APRENDIZAJE SUPERVISADO

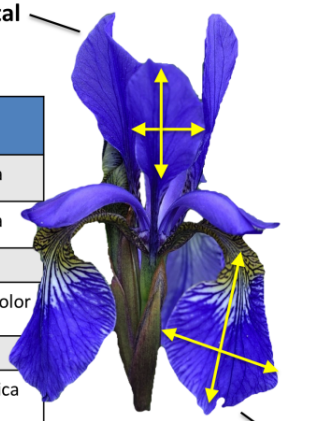
Aprendizaje supervisado

- En el aprendizaje supervisado seguimos contando con nuestra matriz de datos de n individuos, pero distinguiremos entre dos tipos de variables
  - Un conjunto de m predictores o variables de entrada, x_i con $i = 1, \dots, m$
  - Variable de respuesta o salida, normalmente se denota como y
- La variable de salida o respuesta, determina el tipo de problema que estamos tratando
  - Si se trata de una variable numérica, es un **problema de regresión**
    - Predecir el precio de una casa en función de sus características
  - Si se trata de una variable categórica, es un **problema de clasificación**
    - Predecir el tipo de flor iris dadas sus características (versicolor, setosa, virginica)
- El objetivo es ajustar un modelo que relacione las variables de entrada con la de salida, de forma que
  - Seamos capaces de predecir la respuesta lo mejor posible
  - Entendamos mejor la relación entre las variables de entrada y la de salida
    - Este último punto, no lo proporcionan algunas técnicas o algoritmos que funcionan como una “caja negra”

Aprendizaje supervisado

Samples
(instances, observations)

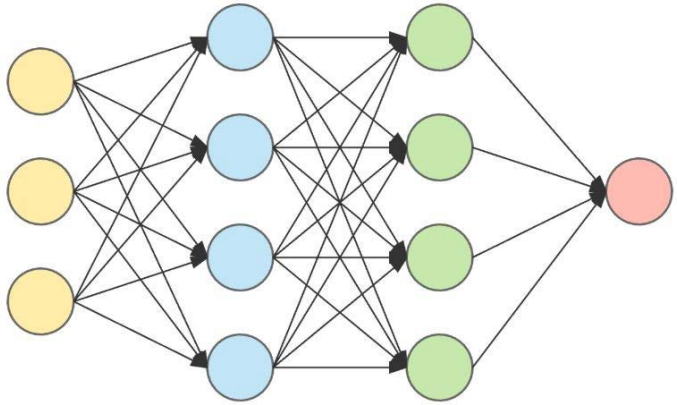
	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica



Features
(attributes, measurements, dimensions)

Class labels
(targets)

Entrenar modelo



Predecir

Sepal length	Sepal width	Petal length	Petal width	Class label
4.9	3.6	1.4	0.3	?
6.1	2.8	1.3	0.9	?
...	...	...	...	...

Midiendo el error en problemas de regresión

- Debemos comparar los resultados pronosticados $\hat{y}_i$ por nuestro modelo con los resultados observados en el conjunto de datos y_i con $i = 1, \dots, n$
- En problemas de predicción se suele utilizar el Error Cuadrático Medio o su raíz cuadrada (*Mean Square Error* o *Root Mean Square Error* en inglés)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Como medida de error a optimizar/reducir, el RMSE es equivalente al MSE, pero el RMSE es más inteligible ya que está en las mismas unidades de medida que la variable de salida

  - Los errores están al cuadrado, luego los errores grandes pesarán mucho!
  - Al buscar un modelo que minimice el MSE o el RMSE los parámetros de dicho modelo se ajustarán para evitar cometer errores “grandes”
- Otra alternativa sería el Error Absoluto Medio, que al tomar valores absolutos da igual peso a errores grandes y pequeños, y que también está en la misma unidad de medida que la variable de salida

Midiendo el error en problemas de clasificación

- Debemos comparar los resultados pronosticados $\hat{y}_i$ por nuestro modelo con los resultados observados en el conjunto de datos y_i con $i = 1, \dots, n$
  - Según la variable de salida sea numérica o categórica, usaremos diferentes tipos de medida de error
- En problemas de clasificación podemos visualizar el error mediante la **matriz de confusión**, que para el caso de una clase binaria sería

		Clase observada	
		1	0
Clase pronosticada	1	*Verdaderos Positivos*	*Falsos Positivos*
	0	*Falsos Negativos*	*Verdaderos Negativos*

- La **tasa de aciertos o exactitud** (*accuracy* en inglés) es, en principio, una buena medida

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN}$$

Otras medidas de rendimiento para problemas de clasificación

- La tasa de aciertos puede presentar problemas en casos en los que las clases estén desequilibradas
  - Si quiero detectar una enfermedad congénita que afecta al 1% de la población, un clasificador naïve que siempre dé negativo como resultado acertará el 99% de las veces!
- Necesitamos medidas que reflejen otros aspectos de la clasificación (**por clase**)
  - $Tasa\ de\ Verdaderos\ Positivos\ (TVP) = \frac{VP}{VP+FN} = \frac{VP}{P}$
    - Más conocida como **exhaustividad** (*recall*) o **sensibilidad** (*sensitivity*)
  - $Tasa\ de\ Verdaderos\ Negativos\ (TVN) = \frac{VN}{VN+FP} = \frac{VN}{N}$
    - Más conocida como **especificidad** (*specificity*)
  - $Valor\ Predictivo\ Positivo\ (VPP) = \frac{VP}{VP+FP}$
    - Más conocida como **precisión** (*precision*)

		Clase observada	
		1	0
Clase pronosticada	1	*VP*	*FP*
	0	*FN*	*VN*

- Para elegir un modelo entre varios posibles necesitamos guiarnos por un único valor, por ello se suele usar alguna medida que combina otras medidas simples

$$medida\ F1 = \underbrace{2 \cdot \frac{VPP \cdot TVP}{VPP + TVP}} = \frac{2 \cdot VP}{2 \cdot VP + FP + FN}$$

Media armónica de precision y exhaustividad (F1 Score)

La media armónica es más adecuada porque es más sensible a que uno de los dos valores sea muy pequeño

Entendiendo el rendimiento de un clasificador

- Podemos obtener modelos (i.e. clasificadores) de muy distinto comportamiento, las medidas de rendimiento nos dicen cómo funcionan
  - La utilidad del clasificador depende además del problema donde se esté aplicando

- Por ejemplo, consideremos un problema donde la clase es binaria (enfermo, no-enfermo) y tres clasificadores con estas medidas:

1. Precisión(enfermo)=0.99 Exhaustividad(enfermo)=0.50
2. Precisión(enfermo)=0.50 Exhaustividad(enfermo)=1
3. Precisión(enfermo)=0.80 Exhaustividad(enfermo)=0.95

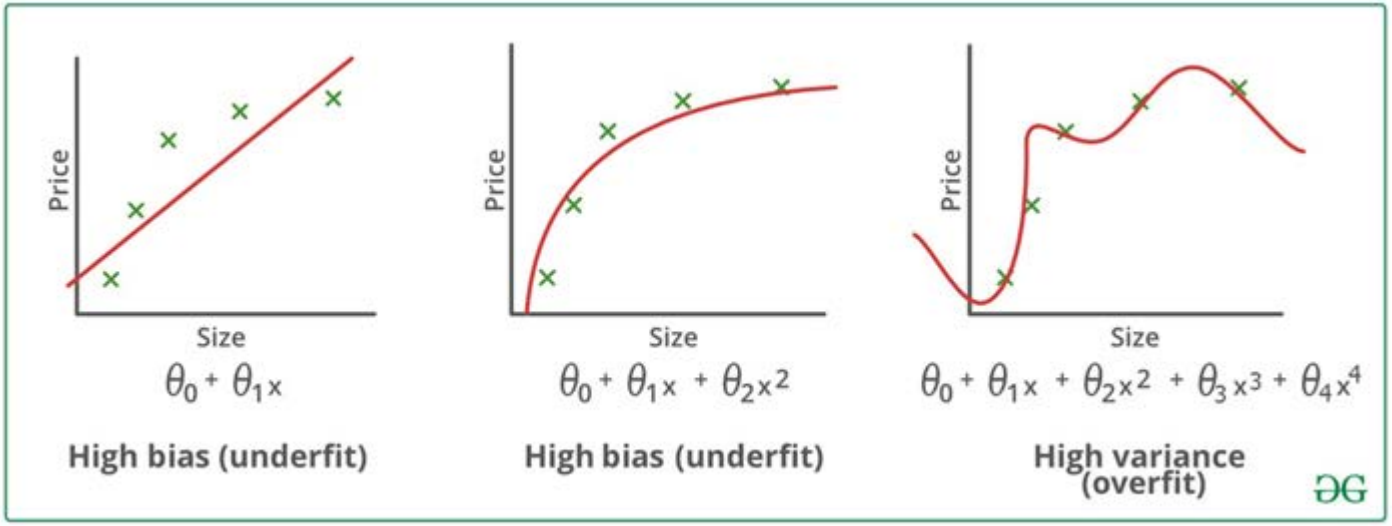
  - ¿Cuál es más útil para detectar una enfermedad leve que se cura con un medicamento? ¿y si es una enfermedad letal y contagiosa? ¿y si en lugar de "enfermo/no-enfermo" la clasificación es "buen-pagador/moroso" para concederle un préstamo?

