Decision Tree

Iran University of Science and Technology

M. S. Tahaei Ph.D.

Fall 2024
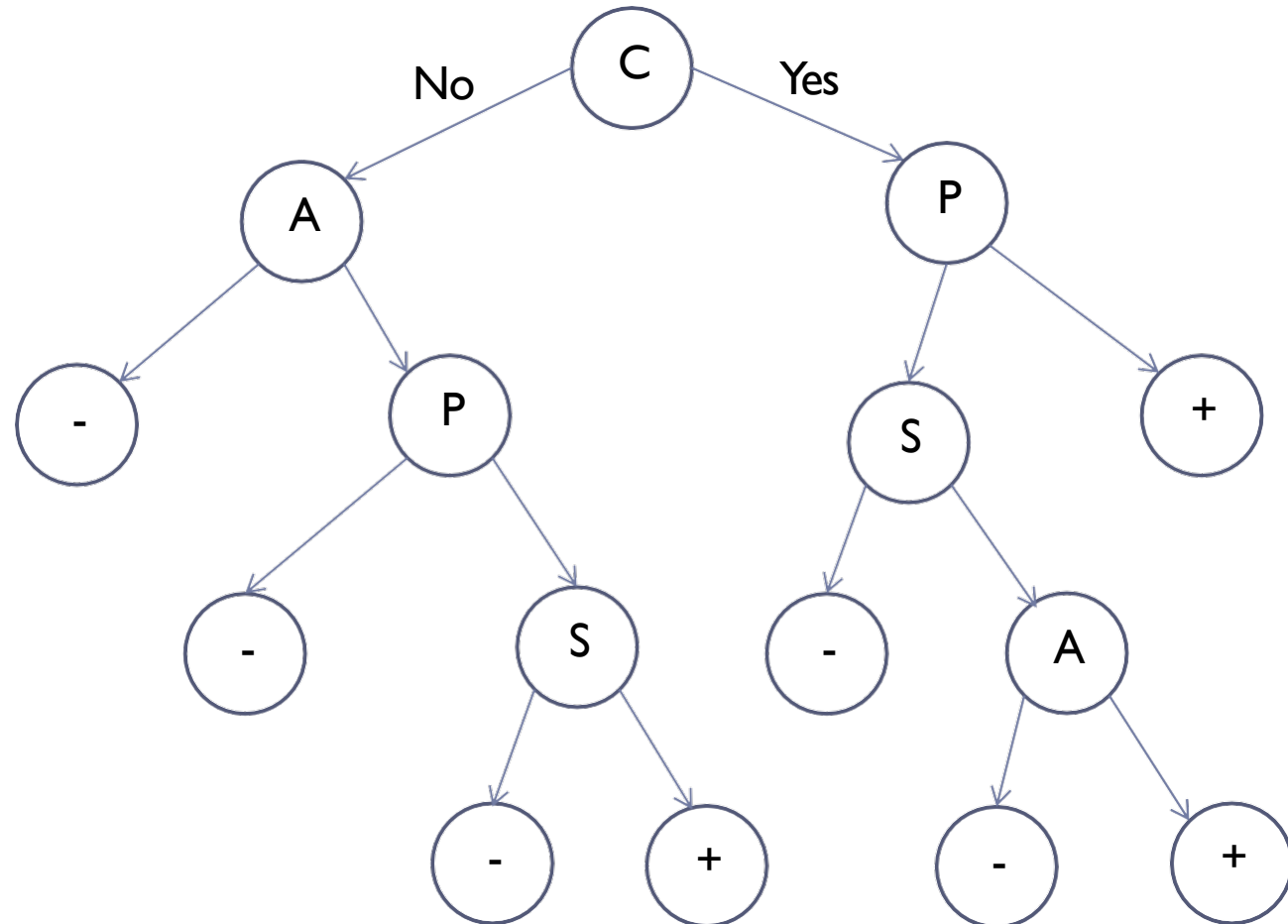
Decision tree

▶ One of the most intuitive classifiers that is easy to understand and construct

   ▸ However, it also works very (very) well

▶ Application: Database mining

Decision tree

▸ Features:
  ▸ A:age>40
  ▸ C:chest pain
  ▸ S:smoking
  ▸ P:physical test

    ▸ Label:
      ▸ Heart disease (+),No heart disease (-)

