uc3m | Universidad Carlos III de Madrid

OPENCOURSEWARE
APRENDIZAJE AUTOMÁTICO PARA EL ANÁLISIS DE DATOS
GRADO EN ESTADÍSTICA Y EMPRESA
Ricardo Aler

M.L. PIPELINE / METODOLOGÍA:

- Ajuste de hiper-parámetros
- Entrenamiento del modelo
- Evaluación del modelo
- Uso del modelo

?

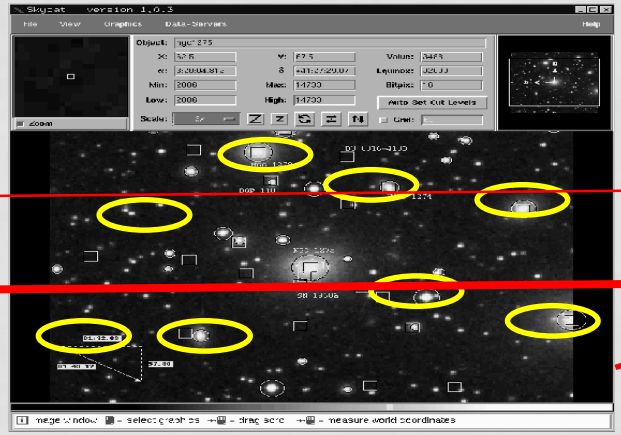
Uso del modelo

Datos Disponibles

Entrenamiento

Validación

Test



Ajuste de hiper-parámetros

60% 92% 70%

Con E y V
Y profundidad 2

Evalúa

83%

PREPROCESADO DE DATOS

REDUCCIÓN DE DIMENSIONALIDAD:

- SELECCIÓN DE ATRIBUTOS (FEATURE SELECTION)
- EXTRACCIÓN DE ATRIBUTOS (FEATURE EXTRACTION),

M.L. PIPELINE O METODOLOGÍA

- **Preprocesado:**
  - **Instancias**
  - **Atributos**
- Aprendizaje:
  - Ajuste de hiper-parámetros
  - Entrenamiento del modelo final
- Evaluación del modelo (train/test o validación cruzada)
- Uso del modelo

Cielo	Temperatura	Humedad	Viento	Tenis
sol	85	85	no	no
sol	80	90	si	no
nubes	83	86	no	si
lluvia	70	96	no	si
lluvia	68	80	no	si
lluvia	65	70	si	no
nubes	64	65	si	si
sol	72	95	no	no
sol	69	70	no	si
lluvia	75	80	no	si
sol	75	70	si	si
nubes	72	90	si	si
nubes	81	75	no	si
lluvia	71	91	si	no

PREPROCESADO

- Instancias / datos:
  - Quitar outliers
  - Muestreo (para tener una muestra representativa pero más pequeña)
    - Concepto de curva de aprendizaje (learning curve)
  - Rebalanceo, en problemas de muestra desbalanceada (undersampling, oversampling, SMOTE – Synthetic Minority Over-sampling Technique, …)
  - Quitar datos "ruidosos": edición de Wilson, … (paquete 'NoiseFiltersR' de R)

- Atributos:
  - Normalización (hacer que todos los rangos estén en 0-1): normalizeFeatures (mlR)
  - Variables dummy / codificación one-hot: puede ser conveniente transformar variables categóricas (discretas) con n valores en n (o n-1) variables binarias: createDummyFeatures (mlR)
  - Imputación (¿qué hacer si hay valores faltantes (NAs)?): imputation (mlR), Multivariate Imputation by Chained Equations (mice en R)
  - Creación de atributos / feature engineering (ej: velocidad, a partir de v_x y v_y, en un problema de predicción de energía eólica)

EDICIÓN DE WILSON: QUITAR DATOS RUIDOSOS EN CLASIFICACIÓN

- Edición de Wilson: elimina instancia $\mathbf{x}_i$ si es clasificada incorrectamente por sus k vecinos:
  - Excepciones en el interior de una clase
  - Algunos puntos en la frontera (suaviza las fronteras)
- Edición de Wilson repetida: repetir hasta que no se pueda aplicar mas

PREPROCESADO

- Instancias / datos:
  - Quitar outliers
  - Remuestreo (para tener una muestra representativa pero más pequeña)
  - Rebalanceo, en problemas de muestra desbalanceada (undersampling, oversampling, SMOTE, …)
  - Quitar datos "ruidosos": regla de edición de Wilson, … (paquete 'NoiseFiltersR' de R)
- Atributos:
  - …
  - **Selección de atributos (feature selection)**
  - **Extracción de atributos (feature extraction): transformación de atributos**

Selección de atributos

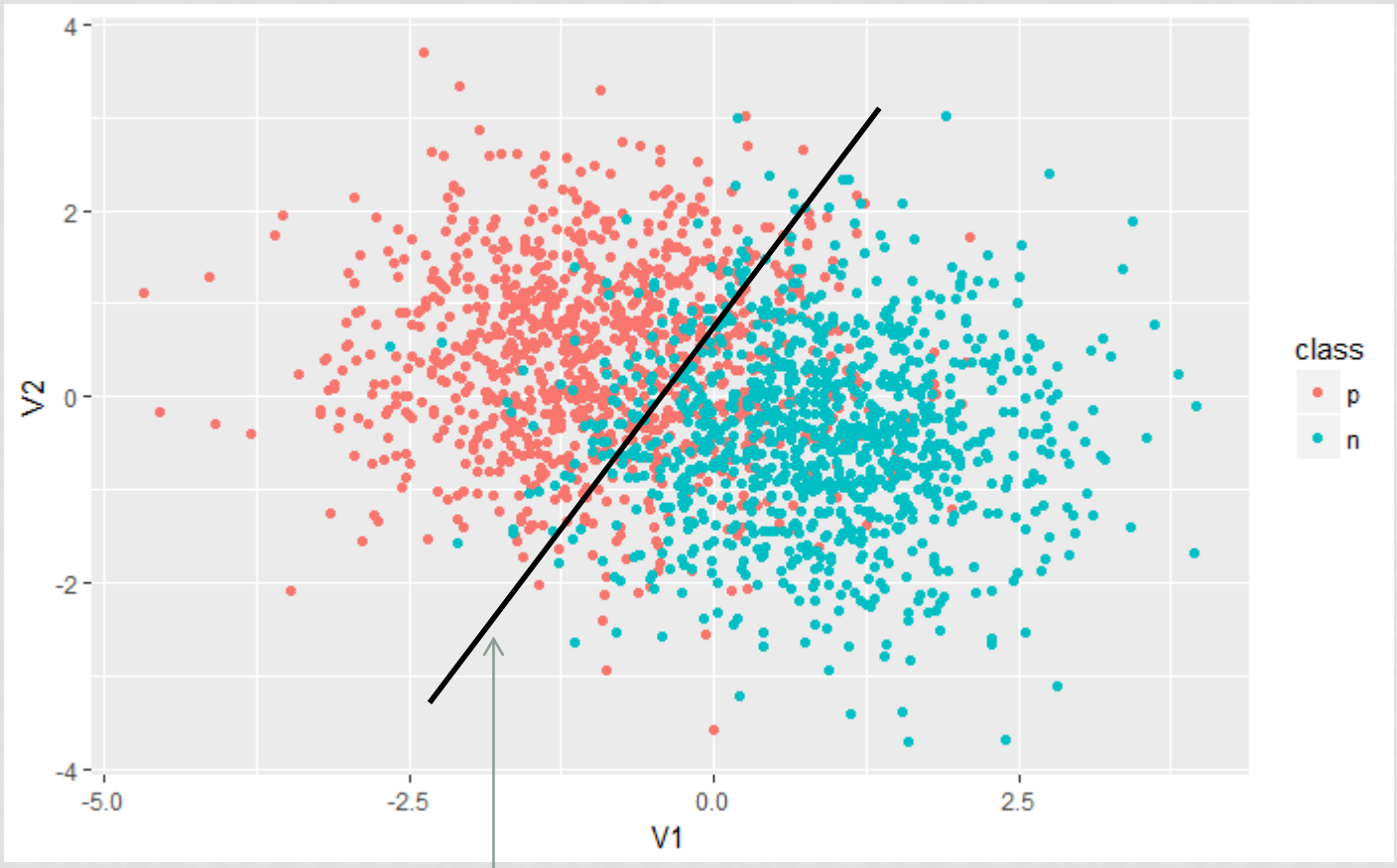
- Algunos atributos pueden ser **redundantes** (como "salario" y "categoría social")
  - Hacen más lento el proceso de aprendizaje
  - Pueden confundir a algunos clasificadores (como el Naive Bayes)

- Otros son **irrelevantes** (como el DNI para predecir si una persona va a devolver un crédito)
  - Sabemos que son dañinos para knn, aunque también pueden llegar a perjudicar a C4.5

- **Maldición de la dimensionalidad**: El número de datos necesarios para construir un modelo predictivo (que no sobreadapte) puede crecer exponencialmente con el número de dimensiones (atributos)

- En ocasiones es útil tener el conocimiento de qué atributos son relevantes para una tarea
- Cuantos menos atributos, más simple es el modelo y mas fácil de entender

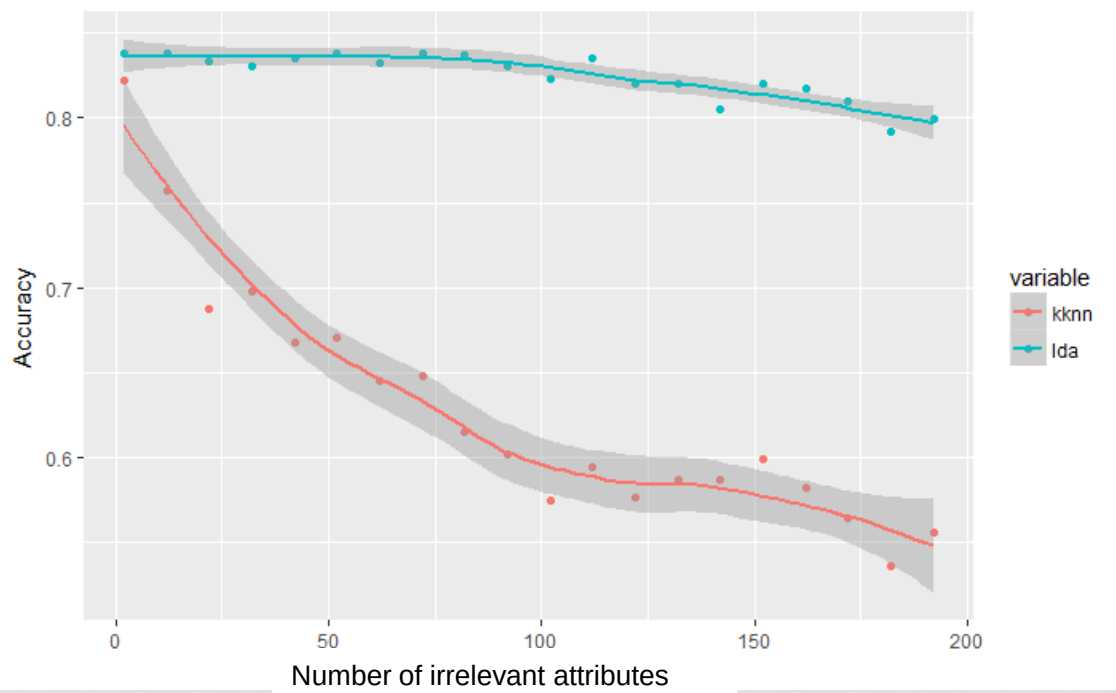
ATRIBUTOS IRRELEVANTES



- Datos artificiales generados a partir de dos gausianas bi-variadas
- 1000 instancias clase roja, 1000 instancias clase azul
- Frontera óptima = línea

EFECTO DE ATRIBUTOS IRRELEVANTES EN KNN Y MODELOS LINEALES

- Se van añadiendo atributos irrelevantes (columnas con valores aleatorios)