- En la fase de aprendizaje, podemos indicar qué medida queremos que "optimice" el clasificador en su aprendizaje
  - P.Ej. La exactitud (aunque tiene problemas cuando la distribución de clases de la variable objetivo está desequilibrada) o la medida F1

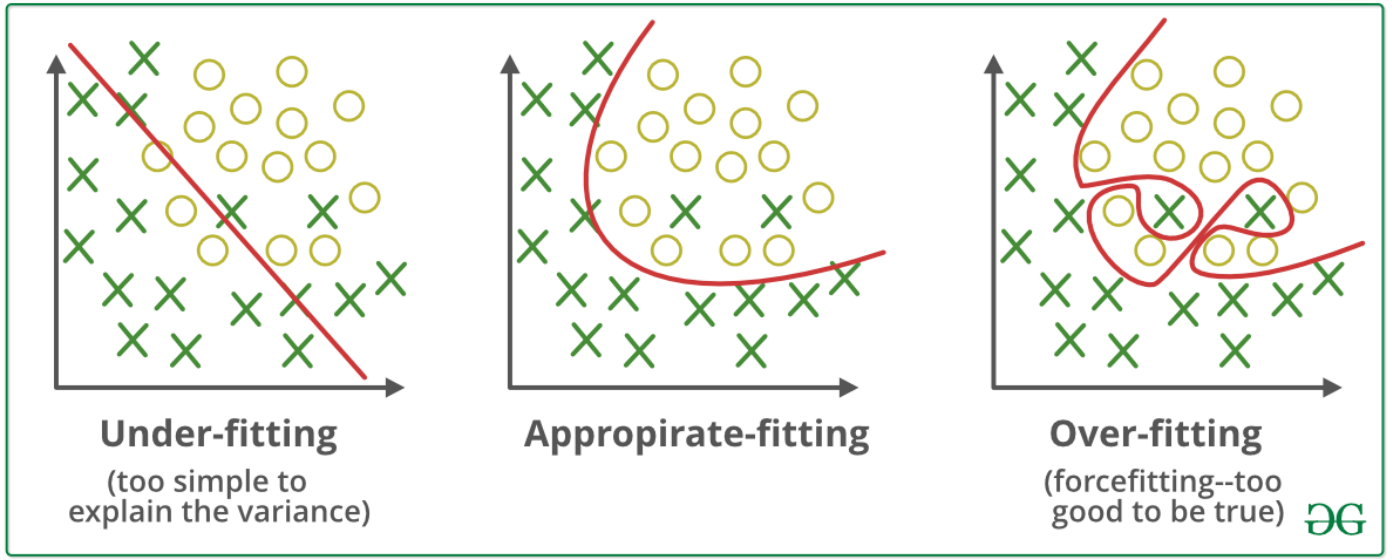
El proceso generador de datos

- Cuando analizamos problemas reales, estos tienen mucha riqueza y variabilidad
  - Además, los datos no estarán exentos de errores de medida, de problemas de incompletitud (podemos no contar con variables relevantes o con valores de las variables para algunos individuos), de situaciones anómalas, de excepciones…
- La variable de salida que observamos no habrá sido generada mediante un modelo matemático (ni determinista, ni estocástico) en base a unas variables de entrada
  - Es decir, en situaciones reales **no existirá un proceso generador de los datos** que usar para validar nuestro modelo
  - Pero no debemos desanimarnos, como dijo el estadístico George Box "*todos los modelos son incorrectos, pero algunos son útiles*" ☺
- Para medir la bondad de nuestra solución solamente contamos con nuestro conjunto de datos y debemos usarlo bien
  - Es posible que nuestro modelo aprenda nuestros datos perfectamente (minimizando el error a cero o casi)
  - En ese caso, seguramente nuestro modelo no sea muy útil porque haya aprendido "de más", haya aprendido "ruido"
  - Existen mecanismos para evitar este problema llamado **sobreaprendizaje** (*overfitting* en inglés)

Under/Over-fitting



Sesgo (bias)
Varianza (variance)



https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

Validación de nuestro modelo

- La validación es una forma de estimar cómo de bueno es nuestro modelo ante datos nuevos

- Como solamente tenemos un conjunto de datos, la validación simple consiste en partirlo en dos conjuntos complementarios (**entrenamiento** y **test**):
  - Un conjunto para **entrenar** nuestro modelo
  - Un conjunto para **evaluar** nuestro modelo

Normalmente, 2/3 para entrenamiento y 1/3 para test, ó 3/4 y 1/4.
Si temenos muuuchos datos (Big Data) puede ser 95% entrenamiento y 5% test.

- Si nuestro modelo está entrenado correctamente su error en el conjunto de entrenamiento no debe ser muy diferente que en el conjunto de test
  - Esto querrá decir que en el entrenamiento no hemos sobre-aprendido y que el modelo generaliza bien cuando se usa con datos con los que no se entrenó

Validación de nuestro modelo

- Si queremos elegir el mejor modelo entre varios, se suele optar por partir los datos en tres conjuntos complementarios (entrenamiento, validación y test):
  - **Entrenamiento**: Para entrenar nuestro modelo
  - **Validación**: Para comparar los modelos parametrizados y elegir el mejor
  - **Test**: Para estimar cómo se comporta el modelo elegido sobre datos limpios

- Esta partición en tres se utiliza especialmente cuando estamos, por ejemplo, probando distintas parametrizaciones de un modelo o comparando distintos modelos
  - En ese caso, el tercer conjunto se hace para no "sobreaprender" ya que las dos primeras particiones las usamos para entrenar y elegir el mejor modelo

- Existen además estrategias más sofisticadas para realizar el entrenamiento y validación de un modelo

Validación cruzada en *k* partes

- La validación cruzada en *k* partes (*k-fold cross validation*) es una forma de estimar cómo de bueno es nuestro modelo ante datos nuevos
- Consiste en los siguientes pasos
  1. Los datos se dividen aleatoriamente en *k* subconjuntos iguales
  2. Se entrena el conjunto con (*k*-1) y se valida con el restante
  3. Se repite *k* veces el paso 2, cambiando el conjunto que se usa para validar
  4. Como medida de error final se suele presentar la media de las *k* medidas de error de validación (aunque puede ser interesante comprobar que las *k* medidas sean similares)

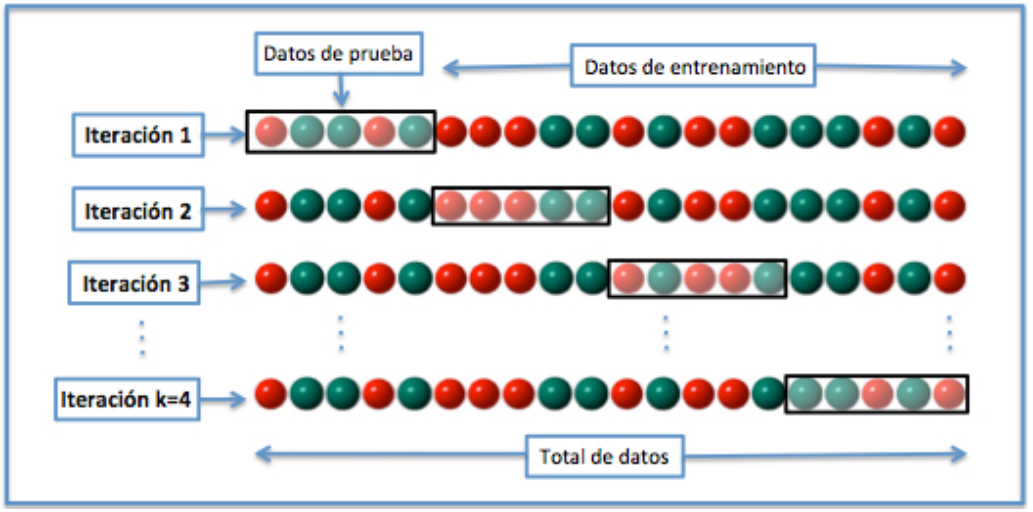


Imagen de Wikimedia Commons

Validación cruzada en *k* partes

- En la validación cruzada en *k* partes todas las observaciones son usadas una vez para validación y *k*-1 veces para entrenamiento
- A veces puede ser interesante **estratificar** las *k* partes:
  - En un problema de regresión significa que la media de la variable respuesta sea similar en todas las partes (o incluso la distribución)
  - En un problema de clasificación significa que la proporción de clases en cada parte sea similar
- Si llevamos la validación cruzada al extremo y tomamos $k = n$, siendo n el número de invididuos en nuestro conjunto de datos, estaríamos utilizando la validación dejando-uno-fuera (*Leave One Out validation* o *LOO validation*)
  - Esta validación es adecuada cuando tenemos pocos datos y no tiene sentido dividirlo en partes mayores

APRENDIZAJE SUPERVISADO
K VECINOS MÁS CERCANOS (K-NN)

Los *k* Vecinos Más Cercanos

- Los *k* vecinos más cercanos (*k nearest neighbors)* o k-NN *es* una técnica de aprendizaje supervisado muy sencilla

- Pertenece al paradigma de aprendizaje perezoso o basado en instancias
  - Perezoso porque no calcula ningún modelo y demora todos los cálculos hasta el momento en que se le presenta un ejemplo nuevo
  - Basado en instancias porque usa todos los individuos disponibles y ante un ejemplo nuevo recupera los más relevantes para componer la solución