Decision tree: structure

▶ Leaves (terminal nodes) represent target variable

  ▸ Each leaf represents a class label

▶ Each internal node corresponds to denotes a test on an attribute

  ▸ Edges to children for each of the possible values of that attribute
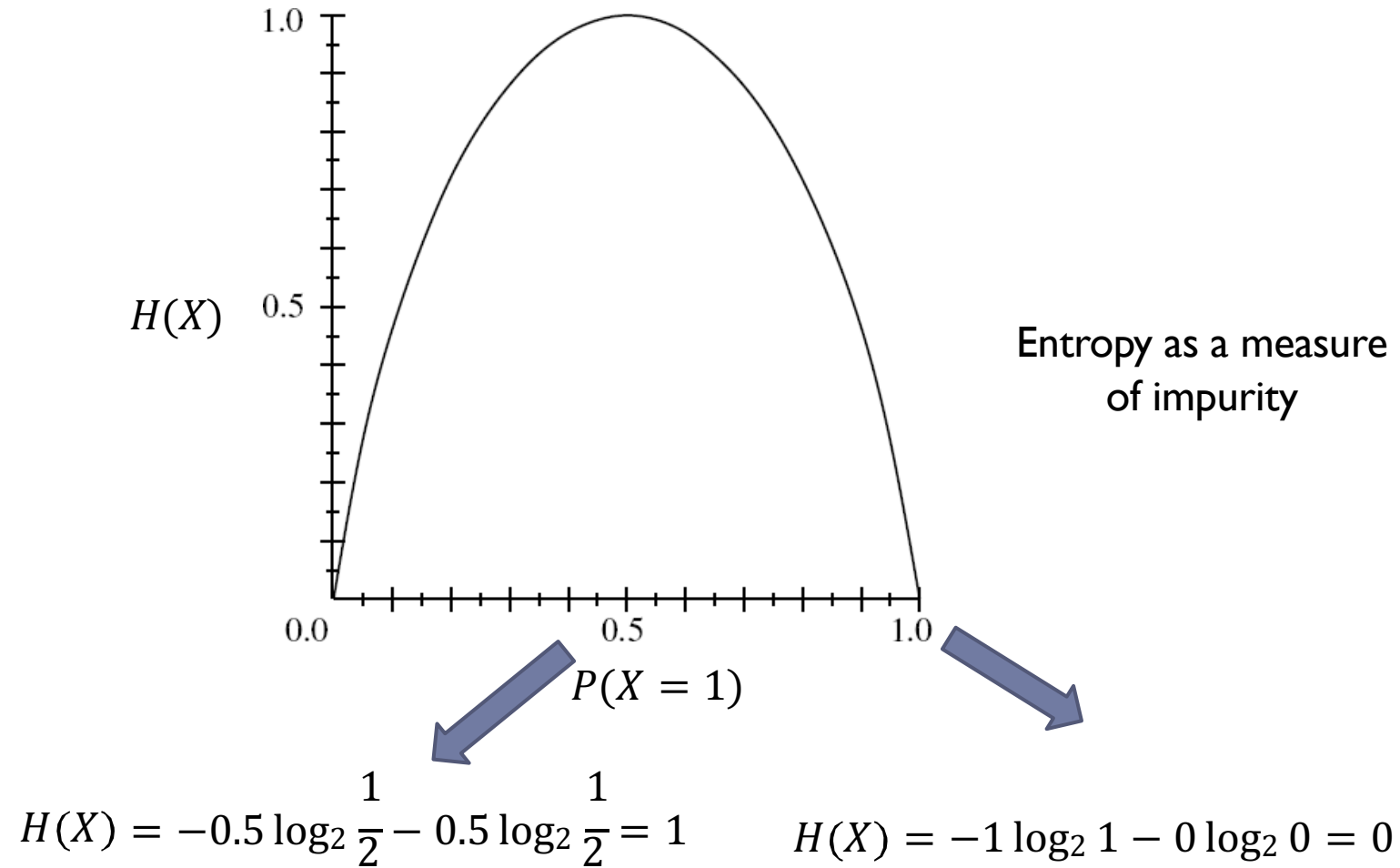
Which attribute is the best?

- A variety of heuristics for picking a good test
  - Information gain: originated with ID3 (Quinlan, 1979).
  - Gini impurity
  - …

- These metrics are applied to each candidate subset, and the resulting values are combined (e.g., averaged) to provide a measure of the quality of the split.

Entropy

$$H(X) = - \sum_{x_i \in X} P(x_i) \log P(x_i)$$

▸ Entropy measures the uncertainty in a specific distribution

▸ Information theory:

  ▸ $H(X)$: <u>expected number of bits</u> needed to encode a randomly drawn value of X (under most efficient code)

    ▸ Most efficient code assigns $-\log P(X = i)$ bits to encode $X = i$

    ▸ $\Rightarrow$ expected number of bits to code one random X is $H(X)$

Entropy for a Boolean variable

$H(X)$

Entropy as a measure of impurity

$P(X = 1)$

$$H(X) = -0.5 \log_2 \frac{1}{2} - 0.5 \log_2 \frac{1}{2} = 1$$

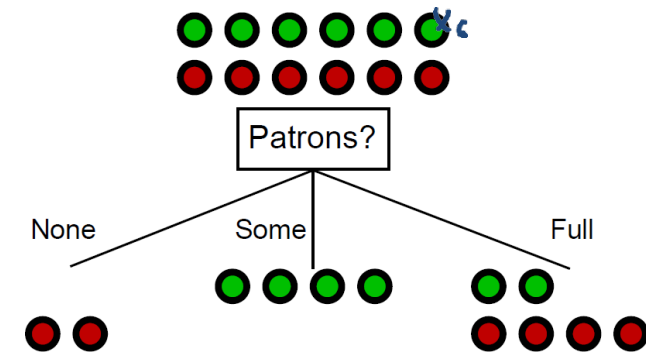$$H(X) = -1 \log_2 1 - 0 \log_2 0 = 0$$

Information Gain (IG)

$$Gain(S, A) \equiv H_S(Y) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H_{S_v}(Y)$$

