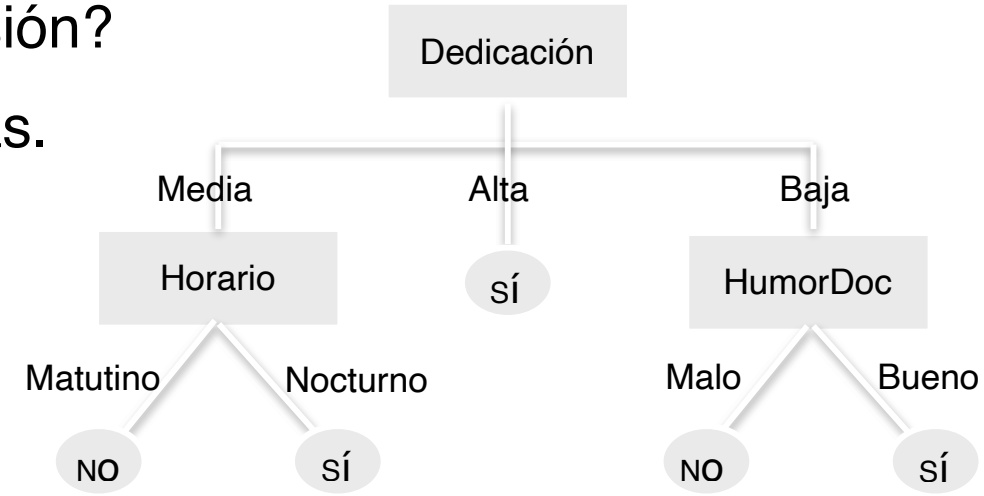
ÁRBOLES DE DECISIÓN

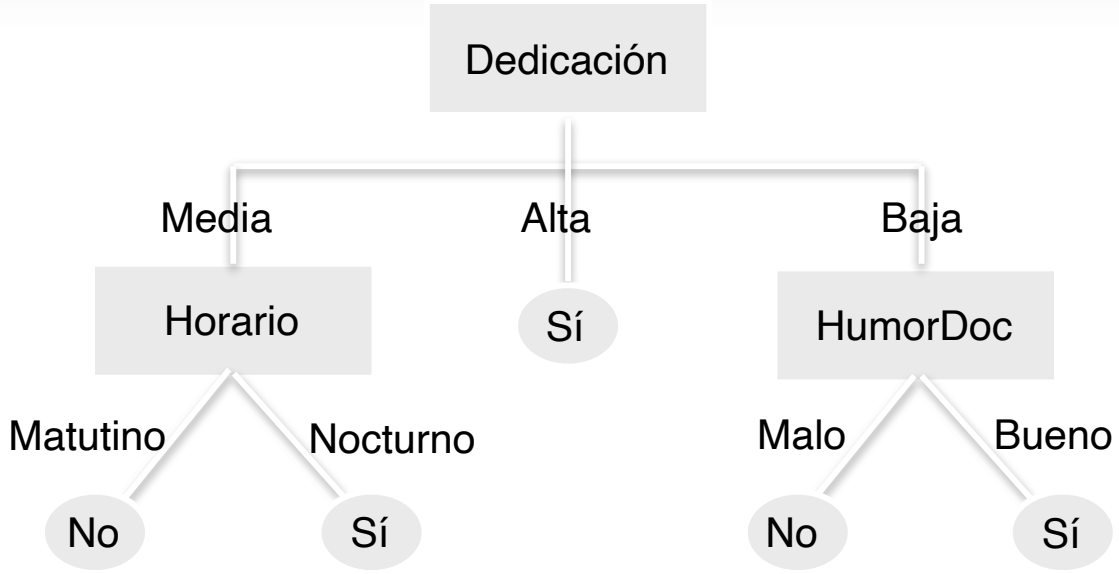
Introducción

- ¿Qué hacen los árboles de decisión?
  - Aproximan funciones discretas.



- ¿Cómo?
  - Clasifican las instancias yendo de la raíz hacia las hojas.
  - En cada nodo se verifica un atributo y se baja por la rama asociada al valor de la instancia.
  - El proceso se repite hasta llegar a una hoja, en donde se encuentra el resultado.

