Aprendizaje Automatizado

Árboles de Clasificación

Árboles de Clasificación

- Estudiaremos un algoritmo para la creación del árbol.
- Selección de atributos comenzando en el nodo raíz.
- Proceso recursivo.

Árboles de Decisión (III)

- Ejemplo de un conjunto de entrenamiento.

Temperatura	Nivel de vibraciones	Horas de funcionamiento	Meses desde revisión	Probabilidad de fallo
ALTA	ALTO	< 1000	> 1 MES	fallará
BAJA	BAJO	< 1000	< 1 MES	no fallará
ALTA	BAJO	>1000	> 1 MES	no fallará
ALTA	BAJO	< 1000	> 1 MES	no fallará
BAJA	ALTO	< 1000	> 1 MES	no fallará
BAJA	ALTO	>1000	> 1 MES	fallará
ALTA	ALTO	< 1000	< 1 MES	fallará

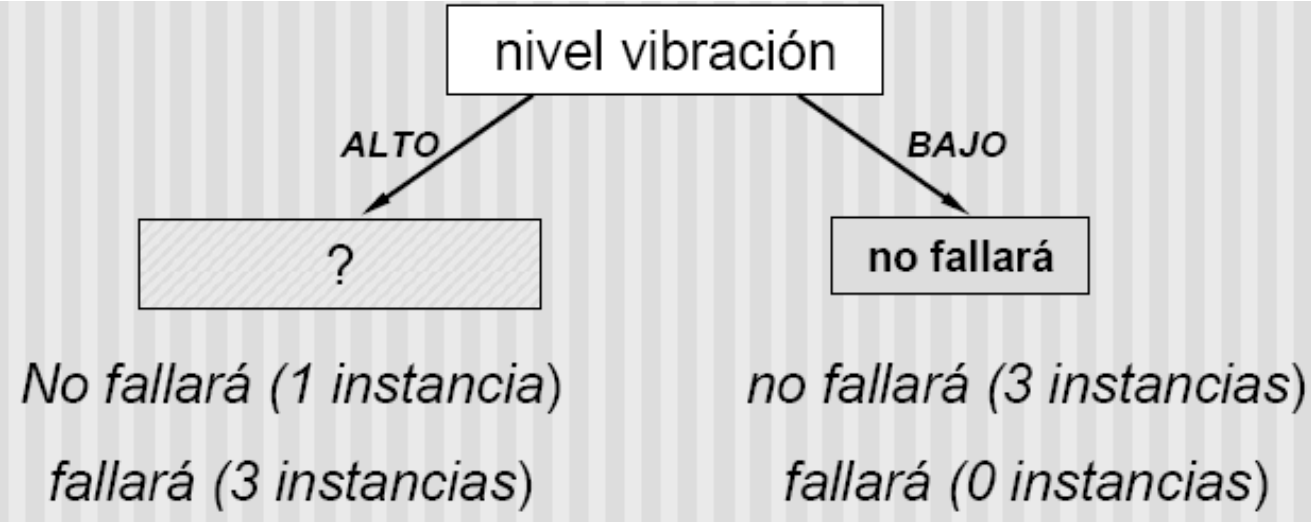
Árboles de Decisión (IV)

Crearemos un árbol a partir de los ejemplos de entrenamiento anteriores. ¿Qué atributo elegir para el primer nodo?

ATRIBUTO	VALORES	CLASE	
		fallará	*no fallará*
Temperatura	Alto	2	2
	Bajo	1	2
Nivel de vibraciones	Alto	3	1
	Bajo	0	3
Horas defuncionamiento	< 1000	2	3
	>1000	1	1
Meses desde revisión	> 1 mes	2	3
	< 1 mes	1	1

Árboles de decisión (V)

- Árbol construido hasta el momento:



-
-
- Qué atributo usamos en el siguiente nivel del árbol (rama izquierda)?

Árboles de Clasificación

- Entrada: Objetos caracterizables mediante propiedades.
- Salida:
  - En árboles de decisión: una decisión (sí o no).
  - En árboles de clasificación: una clase.
- Conjunto de reglas.

