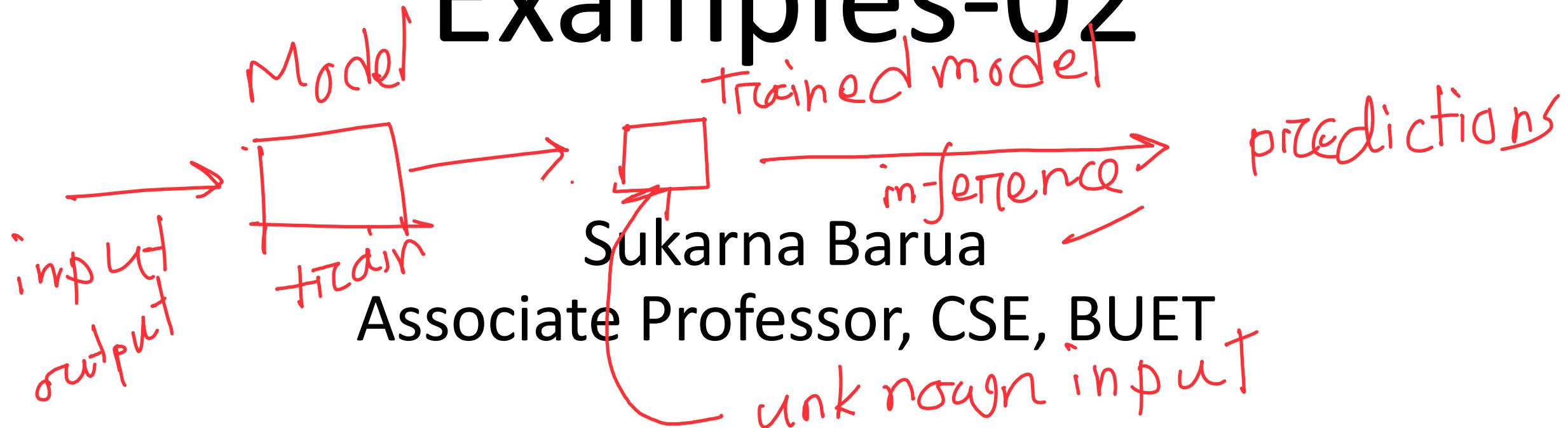


Chapter 18 (AIAMA)

Learning From Examples-02





Decision Tree

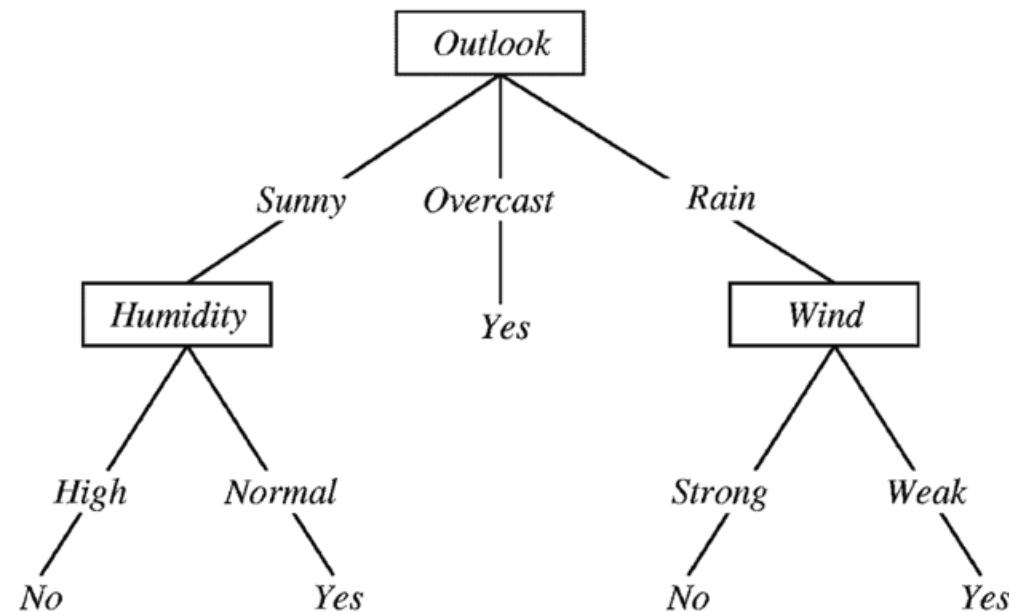
↓
Interpretable

- A decision tree:

- Represents a function that takes as input a vector of attribute values
- Returns a “decision”—a single output value.
- The input and output values can be discrete or continuous.
- For now we will concentrate on problems where
 - Inputs have discrete values
 - Output has exactly two possible values; this is Boolean classification, where each example input will be classified as *true* (a **positive** example) or *false* (a **negative** example).

Decision Tree

- A decision tree for rain forecasting.
 - Input: {Outlook=Sunny, Humidity=Normal, Wind=Strong}
 - Output: Yes [Rainy]



Decision Tree

- In a decision tree:

- Each **internal node** in the tree corresponds to a test of the value of one of the input attributes, A_i
- The **branches** from the node are labeled with the possible values of the attribute, $A_i = v_{ik}$.
- Each **leaf node** in the tree is marked with a outcome to be returned by the function.