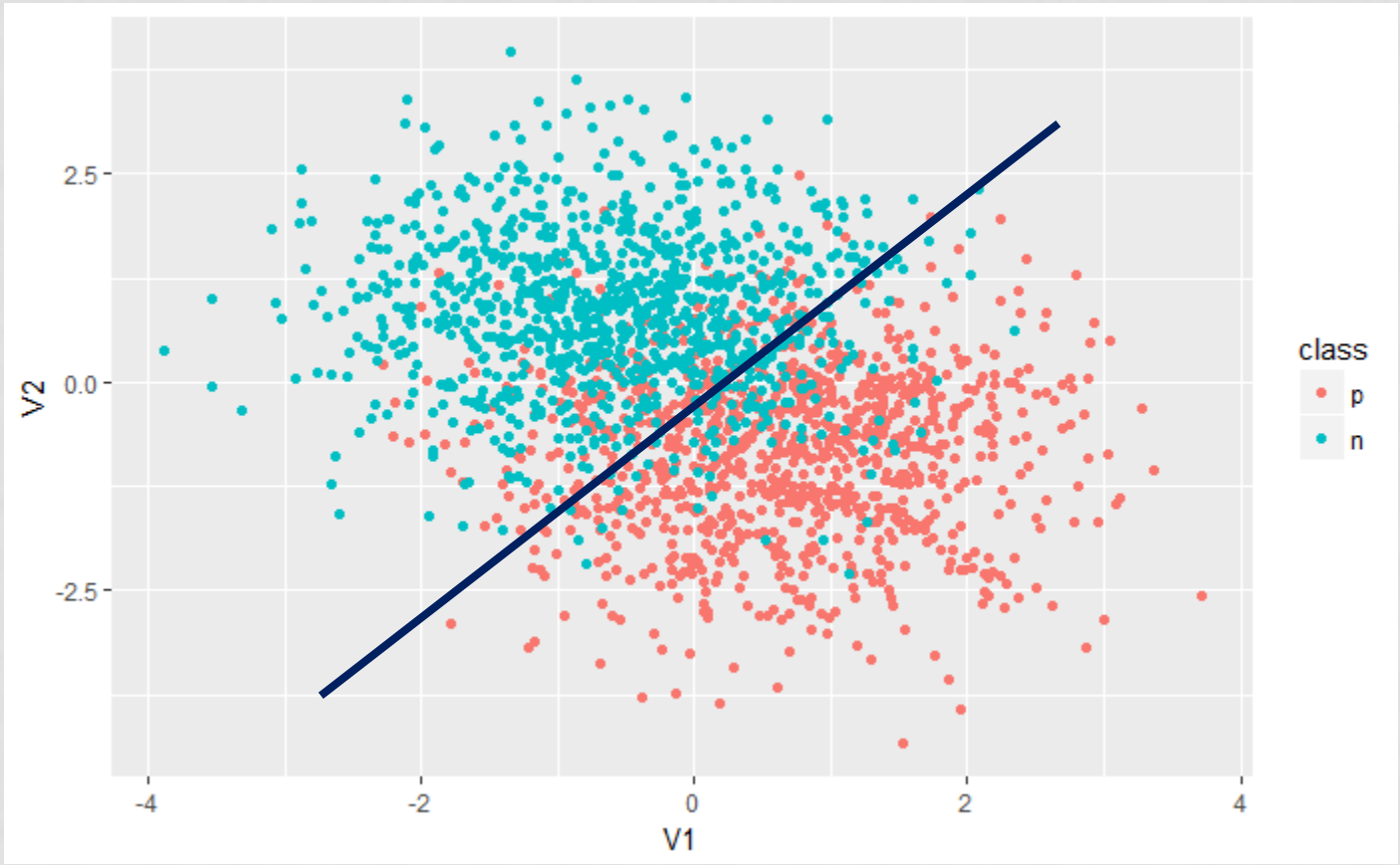
LDA = Linear Discriminant Analysis
KNN = K-nearest neighbour

MALDICIÓN DE LA DIMENSIONALIDAD

- Superficies crecen exponencialmente con el número de dimensiones d.
- Ejemplo: superficie de una esfera
  - 2D: $2\pi r = 10$ instancias
  - 3D: $4\pi r^2 = 100$ instancias
  - 4D: $2\pi^2 r^3 = 1000$ instancias
  - 50D: $2\pi^2 r^{49} = 10^{49}$ instancias
  - dD: $O(r^{d-1})$
- Idea: si la superficie de separación es complicada, el número de instancias necesarias para definir esa frontera puede ser proporcional a la superficie de esa frontera, la cual crece exponencialmente con el número de dimensiones.

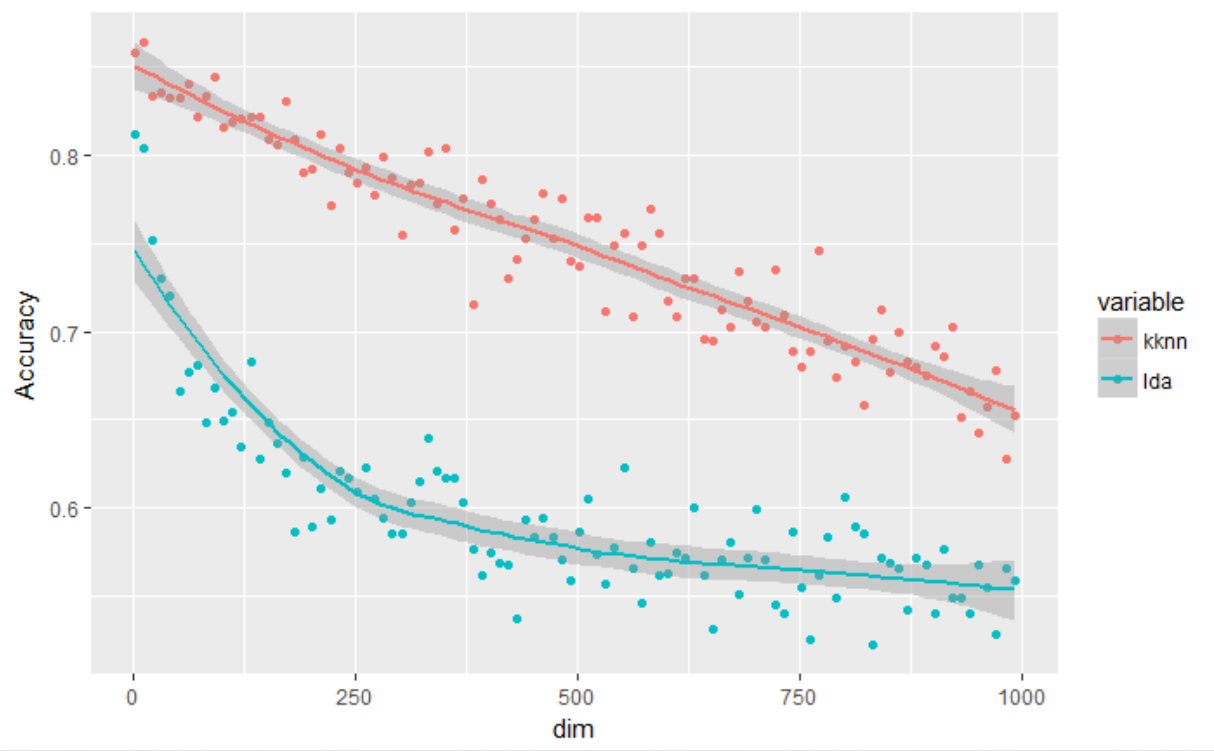
EJEMPLO DE PROBLEMA DE CLASIFICACIÓN EN 2D

- 1000 datos positivos, 1000 datos negativos



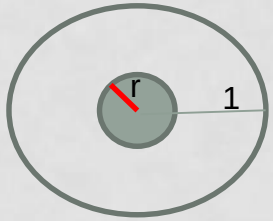
EJEMPLO DE PROBLEMA DE CLASIFICACIÓN EN 2D

- Cómo cambia el porcentaje de aciertos al incrementar la dimensionalidad (KNN, y LDA, un clasificador lineal)



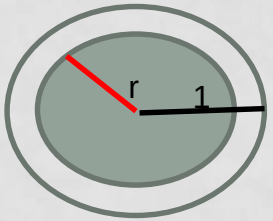
INTUICIÓN DE LA MALDICIÓN DE LA DIMENSIONALIDAD

- En dos dimensiones, el "volumen" de una esfera es πr^2 , o sea, proporcional a r^2.
- En tres dimensiones, el volumen es proporcional a r^3.
- En d dimensiones, proporcional a r^d.
- Supongamos que tenemos 10^4 datos dentro de una esfera de radio = 1. La densidad es $10^4/r^d = 10^4$.
- Supongamos que queremos encontrar el vecino más cercano de un dato situado en el centro de la esfera. ¿A qué distancia **r** estará?



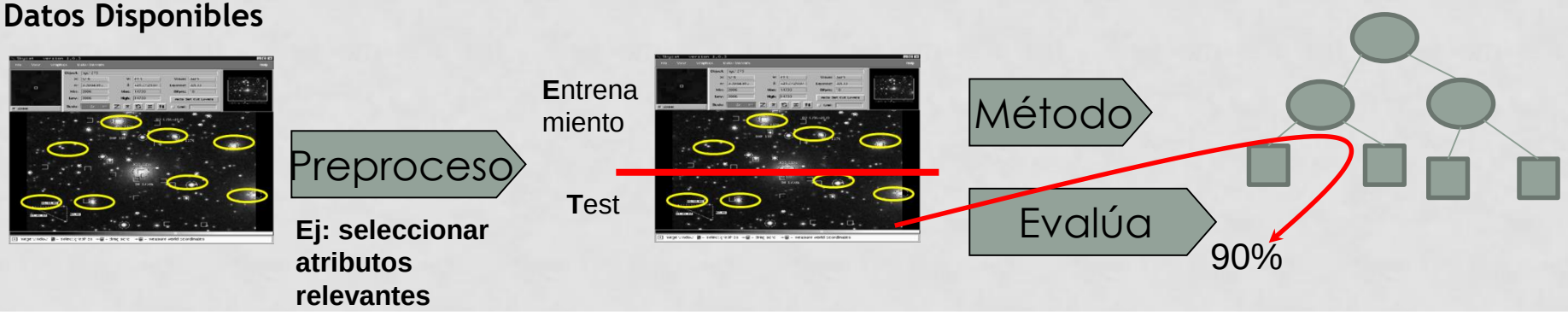
INTUICIÓN DE LA MALDICIÓN DE LA DIMENSIONALIDAD

- Supongamos que queremos encontrar el vecino más cercano de un dato situado en el centro de la esfera. ¿A qué distancia **r** estará?
- Una esfera de radio **r** contendrá densidad * volumen(**r**) datos.
- O sea, número datos = $10^4 * r^d$.
- Si queremos encontrar al menos un vecino, $10^4 * r^d >= 1$
- Despejamos r: $r >= 10^{(-4/d)}$
- Con d=2, r = 0.01para encontrar el vecino más cercano
- Con d=3, r = 0.04
- Con d=50, r = 0.86
- Con d=100, r = 0.91 !!!



¿DÓNDE COLOCAR LA FASE DE PREPROCESO?

- Distinguimos dos tipos de pre-proceso:
  - No dependiente de los datos (ej: quitar el primer atributo porque es un identificador)
  - Dependiente de los datos (prácticamente, todos los demás)
- Existe la tentación de aplicar el pre-proceso a los datos disponibles, y después hacer evaluación y entrenamiento del modelo final.
- Pero esto no es del todo correcto, porque para hacer el preproceso, tenemos que usar datos que después serán usados en test.
- El preproceso dependiente de los datos es realmente parte del proceso de entrenamiento del modelo final.

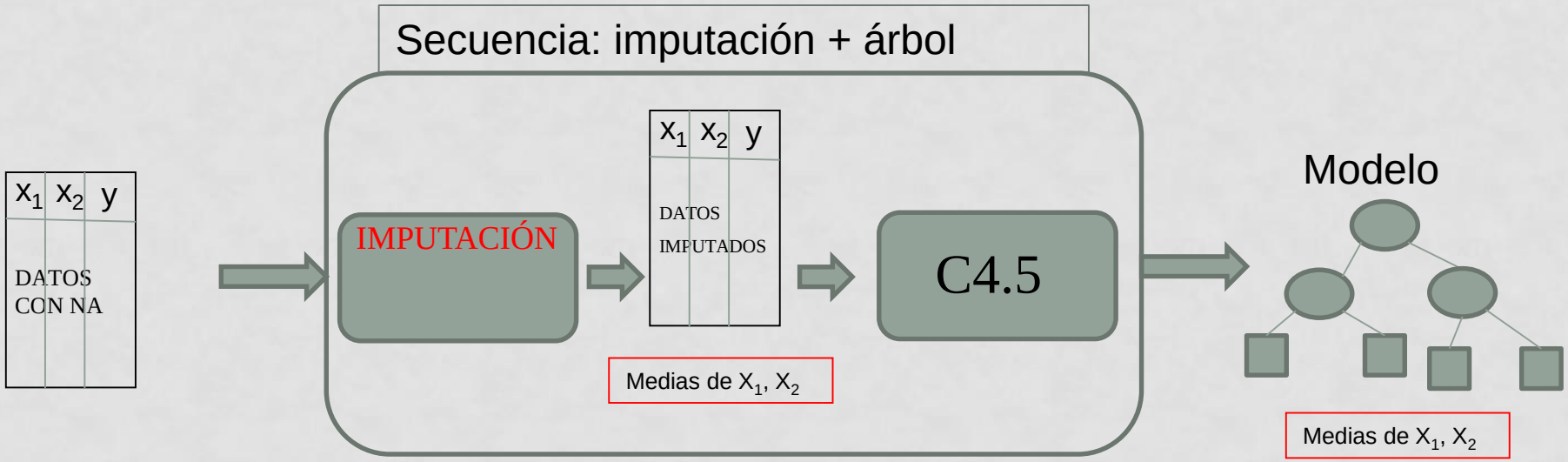


¿DÓNDE COLOCAR LA FASE DE PREPROCESO?

- Distinguimos dos tipos de pre-proceso:
  - No dependiente de los datos (ej: quitar el primer atributo)
  - Dependiente de los datos (prácticamente, todos los demás)
- Existe la tentación de aplicar el pre-proceso a los datos disponibles, y después hacer evaluación y entrenamiento del modelo final.
- Pero esto no es del todo correcto, porque para hacer el preproceso, tenemos que usar datos que después serán usados en test.
- El preproceso dependiente de los datos es realmente parte del proceso de entrenamiento final.

