Decision Trees

Outline

I. Decision trees

II. Learning decision trees from examples

III. Entropy and attribute choice

** Figures are from the textbook site (or by the instructor).

Restaurant Waiting

Problem Decide whether to wait for a table at a restaurant.

Restaurant Waiting

Problem Decide whether to wait for a table at a restaurant.

Output: a Boolean variable *WillWait* (true where we do wait for a table).

Restaurant Waiting

Problem Decide whether to wait for a table at a restaurant.

Output: a Boolean variable *WillWait* (true where we do wait for a table).

Input: a vector of ten attributes, each with discrete values:

Restaurant Waiting

Problem Decide whether to wait for a table at a restaurant.

Output: a Boolean variable *WillWait* (true where we do wait for a table).

Input: a vector of ten attributes, each with discrete values:

1. *Alternate*: whether there is a suitable alternative restaurant nearby.
2. *Bar*: whether the restaurant has a comfortable bar area to wait in.
3. *Fri/Sat*: true on Fridays and Saturdays.
4. *Hungry*: whether we are hungry right now.
5. *Patrons*: how many people are in the restaurant (values: *None*, *Some*, and *Full*).
6. *Price*: the restaurant's price range ($, $$, $$$).
7. *Raining*: whether it is raining outside.
8. *Reservation*: whether we made a reservation.
9. *Type*: the kind of restaurant (*French*, *Italian*, *Thai*, or *burger*).
10. *WaitEstimate*: host's wait estimate: 0-10, 10-30, 30-60, or >60 minutes.

Restaurant Waiting

Problem Decide whether to wait for a table at a restaurant.

Output: a Boolean variable *WillWait* (true where we do wait for a table).

Input: a vector of ten attributes, each with discrete values:

1. *Alternate*: whether there is a suitable alternative restaurant nearby.
2. *Bar*: whether the restaurant has a comfortable bar area to wait in.
3. *Fri/Sat*: true on Fridays and Saturdays.
4. *Hungry*: whether we are hungry right now.
5. *Patrons*: how many people are in the restaurant (values: *None*, *Some*, and *Full*).
6. *Price*: the restaurant's price range ($, $$, $$$).
7. *Raining*: whether it is raining outside.
8. *Reservation*: whether we made a reservation.
9. *Type*: the kind of restaurant (*French*, *Italian*, *Thai*, or *burger*).
10. *WaitEstimate*: host's wait estimate: 0-10, 10-30, 30-60, or >60 minutes.

$2^6 \times 3^2 \times 4^2 = 9{,}216$ possible combinations of attribute values.

Training Examples

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

- The correct output is given for only 12 out of 9,216 examples.

Training Examples

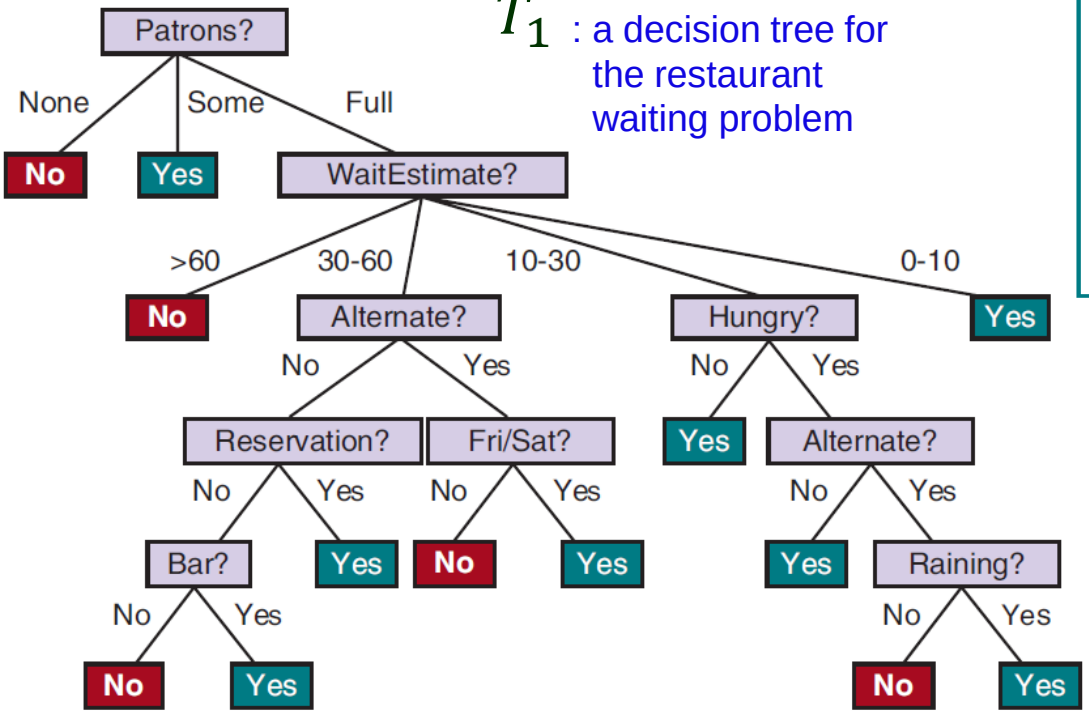
Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

- The correct output is given for only 12 out of 9,216 examples.
- We need to make our best guess at the missing 9,204 output values.

I. What is a Decision Tree?

A *decision tree* maps a set of attribute values to a "decision".

- It performs a sequence of tests, starting at the root and descending down to a leaf (which specifies the output value).
- Each internal node corresponds to a test of the value of an attribute.

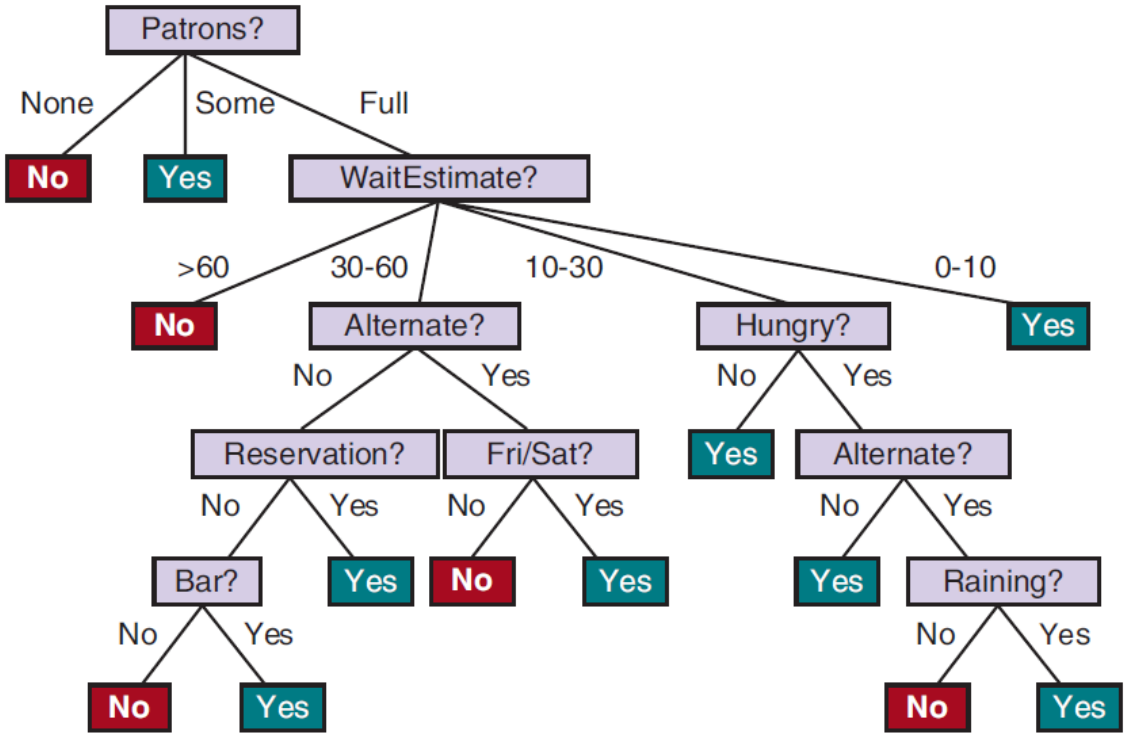


T_1 : a decision tree for the restaurant waiting problem

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

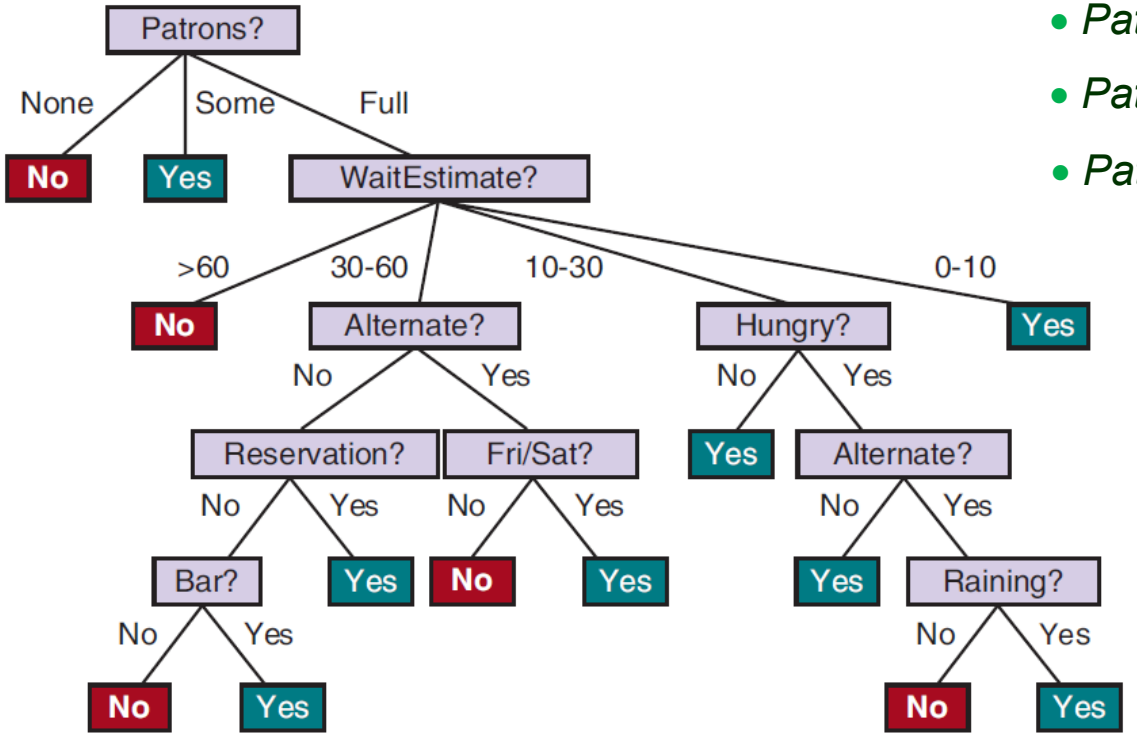
Boolean Classification

- Inputs are discrete values.
- Outputs are either *true* (a positive example) or *false* (a negative one).



Boolean Classification

- Inputs are discrete values.
- Outputs are either *true* (a positive example) or *false* (a negative one).