Decision Tree: Example

- **Problem:** Build a decision tree to decide whether to wait at a restaurant.
- **Learning Goal:**
 - WillWait - to decide whether to wait for a table at a restaurant.
 - Goal is binary valued (i.e., binary classification task)
 - Values: {Yes, No}

Decision Tree: Example

- **Input Attributes**

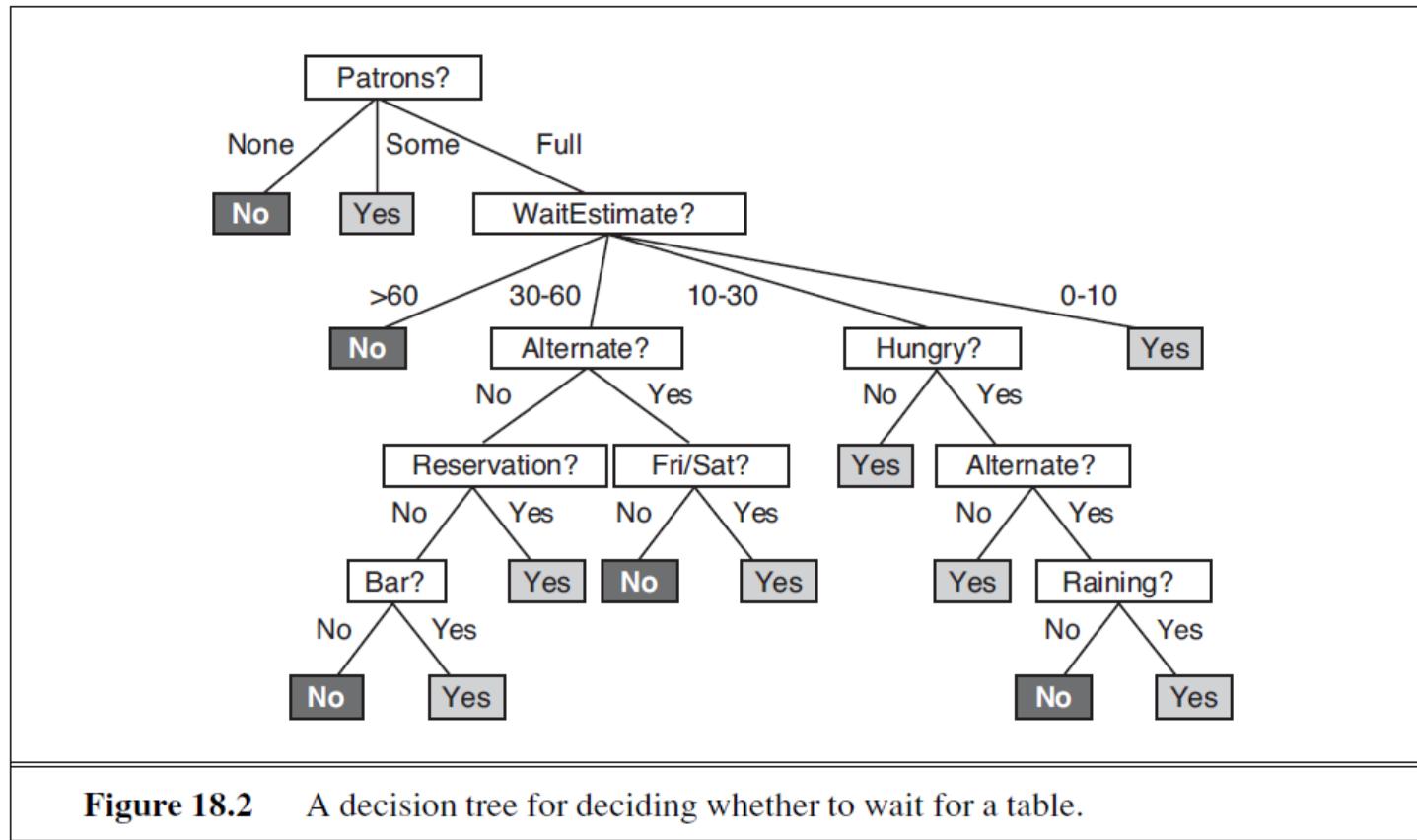
1. *Alternate*: whether there is a suitable alternative restaurant nearby.
2. *Bar* : whether the restaurant has a comfortable bar area to wait in.
3. *Fri/Sat*: *true* on Fridays and Saturdays, *false* otherwise.
4. *Hungry*: whether we are hungry.
5. *Patrons*: how many people are in the restaurant (values are None, Some, and Full).
6. *Price*: *the restaurant's price range (\$, \$\$, \$\$\$)*.

Decision Tree: Example

- **Input Attributes**
 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*: the wait estimated by the host (0–10 minutes, 10–30, 30–60, or >60).
- Note that every variable has a small set of possible values; the value of *WaitEstimate*, for example, is not an integer, rather it is one of the four discrete values 0–10, 10–30, 30–60, or >60.

Decision Tree: Example

- An example decision tree for the restaurant problem



Expressiveness of Decision Trees

- In a Boolean decision tree, the goal attribute is true if and only if the input attributes satisfy one of the paths leading to a leaf with value true.
- $Goal \Leftrightarrow (Path1 \vee Path2 \vee \dots)$, where each $Path$ is a conjunction of attribute-value tests required to follow that path.
- Thus, the whole expression is equivalent to *disjunctive normal form*, which means that any function in propositional logic can be expressed as a decision tree.