¿DÓNDE COLOCAR LA FASE DE PREPROCESO?

- Para hacer pre-proceso dependiente de los datos, lo correcto es construir una secuencia de métodos (o híbrido o Wrapped en mlR)
- Un ejemplo donde el pre-proceso es imputación:

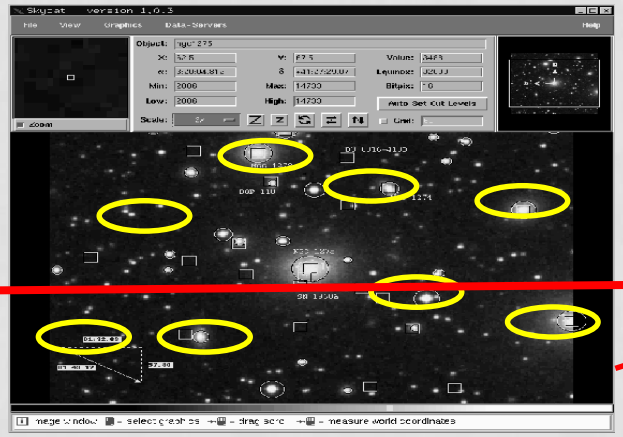


¿DÓNDE COLOCAR LA FASE DE PREPROCESO?

- Y si tenemos que evaluar la secuencia, lo hacemos como con cualquier otro método.
- Si tenemos que imputar los atributos del conjunto de test, usaremos las medias calculadas con el conjunto de entrenamiento. Así no hay "data leakage" de test a entrenamiento.

Datos Disponibles

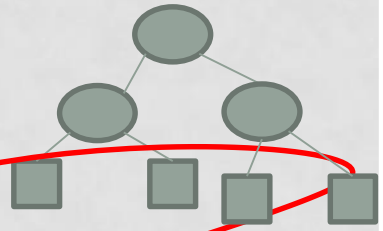
Medias de X_1, X_2



Entrenamiento

Test

Secuencia (imputación+C4.5)

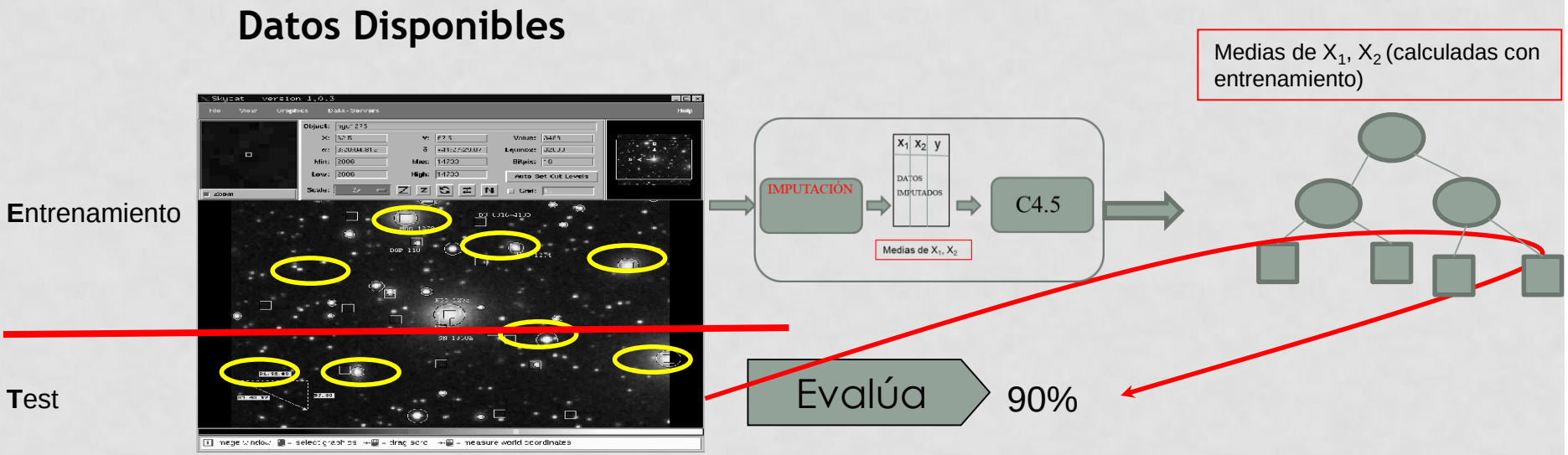


Evalúa

90%

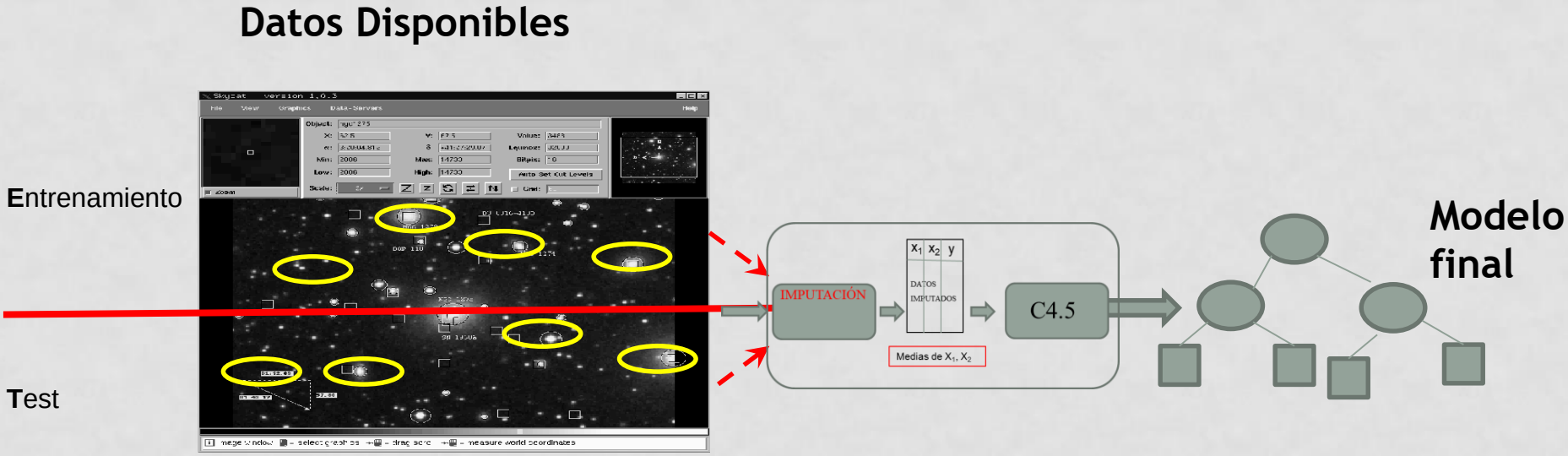
¿DÓNDE COLOCAR LA FASE DE PREPROCESO?

- Y si tenemos que evaluar la secuencia, lo hacemos como con cualquier otro método.
- Si tenemos que imputar los atributos del conjunto de test, usaremos las medias calculadas con el conjunto de entrenamiento. Así no hay "data leakage" de test a entrenamiento.



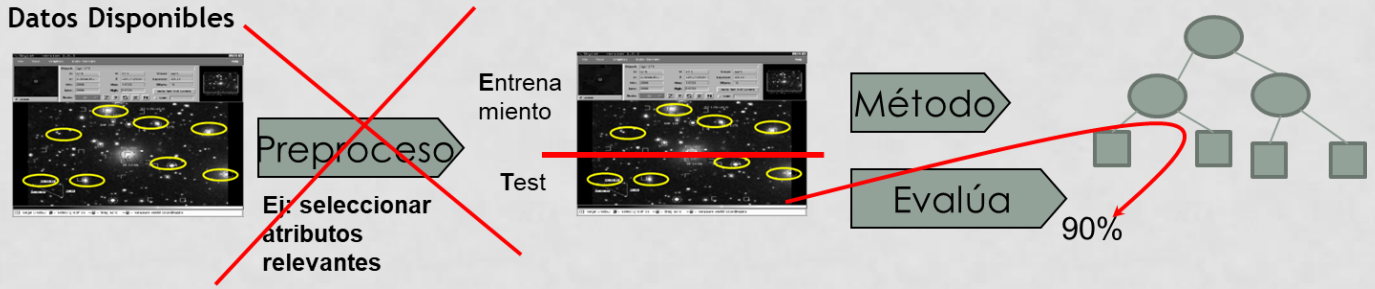
¿DÓNDE COLOCAR LA FASE DE PREPROCESO?

Finalmente, construimos el modelo final con todos los datos, usando la secuencia, como de costumbre.



NOTA

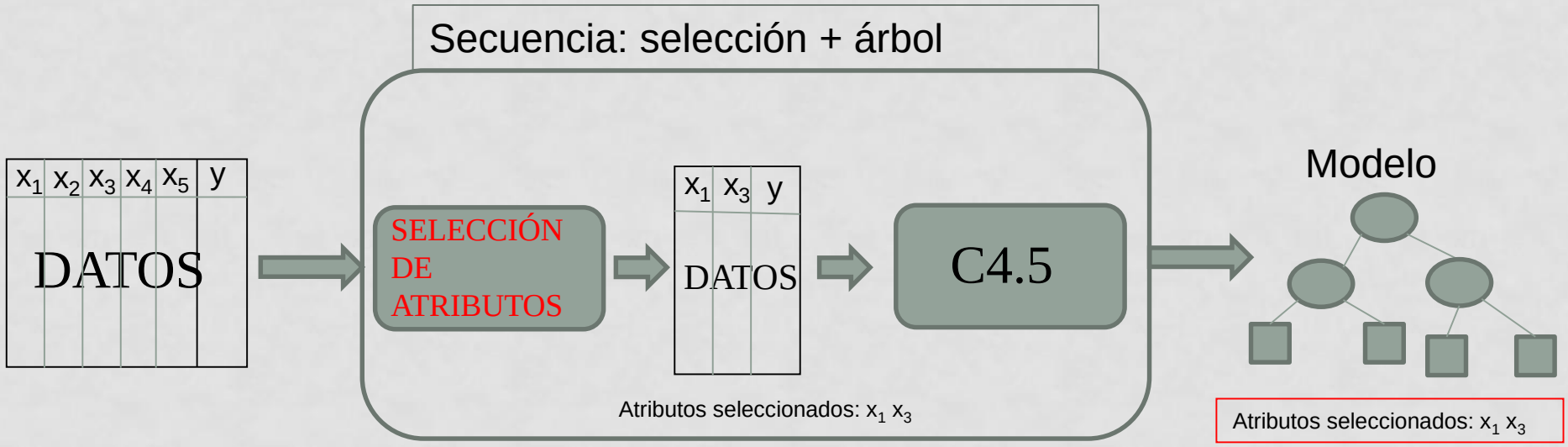
- Así y todo, hacer imputación usando el conjunto de datos disponibles, o sea, siguiendo este esquema, no es excesivamente grave.



- Como norma general, hacer preproceso de los atributos de entrada con todos los datos disponibles no es demasiado grave.
- Incluso, en muchas competiciones de minería de datos, los atributos de entrada del conjunto de test, son conocidos.
- Lo que no se puede hacer nunca es preproceso de la variable de respuesta usando todos los datos disponibles. Se debe hacer sólo con la partición de entrenamiento.

¿DÓNDE COLOCAR LA FASE DE PREPROCESO?

- Un ejemplo donde el pre-proceso es selección de atributos.
- Los atributos seleccionados forman parte del modelo.

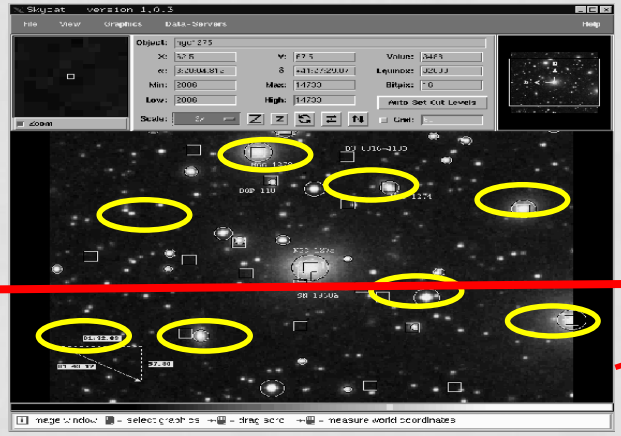


¿DÓNDE COLOCAR LA FASE DE PREPROCESO?

- Y si tenemos que evaluar la secuencia, lo hacemos como con cualquier otro método.
- Del conjunto de test seleccionaremos los mismos atributos que fueron elegidos en entrenamiento.