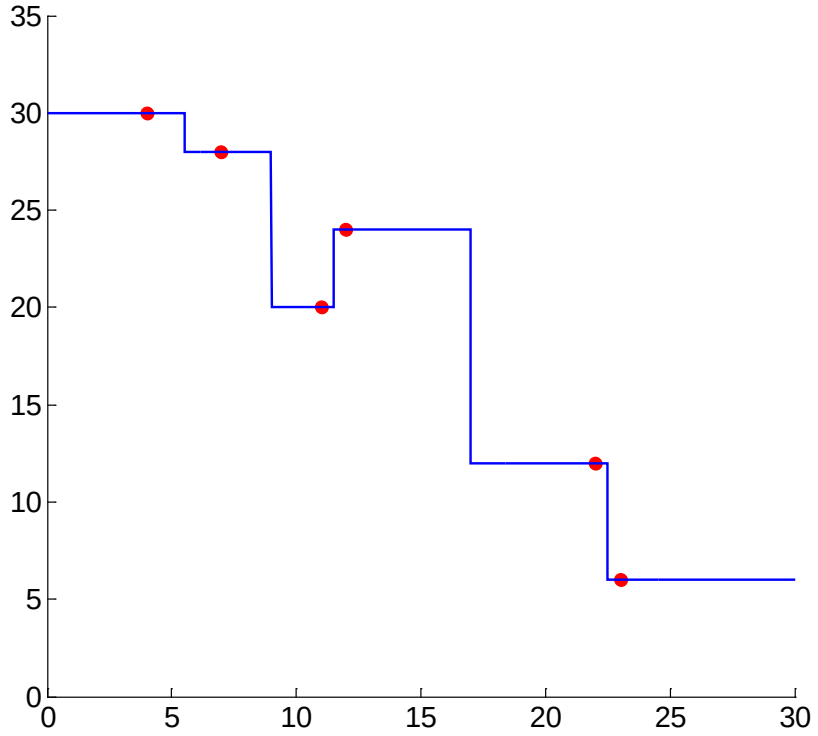
- Se trata de una técnica aplicable en problemas de **regresión** y de **clasificación**

- Dada su sencillez puede ser un buen primer método de aprendizaje automático contra el que comparar otros métodos más sofisticados

El Vecino Más Cercano en un problema de regresión

- Dado un ejemplo nuevo con variable de salida desconocida,
  1. Seleccionar de los n ejemplos disponibles, el ejemplo que más se le parezca
     - Para ello podemos usar una distancia que consideremos adecuada, p.ej. la Euclídea
  2. Ofrecer como solución del ejemplo nuevo la solución del ejemplo seleccionado

Un ejemplo en un problema sencillo de regresión



datos observados

X	11	22	4	12	23	7
Y	20	12	30	24	6	28

Los datos siguen aprox. una relación lineal y el 1-NN parece que produce un sobreajuste, porque la predicción sigue los datos excesivamente

Los *k* vecinos más cercanos (*k* –NN)

- Distancia: Euclídea
  - Se pueden usar otras distancias (L_1, Mahalanobis,…)
- Nº de ejemplos que recupera: *k*
  - Se puede determinar el valor de *k* utilizando alguna estrategia de validación
- Composición de la solución:
  - Si la variable de salida es cuantitativa:
    - **Devuelve como valor solución la media** de los valores solución de los *k* individuos recuperados, es decir, de los *k* individuos más cercanos
    - **Se puede hacer una media ponderada**, donde se dé más peso a los vecinos según su cercanía al ejemplo nuevo . Siendo u el ejemplo nuevo, el peso del vecino v_p es

      $$\psi_p = \frac{1}{d(u,v_p)+\xi}$$

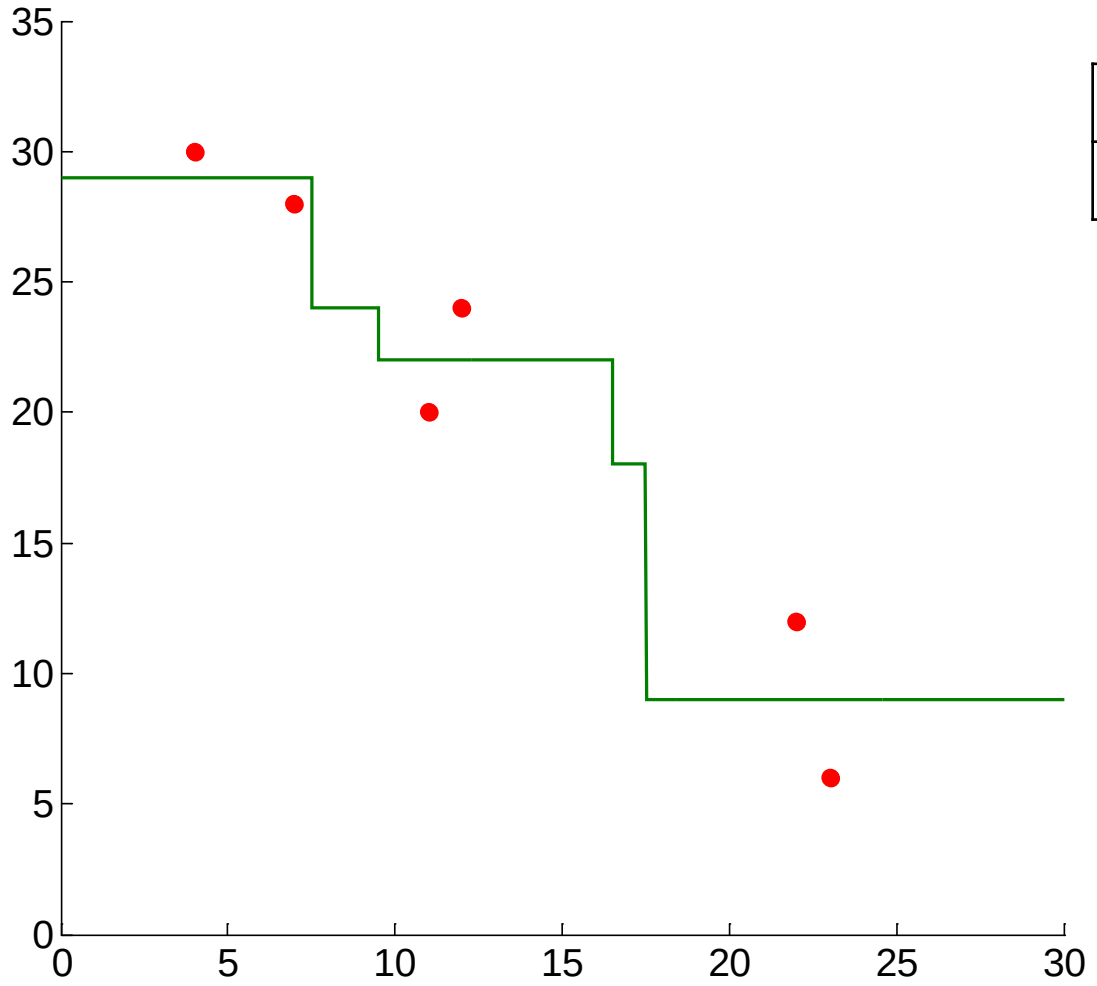
      Con ξ un valor constante muy pequeño que evite que el peso ψ_p tome un valor infinito

      Los pesos de cada vecino se normalizan para que su suma sea 1

      $$\omega_p = \frac{\psi_p}{\sum_{i=1}^{k} \psi_i}$$
  - Si la variable de salida es categórica:
    - Devuelve como valor solución el **valor más frecuente (voto mayoritario)** entre los valores solución de los k individuos recuperados
    - Se puede utilizar la estrategia de voto ponderado, como en el caso cuantitativo

K-NN en un problema de regresión

- Ejemplo con una variable de entrada y $k = 2$



datos observados

X	11	22	4	12	23	7
Y	20	12	30	24	6	28

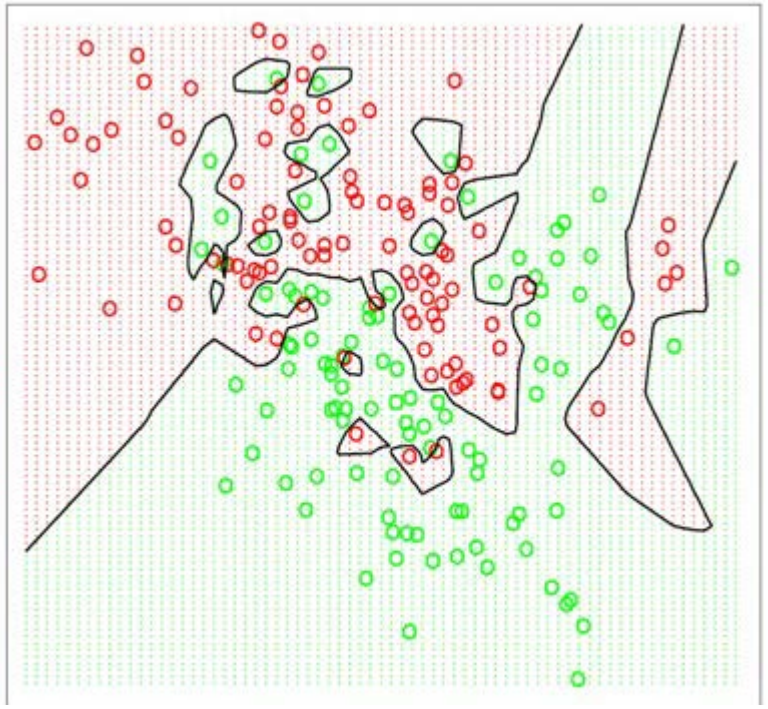
La predicción con $k = 2$ es ligeramente más suave que para $k = 1$
Disminuye el sobreajuste

