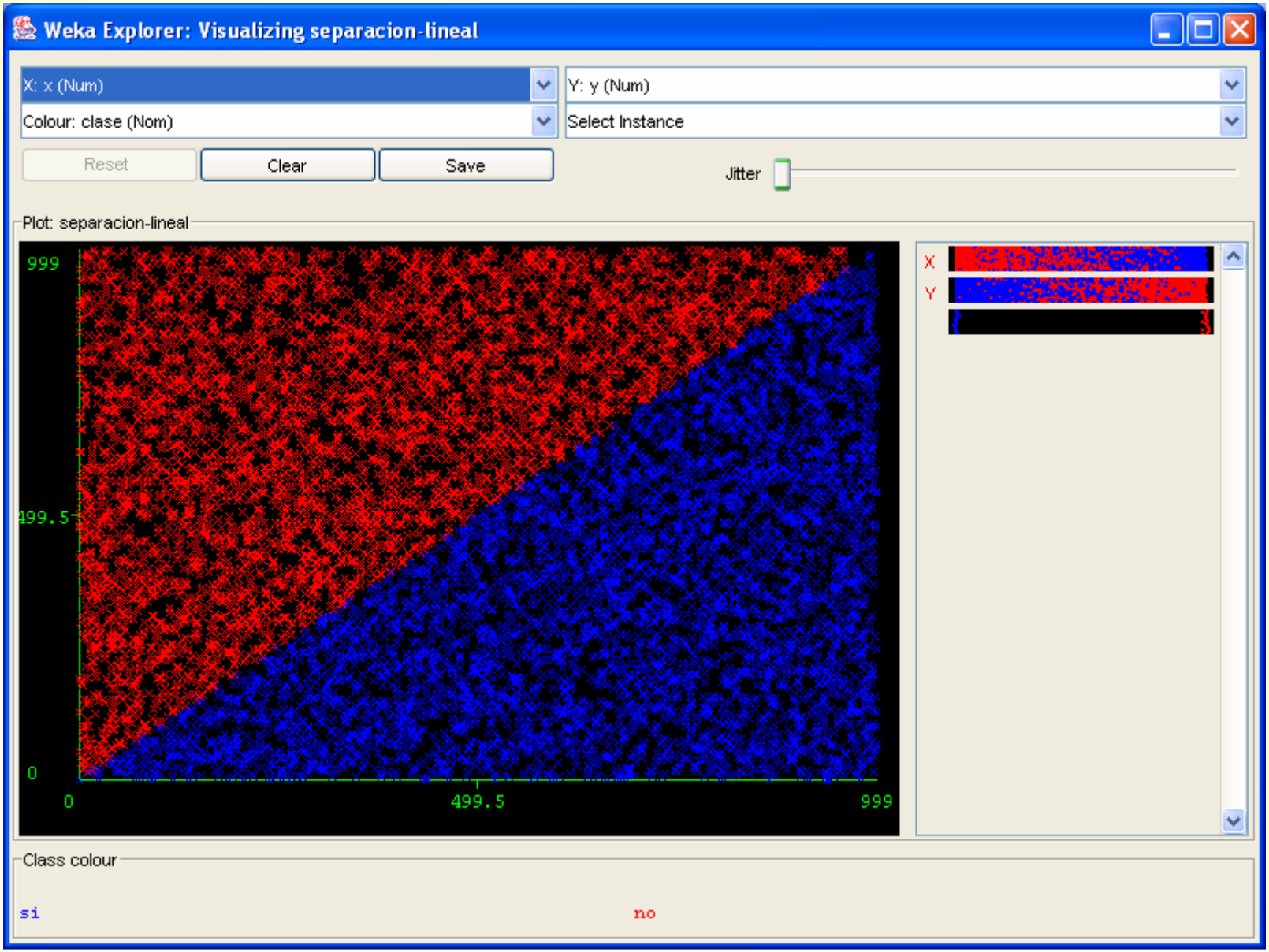
Funciones (para clasificación)

- La salida de las funciones es numérica. Basta con poner un límite. Si lo supera se predice como positivo y si no, como negativo.

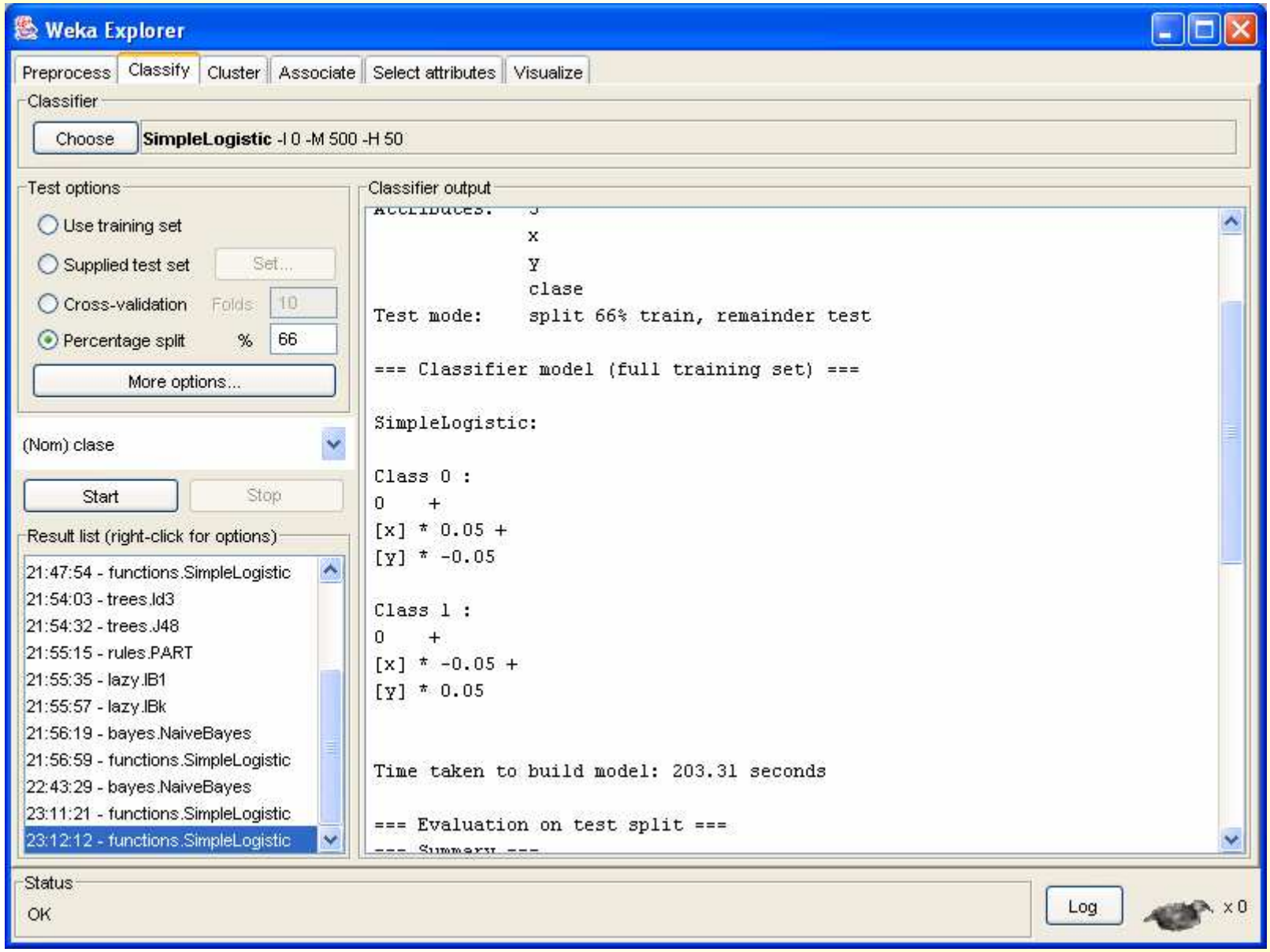
IF **F(X1, X2, X3, ...)** > 0.5 THEN Clase = Si

ELSE Clase = No

Visualización datos separables linealmente

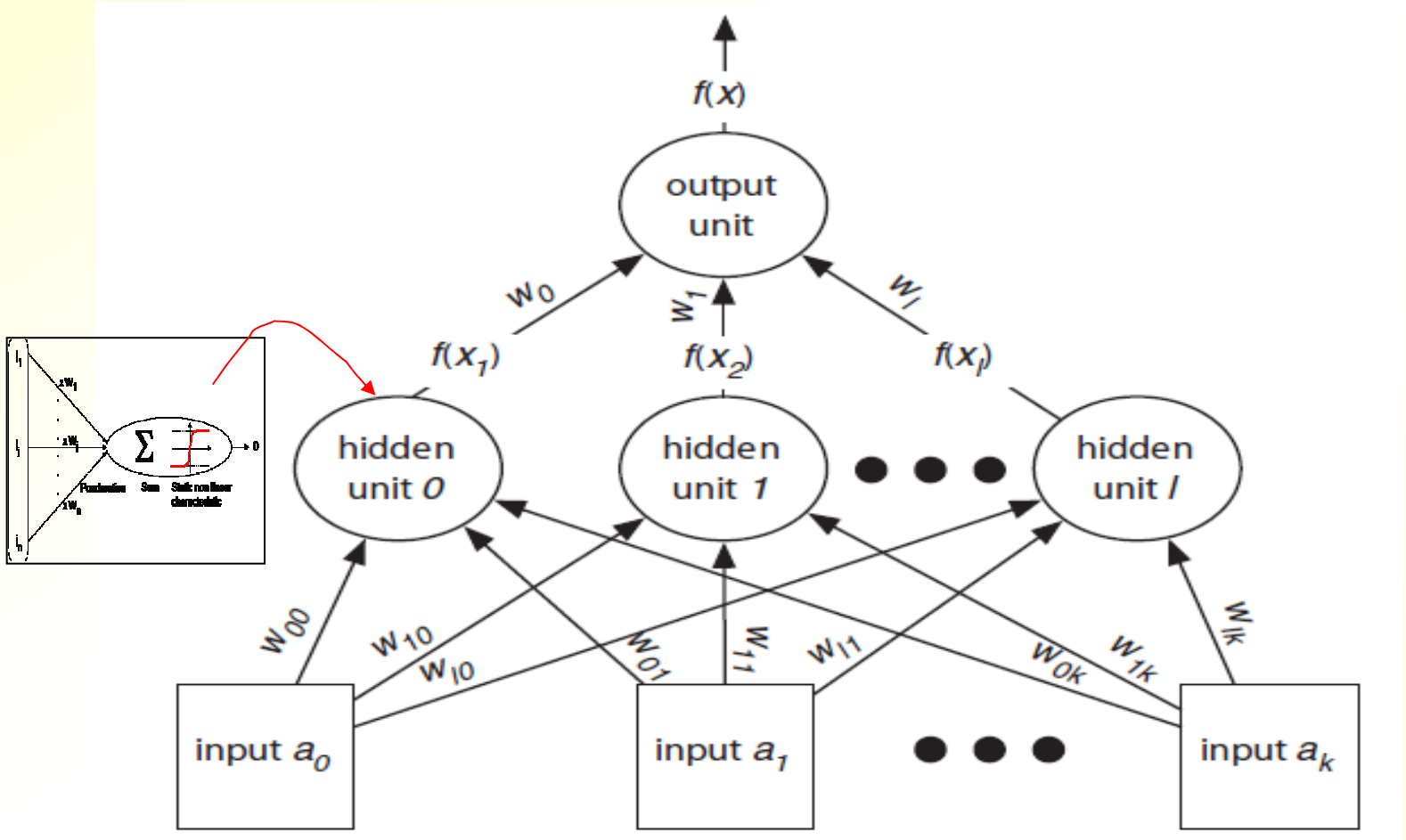


Algoritmo simple logistics (separación lineal)

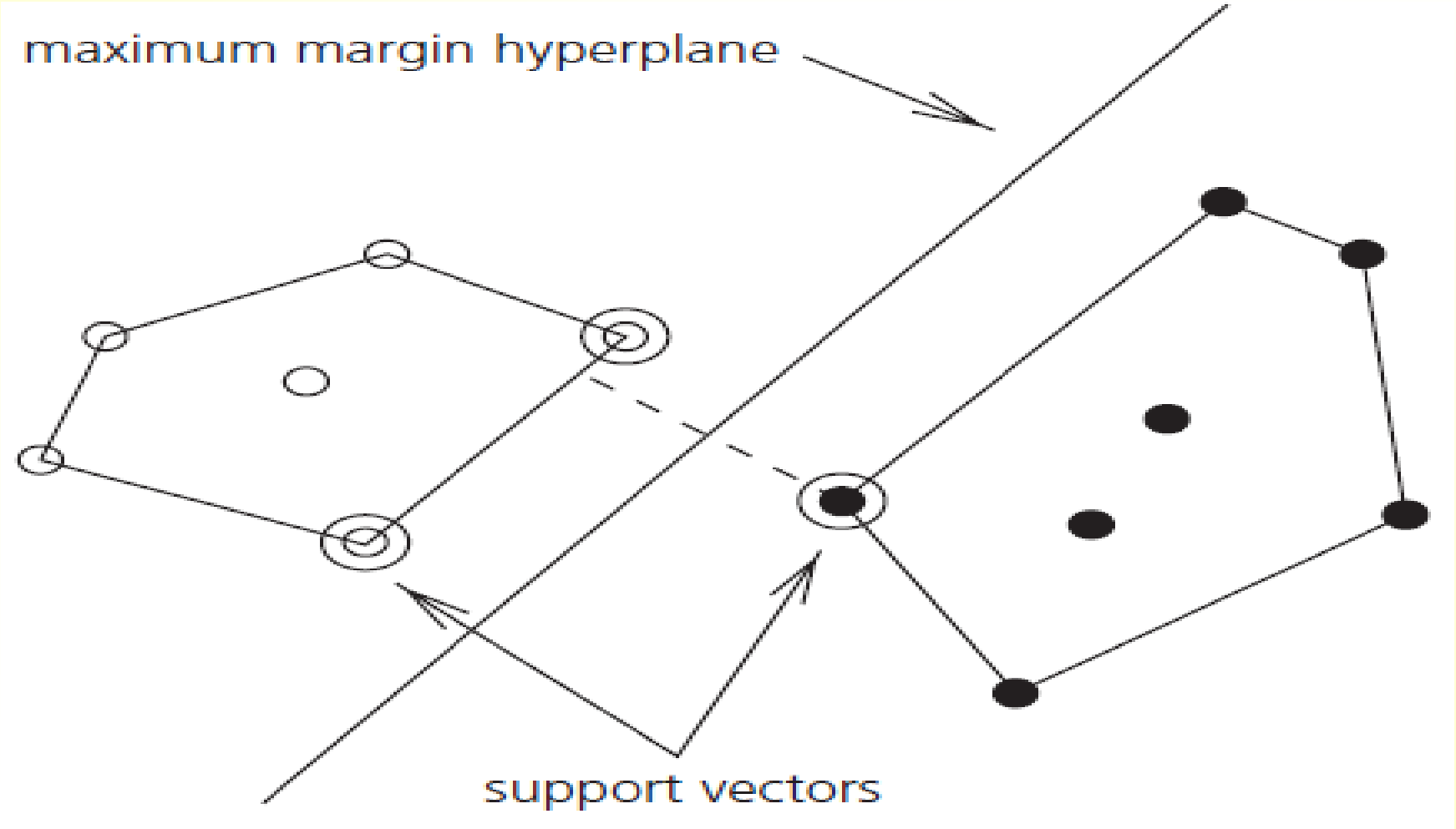


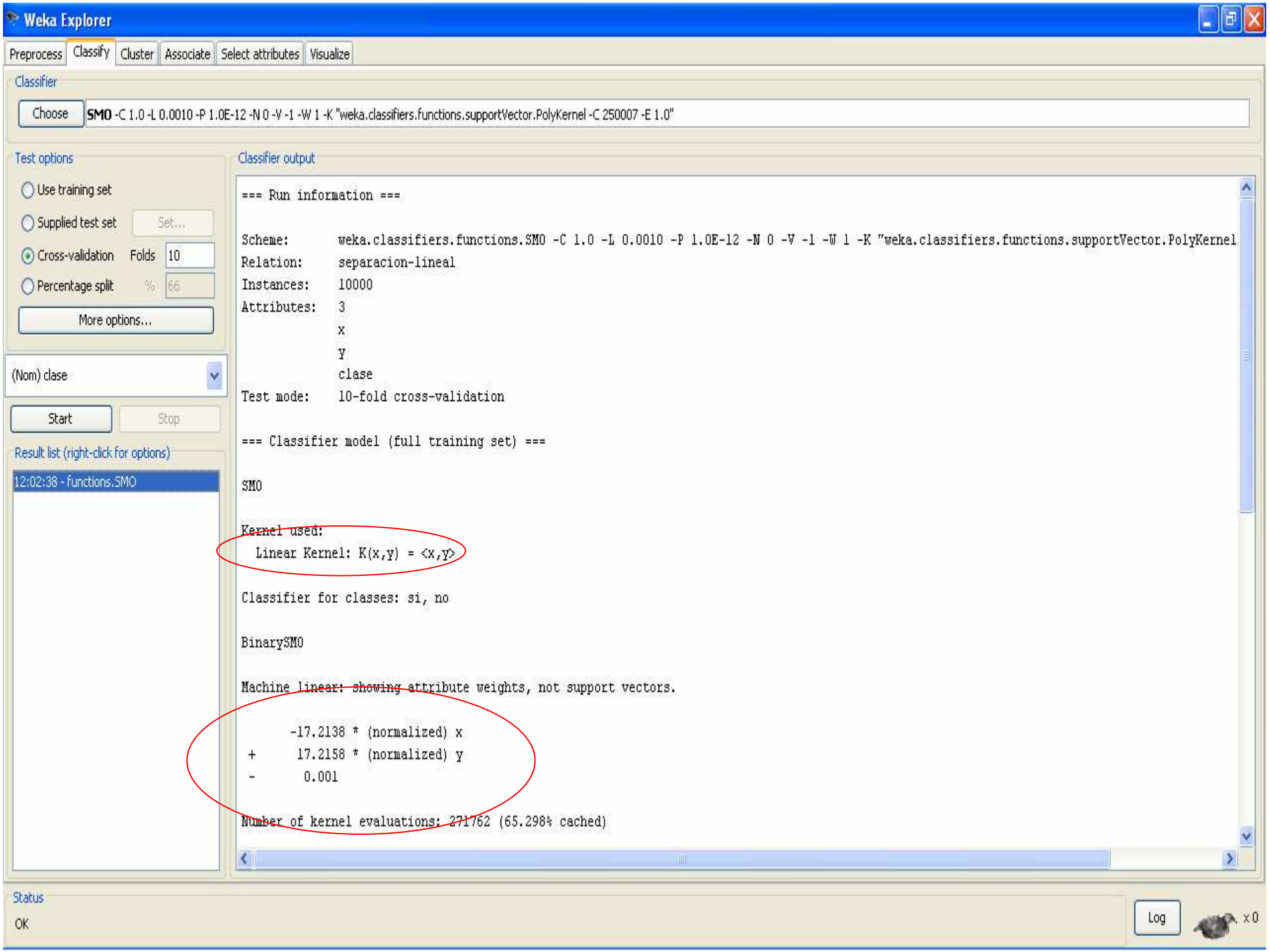
Redes de Neuronas (Multilayer perceptron)

El grado de no linealidad depende del número de neuronas ocultas



Máquinas de vectores de soporte (SVM)




Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
SMO -C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) clase
Start
Stop
Result list (right-click for options)
12:02:38 - functions.SMO
Classifier output
=== Run information ===
Scheme: weka.classifiers.functions.SMO -C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel
Relation: separacion-lineal
Instances: 10000
Attributes: 3
x
y
clase
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
SMO
Kernel used:
Linear Kernel: K(x,y) = <x,y>
Classifier for classes: si, no
BinarySMO
Machine linear: showing attribute weights, not support vectors.
-17.2138 * (normalized) x
+ 17.2158 * (normalized) y
- 0.001
Number of kernel evaluations: 271762 (65.298% cached)
Status
OK
Log
x 0

Tipo de clasificador SVM

- Depende del kernel:
  - Lineal (con kernel polinonial con exponente 1)
  - No lineal:
    - Con kernel polinomial con exponente 2
    - Con kernel RBF

- Árboles de regresión:
  - LMT (M5), ...

Algoritmos de clasificación / regresión (predicción)

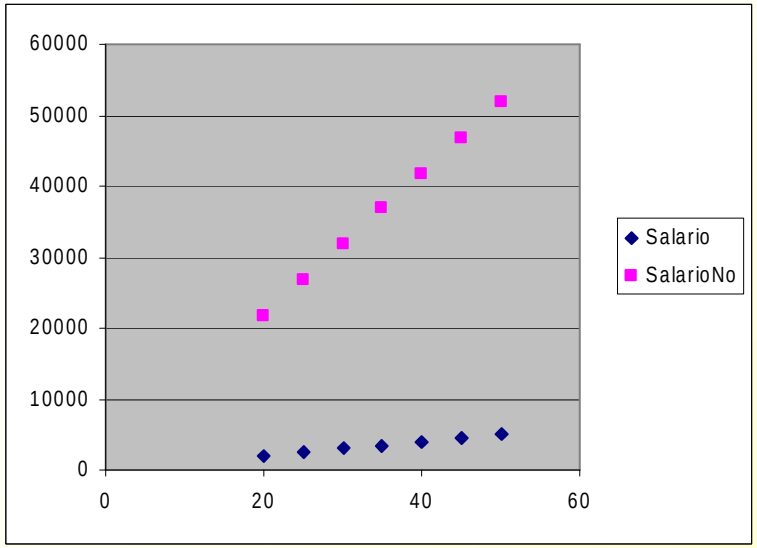
- Árboles de decisión y reglas. Para clasificación
  - Árboles de decisión: ID3, C4.5 (J48), ...
  - Reglas: PART, CN2, AQ, ...
- Funciones:
  - Para regresión: linear regressíon, neural networks
  - Para clasificación: simple logistics, support vector machines (SMO)
- **Árboles de regresión: LMT (M5), ...**
- Técnicas perezosas. Para clasificación y regresión
  - IB1, IBK, ...
- Técnicas Bayesianas. Para clasificación:
  - Naive Bayes
- Metatécnicas. Para clasificación y regresión:
  - Boosting, Bagging, Stacking, Random Forests

Árboles de regresión

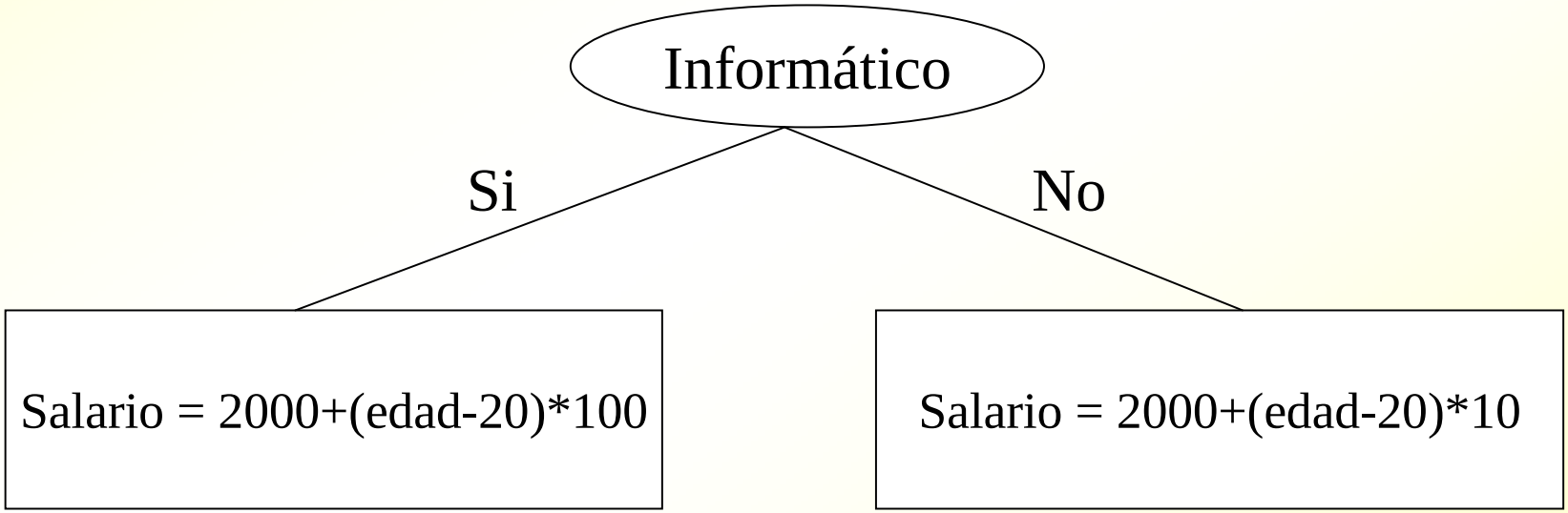
- ¿Y si tenemos atributos nominales y numéricos y queremos predecir cantidades numéricas (regresión)?
- Usar árboles de regresión: tienen funciones (regresión lineal) en las hojas

Árboles de regresión. Ejemplo

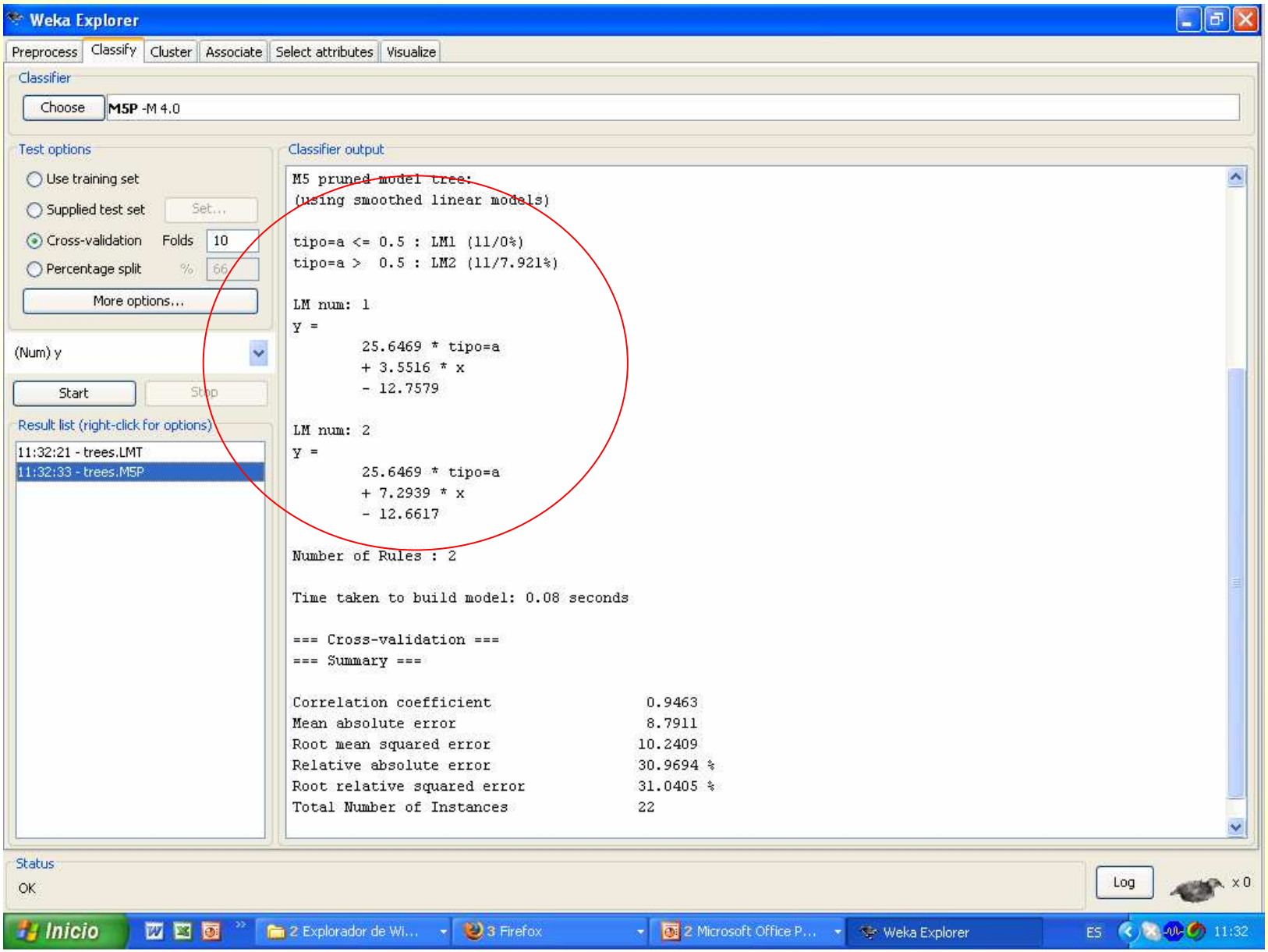
Informático	Edad	Salario
Si	20	2000
Si	25	2500
Si	30	3000
Si	35	3500
Si	40	4000
Si	45	4500
Si	50	5000
No	20	2000
No	25	2050
No	30	2100
No	35	2150
No	40	2200
No	45	2250
No	50	2300