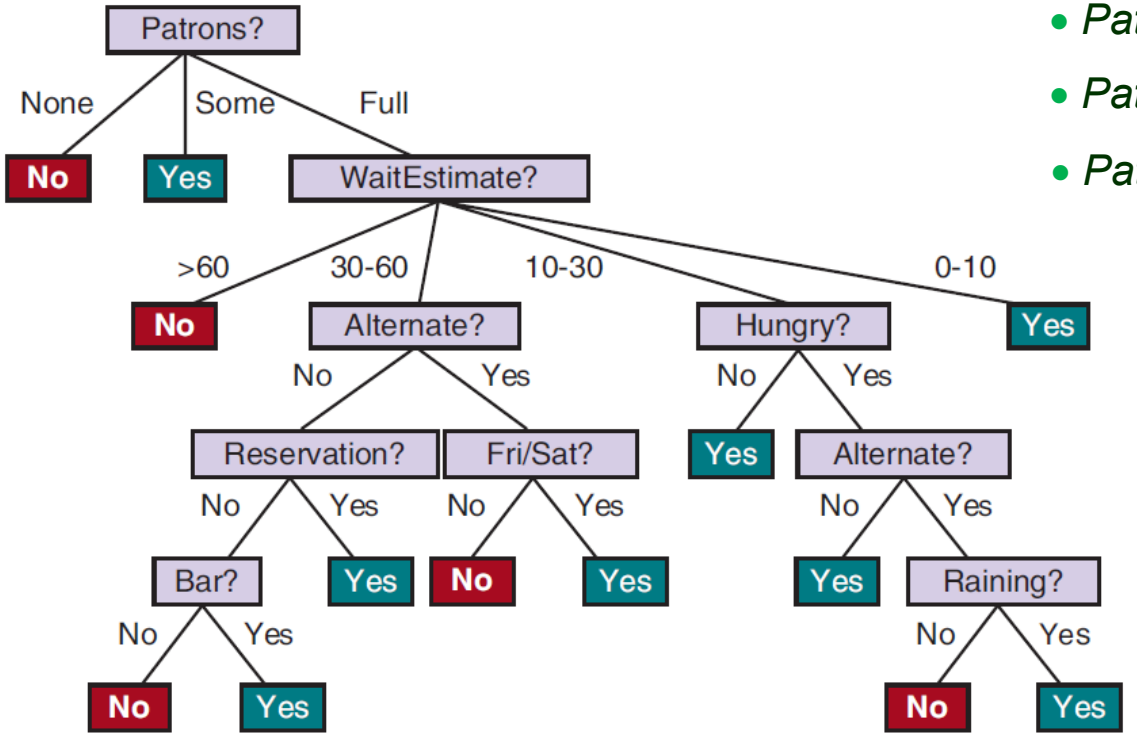


Positive examples:

- *Patrons* = *Some*
- *Patrons* = *Full* ∧ *WaitEstimate* = 0−10
- *Patrons* = *Full*
  ∧ *WaitEstimate* = 10−30
  ∧ *Hungry* = *Yes*
  ∧ *Alternate* = *No*

Boolean Classification

- Inputs are discrete values.
- Outputs are either *true* (a positive example) or *false* (a negative one).



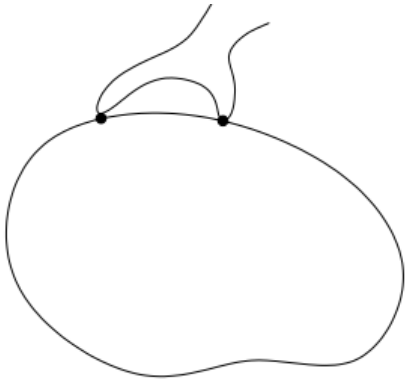
Positive examples:

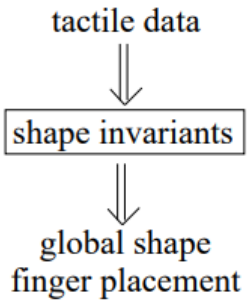
- *Patrons* = *Some*
- *Patrons* = *Full* ∧ *WaitEstimate* = 0−10
- *Patrons* = *Full*
  ∧ *WaitEstimate* = 10−30
  ∧ *Hungry* = *Yes*
  ∧ *Alternate* = *No*

Negative examples:

- *Patrons* = *No*
- *Patrons* = *Full*
  ∧ *WaitEstimate* > 60

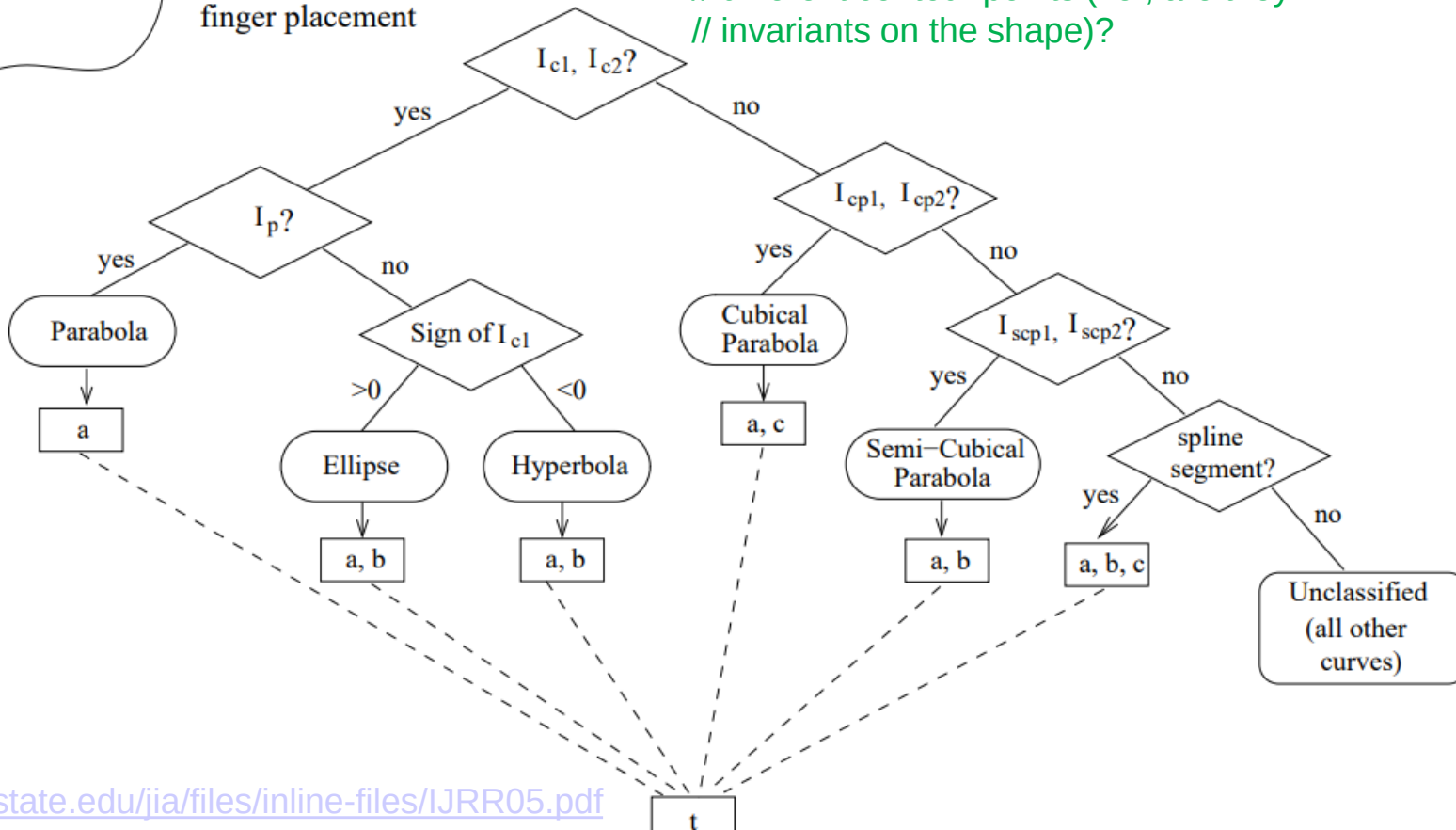
Shape Recognition Tree





A robotic hand can recognize curved shapes from differential invariants (expressions of curvature, torsion, etc.) constructed over tactile data.

// Do values of I_{c1} and I_{c2} vary at
// different contour points (i.e., are they
// invariants on the shape)?



https://faculty.sites.iastate.edu/jia/files/inline-files/IJRR05.pdf

Why Decision Trees?

- ♦ They yield nice, concise, and understandable results.
- ♦ They represent simple rules for classifying instances that are described by discrete attribute values.
- ♦ Decision tree learning algorithms are relatively efficient – *linear* in the size of the decision tree and the size of the data set.
- ♦ Decision trees are often among the first to be tried on a new data set.
- ♠ They are not good with real-valued attributes.

Expressiveness of a Decision Tree

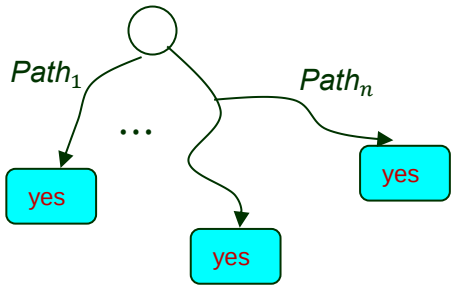
Equivalent logical statement of a decision tree:

$$Output \Leftrightarrow Path_1 \vee Path_2 \vee \cdots$$

(one statement for Output = Yes
and another for Output = No)

where

$$Path_i = (A_{i1} = v_{i1} \wedge A_{i2} = v_{i2} \wedge \cdots)$$



Expressiveness of a Decision Tree

Equivalent logical statement of a decision tree:

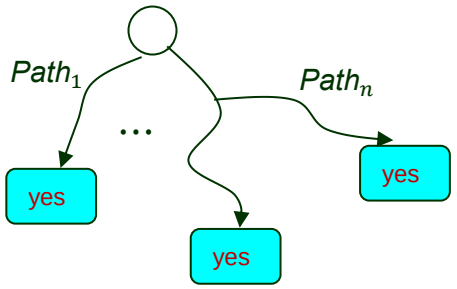
$$Output \Leftrightarrow Path_1 \vee Path_2 \vee \cdots$$

(one statement for Output = Yes
and another for Output = No)

where

$$Path_i = (A_{i1} = v_{i1} \wedge A_{i2} = v_{i2} \wedge \cdots)$$

attribute value



Expressiveness of a Decision Tree

Equivalent logical statement of a decision tree:

$$Output \Leftrightarrow Path_1 \vee Path_2 \vee \cdots$$

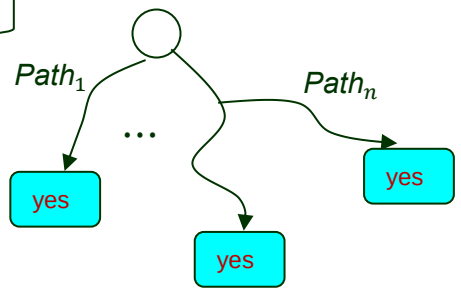
(one statement for Output = Yes and another for Output = No)

where

$$Path_i = (A_{i1} = v_{i1} \wedge A_{i2} = v_{i2} \wedge \cdots)$$

attribute value

Disjunctive normal form (DNF)



Expressiveness of a Decision Tree

Equivalent logical statement of a decision tree:

$$Output \Leftrightarrow Path_1 \vee Path_2 \vee \cdots$$

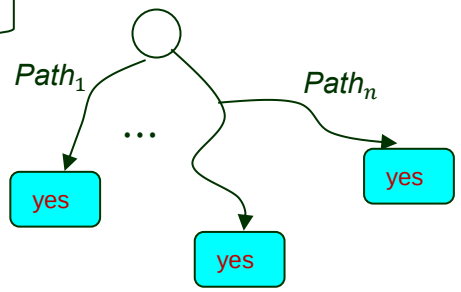
(one statement for Output = Yes and another for Output = No)

where

$$Path_i = (A_{i1} = v_{i1} \wedge A_{i2} = v_{i2} \wedge \cdots)$$

attribute value

Disjunctive normal form (DNF)



Any sentence in propositional logic can be converted into a DNF.