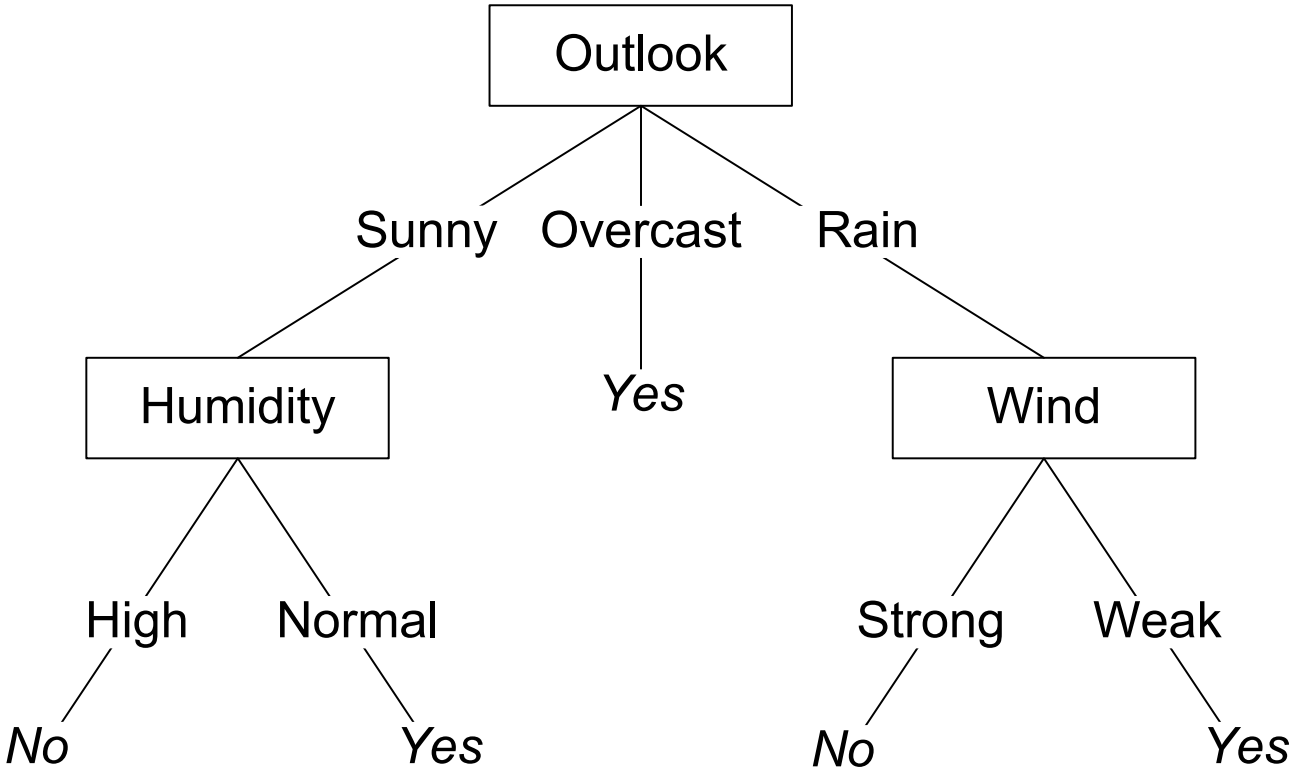
Árboles de Clasificación

- Se clasifican las instancias desde la raíz hacia las hojas, las cuales proveen la clasificación.
- Cada nodo especifica el test de algún atributo.
- Ejemplo: Si

(Outlook = Sunny, Humedity = High, Temperature = Hot, Wind = Strong)

Juego al tenis?

Play Tennis


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

Play Tennis

- Disyunción de conjunciones:

(Outlook = Sunny **And** Humidity = Normal)
Or (Outlook = Overcast)
Or (Outlook = Rain **And** Wind = Weak)

Play Tennis

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Problemas Apropiados

- Las instancias pueden ser representadas por pares (atributo, valor).
- La función objetivo tiene valores discretos (o pueden ser discretizados).
- Pueden ser requeridas descripciones en forma de disjunción.
- Posiblemente existen errores en los datos de entrenamiento (robustos al ruido).
- Posiblemente falta información en algunos de los datos de entrenamiento.

Algoritmo básico para obtener un árbol de decisión

- Encontrar el árbol mas chico es un problema NP
- complete (Quinlan 1986), por lo cual estamos
- forzados a usar algún algoritmo local de búsqueda
- para encontrar soluciones razonables.

Algoritmo básico para obtener un árbol de decisión

- Un árbol puede ser "aprendido" mediante el fraccionamiento del conjunto inicial en subconjuntos basados en una prueba de valor de atributo.
- Este proceso se repite en cada subconjunto derivado de una manera recursiva llamada particionamiento recursivo.
- La recursividad termina cuando el subconjunto en un nodo tiene todo el mismo valor de la variable objetivo, o cuando la partición ya no agrega valor a las predicciones (condición de parada).

Algoritmo básico para obtener un árbol de decisión (I)

- Búsqueda exhaustiva, en profundidad (de arriba hacia abajo), a través del espacio de posibles árboles de decisión (ID3 y C4.5).
- Raíz: el atributo que mejor clasifica los datos

Cuál atributo es el mejor clasificador?

$\Rightarrow$ respuesta basada en la **ganancia de información.**

Algoritmo básico para obtener un árbol de decisión (II)

- Hay **ganancia de información** cuando la división envía instancias con clases distintas a los distintos nodos.
- El atributo que permite obtener mayor ganancia de información es el seleccionado para dividir el nodo.

Algoritmo básico para obtener un árbol de decisión (III)

- El algoritmo ID3 se aplica a atributos discretos.
  - En cada nodo queda seleccionado un atributo y un valor (ej. temperatura = alta).
- El algoritmo C4.5 además se puede aplicar a atributos continuos.
  - En cada nodo queda seleccionado un atributo y un umbral para realizar la división (ej. temperatura > 26).

Algoritmo básico para obtener un árbol de decisión (IV)

- ID3 nunca produce árboles demasiado grandes.
- C4.5 sí, pues puede repetir atributos (temp < 26, temp > 24, temp < 25, etc).
- Un árbol demasiado grande puede producir sobreajuste (*overfitting*).
- Es necesario podar los árboles (*pruning*).

Algoritmos: ID3 (Interactive Dichotomizer Version 3)

- Entropía

Es la medida de la incertidumbre que hay en un sistema. Es decir, ante una determinada situación, la Probabilidad de que ocurra cada uno de los posibles resultados.

Algoritmos: ID3 (Interactive Dichotomizer Version 3)

- Entropía

  $Entropía(S) \equiv - p\oplus \log_2 p\oplus - p\ominus \log_2 p\ominus$

S: conjunto de datos actual.
$p\oplus$ = proporción de ejemplos positivos.
$p\ominus$ = proporción de ejemplos negativos.

Por ejemplo, en el conjunto de datos Play Tennis
$p\oplus = 9/14$, $p\ominus = 5/14$ y $E(S) = 0.940$
En general: $Entropía(S) = - \sum_{i=1,c} pi \log_2 pi$

Algoritmos: ID3 (Interactive Dichotomizer Version 3)

- Por ejemplo:

Si S1 es el subconjunto de S en el cual

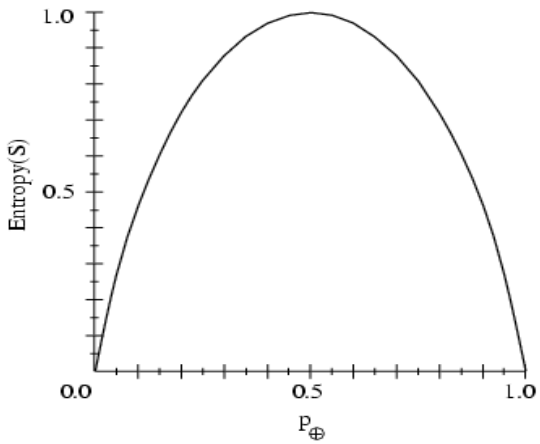
Humedity = High

Entonces:

- $p\oplus = 3/7$
- $p\ominus = 4/7$
- Entropía(S1) = -3/7 log2 3/7 - 4/7 log2 4/7 = 0.985

Entropía y proporción de positivos

Entropy



- S is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\ominus}$ is the proportion of negative examples in S
- Entropy measures the impurity of S

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Ganancia de información

- Mide la reducción esperada de entropía sabiendo el valor del atributo A

$$Gain(S,A) \equiv Entropía(S) - \sum_{v \in Valores(A)} (|Sv|/|S|) Entropía(Sv)$$

Valores(A): Conjunto de posibles valores del atributo A
Sv: Subconjunto de S en el cual el atributo A tiene el valor *v*

Ej: Gain(S, Humedad) = 0.940 - (7/14) 0.985 - (7/14) 0.592

Ent (Sha)

Ent (Shn)

proporción de humedad alta

proporción de humedad normal

Otras medidas para decidir...