Árboles de regresión. Ejemplo



Algoritmo LMT

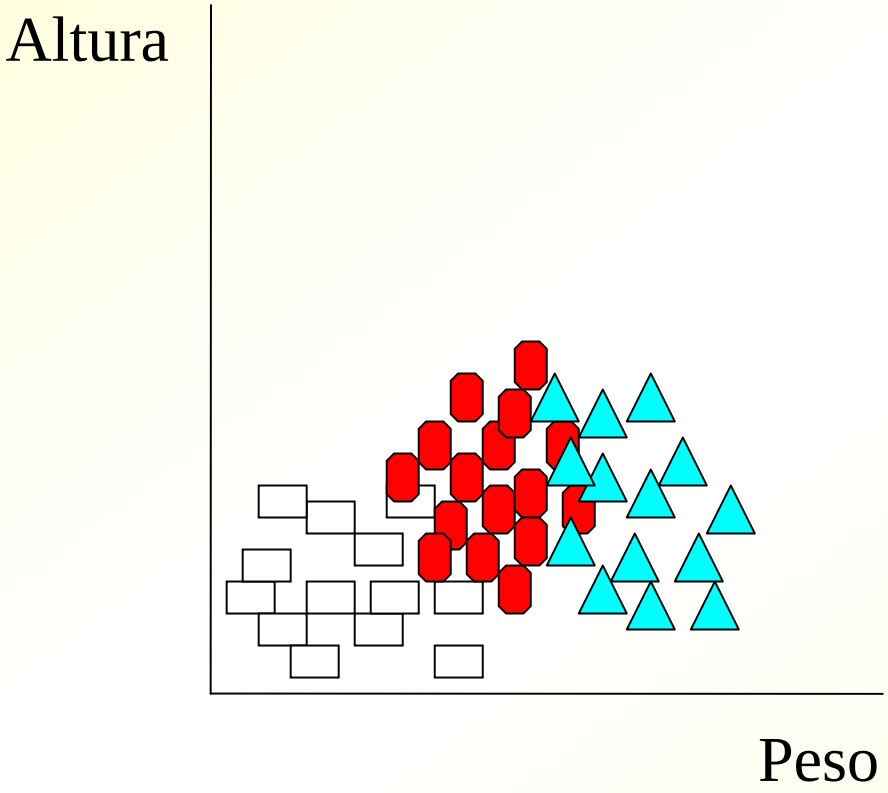


- Técnicas perezosas. Para clasificación y regresión
  - IB1, IBK, ...

Técnicas "perezosas" (almacenar instancias)

- También llamadas técnicas basadas en instancias (o en ejemplos)
- En lugar de construir una estructura predictora (árbol de decisión, reglas, ...), **simplemente** se guardan las instancias (los datos) o representantes de los mismos
- Para clasificar un nuevo dato, simplemente se busca(n) la(s) instancia(s) más "parecida(s)" o cercana(s)
- Parecido a lo que hacen las personas: para resolver un nuevo problema, intentan recordar el caso más parecido que ya sepan resolver
- Ejemplo: sistema legal anglosajón

Técnicas perezosas (clasificación)



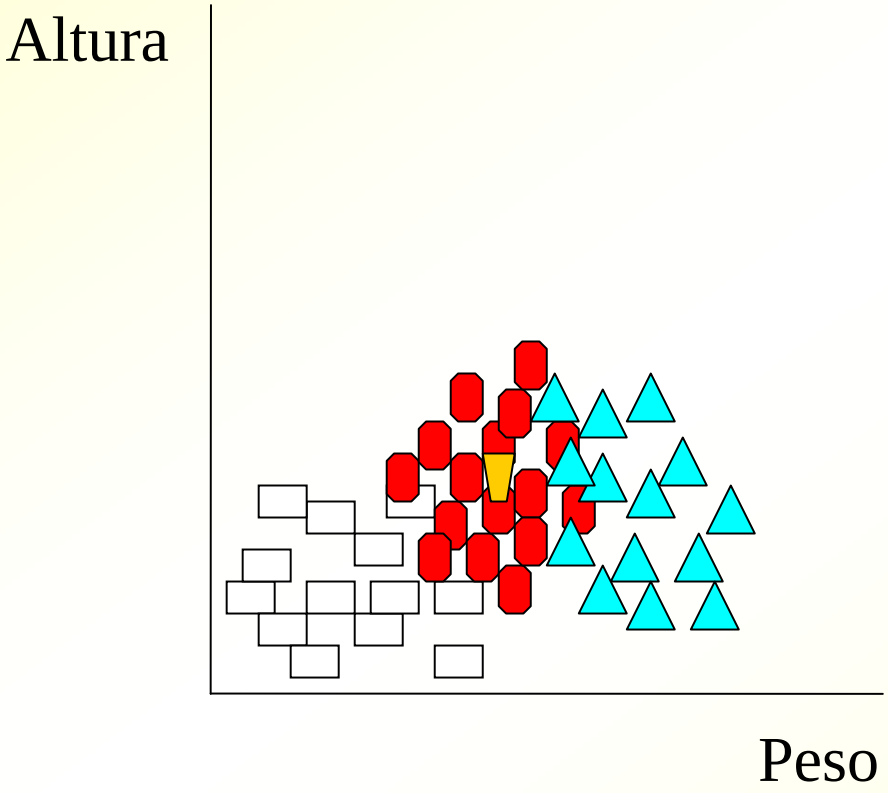
Niño

Adulto

Mayor

Se guardan todos los ejemplos

Técnicas perezosas (clasificación)



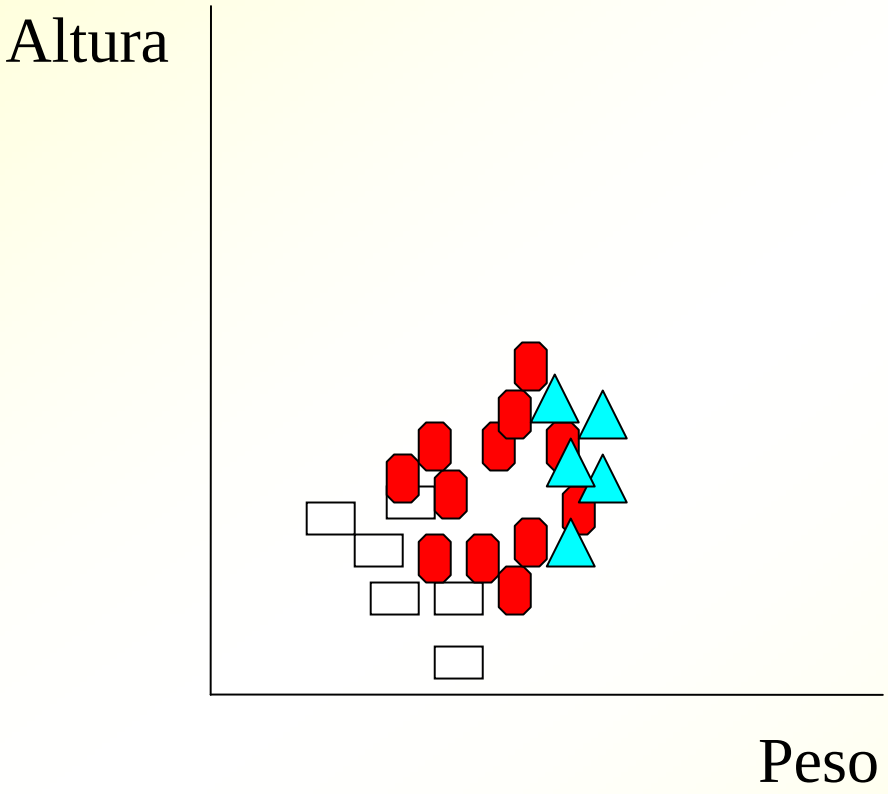
Niño

Adulto

Mayor

Se guardan todos los ejemplos

Técnicas perezosas (clasificación)



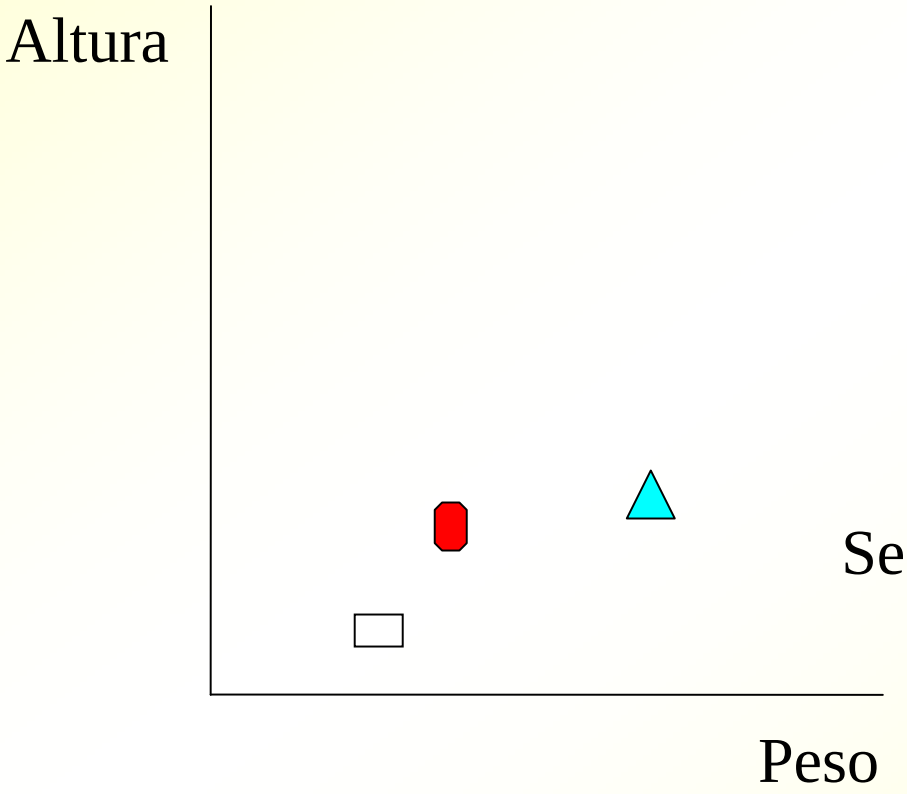
Niño

Adulto

Mayor

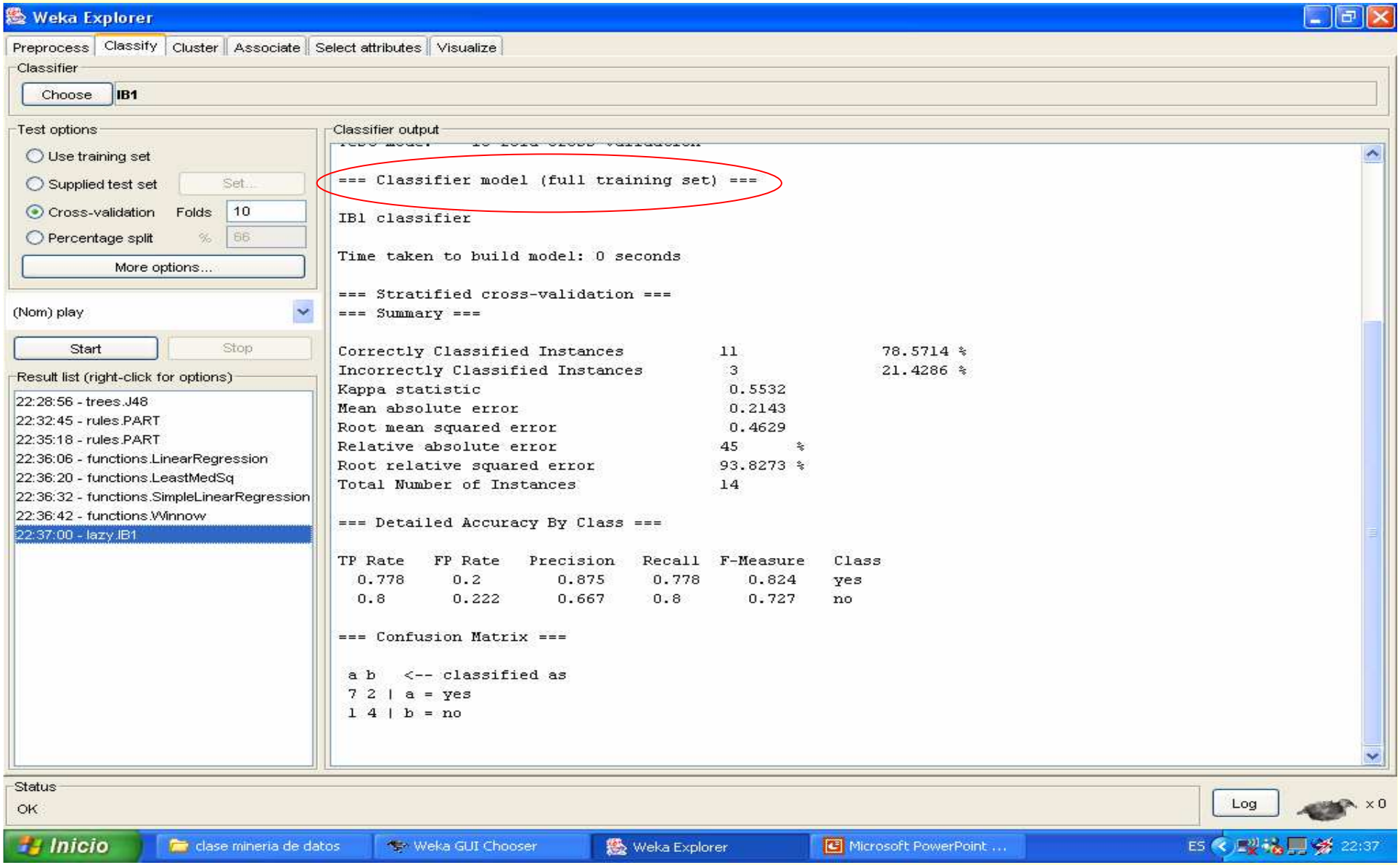
Se guardan las fronteras

Técnicas perezosas (clasificación)



Se guardan un representante por clase (centroide o prototipo)

Algoritmo perezoso IB1 (1 vecino) e IBK (k vecinos)



Técnicas perezosas en regresión

- Ejemplo: predicción de la carga de electricidad según la hora y la temperatura
- Ahora se trata de un problema de regresión
- Simplemente se cogen las n instancias más cercanas y se calcula la media entre ellas
- El IBK permite hacer regresión (el IB1 no)

- Técnicas Bayesianas. Para clasificación:
  - Naive Bayes

Técnicas Bayesianas (almacenar probabilidades)

- (sol, frío, alta, si, clase=????)
- ¿Pr(Tenis = si / cielo = sol, temperatura = frío, humedad = alta, viento = si)?
- ¿Pr(Tenis = no / cielo = sol, temperatura = frío, humedad = alta, viento = si)?
- Naive Bayes:

Pr(si / ...) ~ pr(cielo = sol/si) * pr(humedad = alta /si) *
pr(viento = si /si) * pr(si)

Teorema de Bayes y Naive Bayes

- Pr(A|B) = k*Pr(B|A)*P(A)
- Pr(Tenis = si / cielo = sol, temperatura = frío, humedad = alta, viento = si)
- = k * Pr(cielo = sol, temperatura = frío, humedad = alta, viento = si / Tenis = si) * Pr(Tenis = si)
- Y si suponemos a todos los atributos independientes
- = Pr(si/cielo = sol, temperatura = frío, humedad = alta, viento = si)

  = k* pr(cielo = sol/si) * pr(humedad = alta /si) * pr(viento = si /si) * Pr(si)
- Esto implica que el que haga sol es independiente de la humedad (lo que no es cierto, pero suele funcionar)
- En suma, que podemos calcular Pr(si/...) a partir de:
  - pr(cielo = sol/si) = numero de días soleados y buenos para el tenis dividido por el número de días buenos para el tenis
  - pr(humedad = alta /si)
  - pr(viento = si /si)
  - pr(si) = número de días buenos para el tenis dividido por el número de días totales

Datos de entrada

Día	Cielo	Temperatura	Humedad	Viento	Tenis
1	Soleado	Caliente	Alta	No	No
2	Soleado	Caliente	Alta	Si	No
3	Nublado	Caliente	Alta	No	Si
4	Lluvioso	Templado	Alta	No	Si
5	Lluvioso	Frio	Normal	No	Si
6	Lluvioso	Frio	Normal	Si	No
7	Nublado	Frio	Normal	Si	Si
8	Soleado	Templado	Alta	No	No
9	Soleado	Frio	Normal	No	Si
10	Lluvioso	Templado	Normal	No	Si
11	Soleado	Templado	Normal	Si	Si
12	Nublado	Templado	Alta	Si	Si
13	Nublado	Caliente	Normal	No	Si
14	Lluvioso	Templado	Alta	Si	No

Datos de entrada ordenados

Cielo	Temperatura	Humedad	Viento	Tenis
Soleado	Frio	Normal	No	Si
Soleado	Templado	Normal	Si	Si
Nublado	Frio	Normal	Si	Si
Nublado	Caliente	Alta	No	Si
Nublado	Templado	Alta	Si	Si
Nublado	Caliente	Normal	No	Si
Lluvioso	Templado	Alta	No	Si
Lluvioso	Frio	Normal	No	Si
Lluvioso	Templado	Normal	No	Si
Soleado	Caliente	Alta	No	No
Soleado	Templado	Alta	No	No
Soleado	Caliente	Alta	Si	No
Lluvioso	Frio	Normal	Si	No
Lluvioso	Templado	Alta	Si	No

Técnicas Bayesianas. Ejemplo

P(Cielo/Tenis)

Cielo	Si	No
Sol	**2/9**	3/5
Nubes	4/9	0/5
Lluvia	3/9	2/5

P(Temp/Tenis)

Temperatura	Si	No
Caliente	2/9	2/5
Templado	4/9	2/5
Frio	**3/9**	1/5

P(Hum/Tenis)

Humedad	Si	No
Alta	**3/9**	4/5
Normal	6/9	1/5

Viento	Si	No
Si	**3/9**	3/5
No	6/9	2/5

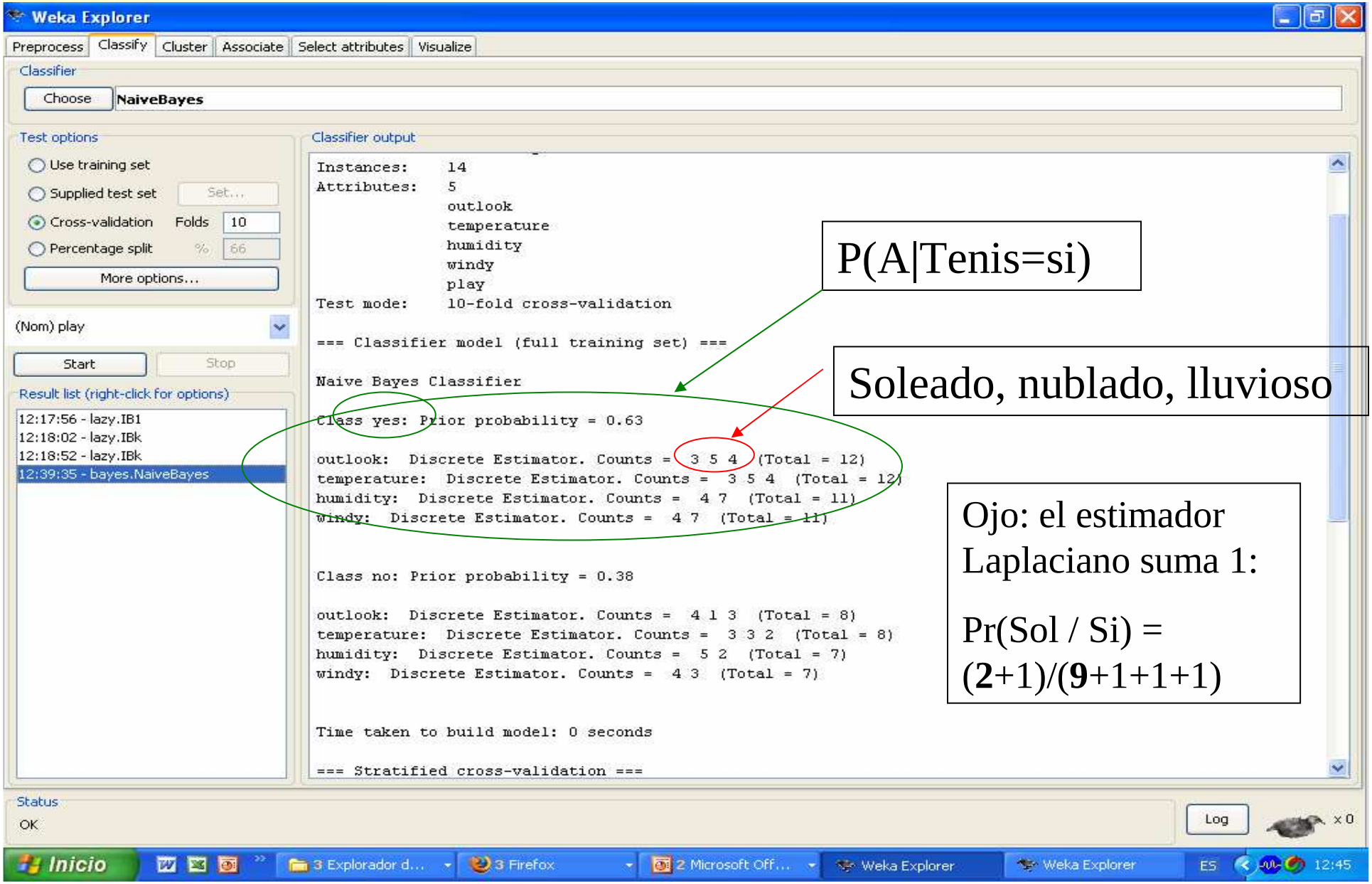
P(Tenis)

Tenis	Si	No
	9/14	5/14

- Pr(si / sol, frío, alta, si) ~ 2/9 * 3/9 * 3/9 * 3/9 * 9/14 = 0.0053
- Pr(no / sol, frío, alta, si) ~ 3/5 * 1/5 * 4/5 * 3/5 * 5/14 = **0.0206**

- Pr (si / ...) = 0.0053 / (0.0053+ 0.0206) = 20.5%
- Pr (no / ...) = 0.0206 / (0.0053+ 0.0206) = 79.5%

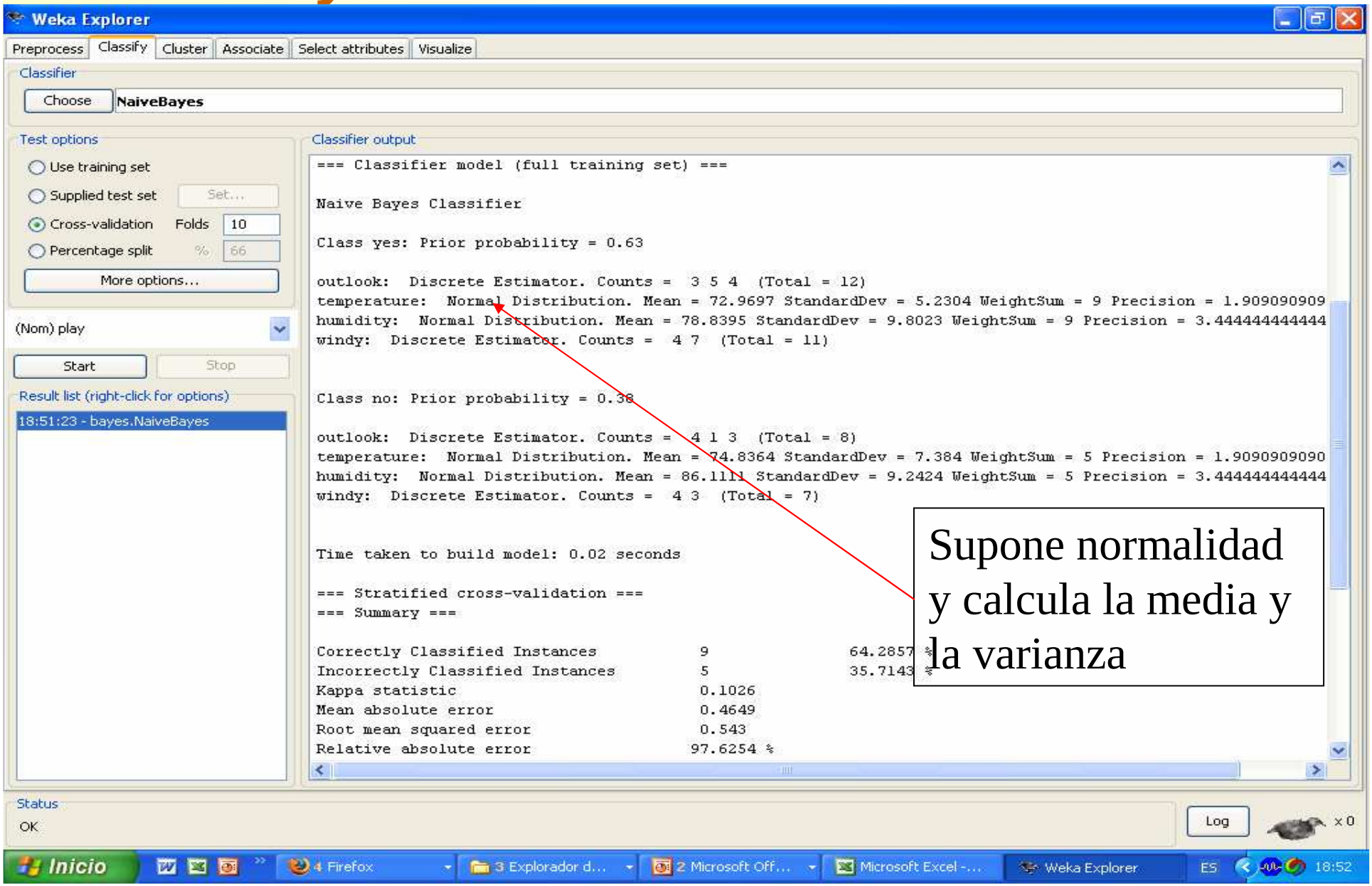
Algoritmo Naive Bayes



Naive Bayes con atributos numéricos

Cielo	Temperatura	Humedad	Viento	Tenis
Sol	85	85	No	No
Sol	80	90	Si	No
Nublado	83	86	No	Si
Lluvia	70	96	No	So
Lluvia	68	80	No	Si
Nublado	64	65	Si	Si
Sol	72	95	No	No
Sol	69	70	No	Si
Lluvia	75	80	No	Si
Sol	75	70	Si	Si
Nublado	72	90	Si	Si
Nublado	81	75	No	Si
Lluvia	71	91	Si	No