Expressiveness of Decision Trees

- Any Boolean function can be represented by a decision tree.
 - Consider the following Boolean function: $f(A, B) = A + B$.
 - Draw a decision tree for the function:

Expressiveness of Decision Trees

- For a wide variety of problems, the decision tree format yields a nice, concise result.
- But some functions cannot be represented concisely:
 - For example, the majority function, which returns true if and only if more than half of the inputs are true, requires an exponentially large decision tree.
- Decision trees are good for some kinds of functions and bad for others.

Expressiveness of Decision Trees

- How many different decision trees can be obtained for a Boolean function with n variables?
 - A truth table over n attributes has 2^n rows, one for each combination of values of the attributes.
 - There are 2^{2^n} different functions.
 - Each function can be represented by a decision tree. Hence, number of decision trees at least 2^{2^n} [why more?]

Expressiveness of Decision Trees

- How many different decision trees can be obtained for a Boolean function with n variables?
 - For 10 Boolean attributes of our restaurant problem there are 2^{1024} or about 10^{308} different functions to choose from.
 - Number of possible decision trees $\geq 10^{308}$

Inducing Decision Trees

- **Training data:** A set of examples where each example a pair $(\underline{x}, \underline{y})$ pair where \underline{x} is a vector of values for the input attributes, \underline{y} is a single Boolean output value.

Example	Input Attributes										Goal <i>WillWait</i>
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	
x_1	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0–10	$y_1 = Yes$
x_2	Yes	No	No	Yes	Full	\$	No	No	Thai	30–60	$y_2 = No$
x_3	No	Yes	No	No	Some	\$	No	No	Burger	0–10	$y_3 = Yes$
x_4	Yes	No	Yes	Yes	Full	\$	Yes	No	Thai	10–30	$y_4 = Yes$
x_5	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	$y_5 = No$
x_6	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0–10	$y_6 = Yes$
x_7	No	Yes	No	No	None	\$	Yes	No	Burger	0–10	$y_7 = No$
x_8	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0–10	$y_8 = Yes$
x_9	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	$y_9 = No$
x_{10}	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10–30	$y_{10} = No$
x_{11}	No	No	No	No	None	\$	No	No	Thai	0–10	$y_{11} = No$
x_{12}	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30–60	$y_{12} = Yes$

Figure 18.3 Examples for the restaurant domain.

Inducing Decision Trees

✓ Build a decision tree that is -

- ✓ Consistent with the examples [*Not always expected though, generalization may suffer*]
- ✓ Is as small as possible. [*Occam's razor*]

Inducing Decision Trees

- **Question:** Can we always find a consistent decision tree given a set of training examples?.
- **Answer:** Yes.
 - However, if two training examples (x, y) and (x', y') have different outputs but same input attributes ($x = x'$), then a consistent decision tree is not possible.

Inducing Decision Trees

 Unfortunately, it is an intractable problem to find the smallest consistent tree; there is no way to efficiently search through *hypotheses space*.

- An NP-hard problem!
- What can we do if cannot find the smallest decision tree?
- **Solution:** Use heuristics to find a closest one.
 - With some simple heuristics, we can find a good approximate solution: a small (*but not the smallest*) consistent tree.
 - This is a greedy approach. [Remember what is *greedy*]

Decision Tree Learning

- **Greedy approach to build a decision tree:**

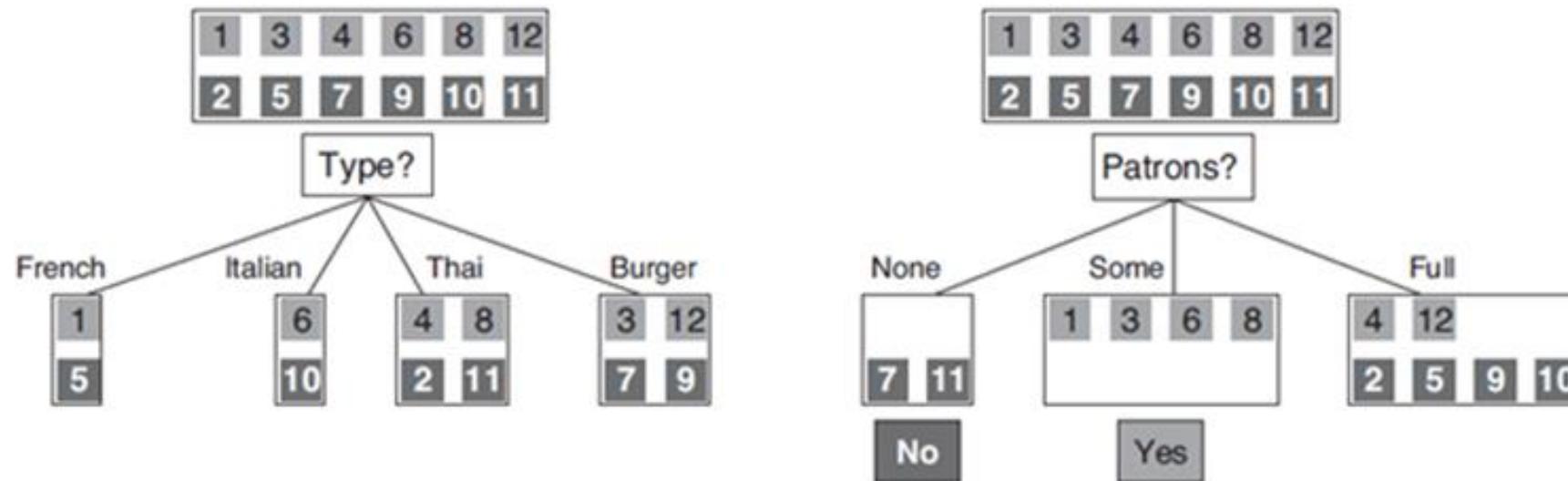
- Start with empty decision tree.
- Select an attribute to test at the next level [node]:
 - *Always select the most important attribute to test first. [greedy strategy]*
- The test creates new branches and divides the problem into smaller subproblems.
- Recurse on each child (created for each branch)