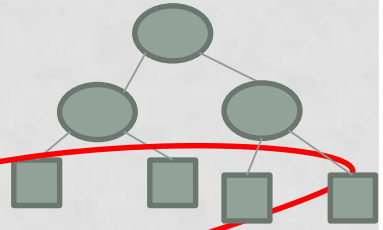
Datos Disponibles

Atributos seleccionados: x_1 x_3



Entrenamiento

Secuencia (selección+C4.5)



Test

Evalúa

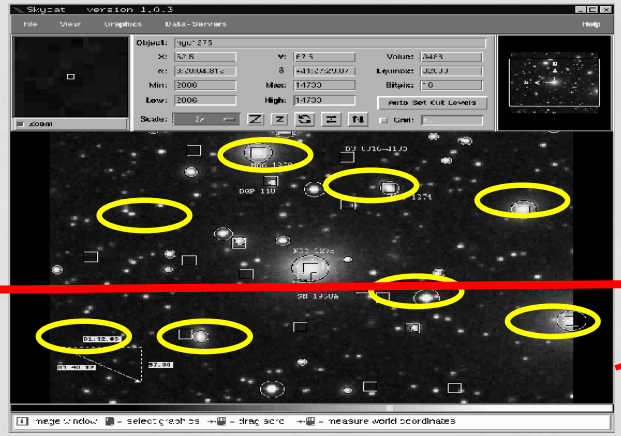
90%

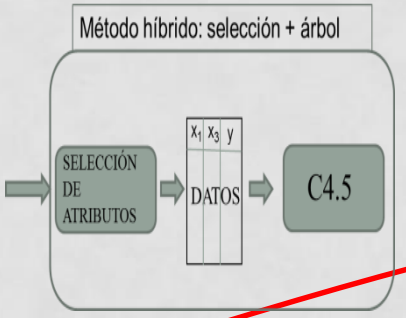
¿DÓNDE COLOCAR LA FASE DE PREPROCESO?

Datos Disponibles

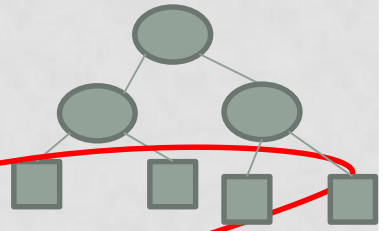
Entrenamiento

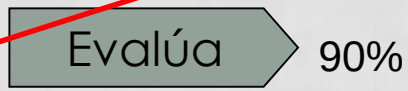
Test





Atributos seleccionados: x_1 x_3

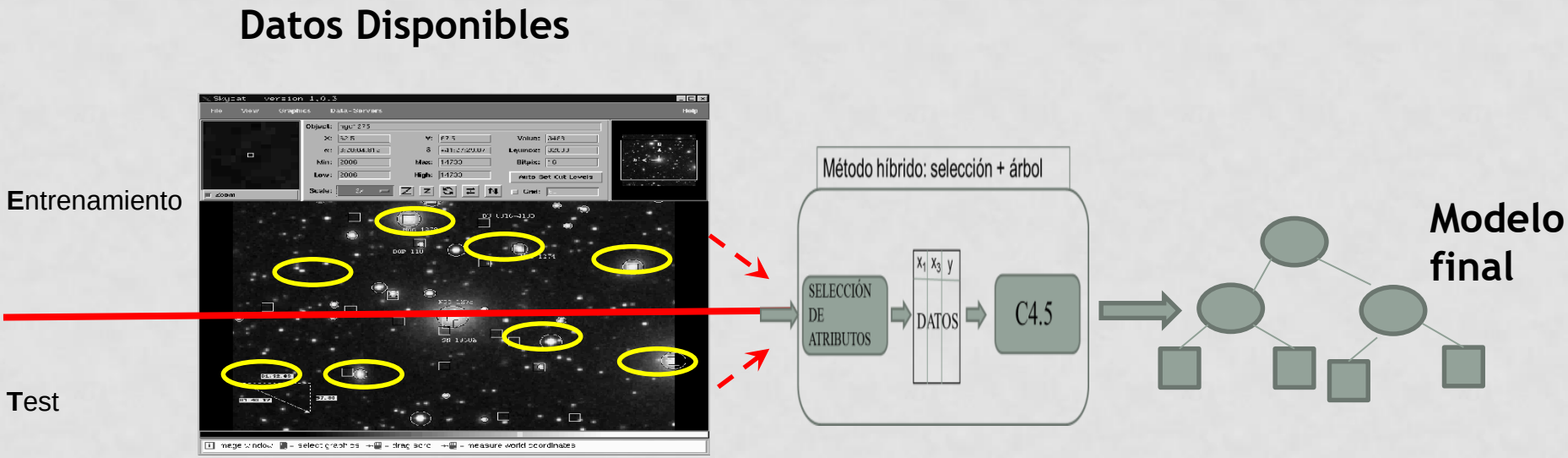




90%

¿DÓNDE COLOCAR LA FASE DE PREPROCESO?

Finalmente, construimos el modelo final con todos los datos, usando la secuencia de métodos, como de costumbre.



Ventajas de la selección de atributos (feature selection)

- Aliviar el efecto de la maldición de la dimensionalidad
- Mejorar la capacidad de generalización (eliminando atributos irrelevantes y redundantes)
- Acelerar el proceso de aprendizaje (aunque se añade el coste de la selección en el preproceso)
- Mejorar la interpretabilidad del modelo (al reducir la complejidad del modelo)

Métodos de selección de atributos

Métodos de selección de atributos

- **Filter**: evaluación de la relevancia de los atributos, uno a uno, usando métodos matemáticos sencillos. Ordenación de los mismos y eliminación de los menos valorados.

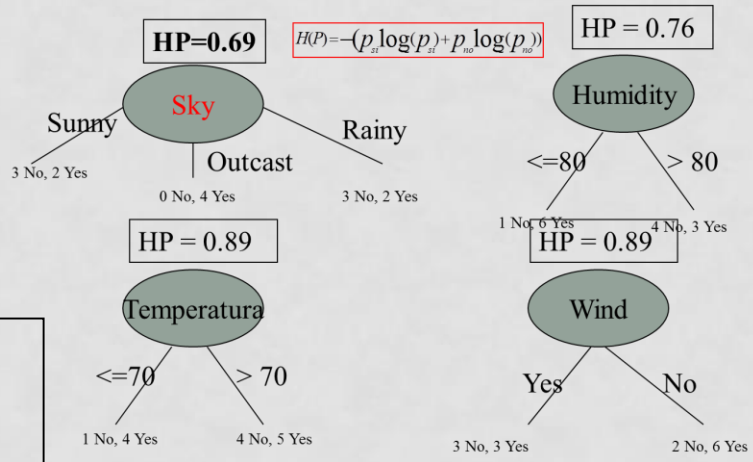
- **Wrapper**: evaluación de **subconjuntos** de atributos. Se evalúan construyendo un modelo, usando sólo los atributos del subconjunto.

Filter

- Dado unos atributos A_1, A_2, ..., A_n, se evalúa cada A_i 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
- Criterios para evaluar a los atributos:
  - Entropía (o information gain), como en los árboles de decisión
  - Información mutua / Mutual information
  - Chi-square
  - F-score, …
- Una vez evaluados y ordenados, se quitan los peores

Filter con entropía (o filter con ganancia de información)

- Ya conocemos esta medida para elegir el mejor atributo para un nodo en árboles de decisión.
- La podemos usar también para ordenar atributos en general
- La ganancia de información es básicamente el inverso de la entropía (técnicamente, es la diferencia entre la entropía inicial de los datos, menos la entropía tras usar el atributo. Cuando más grande sea la diferencia, mejor)



Sirve para atributos discretos y contínuos; y clases discretas

Filter con información mútua

Sirve para atributos y clases discretas (categóricas)

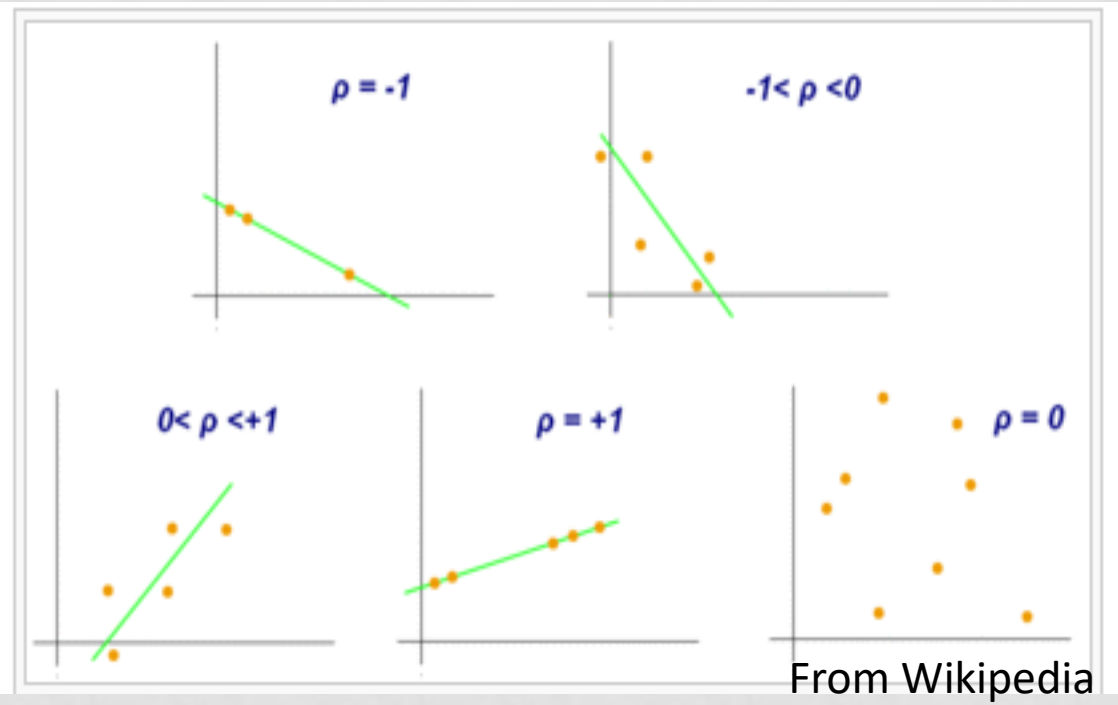
Recordar que si x e y son independientes, p(X,Y)=p(X)*p(Y)

$$I(X,Y) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} P(X = x_i, Y = y_j) \log \frac{P(X = x_i, Y = y_j)}{P(X = x_i)\, P(Y = y_j)}$$

- x son los valores del atributo X, y son los valores de la clase Y
- I(X,Y) = 0 si X e Y son independientes (log(1) = 0)
- I(X,Y) >= 0 (cuanto mas dependientes son las variables, mas positiva es la información mútua)

CORRELACIÓN LINEAL

- Coeficiente de Pearson -1 <= r <= +1
- Covariance(X,Y) = $E((X-\mu_X)(Y-\mu_Y))$
- Correlation(X,Y)= r = $\frac{Cov(X,Y)}{\sigma_X \sigma_Y}$



From Wikipedia

Sirve para atributos y clases contínuas

CHI-CUADRADO

- Supongamos una clase Y, y un atributo X (con dos valores discretos: a y b). $Obs_{x,y}$ es el número de datos observados para los que la clase es y, y el atributo X vale x.
- El número de datos total es
  - $n = Obs_{a,p} + Obs_{b,p} + Obs_{a,n} + Obs_{b,n}$

Atributo X

Clase Y: Datos	X= a	X= b
Y= positivo	$Obs_{a,p}$	$Obs_{b,p}$
Y= negativo	$Obs_{a,n}$	$Obs_{b,p}$

Sirve para atributos y clases discretas (categóricas)

CHI-CUADRADO

- Si la clase Y y el atributo X son independientes, se cumple que: prob(X,Y) = prob(X) * prob(Y)
  - Prob(X=a) = ($Obs_{a,p}$ + $Obs_{a,n}$) / n
  - Prob(X=b) = ($Obs_{b,p}$ + $Obs_{b,n}$) / n
  - Prob(Y=p) = ($Obs_{a,p}$ + $Obs_{b,p}$) / n
  - Prob(Y=n) = ($Obs_{a,n}$ + $Obs_{b,n}$) / n
- Bajo la suposición de independencia,

  Prob(X=x, Y=y) = Prob(X=x)*Prob(Y=y)
- Ej: P(X=a, Y=p) = P(X=a)*P(Y=p)
- El número de datos esperados con X=x e Y=y, supuesto que el atributo y la clase son independientes, será:

  $Esp_{x,y}$ = n*P(X=x,Y=y) = n*P(X=x)*P(Y=y)

Datos	X= a	X= b
Y= positivo	$Obs_{a,p}$ $Esp_{a,p}$	$Obs_{b,p}$ $Esp_{b,p}$
Y= negativo	$Obs_{a,n}$ $Esp_{a,p}$	$Obs_{b,p}$ $Esp_{b,p}$

CHI-CUADRADO

- Bajo la suposición de independencia, X^2 sigue la distribución chi-cuadrado con un grado de libertad

$$X^2 = \sum_{y=p,n} \sum_{x=a,b} \frac{(\mathrm{Obs}_{x,y} - \mathrm{Esp}_{x,y})^2}{\mathrm{Esp}_{x,y}}$$

Datos	X= a	X= b
Y= positivo	$\mathrm{Obs}_{a,p}$ $\mathrm{Esp}_{a,p}$	$\mathrm{Obs}_{b,p}$ $\mathrm{Esp}_{b,p}$
Y= negativo	$\mathrm{Obs}_{a,n}$ $\mathrm{Esp}_{a,p}$	$\mathrm{Obs}_{b,p}$ $\mathrm{Esp}_{b,p}$

