Decision trees

Artificial Intelligence **3rd year INF**

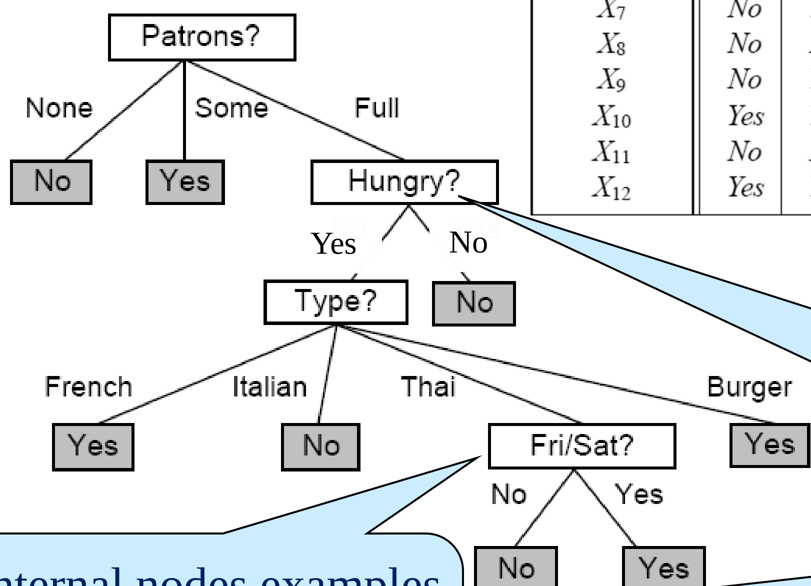
Escuela Politécnica Superior

Decision tree

Training data

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*\$\$\$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*\$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*\$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*\$*	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*\$\$\$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*\$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*\$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$\$\$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*\$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$*	*No*	*No*	*Burger*	*30–60*	*Yes*

Learned tree:



Instances for which *Patrons = full* are associated to this node

At internal nodes examples are separated according to a test on an attribute

Instances that reach this leaf node are classified as *WillWait = Yes*

How does it work?

- A decision tree is a **hierarchical questionnaire** that **splits the data** according to a **sequence of tests on** their **attributes**.
- **Each example**, when processed by the tree, follows a unique **path** from the **root node to** the corresponding **leaf** according to the results of the **tests on the attributes** performed at each of the **intermediate internal nodes.**
- The **class associated to a node** corresponds to the **majority label of the training instances** assigned to that node.
- The **tests** at the internal nodes are determined by **maximizing a quantity** (e.g. the information gain) that favors a **clearer separation of the classes** in the children of such nodes.
- The **class label prediction** for an example takes place at the corresponding **leaf node**.

Learning algorithm

function DECISION-TREE-LEARNING(*examples*, *attributes*, *default*) **returns** a decision tree
    inputs: *examples*, set of examples
            attributes, set of attributes
            default, default value for the goal predicate

    if *examples* is empty **then return** *default*
    else if all *examples* have the same classification **then return** the classification
    else if *attributes* is empty **then return** MAJORITY-VALUE(*examples*)
    else
        best $\leftarrow$ CHOOSE-ATTRIBUTE(*attributes*, *examples*)
        tree $\leftarrow$ a new decision tree with root test *best*
        for each value v_i of *best* **do**
            $examples_i \leftarrow$ {elements of *examples* with $best = v_i$}
            subtree $\leftarrow$ DECISION-TREE-LEARNING($examples_i$, *attributes* $-$ *best*,
                                MAJORITY-VALUE(*examples*))
            add a branch to *tree* with label v_i and subtree *subtree*
        end
        return *tree*

Simplified version of Quinlan's ID3 (1986)

The **best attribute** is the one that provides the largest amount of **information** on the class label.