Decision Tree Learning

- **Greedy strategy:** *Always select the most important attribute to test first.*
 - *Most important attribute* implies the one that makes the most difference to the classification of an example. [*Get leaves as early as possible*]
 - Get correct classification with a small number of test
 - All paths in the tree will be short
 - Tree as a whole will be shallow.
- *Above greedy strategy is a local optimal choice, may not necessarily leads to the globally smallest tree!*

Decision Tree Learning

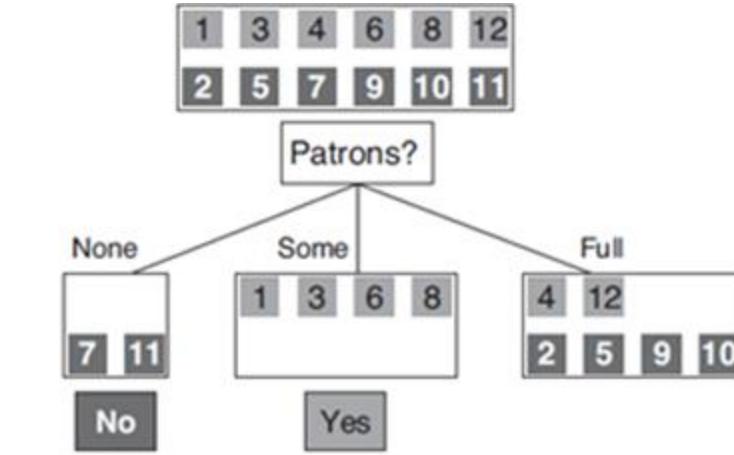
- Which attribute to test at root? Type vs Patron?
 - Type: all subsets (i.e., branches) needs further exploration.
 - Patron: two branches become leaves, only one needs further exploration.



Decision Tree Learning Algorithm

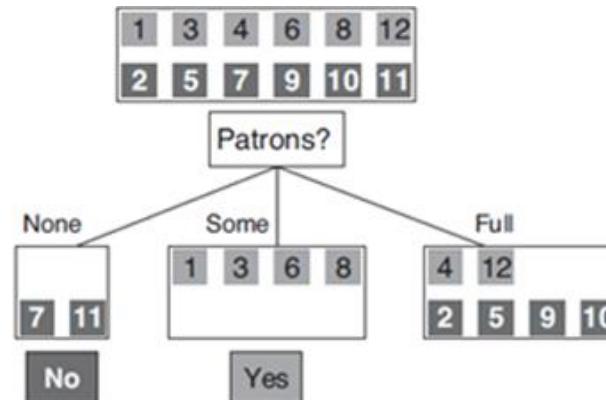
■ Decision tree construction:

- Step 1: Test an attribute at each node.
- Step 2: Partition the examples according to values and create child nodes with relevant examples.
- Step 3: Now consider child node for further tests of attributes except the one which have already been tested in the hierarchy (*recursive operation*)



Decision Tree Learning Algorithm

- **Four cases to consider at a child node:**
 - Case 1: All examples are positive (or all negative) → we are done; we can answer Yes or No. [e.g., *Patron=None*]
 - Case 2: Some positive and some negative examples → then choose the best attribute to split them recursively. [e.g., *Patron=Full*]



Decision Tree Learning Algorithm

- **Four cases to consider at a child node:**
 - Case 3: No examples left → No example has been observed for this combination of attribute values, and we return a default value (e.g., *plurality of parent node*)
 - Case 4: No attributes left, but both positive and negative examples remain → These examples have exactly the same description, but different classifications. This can happen because there is an error or **noise** in the data.
 - *In this case return the plurality classification of the remaining examples.* [tree will not be consistent]

Decision Tree Learning Algorithm

■ Algorithm pseudocode

```
function DECISION-TREE-LEARNING(examples, attributes, parent-examples) returns  
a tree  
  
    if examples is empty then return PLURALITY-VALUE(parent-examples)  
    else if all examples have the same classification then return the classification  
    else if attributes is empty then return PLURALITY-VALUE(examples)  
    else  
         $A \leftarrow \operatorname{argmax}_{a \in \text{attributes}} \text{IMPORTANCE}(a, \text{examples})$   
        tree  $\leftarrow$  a new decision tree with root test A  
        for each value  $v_k$  of A do  
            exs  $\leftarrow \{e : e \in \text{examples} \text{ and } e.A = v_k\}$   
            subtree  $\leftarrow$  DECISION-TREE-LEARNING(exs, attributes - A, examples)  
            add a branch to tree with label (A =  $v_k$ ) and subtree subtree  
    return tree
```