- Podemos ordenar los atributos por X^2 (cuanto más grande, más improbable que X e Y sean independientes)

- También podemos rechazar la hipótesis de independencia con un test de hipótesis.
- Por ejemplo, si $X^2 > 6.61$, podemos rechazar independencia con probabilidad 0.99

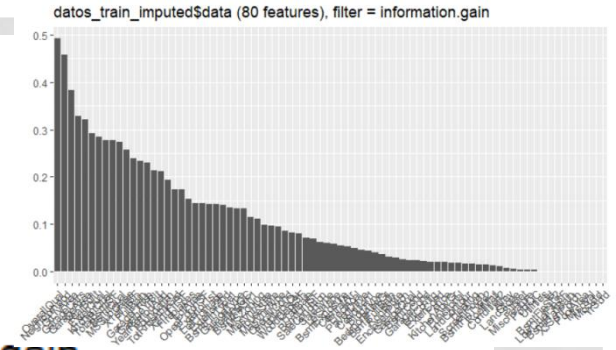
F-SCORE (FISHER SCORE)

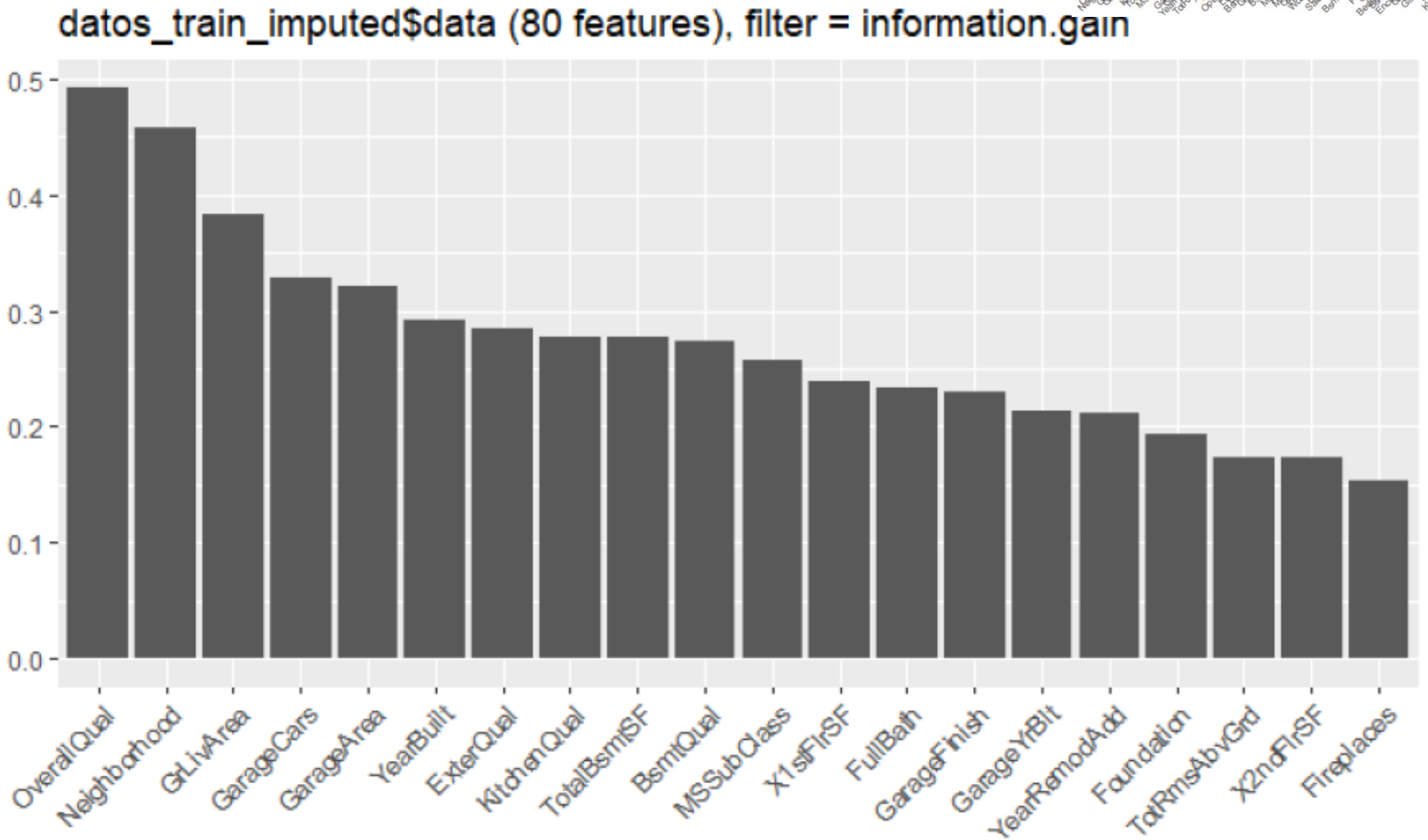
- $FS = (u_+ - u_-)/(sigma_+^2+sigma_-^2)$

Filter

- Ventajas: es rápido
- Desventajas:
  - No elimina atributos redundantes
  - No detecta atributos que funcionan bien de manera conjunta, pero mal de manera separada. De hecho, descartaría esos atributos.

Ejemplo de filter





Idea importante

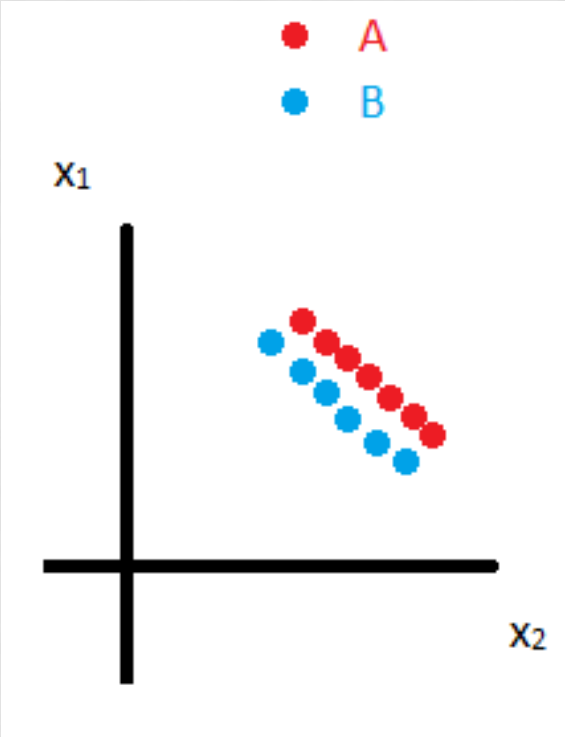
- En ocasiones, dos atributos por separado no dan información, pero juntos sí.
- Ejemplo:
  - Sea un problema de clasificación de textos en dos clases "informática" y "filosofía"
  - Sean los atributos booleanos "inteligencia" y "artificial", que son ciertos si esas palabras aparecen en el texto y falsos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

Idea importante

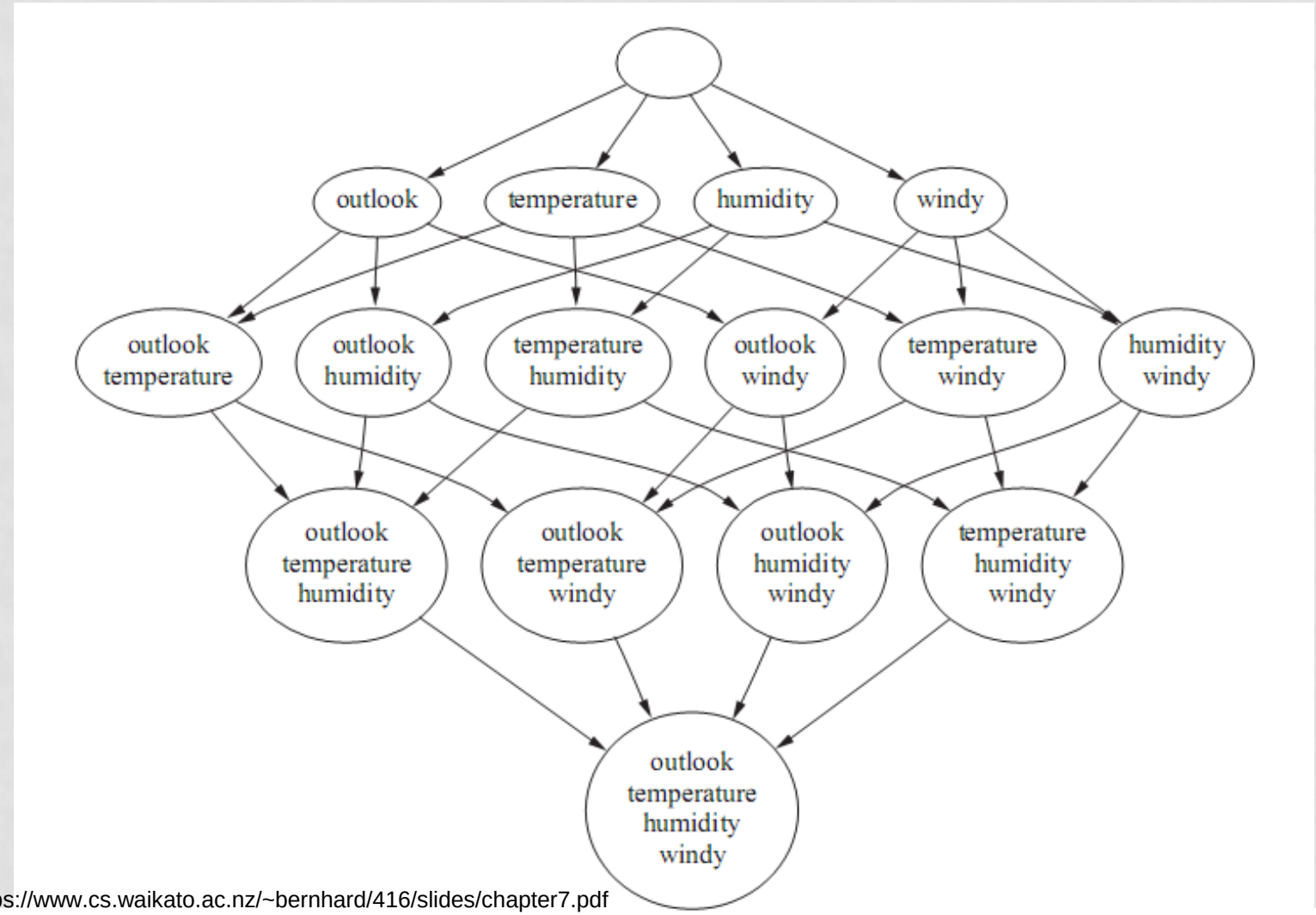
- Otro ejemplo: x1 y x2 por separado no son capaces de separar la clase A de la de B, pero juntas si (un clasificador lineal puede hacerlo)



Búsqueda exhaustiva

- El método mas preciso sería 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

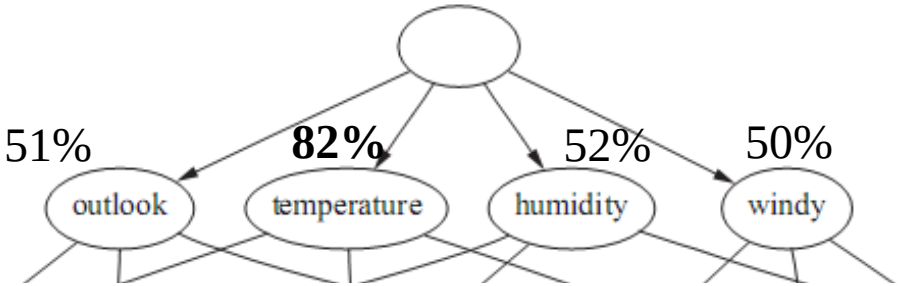
Búsqueda en el espacio de subconjuntos de atributos



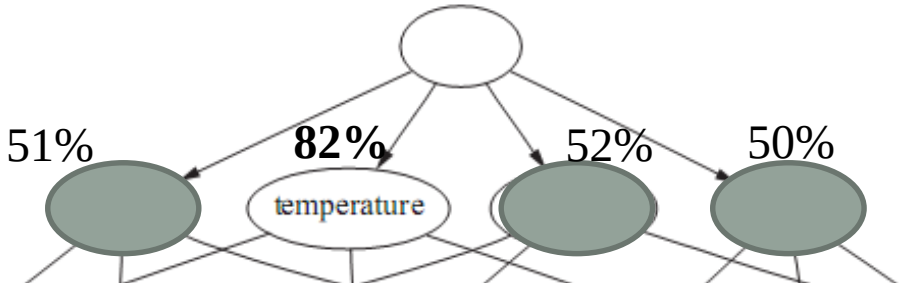
Source: https://www.cs.waikato.ac.nz/~bernhard/416/slides/chapter7.pdf