Introducción



- ¿Cómo son clasificadas...?

  <Ded=Media, Dif=Alta, Hor=Noc, Hum=Alta, Hdoc=Malo>➔ SÍ

  <Ded=Baja, Dif=Alta, Hor=Noc, Hum=Alta, Hdoc=Bueno>➔ SÍ

Introducción

- Cada rama del árbol es una restricción sobre los valores expresada como una conjunción.
- Los árboles se pueden ver como las disyunción de las restricciones representadas por sus ramas

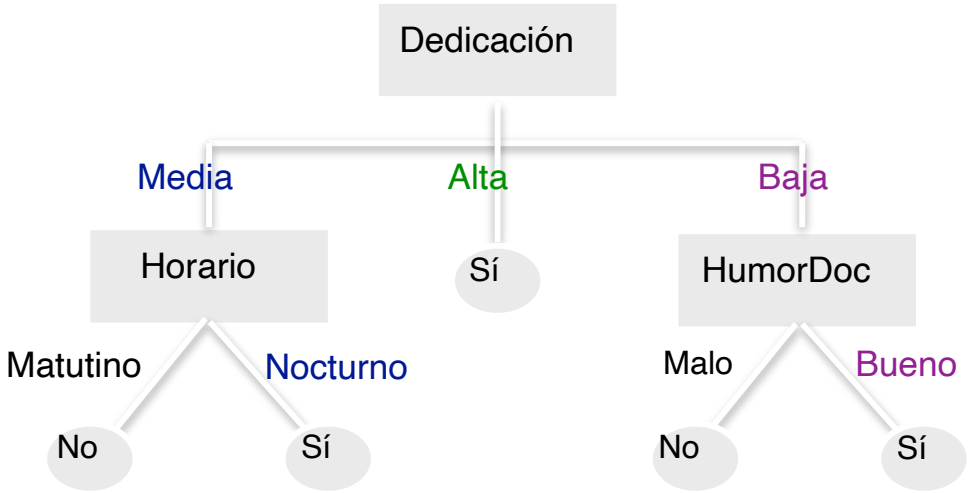
(Dedicación=Media ∧ Horario=Nocturno)

∨

(Dedicación=Alta)

∨

(Dedicación=Baja ∧ HumorDoc=Bueno)



Ejemplos

#Ej	Fiebre	Tos	Olfato	Covid
1	Sin Fiebre	Con Tos	Sin Olfato	Sí
2	Sin Fiebre	Sin Tos	Sin Olfato	Sí
3	Con Fiebre	Con Tos	Con Olfato	No
4	Con Fiebre	Sin Tos	Con Olfato	No
5	Sin Fiebre	Con Tos	Con Olfato	No
6	Sin Fiebre	Sin Tos	Con Olfato	No

#Ej	Tos	Olfato	Covid
1	Sin Tos	Sin Olfato	Sí
2	Con Tos	Sin Olfato	Sí
3	Sin Tos	Con Olfato	No
4	Sin Tos	Con Olfato	No
5	Sin Tos	Con Olfato	Sí

Introducción

- Problemas en los que se aplican:
  - Las instancias se representan como un conjunto de parejas atributo-valor
  - La función objetivo toma valores discretos
  - Las descripciones requieren disyunciones
  - El conjunto de entrenamiento puede contener errores
  - Las instancias de entrenamiento pueden no tener todos los atributos

- Algunas aplicaciones prácticas:
  - Detección de fraudes con tarjetas de crédito.
  - Toma de decisiones médicas.
  - Base para algoritmos más complejos

Introducción

- Problemas en los que se aplican:
  - Las instancias se representan como un conjunto de parejas atributo-valor
  - La función objetivo toma valores discretos
  - Las descripciones requieren disyunciones
  - El conjunto de entrenamiento puede contener errores
  - Las instancias de entrenamiento pueden no tener todos los atributos

- Algunas aplicaciones prácticas:
  - Detección de fraudes con tarjetas de crédito.
  - Toma de decisiones médicas.
  - Base para algoritmos más complejos

Algoritmo básico

- La mayoría de los algoritmos para aprender un A.D. utilizan una técnica *greedy*.