- Function PLURAITY-VALUE selects the most common class/output among the examples

Choosing the Most Important Attribute

- **Which attribute is the most important now?**
 - **Perfect attribute:** One that splits into subsets where each subset contain either all positive or all negative examples. [*all branches become leaf nodes*]
 - **Useless attribute:** One that splits into subsets where each subset contain fairly equal mix of positive and negative examples. [*all branches need recursive exploration*]

Choosing the Most Important Attribute

- Perfect vs. useless? How to measure?
 - A formal measure of perfect vs useless: **Entropy**
 - The fundamental quantity in information theory (*Shannon and Weaver, 1949*).
 - Entropy is a measure of the uncertainty of a random variable.
 - Acquisition of information corresponds to a reduction in entropy.
 - A random variable with only one value—a coin that always comes up heads—has no uncertainty and thus its entropy is defined as zero; thus, we gain no information by observing its value.

Choosing the Most Important Attribute

- **Entropy:** Average number of bits per symbol to encode information.
 - The roll of a fair *four*-sided die has 2 bits of entropy, because it takes two bits to describe one of four equally probable choices.
 - An unfair coin that comes up heads 99% of the time.
 - This coin has less uncertainty than the fair coin—if we guess heads we’ll be wrong only 1% of the time—it’s entropy measure should be close to zero, but positive.

Entropy Measure

- In general, the entropy $H(V)$ of a random variable V with values v_k , each with probability $P(v_k)$, is defined as :

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

Entropy Measure

- Verify that entropies measures are correct.

We can check that the entropy of a fair coin flip is indeed 1 bit:

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

If the coin is loaded to give 99% heads, we get

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

Entropy Measure

- Define $B(q)$ as the entropy of a Boolean random variable that is true with probability q :

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

- If a training set contains p positive examples and n negative examples, then the entropy of the goal attribute on the whole set is:

$$H(\text{Goal}) = B\left(\frac{p}{p+n}\right).$$

Entropy Measure in Decision Tree

- **Decision tree contest:** Entropy represents an impurity measure of the set.
 - A set with 5 positive and 5 negative examples: Most impure, entropy should be highest.
 - A set with 10 positive and 0 negative examples: Purest, entropy should be the lowest.

Entropy Before Split

- A set with p positive and n negative examples.

- Entropy of the set is: $B\left(\frac{p}{p+n}\right)$

Entropy After Splitting

- An attribute A with d distinct values divides the training set E into subsets E_1, E_2, \dots, E_d .
- Each subset E_k has p_k positive examples and n_k negative examples, with entropy of $B(p_k/(p_k + n_k))$ bits of information. [E_k contains examples with $A = k$ th value]
- A randomly chosen example from the training set has the k th value for the attribute with probability $(p_k + n_k)/(p + n)$, so the expected entropy (*weighted average*) of the d subsets after splitting on attribute A is:

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

Information Gain

- **Information Gain:** The amount of information gain is the amount of entropy reduction after the split on attribute A.
- Assume E_{BS} = Entropy before split, E_{AS} = Entropy after split [*weighted average*]
 - Hence, information gain ($= \text{reduction of entropy}$):

$$\text{Gain} = E_{BS} - E_{AS} \quad [\text{reduction of entropy}]$$

- Hence, this is simply:

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

Information Gain

- **Information Gain:** $Gain(A) = B\left(\frac{p}{p+n}\right) - Remainder(A)$
- Compute the information gain for the attributes Patron and Type:

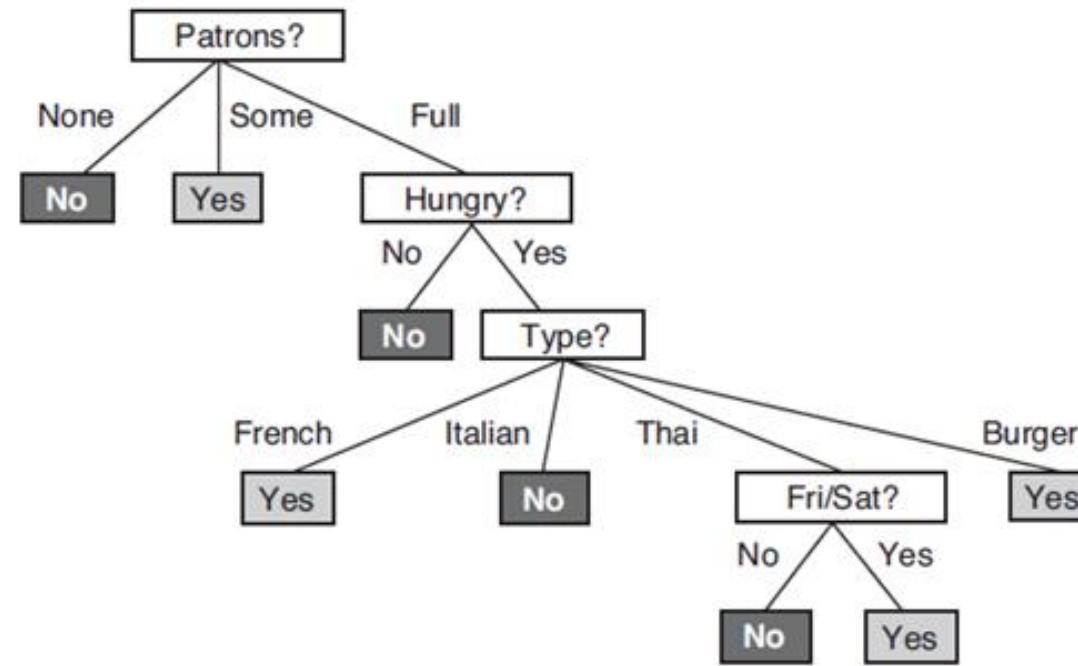
$$Gain(Patrons) = 1 - \left[\frac{2}{12}B\left(\frac{0}{2}\right) + \frac{4}{12}B\left(\frac{4}{4}\right) + \frac{6}{12}B\left(\frac{2}{6}\right) \right] \approx 0.541 \text{ bits},$$