- Ganancia de información

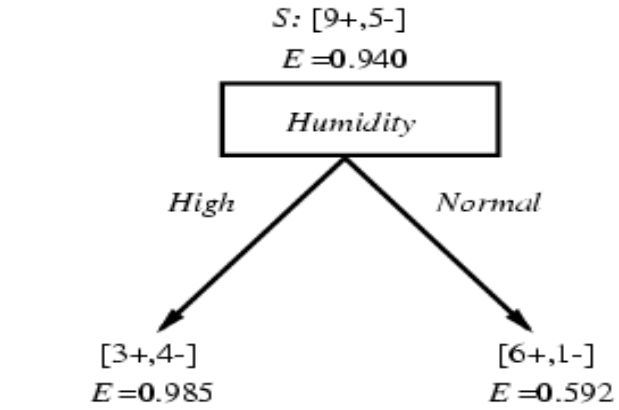
$$Gain(S,A) \equiv Entropía(S) - \sum_{v \in Valores(A)} (|Sv|/|S|)Entropía(Sv)$$

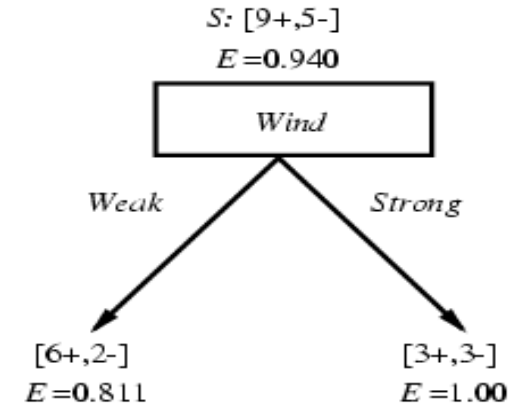
- Impureza de Ginni
-
- Reducción de la varianza
-

Play Tennis

Selecting the Next Attribute

Which attribute is the best classifier?





Gain (S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

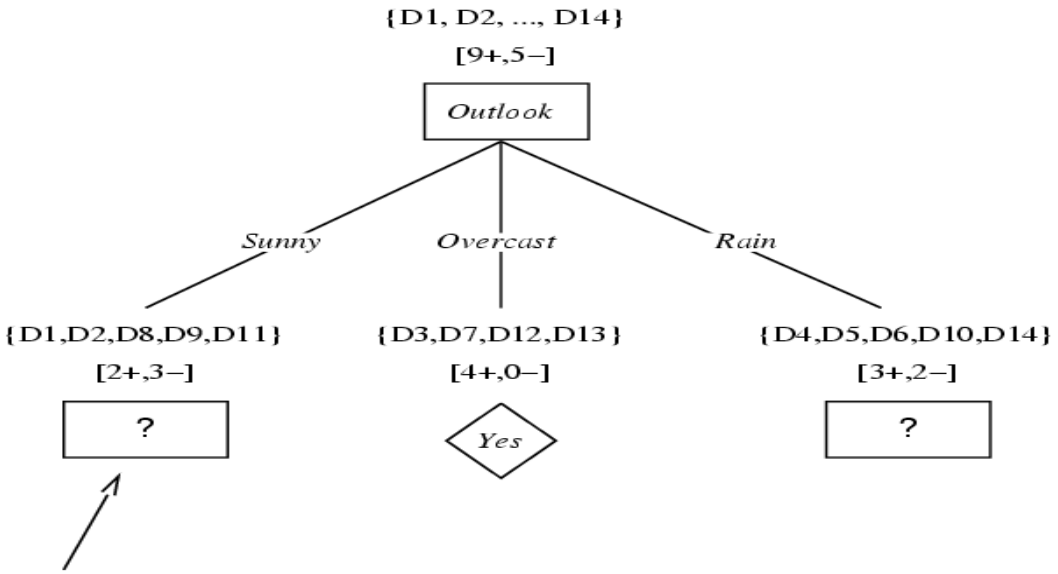
Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

Play Tennis

- Gain(*S*,Outlook) = 0.246
- Gain(S,Humidity) = 0.151
- Gain(S,Wind) = 0.048
- Gain(S,Temperature) = 0.029

$\Rightarrow$ Outlook es el atributo del nodo raíz.

Play Tennis



Which attribute should be tested here?