Naive Bayes con atributos numéricos



- Metatécnicas. Para clasificación y regresión:
  - Boosting, Bagging, Stacking, Random Forests

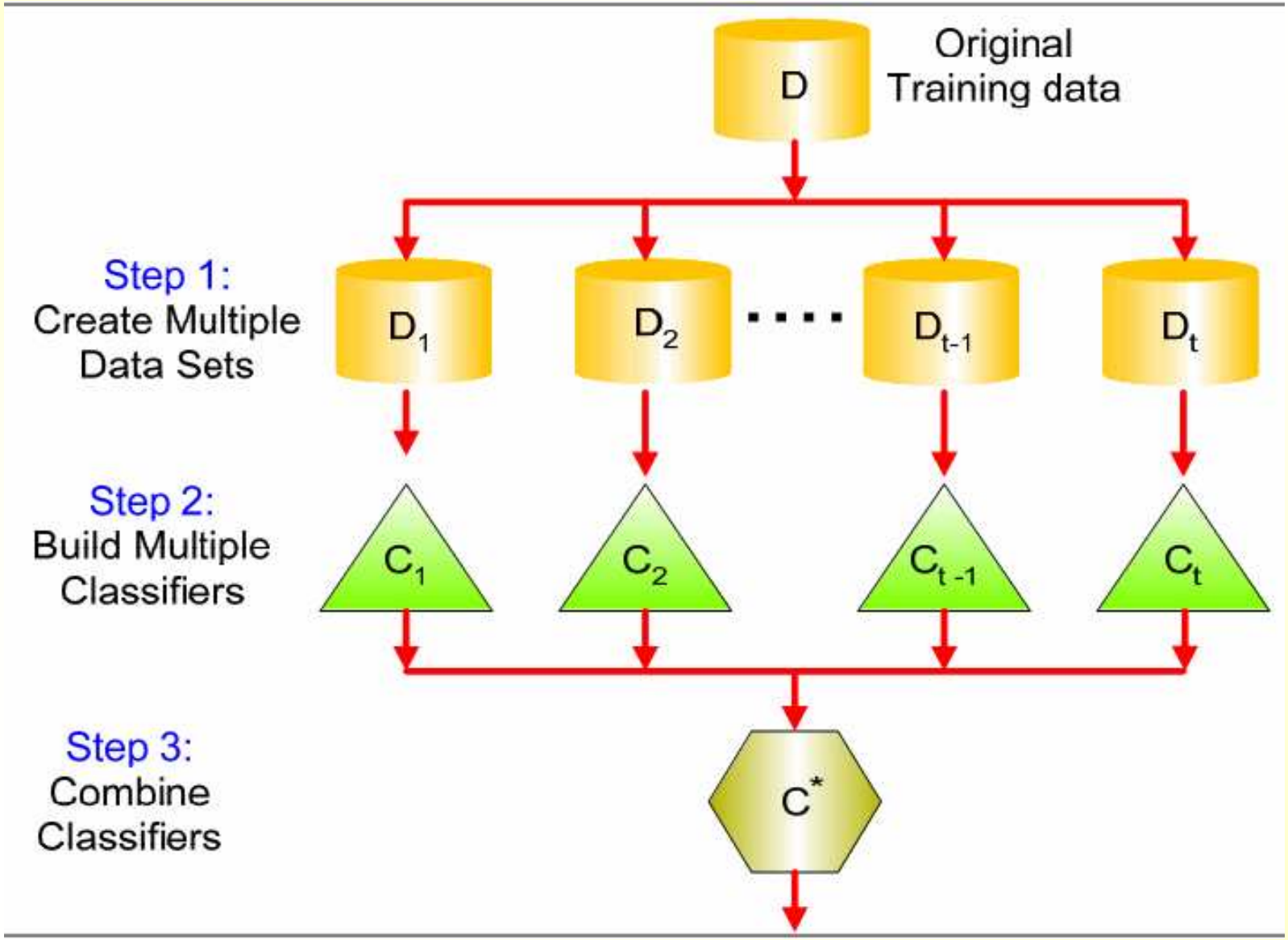
Meta-algoritmos o conjuntos de clasificadores (ensembles of classifiers)

- Construyen varios predictores (clasificación o regresión) y después los usan de manera conjunta
- Suelen ser mas precisos que los algoritmos individuales, siempre que los algoritmos base predigan mejor que el azar
- La idea es que si los distintos clasificadores no están correlacionados en los errores, el uso conjunto de todos ellos será mejor que usar cualquiera de ellos por separado
- Tipos principales:
  - Bagging: construye varios modelos con el mismo algoritmo (varios árboles de decisión, por ejemplo) y cuando llega un dato de test, la clase se decide por votación. Menos sensible al ruido.
  - Boosting: construye varios modelos de manera secuencial, cada uno se centra en los datos que el anterior clasificaba mal. Funciona muy bien, pero es sensible al ruido en los datos.
  - Stacking: usa la salida de clasificadores heterogéneos y un metaclasificador
  - Random Forests: Crea un ensemble de varios árboles de decisión

Bagging (Bootstrap aggregating)

- Justificación: un algoritmo de aprendizaje automático genera clasificadores distintos si le pasamos datos de entrenamiento distintos
- Si el algoritmo es inestable, pequeñas diferencias en los datos de entrenamiento darán lugar a clasificadores muy distintos
  - Inestables: redes de neuronas, árboles de decisión, árboles de regresión, decision stumps (árboles con un solo nodo), …
  - Estables: vecino más cercano (IB1, IBK, …)
- Solución: generar muchos conjuntos de entrenamiento y entrenar con cada uno de ellos un clasificador. La clase del clasificador agregado se decidirá por votación
- Los diferentes conjuntos de entrenamiento se generan a partir del conjunto de entrenamiento original por medio de muestreo aleatorio.

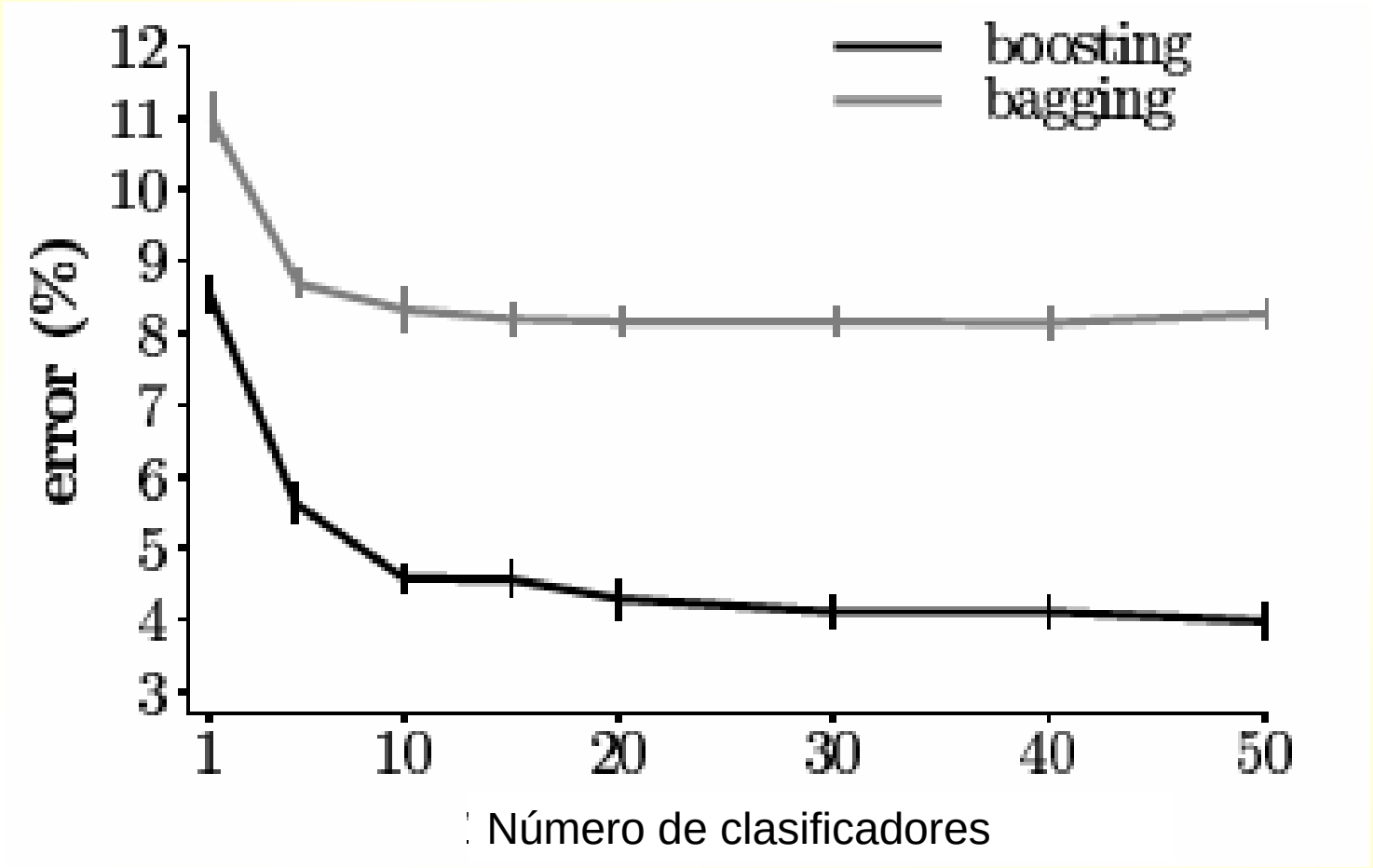
Bagging (Bootstrap aggregating)



Randomization

- Nota: también se pueden crear conjuntos de clasificadores generando distintos clasificadores a partir del mismo conjunto de entrenamiento, mediante randomización
- Ej: en J49 (C4.5) siempre se elige el mejor atributo para cada nodo. En lugar de eso, elegir para cada nodo un atributo elegido aleatoriamente de entre los 5 mejores. Cada vez que ejecutemos J48, se creará un clasificador distinto, incluso partiendo de los mismos datos

Bagging y descenso del error



¿Porqué funciona?

- Supongamos que hay 25 clasificadores
- Cada clasificador tiene un error ε=0.35
- Si los errores de los clasificadores son independientes o no correlacionados (es decir, si no se equivocan a la vez)
- El error del clasificador conjunto será:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

- Pero es difícil conseguir clasificadores no correlacionados

Adaboost (boosting)

- Al igual que Bagging, Boosting se basa en entrenar varios clasificadores con distintas muestras de entrenamiento
- Pero en Boosting, las muestras se construyen de manera secuencial
- Los datos de entrenamiento son una lista de tuplas $\{(x_1,y_1), \ldots, (x_a,y_a), \ldots, (x_d,y_d)\}$
- Cada dato tiene un peso w_a, inicialmente todos $w_a=1/d$
- Los pesos se irán adaptando, de manera que los datos difíciles tendrán mas peso y los más fáciles, menos
- Los pesos se pueden utilizar mediante remuestreo, o bien hay algoritmos (como J48 o NN) que pueden utilizar directamente datos con pesos

Adaboost (boosting)

1. Inicialmente, todos los datos de entrenamiento tienen el mismo peso (w_a=1/d)
2. Construye un clasificador h_0 (con, por ejemplo, C4.5). Su error es e_0
3. Repite mientras $0<e_i<0.5$
    1. Observa en que datos falla h_{i-1}
    2. Construye un nuevo conjunto de entrenamiento, dándole mas importancia a los datos fallidos:
        1. Si h_{i-1} clasifica mal el dato (x_a,y_a), aumenta el peso $w_a = w_a^* \, e_{i-1}/(1-e_{i-1})$
        2. Si h_{i-1} clasifica bien el dato (x_a,y_a), decrementa el peso $w_a = w_a^* \, (1-e_{i-1})/e_{i-1}$
    3. Construye un nuevo clasificador h_i con los nuevos datos. Su error es e_i (calculado sobre la muestra con pesos)

El clasificador final *f* es una combinación de todos los h_i. Los coeficientes alfa dependen de lo preciso que sea el clasificador h_i (de su porcentaje de aciertos)

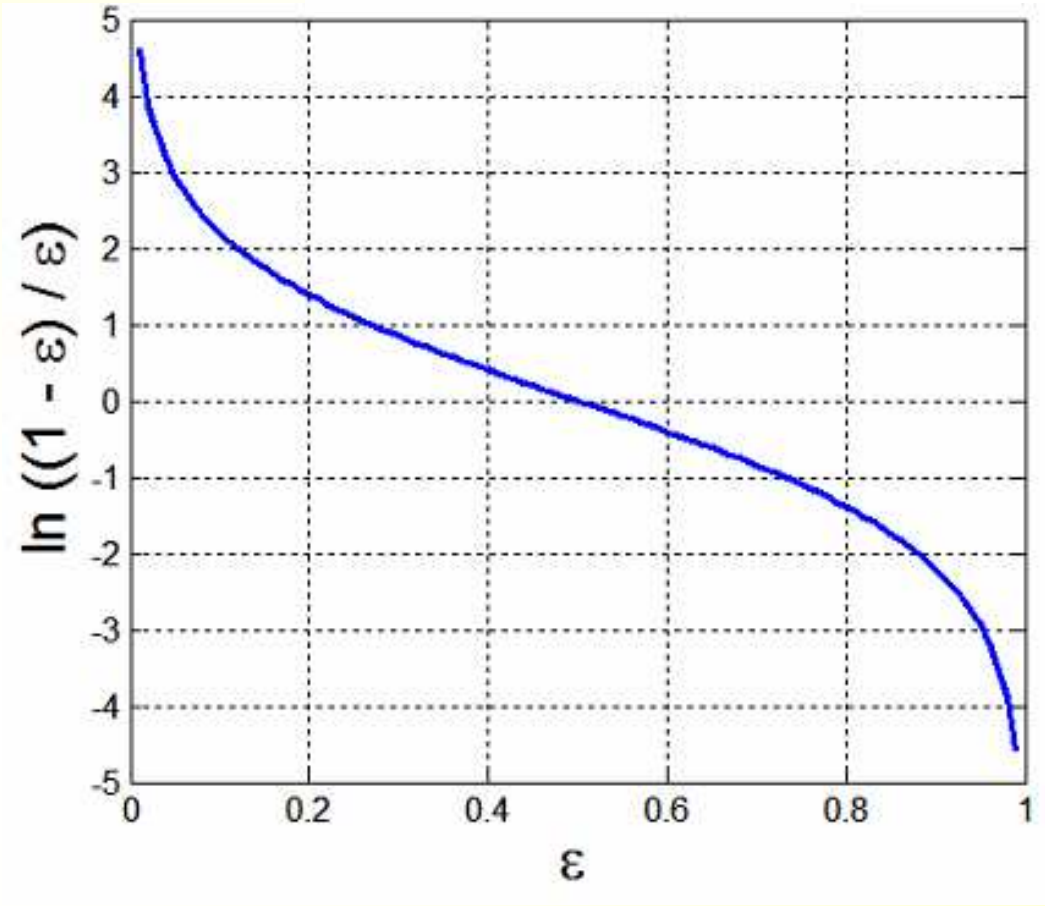
$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

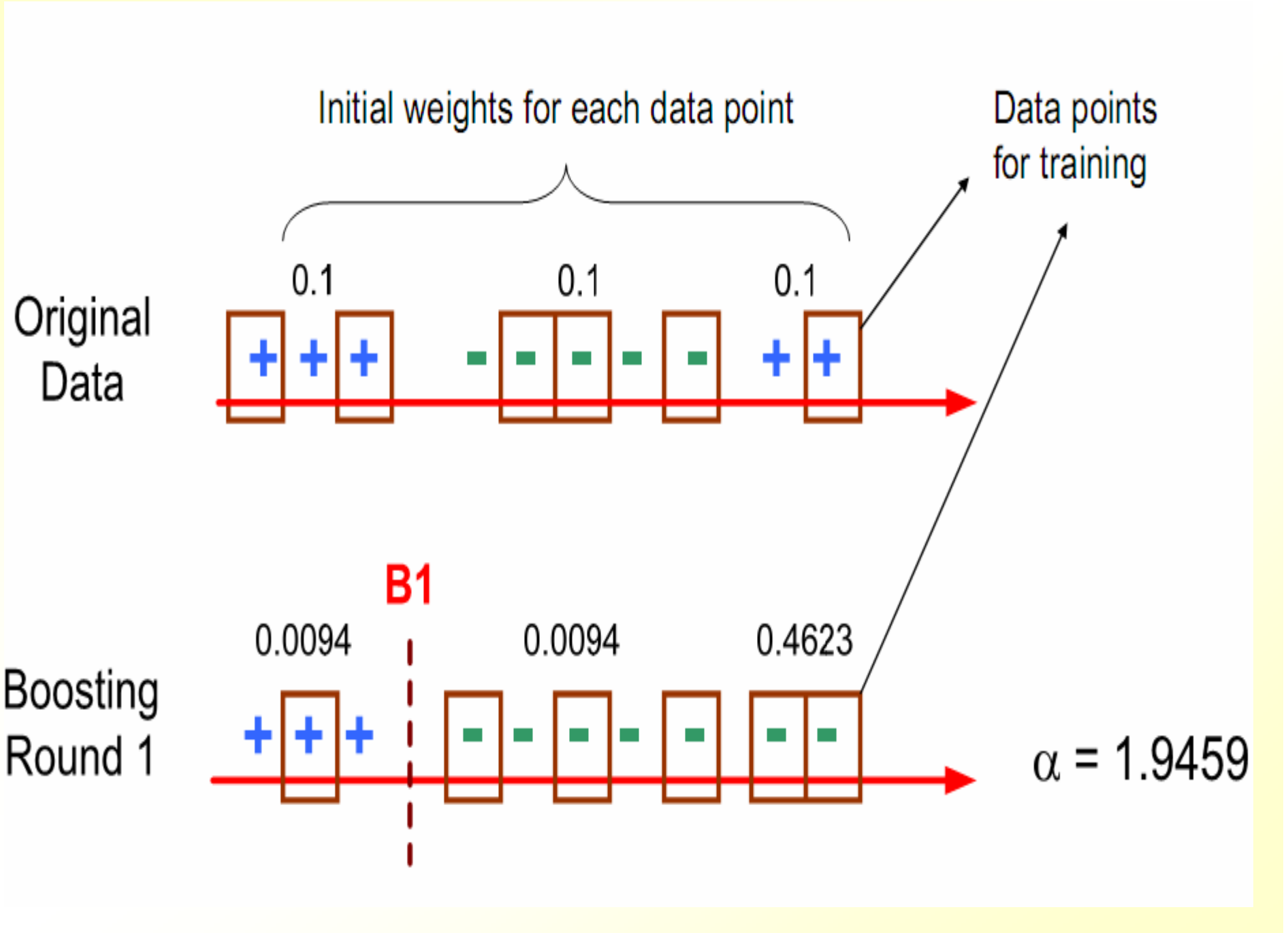
Nota: h_i tiene que devolver un valor +1 o -1 (clase positiva o negativa en problemas biclase), o un valor intermedio.

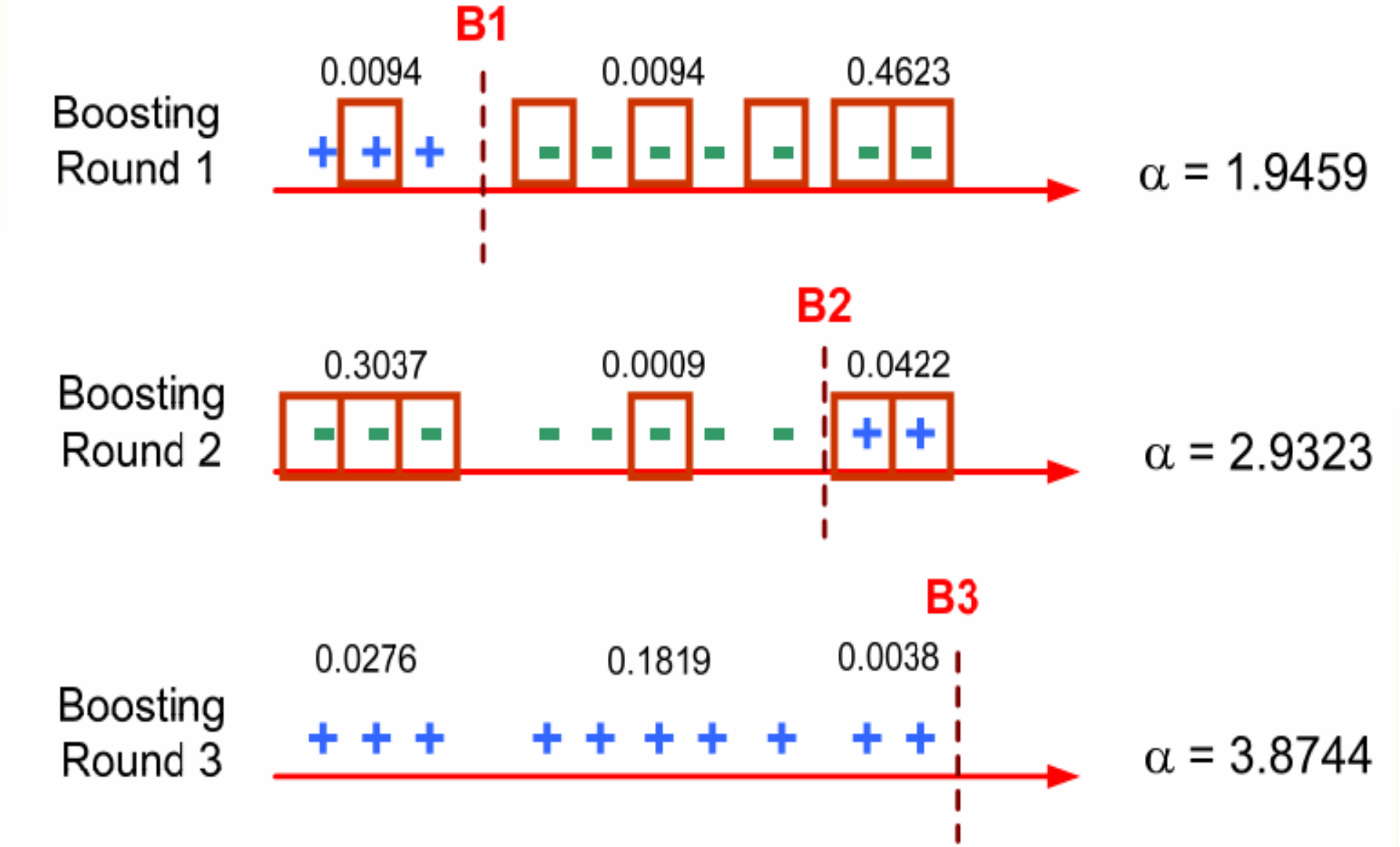
$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x).$$

Importancia de cada clasificador

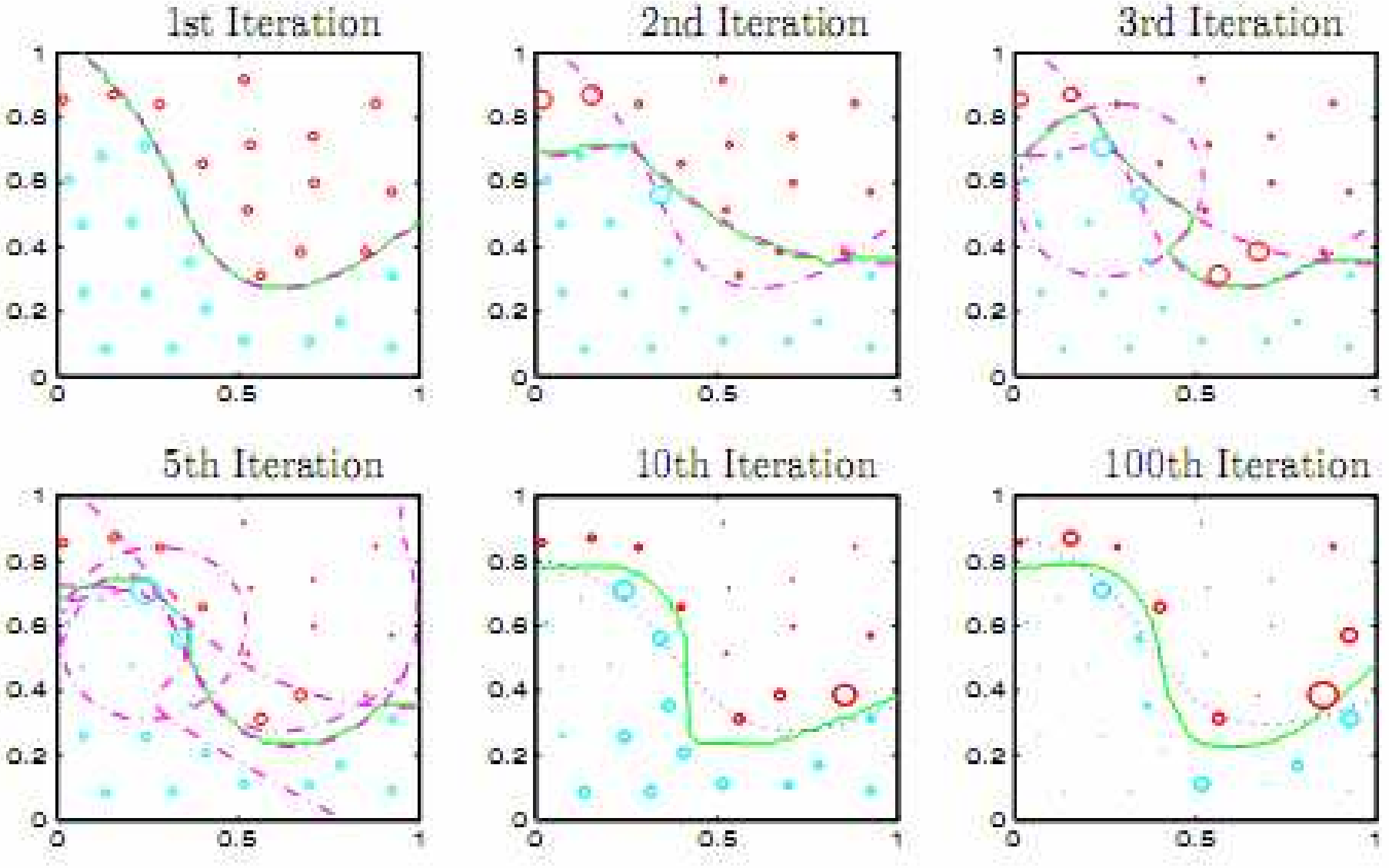
$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$