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

- Patron is a better attribute than Gain! Hence, choose Patron over Type.
- *Choose the attribute which gives the highest information gain!*

Final Decision Tree

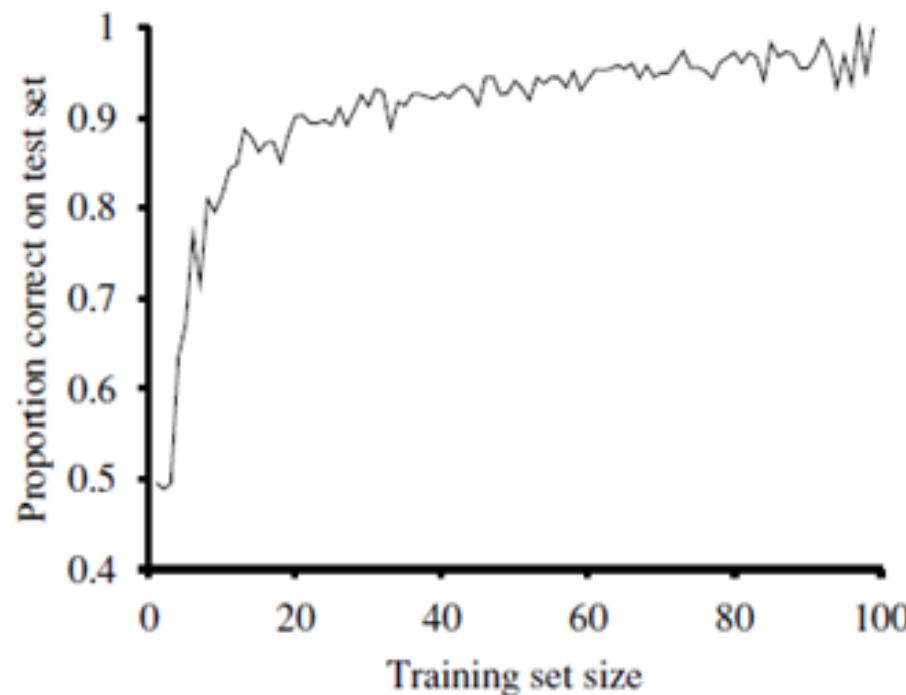
- Final decision tree constructed from given examples.



- **Question:** What is the information gain for Type attribute at level 3?

Decision Tree Learning Curve

- Experiment with 100 examples: Construct decision trees
 - Split into train and test (e.g., 1 and 99, 2 and 98, etc.)
 - Random split 20 times and report average accuracy on test set
 - Note: As the training size grows, accuracy increases.



Overfitting

- **Overfitting:** Too much importance on every training example
 - Complex tree
 - Suffers generalization: very low training error, but very high test error.
 - Misses important concepts
- A consistent decision tree over training data may result in a complex tree
 - Noisy data may also induce complexity in the tree

Overfitting Solution

- **Pre-prune:** Prune the tree before it gets large
 - Early stopping: limit depth during tree construction
- **Post-pruning:** Prune the tree after construction
 - Remove irrelevant nodes
 - Replace internal nodes with most common class [*Only replace if test error do not increase*]

Decision Tree Issues

- **Missing values:** Some examples have missing value in some attributes
 - Solution: Replace missing value with mean of the attribute over the entire dataset
- **Continuous-valued attribute:** For example, height is continuous-valued.
 - Convert to categorical attribute:
 - Height $> 40\text{cm}$: *Tall* and Height $< 40\text{cm}$: *Short*

8. (a) Consider the following data set comprised of three binary input attributes (A_1, A_2 , and A_3) and one binary output y :

Example	A_1	A_2	A_3	Output y
x_1	1	0	0	0
x_2	1	0	1	0
x_3	0	1	0	0
x_4	1	1	1	1
x_5	1	1	0	1

Construct a decision tree for these data. Show the entropy and information-gain computations made to determine the attribute to split at each node.