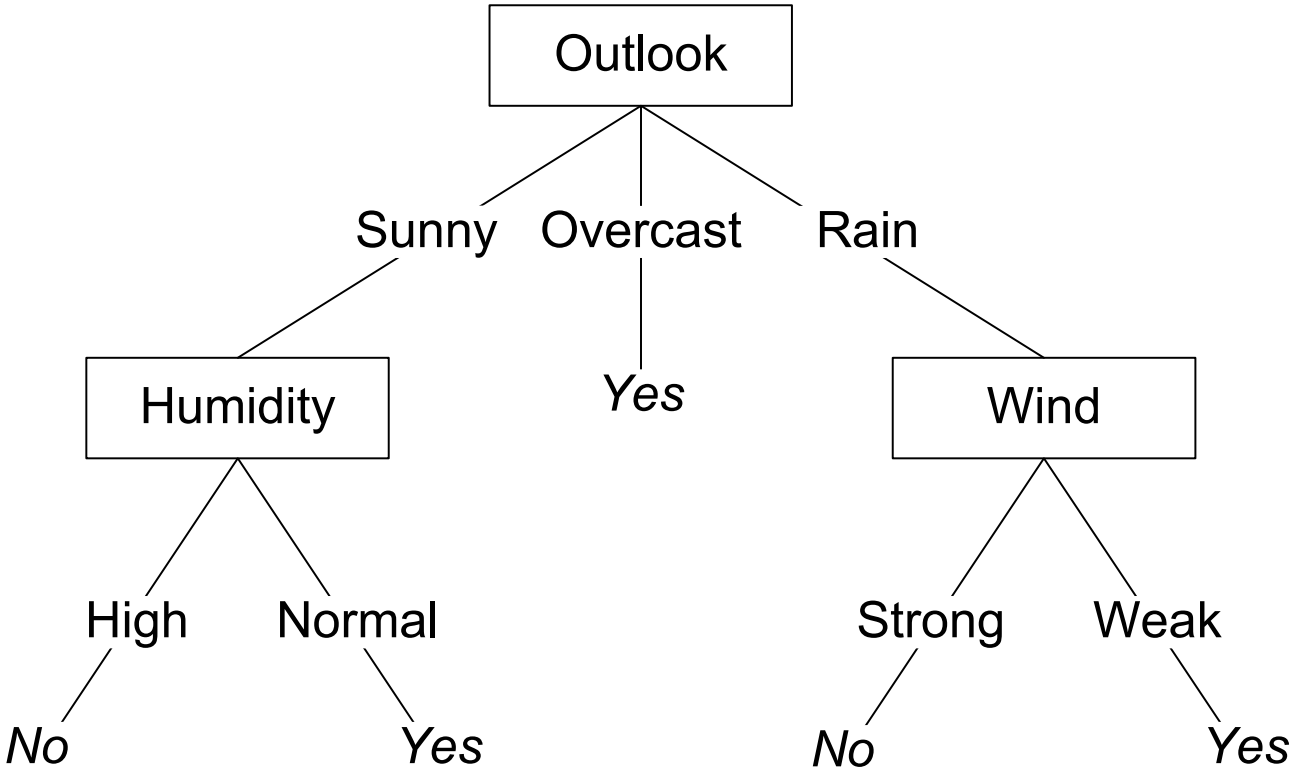
S_{sunny} = {D1,D2,D8,D9,D11}

$Gain\,(S_{sunny}, Humidity) = .970 - (3/5)\,0.0 - (2/5)\,0.0 = .970$

$Gain\,(S_{sunny}, Temperature) = .970 - (2/5)\,0.0 - (2/5)\,1.0 - (1/5)\,0.0 = .570$

$Gain\,(S_{sunny}, Wind) = .970 - (2/5)\,1.0 - (3/5)\,.918 = .019$

Play Tennis


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

Sobreentrenamiento

Porque —overfitting?

Un modelo puede ser más complejo
de lo que la función objetivo (conceptual) puede ser, cuando **trata de satisfacer datos ruidosos** también.

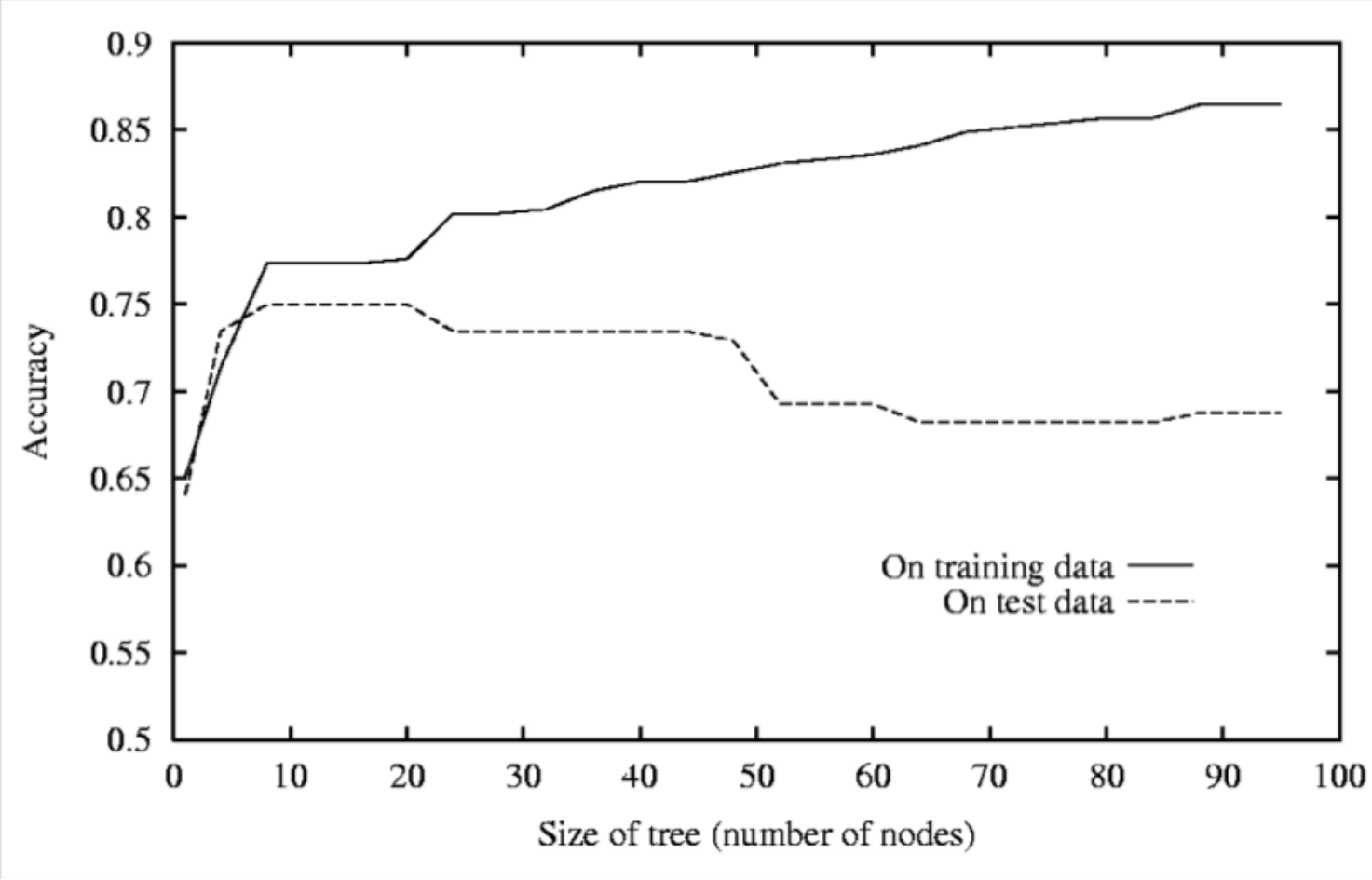
Por Ej: instancia etiquetada incorrectamente como Negativo: (Sunny; Hot; Normal; Strong; PlayTennis = No)

Sobreentrenamiento

Definición de overfitting

En una hipotesis (modelo) se dice que existe sobre-entrenamiento si existe alguna otra Hipotesis que tiene **mayor** error sobre los datos de entrenamiento pero **menor** error sobre todos los datos.

Sobreentrenamiento



Sobreentrenamiento

- Se debe evitar el sobreentrenamiento
  - Parar el crecimiento del árbol.
  - Postprocesamiento del árbol (poda)
  -

Cómo?