Initial weights for each data point
Data points for training
Original Data
0.1
0.1
0.1
B1
Boosting Round 1
0.0094
0.0094
0.4623
α = 1.9459


B1
0.0094
0.0094
0.4623
Boosting
Round 1
α = 1.9459
B2
0.3037
0.0009
0.0422
Boosting
Round 2
α = 2.9323
B3
0.0276
0.1819
0.0038
Boosting
Round 3
α = 3.8744

Iteraciones de Adaboost

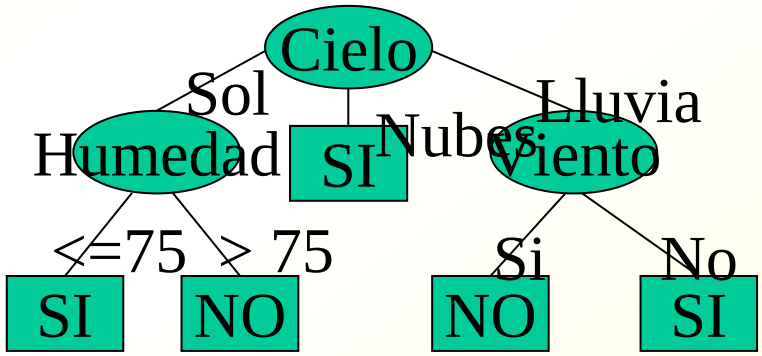


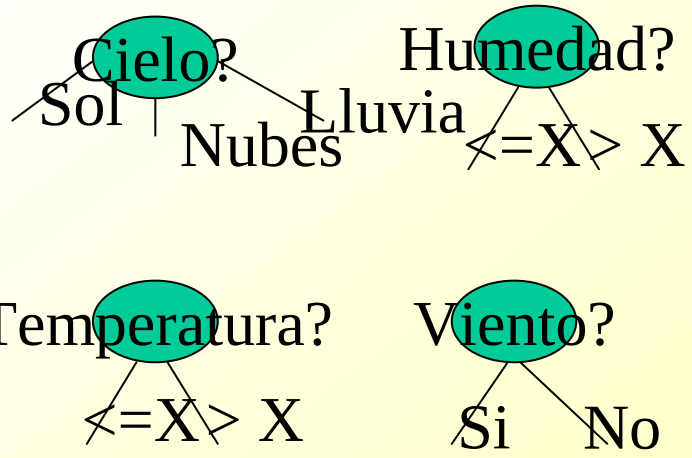
Problemas de Boosting

- Si los datos tienen ruido, Boosting se puede sobreadaptar al ruido

Random Forests

- Es Bagging con árboles de decisión (por ejemplo, creados con J48)
- Pero en cada nodo se pone, no el mejor atributo, sino:
  - El mejor de un subconjunto aleatorio de m atributos
  - Un atributo elegido aleatoriamente de entre los m mejores





Random Forests

- Sólo dos parámetros: número *k* de árboles en el ensemble y número *m* de atributos para ser tenidos en cuenta en cada creación de nodo
- Alta precisión y menos sensible al ruido que Boosting

Nombres de algoritmos

- Árboles de decisión y reglas. Para clasificación
  - Árboles de decisión: ID3, C4.5 (J48), ...
  - Reglas: PART, ...
- Funciones:
  - Para regresión: linear regressíon, neural networks
  - Para clasificación: simple logistics, support vector machines (SMO)
- Árboles de regresión: LMT (M5), ...
- Técnicas perezosas. Para clasificación y regresión
  - IB1, IBK, Locally weighted regression, …
- Técnicas Bayesianas. Para clasificación:
  - Naive Bayes
- Metatécnicas. Para clasificación y regresión:
  - AdaboostM1, Bagging, Stacking, StackingC, Random Forests

EVALUACIÓN DEL CONOCIMIENTO MINADO

Evaluación: entrenamiento y test

- Una vez obtenido el conocimiento es necesario validarlo para observar su comportamiento con datos no vistos
- Ejemplo: si a un alumno se le evalúa (examen) con los mismos problemas con los que aprendió, no se demuestra su capacidad de generalización
- Solución: dividir el conjunto de datos en un subconjunto para entrenamiento (66%) y otro para test (33%)
- Problema: es posible que por azar, los datos de entrenamiento y test estén sesgados
  - Ejemplo de sesgo: Sea un problema para determinar qué tipo de personas compran aparatos de DVD. Puede ocurrir por casualidad que en los datos de entrenamiento aparezcan muchas mas mujeres que hombres. El sistema creerá que hay una correlación entre el sexo y la clase.

Evaluación: entrenamiento y test múltiples veces (repetido)

- Consiste en partir el conjunto de datos totales múltiples veces y calcular el porcentaje de aciertos medio
- La idea es que los sesgos de unas y otras particiones se cancelen
- **Método:**
  - **Repetir múltiples veces:**
    1. **Desordenar el conjunto de datos total aleatoriamente**
    2. **Escoger los primeros 70% para entrenamiento y construir el modelo con ellos**
    3. **Escoger los últimos 30% para el test y estimar el porcentaje de aciertos**
  - **Calcular el porcentaje de aciertos medio**

Particiones estratificadas

- Es conveniente que las particiones sean **estratificadas**
- La proporción entre las clases que existe en el conjunto de datos original, se intenta mantener en los conjuntos de entrenamiento y test
- Ejemplo: si en el conjunto original un 65% de los datos pertenecen a la clase positiva, la estratificación intentará que esa proporción se mantenga en entrenamiento y test

entrenamiento y test múltiples veces (repetido)

- Problema: las distintas particiones de test no son independientes (pueden solaparse unas con otras por casualidad)
- Explicación: en el caso extremo, si por casualidad todas las particiones de test contuvieran exactamente los mismos datos, el repetir muchas veces el cálculo en test no nos aportaría ninguna información adicional
- El caso extremo no ocurre, pero siempre hay algún solape entre las particiones de test
- Lo ideal es que las particiones de test no solapen

Validación cruzada (crossvalidation)

- Solución: dividir **varias veces** el mismo conjunto de datos en entrenamiento y test y calcular la media. Así, las particiones de test no solaparán.
- Se divide el conjunto de datos original en k partes. Con k=3 tenemos los subconjuntos A, B, y C.
- Tres iteraciones:
  - Aprender con A, B y test con C (T1 = % aciertos con C)
  - Aprender con A, C y test con B (T2 = % aciertos con B)
  - Aprender con B, C y test con A (T3 = % aciertos con A)
  - % aciertos esperado T = (T1+T2+T3)/3
- El clasificador final CF se construye **con todos los datos (los tres conjuntos** A, B y C). Se supone que T es una estimación del porcentaje de aciertos de CF
- Se suele utilizar k=10

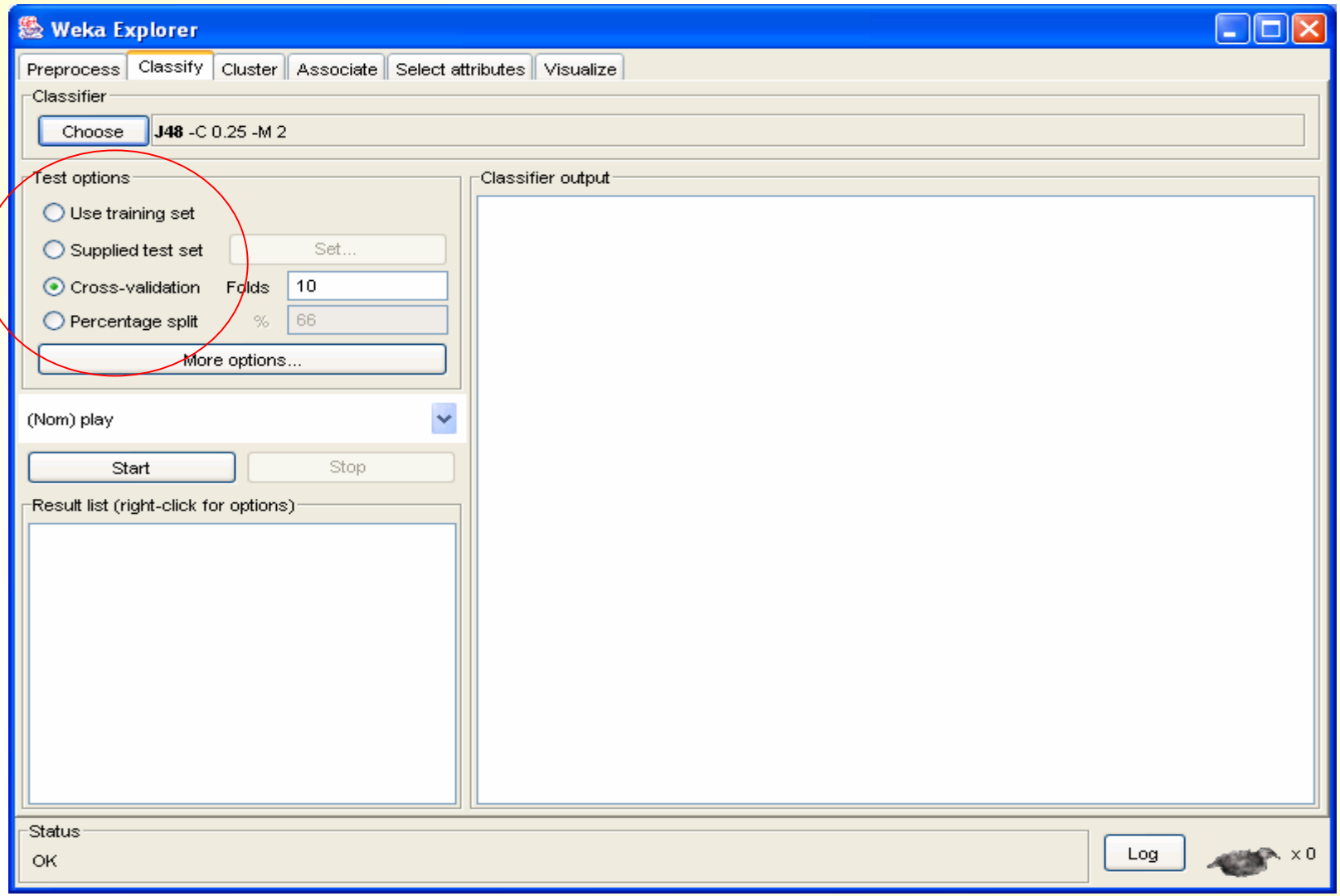
Validación cruzada (crossvalidation)

- El método de validación cruzada utiliza muy bien los datos al calcular el porcentaje de aciertos esperado, porque todos ellos se utilizan para test (en alguna partición).
- De hecho, todos los datos figuran como entrenamiento o test en alguno de los ciclos de validación cruzada.
- Las particiones de test de los distintos ciclos son independientes (no solapan)
- Nota: a cada una de las k divisiones de los datos de entrenamiento se la denomina *fold*

Leave-one-out

- Es una validación cruzada con k = número de datos de entrenamiento
- Si hay n datos de entrenamiento, repetir k=n veces:
  - Reservar el dato número n para test
  - Entrenar con los n-1 datos restantes
  - Hacer el test con el dato n (el resultado sólo puede ser acierto o fallo)
- El porcentaje de aciertos esperado será:
  - (aciertos/n)*100
- Es preciso porque se usan casi todos los datos para entrenar, y a la vez todos los datos figuran como test en alguno de los ciclos
- Pero es costoso en tiempo (hay que lanzar el algoritmo de aprendizaje n veces)

Métodos de evaluación: conjunto de entrenamiento, conjunto de test, validación cruzada, partición del conjunto de entrenamiento



Criterios básicos para evaluar

- En problemas de clasificación, si tenemos 2 clases (o M), el porcentaje de aciertos a superar es el 50% (o 100*1/M).
  - De otra manera, sería mejor tirar una moneda (azar) que utilizar el clasificador para predecir

- En problemas de clasificación, si tenemos una clase con muchos más datos que otra, el porcentaje de aciertos a superar es el porcentaje de datos de la clase mayoritaria
  - Ej: Sean dos clases (+ y -). Hay 90 datos + y 10 -. Un clasificador que prediga siempre + (independientemente de los atributos), ya acertará en un 90%. Hay que hacerlo mejor que eso.

Criterios básicos para evaluar. Coste

- En ocasiones el coste de fallar en una clase no es el mismo que fallar en otra
- Por ejemplo, para un clasificador de cáncer si/no, es preferible predecir que una persona tiene cáncer (sin tenerlo) que predecir que no lo tiene (teniéndolo)
- Ambos casos disminuyen el porcentaje de aciertos, pero lo primero tiene menos coste que lo segundo
- Para analizar esos casos es conveniente utilizar la matriz de confusión

Evaluación. La matriz de confusión y el coste

- Sea un problema con dos clases + y – (positivo y negativo)
- Los datos correctamente clasificados están en la diagonal, los incorrectos fuera de ella
  - El porcentaje de aciertos es
    (TP+TN)/(TP+TN+FN+FP)
  - El porcentaje de aciertos de + es:
    TP rate = TP / positivos = TP/(TP+FN)
  - El porcentaje de aciertos – es:
    TN rate = TN / negativos = TN/(FP+TN)

	Clasificado como +	Clasificado como -
Dato realmente +	**TP (true positive)**	FN (false negative)
Dato realmente -	FP (false positive)	**TN (true negative)**

De entre todos los datos positivos, cuantos clasificamos correctamente. Mide lo bien que acertamos en la clase +

Evaluación. La matriz de confusión y el coste

- Supongamos que en el problema de predecir cáncer si/no tenemos dos matrices de confusión. ¿Cuál es la mejor situación?
- Nótese que el % de aciertos es (90+60)/200 = 75% en los dos casos

	Clasificado como +	Clasificado como -
Dato realmente +	**TP 90**	FN 10
Dato realmente -	FP 40	**TN 60**

	Clasificado como +	Clasificado como -
Dato realmente +	**TP 60**	FN 40
Dato realmente -	FP 10	**TN 90**

Notese también que en los datos hay 100 personas con cáncer y 100 personas sin cáncer (sumar las líneas horizontales)

Evaluación. La matriz de confusión y el coste

- En este caso es mejor disminuir el número de falsos negativos (pacientes que tienen cáncer, pero que el clasificador no lo detecta). O lo que es lo mismo, maximizar los TP.
- Es mejor el clasificador que nos de la matriz de la izquierda

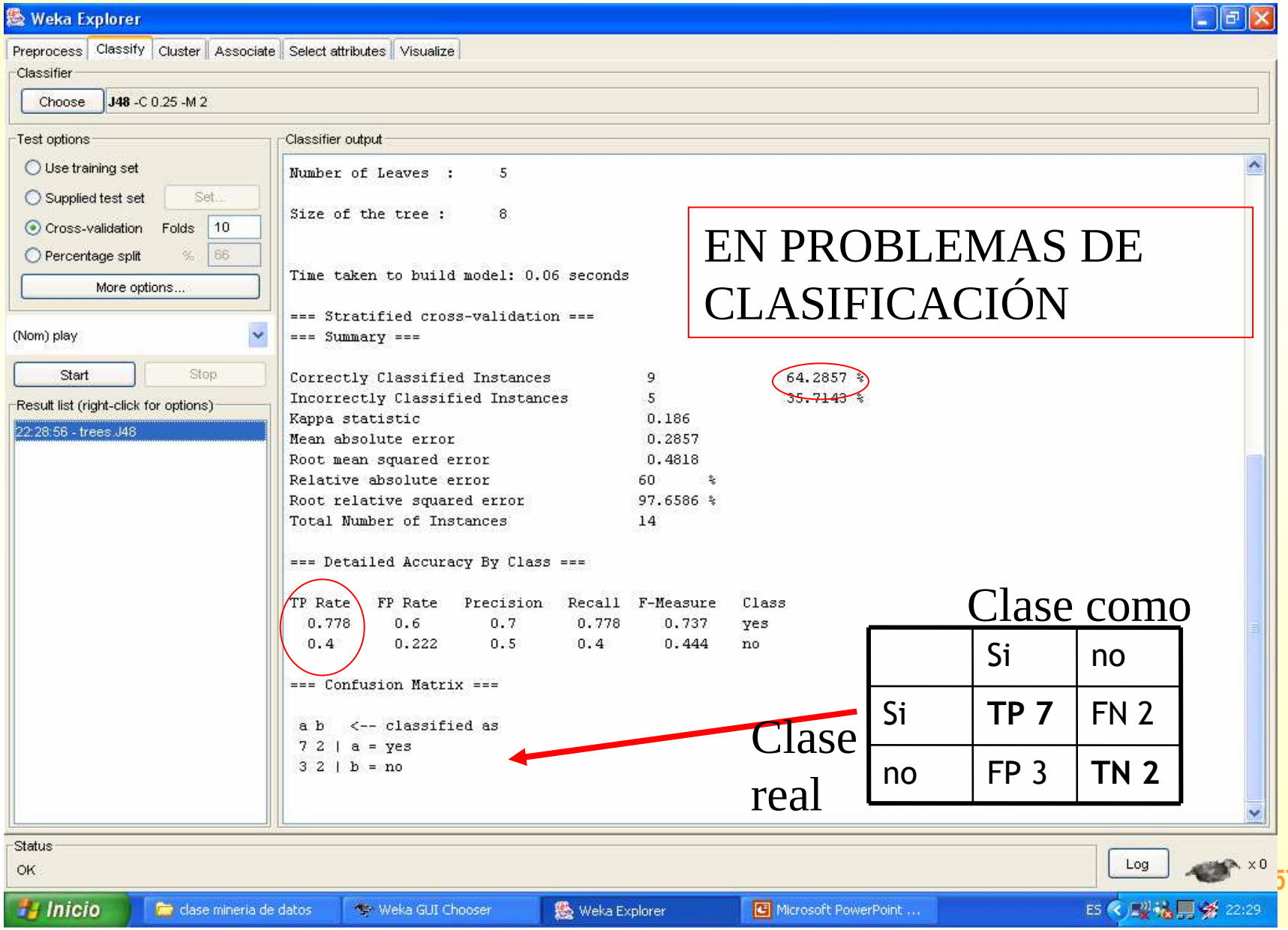
	Clasificado como +	Clasificado como -
Dato realmente +	**TP 90**	FN 10
Dato realmente -	FP 40	**TN 60**

	Clasificado como +	Clasificado como -
Dato realmente +	**TP 60**	FN 40
Dato realmente -	FP 10	**TN 90**

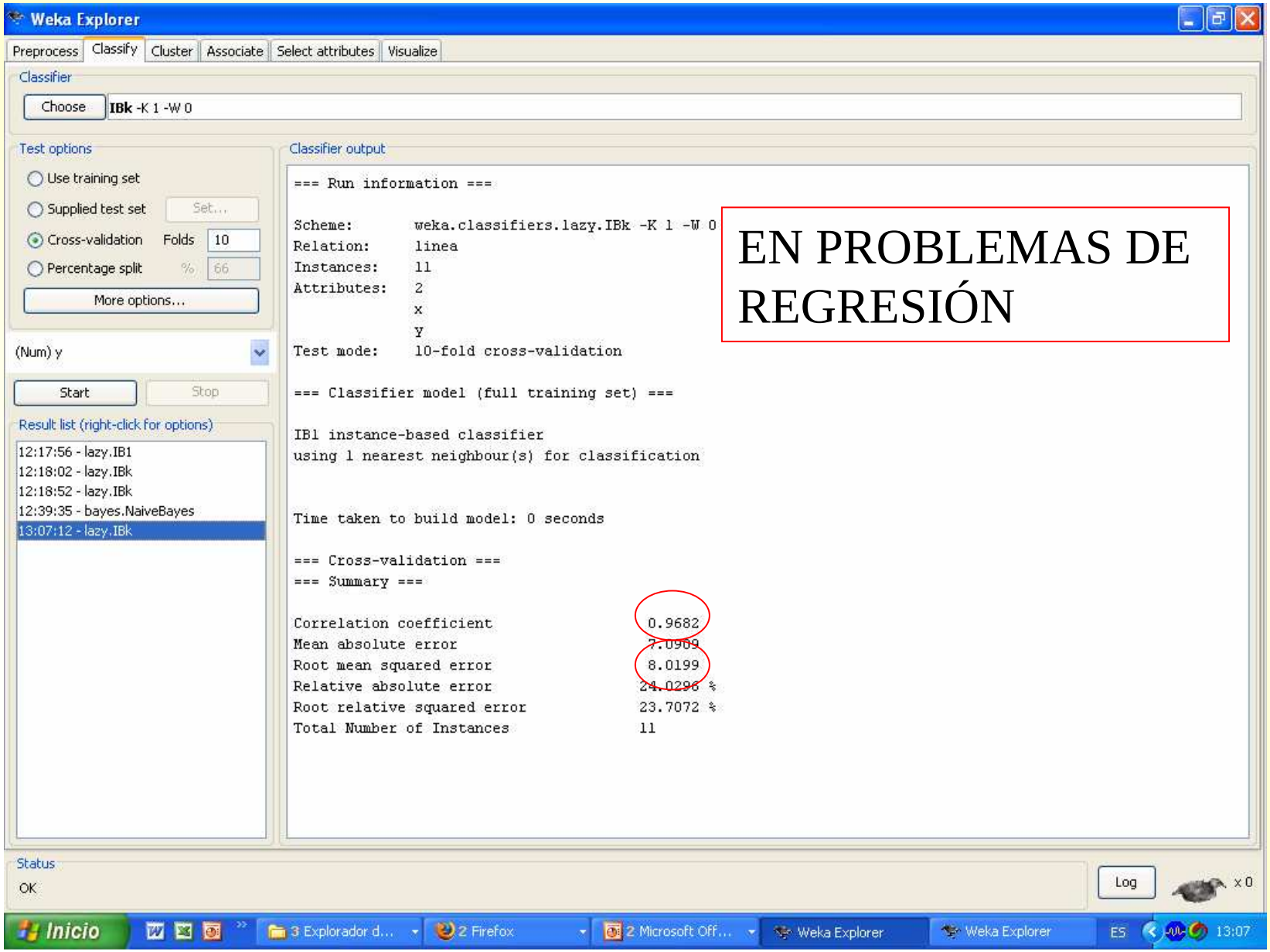
Evaluación. La matriz de confusión y el coste

- Si vemos que el porcentaje de aciertos de la clase positiva es bajo, podemos intentar incrementarlo duplicando las instancias positivas en el conjunto de entrenamiento
- O utilizando directamente matrices de coste (metacost en Weka)

Visualización de resultados: % de aciertos, % de aciertos por clase (“true positive”), matriz de confusión



Visualización de resultados en Regresión



Performance measure	Formula
mean-squared error	$\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{n}$
root mean-squared error	$\sqrt{\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{n}}$
mean absolute error	$\frac{\lvert p_1-a_1\rvert+\ldots+\lvert p_n-a_n\rvert}{n}$
relative squared error	$\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{(a_1-\bar{a})^2+\ldots+(a_n-\bar{a})^2}$, where $\bar{a}=\frac{1}{n}\sum_i a_i$
root relative squared error	$\sqrt{\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{(a_1-\bar{a})^2+\ldots+(a_n-\bar{a})^2}}$
relative absolute error	$\frac{\lvert p_1-a_1\rvert+\ldots+\lvert p_n-a_n\rvert}{\lvert a_1-\bar{a}\rvert+\ldots+\lvert a_n-\bar{a}\rvert}$
correlation coefficient	$\frac{S_{PA}}{\sqrt{S_P S_A}}$, where $S_{PA}=\frac{\sum_i (p_i-\bar{p})(a_i-\bar{a})}{n-1}$, $S_p=\frac{\sum_i (p_i-\bar{p})^2}{n-1}$, and $S_A=\frac{\sum_i (a_i-\bar{a})^2}{n-1}$

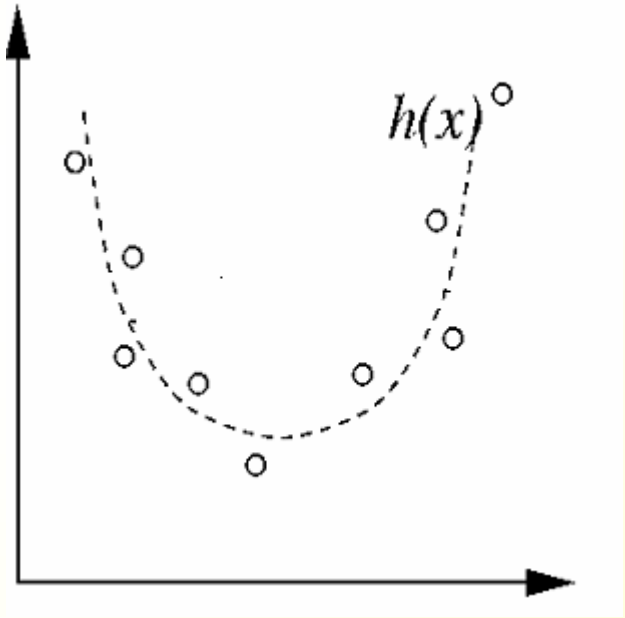
* p are predicted values and a are actual values.