Expressiveness of a Decision Tree

Equivalent logical statement of a decision tree:

$$Output \Leftrightarrow Path_1 \vee Path_2 \vee \cdots$$

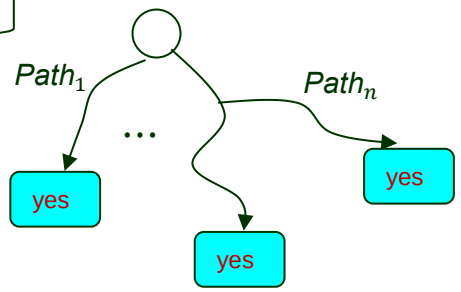
(one statement for Output = Yes
and another for Output = No)

where

$$Path_i = (A_{i1} = v_{i1} \wedge A_{i2} = v_{i2} \wedge \cdots)$$

attribute value

Disjunctive normal form (DNF)



Any sentence in propositional logic can be converted into a DNF.

⇩

Any sentence in propositional logic can expressed as a decision tree.

II. Learning DTs from Examples

Problem Find a tree that is

a) consistent with the training examples and

b) as small as possible.

II. Learning DTs from Examples

Problem Find a tree that is

a) consistent with the training examples and

b) as small as possible.

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

II. Learning DTs from Examples

Problem Find a tree that is

a) consistent with the training examples and

b) as small as possible.

- Generally, it's *intractable* to find a smallest consistent tree.

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

II. Learning DTs from Examples

Problem Find a tree that is

a) consistent with the training examples and

b) as small as possible.

- Generally, it's *intractable* to find a smallest consistent tree.
- A *suboptimal tree* can be constructed with some simple heuristics.

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

II. Learning DTs from Examples

Problem Find a tree that is
a) consistent with the training examples and
b) as small as possible.

- Generally, it's *intractable* to find a smallest consistent tree.
- A *suboptimal tree* can be constructed with some simple heuristics.

Greedy divide-and-conquer strategy:

- Test the most important attribute first.

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

II. Learning DTs from Examples

Problem Find a tree that is

a) consistent with the training examples and

b) as small as possible.

- Generally, it's *intractable* to find a smallest consistent tree.
- A *suboptimal tree* can be constructed with some simple heuristics.

Greedy divide-and-conquer strategy:

- Test the most important attribute first.

  make the most difference to the classification

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

II. Learning DTs from Examples

Problem Find a tree that is
a) consistent with the training examples and
b) as small as possible.

- Generally, it's *intractable* to find a smallest consistent tree.
- A *suboptimal tree* can be constructed with some simple heuristics.

Greedy divide-and-conquer strategy:

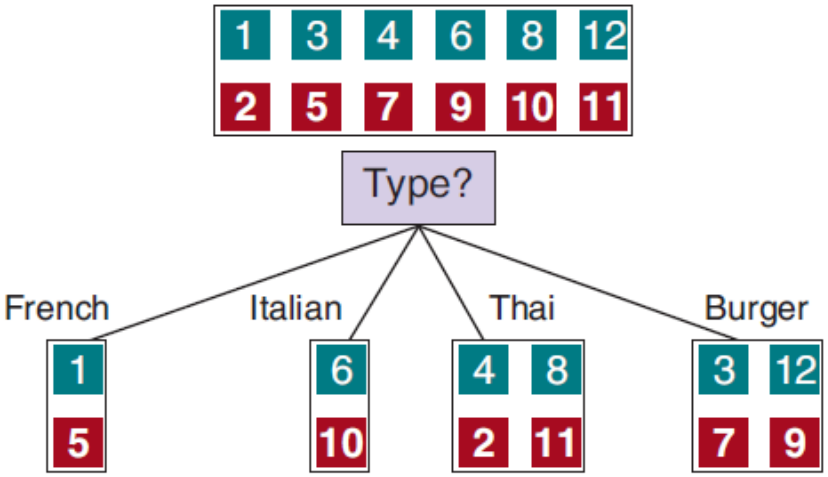
- Test the most important attribute first.

  make the most difference to the classification

- Recursively solve the smaller subproblems.

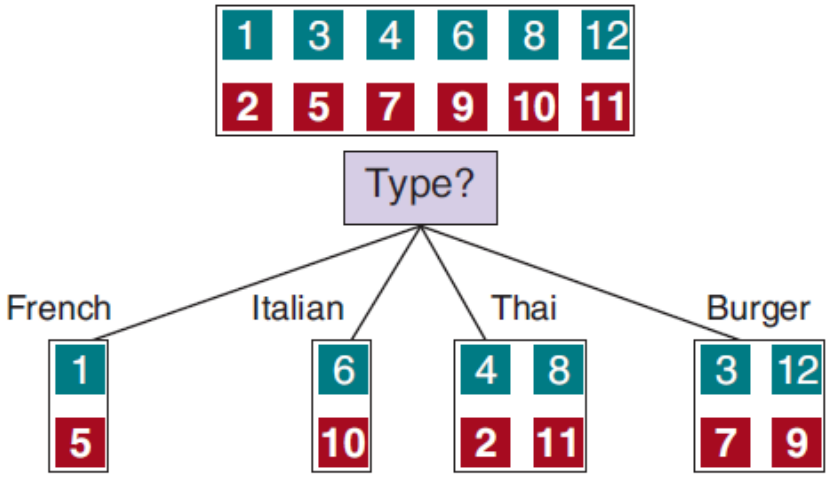
Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Poor and Good Attributes



Poor attribute *Type*:

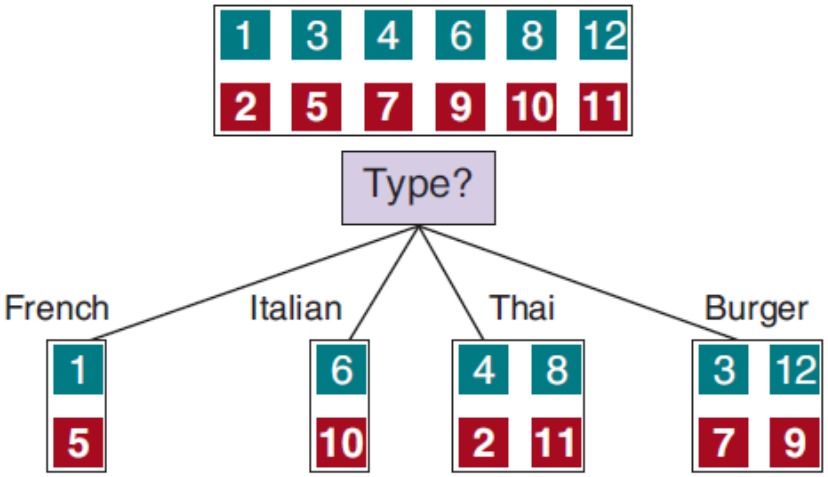
Poor and Good Attributes



Poor attribute *Type*:

- four outcomes

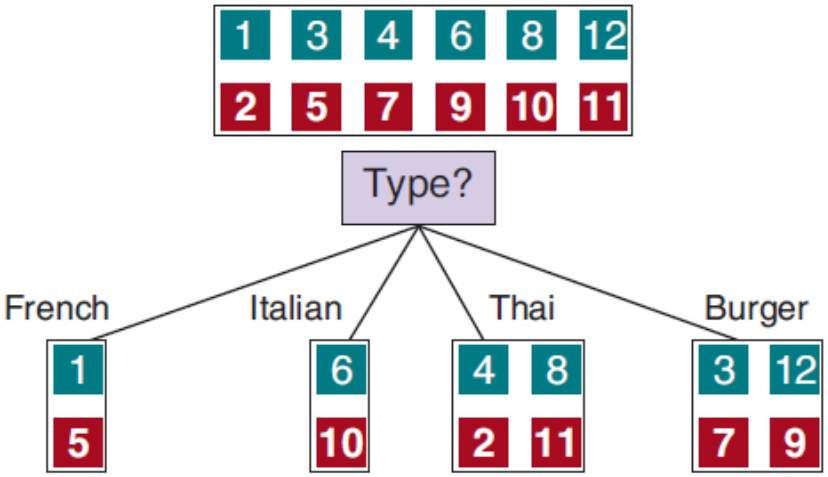
Poor and Good Attributes

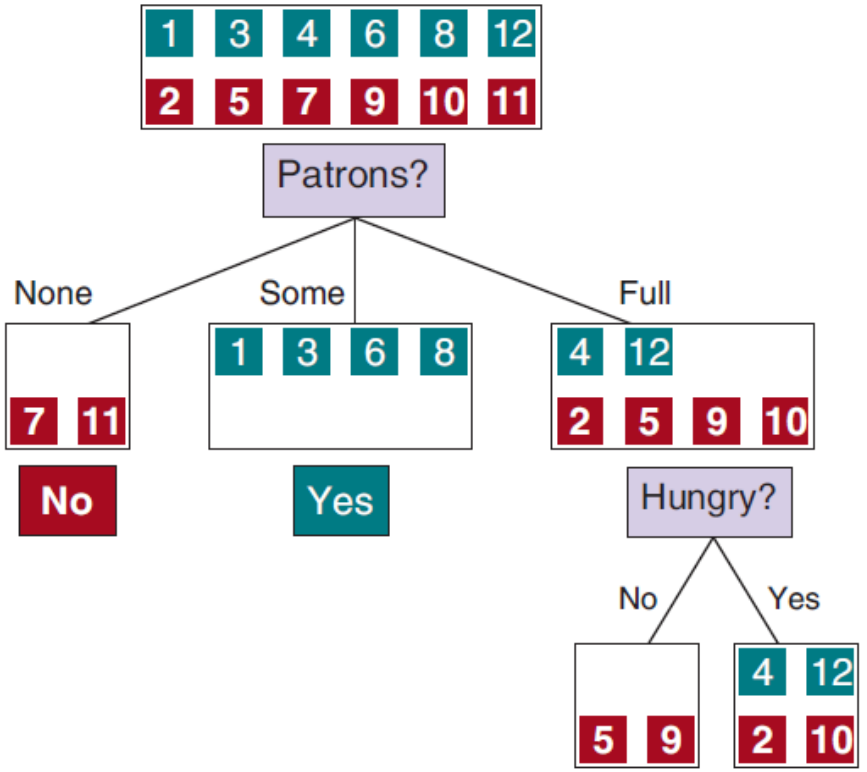


Poor attribute *Type*:

- four outcomes
- each with the same number of positive as negative examples

Poor and Good Attributes



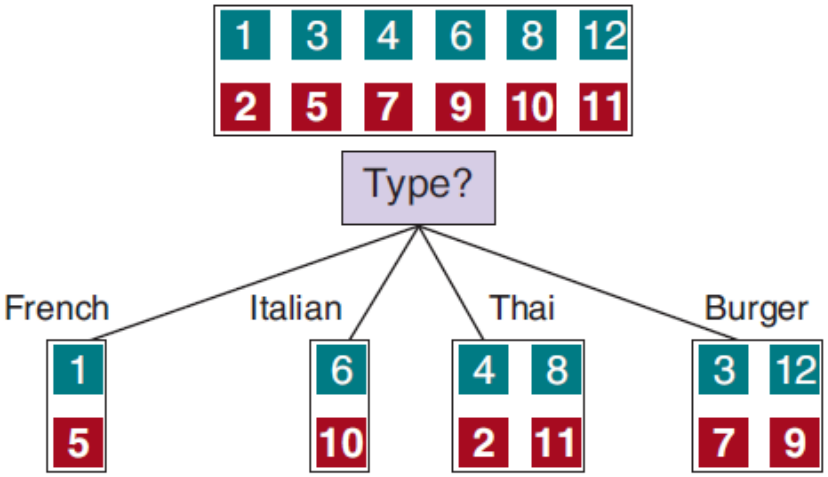


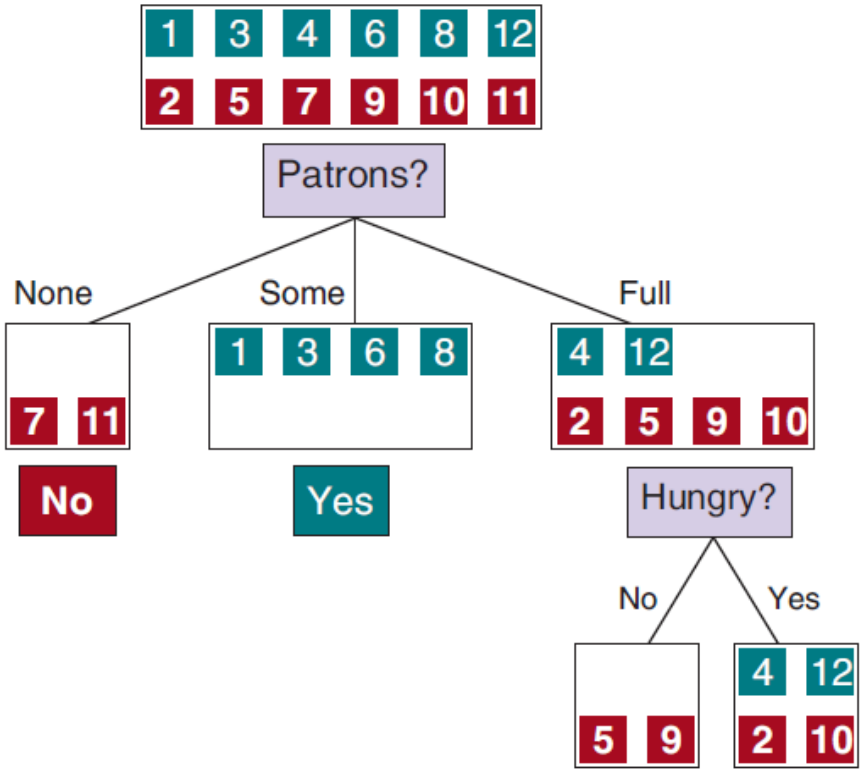
Poor attribute *Type*:

- four outcomes
- each with the same number of positive as negative examples

Good attributes *Patrons* and *Hungry*:

Poor and Good Attributes





Poor attribute *Type*:

- four outcomes
- each with the same number of positive as negative examples

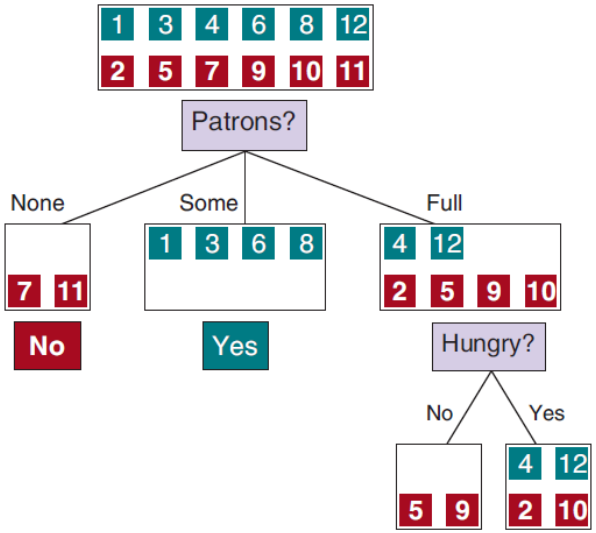
Good attributes *Patrons* and *Hungry*:

- effective separations of positive and negative examples

Four Cases

1. All positive (or all negative) remaining examples.

Done.



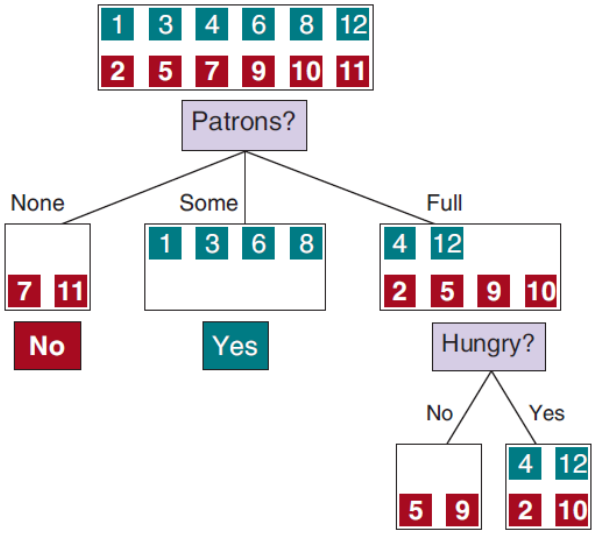
Four Cases

1. All positive (or all negative) remaining examples.

   Done.

2. Mixed positive and negative remaining examples.

   Choose the best attribute to split them.



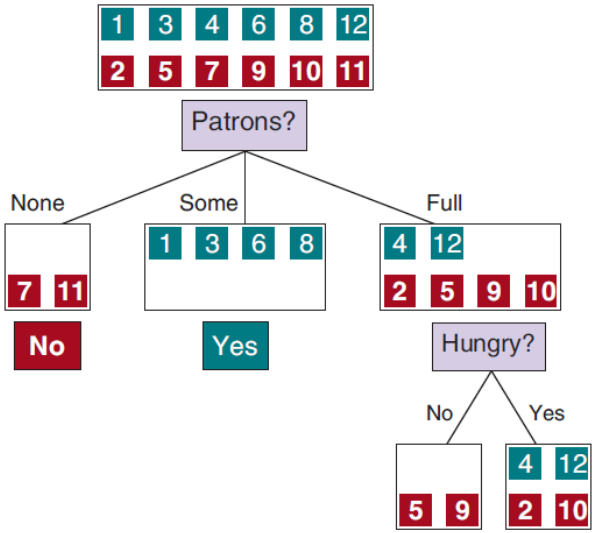
Four Cases

1. All positive (or all negative) remaining examples.

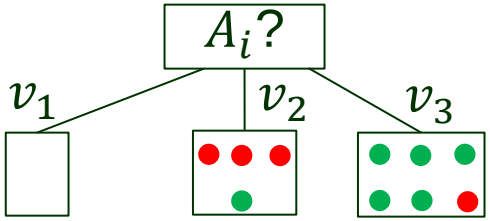
   Done.

2. Mixed positive and negative remaining examples.

   Choose the best attribute to split them.



3. No examples left.



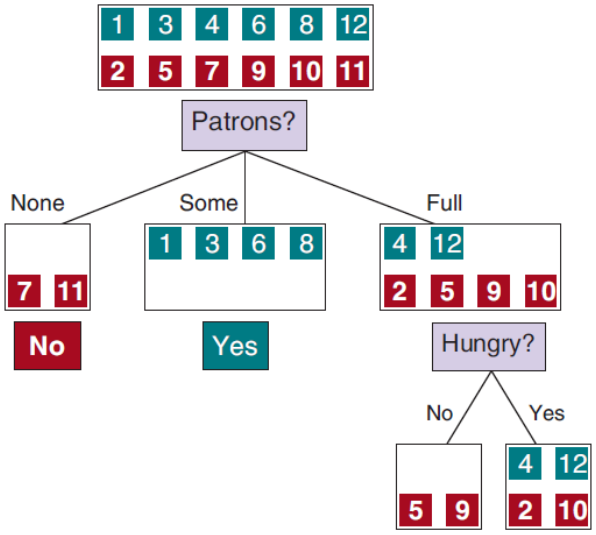
Four Cases

1. All positive (or all negative) remaining examples.

   Done.

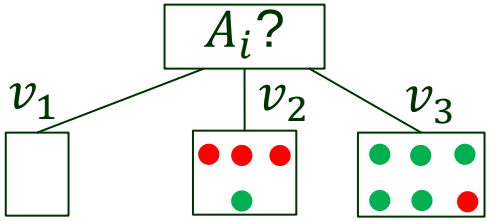
2. Mixed positive and negative remaining examples.

   Choose the best attribute to split them.



3. No examples left.

   Return the most common output value (e.g., •) for the example set used in constructing the parent (A_i?)



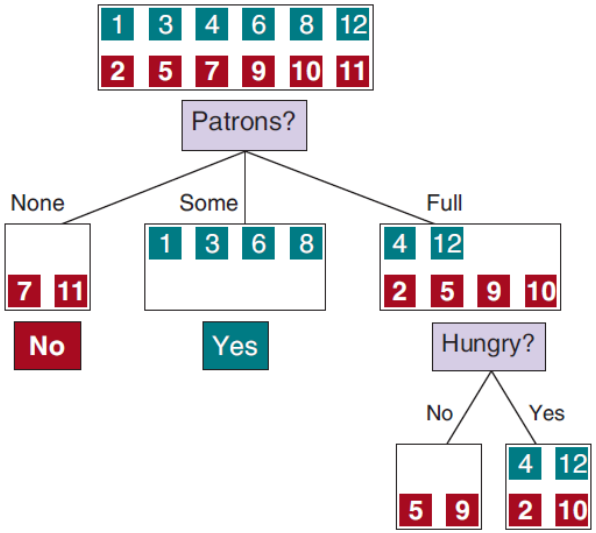
Four Cases

1. All positive (or all negative) remaining examples.

   Done.

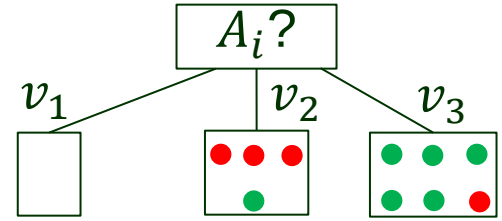
2. Mixed positive and negative remaining examples.

   Choose the best attribute to split them.



3. No examples left.

   Return the most common output value (e.g., •) for the example set used in constructing the parent (A_i?)



Yes (6 • vs 4 • in the set queried with A_i?)

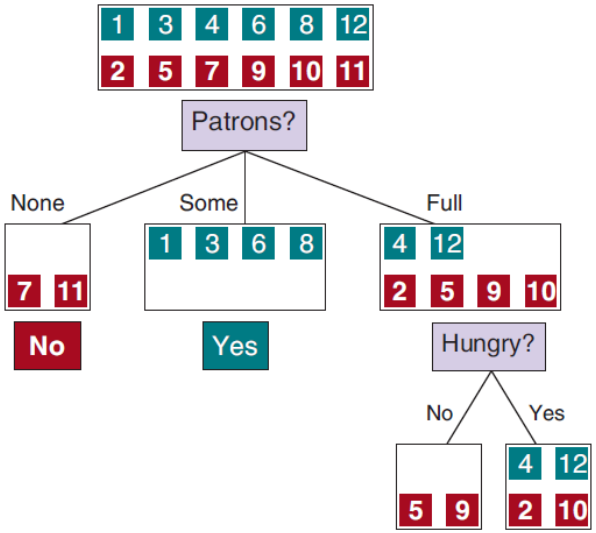
Four Cases

1. All positive (or all negative) remaining examples.

   Done.

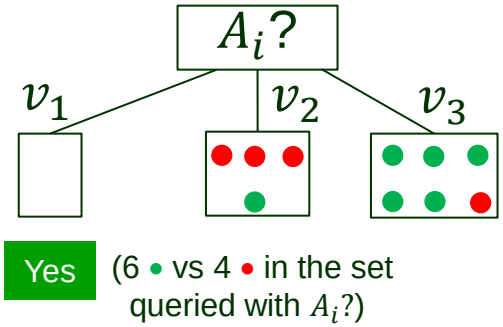
2. Mixed positive and negative remaining examples.

   Choose the best attribute to split them.

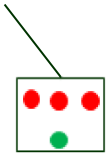


3. No examples left.

   Return the most common output value (e.g., •) for the example set used in constructing the parent (A_i?)