Búsqueda en el espacio de subconjuntos de atributos

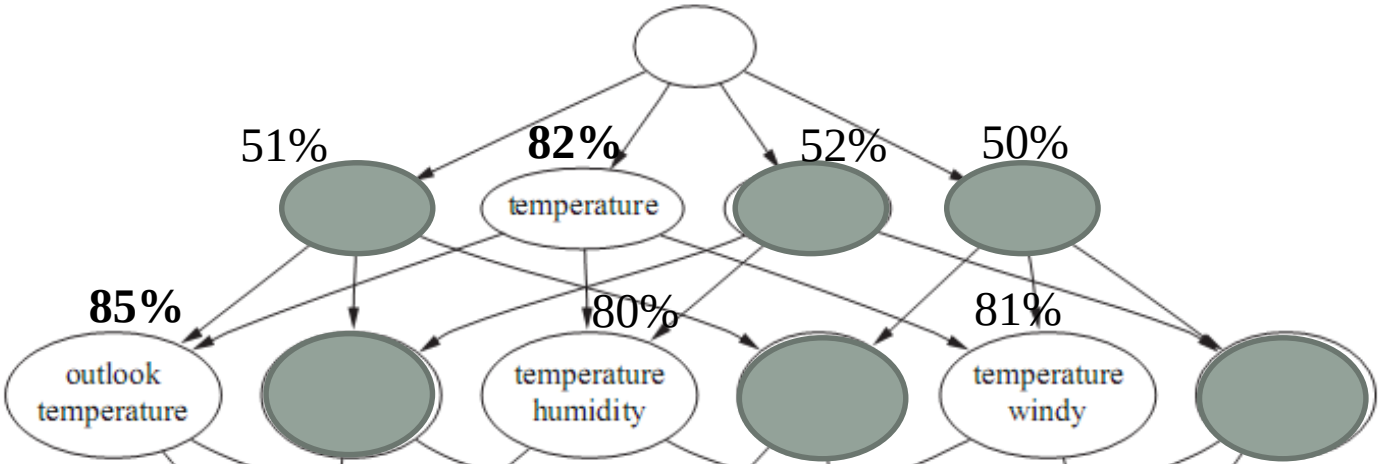
Búsqueda en el espacio de subconjuntos de atributos



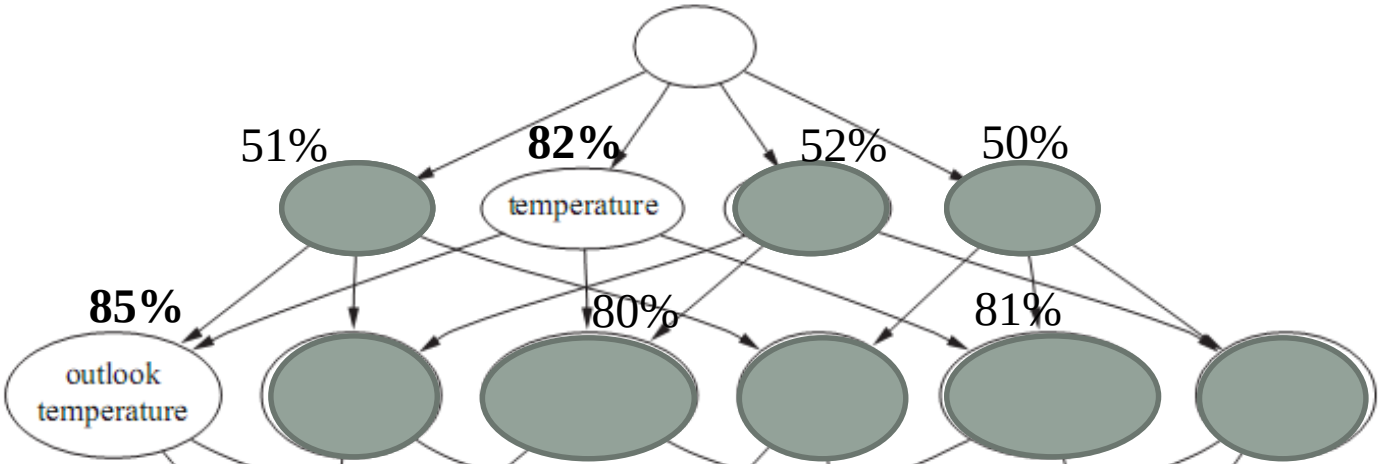
Búsqueda en el espacio de subconjuntos de atributos



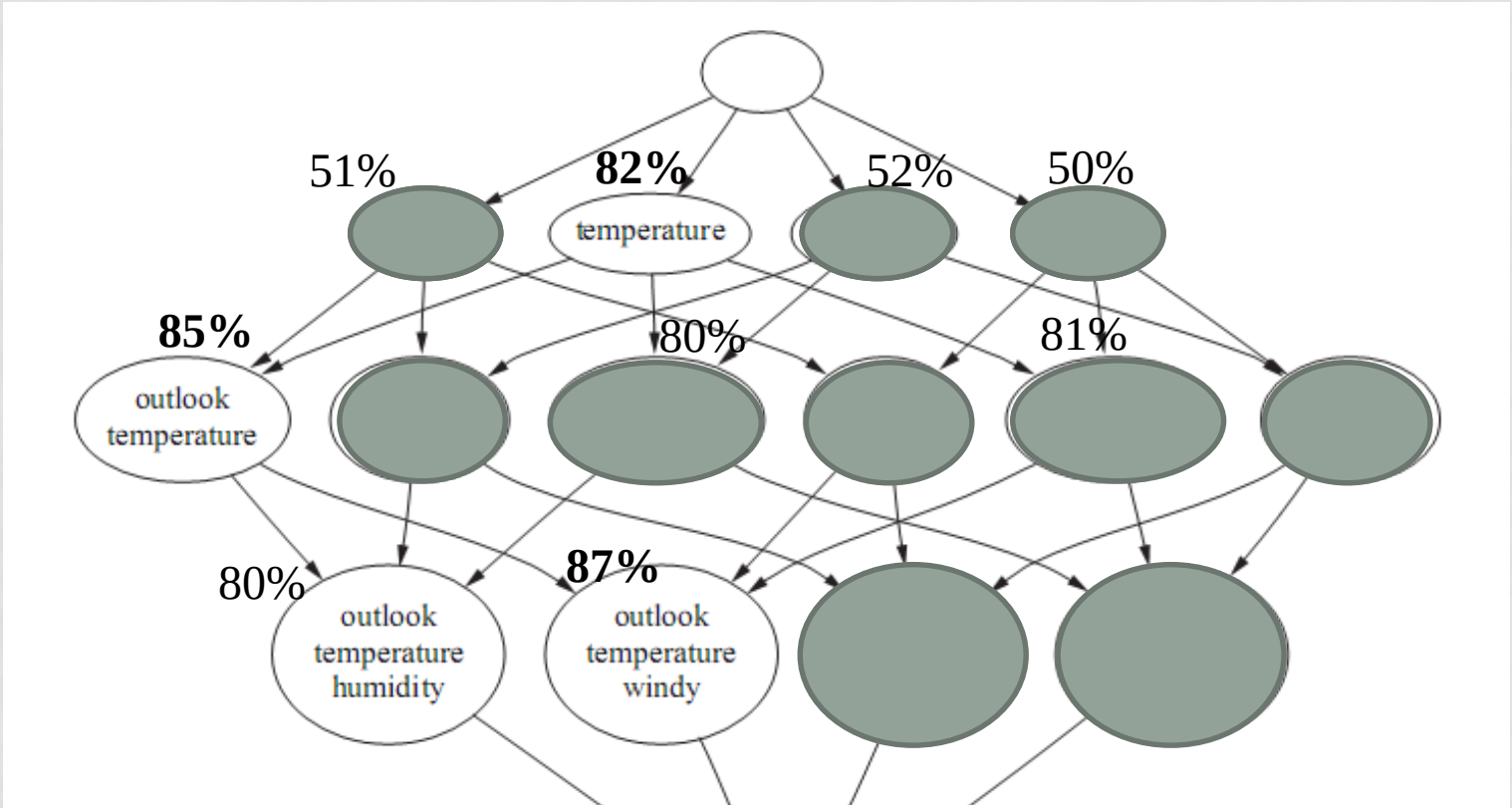
Búsqueda en el espacio de subconjuntos de atributos



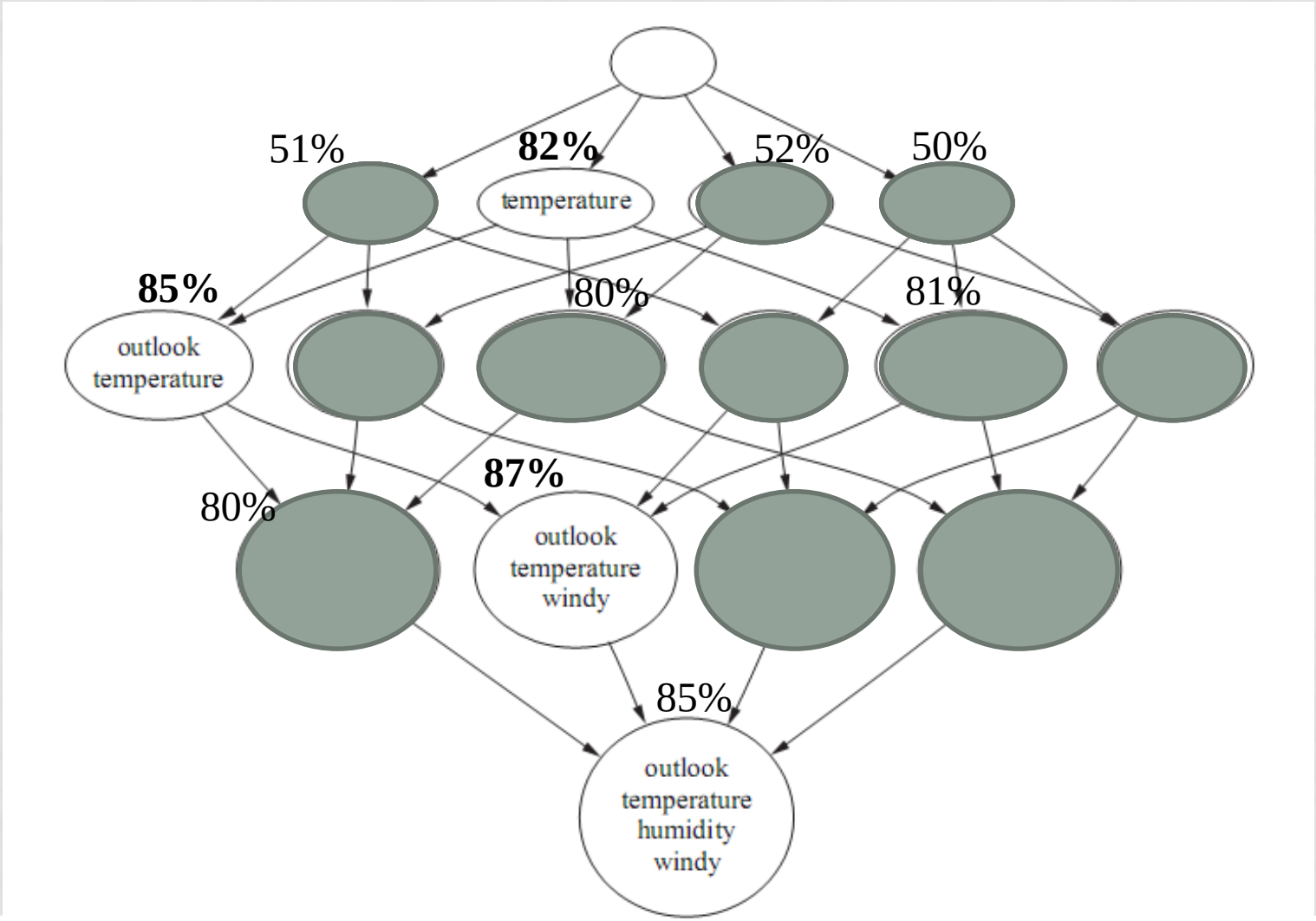
Búsqueda en el espacio de subconjuntos de atributos