La Sobreadaptación o sobreaprendizaje ("overfitting")

- Se produce sobreadaptación cuando el clasificador obtiene un alto porcentaje de aciertos en entrenamiento pero pequeño en test (es decir, no generaliza bien)
- Se puede decir que el clasificador está **memorizando** los datos en lugar de **generalizando**
- Ej: Un alumno aprende a realizar perfectamente los problemas de exámenes anteriores, pero no sabe resolver los del examen final
- Podemos detectarlo también porque en validación cruzada saldrán porcentajes cercanos al azar

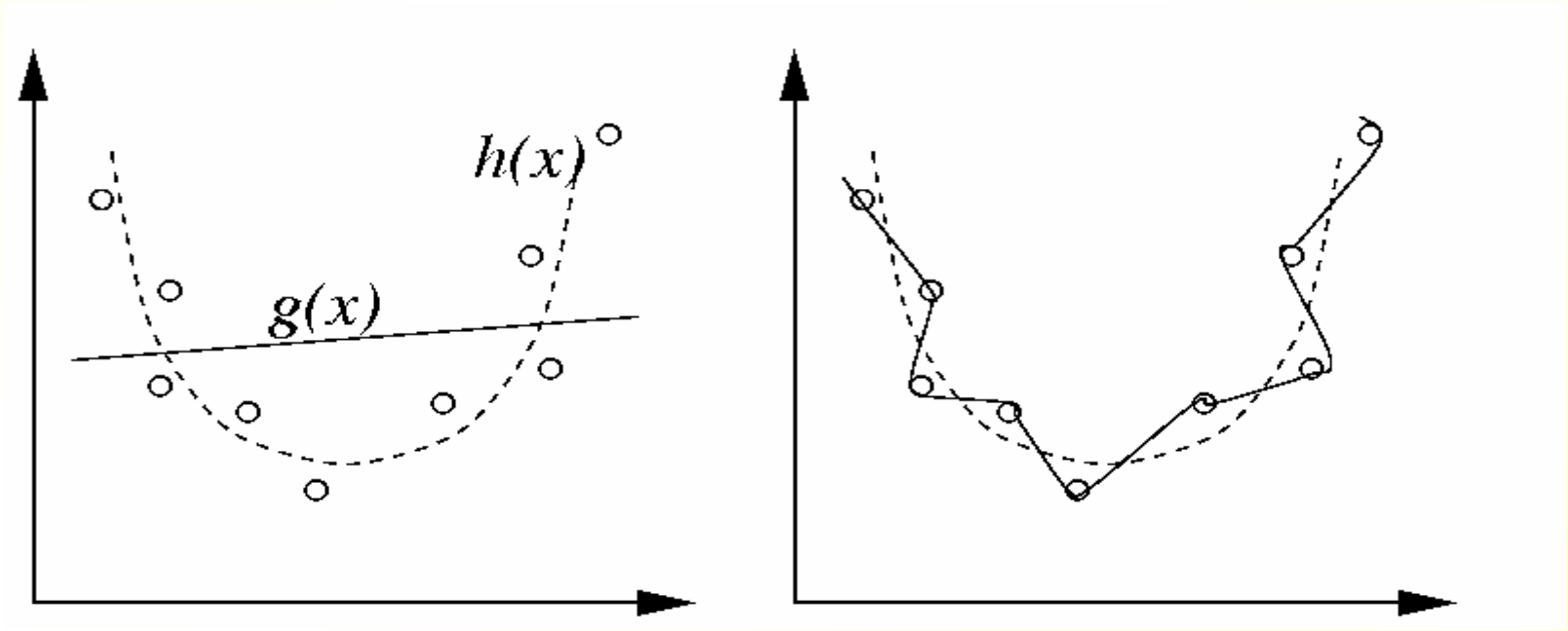
Idea de sobreadaptación

- Supongamos que se trata de un problema de regresión y los datos están distribuidos según una parábola, pero hay algo de **ruido**
- Es decir, el modelo subyacente es una parábola, pero los datos muestran ligeras variaciones (ruido)



Sobreadaptación/subadaptación

- Derecha: el modelo se ha sobreadaptado al ruido porque es demasiado complejo
- Izquierda: el modelo lineal **g(x)** es demasiado simple para aproximar una parábola y subadapta los datos
- Conclusión: tiene que haber un equilibrio en la complejidad del clasificador (o del modelo en general)

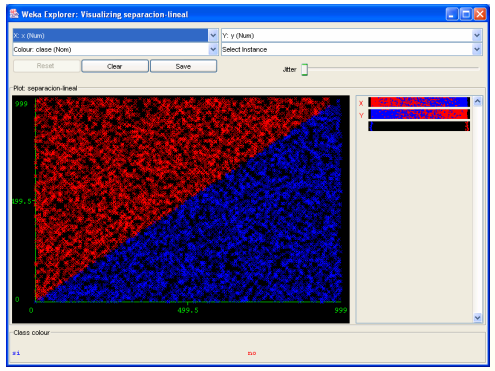


Sobreadaptación/subadaptación

- En general un algoritmo de aprendizaje va a generar clasificadores o regresores de determinada complejidad.
- Tenemos cierto control sobre la complejidad de los predictores mediante parámetros del algoritmo:
  - Redes de neuronas: número de neuronas ocultas
  - Polinomios: grado del polinomio
  - Árboles de decisión: número de nodos
- Tenemos que acertar con la complejidad apropiada: probar distintos valores del parámetro
- Afortunadamente, en muchas ocasiones el valor por omisión del parámetro nos va a proporcionar buenos resultados

Sobreadaptación/subadaptación de un clasificador lineal

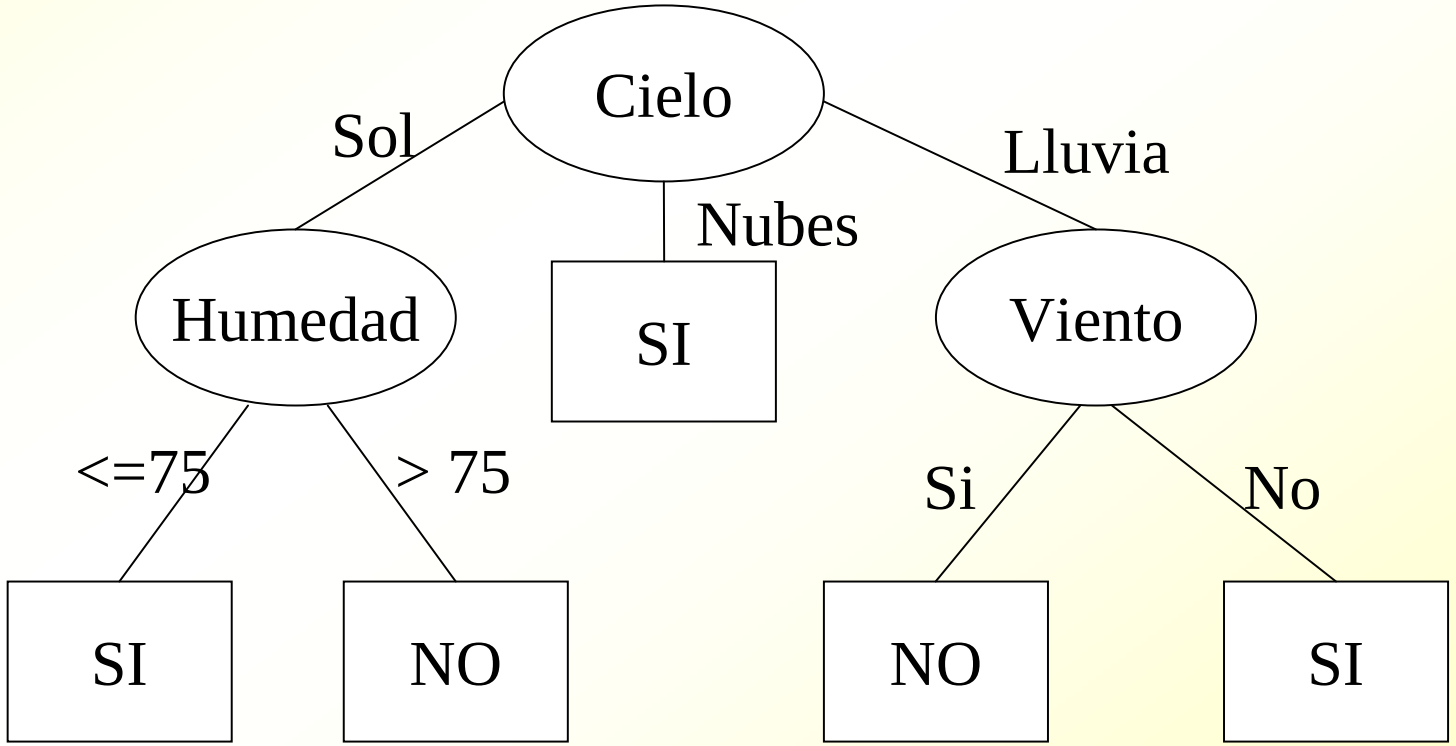
- Sea un problema de clasificación biclase con **1000 atributos**
- Disponemos de un algoritmo que genera clasificadores lineales (como el logistic regresion)
- Supongamos que tenemos **1000 datos de entrenamiento** (y por ejemplo 10000 para test)
- ?Cuál será el porcentaje de aciertos en entrenamiento?
- ?Cuál será el porcentaje de aciertos en test?



$$Y = A_1*X_1 + A_2*X_2 + A_3*X_3 + \ldots + A_{1000}*X_{1000} + A_0$$

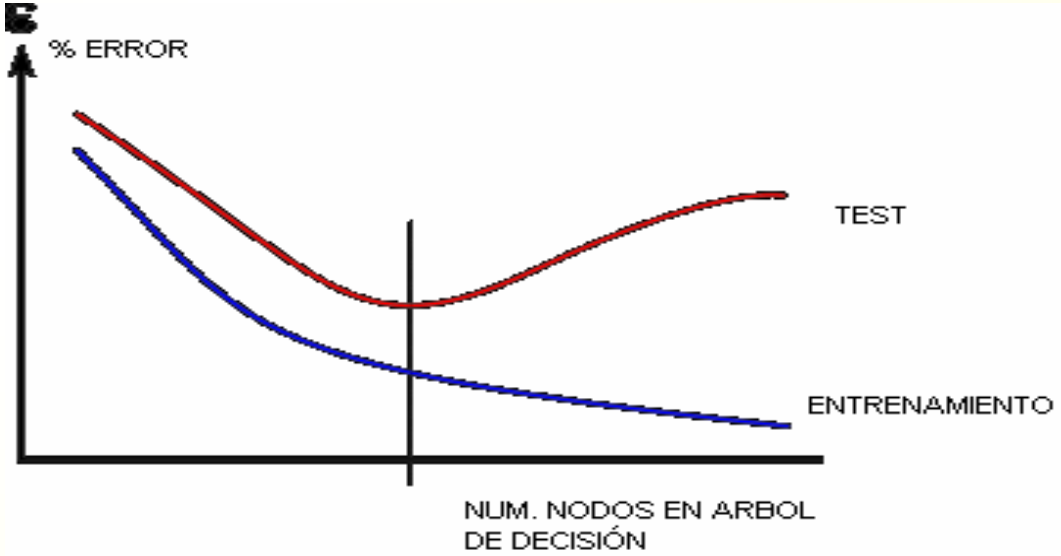
Sobreadaptación por excesiva complejidad del clasificador

Los árboles de decisión se van construyendo nodo a nodo. Cuantos más nodos, más complejo y más probable que sobreadapte

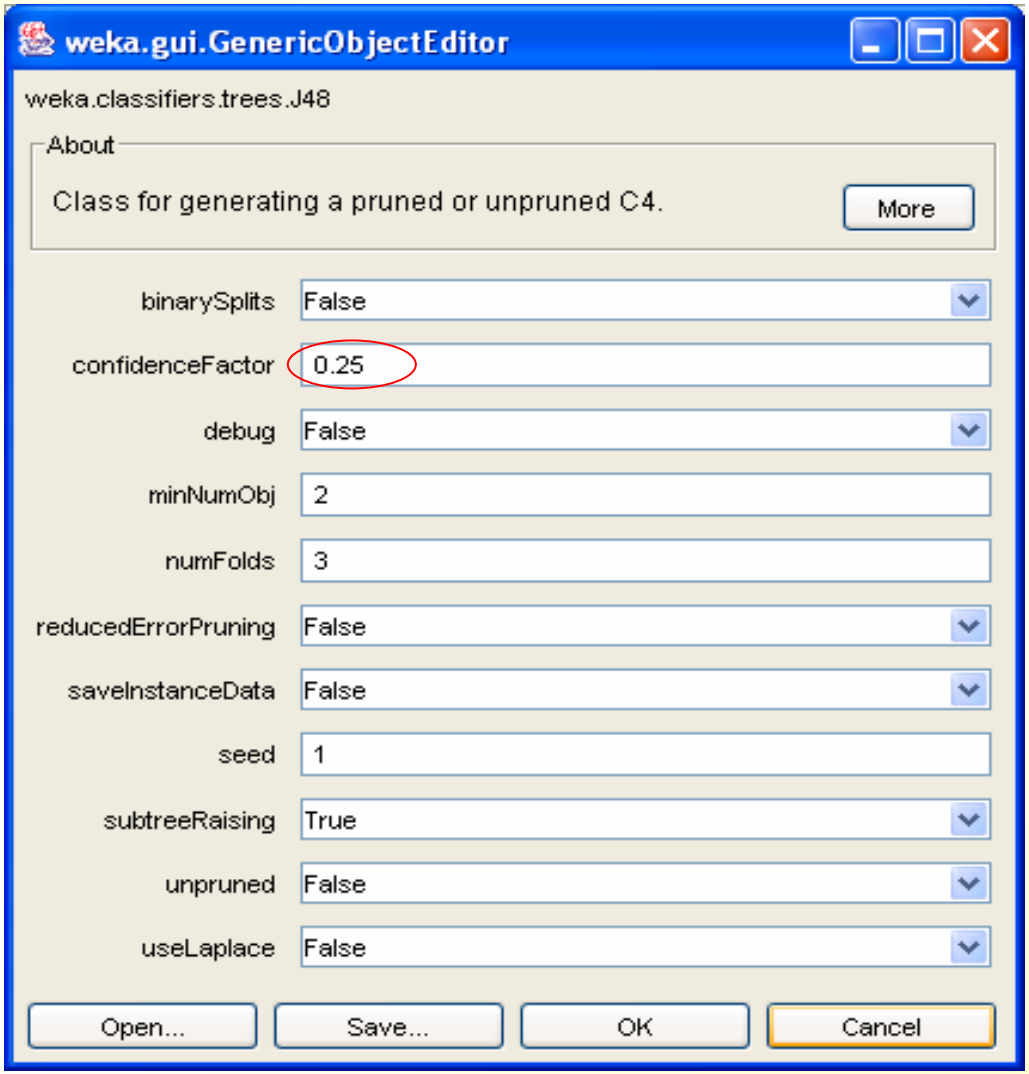


Sobreadaptación por excesiva complejidad del clasificador

- Al principio, incrementar el tamaño del árbol de decisión disminuye el error en entrenamiento y test
- Pasada cierta complejidad del árbol, el error sigue decreciendo en entrenamiento pero crece en test
- Muchos algoritmos tienen parámetros que permiten controlar la complejidad del clasificador (en árboles de decisión, el parámetro de poda detiene el crecimiento)



Parámetro de j48 contra la sobreadaptación



Sobreadaptación. Resumen

- Factores que influyen: ruido, número de datos y complejidad del clasificador
- Ej: si hay pocos datos y permitimos gran complejidad al clasificador (“que crezca mucho”) habrá sobreadaptación (memorización)
- Ej: si hay ruido en los datos y permitimos gran complejidad al clasificador, se sobreadaptará al ruido
- Ej: pero si la complejidad del clasificador es insuficiente, habrá subadaptación

Teorema "No Free Lunch"

A1	A2	A3	OR
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

- Existen 2^8=256 funciones binarias distintas
- Supongamos que le damos a aprender cada una de ellas al J48
- Sea $\%_i$ los aciertos de J48 con la función binaria *i*
- $(\%_1 + \%_2 + \ldots + \%_{256})/256 =$ **??**

Teorema "No Free Lunch" I

A1	A2	A3	OR
0	**0**	**0**	**0**
0	**0**	**1**	**1**
0	**1**	**0**	**1**
0	**1**	**1**	**1**
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

- Supongamos que le planteamos a un algoritmo de minería de datos la función binaria OR
- Los datos en negrita son los de entrenamiento
- Supongamos que las predicciones dadas por el algoritmo de minería de datos acierta un **P%** con el resto de los datos (aparecen subrayados)

Teorema "No Free Lunch" II

A1	A2	A3	OR
0	**0**	**0**	**0**
0	**0**	**1**	**1**
0	**1**	**0**	**1**
0	**1**	**1**	**1**
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

A1	A2	A3	FBIN
0	**0**	**0**	**0**
0	**0**	**1**	**1**
0	**1**	**0**	**1**
0	**1**	**1**	**1**
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

- Si ahora le planteamos al algoritmo otra función binaria (FBIN), el porcentaje de aciertos sería **100-P%**
- Es decir, donde se acertaba en OR, se falla en FBIN
- En media:

(100-P%+P%)/2 = 50%

Teorema "No Free Lunch" III

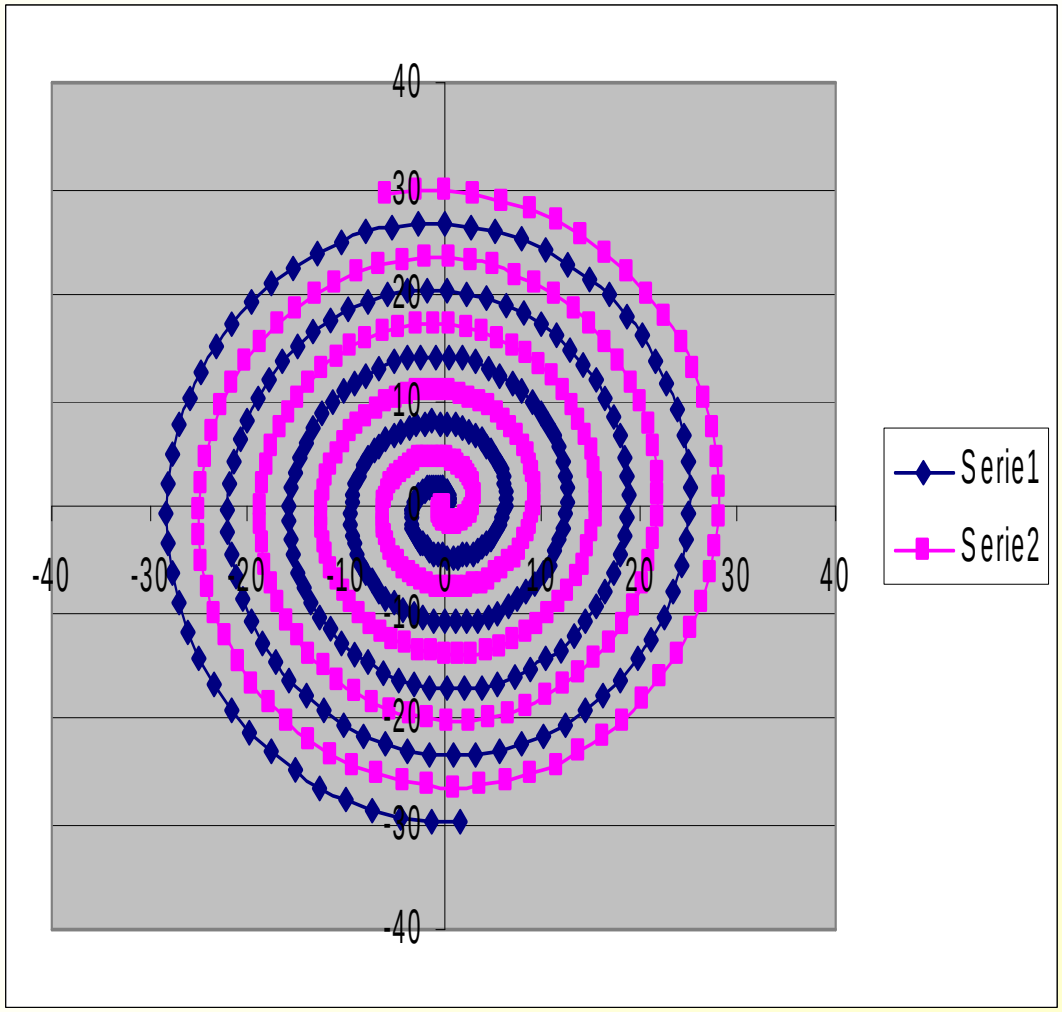
- Cuando todas las combinaciones de los datos son posibles, las predicciones de un algoritmo de minería de datos concreto no son mejores que el azar (en media)
- No hay ningún algoritmo de minería de datos que sea mejor que todos los demás en todos los posibles problemas

Teorema "No Free Lunch" IV

- El aprendizaje de regularidades es posible porque en el mundo real, no todas las posibles combinaciones de los datos son posibles
- La minería de datos funciona porque en el mundo real es posible suponer determinadas cosas acerca de los datos. Por ejemplo, que su comportamiento es simple, o suave en las funciones continuas
- Cada algoritmo hace suposiciones distintas (*bias*) acerca de los datos. El algoritmo funcionará mejor o peor según acierte o falle en esas suposiciones

Espiral

Imposible aprender a separar dos espirales con reglas o árboles de decisión, aunque ib1 lo hace relativamente bien



Paridad par

A1	A2	A3	¿PAR?
0	0	0	SI
0	0	1	NO
0	1	0	NO
0	1	1	SI
1	0	0	NO
1	0	1	SI
1	1	0	SI
1	1	1	NO

- Id3: 1%
- j48: 40%
- PART: 15%
- IB1: 0%
- IBK: 0%
- NBAYES: 36%
- SIMPLE LOG.: 50%

- Ningún sistema es capaz de aprender la regularidad “número par de 1s” porque su lenguaje de representación no lo permite

Otras medidas de evaluación: comprensibilidad

- En ocasiones es importante evaluar el conocimiento obtenido con otras medidas
- Comprensibilidad: si el conocimiento es fácilmente comprensible para un ser humano. Útil para evaluar si el conocimiento es correcto o para tomar decisiones en base al conocimiento obtenido
- Muy relacionado con el tamaño (número de reglas o nodos en el árbol de decisión)
- A veces merece la pena perder en porcentaje de aciertos (= subadaptación) para ganar en comprensibilidad (construyendo árboles de decisión más pequeños, discretizando atributos, etc.)

Comparación de varios algoritmos de generación de clasificadores

- Tenemos un conjunto de datos
- Sabemos que algunos algoritmos de generación de clasificadores funcionará mejor sobre esos datos que otros
- ?Cuál usar, ej: J48 o SVM?
- Podemos hacer validación cruzada y quedarnos con el mas alto
- En la práctica esto es suficiente
- Pero en ocasiones la diferencia puede ser debida al azar y no ser significativa estadísticamente
- Ejemplo: buscamos un algoritmo preciso pero también rápido (lo vamos a utilizar en una situación dinámica que exige un entrenamiento rápido). Tenemos un algoritmo A (J48) que obtiene un 90% de aciertos, y otro B (NN) que obtiene un 92%, pero B es 100 veces mas lento que A. ?Merece la pena usar B?

Comparación de varios algoritmos de generación de clasificadores

- Ejemplo, sobre un conjunto de datos E, J48 puede obtener un 90% de aciertos (en 10-fold crossvalidation) y NN 92%. ?Podemos asegurar que NN es mejor que J48 en este dominio?
- No necesariamente, si usaramos otro conjunto de datos E', puede que J48 sacara 92% Y NN 89%
- Existe variabilidad, debido a que no disponemos del conjunto total de datos (que puede ser infinito), sino muestras finitas y pequeñas E,E', …
- Necesitamos saber como de grande es esa variabilidad (varianza)

Comparación de varios algoritmos (A y B) de generación de clasificadores

- Necesitamos saber como de grande es esa variabilidad (varianza)
- Hacemos la validación cruzada muchas veces

- Para cada algoritmo, repetir 10 veces
  - Desordenar los datos de entrenamiento
  - Calcular P_i de validación cruzada (de por ejemplo, 10 folds)
- Realizar test estadístico (t-test) para ver si las diferencias son significativas. Si la varianza es pequeña es más fácil que la diferencia sea significativa

Comparación de varios algoritmos (A y B) de generación de clasificadores

- Idea importante: el que la diferencia sea significativa depende mas de que la varianza sea pequeña que de que las medias estén muy separadas.
- ?Cuál de estos dos casos es mas probable que corresponda a una diferencia significativa?. Hacemos para A y B 10 crossvalidations de 10 folds cada una. (media, desviación)
  - A = (90%, 8%), B=(94%, 7%)
  - A = (90%, 0.001%), B=(91%, 0.002%)

Comparación de varios algoritmos (A y B) de generación de clasificadores

- ¿Cómo lo hace Weka?
- Data Mining, practical machine learning tools and techniques. Second Edition. Witten and Frank. Página 153.

Comparación de varios algoritmos de generación de clasificadores

SELECCIÓN DE ATRIBUTOS

Selección de atributos

- Algunos atributos pueden ser **redundantes** (como “salario” y “categoría social”) y hacen más lento el proceso de aprendizaje
- Otros son **irrelevantes** (como el DNI para predecir si una persona va a devolver un crédito)
- En ocasiones el exceso de atributos puede llevar a sobreaprendizaje, pues incrementa la complejidad del modelo (sobre todo si hay pocos datos)
- En ocasiones es útil tener el conocimiento de qué atributos son relevantes para una tarea
- Existen algoritmos de selección de atributos

Idea importante a tener en cuenta

- En ocasiones, dos atributos por separado no dan información, pero juntos sí
- Ejemplo:
  - Sea un problema de clasificación de textos en dos clases "informática" y "filosofía"
  - Sean los atributos booleanos "inteligencia" y "artificial", que son ciertos si esas palabras aparecen en el texto y falsos en caso contrario
  - Por separado no permiten distinguir entre informática y filosofía:

    IF inteligencia=si **THEN** ?; **IF** artificial=si **THEN** ?
  - Pero juntos sí:

    IF inteligencia=si **Y** artificial=si **THEN** "informática"
- Por tanto, el objetivo último de la selección de atributos es encontrar el **subconjunto** mínimo de atributos que hace óptima la predicción

Métodos de selección de atributos

- El método mas preciso es la búsqueda exhaustiva
- Supongamos que tenemos 4 atributos A, B, C, D
- Sería necesario comprobar la validez de todos los posibles subconjuntos (2^4=16): {A, B, C, D}, {A, B, C}, {A, B, D}, {B, C, D}, {A, C, D}, {A, B}, {A, C}, ..., {A}, {B}, {C}, {D}
- En general, el método es poco práctico: 2^n posibles subconjuntos

Métodos de selección de atributos

- Evaluación individual de atributos (**Ranker** u ordenación)
- Evaluación de subconjuntos de atributos:
  - Cfs**SubsetEval**: se evalúan los atributos de manera separada, pero hay cierta evaluación conjunta
  - **SubsetEval**: se evalúan realmente de manera conjunta
- La anterior es la nomenclatura usada en Weka, pero tradicionalmente a los métodos se los divide en dos tipos:
  - **Filter**: los atributos se evalúan de manera independientemente. Esto incluiría a los métodos Ranker y (hasta cierto punto) Cfs**SubsetEval** de Weka
  - **Wrapper**: los atributos se evalúan de manera conjunta. Esto incluiría a todos los métodos subsetEval de Weka y principalmente a ClassifierSubsetEval y Wrapper

Selección de atributos. Ranker

- Dado unos atributos A_1, A_2, ..., A_n
- Se evalúa cada uno de manera independiente, calculando medidas de correlación del atributo con la clase
- Un atributo A_1 está correlacionado con la clase, si conocer su valor implica que podemos predecir la clase con cierta probabilidad
- Por ejemplo, el sexo de una persona está correlacionado (de momento) con que le guste el fútbol. Su DNI no lo está
- Por ejemplo, el salario de una persona está correlacionado con el hecho de que vaya a devolver un crédito

Selección de atributos. Ranker

- Se evalúa cada atributo con algún estadístico que detecte la correlación (ej: chi-cuadrado, infogain, etc.)
- Se ordenan los atributos según ese valor
- Se seleccionan los k mejores
- Método muy rápido
- Problemas:
  - No detecta atributos redundantes
  - En ocasiones no tiene sentido evaluar a los atributos por separado, sino en conjunto.
  - Ej: las aparición de las palabras “inteligencia” y “artificial” no está excesivamente correlacionado por separado con textos de informática, pero juntas su correlación se incrementa notablemente

Problema con atributos redundantes (ej: Naive Bayes)

P(Cielo/Tenis)

Cielo	Si	No
Sol	**2/9**	3/5
Nubes	4/9	0/5
Lluvia	3/9	2/5

P(Temp/Tenis)

Temperatura	Si	No
Caliente	2/9	2/5
Templado	4/9	2/5
Frio	**3/9**	1/5

P(Hum/Tenis)

Humedad	Si	No
Alta	**3/9**	4/5
Normal	6/9	1/5

Viento	Si	No
Si	**3/9**	3/5
No	6/9	2/5

P(Tenis)