4. No attributes left but both positive and negative examples.



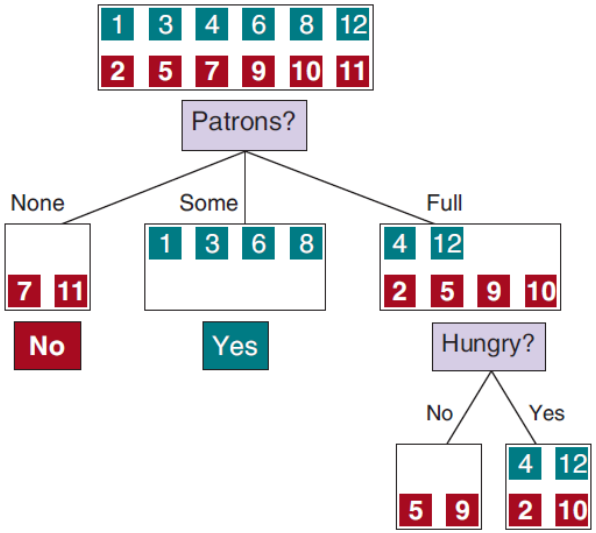
Four Cases

1. All positive (or all negative) remaining examples.

   Done.

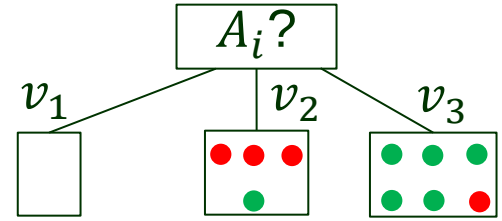
2. Mixed positive and negative remaining examples.

   Choose the best attribute to split them.



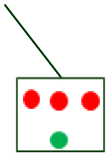
3. No examples left.

   Return the most common output value (e.g., •) for the example set used in constructing the parent (A_i?)



Yes (6 • vs 4 • in the set queried with A_i?)

4. No attributes left but both positive and negative examples.



Return the most common output value of these examples (by a *majority vote*)

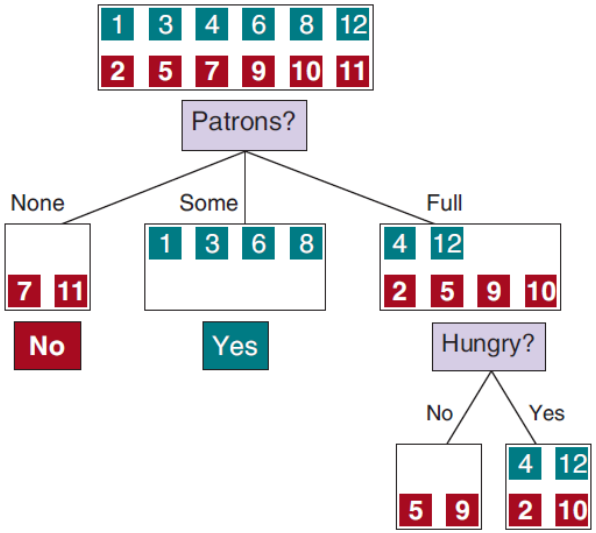
Four Cases

1. All positive (or all negative) remaining examples.

   Done.

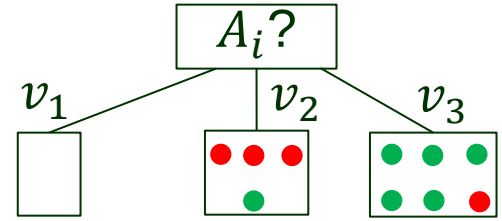
2. Mixed positive and negative remaining examples.

   Choose the best attribute to split them.



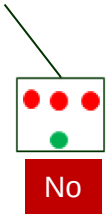
3. No examples left.

   Return the most common output value (e.g., •) for the example set used in constructing the parent (A_i?)



Yes (6 • vs 4 • in the set queried with A_i?)

4. No attributes left but both positive and negative examples.



Return the most common output value of these examples (by a *majority vote*)

Algorithm for Learning DTs

function LEARN-DECISION-TREE($examples$, $attributes$, $parent\_examples$) **returns** a tree

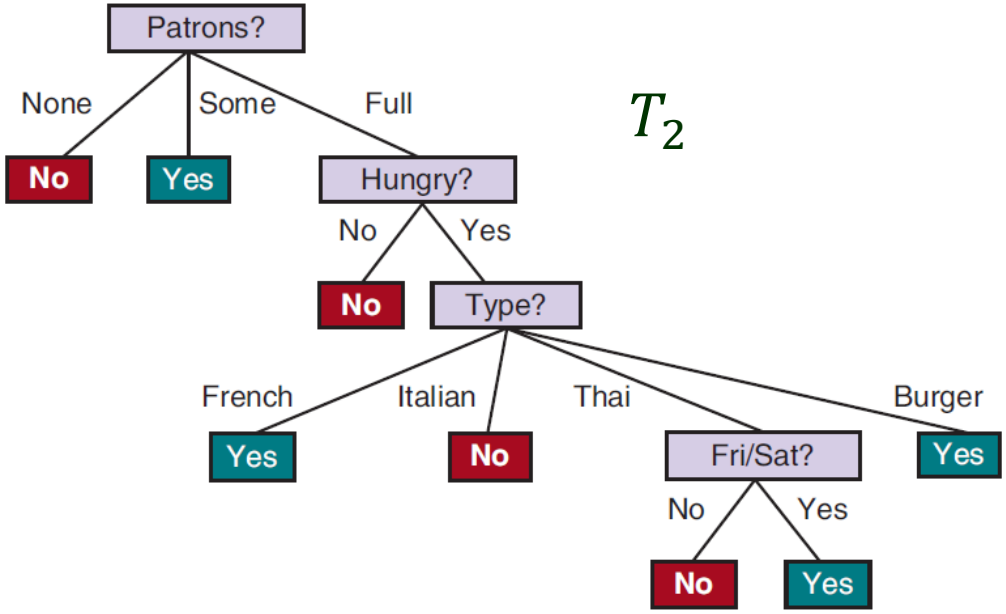
if $examples$ is empty **then return** PLURALITY-VALUE($parent\_examples$) // case 3
else if all $examples$ have the same classification **then return** the classification // case 1
else if $attributes$ is empty **then return** PLURALITY-VALUE($examples$) // case 4
else // case 2
    $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, $examples$)
    $tree \leftarrow$ a new decision tree with root test A
    for each value v of A **do**
        $exs \leftarrow \{e : e \in examples$ **and** $e.A = v\}$
        $subtree \leftarrow$ LEARN-DECISION-TREE(exs, $attributes - A$, $examples$)
        add a branch to $tree$ with label $(A = v)$ and subtree $subtree$
    return $tree$

Algorithm for Learning DTs

function LEARN-DECISION-TREE($examples$, $attributes$, $parent\_examples$) **returns** a tree

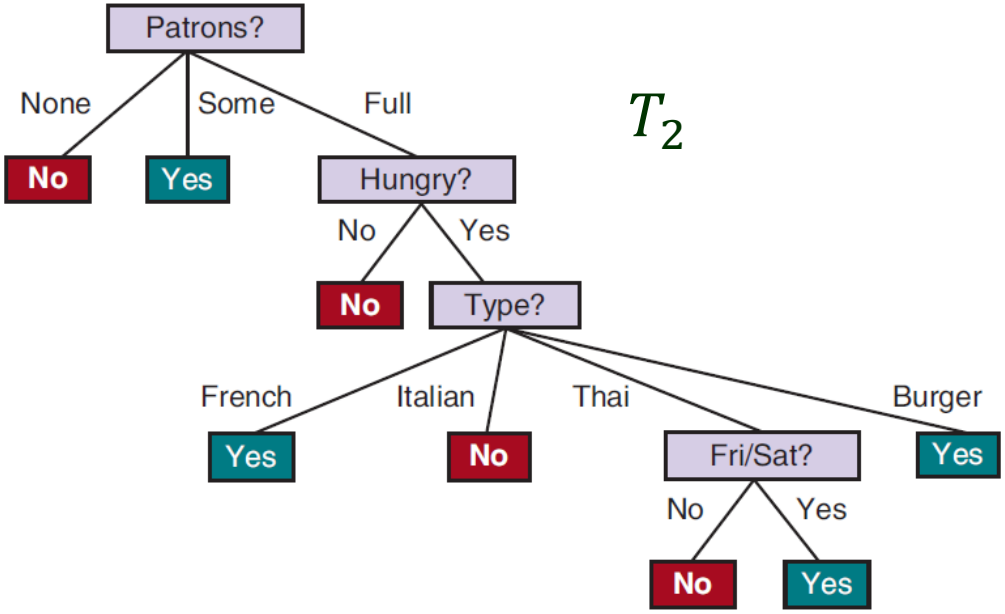
if $examples$ is empty **then return** PLURALITY-VALUE($parent\_examples$) // case 3
else if all $examples$ have the same classification **then return** the classification // case 1
else if $attributes$ is empty **then return** PLURALITY-VALUE($examples$) // case 4
else // case 2
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, $examples$)
  $tree \leftarrow$ a new decision tree with root test A
  for each value v of A **do**
    $exs \leftarrow \{e : e \in examples$ **and** $e.A = v\}$
    $subtree \leftarrow$ LEARN-DECISION-TREE(exs, $attributes - A$, $examples$)
    add a branch to $tree$ with label $(A = v)$ and subtree $subtree$
  return $tree$