- ▸ A: variable used to split samples
- ▸ Y: target variable
- ▸ S: samples
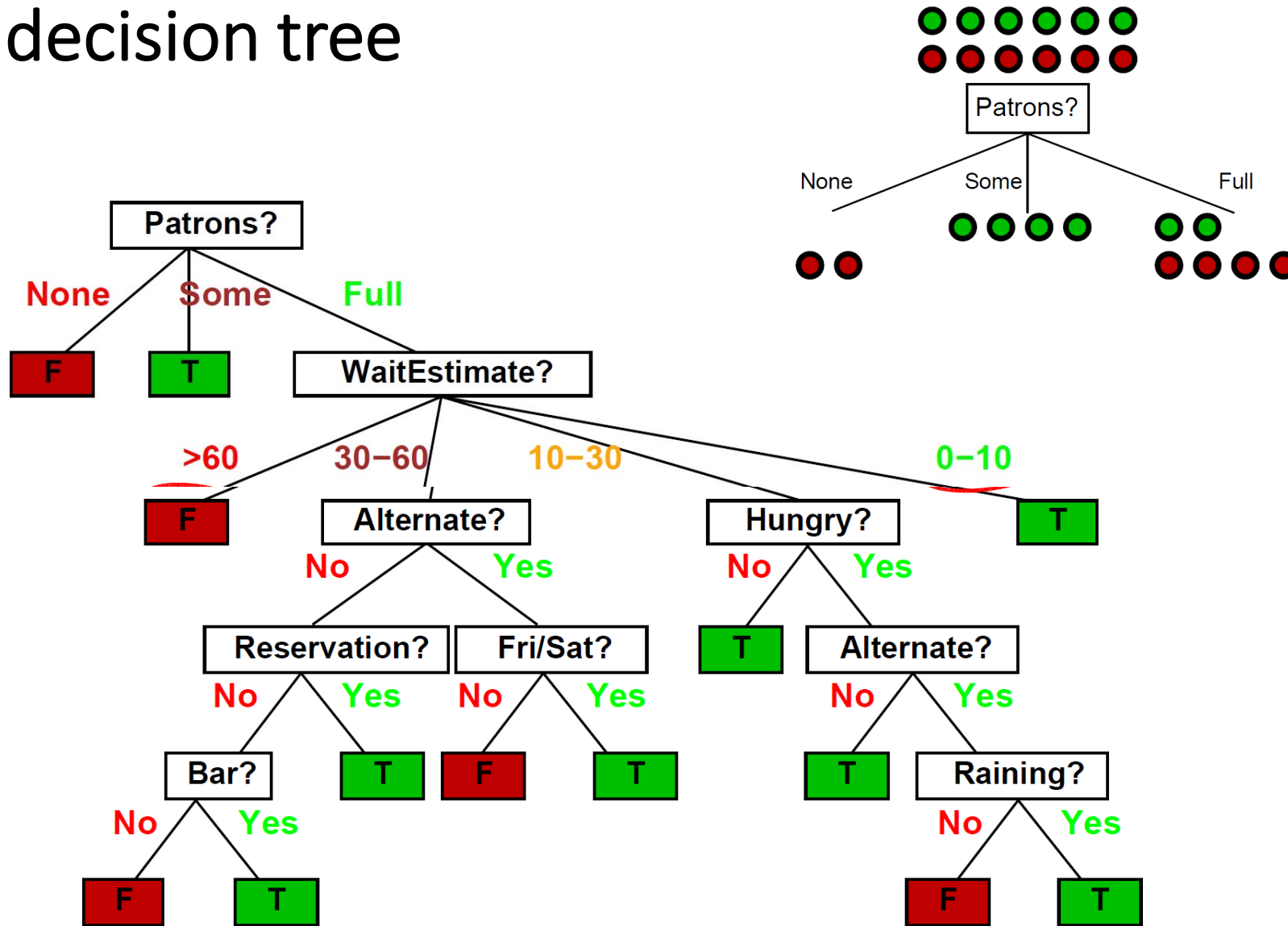
From a spreadsheet to a decision node

Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	Bar	Fri	Hun	Pat	$Price$	$Rain$	Res	$Type$	Est	$WillWait$
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Classification of examples is positive (T) or negative (F)

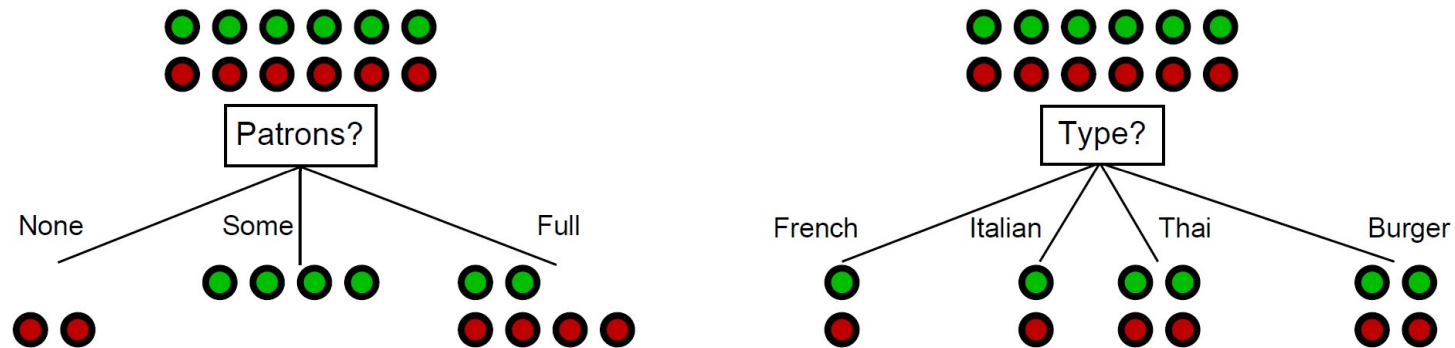[AI book of Stuart Russell and Peter Norvig]

A learned decision tree

[AI book of Stuart Russell and Peter Norvig]

Construct the tree

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

Patrons? is a better choice—gives **information** about the classification

For a training set containing p positive examples and n negative examples, we have:

$$H(\frac{p}{p+n},\frac{n}{p+n}) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

How to pick nodes?