Recursion

Measuring information: Binary Entropy

"r.v." means "random variable"

"w.p." means "with probability"

Consider the binary r.v. $X = \begin{cases} x_1 & w.p.\ p \\ x_0 & w.p.\ q = 1 - p \end{cases}$

Entropy of r.v. X

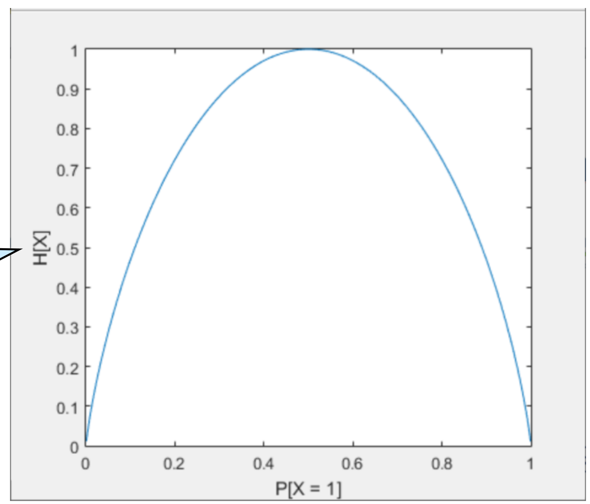
$$H(X) = H_b(p) = -p\ \log_2 p - q \log_2 q \text{ [in bits]}$$

$p = P[X = x_1]$
$q = P[X = x_0]$

$0 \le p, q \ \le 1$
$p + q = 1$

A common unit to measure information

$0 \le H(X) \le 1$
Information cannot be negative.



Another, less common measurement unit for information

Also: $H(X) = -p \log p - q \log q$ **[in nats]**

Natural logarithm

Message encoding: Entropy rate

$x_1 = ‘a’$
$x_0 = ‘b’$

- **Information contents of messages with random {‘a’,’b’}:**
  - **Message 1: “aaaaaaaaaaaaaaaaaaaa”**

$$\hat{p} = 1; \hat{q} = 0$$

Estimate the probabilities from the frequencies of the symbols in the message

$H_b(p)$ is minimum for p $= 1, q = 0$
p $= 0, q = 1$

$$H(X) = 0\ bits$$

Send message: “20 a’s”
When length of the actual message is very large, the average amount of information per symbol that needs to be transmitted approaches 0 bits

  - **Message 2: “aaababaaabaaabaaaaab”**

$$\hat{p} = \frac{3}{4}; \hat{q} = \frac{1}{4}$$

$$H(X) = 0.81\ bits$$

  - **Message 3: “abbababbaababbaaabba”**

$H_b(p)$ is maximum for p $= q = \frac{1}{2}$

$$\hat{p} = \frac{1}{2}; \hat{q} = \frac{1}{2}$$

$$H(X) = 1\ bit$$

All symbols need to be transmitted: On average 1 bit per symbol

Entropy for a discrete r.v.

- **Consider the discrete r.v.**

$\mathcal{X}$ is the space in which the random variable takes values

$$X \in \mathcal{X} = \{x_1, x_2, \ldots, x_{|\mathcal{X}|}\}$$

Random variable X can take $|\mathcal{X}|$ different values

Maximum: $H(X) = \log_2 |\mathcal{X}|$
for $P[X = x_i] = \frac{1}{|\mathcal{X}|}$, $i = 1,2, \ldots, |\mathcal{X}|$

$$H(X) = -\sum_{i=1}^{|\mathcal{X}|} P[X = x_i] \log_2 P[X = x_i] \qquad \text{[in bits]}$$

If a sender wants to transmit values (sampled independently) of the r.v. X to a receiver, the entropy measures the average amount of bits per symbol of the minimal length message

Message encoding: Entropy rate

$x_1 =$ 'a'
$x_2 =$ 'b'
$x_3 =$ 'c'
$x_4 =$ 'd'

- **Information contents of messages with {'a','b','c','d'}:**
  - **Message 1:** $p('a') = p('b') = p('c') = p('d') = 1/2$

$H(X) = 2\ bits$

Symbol	'a'	'b'	'c'	'd'
Endoding	00	01	10	11

Fixed length code

  - **Message 2:** $p('a') = \frac{1}{2}$; $p('b') = p('c') = \frac{1}{4}$; $p('d') = 0$

$H(X) = 1.5\ bits$

Symbol	'a'	'b'	'c'	'd'
Endoding	0	10	11	-

Variable length prefix code

- Average # bits per symbol of encoded message:

$p('a') \times 1\ bit + \ p('b') \times 2\ bit + \ p('c') \times 2\ bits + \ p('d') \times 0\ bits = 1.5\ bits$