  - Usar un conjunto de ejemplos de validación
  - Usar estadísticas

Statistics Toolbox - Matlab

Cómo crear un árbol de clasificación para los datos del conjunto `ionosphere`:

```
load ionosphere % contains X and Y variables
ctree = fitctree(X,Y)
```

```
ctree =

  ClassificationTree
            PredictorNames: {1x34 cell}
              ResponseName: 'Y'
                ClassNames: {'b'  'g'}
            ScoreTransform: 'none'
    CategoricalPredictors: []
             NumObservations: 351
```

Visualización de un árbol

`view(tree)` retorna una descripción en modo texto

`view(tree,'mode','graph')` retorna una descripción gráfica

Ejemplos de visualización

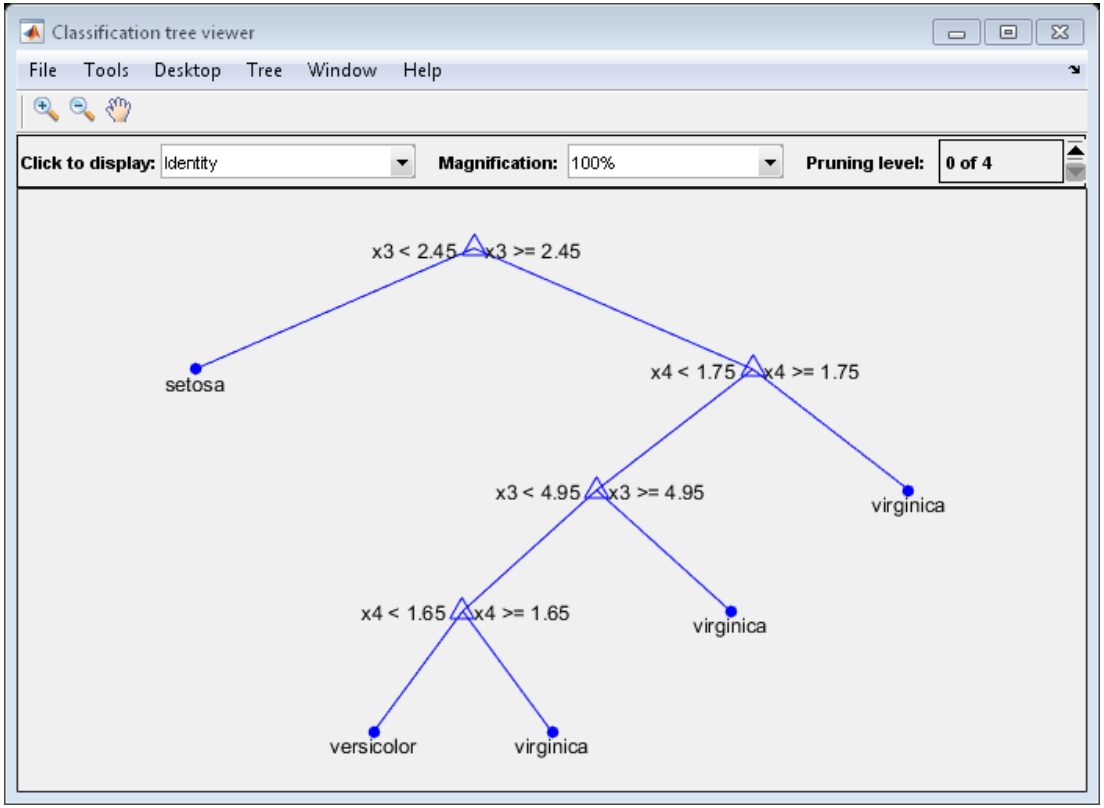
```
load fisheriris % load the sample data
ctree = fitctree(meas,species); % create
classification tree

view(ctree) % text description
Decision tree for classification
1  if x3<2.45 then node 2 elseif x3>=2.45 then node
3 else setosa
2  class = setosa
3  if x4<1.75 then node 4 elseif x4>=1.75 then node
5 else versicolor
```

Ejemplos de Visualización

```
4  if x3<4.95 then node 6 elseif x3>=4.95
then node 7 else versicolor
5  class = virginica
6  if x4<1.65 then node 8 elseif x4>=1.65
then node 9 else versicolor
7  class = virginica
8  class = versicolor
9  class = virginica
```

Ejemplos de Visualización

```
view(ctree,'mode','graph') % graphic description
```



Cómo se crean árboles con `fitctree`

La función `fitctree` realiza los siguientes pasos para crear árboles de decisión:

1. Comienza con todos los datos de entrada y examina todos los posibles splits binarios sobre cada predictor (variables de entrada).
2. Selecciona el mejor split de acuerdo a un criterio de optimización considerando las restricciones `MinLeaf` y `MinParent`.
3. Realiza el split.
4. Repite recursivamente para los dos nodos hijos.

Regla de parada

Se detiene el crecimiento del árbol cuando ocurre alguna de las siguientes condiciones:

El nodo es puro (contiene sólo observaciones de una clase).
Hay menos de `MinParent` observaciones en ese nodo.
Cualquier split aplicado sobre ese nodo produciría hijos con menos de `MinLeaf` observaciones.

Criterios de optimización

El par valor de '`SplitCriterion`' se puede configurar con una de las tres siguientes medidas:

'gdi' (índice de diversidad de Gini, the default)
'twoing'
'deviance' (Entropía)

Clasificación

```
Ynew = predict(tree,Xnew);
o
Ynew = tree(Xnew);
```

`Xnew` debe tener el mismo número de columnas que el conjunto original de datos X

Mejorando los árboles

1. Error de Entrenamiento (Resubstitution)
2. Validación Cruzada (Cross Validation)
3. Control de profundidad (Leafiness)
4. Poda (Pruning)

1. Error de entrenamiento

El error de entrenamiento (Resubstitution error) es la diferencia entre la respuesta en los datos de entrenamiento y las predicciones que hace el árbol sobre esos mismos datos.

El error de entrenamiento es optimista con respecto a la clasificación de nuevos datos.