- A chosen attribute A, with K distinct values, divides the training set

- E into subsets $E_1, \dots, E_K$.

- The **Expected Entropy (EH) remaining** after trying attribute A

- (with branches $i=1,2,\dots,K$) is

$$EH(A) = \sum_{i=1}^{K} \frac{p_i + n_i}{p+n} H\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

- **Information gain (I)** or **reduction in entropy** for this attribute is:

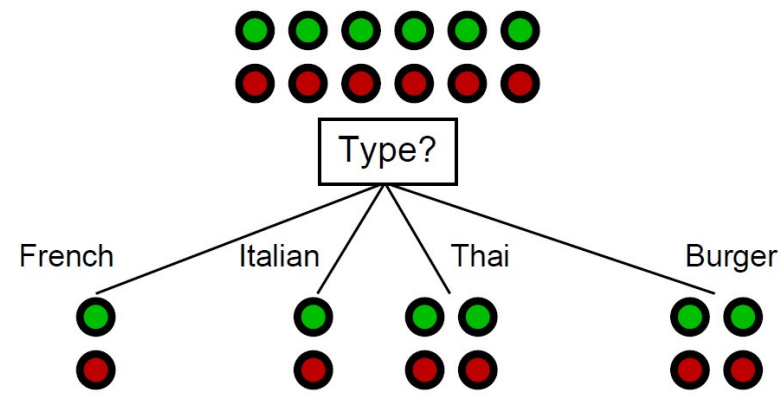$$I(A) = H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - EH(A)$$

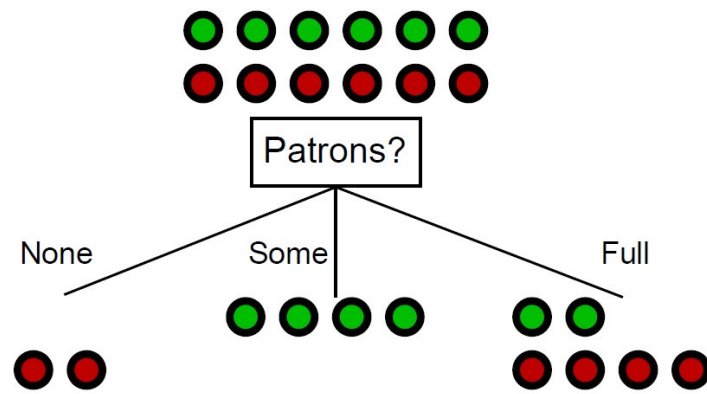- Choose the attribute with the largest I

Example

- **Convention**: For the training set, $p = n = 6$, $H(6/12, 6/12) = 1$ bit

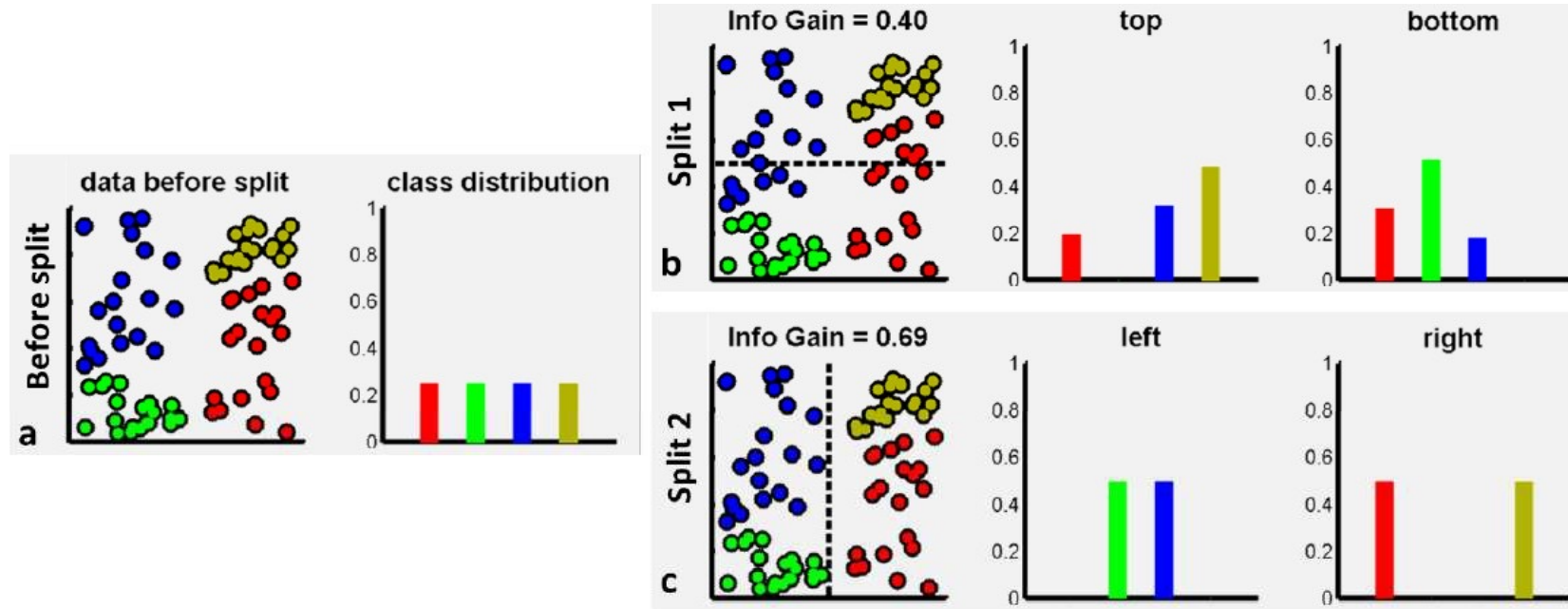- Consider the attributes *Patrons* and *Type* (and others too):