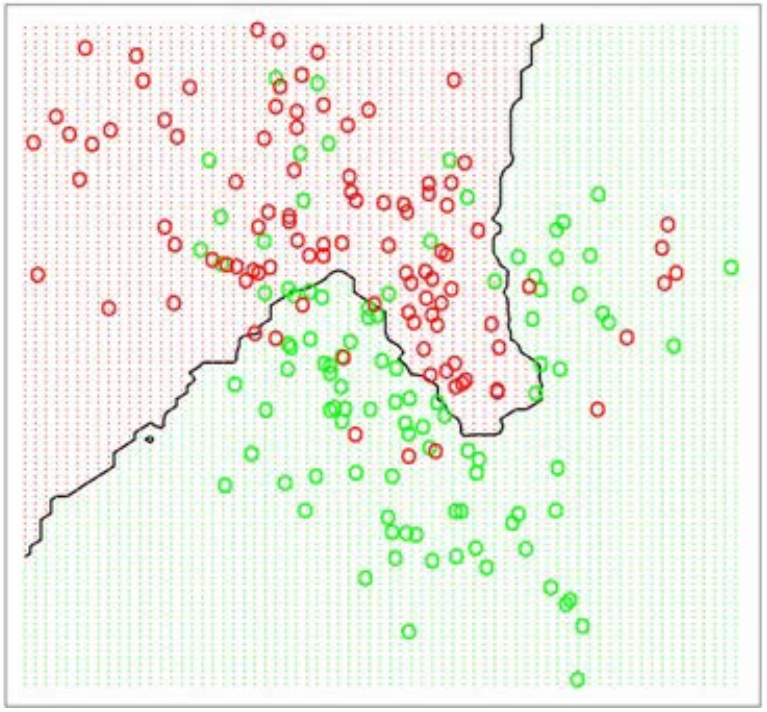
En general, a medida que aumenta k, el suavizado es más acusado (si $k = n$ se obtiene la media)

El k-NN presenta carencias inevitables:
No extrapola bien en los extremos
La predicción es escalonada

K-NN en un problema de clasificación

- Ejemplo con dos variables de entrada y $k = 1$ (izq.) Vs $k = 15$ (dcha.)





Si aumenta k disminuye el sobreajuste y aumenta el suavizado

Si $k = n$ se obtiene la clase mayoritaria

No funciona bien en regiones despobladas, siempre es difícil predecir en una zona donde no hay datos.

Imágenes de Hastie et al. (2001). Elements of Statistical Learning

Determinando la *k* en el *k*-NN

- Aunque el k-NN no estime ningún modelo, debemos determinar el valor de k
  - Si es muy pequeño, corre el riesgo de sobreaprender (aprender conocimiento espúreo)
  - Si es muy grande, corre el riesgo de generalizar demasiado
- Podemos usar una estrategia de validación cruzada para encontrar el valor de k
  - Determinamos un rango de valores de k, por ejemplo $k \in [1,10]$
  - Determinamos una estrategia de validación, p.ej. validación cruzada en k' partes (OJO, ponemos k' porque no es la k del k-NN)
  - Elegimos la k que minimiza el error de validación

El *k*-NN y las variables de entrada

- El k-NN funciona bien en problemas de pocas dimensiones (variables) con datos abundantes.
  - En estos problemas, puede encontrar puntos cercanos relevantes
- El k-NN sufre la maldición de la dimensionalidad
  - Al aumentar los dimensiones los vecinos no suelen estar tan cerca
- Además, su rendimiento empeora si hay variables irrelevantes
  - P.ej. Un punto puede ser muy parecido a otro en cuanto a las variables más relevantes, pero si consideramos variables irrelevantes puede ser que no se le considere como uno de los vecinos más cercanos
- Por tanto, se deben evitar incluir variables irrelevantes o redundantes (este tipo de tarea se llama selección de variables o *feature selection*)
  - Si esto no se sabe a priori, **se puede usar validación cruzada** para seleccionar las variables de entrada que se considerarán de entre todas las posibles
  - Se elegirá aquel subconjunto de variables que minimice el error de validación

Tiempo de cálculo de los *k* Vecinos Más Cercanos

- La búsqueda de los vecinos más cercanos, como mide la distancia a todos los ejemplos, es lógicamente de complejidad $O(\mathrm{n})$, donde n es el número de ejemplos
  - Esto hace que sea un mal método para trabajar con conjuntos de datos muy grandes en tiempo real
- Pero existen técnicas para aliviar este tiempo de búsqueda
  - En primer lugar, se puede **paralelizar** el proceso de forma muy sencilla
    - Dividir los n ejemplos en varias partes y recuperar los k vecinos de cada una
    - Agregar todos los vecinos "parciales" y recuperar los k vecinos "globales"
  - Estrategias basadas en **árboles k-d**
    - Subdividen el espacio de entrada en regiones y reducen la complejidad a $O(\log(n))$ en un espacio con puntos aleatoriamente distribuidos
  - Estrategias aproximadas basadas en **tablas hash**
    - No hacen la búsqueda exacta sino una búsqueda aproximada sobre un conjunto de tablas hash que representan proyecciones de los datos en espacios unidimensionales
      - La consulta de una tabla hash es de complejidad $O(1)$

APRENDIZAJE SUPERVISADO
ÁRBOLES DE DECISIÓN

Árboles de decisión

- Los árboles de decisión son una técnica de aprendizaje supervisado
- Pueden usarse en problemas de clasificación o de regresión
  - Aquí los veremos únicamente en clasificación
- Permite el aprendizaje de conceptos de forma inductiva
  - A partir de un conjunto de ejemplos de entrenamiento,
  - Construye un **árbol de decisión** donde cada nodo pregunta por el valor de una variable
  - El árbol construido sirve para clasificar ejemplos nuevos (sin etiquetar o del conjunto de validación)
    - Colocando el ejemplo nuevo en la raíz y respondiendo las preguntas
    - Con ello desciende por las ramas hasta llegar a un nodo hoja
- El árbol se construye recursivamente
  - En cada nodo se hace una pregunta que busca disminuir la entropía de los nodos hijos
- El árbol aprendido es muy **interpretable** y puede ser analizado por el experto para entender la clasificación o definir conceptos
  - Aunque el árbol puede ser tan grande que sea muy difícil de interpretar

Un problema de clasificación

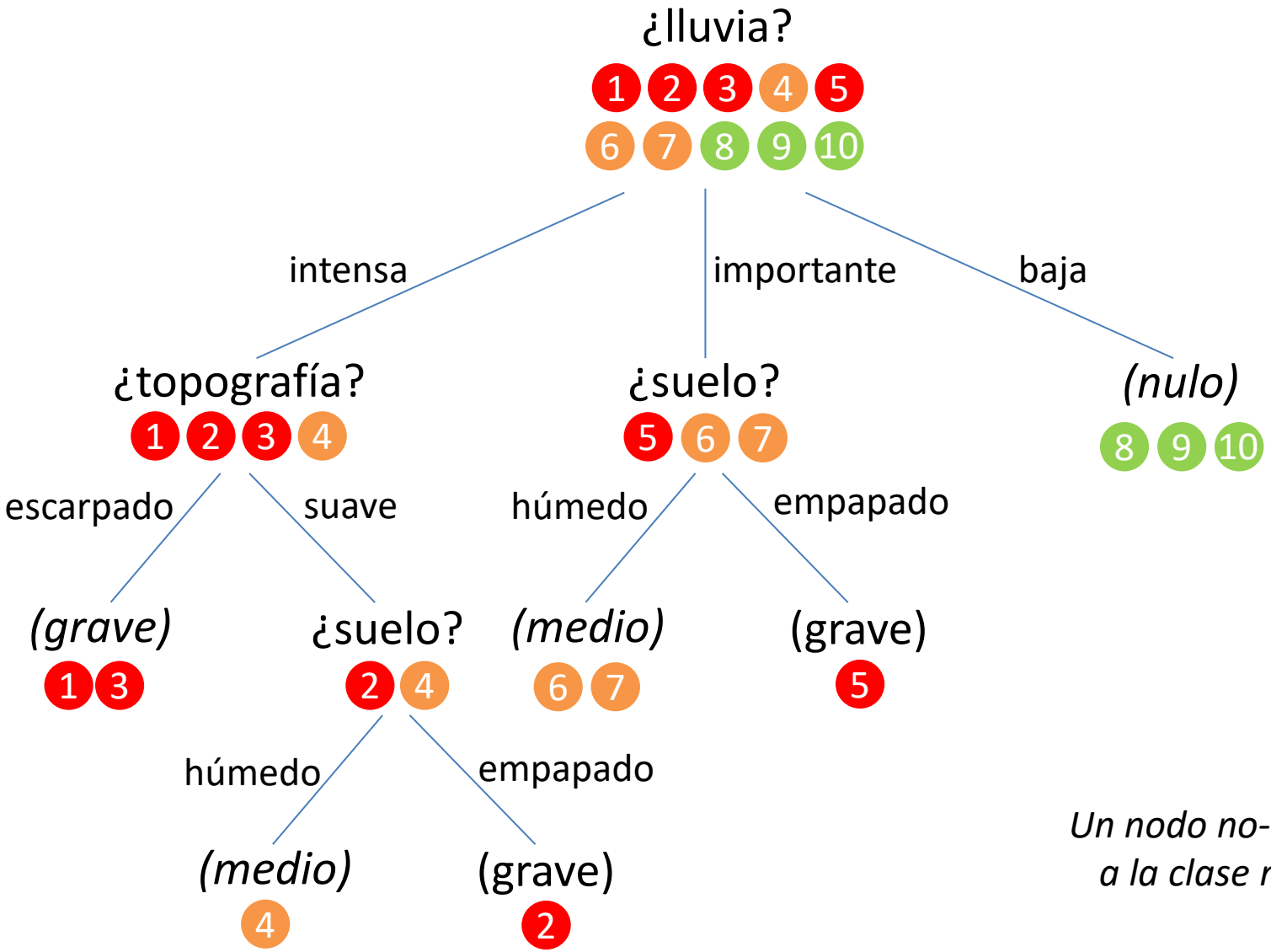
- Aprender a clasificar situaciones de riesgo de inundación
  - En función de las variables *lluvia, suelo y topografía*
- Ejemplos de entrenamiento:

Caso	Lluvia	Suelo	Topografía	Riesgo
1	intensa	empapado	escarpada	grave
2	intensa	empapado	suave	grave
3	intensa	húmedo	escarpada	grave
4	intensa	húmedo	suave	medio
5	importante	empapado	escarpada	grave
6	importante	húmedo	escarpada	medio
7	importante	húmedo	suave	medio
8	baja	empapado	escarpada	nulo
9	baja	húmedo	escarpada	nulo
10	baja	húmedo	suave	nulo

Representación arborescente

- Representación natural de criterios de decisión de las personas
  - Nodos: preguntas (atributos o características)
  - Arcos: posibles respuestas (valores posibles en los ejemplos)
  - Hojas están etiquetadas con la predicción (clase de ese ejemplo)

Árbol de decisión de nuestro problema



Un nodo no-hoja clasifica a la clase mayoritaria

Construcción de árboles de decisión

- Se trata de un **proceso inductivo de la raíz a las hojas** (*top-down*)

- En cada nodo se da una **selección voraz del mejor atributo**
  - Calcular un criterio de optimalidad para todos los atributos
  - Seleccionar el atributo que mejor valor de optimalidad tiene
  - Generar los nodos hijos en función del atributo seleccionado

- **Divide y vencerás** aplicado recursivamente
  - Se dividen los ejemplos de cada nodo en función del atributo elegido
  - El proceso se aplica recursivamente en los nodos hijos
  - El proceso termina si:
    - Todos los elementos pertenecen a la misma clase
    - No hay más atributos que elegir

- El algoritmo más famoso que construye árboles de decisión es el ID3 (Quinlan, 86)
  - Usa teoría de la información para estimar el mejor candidato
  - Se consigue complejidad lineal

Algoritmo ID3

- El árbol se construye de arriba a abajo, trabajando por niveles y dentro de cada nivel trabaja por nodos
- En cada nodo se pretende:
  - Obtener el atributo en base al cual obtener los nodos hijos
  - Se selecciona el que mejor discrimine entre el conjunto de ejemplos
    - El atributo más discriminante será aquél que conduzca a un estado con menor entropía o menor desorden (mayor información)
  - Heurística para obtener árboles pequeños (en profundidad)

- La entropía (Shannon, 1948) mide la ausencia de homogeneidad de un conjunto de ejemplos con respecto a su clase
  - Es una medida estándar de "desorden" (*0 es homogeneidad total)*
    - Utilizada en física y en teoría de la información
- **Ganancia de información** es la diferencia entre la entropía del conjunto original y la de los subconjuntos obtenidos
  - También se emplea el término **disminución de la entropía**

Entropía

- En un problema de clasificación donde la variable de salida y tiene s_i posibles clases con $i = 1, \dots, p$, la entropía de un nodo N se define como:

$$E(N) = -\sum_{i=1}^{p} P(s_i) \log_2 P(s_i),$$

siendo $P(s_i)$ la probabilidad de que un ejemplo tenga la clase s_i en el nodo N, y calculándose únicamente para las clases observadas en el nodo N (para evitar el cero en el logaritmo)

- A esta entropía se le llama entropía inicial de un nodo N (antes de clasificar los ejemplos que contiene en base a alguno de los atributos)

- La entropía final de un nodo N tras usar el atributo A que tiene q valores (a_j)

$$E(N|A) = \sum_{j=1}^{q} P(a_j) \left(-\sum_{i=1}^{p} P(s_i|a_j) \log_2 P(s_i|a_j) \right),$$

siendo $P(s_i|a_j)$ la probabilidad de que un ejemplo del nodo hijo con $A = a_j$ tenga la clase s_i

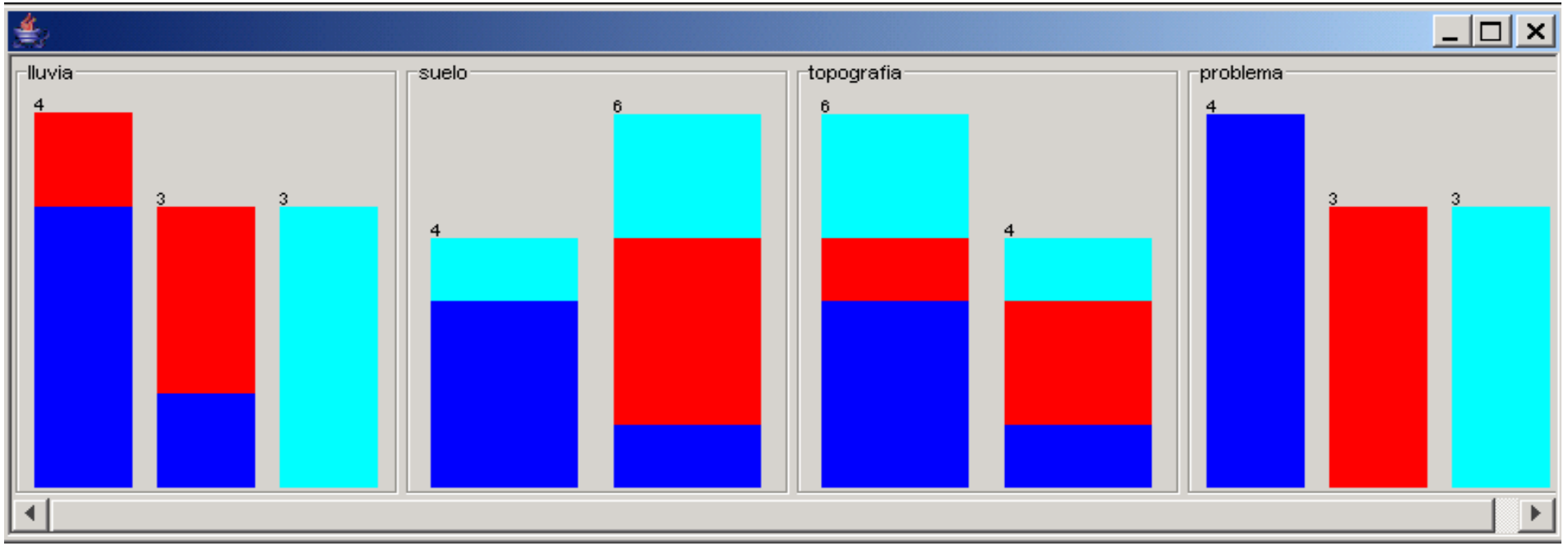
Disminución de la Entropía o Ganancia de Información

- Para cada atributo A, B, C ... se calcula la disminución de entropía (DE) causada por su utilización
  - $DE(N, A) = E(N) - E(N|A)$
  - $DE(N, B) = E(N) - E(N|B)$
  - $DE(N, C) = E(N) - E(N|C)$...

- En cada nodo, se selecciona aquel atributo que mayor **disminución de entropía** genera (o **ganancia de información)**