An alternative: Gini impurity

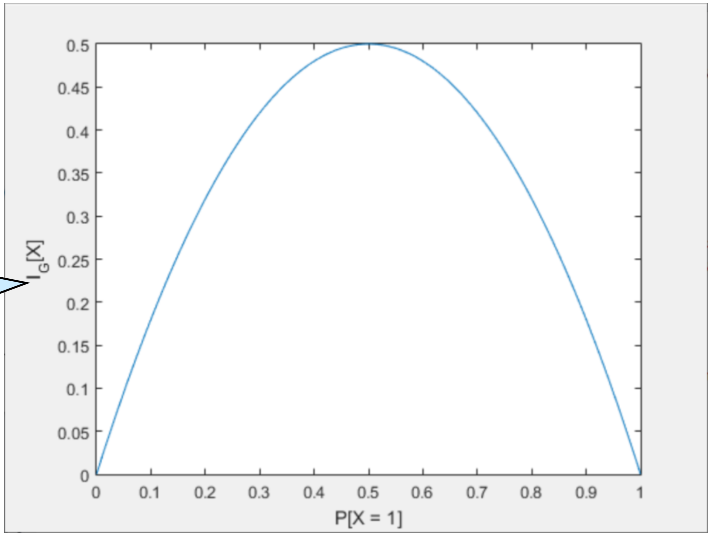
- **Consider the discrete r.v.** $X \in \mathcal{X} = \{x_1, x_2, \dots, x_{|\mathcal{X}|}\}$

$$I_G(X) = 1 - \sum_{i=1}^{|\mathcal{X}|} (P[X = x_i])^2$$

Used to determine splits in CART decision trees (Breiman et al. 1984)

- **Gini Impurity for a binary r.v.** $I_G(X) = 1 - p^2 - q^2$

Similar shape to the binary entropy



Conditional entropy

- **Consider the discrete r.v.'s:** $X \in \mathcal{X} = \{x_1, x_2, \dots, x_{|\mathcal{X}|}\}$

  $Y \in \mathcal{Y} = \{y_1, y_2, \dots, y_{|\mathcal{Y}|}\}$

- Entropy of r.v. Y conditioned on $X = x_i$

$$H(Y \mid X = x_i) = \sum_{j=1}^{|\mathcal{Y}|} P[Y = y_j | X = x_i] \log_2 P[Y = y_j | X = x_i]$$

- **Conditional entropy:**

$$H(Y \mid X) = -\sum_{i=1}^{|\mathcal{Y}|} P[X = x_i] H(Y \mid X = x_i)$$

Average over the possible values of X

$H(Y|X) \leq H(Y)$

$H(Y|X) = H(Y)$ iff X and Y are independent

If a sender wants to transmit values of Y, the conditional entropy measures the average number of bits per symbol of the minimal message, assuming the value of X is known by the receiver

Information gain

$H(Y) \geq H(Y|X)$
Therefore, $IG(Y \mid X) \geq 0$
$IG(Y \mid X) = 0$ iff X and Y are independent

$IG(Y \mid X) = H(Y) - H(Y|X)$ **[in bits]**

Measures the average number of bits per symbol of the minimal message to transmit values of the random variable Y that one saves by assuming that the receiver knows the value of X

- Select *best* **attribute** as the one that **maximizes the information gain** of the class given by that attribute.

Split at the root of the tree

How do I determine the first question in the decision tree?

- Class variable: $WillWait \in \{yes, no\}$

  Number of instances: $N = 12\ (N_{yes} = 6; N_{no} = 6)$

$$H(WillWait) = H_b\left(\frac{6}{12}\right) = 1\text{ bit}$$

- The best attribute to make a split at the root of the node is the one that maximizes the Information Gain.
  - $IG(WillWait|Type) = 0$ bits
  - $IG(WillWait|Patrons) = 0.64$ bits
  - …

No information is gained

Largest value of the information gain. Best attribute: *Patrons*

Check the other values!

IG of *WillWait* from *Patrons*?

- Attribute: $Patrons \in \{none, some, full\}$
  - $N_{none} = 2 \;\; (N_{yes,none} = 0; N_{no,none} = 2)$

    $$H(WillWait|none) = H_b\left(\frac{0}{2}\right) = 0 \text{ bits}$$
  - $N_{some} = 4 \; (N_{yes,some} = 4; N_{no,some} = 0)$

    $$H(WillWait|some) = H_b\left(\frac{4}{4}\right) = 0 \text{ bits}$$
  - $N_{full} = 6 \;\; (N_{yes,full} = 2; N_{no,full} = 4)$

    $$H(WillWait|full) = H_b\left(\frac{2}{6}\right) = 0.92 \text{ bits}$$

$$H(WillWait|Patrons) = \frac{2}{12}0 + \frac{2}{12}0 + \frac{6}{12}0.92 = 0.46 \text{ bits}$$

$$IG(WillWait|Patrons) = 1 - 0.46 = 0.64 \text{ bits}$$

Recursion: Split at the node *Patrons* = *full*

How do I determine the next question in the decision tree?