$$I(Patrons) = 1 - [\frac{2}{12}H(0,1) + \frac{4}{12}H(1,0) + \frac{6}{12}H(\frac{2}{6},\frac{4}{6})] = .0541 \text{ bits}$$

$$I(Type) = 1 - [\frac{2}{12}H(\frac{1}{2},\frac{1}{2}) + \frac{2}{12}H(\frac{1}{2},\frac{1}{2}) + \frac{4}{12}H(\frac{2}{4},\frac{2}{4}) + \frac{4}{12}H(\frac{2}{4},\frac{2}{4})] = 0 \text{ bits}$$

[Hwee Tou Ng & Stuart Russell]

Use information gain to decide splits

$$I_j = H(\mathcal{S}_j) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}_j^i|}{|\mathcal{S}_j|} H(\mathcal{S}_j^i)$$

[Criminisi et al, 2011]

Decision tree: learning

- Decision tree learning: construction of a decision tree from training samples.
  - Decision trees used in data mining are usually classification trees

- There are many specific decision-tree learning algorithms, such as:
  - ID3
  - C4.5

- Approximates functions of usually discrete domain
  - The learned function is represented by a decision tree

Decision tree learning

- Learning an optimal decision tree is NP-Complete
  - Instead, we use a greedy search based on a heuristic
  - We cannot guarantee to return the globally-optimal decision tree.

- The most common strategy for DT learning is a greedy top-down approach
  - chooses a variable at each step that best splits the set of items.

- Tree is constructed by splitting samples into subsets based on an attribute value test in a recursive manner

How to construct basic decision tree?

▶ We prefer decisions leading to a simple, compact tree with few nodes

▶ Which attribute at the root?

   ▶ Measure: how well the attributes splits the set into homogeneous subsets (having same value of target)

      ▶ Homogeneity of the target variable within the subsets.

▶ How to form descendant?

   ▶ Descendant is created for each possible value of A

      ▶ Training examples are sorted to descendant nodes

Constructing a decision tree

Function FindTree(S,A) $\longrightarrow$ S: samples, A: attributes

    If empty(A) or all labels of the samples in S are the same

        status = leaf

        class = most common class in the labels of S

    else

        status = internal

        a ←bestAttribute(S,A)

        LeftNode = FindTree(S(a=1),A \ {a})

        RightNode = FindTree(S(a=0),A \ {a})

    end

end

Recursive calls to create left and right subtrees
S(a=1) is the set of samples in S for which a=1

Top down, Greedy, No backtrack

Constructing a decision tree

Function FindTree(S,A)

    If empty(A) or all labels of the samples in S are the same

        status = leaf

        class = most common class in the labels of S else

        status = internal

        a ←bestAttribute(S,A)

        LeftNode = FindTree(S(a=1),A \ {a})

        RightNode =   Tree is constructed by splitting samples into subsets based on

      end

    end

an attribute value test in a recursive manner

- The recursion is completed when the subset at a node has all the same value of the target variable

- or when splitting no longer adds value to the predictions.

ID3

ID3 (Examples, Target_Attribute, Attributes)

Create a root node for the tree

If all examples are positive, return the single-node tree Root, with label = +

If all examples are negative, return the single-node tree Root, with label = -

If number of predicting attributes is empty then

    return Root, with label = most common value of the target attribute in the examples

else

    A = The Attribute that best classifies examples.

    Testing attribute for Root = A. for each

    possible value, v_i, of A

        Add a new tree branch below Root, corresponding to the test A = v_i . Let

        Examples(v_i) be the subset of examples that have the value for A

        if Examples(v_i) is empty then

            below this new branch add a leaf node with label = most common target value in the examples