- Aplicado al ejemplo
  - Hay 3 clases y 3 atributos

ID3: ejemplo



intensa

importante

baja

empapado

húmedo

escarpada

suave

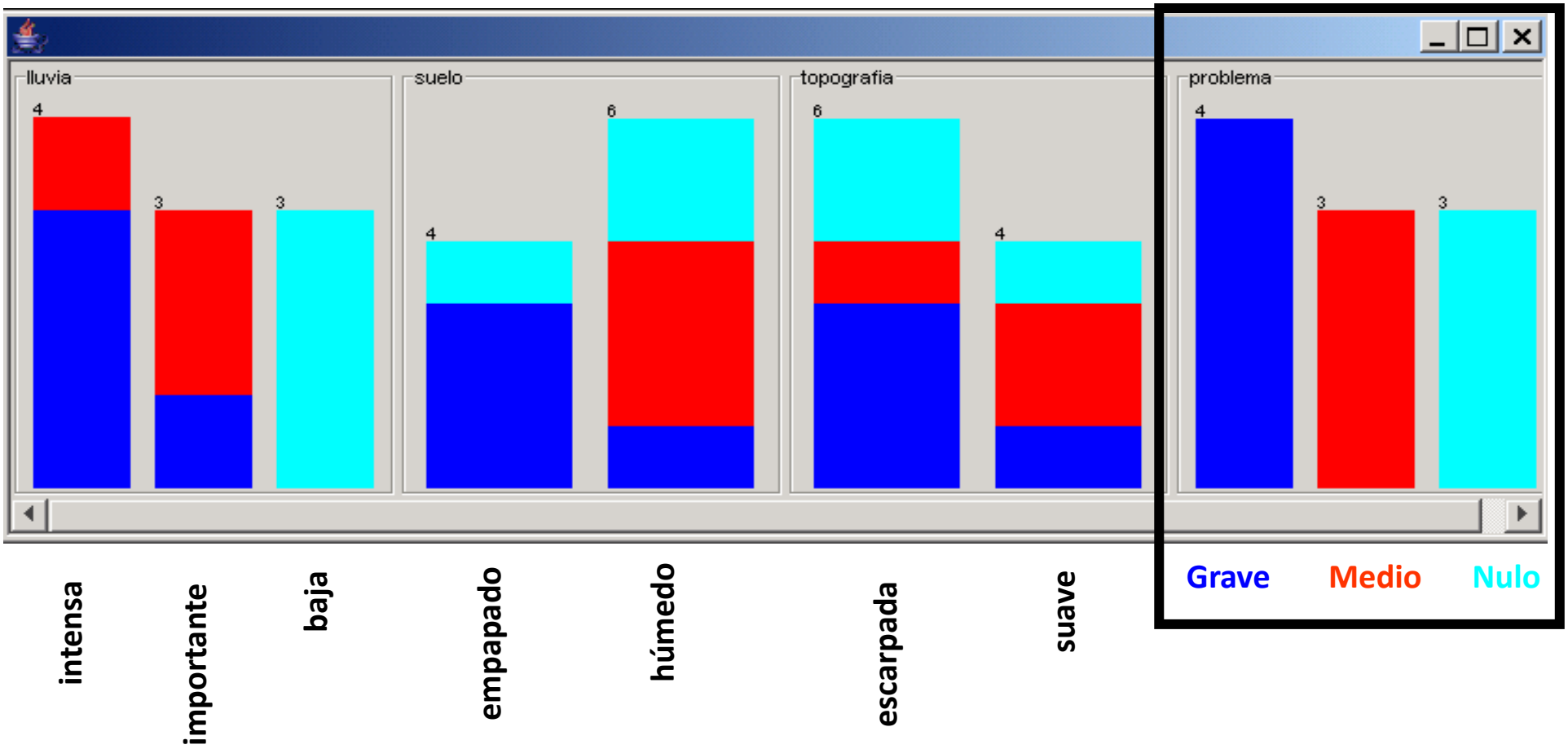
Grave Medio Nulo

ID3: ejemplo

- Entropía inicial en la raíz del árbol: (del problema global)

P(grave) = 0,4 P(medio) = 0,3 P(nulo) = 0,3

$E(raíz) = -0,4 \log_2 0,4 - 0,3 \log_2 0,3 - 0,3 \log_2 0,3 = 1,571$



ID3: ejemplo

- Entropía final clasificando según lluvia (A):

  a_1: lluvia intensa,

  a_2: lluvia importante,

  a_3: lluvia baja

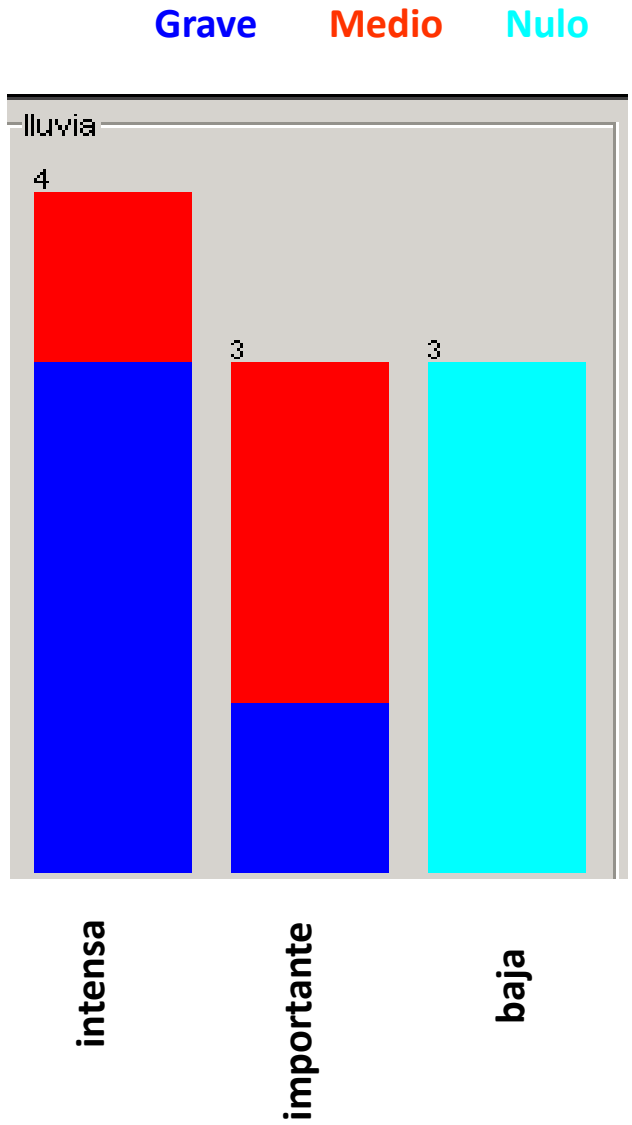
$E(raíz|a_1)= -0,75 \log_2 0,75 – 0,25 \log_2 0,25 = 0,811$

$E(raíz|a_2)= 0,918$

$E(raíz|a_3)= 0$

$E(raíz|A)= 0,4*0,811 + 0,3*0,918= 0,6$

$DE(raíz,A) = 1,571 – 0,60 = 0,971$

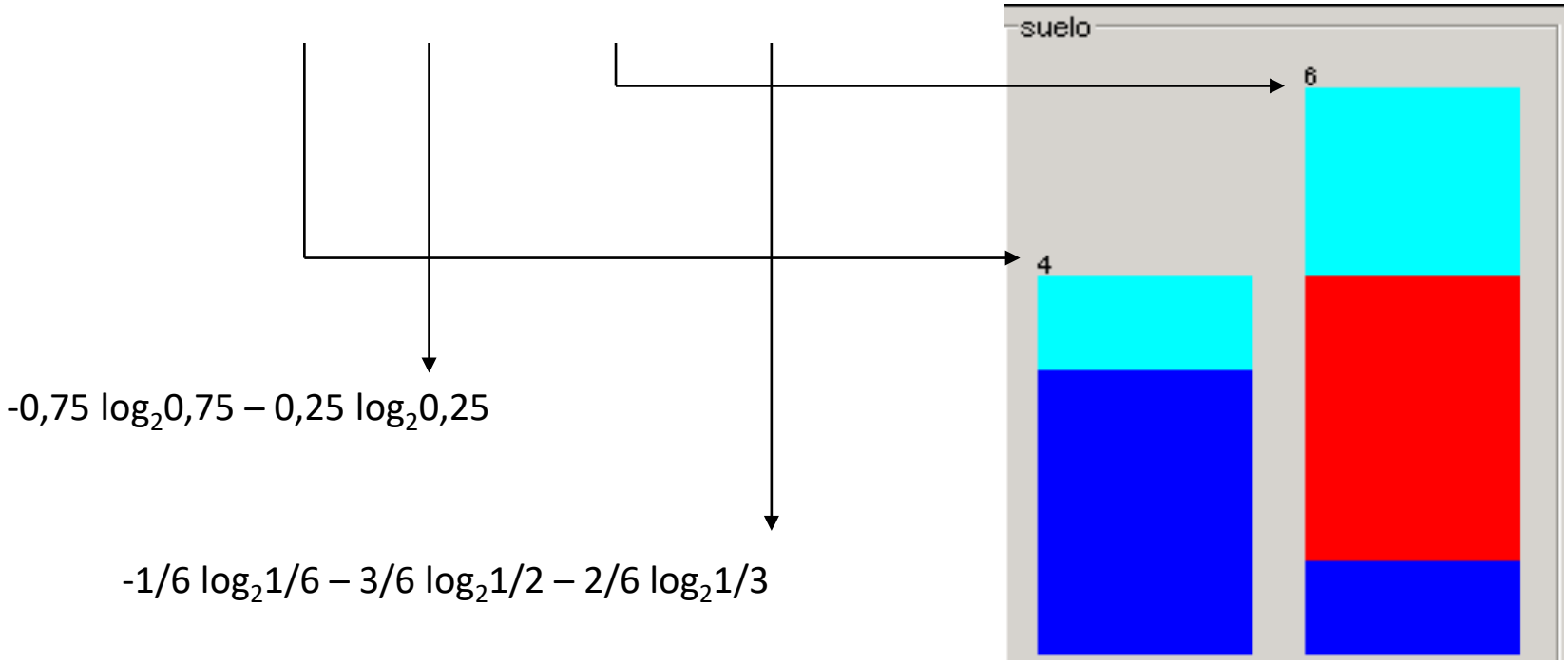


ID3: ejemplo

Grave **Medio** **Nulo**

- Entropía final clasificando según suelo (B):

$$E(raíz|B) = 0,4*0,811 + 0,6*1,459 = 1,20$$



$$DE(raíz,B) = 1,571 – 1,20 = 0,371$$

ID3: ejemplo

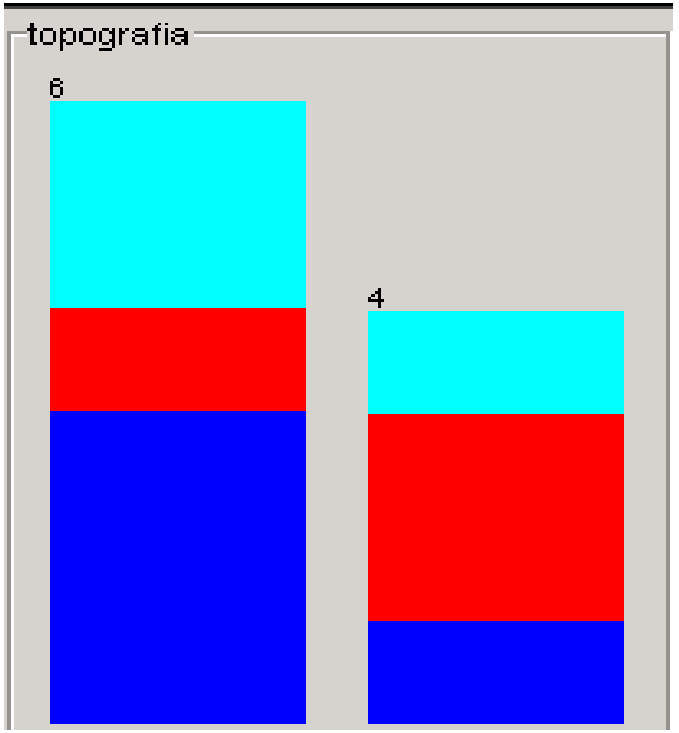
Grave **Medio** **Nulo**

- Entropía final clasificando según topografía (C):