- En particular, vamos a ver el ID3:
  - Construye el árbol de la raíz a las hojas (*top-down*).
  - En cada paso se decide por cuál atributo se debe preguntar y se genera una rama por cada posible valor del atributo seleccionado.
  - Se repite recursivamente el proceso en cada una de las ramas generadas, tomando aquellas instancias que tienen el valor de la rama en el atributo seleccionado.
  - Nunca se vuelve hacia atrás una decisión; nunca se verifica que el atributo seleccionado haya sido realmente el mejor.

Algoritmo básico

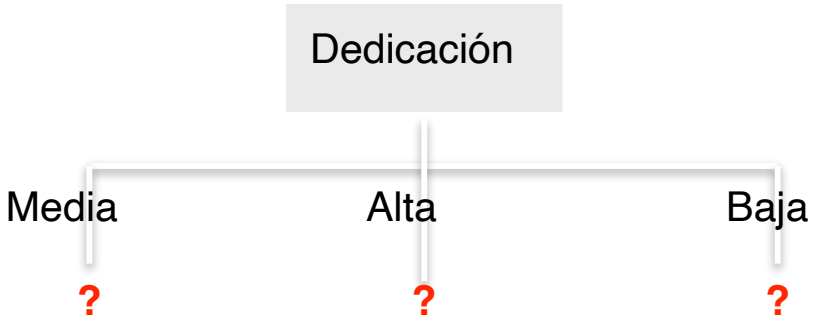
- Crear una raíz
- Si todos los ej. tienen el mismo valor → etiquetar con ese valor
- Si no me quedan atributos → etiquetar con el valor más común
- En caso contrario:
  - La raíz pregunta por A, atributo que mejor clasifica los ejemplos
  - Para cada valor v_i de A
    - Genero una rama
    - $Ejemplos_{vi}$={ejemplos en los cuales A=v_i }
    - Si $Ejemplos_{vi}$ es vacío → etiquetar con el valor más probable
    - En caso contrario → ID3($Ejemplos_{vi}$, Atributos -{A})

Algoritmo básico - Ejemplo

- En nuestro ejemplo, un oráculo determina que el mejor atributo es Dedicación:

#Ej	Dedicación	Dificultad	Horario	Humedad	Humor Doc	Salva
1	Alta	Alta	Nocturno	Media	Bueno	Sí
2	Baja	Media	Matutino	Alta	Malo	No
3	Media	Alta	Nocturno	Media	Malo	Sí
4	Media	Alta	Matutino	Alta	Bueno	No

- Creamos una rama por cada posible valor:



Algoritmo básico - Ejemplo

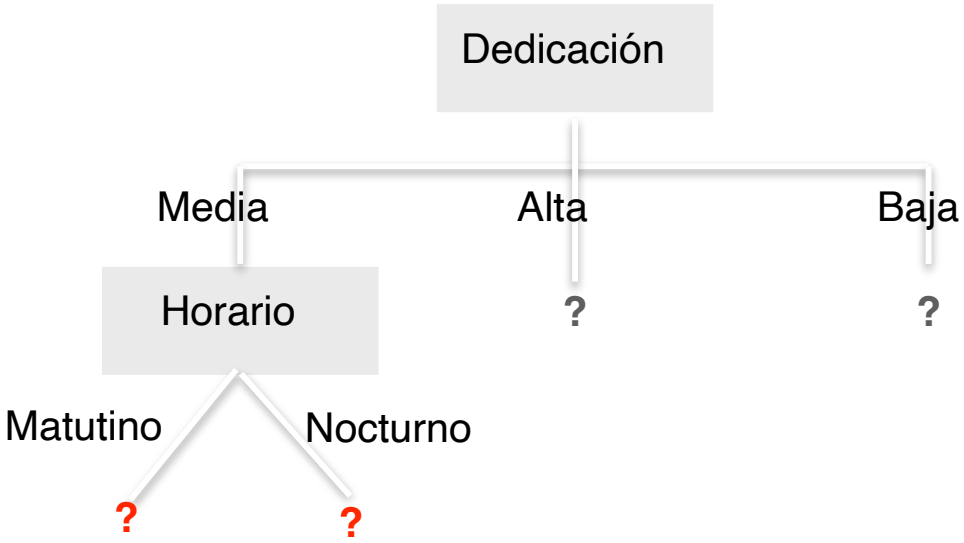
- Aplicamos recursivamente ID3 en la primera rama:

- Los ejemplos se reducen al 3 y 4 (dedicación=media), y el atributo dedicación ya no se puede seleccionar:

#Ej	~~Dedicación~~	Dificultad	Horario	Humedad	Humor Doc	Salva
3	~~Media~~	Alta	Nocturno	Media	Malo	Sí
4	~~Media~~	Alta	Matutino	Alta	Bueno	No

Algoritmo básico - Ejemplo

- Determinamos para este nodo que el mejor atributo es Horario.
- Creamos dos ramas, una por cada valor posible de Horario.



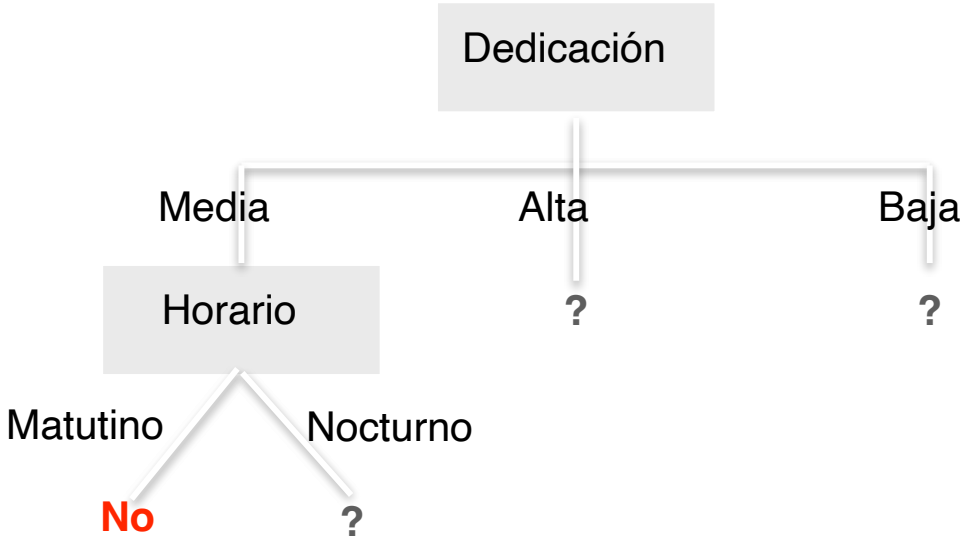
- Aplicamos recursivamente ID3 en cada rama de horario.

Algoritmo básico - Ejemplo

- En la primera rama considero los ejemplos con horario matutino:

#Ej	~~Dedicación~~	Dificultad	~~Horario~~	Humedad	Humor Doc	Salva
4	~~Media~~	Alta	~~Matutino~~	Alta	Bueno	No

- Como todos los ejemplos son negativos, es una hoja con valor “no”.