- Training instances at node *Patrons* = *full*:

$$\{X_2, X_4, X_5, X_9, X_{10}, X_{12}\}$$

$$N = 6\ (N_{yes} = 2; N_{no} = 4) \Rightarrow$$

$$H(WillWait) = H_b\left(\frac{2}{6}\right) = 0.92 \text{ bits}$$

Check this!

- The best attribute to make a split at this node is $Hungry$

$$H(WillWait|Hungry) = \frac{4}{6}H_b\left(\frac{2}{4}\right) + \frac{2}{6}H_b\left(\frac{0}{2}\right) = 0.67 \text{ bits}$$

$$IG(WillWait|Hungry) = 0.92 - 0.67 = 0.25 \text{ bits}$$

When does one stop splitting a node?

- The training examples assigned to that node belong to the same class. [The leaf node assigns that class label]
- Node has no examples associated to it. [The leaf node assigns the default class label]
- No more attributes left for splitting the data. [The leaf node assigns the majority class label in that node]
- Prepruning (limit the tree size to avoid **overfitting**)
  - The number of training examples associated to the node is below a threshold.
  - The Impurity Gain is below a threshold.

E.g. Threshold = I_G of a random split

[The leaf node assigns the majority class label in that node]

Underfitting / overfitting

- **Underfitting**

  The type of predictor considered **has low expressive capacity.** In consequence, it is no able to capture the dependencies between the attributes and the variable to be predicted.

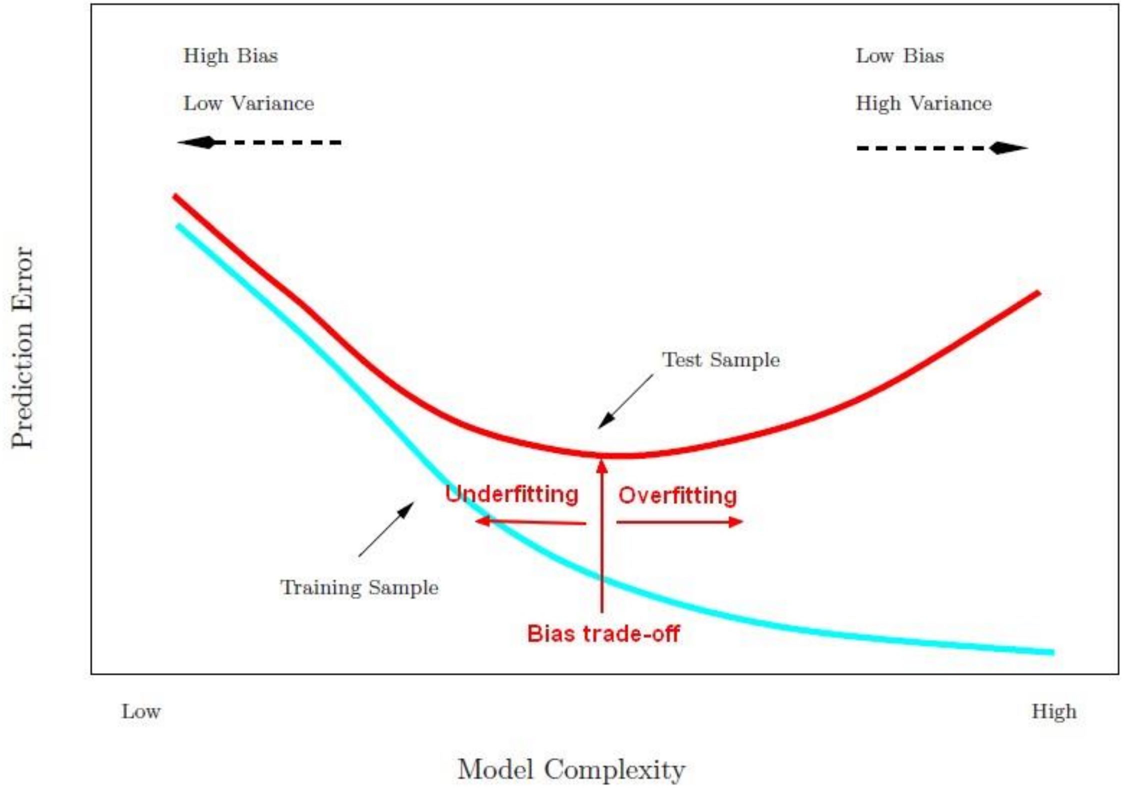
  The error of the predictor is too high.