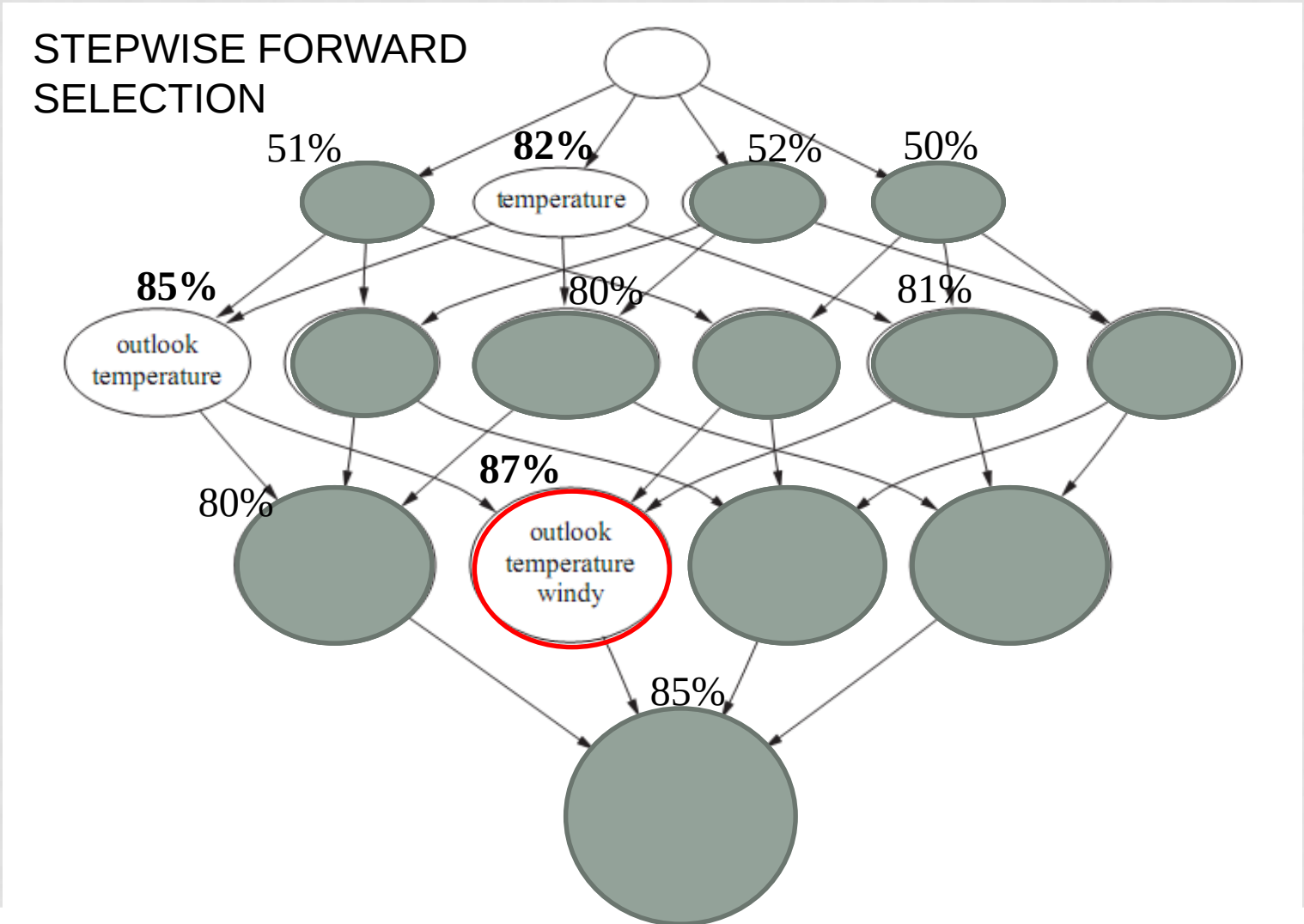
Búsqueda en el espacio de subconjuntos de atributos



Búsqueda en el espacio de subconjuntos de atributos



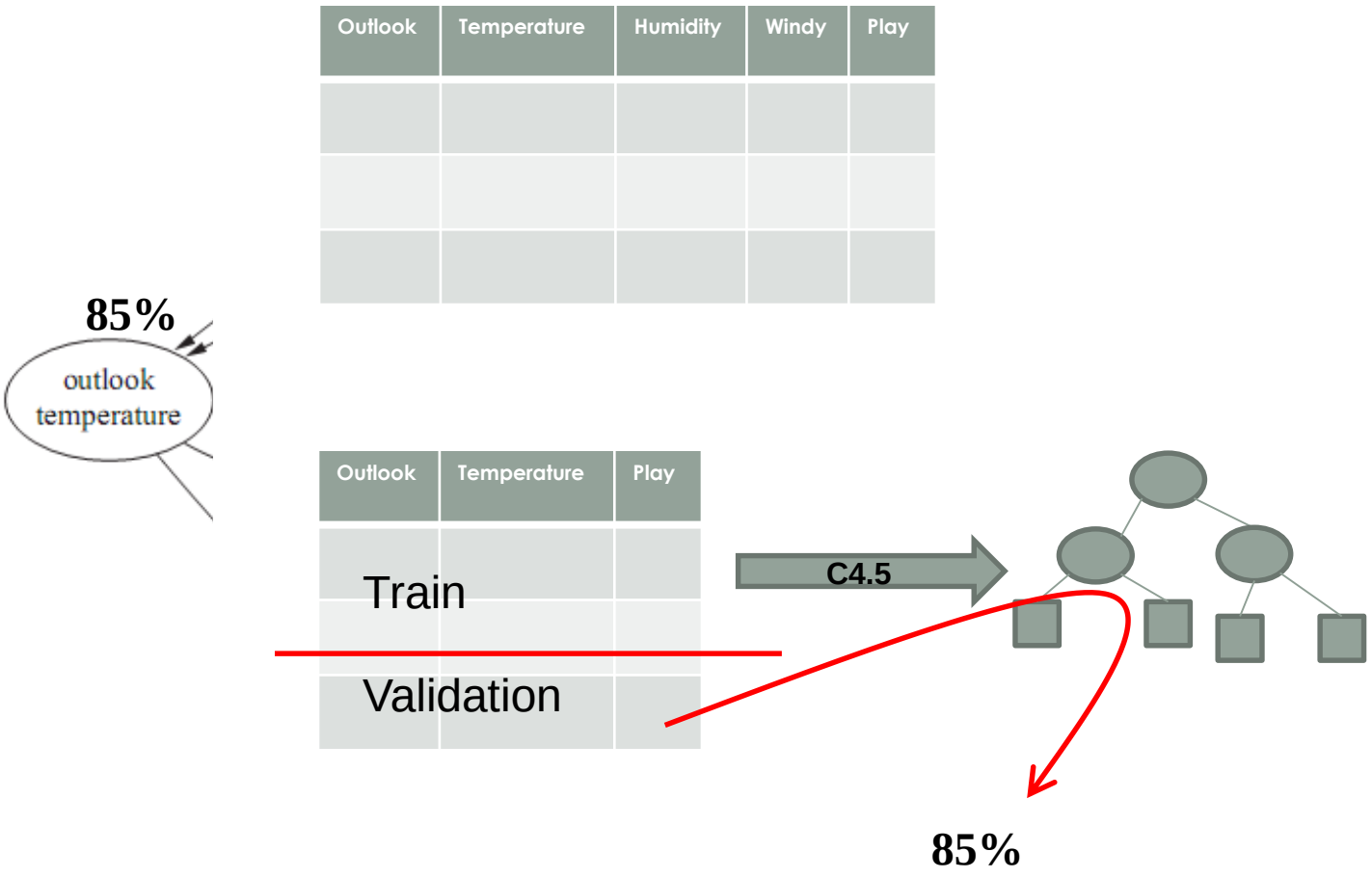
Búsqueda en el espacio de subconjuntos de atributos



Selección de subconjuntos (subset selection)

- Estos métodos recorren un espacio de subconjuntos de atributos,
- No se recorre el espacio entero (eso sería búsqueda exhaustiva), sino sólo aquellos subconjuntos más prometedores
  - En el caso anterior hicimos búsqueda hacia adelante (stepwise forward selection)
  - También podríamos hacer búsqueda hacia atrás, partiendo del conjunto completo de atributos, e ir quitando, uno a uno (stepwise backward selection)
  - Hay otros métodos de búsqueda: algoritmos genéticos, best first, …
- Se evalúan **subconjuntos** de manera conjunta
- Pero, ¿cómo evaluar subconjuntos de atributos?: Wrapper

Evaluación de subconjuntos Wrapper



Evaluación de subconjuntos Wrapper

- Los métodos *Wrapper* evalúan un subconjunto de atributos ejecutando un algoritmo de entrenamiento de modelos concreto, sobre un conjunto de datos, únicamente con las columnas que se quiere evaluar.
- Obtienen subconjuntos de atributos adecuados para un algoritmo concreto (ej: C4.5)
- Ventaja: evalúan a los atributos de los subconjuntos de manera realmente conjunta
- Desventaja: son muy lentos (hay que ejecutar un algoritmo de aprendizaje muchas veces)

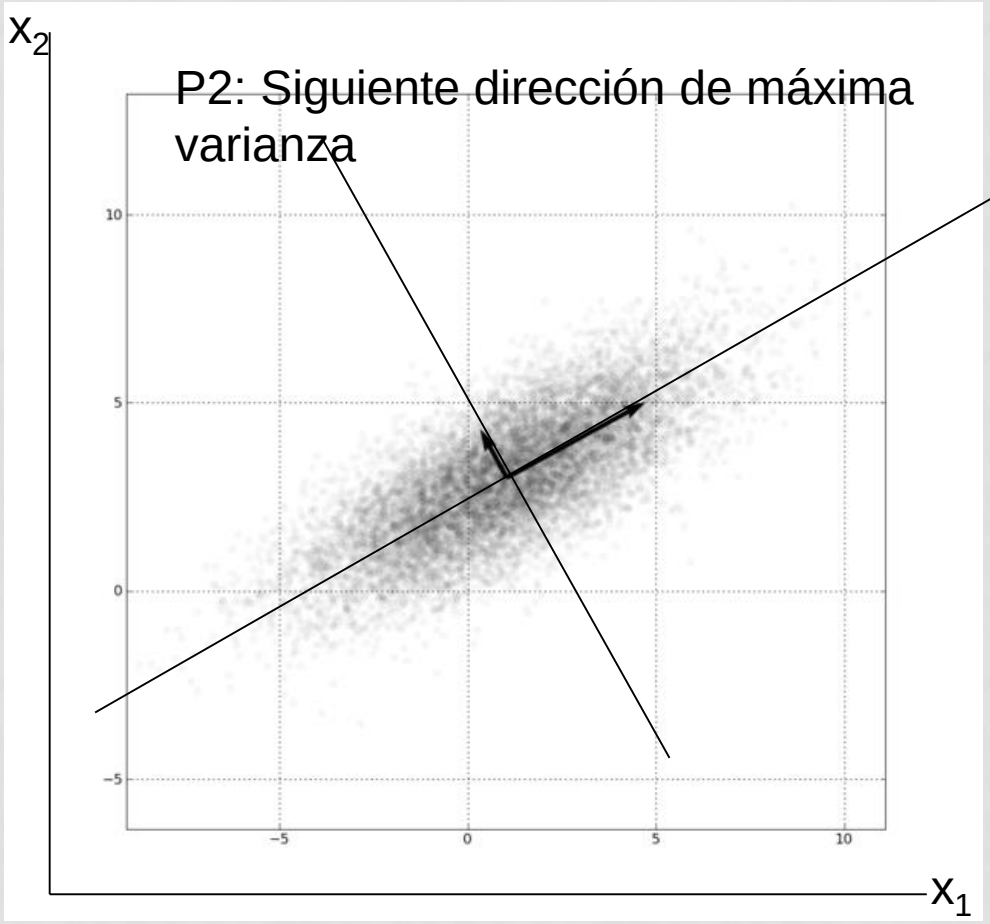
EXTRACCIÓN DE ATRIBUTOS (FEATURE EXTRACTION)

- Principal Component Analysis (PCA)