Suppose a bank wants to decide whether to approve a small loan (output y). They use three simple binary attributes about an applicant:

- A_1 = Has a stable job? (1 = Yes, 0 = No)
- A_2 = Has no unpaid loans? (1 = Yes, 0 = No)
- A_3 = Owns a house? (1 = Yes, 0 = No)
- Output y = Loan approved? (1 = Yes, 0 = No)

train the
dataset

Entropy: metric for gaining information from a dataset.

$$\rightarrow E(\{x_1, x_2, x_3, x_4, x_5\}) = - \left[\frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5} \right]$$

$$= 0.971$$

$$- [p \log_2 p + (1-p) \log_2 (1-p)]$$

A₁ A₂ A₃

Example

	A_1	A_2	A_3	Output y
x_1	1	0	0	0
x_2	1	0	1	0
x_3	0	1	0	0
x_4	1	1	1	1
x_5	1	1	0	1

$$E\{x_3\} = - \left(1 \log_2 1 + 0 \log_2 0 \right) = 0$$

$$E_2 \left\{ \frac{x_1 x_2}{x_4 x_5} \right\} = \left(\frac{2}{4} \log_2 \frac{2}{4} + \frac{2}{4} \log_2 \frac{2}{4} \right)$$

$$= - \frac{1}{2} \times 2 \log_2 \frac{1}{2} - \frac{1}{2} \times 2 \log_2 \frac{1}{2} = 1$$

information gain after splitting A_1 :

$$= \underline{0.971} -$$

$$\left[E_1 \times \frac{1}{5} + E_2 \times \frac{4}{5} \right]$$

$$= 0.971 - \left[0 \times \frac{1}{5} + 1 \times \frac{4}{5} \right] = \boxed{0.171}$$

Example	A_1	A_2	A_3	Output y
x_1	1	0	0	0
x_2	1	0	1	0
x_3	0	1	0	0
x_4	1	1	1	1
x_5	1	1	0	1

A_2 Split

0

$\{x_1, x_2\}$

$$\sqrt{\frac{2}{5}}$$

$\{x_3, x_4, x_5\}$

$$\sqrt{\frac{3}{5}}$$

infor. gain
 $= 0.071 -$

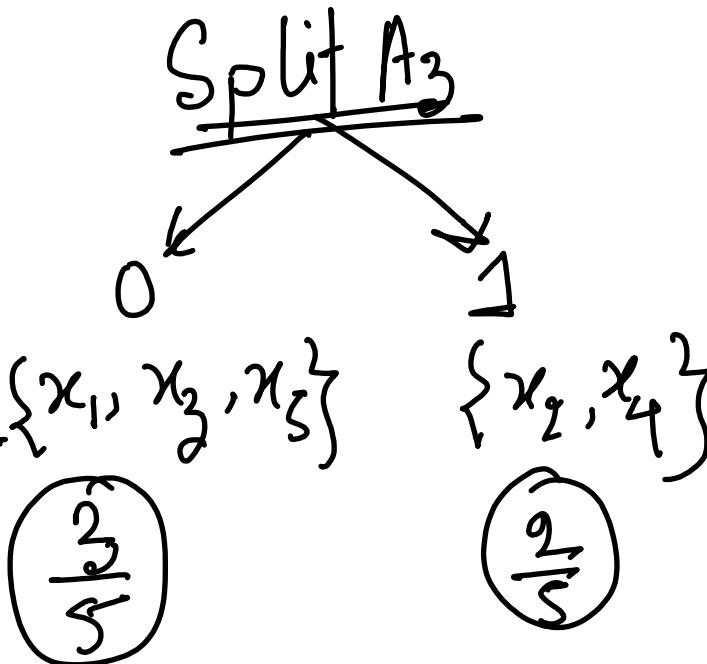
$$E_1 \{x_1, x_2\} = 0$$

$$\begin{aligned} E_2 \{x_3, x_4, x_5\} &= \\ &- \left[\frac{1}{3} \log_2 \frac{1}{3} + \frac{2}{3} \log_2 \frac{2}{3} \right] \\ &= 0.018 \end{aligned}$$

$$\begin{aligned} &\left[\frac{2}{5} \times 0 + \right. \\ &\quad \left. \frac{3}{5} \times 0.018 \right] \end{aligned}$$

$$= 0.42$$

Example	A_1	A_2	A_3	Output y
x_1	1	0	0	0
x_2	1	0	1	0
x_3	0	1	0	0
x_4	1	1	1	1
x_5	1	1	0	1



$$E_1 \{x_1, x_3, x_5\} : - \left[\frac{2}{3} \log_2 \frac{2}{3} + \frac{1}{3} \log_2 \frac{1}{3} \right]$$

$$= 0.518$$

$$E_2 \{x_2, x_4\} = 1$$

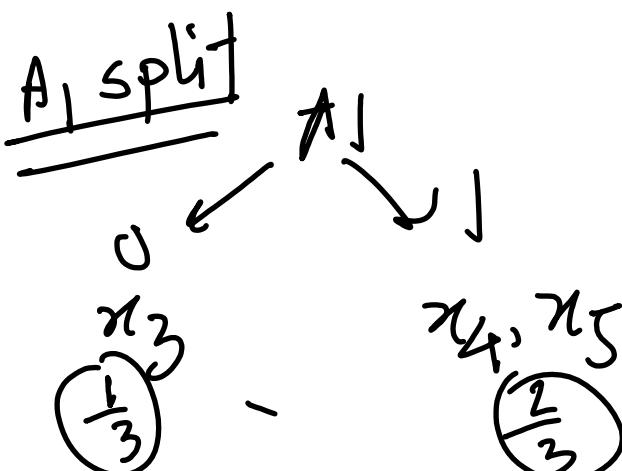
$$\begin{aligned}
 & IG(A_3) \\
 &= 0.07 \\
 &- \left[\frac{3}{5} \times 0.518 + \frac{2}{5} \times 1 \right] \\
 &= 0.02
 \end{aligned}$$