Error de entrenamiento

```
load fisheriris
ctree = fitctree(meas,species);
resuberror = resubLoss(ctree)
resuberror =

    0.0200
```

2 - Poda (Prunning)

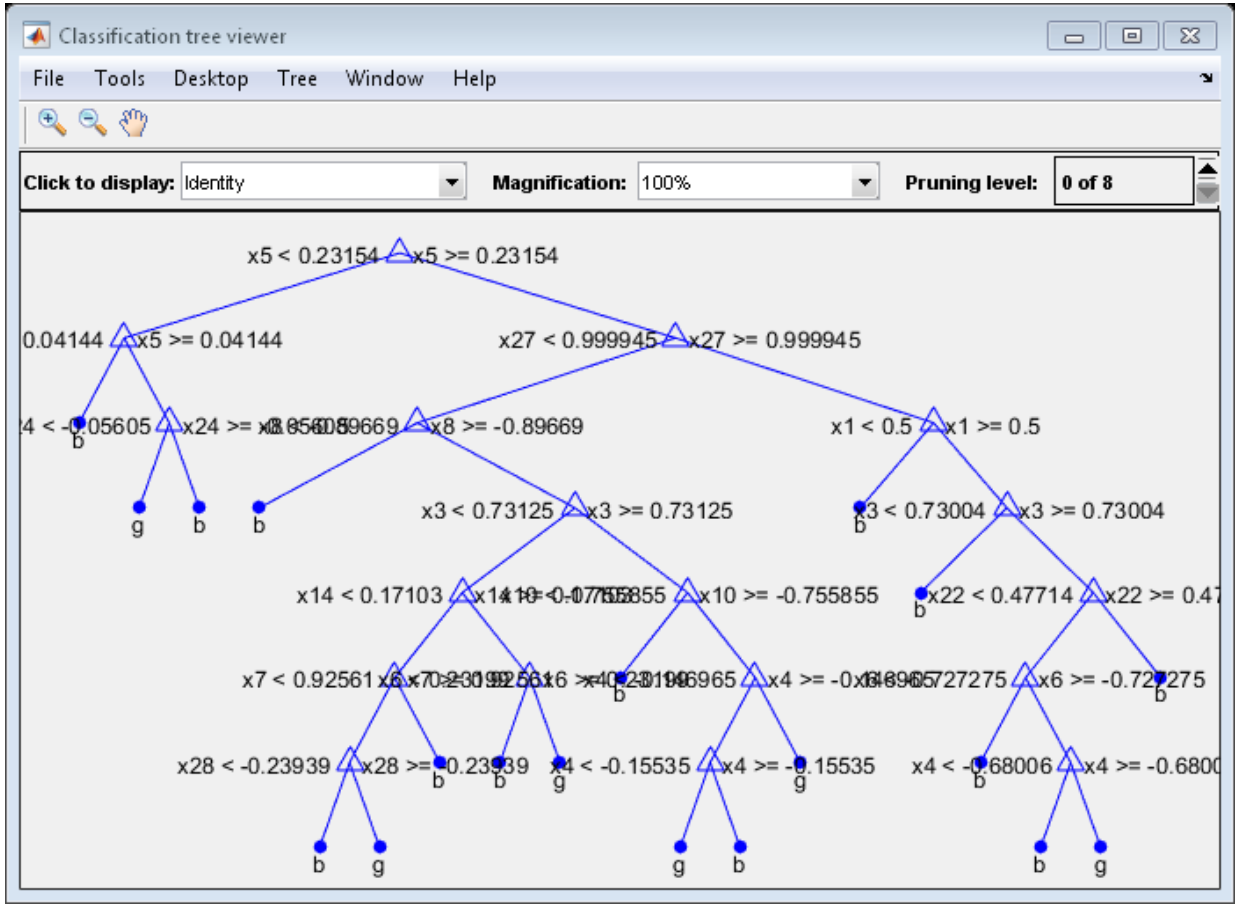
La poda optimiza el árbol modificando la frondosidad (leafiness) juntando dos hojas en una.

Se realiza una poda:

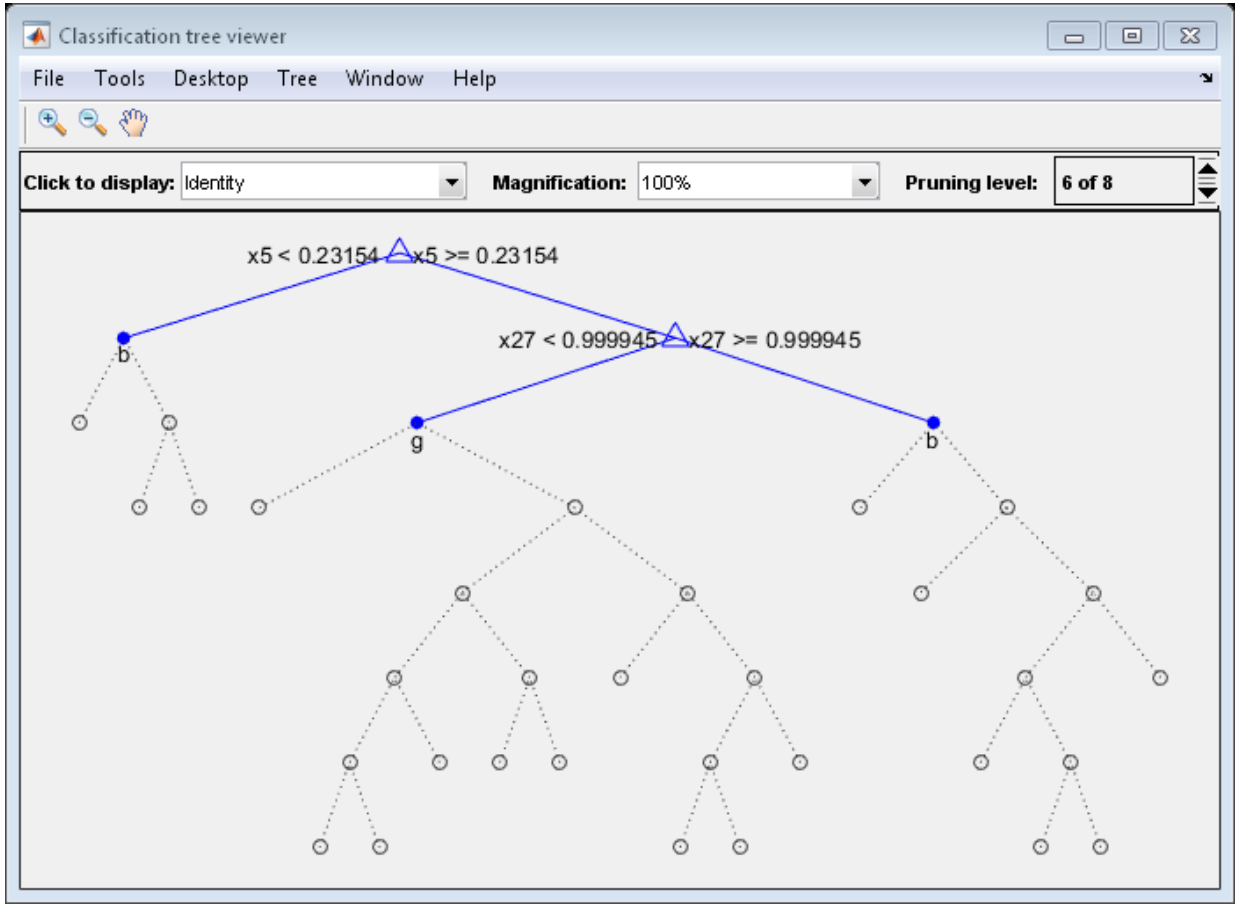
- con la función `prune`.
- interactivamente con la visualización en modo gráfico

```
view(tree,'mode','graph')
```