Tenis	Si	No
	9/14	5/14

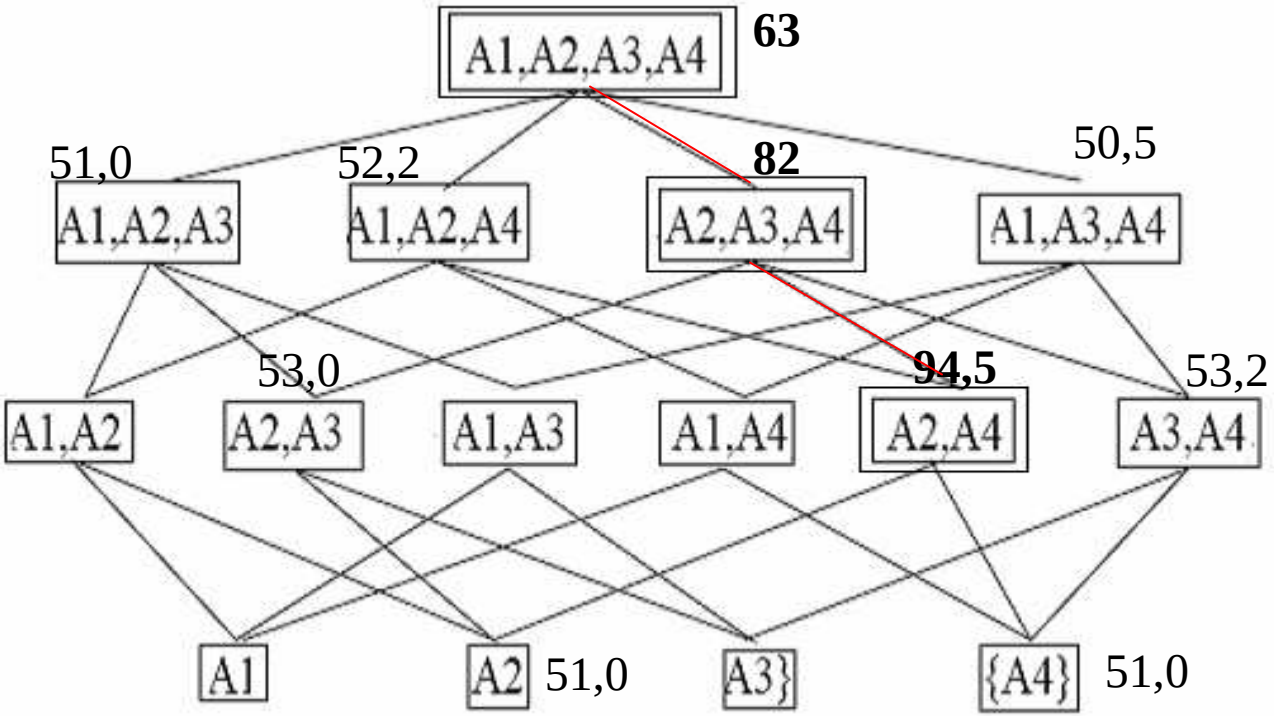
- Pr(si / sol, caliente, alta, si) ~ 2/9 * 2/9 * 3/9 * 3/9 * 9/14
- Pr(no / sol, caliente, alta, si) ~ 3/5 * 2/5 * 4/5 * 3/5 * 5/14

Supongamos que Cielo y Temperatura están correlacionados. El mismo atributo contaría dos veces

Selección de atributos. Evaluación de subconjuntos

- Estos métodos recorren un espacio de búsqueda de subconjuntos de atributos, evaluando **subconjuntos** completos de atributos
- No se recorre el espacio entero (eso sería búsqueda exhaustiva), sino sólo aquellos subconjuntos más prometedores
- Se evalúa el subconjunto de manera conjunta

- Tipos:
  - CfsSubsetEval
  - SubsetEval



Selección de atributos. Filter

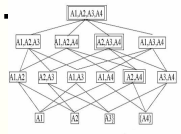
- Los métodos Cfs**SubsetEval** evalúan un subconjunto de atributos calculando:
  - La media de las correlaciones (o similar) de cada atributo con la clase
  - Descontando puntos por redundancias entre atributos
- Método rápido
- Problemas: elimina atributos redundantes, pero como Ranker, puede eliminar atributos que por si solos no están correlacionados con la clase, pero con otro atributo si que lo están (ej: “inteligencia artificial”)

Selección de atributos. Wrapper

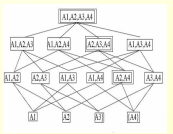
- Los métodos *SubsetEval* evalúan un subconjunto de atributos ejecutando un algoritmo de minería de datos (MD) concreto, sobre un conjunto de entrenamiento
- El valor del subconjunto es el porcentaje de aciertos obtenido con esos atributos
- Son lentos
- Obtienen subconjuntos de atributos adecuados para un algoritmo de MD concreto
- Evalúan a los atributos de los subconjuntos de manera realmente conjunta

Métodos de selección de atributos

- Recordatorio de métodos de selección de atributos:
  - Evaluación **individual** de atributos (Ranker/AttributeEval)
  - Evaluación de **subconjuntos** de atribu (SubsetEval):



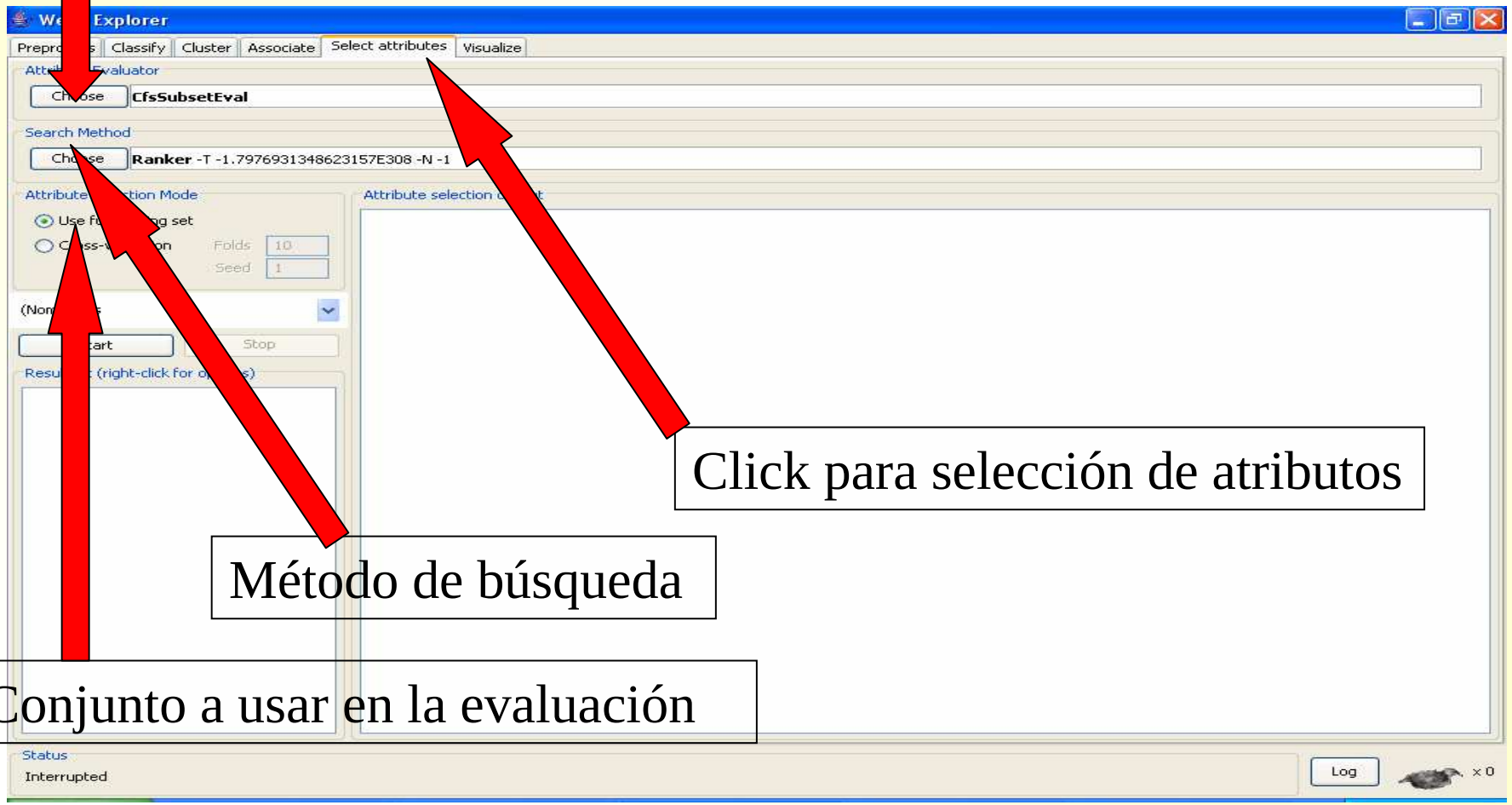
    - CfsSubsetEval: se evalúan los atributos de manera individual, pero tiene en cuenta la redundancia entre atributos
    - SubsetEval: se evalúan de manera conjunta



- Hay que definir:
  - Una manera de moverse por el espacio de búsqueda

Selección de atributos

Método evaluación de subconjuntos de atributos



Click para selección de atributos

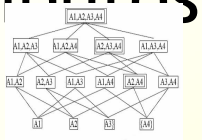
Método de búsqueda

Conjunto a usar en la evaluación

Selección atributos. Tipos

- Evaluación de atributos individuales, Ranker/AttributeEval:
  - Búsqueda: **Ranker**
  - Evaluador: ChiSquare**AttributeEval**, GainRatio**AttributeEval**, InfoGain**AttributeEval**

- Evaluación de subconjuntos de atributos (SubsetEval):



  - Búsqueda: greedy, stepwise, genetic, ...
  - Evaluador:
    - Cfs**SubsetEval**
    - **SubsetEval**:
      - Classifier**SubsetEval**
      - Wrapper**SubsetEval**

Evaluadores de atributos (Ranker)

- ChiSquared**AttributeEval**: usa el estadístico Chi-squared para evaluar el valor predictivo del atributo
- GainRatio**AttributeEval**: usa gainratio
- InfoGain**AttributeEval**: usa infogain

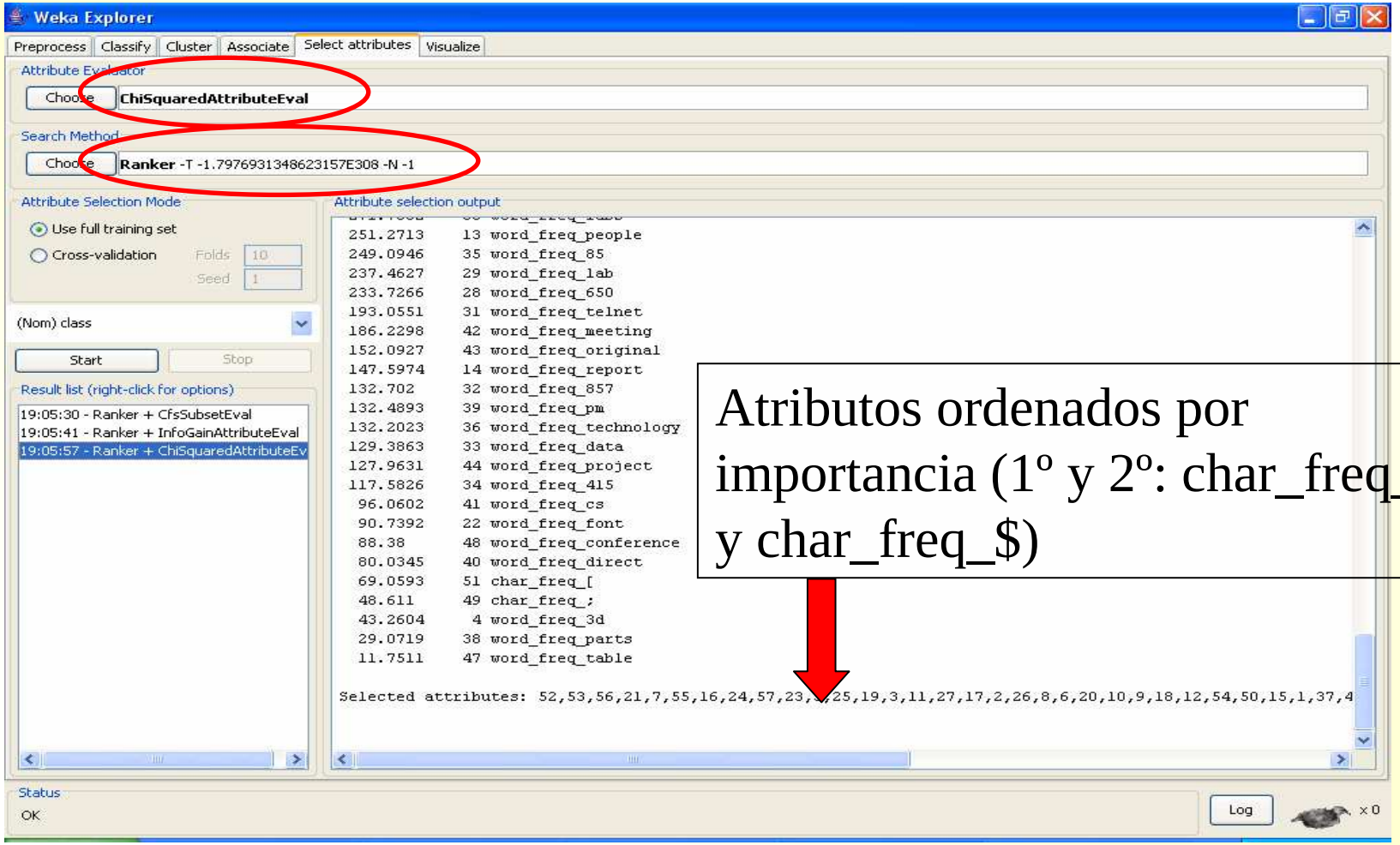
Evaluadores de subconjuntos

- **CfsSubsetEval :** rápidos
  - Cfs**SubsetEval**: considera el valor predictivo (correlación) de cada atributo y de su redundancia
- **SubsetEval**: más lentos
  - Classifier**SubsetEval**: usa un clasificador para evaluar el conjunto
  - Wrapper**SubsetEval**: classificador + validación cruzada

Métodos de búsqueda

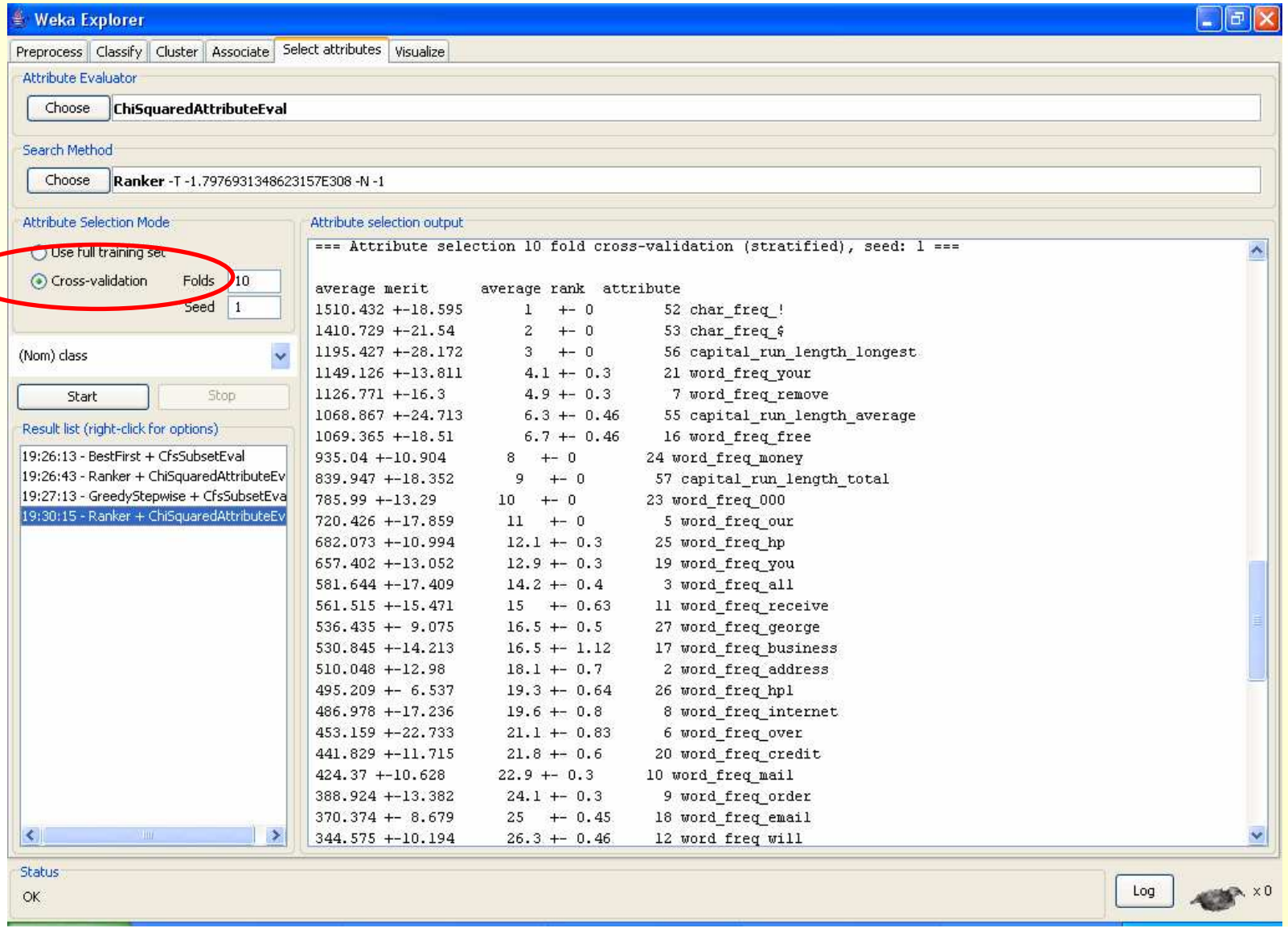
- BestFirst: Mejor primero (lento)
- ExhaustiveSearch: Búsqueda exhaustiva (muy lento)
- GeneticSearch: Búsqueda genética (rápido)
- GreedyStepWise: Escalada (muy rápido)
- RankSearch: Primero ordena los atributos y después construye el subconjunto de manera incremental, en dirección del mejor al peor, hasta que no merece la pena añadir nuevos atributos (rápido)

Selección Ranker

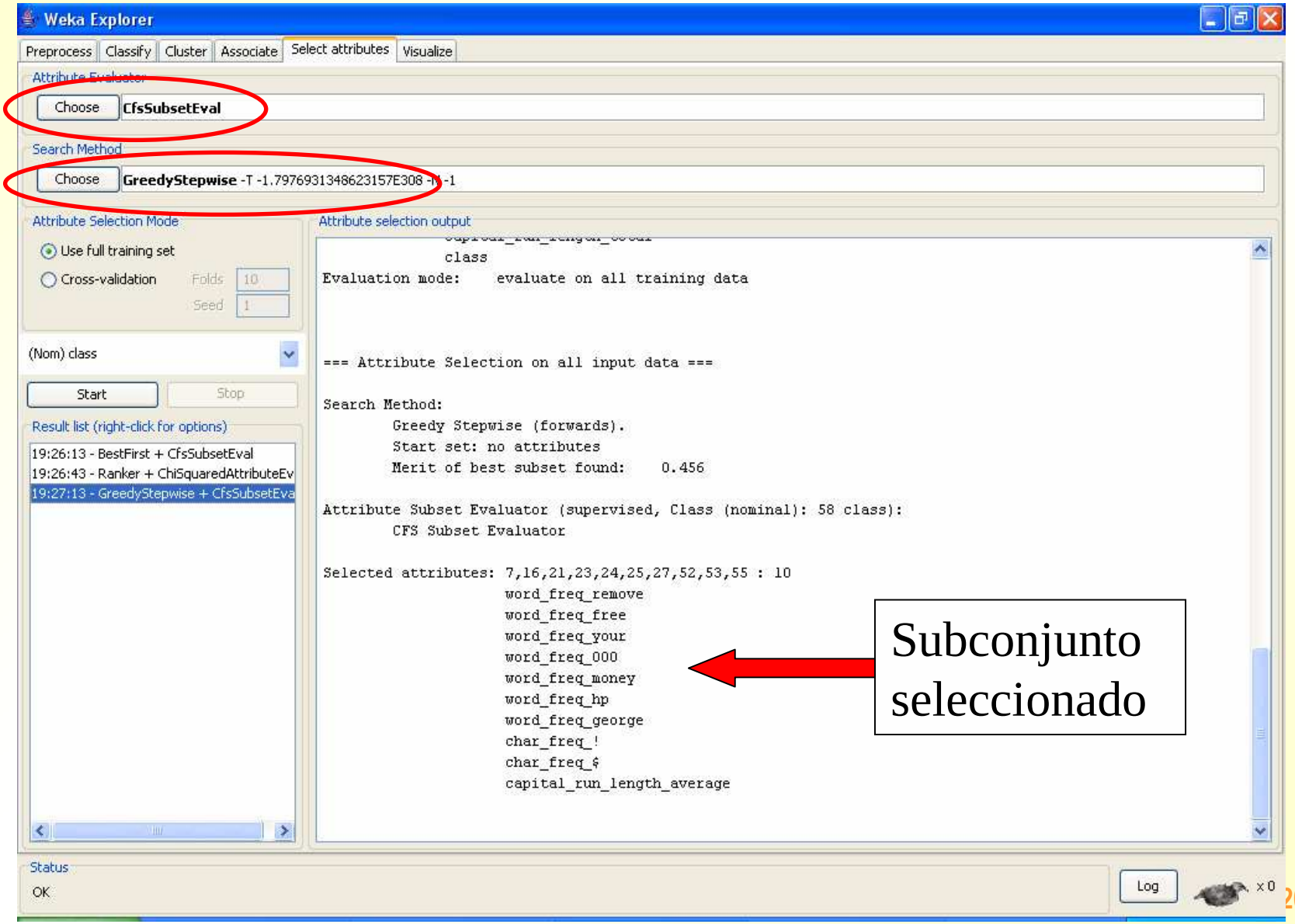


Atributos ordenados por importancia (1º y 2º: char_freq_! y char_freq_$)

Selección Ranker con validación cruzada (mérito y rango medios)