        else below this new branch add subtree **ID3 (Examples(v_i), Target_Attribute, Attributes –**
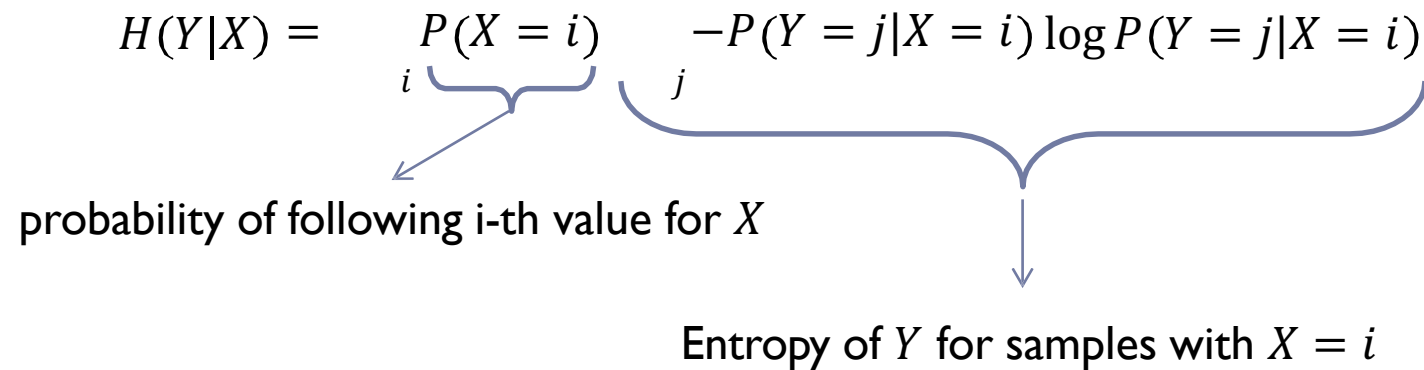
        {A})

return Root

Mutual Information

▸ The expected reduction in entropy of Y caused by knowing X:

$$I(X, Y) = H(Y) - H(Y|X)$$

$$= - \sum_i \sum_j P(X = i, Y = j) \log \frac{P(X = i)P(Y = j)}{P(X = i, Y = j)}$$

▸ Mutual information in decision tree:

　▸ $H(Y)$: Entropy of Y (i.e., labels) before splitting samples

　▸ $H(Y|X)$: Entropy of Y after splitting samples based on attribute X

　　▸ It shows expectation of label entropy obtained in different splits (where splits are formed based on the value of attribute X)

Conditional entropy

$$H(Y|X) = - \sum_i \sum_j P(X = i, Y = j) \log P(Y = j | X = i)$$

$$H(Y|X) = \underbrace{P(X = i)}_{i} \underbrace{-P(Y = j | X = i) \log P(Y = j | X = i)}_{j}$$

probability of following i-th value for X
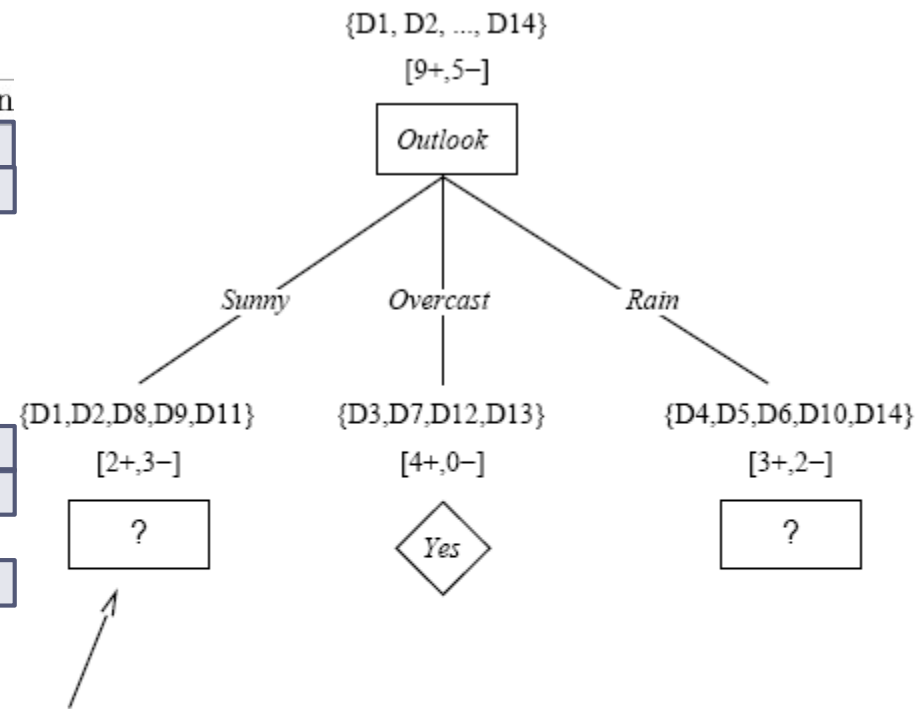
Entropy of Y for samples with $X = i$

How to find the best attribute?

- Information gain as our criteria for a good split
  - attribute that maximizes information gain

- When a set of S samples have been sorted to a node, choose j-th attribute for test in this node where:

$$j = \underset{i \in \text{remaining atts.}}{\arg\max} Gain(S, X_i)$$

$$= \underset{i \in \text{remaining atts.}}{\arg\max} H_S(Y) - H_S(Y|X_i)$$

$$= \underset{i \in \text{remaining atts.}}{\arg\min} H_S(Y|X_i)$$

Information Gain: Example

Day	Outlook	Temperature	Humidity	Wind	PlayTen
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

{D1, D2, ..., D14}

[9+,5−]

Outlook

Sunny   Overcast   Rain

{D1,D2,D8,D9,D11}   {D3,D7,D12,D13}   {D4,D5,D6,D10,D14}

[2+,3−]   [4+,0−]   [3+,2−]

?   Yes   ?