Output Decision Tree

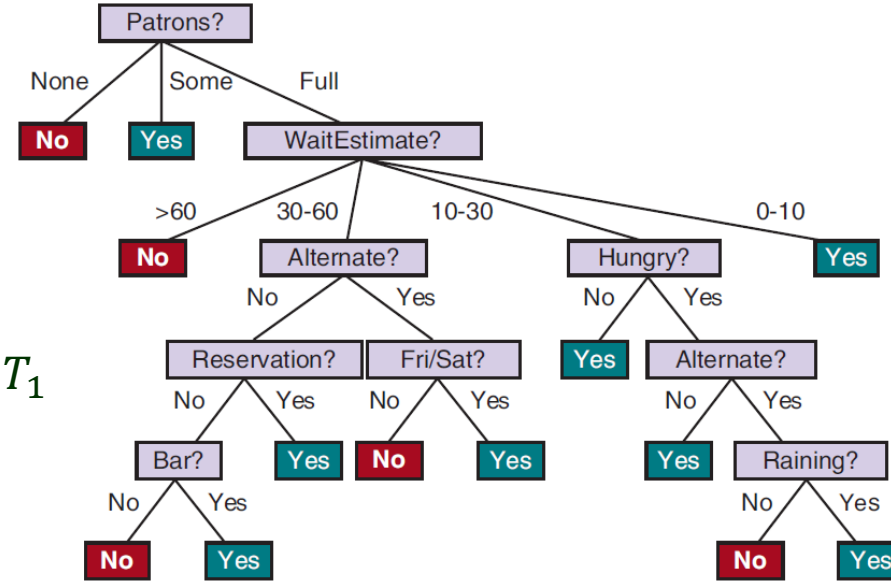


T_2

Output Decision Tree

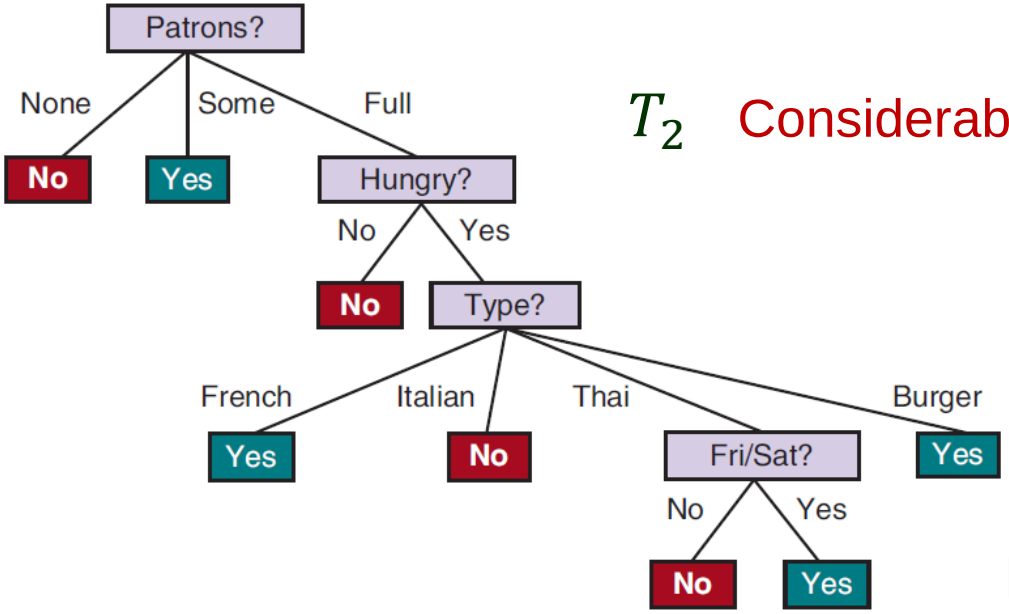


T_2

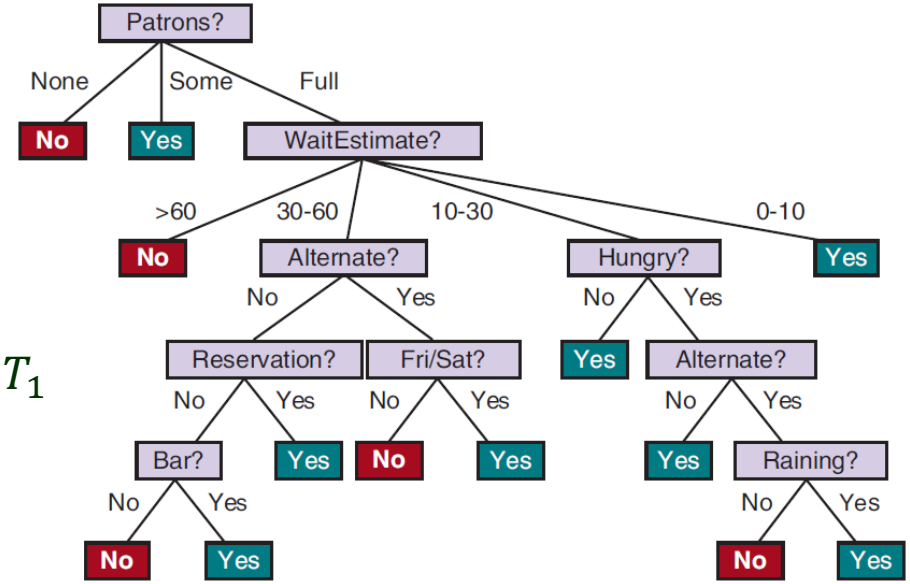


Original decision tree T_1

Output Decision Tree

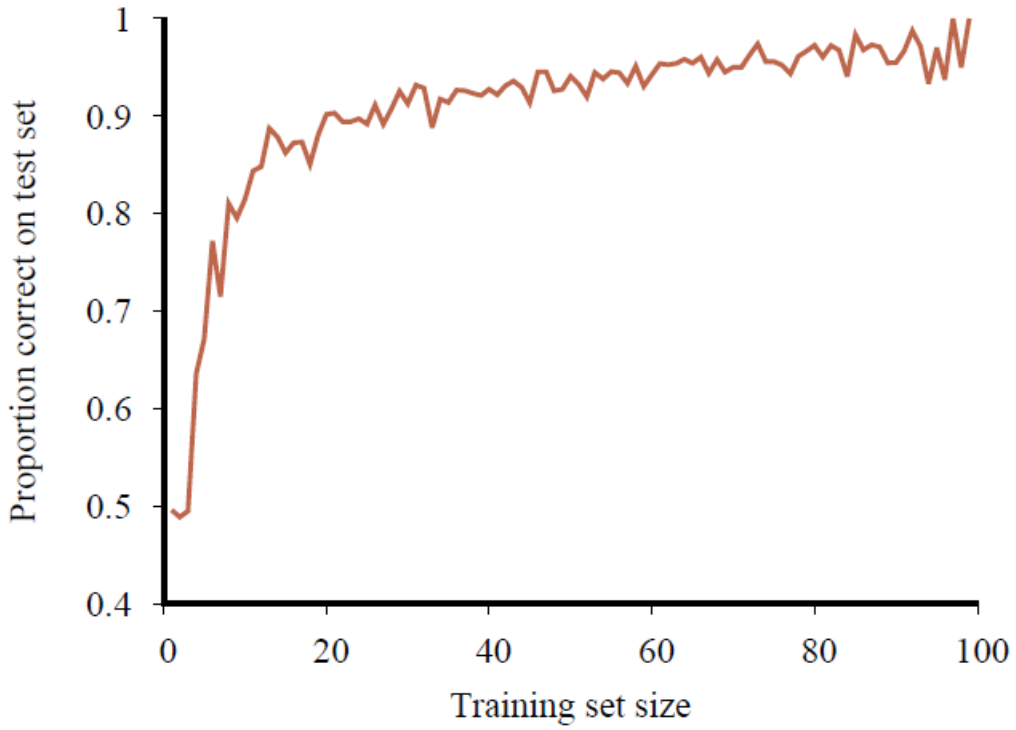


T_2 Considerably simpler!



Original decision tree T_1

Learning Curve



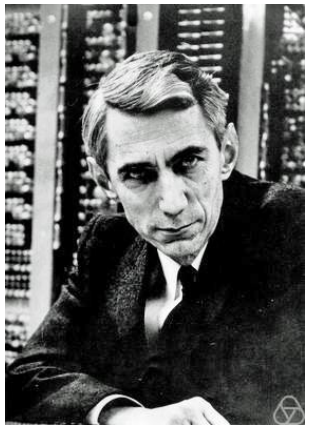
"happy graph"

- 100 randomly generated examples in the restaurant domain.
- Split them randomly into a training set and a test set.
  - Split 20 times for each size (1 – 99) of the training set.
  - Average the results of the 20 trials.

III. Entropy and Attribute Choice

Importance of an attribute is measured using the notion of information gain defined in terms of (Shannon) *entropy*.

measure of the uncertainty of a random variable:
the more information, the less entropy.

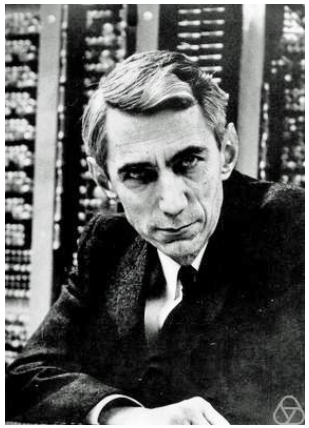


Claude Shannon (MIT)

"Father of information theory"
National Medal of Science (1966)

III. Entropy and Attribute Choice

Importance of an attribute is measured using the notion of information gain defined in terms of (Shannon) *entropy*.



Claude Shannon (MIT)

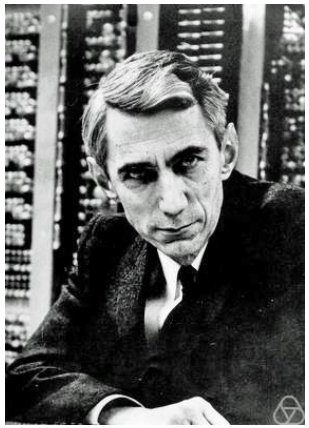
"Father of information theory"
National Medal of Science (1966)

measure of the uncertainty of a random variable:
the more information, the less entropy.

- A fair coin has 1 bit of entropy as it is equally likely to come up heads and tails when flipped.

III. Entropy and Attribute Choice

Importance of an attribute is measured using the notion of information gain defined in terms of (Shannon) *entropy*.



Claude Shannon (MIT)

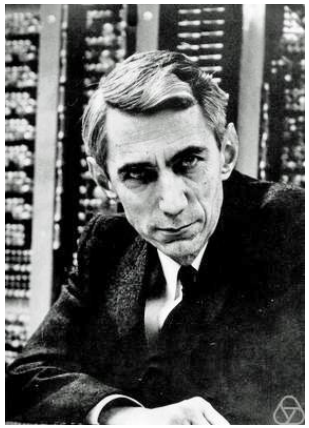
"Father of information theory"
National Medal of Science (1966)

measure of the uncertainty of a random variable:
the more information, the less entropy.

- A fair coin has 1 bit of entropy as it is equally likely to come up heads and tails when flipped.
- A four-sided die has 2 bits of entropy since there are 2^2 equally likely choices.

III. Entropy and Attribute Choice

Importance of an attribute is measured using the notion of information gain defined in terms of (Shannon) *entropy*.



Claude Shannon (MIT)

"Father of information theory"
National Medal of Science (1966)

measure of the uncertainty of a random variable:
the more information, the less entropy.

- A fair coin has 1 bit of entropy as it is equally likely to come up heads and tails when flipped.
- A four-sided die has 2 bits of entropy since there are 2^2 equally likely choices.
- An unfair coin that comes up heads 99% of the time would have an entropy measure that is close to zero.

Entropy

Random variable V with values v_k having probability $P(v_k)$.

Entropy of V:

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)}$$

$$= -\sum_k P(v_k) \log_2 P(v_k)$$

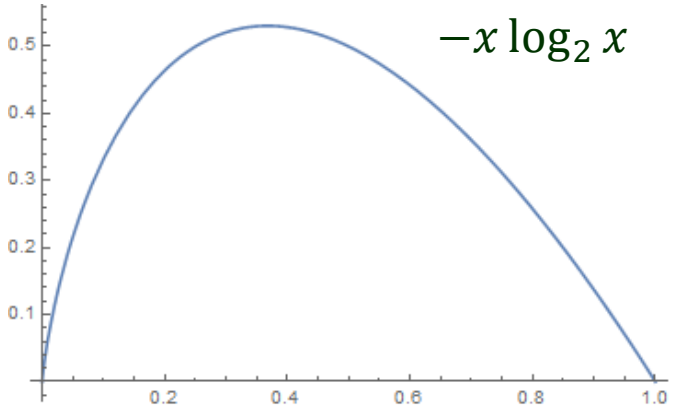
Entropy

Random variable V with values v_k having probability $P(v_k)$.

Entropy of V:

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)}$$

$$= -\sum_k P(v_k) \log_2 P(v_k)$$



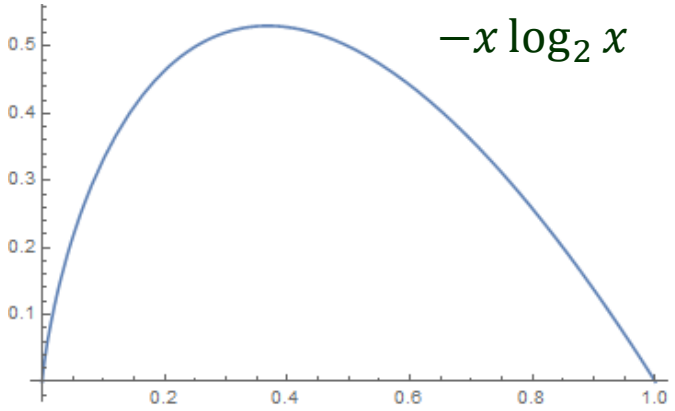
Entropy

Random variable V with values v_k having probability $P(v_k)$.