- **Overfitting**

  The type of predictor considered is **too flexible** and learns spurious patterns that are not relevant for prediction (e.g. sampling fluctuations, noise, outliers, etc.).

  Training estimate of the expected loss is too optimistic and underestimates the actual error.

Underfitting / overfitting



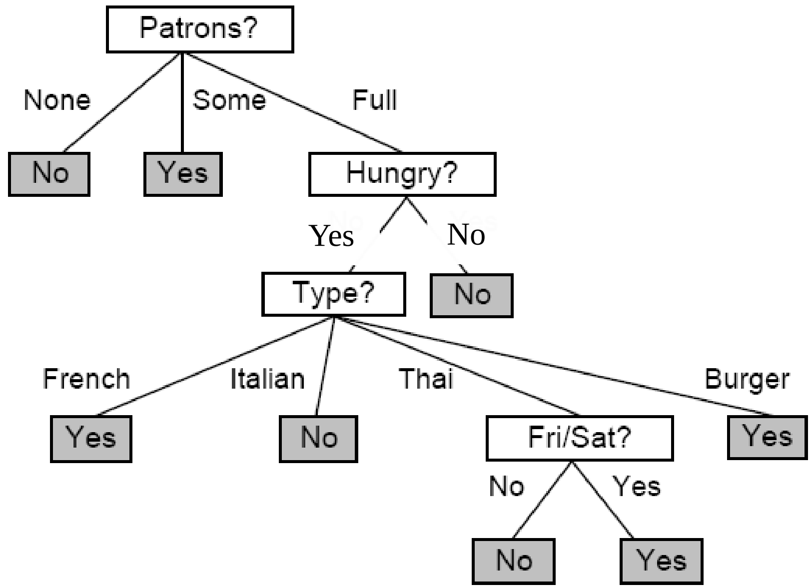
Source: https://gerardnico.com/ under license CC Attribution-Noncommercial-Share Alike 4.0 International

Pruning to avoid overfitting in DT's

- Bias towards smaller (less complex) trees.
  - Prepruning
  - Postpruning: Grow tree to a large size and then prune subtrees that do not provide significant gains in predictive accuracy.
    - Consider an internal node.
    - If turning that node into a leaf does not lead to a significant decrease in the predictive accuracy of the pruned decision tree, then eliminate the subtree which has that node as its root.
    - For this process, accuracy can be estimated on a separate validation set (reduced error pruning), or by CV (e.g. as in CART)
    - Continue pruning until significant deterioration of accuracy

Postpruning is generally preferred. This is a common strategy in machine learning: consider first a potentially complex model and then penalize complexity

Interpretability: Rule extraction



System of rules:

The **group stays if**

either the restaurant has **some patrons**

or (the restaurant is **full and** the **group is hungry and**

(the type of food is **French or** (**Thai and** it is **Fri/Sat**) **or Burger**)

Otherwise, the **group leaves**.