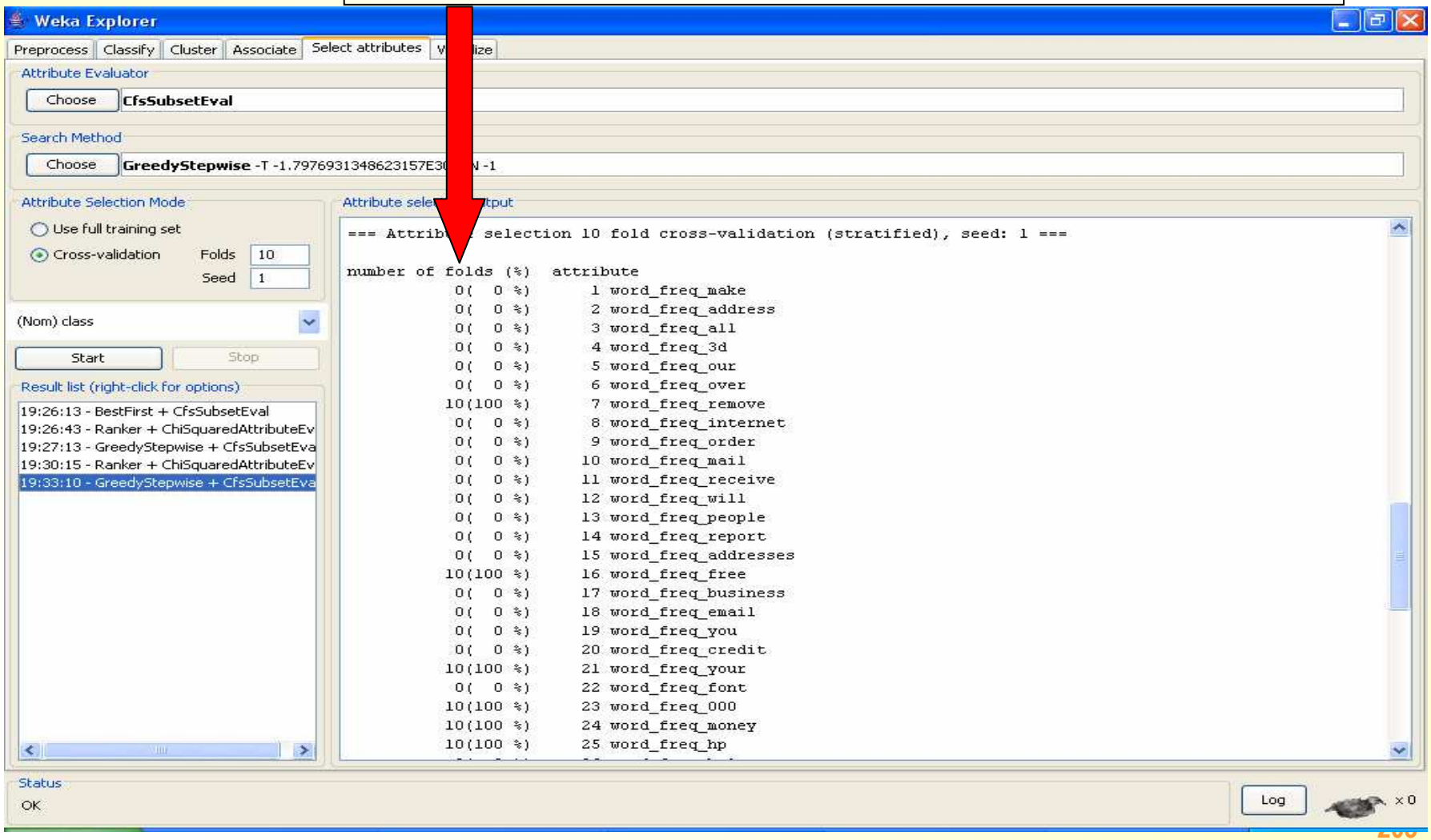


Selección Filter



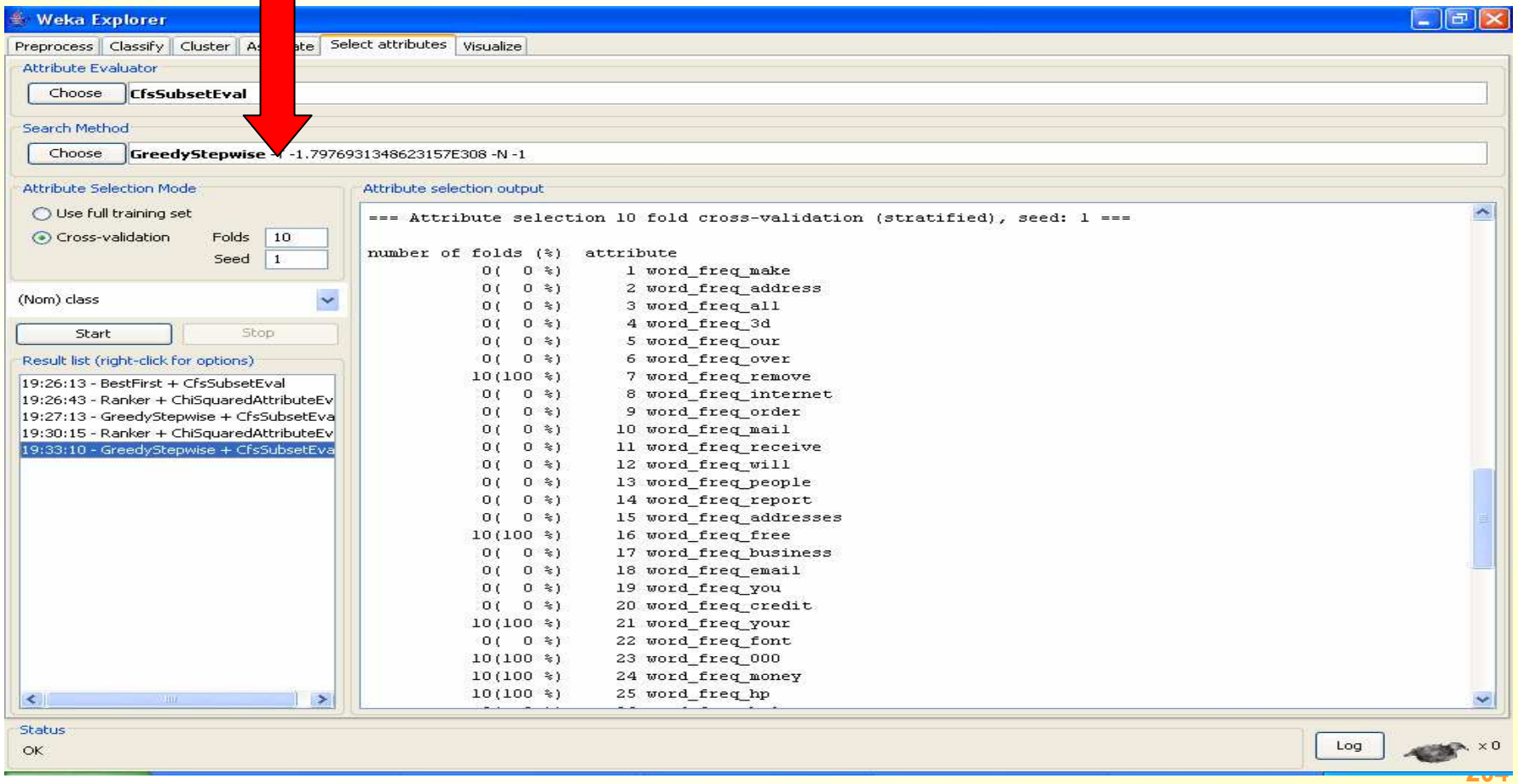
Selección Filter con validación cruzada

Número de folds en que el atributo fue seleccionado

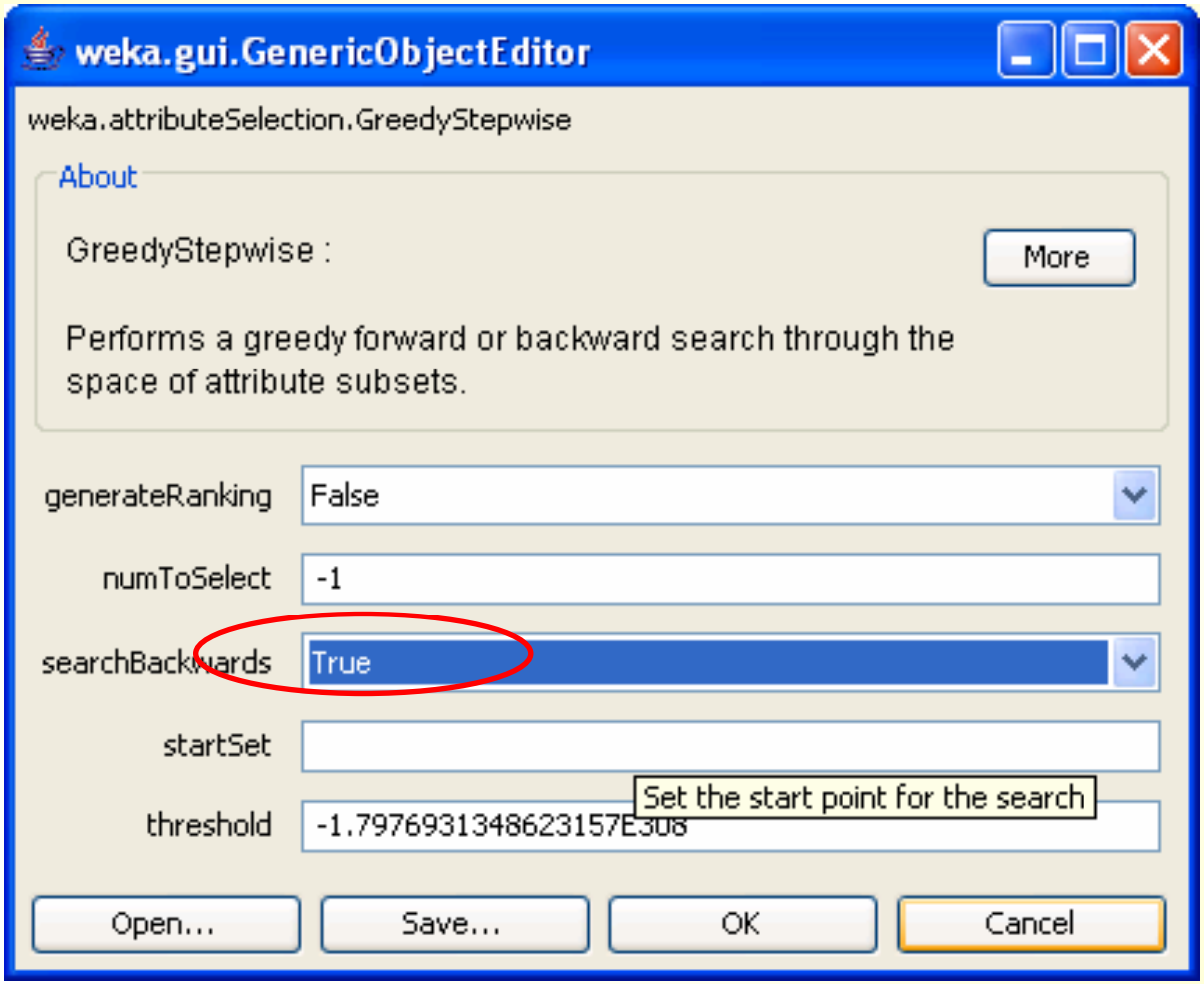


Búsqueda partiendo del conjunto total de atributos (backward en lugar de forward)

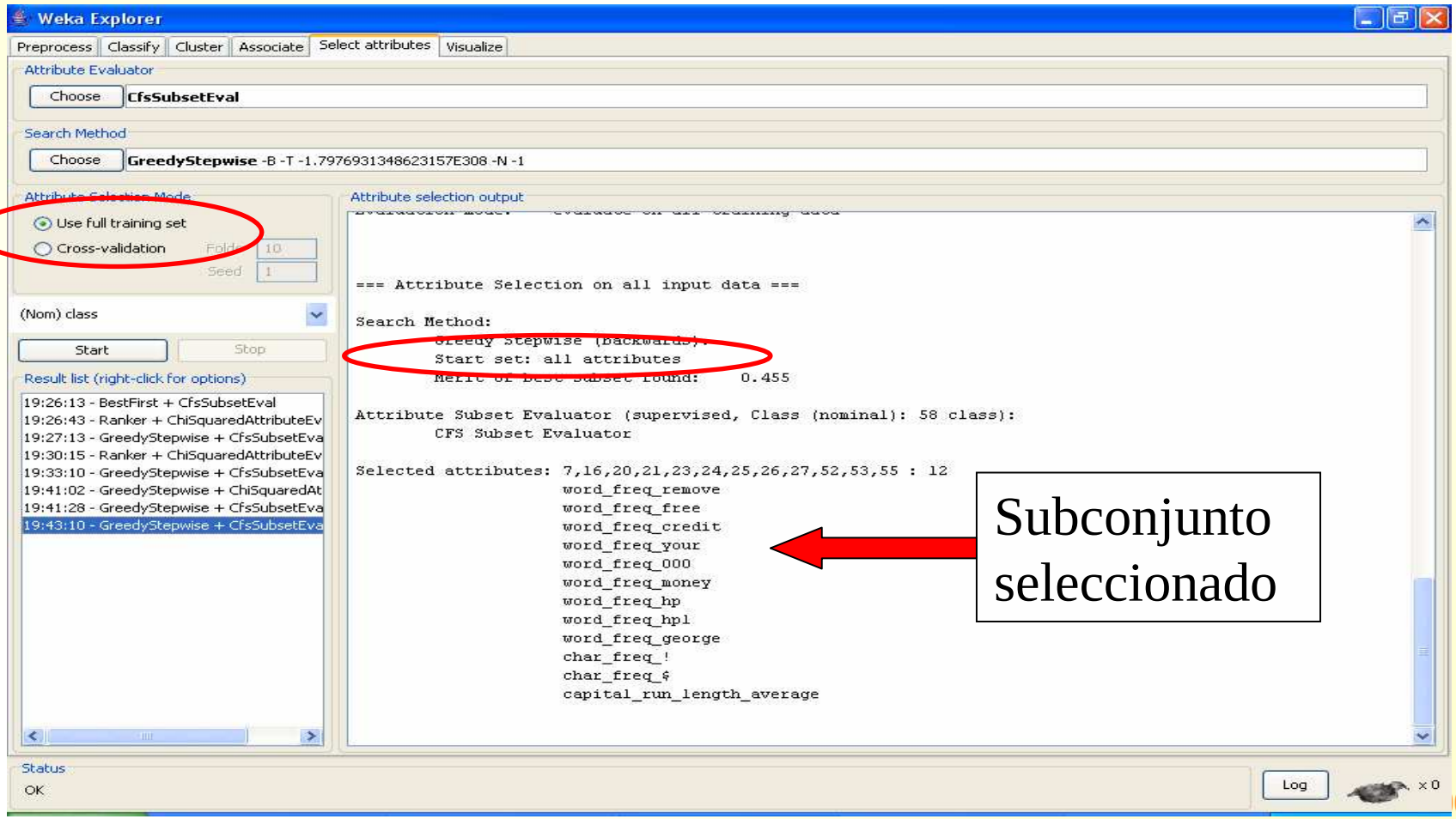
Click para parámetros de la búsqueda



Búsqueda partiendo del conjunto total de atributos (backward en lugar de forward)

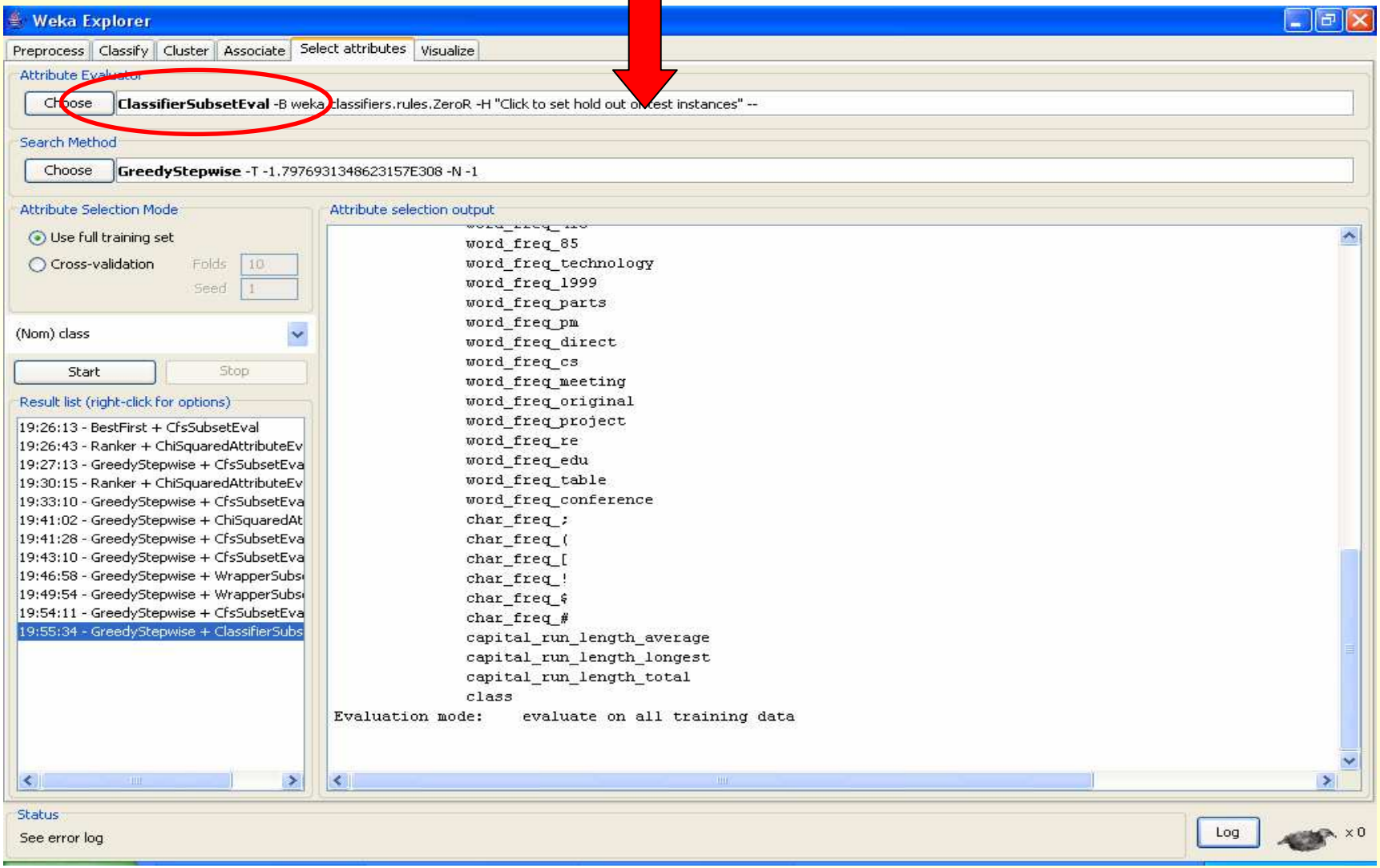


Búsqueda partiendo del conjunto total de atributos (backward en lugar de forward)

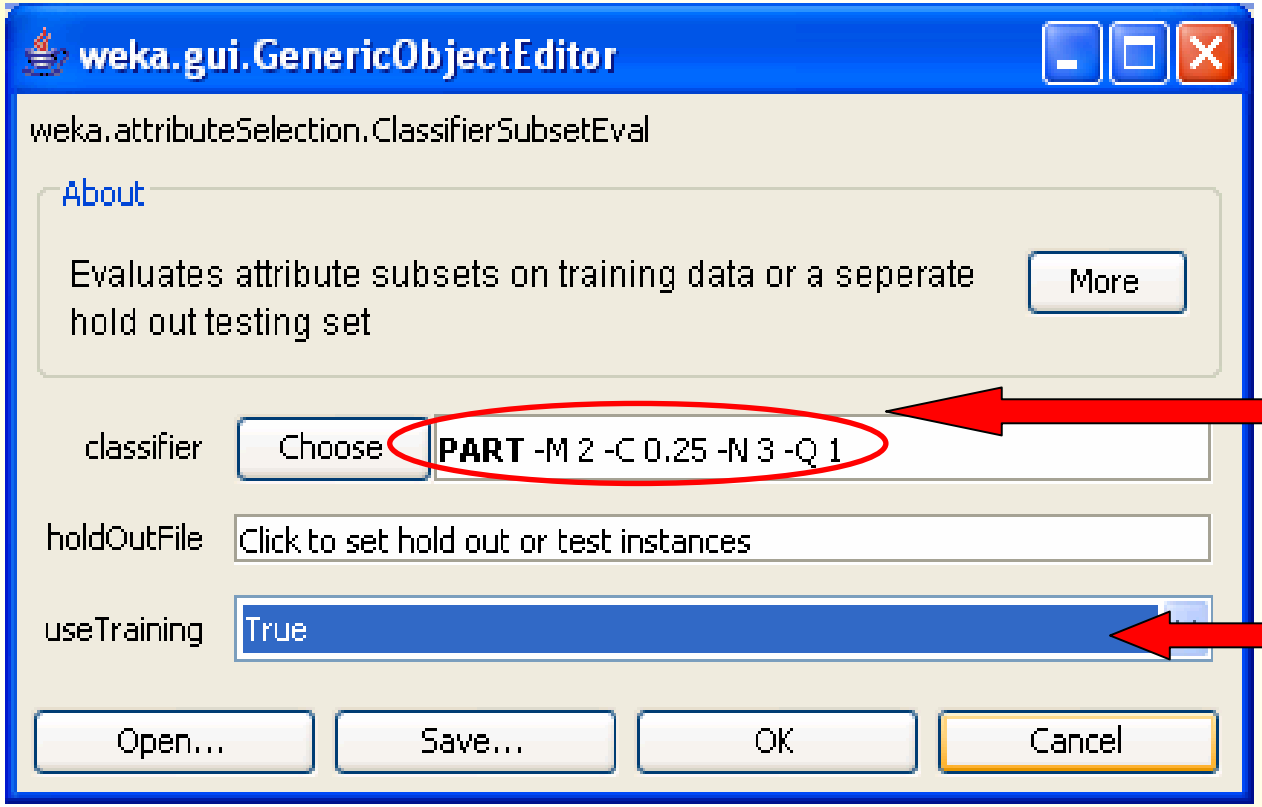


Método Wrapper

Click para parámetros de Wrapper



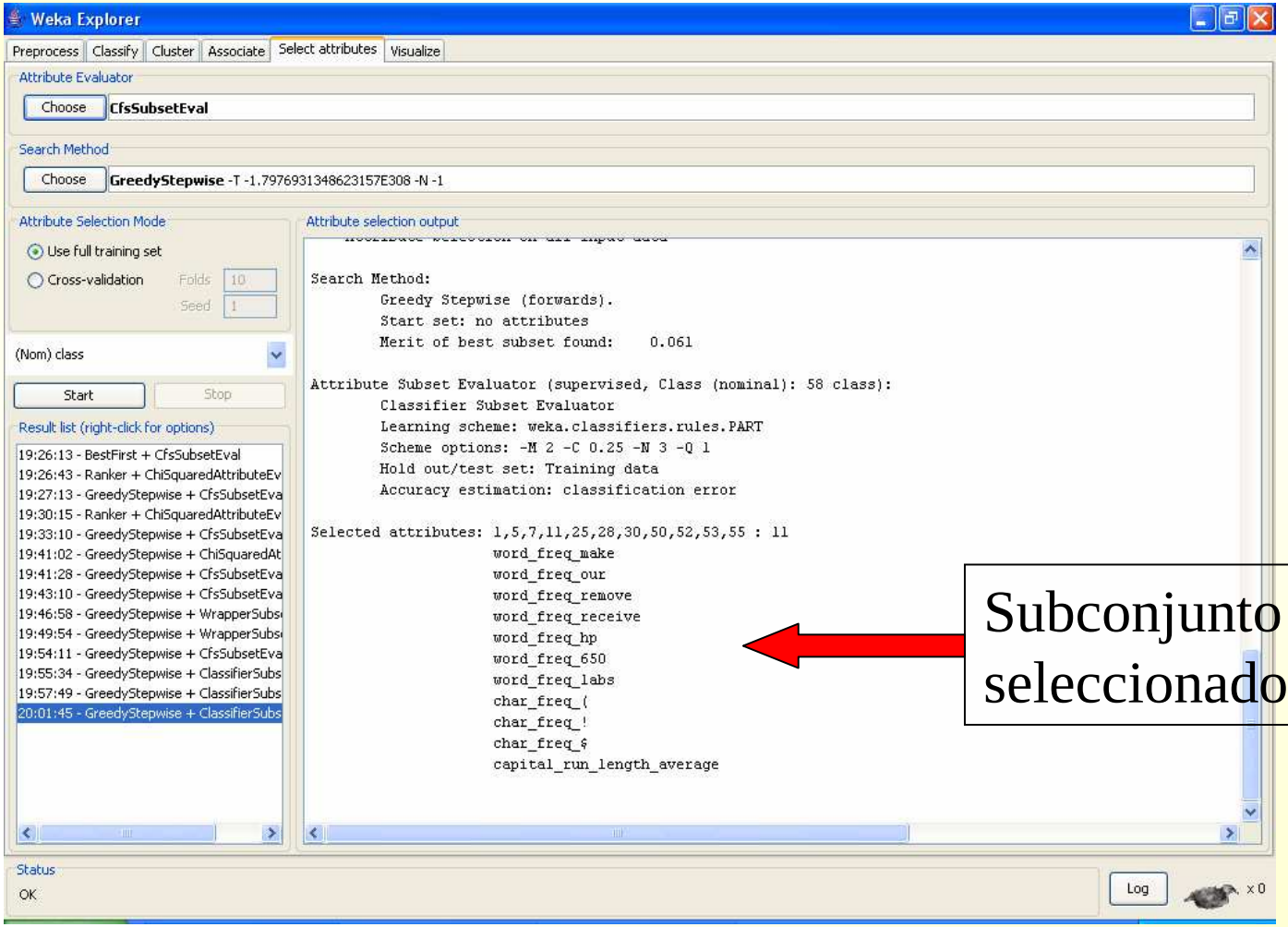
Selección parámetros Wrapper



Usaremos PART como clasificador

Usaremos el conjunto de entrenamiento para calcular los aciertos

Resultados Wrapper (¡lento!)

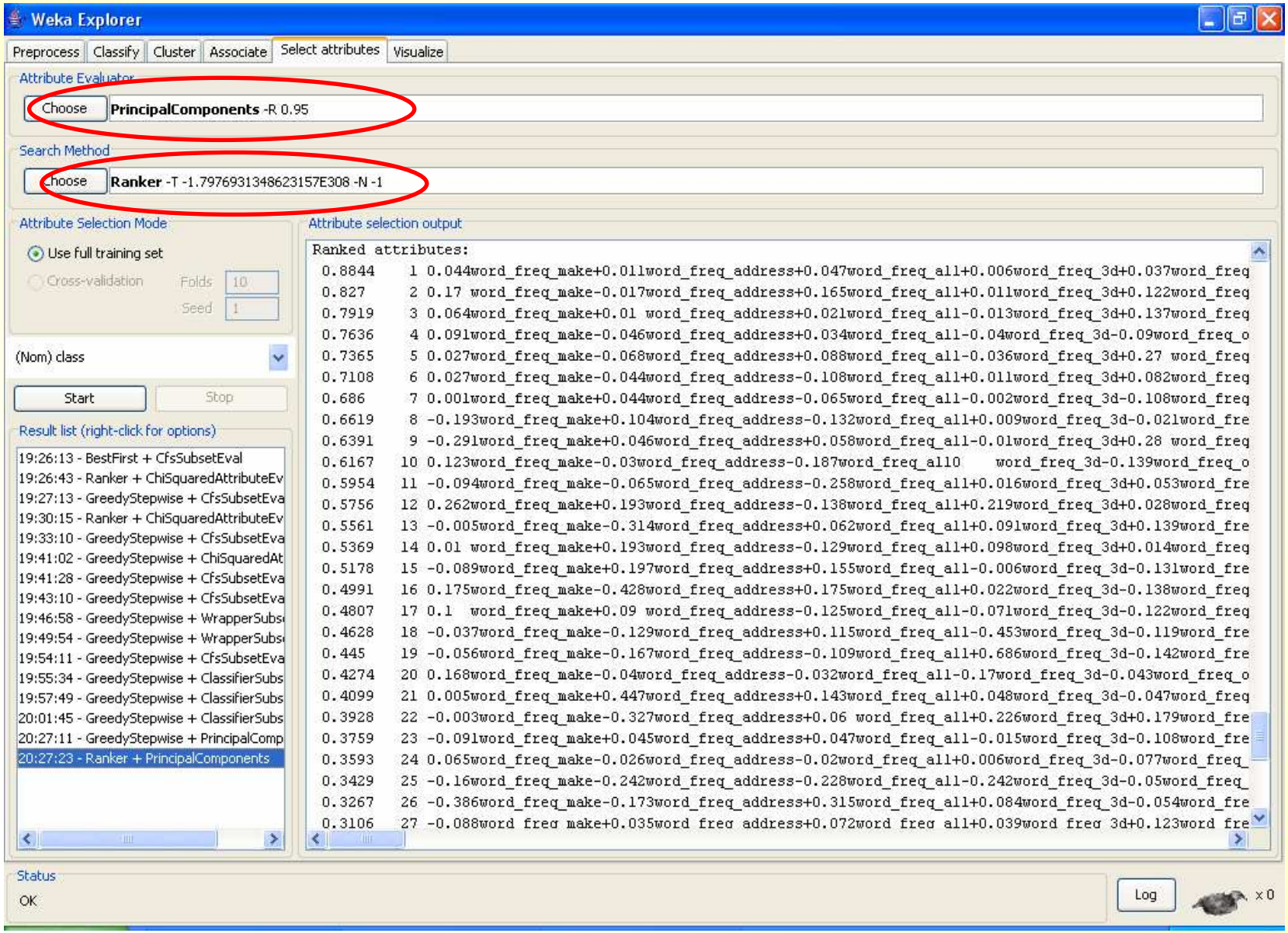


Subconjunto seleccionado

Selección con Principal Component Analysis (PCA)

- Este método construye nuevos atributos como combinación lineal de los anteriores
- Esos nuevos atributos están ordenados por importancia (varianza explicada)
- Se puede reducir la dimensionalidad escogiendo sólo algunos de los atributos

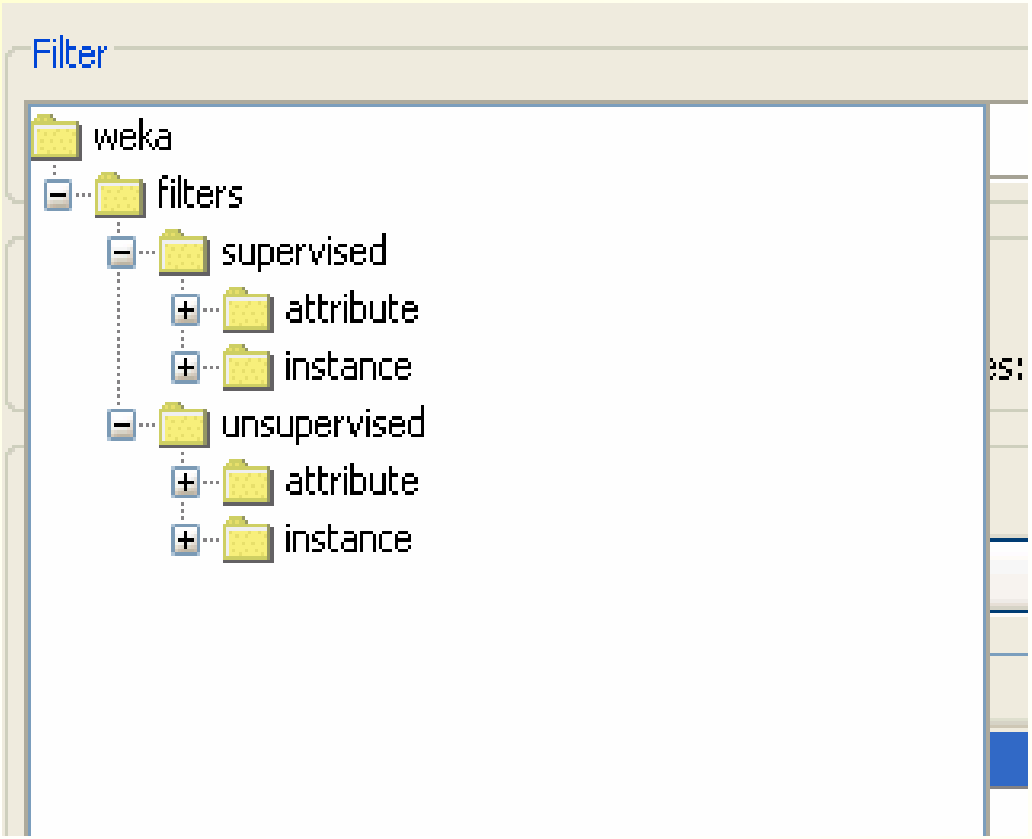
Resultados PCA



Notas selección de atributos

- Permite ver por pantalla los atributos seleccionados
- Pero no permite utilizar esa selección de atributos automáticamente para clasificar
- Para ello es necesario ir a la pestaña “preprocess” y seleccionar el filtro “Attribute Selection”