Ejemplo de poda

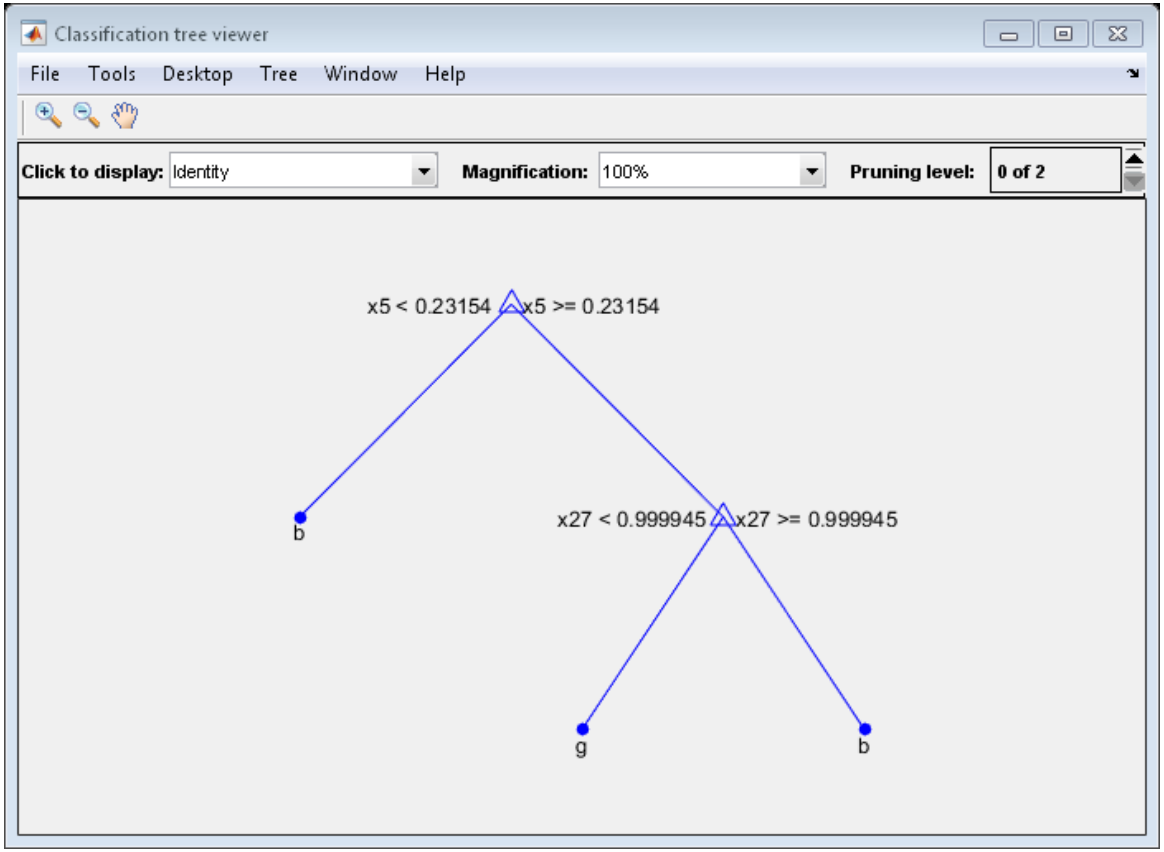


Ejemplo de poda



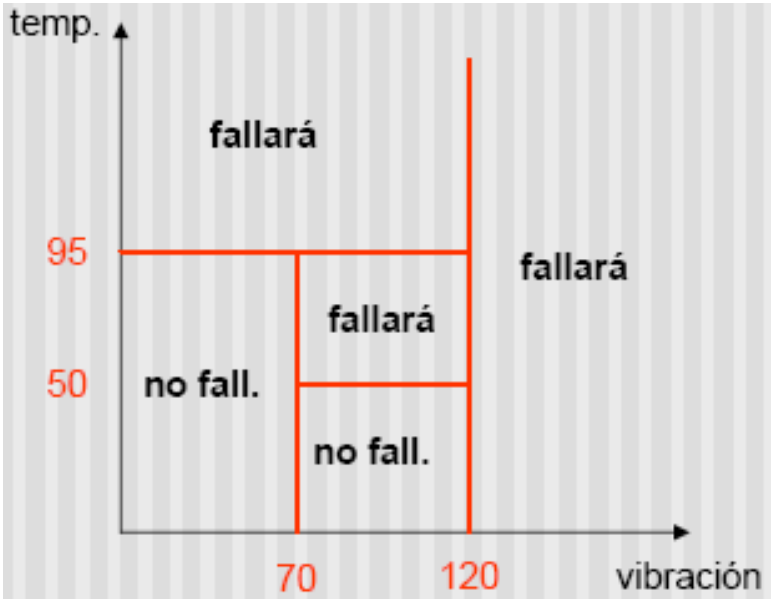
```
[~,~,~,bestlevel] = cvLoss(tree,...

'SubTrees','All','TreeSize','min')
bestlevel =
     6

view(tree,'Mode','Graph','Prune',6)
```