Which attribute should be tested here?

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

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

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

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

ID3 algorithm: Properties

- The algorithm
  - either reaches homogenous nodes
  - or runs out of attributes

- Guaranteed to find a tree consistent with any conflict-free training set
  - ID3 hypothesis space of all DTs contains all discrete-valued functions
  - Conflict free training set: identical feature vectors always assigned the same class

- But not necessarily find the simplest tree (containing minimum number of nodes).
  - a greedy algorithm with locally-optimal decisions at each node (no backtrack).

Decision tree as a rule base

▶ Decision tree = a set of rules

▶ Disjunctions of conjunctions of test on the attribute values
  ▶ Each path from root to a leaf = conjunction of attribute tests
  ▶ All of the leafs with $y = i$ are considered to find rule for $y = i$

How partition instance space?

▸ Decision tree

   ▸ Partition the instance space into axis-parallel regions, labeled with class value
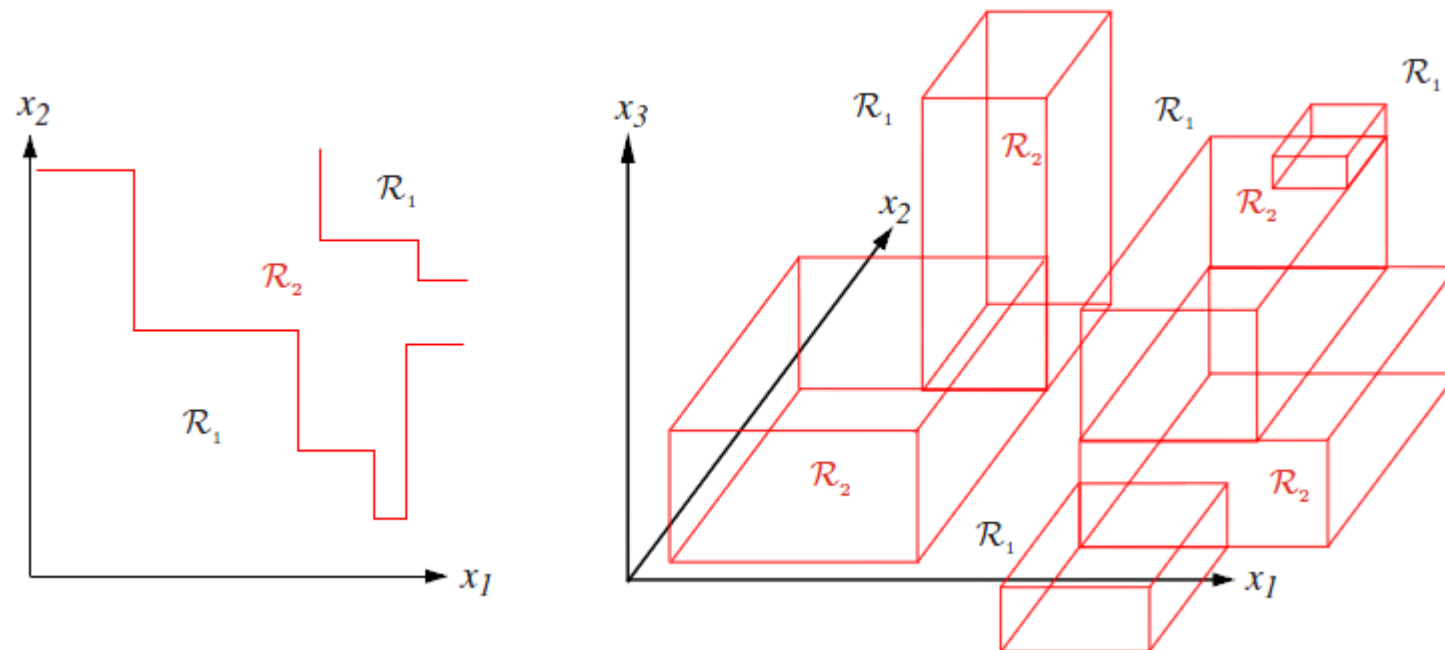

[Duda & Hurt's Book]

Over-fitting problem

▸ ID3 perfectly classifies training data (for consistent data)

- ▸ It tries to memorize every training data
- ▸ Poor decisions when very little data (it may not reflect reliable trends)
  - ▸ Noise in the training data: the tree is erroneously fitting.
  - ▸ A node that "should" be pure but had a single (or few) exception(s)?

▸ For many (non relevant) attributes, the algorithm will continue to split nodes

- ▸ leads to over-fitting!