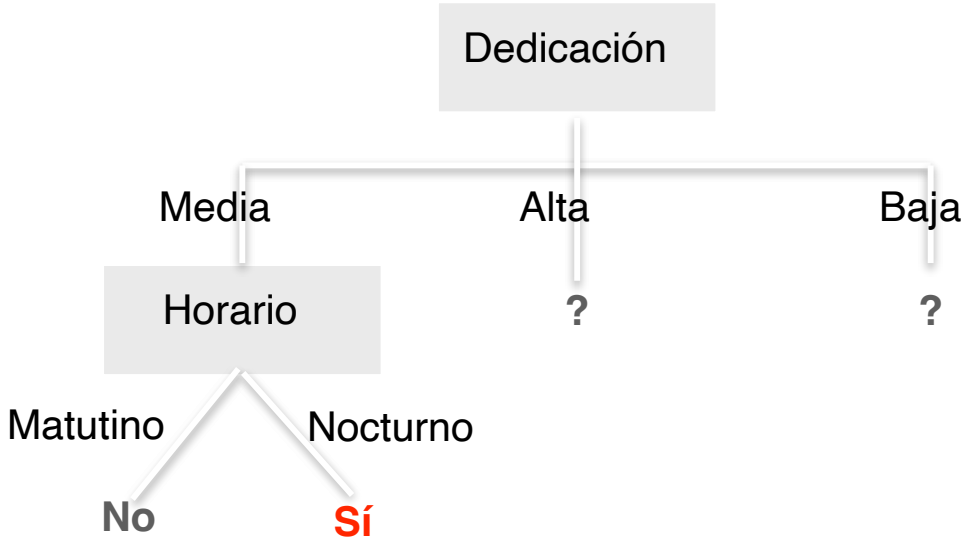


Algoritmo básico

- En la segunda, considero los ejemplos con horario nocturno:

#Ej	~~Dedicación~~	Dificultad	~~Horario~~	Humedad	Humor Doc	Salva
3	~~Media~~	Alta	~~Nocturno~~	Media	Malo	Sí

- Todos los ejemplos son positivos, creo una hoja con valor “sí”.



- ¿Qué pasaría si además existieran exámenes vespertinos?

Selección del Atributo

- Queda por determinar cómo seleccionamos el "mejor atributo" en el algoritmo anterior: vamos a utilizar una medida de cuánto un atributo separa a los ejemplos según la clasificación objetivo.

- **Entropía**: Sea S un conjunto de ejemplos, c una función objetivo que toma los valores $c_0 \ldots c_n$:

$$Entropia(S) = - \sum_{c_i \in C} p_{c_i} . log(p_{c_i})$$

donde p_{c_i} es la proporción de ejemplos $x \in S, f(x) = c_i$

- En particular para funciones booleanas:

$$Entropia(S) = - p_+ . log(p_+) - p_- . log(p_-)$$

Selección del Atributo

- La entropía determina la cantidad mínima de bits que en promedio se requiere para codificar los elementos de S.
- Es una manera de medir la heterogeneidad de los datos: cuanto más homogéneos, menor será la entropía.

- Algunos ejemplos:

$$Entropia([9+,0-]) = -\frac{9}{9}.log(\frac{9}{9}) = 0$$

$$Entropia([9+,9-]) = -2.\frac{9}{18}.log(\frac{9}{18}) = 1$$

$$Entropia([9+,5-]) = -\frac{9}{14}.log(\frac{9}{14}) - \frac{5}{14}.log(\frac{5}{14}) = 0{,}940$$

Selección del Atributo

- ¿Cómo utilizamos la entropía para elegir el atributo?
  - Definimos la Ganancia de Información de un atributo *A* sobre una muestra *S*:

$$Ganancia(S, A) = Entropia(S) - \sum_{v \in Val(A)} \frac{|S_v|}{|S|} . Entropia(S_v)$$

  - Buscamos medir la reducción en la entropía al particionar por el atributo A: los subconjuntos que se obtienen al elegir A son lo más homogéneos posible.
  - Ganancia es el número de bits que nos ahorramos si sabemos el valor del atributo A.

Selección del Atributo

#Ej	Dedicación	Dificultad	Horario	Humedad	Humor Doc	Salva
1	Alta	Alta	Nocturno	Media	Bueno	Sí
2	Baja	Media	Matutino	Alta	Malo	No
3	Media	Alta	Nocturno	Media	Malo	Sí
4	Media	Alta	Matutino	Alta	Bueno	No

- Calculemos la entropía de S:

  S=[2+, 2-]

$$Entropia(S) = -\frac{1}{2}.log(\frac{1}{2}) - \frac{1}{2}.log(\frac{1}{2}) = 1$$

- En nuestro ejemplo, ¿no convendría particionar primero por HumorDoc en lugar de utilizar Dedicación?