$$-1/4 \log_2 1/4 - 2/4 \log_2 2/4 - 1/4 \log_2 1/4$$

$$E(\text{raíz}|C) = 0{,}6*1{,}459 + 0{,}4*1{,}50 = 1{,}475$$

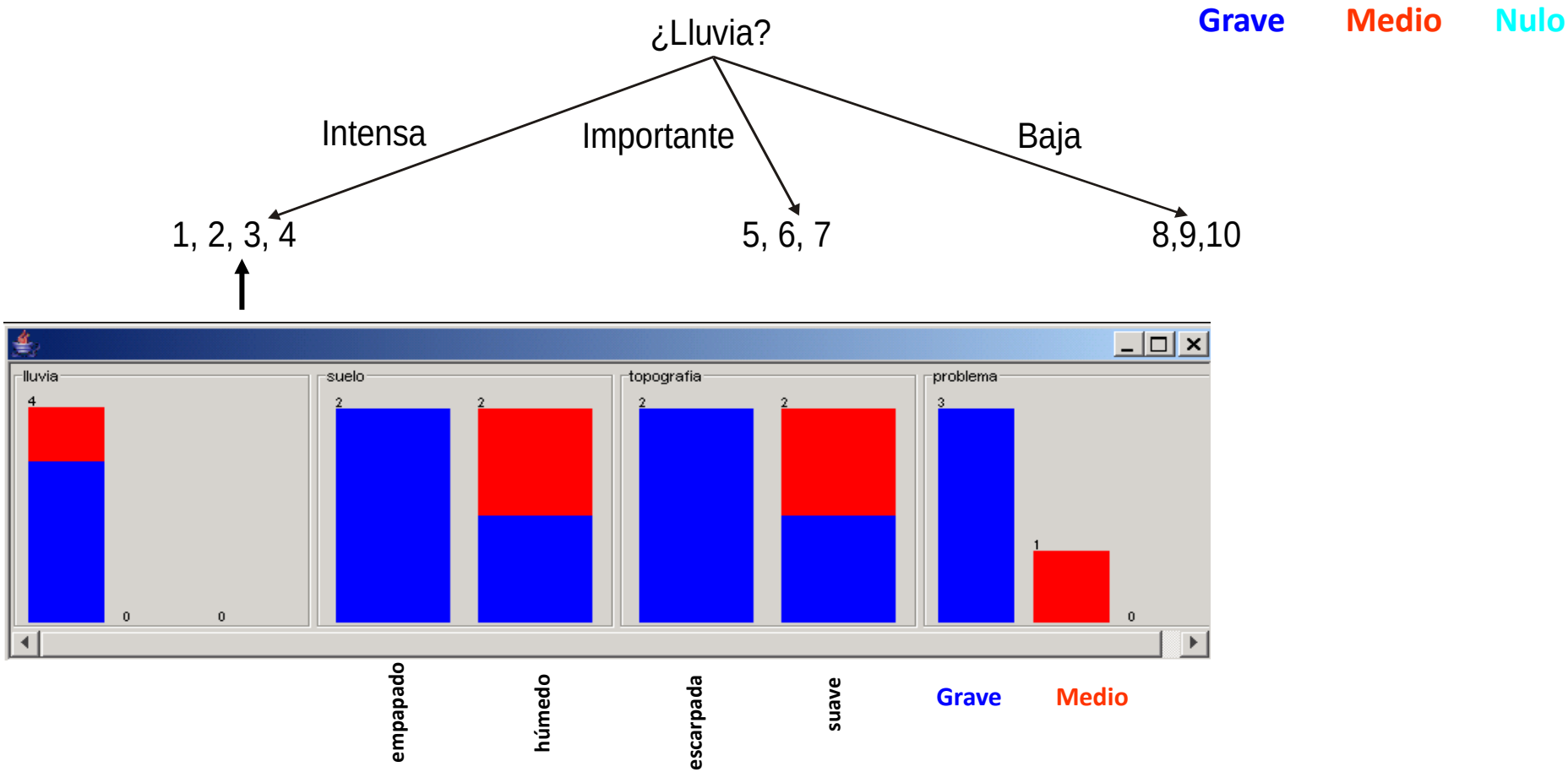
$$-1/6 \log_2 1/6 - 3/6 \log_2 1/2 - 2/6 \log_2 1/3$$



$$DE(\text{raíz},C) = 1{,}571 - 1{,}475 = 0{,}096$$

La mayor disminución de entropía se consigue con el atributo A y por ello éste es el seleccionado para el primer nivel del árbol

ID3: ejemplo



- En la siguiente iteración se vuelve a aplicar el algoritmo sobre cada uno de los tres nuevos nodos, considerando en cada uno el subconjunto de ejemplos obtenido y habiendo eliminado el atributo lluvia del conjunto de atributos

Interpretando el árbol del ID3

- Selección “automática” de variables relevantes
  - Como en cada nodo se elige el atributo que más “información” aporta a la clasificación, el ID3 por definición está eligiendo las variables más relevantes para solucionar el problema
  - En ese sentido, no es grave que haya variables irrelevantes entre las variables de entrada → el algoritmo no las elegirá (en principio)
  - Para un nodo determinado, hay variables que pueden “medir” el mismo aspecto y/o tener similar poder discriminante, la selección de una u otra dependerá de su valor de disminución de la entropía
    - Sin embargo, la elección puede condicionar la estructura del árbol a partir de ese nodo
- Legibilidad de la solución
  - La estructura arborescente es **inteligible** por un experto
  - El experto puede aprender sobre el problema que tiene entre manos leyendo el árbol, por ejemplo:
    - Qué variables tienen mayor poder discriminante
    - Qué valores de variables están asociados con una clase u otra
    - Etc

Finalización del ID3

- Terminación:
  - La expansión de un nodo se detiene cuando todos sus ejemplos pertenecen a la misma clase (≡ entropía nula)
  - El proceso se detiene cuando no se puede seguir expandiendo ningún nodo porque no quedan atributos (variables de entrada)
    - Si hay variables cuantitativas, éstas se pueden usar muchas veces (*ver más adelante*)
  - A las hojas se les asigna la clase mayoritaria a la que pertenecen sus ejemplos

- Riesgo de sobreaprendizaje
  - Si extendemos un árbol hasta el final, estamos obteniendo una representación que aprende lo mejor posible los datos de partida
    - Es posible que algunas ramificaciones aprendan errores o aspectos de los datos de entrenamiento no generalizables a otras muestras
  - Eso constituye un riesgo a evitar y para ello hay alternativas que no generan árboles completos. Por ejemplo:
    - Utilizar un umbral de disminución de la entropía por debajo del cual no ramificar
    - Usar mecanismos de poda
  - Además, siempre conviene evaluar el sobreaprendizaje de un árbol (ya sea completo o no) mediante una estrategia de validación

Poda del árbol de decisión

- La poda del árbol de decisión se realiza para
  - Simplificar el árbol y ganar en capacidad de interpretación y legibilidad
  - Evitar el sobreajuste y mejorar la capacidad de generalización

- La poda consiste principalmente en cortar las ramas malas (subárboles) haciendo que el árbol no se expanda en dicho punto

- Vamos a ver dos ejemplos:
  - Poda mediante reducción del error
  - Poda pesimista

Poda mediante reducción del error

- Requiere un conjunto de ejemplos no usados para generar el árbol

- Consta de los siguientes pasos:
  1. Clasifica dichos ejemplos y calcula el error que cometen todas las hojas
  2. El número de errores de un subárbol es la suma de los errores de todas sus hojas
  3. Calcula el número de errores que cometería el nodo que origina el subárbol si no se expandiera
  4. Si los errores sin expandir son iguales o menores que expandido, se poda el subárbol
     - Se recalcula el número de errores de los subárboles que contenían dicho subárbol

  – **Ventaja**: Evita el sobreajuste de forma efectiva
  – **Inconveniente**: Requiere de nuevos ejemplos