Entropy of V:

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)}$$

$$= -\sum_k P(v_k) \log_2 P(v_k)$$



Entropy measures the *expected amount of information* conveyed by identifying the outcome of a random trial.

Entropy Examples

- A fair coin

$$H(\textit{Fair}) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1 \text{ (bit)}$$

There is *maximum* surprise in this case.

Entropy Examples

- A fair coin

$$H(\textit{Fair}) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1 \text{ (bit)}$$

There is *maximum* surprise in this case.

- A coin with known outcome (e.g., head)

$$H(\textit{Head}) = -1 \cdot \log_2 1 = 0 \text{ (bit)}$$

There is no uncertainty at all – *no information*.

Entropy Examples

- A fair coin

$$H(\textit{Fair}) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1 \text{ (bit)}$$

There is *maximum* surprise in this case.

- A coin with known outcome (e.g., head)

$$H(\textit{Head}) = -1 \cdot \log_2 1 = 0 \text{ (bit)}$$

There is no uncertainty at all – *no information*.

- A loaded coin with 99% heads

$$H(\textit{Loaded}) = -(0.99 \log_2 0.99 + 0.01 \log_2 0.01) \approx 0.08 \text{ (bits)}$$

Entropy Examples

- A fair coin

$$H(\textit{Fair}) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1 \text{ (bit)}$$

There is *maximum* surprise in this case.

- A coin with known outcome (e.g., head)

$$H(\textit{Head}) = -1 \cdot \log_2 1 = 0 \text{ (bit)}$$

There is no uncertainty at all – *no information*.

- A loaded coin with 99% heads

$$H(\textit{Loaded}) = -(0.99 \log_2 0.99 + 0.01 \log_2 0.01) \approx 0.08 \text{ (bits)}$$

- A four-sided die

$$H(\textit{Die4}) = -(4 \cdot 0.25 \log_2 0.25) = 2 \text{ (bits)}$$

Entropy of a Boolean Variable

A Boolean random variable that is true with probability q.

$$B(q) = -(q \log_2 q + (1-q) \log_2 (1-q))$$

Entropy of a Boolean Variable

A Boolean random variable that is true with probability q.

$$B(q) = -(q \log_2 q + (1-q) \log_2 (1-q)) \quad \Rightarrow \quad B(q) = B(1-q)$$

Entropy of a Boolean Variable

A Boolean random variable that is true with probability q.

$$B(q) = -(q \log_2 q + (1-q) \log_2 (1-q)) \quad \Rightarrow \quad B(q) = B(1-q)$$

$$H(\textit{Loaded}) = B(0.99) \approx 0.08$$

Entropy of a Boolean Variable

A Boolean random variable that is true with probability q.

$$B(q) = -(q \log_2 q + (1-q) \log_2 (1-q)) \quad \Rightarrow \quad B(q) = B(1-q)$$

$$H(\textit{Loaded}) = B(0.99) \approx 0.08$$

- A training set contains p positive examples and n negative examples.

Entropy of a Boolean Variable

A Boolean random variable that is true with probability q.

$$B(q) = -(q \log_2 q + (1-q) \log_2 (1-q)) \quad \Rightarrow \quad B(q) = B(1-q)$$

$$H(\textit{Loaded}) = B(0.99) \approx 0.08$$

- A training set contains p positive examples and n negative examples.

$$H(\textit{Output}) = B\left(\frac{p}{p+n}\right)$$

Entropy of a Boolean Variable

A Boolean random variable that is true with probability q.

$$B(q) = -(q \log_2 q + (1-q)\log_2(1-q)) \quad \Rightarrow \quad B(q) = B(1-q)$$

$$H(\textit{Loaded}) = B(0.99) \approx 0.08$$

- A training set contains p positive examples and n negative examples.

$$H(\textit{Output}) = B\left(\frac{p}{p+n}\right)$$

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Entropy of a Boolean Variable

A Boolean random variable that is true with probability q.

$$B(q) = -(q \log_2 q + (1-q)\log_2(1-q)) \quad \Rightarrow \quad B(q) = B(1-q)$$

$$H(\textit{Loaded}) = B(0.99) \approx 0.08$$

- A training set contains p positive examples and n negative examples.

$$H(\textit{Output}) = B\left(\frac{p}{p+n}\right)$$

$$p = n = 6$$

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Entropy of a Boolean Variable

A Boolean random variable that is true with probability q.

$$B(q) = -(q \log_2 q + (1-q)\log_2(1-q)) \quad \Rightarrow \quad B(q) = B(1-q)$$

$$H(\textit{Loaded}) = B(0.99) \approx 0.08$$

- A training set contains p positive examples and n negative examples.

$$H(\textit{Output}) = B\left(\frac{p}{p+n}\right)$$

$$p = n = 6$$

$$\Downarrow$$

$$B(0.5) = 1$$

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Choice of an Attribute

- An attribute A has d distinct values.

Choice of an Attribute

- An attribute A has d distinct values.
- A divides the training set E into subsets $E_1, \ldots, E_d$.

Choice of an Attribute

- An attribute A has d distinct values.
- A divides the training set E into subsets $E_1, \ldots, E_d$.
  - Each subset E_k has p_k positive examples and n_k negative examples.

$$p = \sum_{k=1}^{d} p_k \quad \text{and} \quad n = \sum_{k=1}^{d} n_k$$

Choice of an Attribute

- An attribute A has d distinct values.
- A divides the training set E into subsets $E_1, \ldots, E_d$.
  - Each subset E_k has p_k positive examples and n_k negative examples.

$$p = \sum_{k=1}^{d} p_k \quad \text{and} \quad n = \sum_{k=1}^{d} n_k$$

  - This implies an additional $B\left(\frac{p_k}{p_k+n_k}\right)$ bits of information.

Choice of an Attribute

- An attribute A has d distinct values.
- A divides the training set E into subsets $E_1, \ldots, E_d$.
  - Each subset E_k has p_k positive examples and n_k negative examples.

$$p = \sum_{k=1}^{d} p_k \quad \text{and} \quad n = \sum_{k=1}^{d} n_k$$

  - This implies an additional $B\left(\frac{p_k}{p_k+n_k}\right)$ bits of information.
  - Consider a randomly chosen example from the training set.
    - Its attribute A value is in E_k with probability $(p_k + n_k)/(p + n)$.

Choice of an Attribute

- An attribute A has d distinct values.
- A divides the training set E into subsets $E_1, \ldots, E_d$.
  - Each subset E_k has p_k positive examples and n_k negative examples.

$$p = \sum_{k=1}^{d} p_k \quad \text{and} \quad n = \sum_{k=1}^{d} n_k$$

  - This implies an additional $B\left(\frac{p_k}{p_k+n_k}\right)$ bits of information.
  - Consider a randomly chosen example from the training set.
    - Its attribute A value is in E_k with probability $(p_k + n_k)/(p + n)$.
- Expected entropy remaining after testing attribute A is

$$\mathit{Remainder}(A) = \sum_{k=1}^{d} \frac{p_k+n_k}{p+n} B\left(\frac{p_k}{p_k+n_k}\right)$$

Information Gain

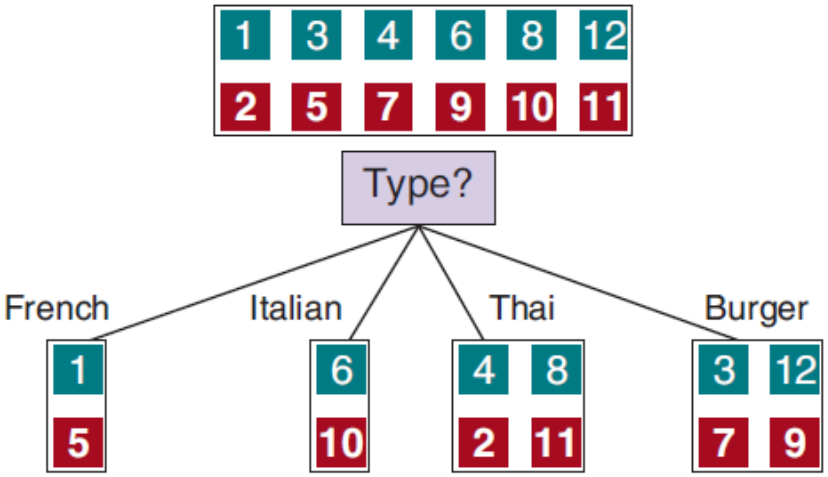
Expected reduction in entropy from the attribute test on A:

$$Gain(A) = B\left(\frac{p}{p+n}\right) - Remainder(A)$$

$$= B\left(\frac{p}{p+n}\right) - \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B\left(\frac{p_k}{p_k + n_k}\right)$$

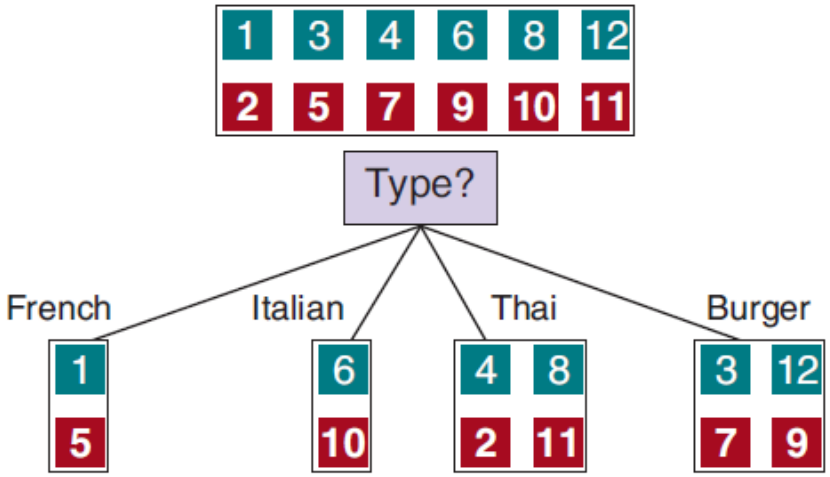
Choose the attribute A among the remaining attributes to maximize $Gain(A)$.