Selección del Atributo

- Calculemos la ganancia de particionar por dedicación:

$S_{Ded=Alta} \leftarrow [1+, 0-]$ $S_{Ded=Media} \leftarrow [1+, 1-]$ $S_{Ded=Baja} \leftarrow [0+, 1-]$

$$Gan(S, Ded) = 1 - \frac{1}{4} . E(S_{Ded=Alta}) - \frac{2}{4} . E(S_{Ded=Media}) - \frac{1}{4} . E(S_{Ded=Baja})$$

$$= 1 - \frac{1}{4}.0 - \frac{1}{2}.1 - \frac{1}{4}.0 = 0{,}5$$

- Y por humor docente:

$S_{HDoc=Bueno} \leftarrow [1+, 1-]$ $S_{HDoc=Malo} \leftarrow [1+, 1-]$

$$Gan(S, HDoc) = 1 - \frac{2}{4} . E(S_{HDoc=Bueno}) - \frac{2}{4} . E(S_{HDoc=Malo})$$

$$= 1 - \frac{1}{2} - \frac{1}{2} = 0$$

- HumorDoc no es una mejor elección que Dedicación.

Selección del Atributo

- Sin embargo, Horario sí es una mejor elección.

$S_{Hor=Mat} \leftarrow [0+, 2-]$ $S_{Hor=Noct} \leftarrow [2+, 0-]$

$$Gan(S, Hor) = 1 - \frac{2}{4} . E(S_{Hor=Mat}) - \frac{2}{4} . E(S_{Hor=Noct})$$

$$= 1 - \frac{1}{2}.0 - \frac{1}{2}.0 = 1$$

Búsqueda con ID3

- ID3 realiza una búsqueda en el espacio de árboles, del más sencillo al más complejo.

- El espacio de búsqueda es completo: como toda función discreta se puede representar con un árbol, estamos seguros de que el concepto objetivo está en el espacio de hipótesis.

- En todo momento ID3 mantiene una única hipótesis: no hay forma de saber cuáles ni cuántos árboles equivalentes hay en el espacio.

Búsqueda con ID3

- ID3 no realiza backtracking: una vez elegido un atributo, nunca se pregunta si esa fue la mejor opción. Esto puede conducir a una solución que es óptima local pero no globalmente

- Se utilizan todos los ejemplos en cada paso: esto evita que el algoritmo sea muy sensible al ruido, a diferencia de, por ejemplo, candidate-elimination.

Sesgo Inductivo

- ¿Cuál es el sesgo inductivo del algoritmo ID3?

- Elige un árbol que:
  - cubre los ejemplos.
  - es el más simple posible.
  - los atributos con más ganancia de información están más cerca de la raíz.

- A diferencia del candidate-elimination, el sesgo está en el algoritmo y no en el espacio.

Sesgo Inductivo

- Definiciones:

  Sesgo preferencial: el algoritmo prefiere ciertas hipótesis sobre otras.

  Sesgo restrictivo: se maneja un espacio de hipótesis incompleto.

- Por ejemplo, el ID3 tiene un sesgo preferencial, mientras que para el Candidate-elimination se establece uno restrictivo.

- Es deseable tener únicamente sesgo preferencial ya que asegura que el concepto objetivo está en H.

- El aprendizaje puede tener sesgos de ambos tipos.

Sesgo Inductivo

- ¿Por qué preferir hipótesis más simples?

  Navaja de Occam: Prefiera la hipótesis más simple que se ajuste a los datos.

  William de Ockham (S.XIV)

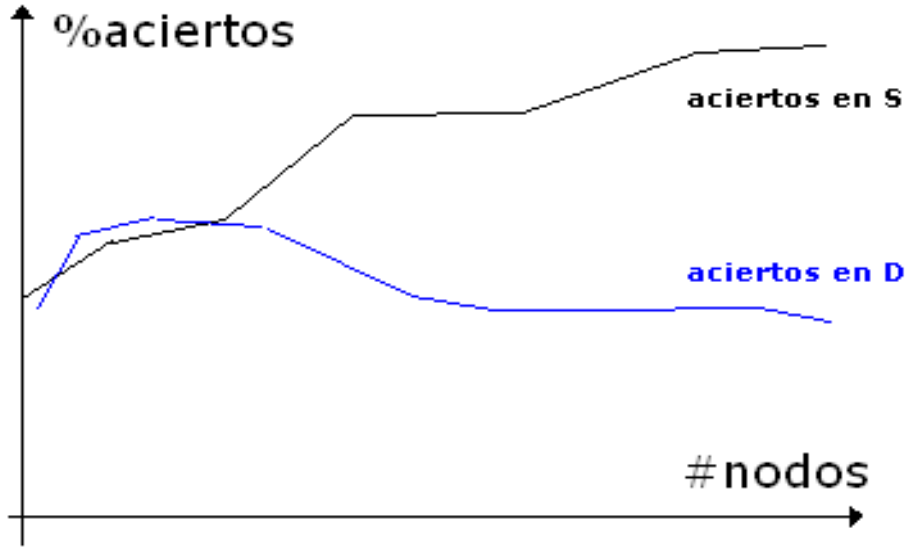
- Como hay menos hipótesis simples, es menos probable que una de ellas se ajuste a los datos “porque sí”.
- Sin embargo:
  - ¿Por qué no preferir hipótesis con exactamente 13 nodos?
  - Dos algoritmos con distinta representación interna pueden llegar a árboles completamente distintos.
- Más adelante veremos una justificación bayesiana a este principio...