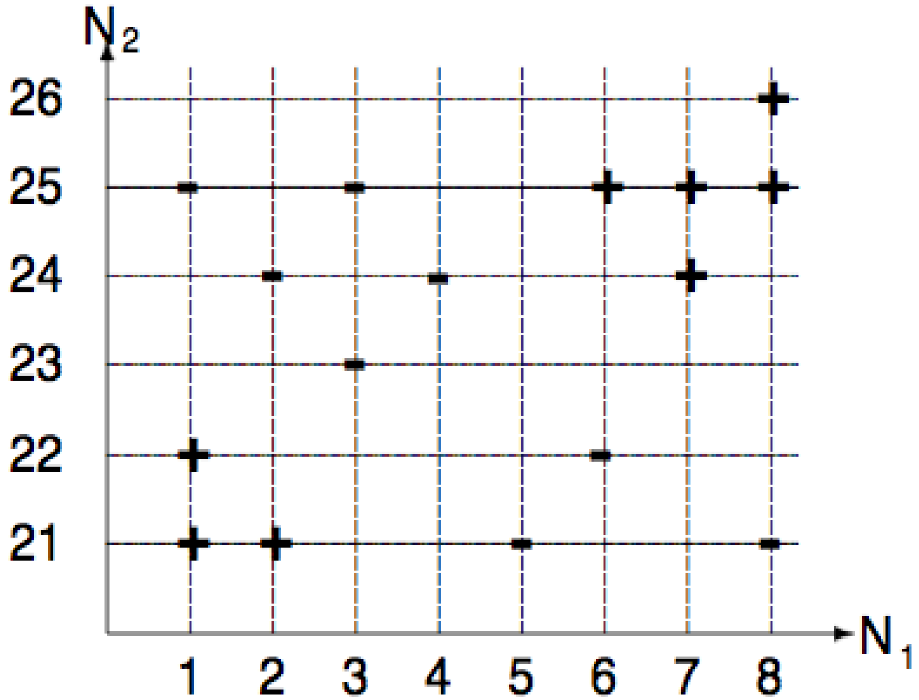
The model is interpretable!

C4.5 Decision tree

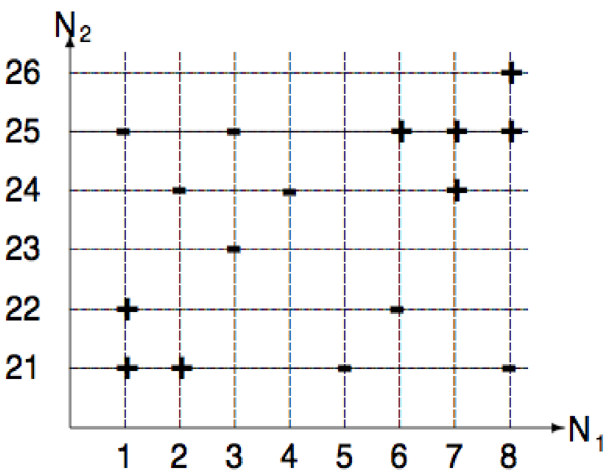
- Evolution of ID3 by Quinlan (1992)
- Includes
  - Tests based on numerical attributes
  - Fuzzy decisions
  - Post-pruning
  - Normalization of information gain for multivalued attributes.
  - Handling of missing values
  - Rule extraction and pruning

C4.5: Handling of numerical attributes

- Attributes: N_1, N_2
- Class: “+”, “-” $H(Class) = H_b\left(\frac{8}{16}\right) = 1$ bit