Filtros (pestaña de preproceso)

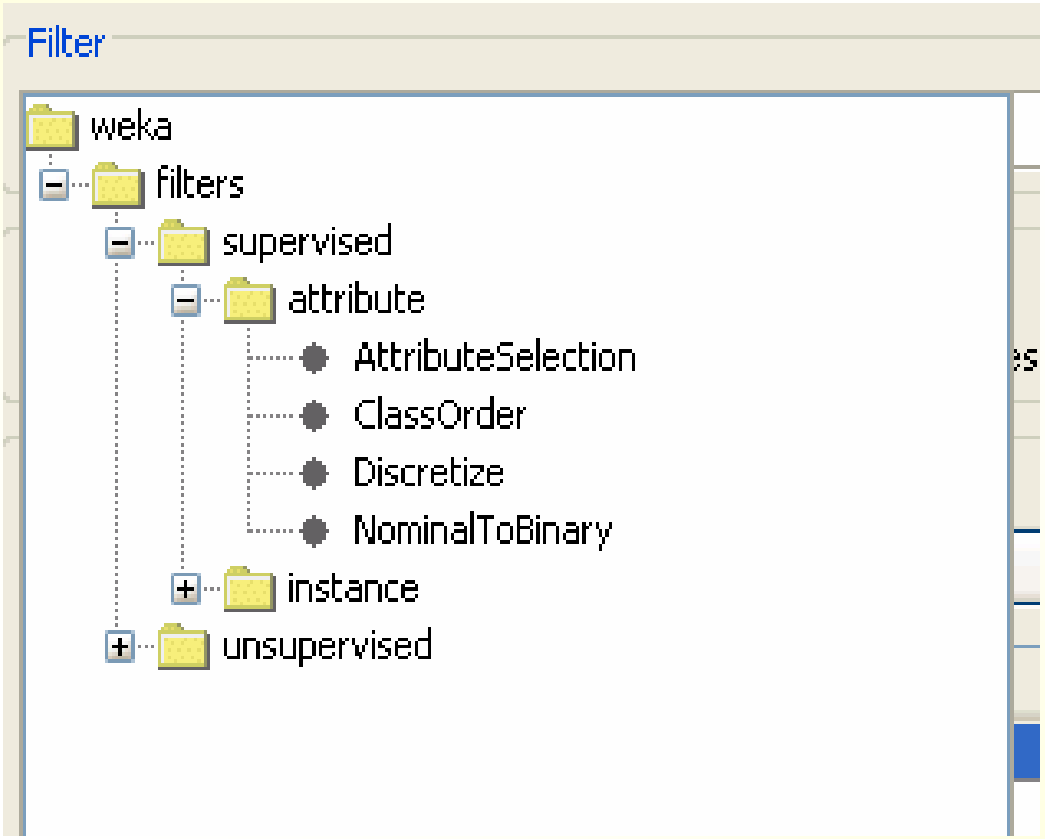


*Supervisados (tienen en cuenta la clase)

*No supervisados

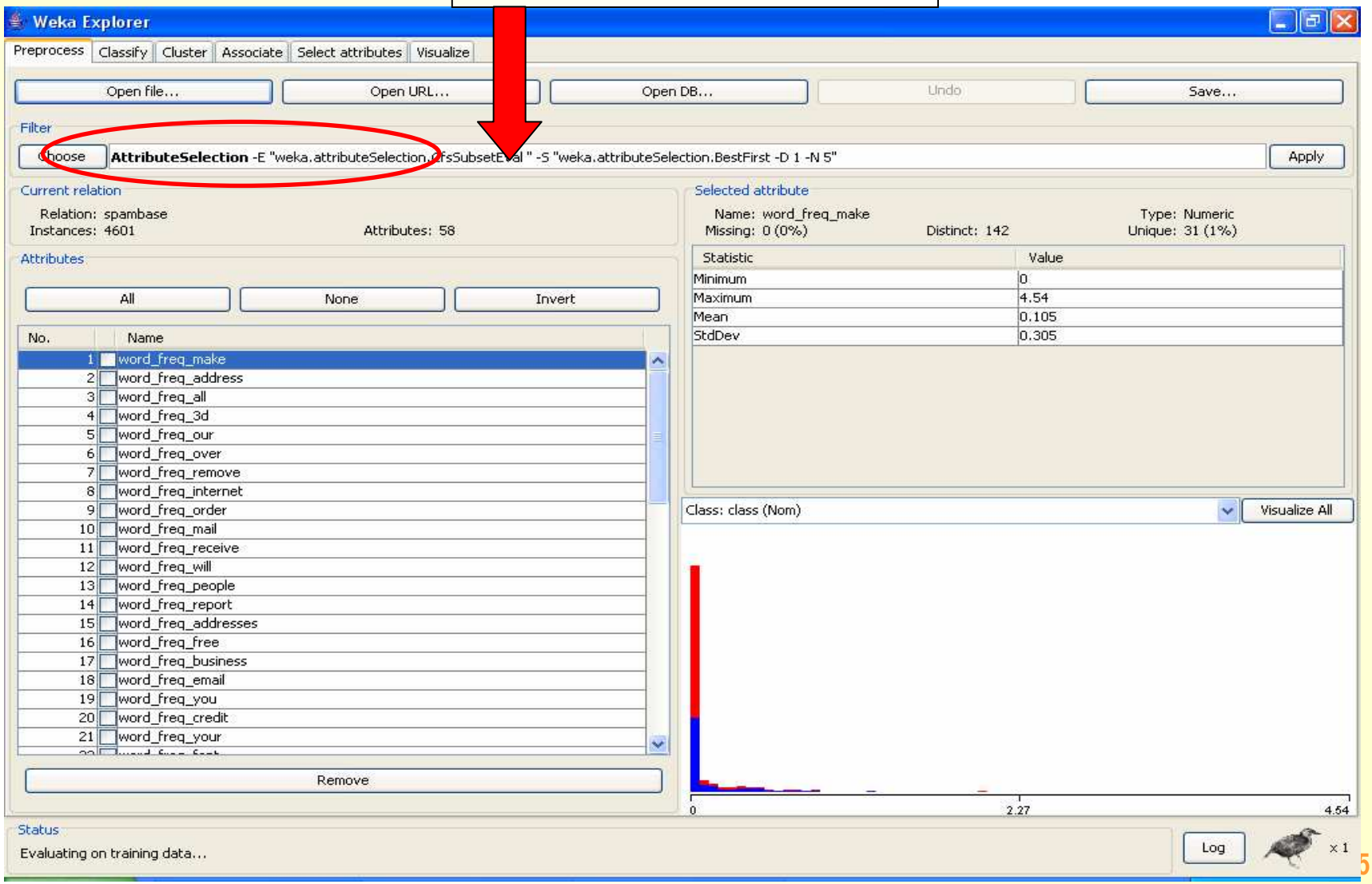
De atributos y de instancias (datos)

Filtro de selección de atributos

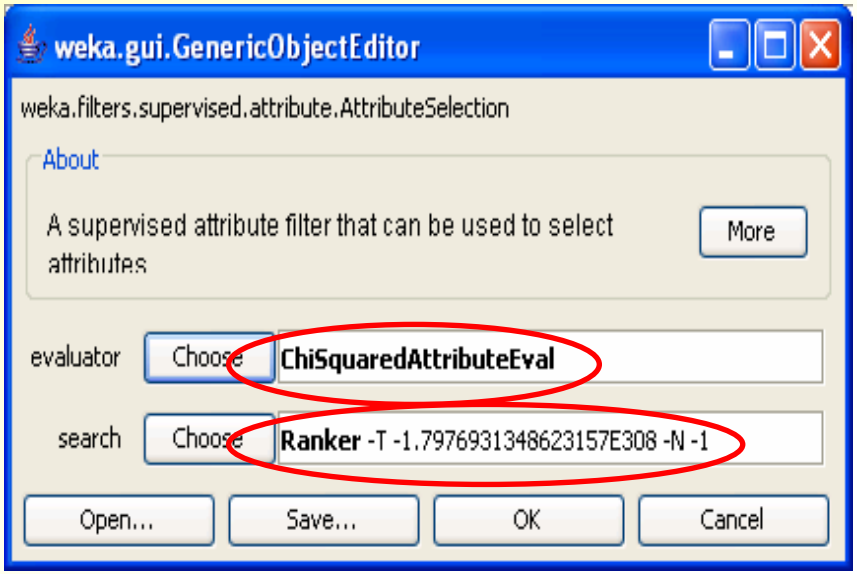


Filtro de selección de atributos

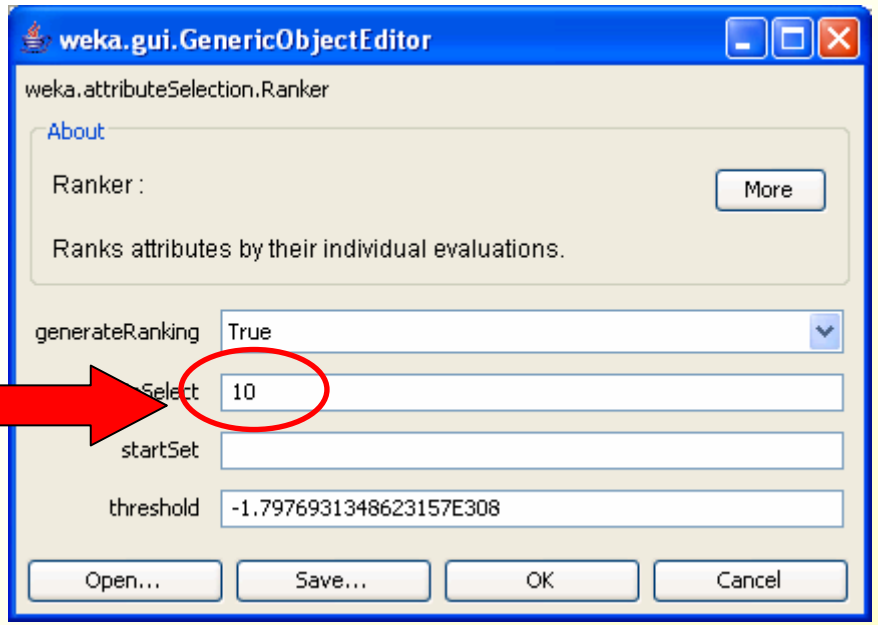
Click para parámetros



Parámetros de selección de atributos

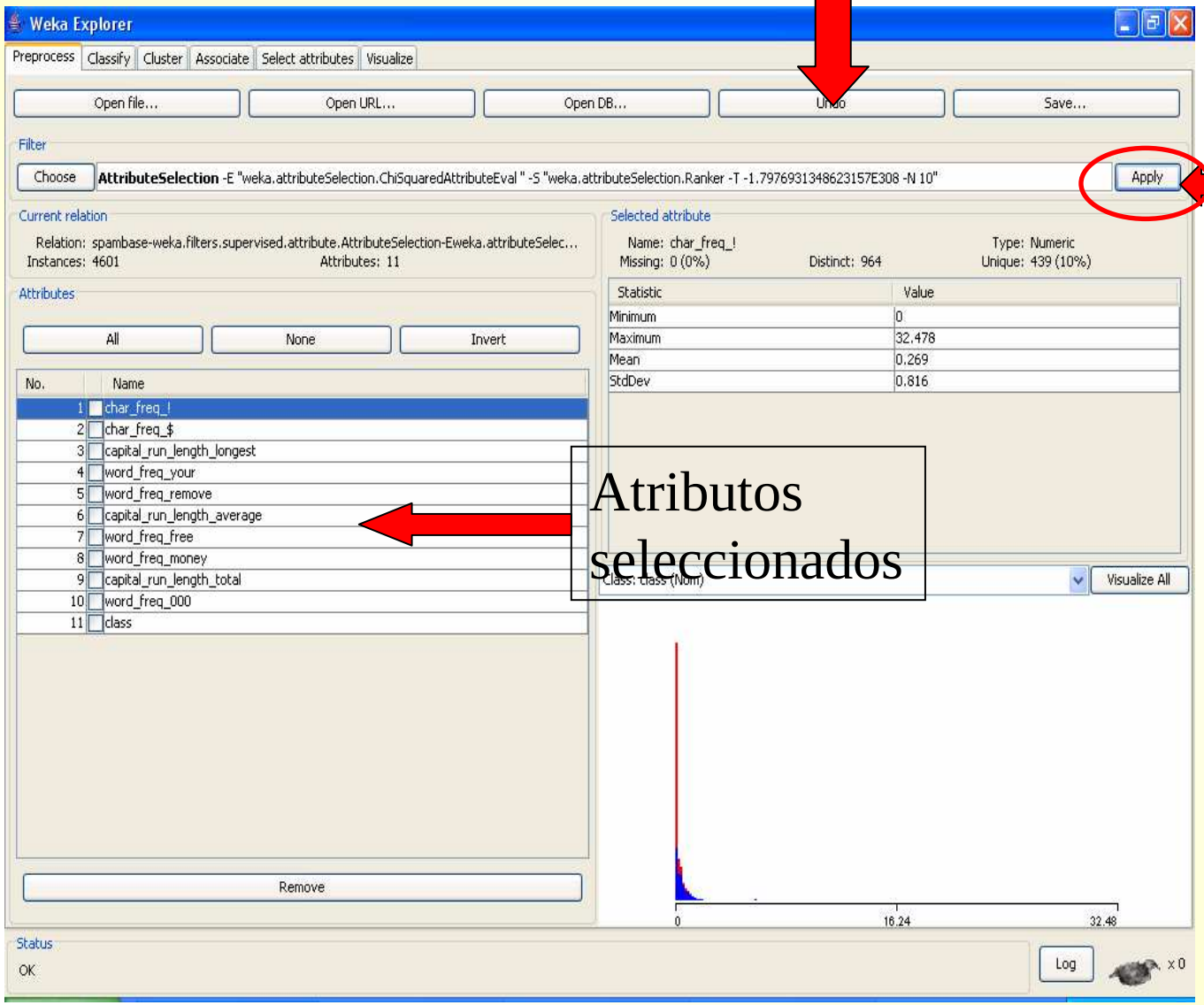


Seleccionaremos los 10 mejores atributos, tras la ordenación



Resultados de la selección

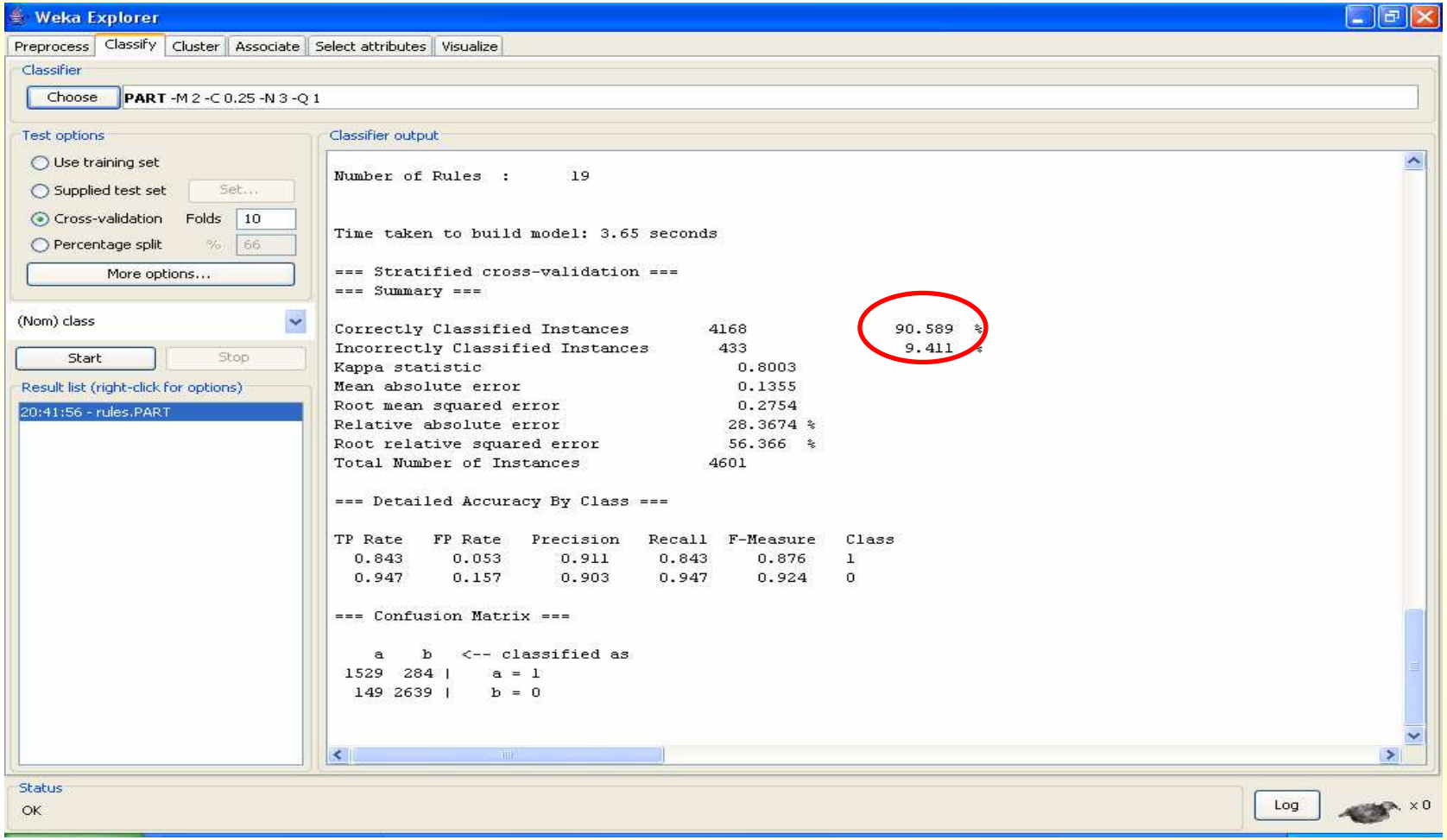
Podemos deshacer los cambios



¡Hay que pulsar Apply!

Atributos seleccionados

Resultados de la clasificación con los nuevos atributos



CURVAS DE APRENDIZAJE EN WEKA

http://weka.wiki.sourceforge.net/Learning+curves

- MINERÍA DE TEXTOS

Minería de Textos

- Se trata de realizar tareas de minería de datos, donde los datos son textos (páginas web, periódicos, …)
- Un texto es un dato
- Pero no está representado como una lista de valores para ciertos atributos
- Por tanto, habrá que convertir cada texto a una represenación de atributos/valores

Minería de textos. Tareas típicas

- **Clasificación**: asignación de **una** clases a cada documento
- **Categorización**: asignación de **varias** clases a cada documento
- **Agrupamiento (clustering)** de documentos: para organizar los documentos jerárquicamente, según alguna medida de similitud

Minería de textos. Ejemplos de aplicación

- Ordenación automática de correo electrónico en carpetas, según va llegando (**categorización** o **clasificación**)
- Filtro automático de spam/no-spam (**clasificación** binaria)
- Filtro de noticias de un periódico relevante/no-relevante para el usuario (**clasificación**)
- Un agente (programa) personal busca por internet páginas interesantes para el usuario (**clasificación**)
- Un agente de un directorio de páginas (como el directorio google) va buscando páginas y las **categoriza** automáticamente
- Para crear inicialmente la estructura de categorías, sería necesario partir de un conjunto grande de páginas y, o bien categorizarlas a mano, o bien utilizar un algoritmo de **clustering** para descubrir categorías

Minería de textos. Fases

1. Generación de corpus: obtención de gran cantidad de páginas Web y etiquetado, para el aprendizaje
2. Preprocesado: eliminación de códigos html, signos de puntuación, palabras con poca semántica, etc.
3. Generación de atributos: representación de los textos mediante atributos-valores (tabla datos x atributos)
4. Reducción de dimensionalidad
5. Aplicación de algoritmos de clasificación o agrupamiento

Minería de textos. Generación de corpus

- Se pueden obtener de diversas fuentes, como directorios:
  - Directorio google: páginas preclasificadas por temas
  - Corpus standard de noticias reuters-21578
  - Etc.

Minería de textos. Corpus standard

- Reuters: 10700 documentos etiquetados con 30000 términos y 135 categorías (21MB). Un documento pertenece a múltiples categorías
- OHSUMED: 348566 abstracts de revistas médicas. 230000 términos (400MB). Cada uno etiquetado con múltiples palabras clave.
- 20NG: 18800 mensajes de Usenet clasificados en 20 temas. 94000 términos (95MBs). Jerarquía de clases/etiquetas con 5 clases en el nivel más alto y 20 en el más bajo
- Industry: 10000 homepages de compañías de 105 sectores industriales (publicidad, carbón, ferrocarril, ...). Jerarquía con 80 clases arriba y 105 abajo.

Minería de textos. Preproceso

- Eliminación de:
  - Códigos html innecesarios (<body>, <p>, <href ...>, etc.)
  - Números
  - Signos de puntuación
  - Palabras con poca semántica (artículos, preposiciones, conjunciones, ...): *stopwords (*buscar en google spanish stopwords*)*

Minería de textos. Generación de atributos

- Es necesario convertir el texto plano en algo caracterizado por atributos (tabla datos*atributos), para poder aplicar algoritmos de minería de datos
- Técnicas:
  - Bolsas de palabras (bags of words)
  - Frases
  - N-gramas
  - Categorías de conceptos

Bolsas de palabras

- Un texto se representa como un vector con tantos componentes como palabras haya en el texto. Cada componente contiene la frecuencia de la palabra en el texto
- Ignora el orden de aparición de las palabras
- Ejemplo: “En un agujero en el suelo vivía un Hobbit. No un agujero húmedo, sucio, repugnante, con restos de gusanos y olor a fango, era un agujero Hobbit ...”

EN	UN	AGUJERO	EL	SUELO	HOBBIT	...
A1	A2	A3	A4	A5	A6	...
2	4	3	1	1	2	...

N-gramas

- En ocasiones, los atributos relevantes son **secuencias** de palabras.
  - Ej: computer science, source code, inteligencia artificial, minería de datos, ...
- N-gramas: como *bag of words*, pero los atributos son todas las tuplas de palabras de longitud n
- Ejemplo (bigramas)

EN-UN	UN-AGUJERO	AGUJERO-EN	EN-EL	EL-SUELO	NO-UN	...
A1	A2	A3	A4	A5	A6	...
1	2	1	1	1	1	...

N-gramas

- Problema: hay demasiados (todas las combinaciones posibles de dos palabras)
- Los bigramas funcionan bien en algunos problemas pero mal en otros (ej: en zoología es suficiente con unigramas)

Frases

- Considera el documento como un conjunto de frases sintácticas
- Permite mantener el contexto de las palabras
- Ej: a cada palabra, se le añade la información de qué papel juega en la frase, gramaticalmente (nombre, adjetivo, verbo, …) o sintácticamente (sujeto, predicado, objeto directo, …).
  - Ej: “En un agujero en el suelo”
  - Agujero-sujeto, suelo-complemento_circunstancial

Categorías de conceptos

- Considera sólo la raíz morfológica (“stem”) de las palabras
- Ejemplo: información, informando, informador, ... Se representan como “inform”
- Reduce la dimensionalidad del *bag of words*
- Problemas: “informal” e “informar” se pueden confundir

Problemas de las representaciones

- Sinonimia: diferentes palabras con el mismo significado se representan con diferentes atributos (ej: coche y automóvil)
- Quasi-sinonimia: palabras relacionadas con la misma materia (ej: declaración y comunicado)
- Polisemia: palabras iguales con distinto significado (ej: la masa (cemento, concepto físico, panadero, Increible Hulk, …)
- Lemas: palabras distintas con la misma raíz (ej: informal e informante)

Reducción de dimensionalidad

- Eliminación de palabras muy muy poco frecuentes
- Eliminación de palabras muy frecuentes (suelen ser las “stopwords”: el, la, que, y, ...)
- Se pueden utilizar técnicas de selección de atributos típicas de minería de datos
- Se suele utilizar Indexación Semántica Latente (LSI), que es una técnica de proyección a espacios de dimensionalidad menor, similar a análisis de componentes principales (técnica existente en Weka: PCA)
- LSI también reduce los problemas de sinonimia y polisemia

Reducción de dimensionalidad. PCA

- Partimos de los atributos originales A1, A2, A3, …, An
- PCA crea combinaciones de atributos ortogonales (independientes) que cubran el máximo de varianza:
  - A1’ = 3*A1+2*A2- … - An
  - A2’ = A1-2*A2+ … +5* An
  - …
- **Elimina redundancia**. ejemplo: altura x peso
  - A’ = peso – altura; P’ = peso+altura
- Y los ordena. Nos podemos quedar con los M primeros mas relevantes (que expliquen, por ejemplo, el 90% de la varianza)
- Hay un método Weka de selección de atributos mediante PCA

Cálculo de frecuencias (term frequency)

- Las frecuencias de aparición ("term frequency") (x^{tf}_{ij}) de la palabra *i* en el documento *j* se suele calcular como:

$$x^{tf}_{ij} = \frac{n_{ij}}{\sum_{k \in j} n_{kj}})$$

- n_{ij} es el número de veces que la palabra *i* aparece en el documento *j*. El sumatorio suma las cuentas de todas las palabras en el documento *j*

Normalización "inverse document frequency"

- Posteriormente, las x^{tf}_{ij} se suelen normalizar con *inverse document frequency*:

$$x_{ij}^{tfidf} = x_{ij}^{tf} \log\left(\frac{|D|}{DF_i}\right) \quad \text{O bien} \quad x_{ij}^{tfidf} = x_{ij}^{tf} \log\left(\frac{1+|D|}{DF_i}\right)$$

- $|D|$ es el número de documentos y DF_i es el número de documentos en los que aparece la palabra *i*
- La idea es premiar aquellas palabras que aparecen en pocos documentos ($|D| >> DF_i$), porque se supone que son las mas discriminativas

Normalización "inverse document frequency"

- TFIDF es útil para problemas de clustering (aprendizaje supervisado)
- Para problemas de clasificación puede funcionar peor
- La razón es que en clasificación, para decidir si un atributo es relevante, habría que utilizar la clase
- Pero en TFIDF se le da más/menos importancia a un término independientemente de la clase

Minería de textos. Utilización práctica

- Se puede utilizar cualquier algoritmo de clasificación
- Naive Bayes ha sido usado con bastante éxito
- Ejemplos:
  - Uso de Naive Bayes para clasificar artículos en grupos de News (20 newsgroups). Conjunto de entrenamiento: 1000 artículos. Precisión obtenida: 89%
  - Newsweeder: clasificación de artículos de news en interesantes y no interesantes (etiquetados por el usuario).
  - Filtros de spam: clasificadores bayesianos de e-mails en spam y no-spam (ej: spamassassin)