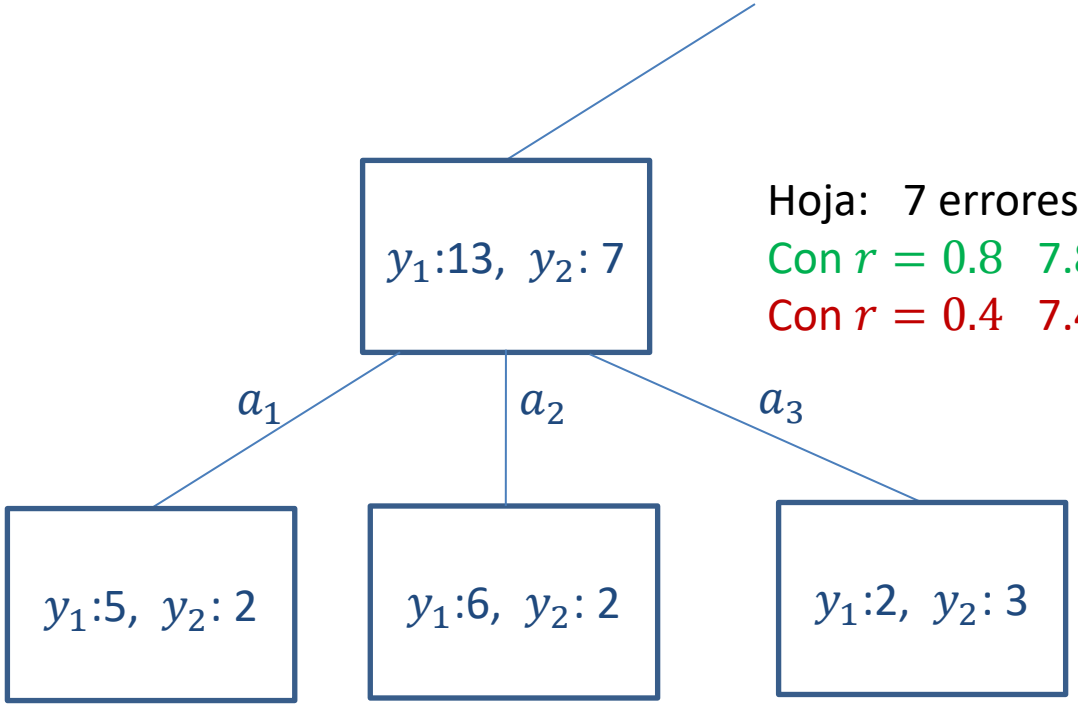
Poda pesimista

- Consta de los siguientes pasos:
  1. Clasifica un conjunto de ejemplos (usado o no para generar el árbol)
  2. Para cada nodo hoja, calcula los errores que comete y añade un valor prefijado r
  3. El número de errores de un subárbol es la suma de los errores de todas sus hojas
  4. Calcula el número de errores que cometería el nodo que origina el subárbol si no se expandiera e increméntalo también con r
  5. Si los errores sin expandir son iguales o menores que expandido, se poda el subárbol
     - Se recalcula el número de errores de los subárboles que contenían dicho subárbol

  – Ventaja: No necesita ejemplos extra
  – Inconveniente: El número de elementos en la hoja no afecta

Ejemplo de poda pesimista

Poda pesimista con $r = 0.8$ y $r = 0.4$



Hoja: 7 errores
Con $r = 0.8$ 7.8 errores
Con $r = 0.4$ 7.4 errores

TOTAL= 6 errores

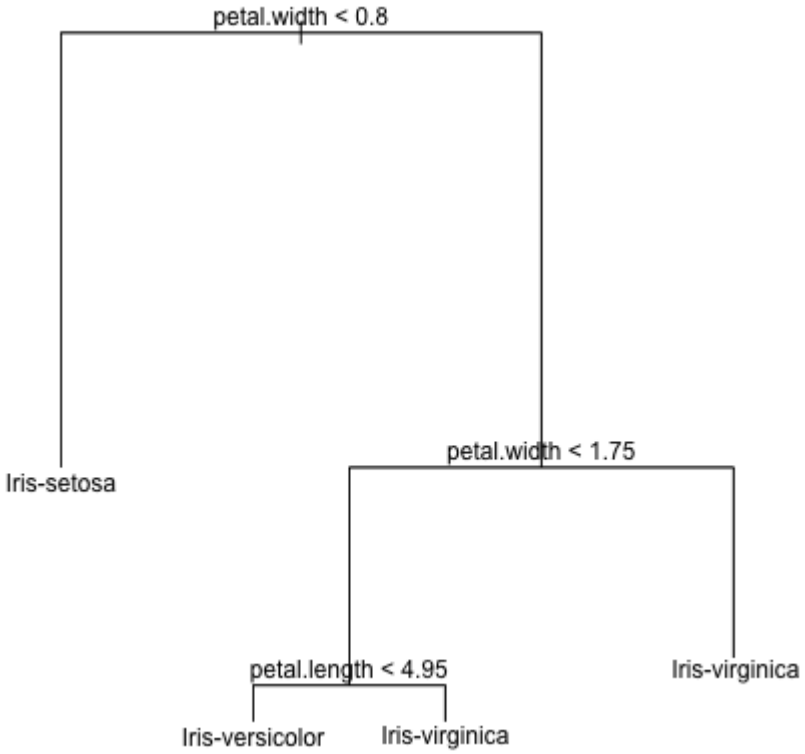
Con $r = 0.8$	2.8 errores	2.8 errores	2.8 errores	TOTAL=8.4 errores	**Podar**
Con $r = 0.4$	2.4 errores	2.4 errores	2.4 errores	TOTAL=7.2 errores	**No podar**

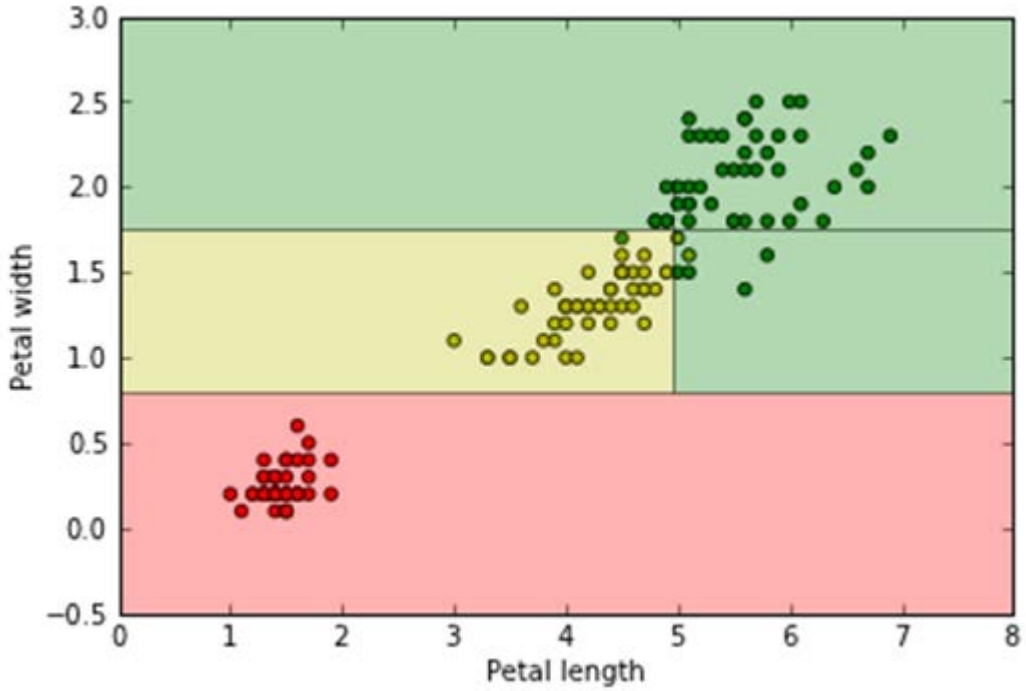
Manejo de variables de entrada continuas en ID3

- Para manejar variables de entrada numéricas se discretizan los valores de las variables en intervalos generalmente booleanos.
  - En primer lugar, ordenamos los valores de la variable cuantitativa con su respectivo valor de la clase
  - Se determinan los puntos umbrales que son los valores para los cuales la clase cambia
  - De entre los umbrales elegimos el que produce mayor disminución de la entropía.
- Ejemplo con una variable continua:
    - X_i: 40,48,60,72,80,90
    - Y: no, no, si, si, si, no
  - Puntos umbrales candidatos
    - Entre 48 y 60 (el punto medio sería 54)
      - La partición binaria candidata según el atributo X podría ser $X_i < 54$
    - Entre 80 y 90 (el punto medio sería 85)
      - La partición binaria candidata según el atributo X podría ser $X_i < 85$
- En nuestro ejemplo, elegiríamos la partición que genera la pregunta $X_i < 54$ porque genera una mayor disminución de la entropía
  - Si hubiera otras variables (numéricas o categóricas) a considerar, tendríamos que comparar la disminución de la entropía $X_i < 54$ con la disminución de la entropía que generan las particiones usando las otras variables

Ejemplo con el problema de la flor del Iris

- Si usamos los datos del Iris y dos variables ancho y largo del pétalo, podemos obtener un árbol como éste





Observa que en el nodo raíz se podría haber elegido también petal.length < 2.5
El resto del árbol no cambiaría, pero el color de las regiones donde no hay observaciones sí → En regiones sin observaciones es arriesgado "opinar"

Refinamientos del ID3

- El propio Quinlan creó una versión posterior del ID3 que es el algoritmo C4.5 (Quinlan 93)
  - Soluciona un pequeño problema de ID3: tiene una cierta tendencia a favorecer la elección de atributos con muchos valores posibles, lo que redunda en una peor generalización de las observaciones
  - Para evitarlo el algoritmo C4.5 utiliza como criterio de selección de atributos la ratio de ganancia de información
  - El algoritmo también incluye manejo de variables continuas, valores perdidos y mecanismo de poda
- Quinlan refinó el algoritmo C4.5 en la versión C5.0, pero es una versión comercial licenciada a través de la empresa *RuleQuest Research*
  - *Existen versiones libres, pero que son también interpretaciones "libres" del algoritmo*
- En sklearn se usa CART (Classification and Regression Trees) que es muy parecido a C4.5.