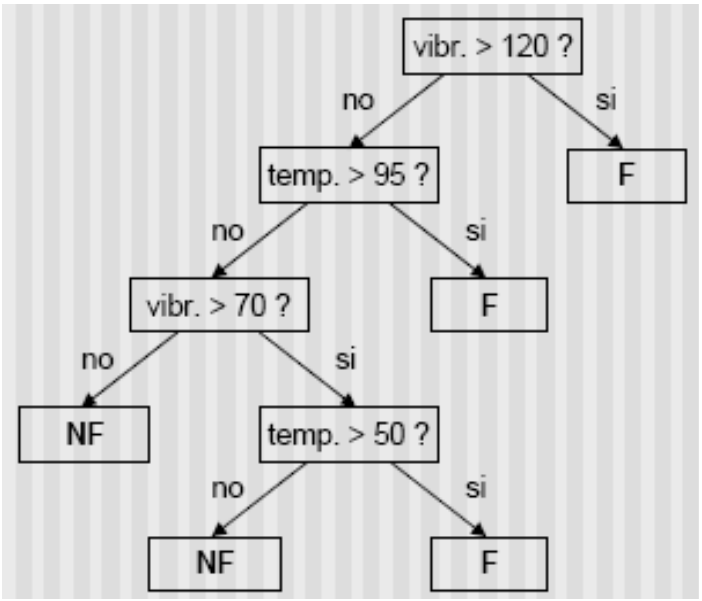
Ejemplo de poda



Árboles de Decisión - Resumen (I)

- Capacidad de representación:
  - No muy elevada, las superficies de decisión son siempre perpendiculares a los ejes:





Árboles de Decisión - Resumen (II)

- Legibilidad: muy alta. Uno de los mejores modelos en este sentido.
- Tiempo de cómputo on-line: muy rápido. Clasificar un nuevo ejemplo es recorrer el árbol hasta alcanzar un nodo hoja.
- Tiempo de cómputo off-line: rápido. Los algoritmos son simples.

Árboles de Decisión - Resumen (III)

- Parámetros a ajustar: nivel de confianza para la poda (el valor por defecto 25% da buenos resultados).
- Robustez ante instancias de entrenamiento ruidosas: robusto.
- Sobreentrenamiento o sobreajuste: Se controla a través de una poda.

Matlab - Statistics Toolbox (Antes)

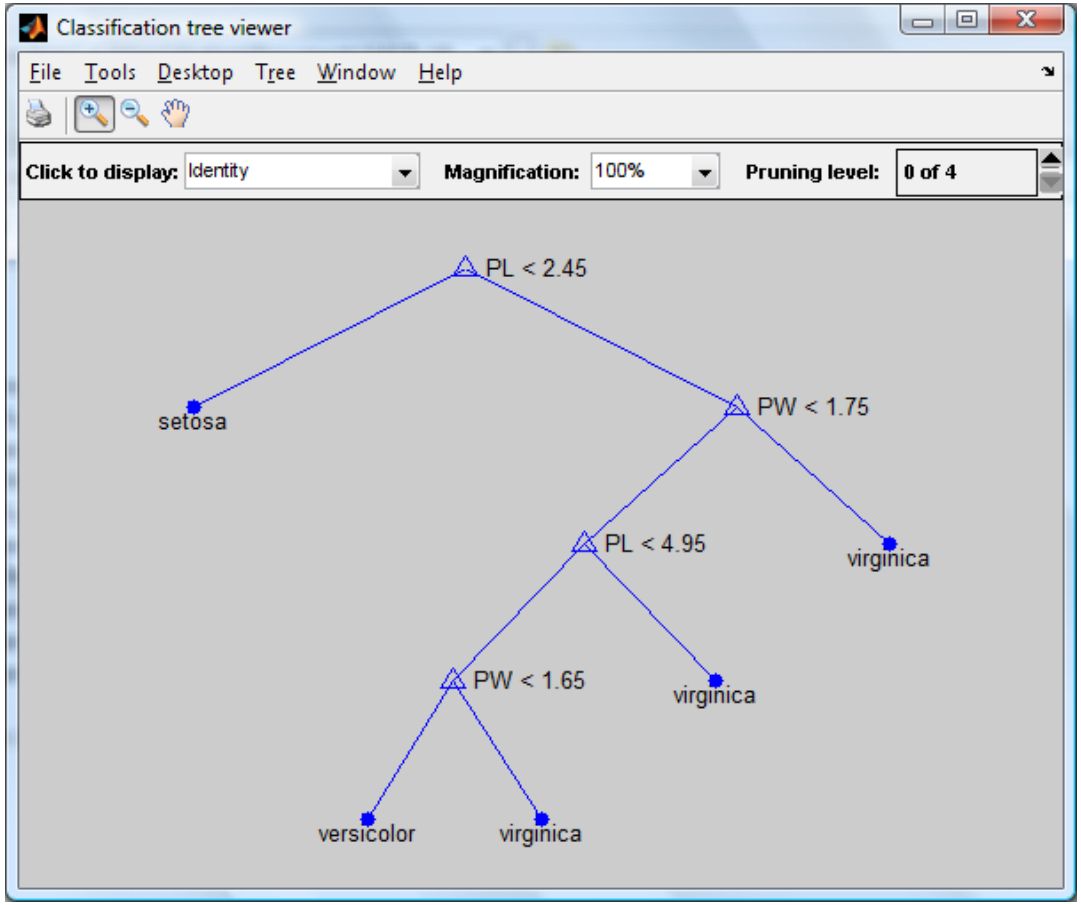
- La clase `@classregtree` está diseñada para manipular árboles de regresión y árboles de decisión (CART).
- Ejemplo:

```
>> load fisheriris;
>> t = classregtree(datos, especies,
'names', {'SL' 'SW' 'PL' 'PW'})
```

`t = classregtree(X,y)` crea un árbol de decisión `t` para una respuesta predicha `y` en función de los predictores en las columnas de `X`.

Matlab - Statistics Toolbox (Antes)

```
>> view(t)
```



Matlab - Statistics Toolbox (Antes)

- `t = classregtree(X,y,param1,val1,param2,val2)`
- 'method' — Puede ser 'classification' (por defecto si `y` es texto o una variable categórica) o 'regression' (por defecto si `y` es numérica).
- 'names' — Un arreglo tipo `cell` de nombres para los atributos, en el orden en el cual aparecen en `X` .

Bibliografía

- Machine Learning - Tom Mitchell – McGrawHill
- Statictics Toolbox User’s Guide (http://www.mathworks.com/access/helpdesk/help/pdf_doc/stats/stats.pdf).
- Curso de doctorado "Aprendizaje Automatizado y Data Mining" Grupo de Ingeniería de Sistemas y Automática (Universidad Miguel Hernández)
- http://isa.umh.es/asignaturas/aprendizaje/index.html