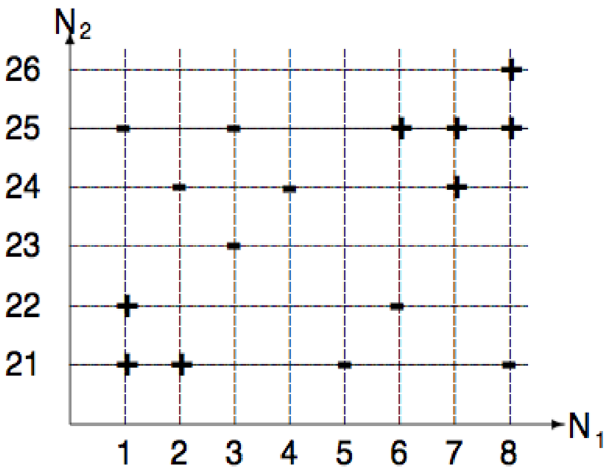


Tests on N_1



Pregunta	Rama "No"	Rama "Sí"	Entropía clase en Rama "No"	Entropía clase en Rama "Sí"	H(clase | Pre-gunta)	IG
$N_1>1$	2+, 1-	6+, 7-	H(2/3, 1/3) = 0.918 bits	H(6/13, 7/13) = 0.996 bits	3/16*0.918+ 13/16*0.996 = 0.981 bits	1-0.981= 0.019 bits
$N_1>2$	3+, 2-	5+, 6-	H(3/5, 2/5) = 0.971 bits	H(5/11, 6/11) = 0.994 bits	5/16*0.971+ 11/16*0.994 = 0.987 bits	1-0.987= 0.013 bits
$N_1>3$	3+, 4-	5+, 4-	H(3/7, 4/7) = 0.985 bits	H(5/9, 4/9) = 0.991 bits	7/16*0.985+ 9/16*0.991 = 0.988 bits	1-0.988= 0.012 bits
$N_1>4$	3+, 5-	5+, 3-	H(3/8, 5/8) = 0.954 bits	H(5/8, 3/8) = 0.954 bits	8/16*0.954+ 8/16*0.954 = 0.954 bits	1-0.954= 0.046 bits
$N_1>5$	3+, 6-	5+, 2-	H(3/9, 6/9) = 0.918 bits	H(5/7, 2/7) = 0.863 bits	9/16*0.918+ 7/16*0.863 = 0.894 bits	1-0.894= 0.106 bits
$N_1>6$	4+, 7-	4+, 1-	H(4/11, 7/11)= 0.946 bits	H(4/5, 1/5) = 0.722 bits	11/16*0.946+ 5/16*0.722 = 0.876 bits	1-0.876= 0.124 bits
$N_1>7$	6+, 7-	2+, 1-	H(6/13, 7/13)= 0.996 bits	H(2/3, 1/3) = 0.918 bits	13/16*0.996+ 3/16*0.918 = 0.981 bits	1-0.981= 0.019 bits
$N_1>8$	8+, 8-	0+, 0-	1 bit	--	16/16*1+ 0/16*-- = 0 bits	1-1 = 0 bits

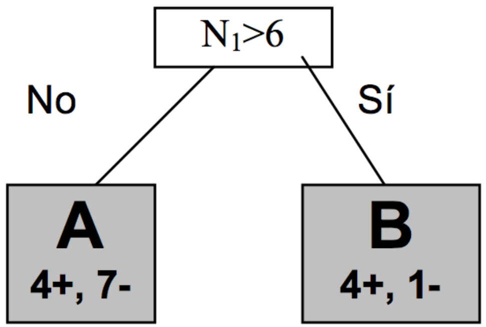
Tests on N_2

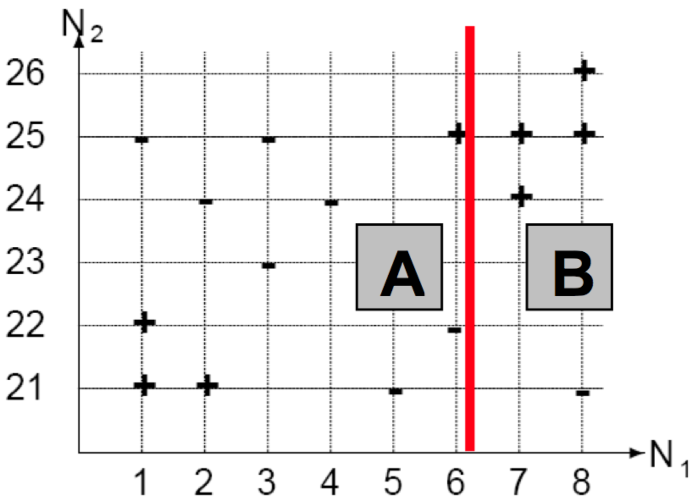


Pregunta	Rama "No"	Rama "Sí"	Entropía clase en Rama "No"	Entropía clase en Rama "Sí"	H(clase | Pre-gunta)	IG
N_2>21	2+, 2-	6+, 6-	H(2/4, 2/4) = 1 bits	H(6/12, 6/12) = 1 bits	4/16*1+ 12/16*1 = 1 bits	1-1= 0 bits
N_2>22	3+, 3-	5+, 5-	H(3/6, 3/6) = 1 bits	H(5/10, 5/10) = 1 bits	6/16*1+ 10/16*1 = 1 bits	1-1= 0 bits
N_2>23	3+, 4-	5+, 4-	H(3/7, 4/7) = 0.985 bits	H(5/9, 4/9) = 0.991 bits	7/16*0.985+ 9/16*0.991 = 0.988 bits	1-0.988= 0.012 bits
N_2>24	4+, 6-	4+, 2-	H(4/10, 6/10)= 0.971 bits	H(4/6, 2/6) = 0.918 bits	10/16*0.971+ 6/16*0.918 = 0.951 bits	1-0.951= 0.049 bits
N_2>25	7+, 8-	1+, 0-	H(7/15, 8/15)= 0.997 bits	H(1/1, 0/1) = 0 bits	15/16*0.997+ 1/16*0 = 0.935 bits	1-0.894= 0.065 bits