Variaciones al ID3

- Algunas de las variaciones implementadas:
  - Evitar sobreajustes al conjunto de entrenamiento.
  - Manejar atributos con valores continuos.
  - Utilizar medidas alternativas para la elección del atributo.
  - Manejar ejemplos de entrenamiento con atributos incompletos.
  - Preguntar por múltiples atributos en un nodo.
  - Combinación lineal de atributos numéricos.

Evitar sobreajustes

- **Definición:** h ∈ H se sobreajusta a un conjunto de entrenamiento si existe h' ∈ H tal que h comete menos errores en el conjunto de entrenamiento que h', pero comete más errores en todo el dominio



Evitar sobreajustes

- El sobreajuste se puede dar por:
  - Error en los datos de entrenamiento: el árbol aprende a clasificar correctamente ejemplos erróneos, que lo inducen a clasificar erróneamente ejemplos del dominio.
  - Cantidad insuficiente de ejemplos: aparecen regularidades que no se cumplen en el dominio. En nuestro ejemplo: ¿es tan importante el horario en que se da un examen?

- ¿Cómo evitarlo?
  - Detener el crecimiento del árbol antes que se ajuste perfectamente a los datos.
  - Luego de obtenido el árbol, aplicar técnicas de podado.

Evitar sobreajustes

- ¿Cómo determinamos el tamaño óptimo del árbol?
  - Con un conjunto de ejemplos no utilizados durante el entrenamiento (conjunto de validación).
  - Aplicando técnicas estadísticas para evaluar si es conveniente agregar/quitar nodos.
  - Usando una medida de complejidad para codificar los datos y el tamaño del árbol, y parando cuando esa medida se minimiza.

Evitar sobreajustes

- Poda por reducción de error
  - Consideramos cada nodo, y lo sustituimos por la valoración más común de los ejemplos que engloba.
  - Esto se realiza mientras mejoran los aciertos sobre el conjunto de validación.
  - La principal desventaja es que se precisa un conjunto relativamente grande de ejemplos.

Evitar sobreajustes

- Poda de reglas [C4.5]
  - Transformamos el árbol en el conjunto de reglas que representa.
  - Quitamos condiciones de las reglas de forma que éstas mejoren su estimación en el conjunto de validación.
  - Ordenamos las reglas de acuerdo a su porcentaje de acierto y, ante nuevas instancias, las aplicamos en ese orden.

Evitar sobreajustes

- Ventajas de la poda de reglas
  - Como separamos las ramas, se puede cortar atributos “comunes” a varias ramas en algunas de ellas pero no en otras.
  - Al convertir el árbol perdemos el orden en que aparecían los atributos: es más fácil sacar conjunciones que quitar ramas enteras de un árbol.
  - Las reglas son legibles para un ser humano.

Atributos con valores continuos

- ¿Qué pasaría si la humedad viene dada por su porcentaje?

- Se pueden generar dinámicamente intervalos para estos atributos, para luego competir como atributos discretos.

		58,5%		87,5%	
Humedad	50%	67%	71%	82%	93%
Salva	Sí	No	No	No	Sí

- Los árboles de regresión permiten manejar los casos en donde la propia función objetivo es continua.