Example	A_1	A_2	A_3	Output y
x_1	1	0	0	0
x_2	1	0	1	0
x_3	0	1	0	0
x_4	1	1	1	1
x_5	1	1	0	1

$\rightarrow E(x_3 \ x_4 \ x_5)$

$$= - \left(\frac{1}{3} \log_2 \frac{1}{3} + \frac{2}{3} \log_2 \frac{2}{3} \right)$$

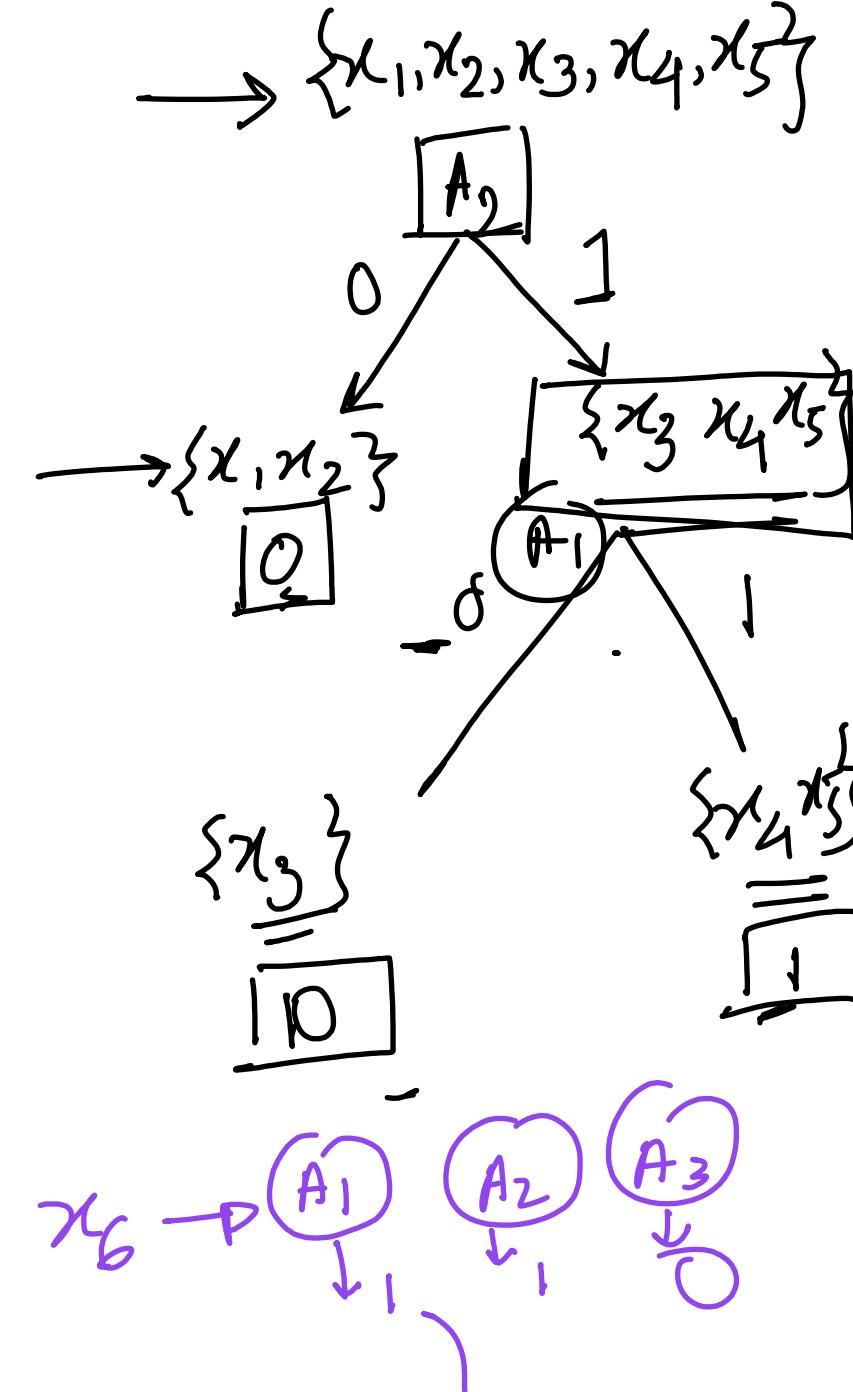
$x_3 \ x_4 \ x_5$



$$E(x_3) = 0$$

$$E_2 \{x_4, x_5\} = 0$$

$$IG(A_1) = 0.918 - 0 = 0.918$$



A_2 ~~split~~

$\underline{A_3}$ split \rightarrow No need

$x_6 \downarrow$
 $x_6 \rightarrow$ loan
approved?