Smaller than
I_G of test $(N_1 > 6?)$

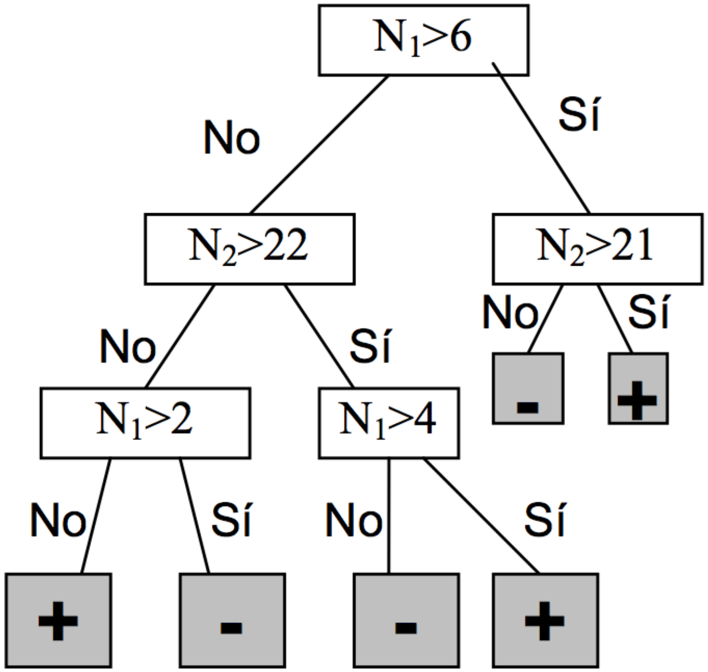
Test at root node

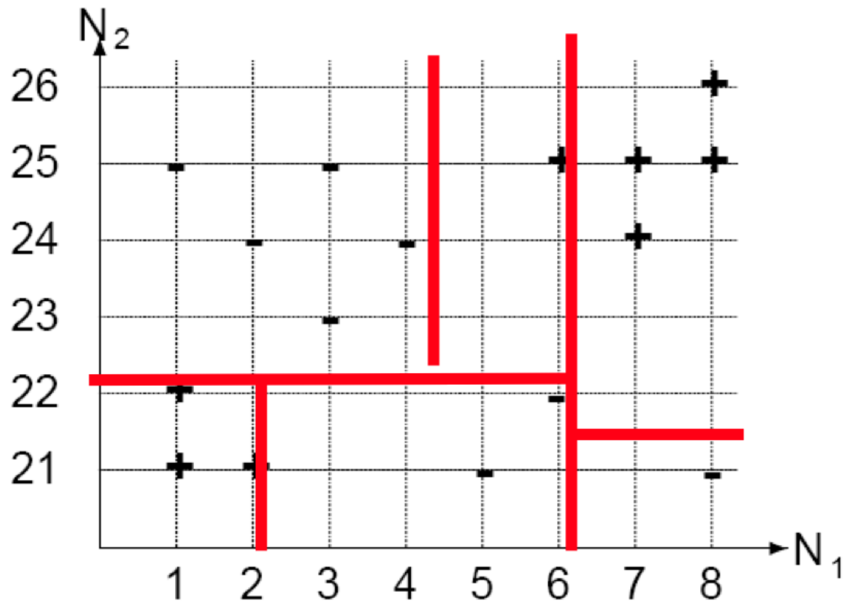




- The original attribute space has been partition into 2 disjoint subspaces (A and B)
- Using a "**divide an conquer**" strategy, and recursively partition A and B separately.

Final C4.5 decision tree





Decision trees: pros & cons

- **Advantages**
  - Simple implementation.
  - Interpretable results.
  - Fast training & prediction.
- **Drawbacks**
  - Not very accurate predictions. However, they can be used as base learners for an ensemble.

https://scikit-learn.org/stable/modules/tree.html

Decision forests: Ensembles of DT's

- **Randomization**
  - **Bagging**
  - **Random forest**
- **Randomization + optimization**
  - **Boosting**
  - **Gradient boosting**
  - **Xgboost (Extreme Gradient Boosting)**

  [https://xgboost.readthedocs.io/en/latest/]

> Bagging and boosting ensembles can also be composed of other types of base learners, such as neural networks

> Random forest, gradient boosting, and xgboost have excellent off-the-shelf performance in non-structured problems

[https://scikit-learn.org/stable/modules/ensemble.html]