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

PCA

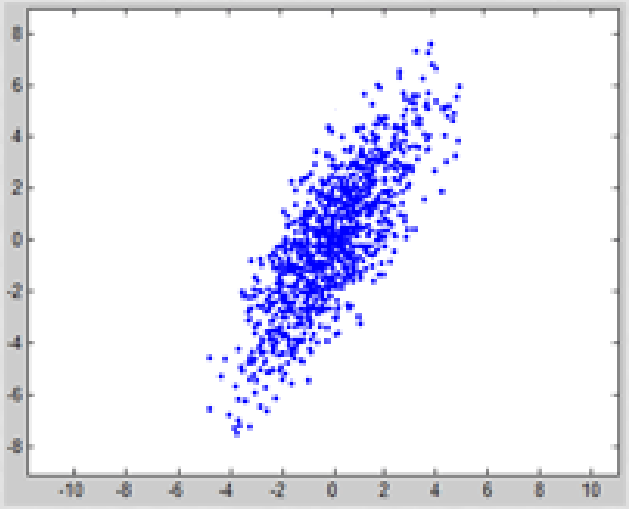


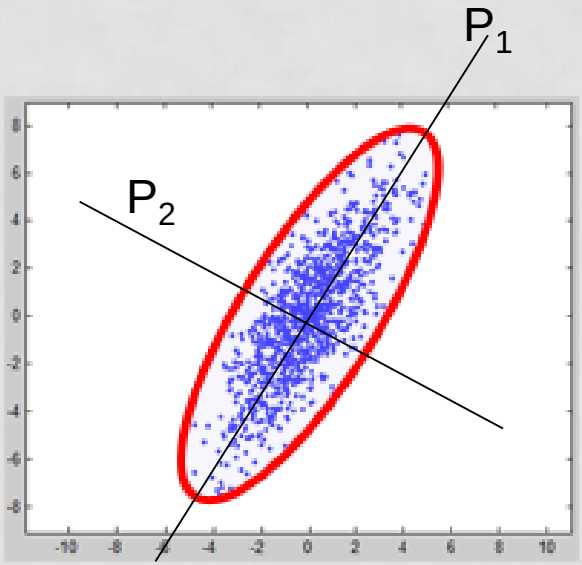
Crea dos atributos nuevos: P1 y P2

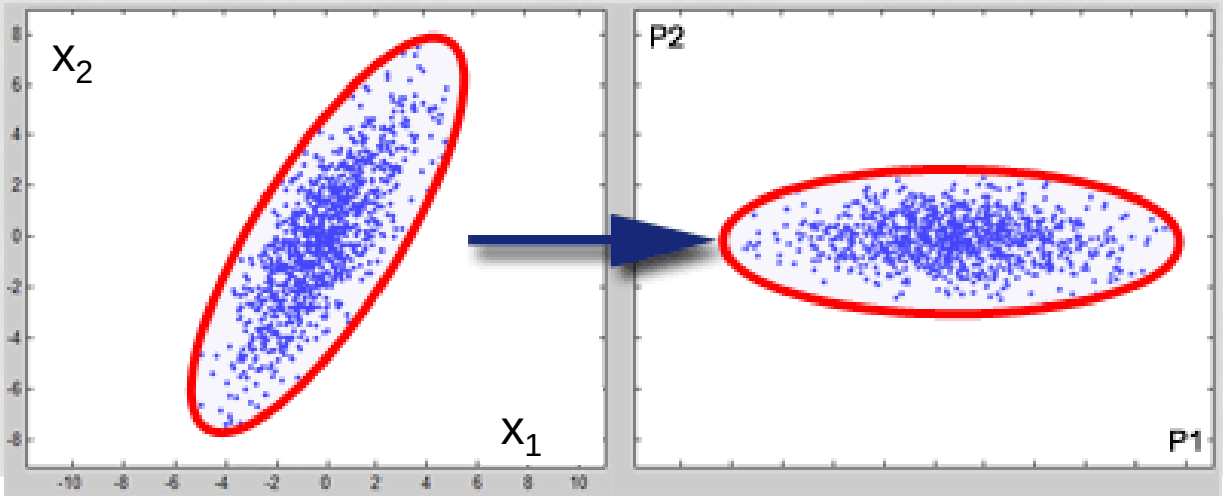
Source: https://support.numxl.com/hc/en-us/articles/216532603-Principal-Component-Analysis-PCA-101?mobile_site=true

Transformación realizada por PCA

Source: https://support.numxl.com/hc/en-us/articles/216532603-Principal-Component-Analysis-PCA-101?mobile_site=true







- Se trata de transformaciones lineales
- Elimina redundancia (correlación) de los datos

$P_1 = k_{11}*x_1+k_{12}*x_2$

$P_2 = k_{21}*x_1+k_{22}*x_2$

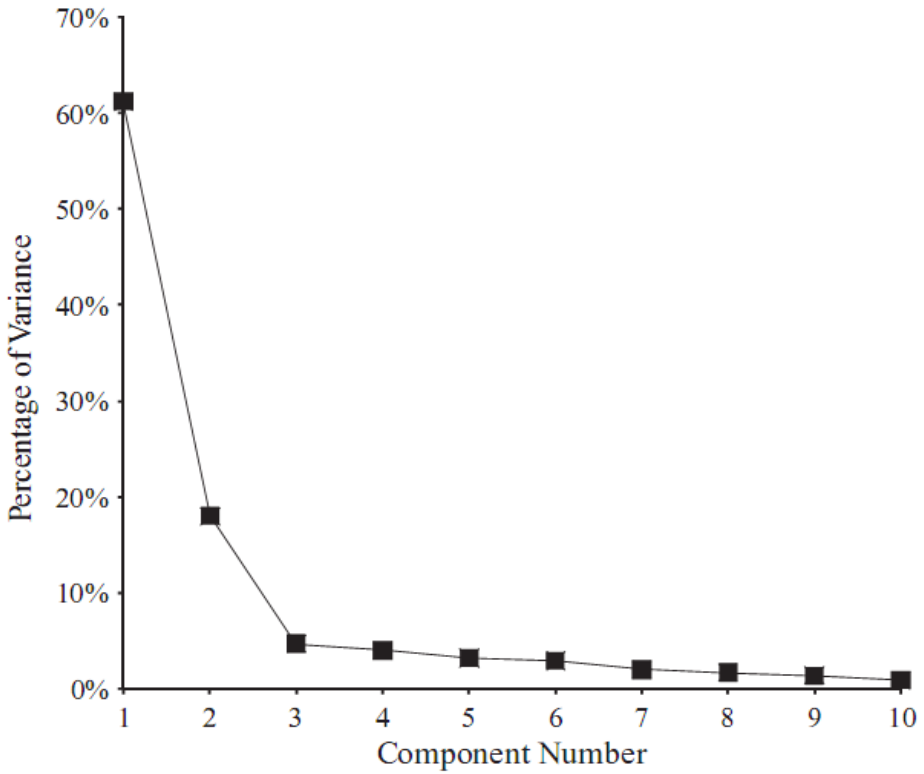
$\mathbf{P} = \mathbf{X}^t * k$

k = matriz de *loadings*

PCA: aparte de transformación, también selección

Axis	Variance	Cumulative
1	61.2%	61.2%
2	18.0%	79.2%
3	4.7%	83.9%
4	4.0%	87.9%
5	3.2%	91.1%
6	2.9%	94.0%
7	2.0%	96.0%
8	1.7%	97.7%
9	1.4%	99.1%
10	0.9%	100.0%

(a)



(b)

• Normalmente se suele coger tantos atributos como 95% de la varianza (7, en este caso)

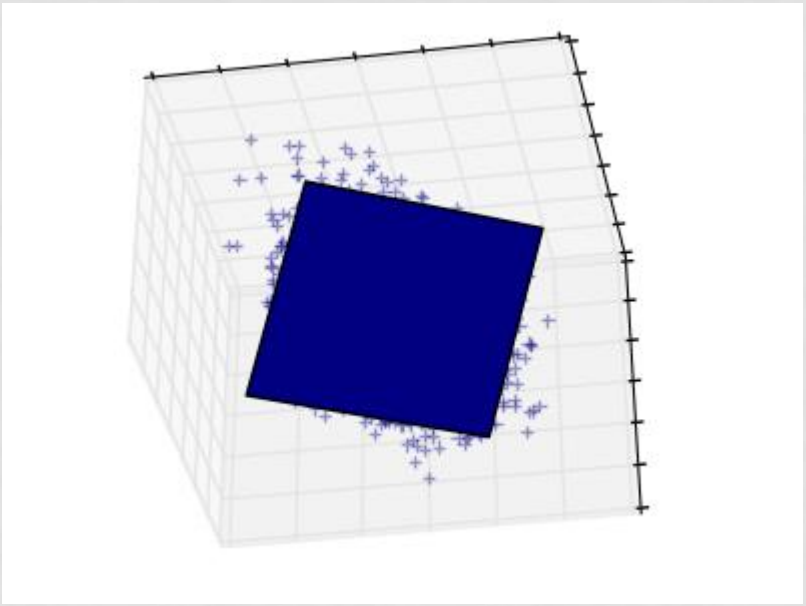
•Si sólo unos pocos atributos explican la mayor parte de la varianza, el resto sobran (ej: datos en forma de elipse en 20 dimensiones)

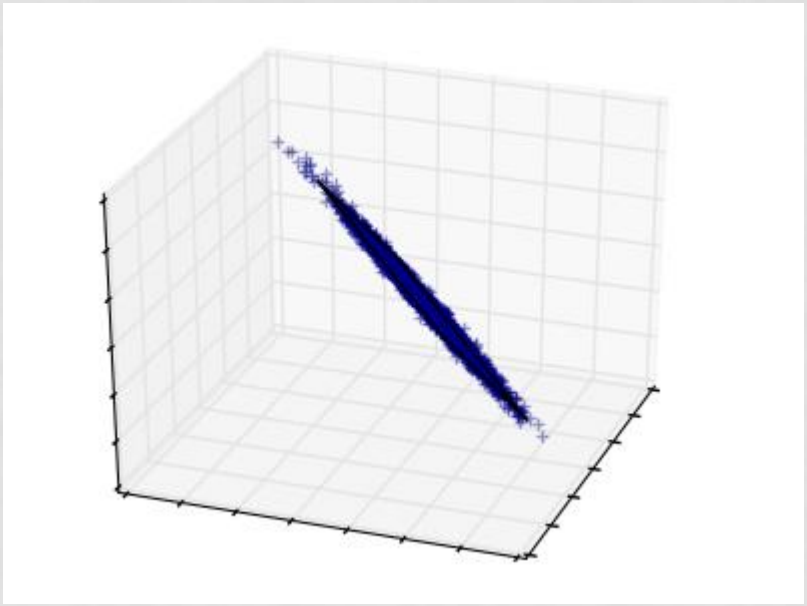
Ventajas de PCA

- Utilidad 1: puede determinar la dimensionalidad "real" de los datos
  - Ej: imaginar datos en forma de elipse de 2 dimensiones embebida en 20 dimensiones). PCA identificará fácilmente que con sólo 2 dimensiones se explica toda la varianza
- Utilidad 2: Decorrelación de los atributos

PCA Y LA DIMENSIONALIDAD REAL DE LOS DATOS

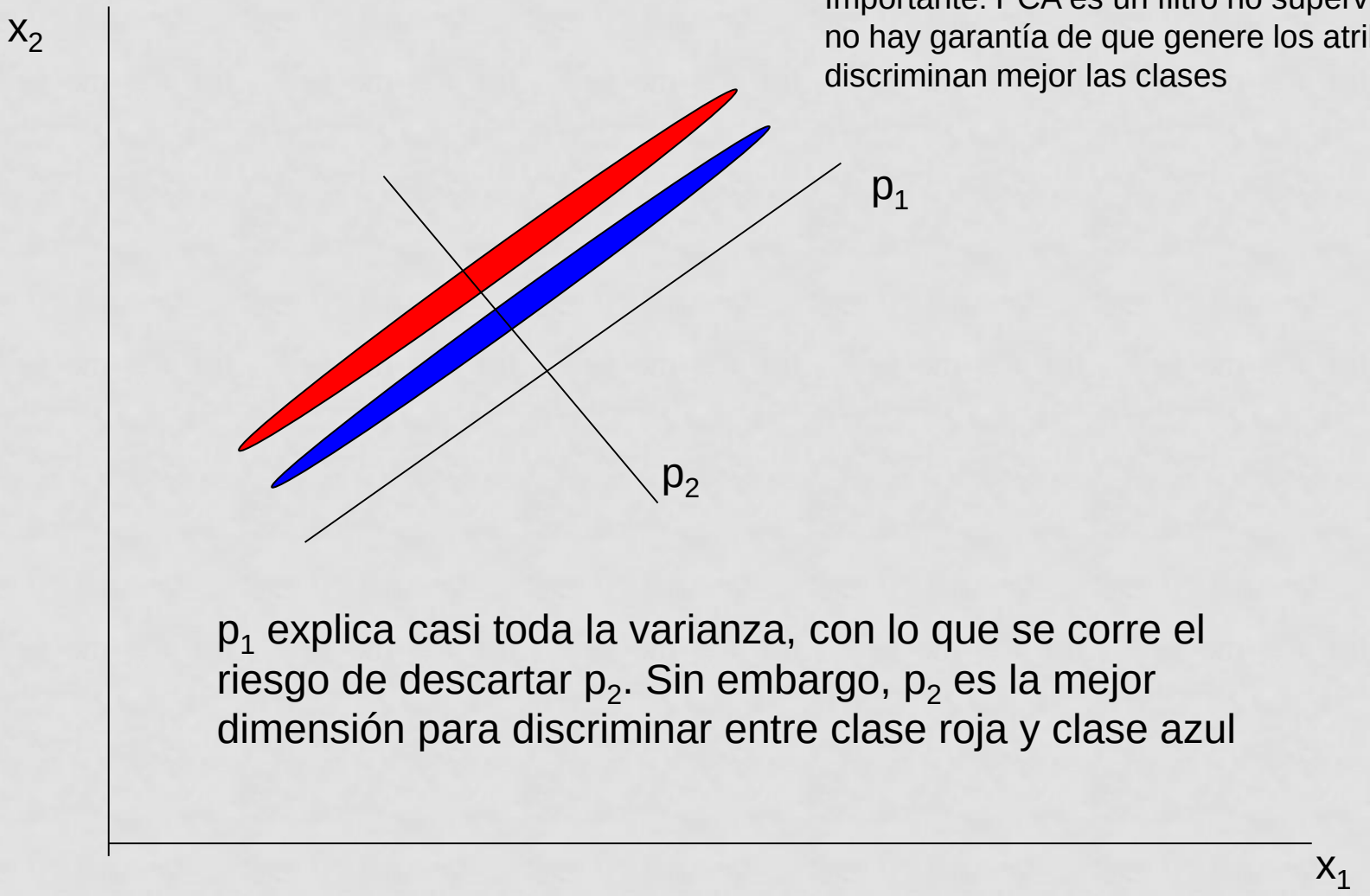
- Ejemplo de un conjunto de datos de dos dimensiones, insertado ("embedded") en un espacio de tres dimensiones.





Cuidado, PCA es no supervisado

Importante: PCA es un filtro no supervisado, con lo que no hay garantía de que genere los atributos que discriminan mejor las clases



p_1 explica casi toda la varianza, con lo que se corre el riesgo de descartar p_2. Sin embargo, p_2 es la mejor dimensión para discriminar entre clase roja y clase azul