Information Gain



$$Gain(A) = B\left(\frac{p}{p+n}\right) - \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B\left(\frac{p_k}{p_k + n_k}\right)$$

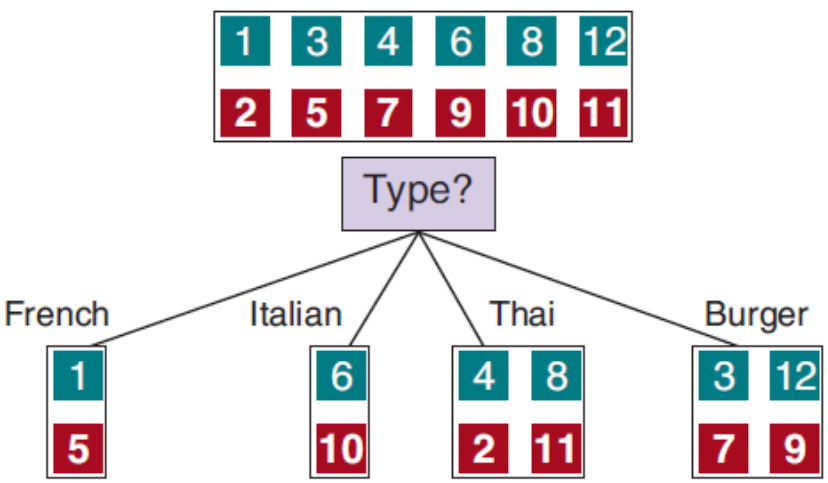
Information Gain

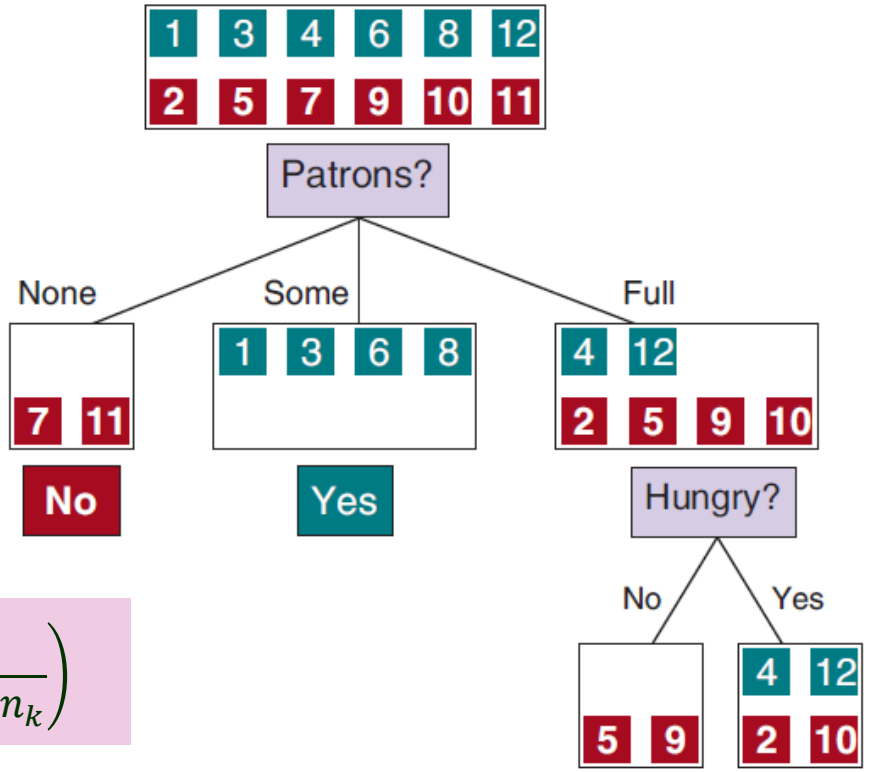


$$Gain(A) = B\left(\frac{p}{p+n}\right) - \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B\left(\frac{p_k}{p_k + n_k}\right)$$

$$Gain(Type) = B\left(\frac{1}{2}\right) - \left(\frac{2}{12}B\left(\frac{1}{2}\right) + \frac{2}{12}B\left(\frac{1}{2}\right) + \frac{4}{12}B\left(\frac{2}{4}\right) + \frac{4}{12}B\left(\frac{2}{4}\right)\right) = 0 \text{ bits}$$

Information Gain

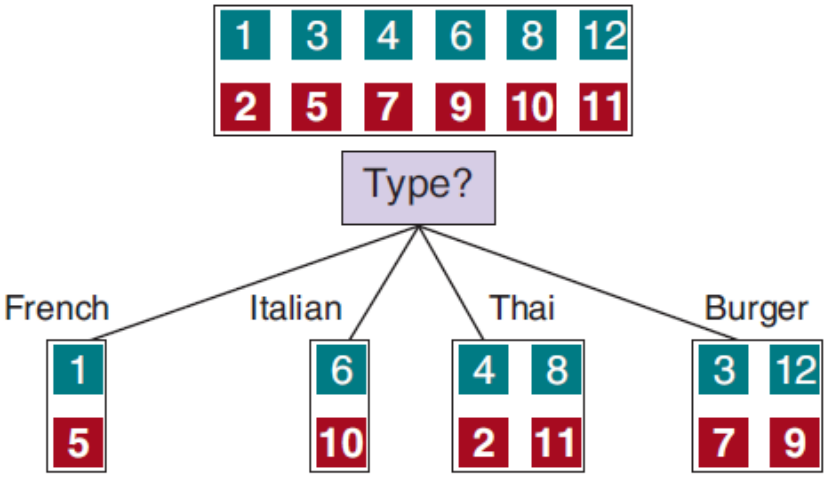


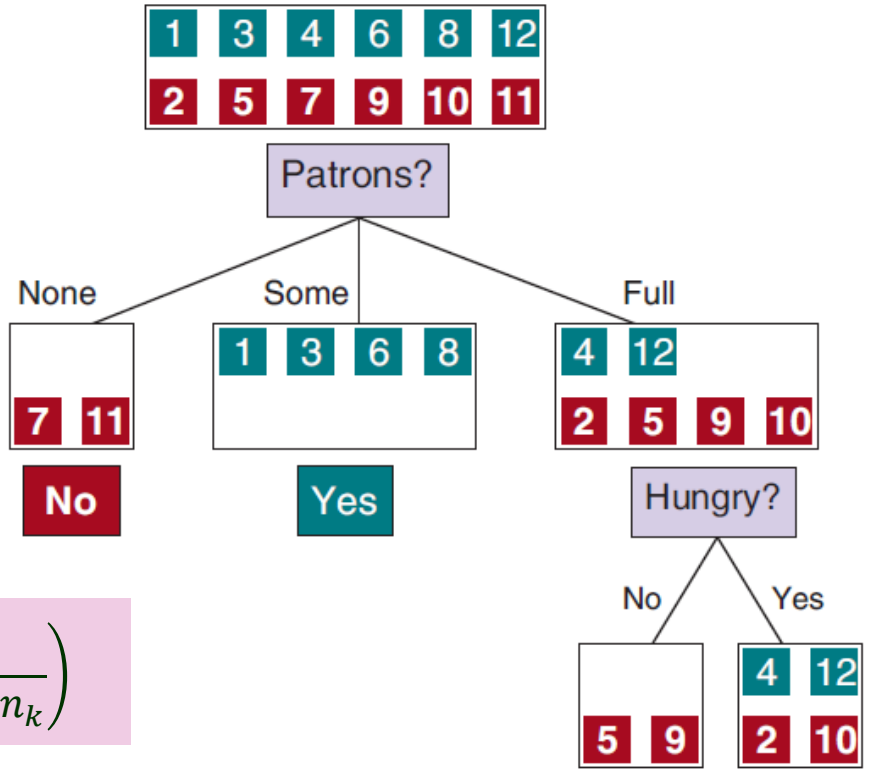


$$Gain(A) = B\left(\frac{p}{p+n}\right) - \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B\left(\frac{p_k}{p_k + n_k}\right)$$

$$Gain(Type) = B\left(\frac{1}{2}\right) - \left(\frac{2}{12}B\left(\frac{1}{2}\right) + \frac{2}{12}B\left(\frac{1}{2}\right) + \frac{4}{12}B\left(\frac{2}{4}\right) + \frac{4}{12}B\left(\frac{2}{4}\right)\right) = 0 \text{ bits}$$

Information Gain



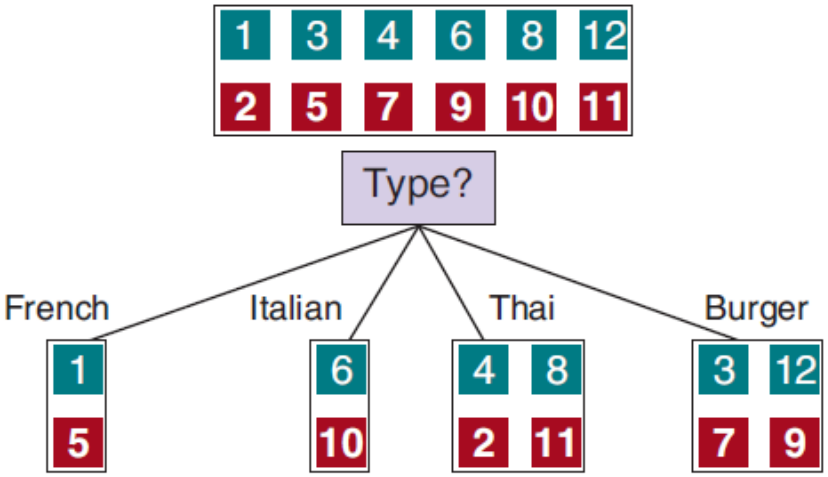


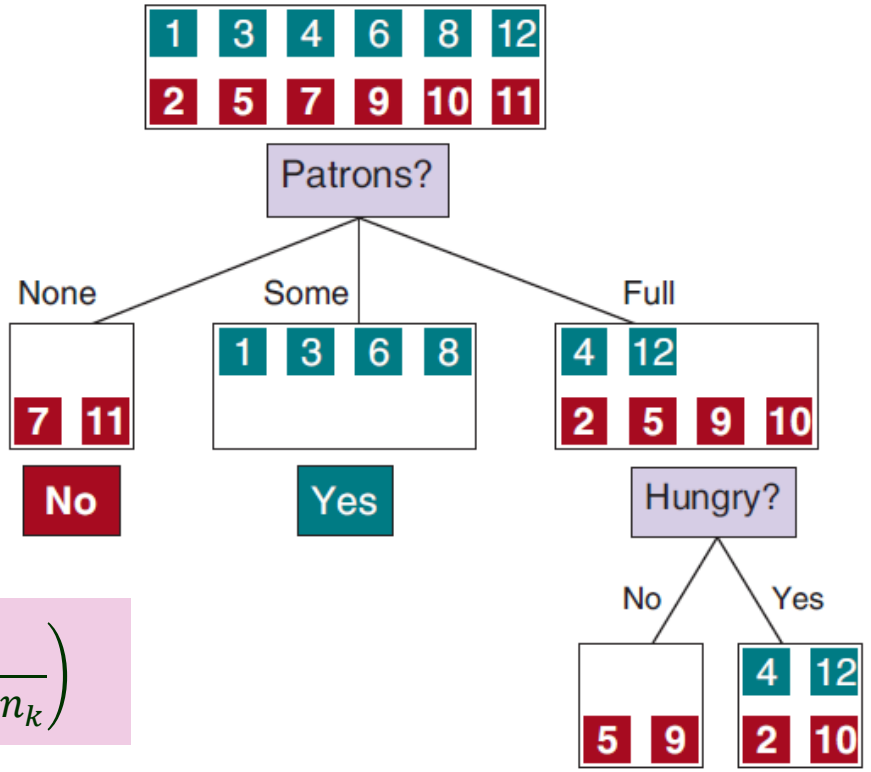
$$Gain(A) = B\left(\frac{p}{p+n}\right) - \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B\left(\frac{p_k}{p_k + n_k}\right)$$

$$Gain(Type) = B\left(\frac{1}{2}\right) - \left(\frac{2}{12}B\left(\frac{1}{2}\right) + \frac{2}{12}B\left(\frac{1}{2}\right) + \frac{4}{12}B\left(\frac{2}{4}\right) + \frac{4}{12}B\left(\frac{2}{4}\right)\right) = 0 \text{ bits}$$

$$Gain(Patrons) = B\left(\frac{1}{2}\right) - \left(\frac{2}{12}B\left(\frac{0}{2}\right) + \frac{4}{12}B\left(\frac{4}{4}\right) + \frac{6}{12}B\left(\frac{2}{6}\right)\right) \approx 0.541 \text{ bits}$$

Information Gain





$$Gain(A) = B\left(\frac{p}{p+n}\right) - \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B\left(\frac{p_k}{p_k + n_k}\right)$$

$$Gain(Type) = B\left(\frac{1}{2}\right) - \left(\frac{2}{12}B\left(\frac{1}{2}\right) + \frac{2}{12}B\left(\frac{1}{2}\right) + \frac{4}{12}B\left(\frac{2}{4}\right) + \frac{4}{12}B\left(\frac{2}{4}\right)\right) = 0 \text{ bits}$$

$$Gain(Patrons) = B\left(\frac{1}{2}\right) - \left(\frac{2}{12}B\left(\frac{0}{2}\right) + \frac{4}{12}B\left(\frac{4}{4}\right) + \frac{6}{12}B\left(\frac{2}{6}\right)\right) \approx 0.541 \text{ bits}$$