Alternativas en la selección del atributo

- Ganancia favorece a los atributos con más valores posibles. Por ejemplo: ¿qué pasa con un atributo de tipo fecha?
- Una medida alternativa que penaliza la distribución uniforme de valores es:

$$SplitInformation(S,A) = -\sum_{v \in Val(A)} \frac{|S_v|}{|S|} log(\frac{|S_v|}{|S|})$$

$$GainRatio(S,A) = \frac{Ganancia(S,A)}{SplitInformation(S,A)}$$

- Otra medida es la reducción de “impureza”:

$$Gini(S) = 1 - \sum p_i^2 = 1 - p_+^2 - p_-^2$$

$$ImpurityReduction(S,A) = Gini(S) - \sum_{v \in Val(A)} \frac{|S_v|}{|S|} Gini(S_v)$$

Ejemplos con atributos incompletos

- En algunos casos, los ejemplos de entrenamiento tienen algunos de sus atributos sin valores.
- Para suplirlos se utilizan los otros ejemplos que sí los tengan.
- Una estrategia es asignar el valor más común de ese atributo faltante en todo el conjunto de entrenamiento o en el subconjunto de ese nodo.
- El valor también puede asignarse de acuerdo a cierta probabilidad, usualmente estimada a partir de los otros ejemplos.

Otras extensiones

- Preguntar por múltiples atributos en un nodo
  - Su ventaja es que aumenta el conjunto que se procesa en cada rama.
  - Pero.... ¿cómo se determina cuál es la mejor partición de valores?

- Combinación lineal de atributos numéricos
  - Hasta ahora, los cortes son ortogonales a los ejes del espacio.
  - Se pueden hacer cortes oblicuos combinando varios atributos.
  - La desventaja es que los árboles resultantes ya no son tan “entendibles”.

- Múltiples clases en lugar de valores booleanos:
  - ¿se debe separar una clase de las otras? ¿o un grupo de ellas vs. el resto?
  - una solución es generar un árbol por cada valor posible de la salida. Para clasificar, se utilizan todos los árboles.
  - ¿qué sucede si más de un árbol lo clasifica como de su clase?

Random Forest

- Para atenuar la sensibilidad de un árbol al conjunto de entrenamiento, se pueden entrenar varios árboles (poco correlacionados) y ensamblarlos con votación.

- Existen varias alternativas, por ejemplo:
  - Random Forest
  - Extra Trees
  - AdaBoost

Random Forest

- Se utiliza Bagging:
  - Dado D, se crean **m** nuevos conjuntos D_i seleccionando al azar **k** elementos con repetición de forma uniforme.

    Por ejemplo, con D= [1, 2, 3, 4, 5] y k=|D|:

    D_1=[1, 2, 2, 5, 5] … D_m=[3, 3, 3, 4, 5],

  - Se construye un árbol de decisión por cada D_i

- Además, se modifica la selección de atributos del algoritmo de base: solo se toman en cuenta **p** atributos elegidos al azar.

- La clasificación se obtiene por votación.

Random Forest

- El algoritmo tiene tes hiperparámetros a determinar:
  - la cantidad de árboles **m** (normalmente, al menos 100)
  - la cantidad de ejemplos en el dataset **k** (puede ser |D|)
  - la cantidad de atributos **p** ($\sqrt{a}$, a/3, … con a=|A|)
- Estos parámetros se determinan con un conjunto de validación (!) o validación cruzada (lo veremos más adelante).
- El error del modelo se puede estimar evaluando cada elemento en aquellos árboles que no lo usaron para su construcción.

Extra Trees / AdaBoost

- Extra Trees:
  - utiliza todo el conjunto de entrenamiento.
  - al igual que Random Forest se seleccionan k atributos al azar
  - pero además se seleccionan aleatoriamente dos ramas con valores / punto de corte para cada atributo.

Extra Trees / AdaBoost

- Ada Boost:
  - Técnica general en donde se entrenan varios clasificadores débiles.
  - Por ejemplo, utilizar árboles de decisión realmente muy pequeños (Decision Stomps)
  - Los ejemplos tienen peso; los clasificadores se entrenan sobre aquellos más difíciles para los anteriores.
  - Los ejemplos se combinan con pesos