Over-fitting problem: an example

Consider adding a (noisy) training example:

Outlook	Temp	Humidity	Wind	PlayTennis
Sunny	Hot	Normal	Strong	No
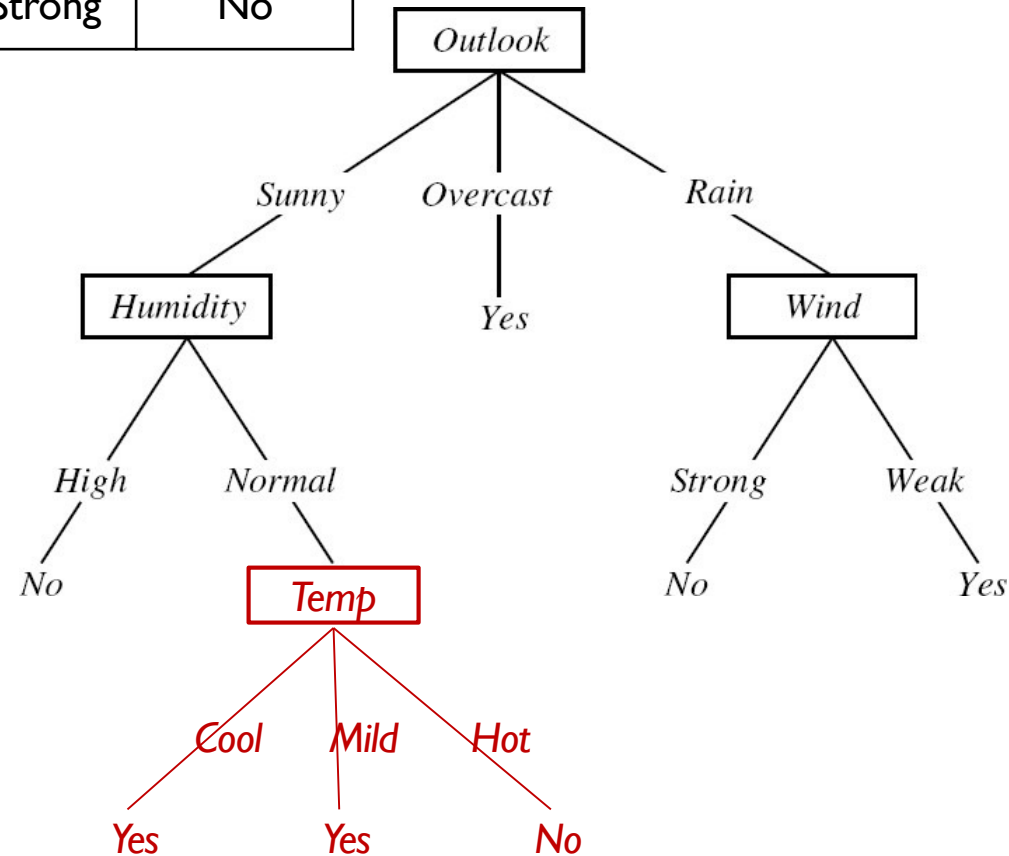
Over-fitting in decision tree learning

- Hypothesis space H:decision trees
- Training (emprical) error of $h \in H$:$error_{train}(h)$
- Expected error of $h \in H$:$error_{true}(h)$
- h overfits training data if there is a $h' \in H$ such that
  - $error_{train}(h) < error_{train}(h')$
  - $error_{true}(h) > error_{true}(h')$
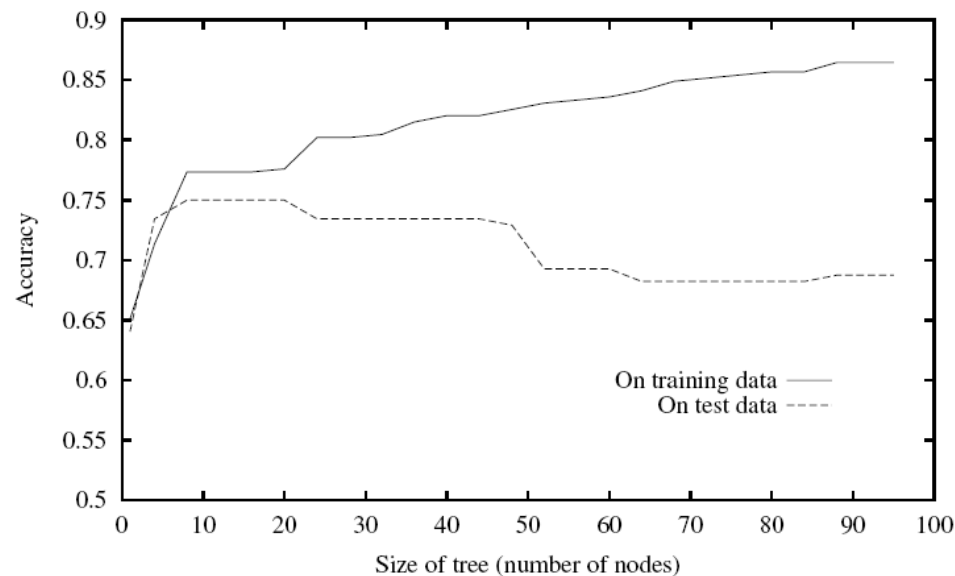
A question?

▸ How can it be made smaller and simpler?

  ▸ Early stopping

    ▸ When should a node be declared as a leaf?

    ▸ If a leaf node is impure, how should the category label be assigned?

  ▸ Pruning?

    ▸ Build a full tree and then post-process it

Avoiding overfitting

1) **Stop growing** when the data split is not statistically significant.

2) Grow full tree and then **prune** it.
   - More successful than stop growing in practice.

3) How to select "best" tree:
   - Measure performance over separate validation set
   - MDL: minimize
     - $size(tree) + size(missclassifications(tree))$

Reduced-error pruning

- ▸ Split data into train and validation set
- ▸ Build tree using training set
- ▸ Do until further pruning is harmful:
  - ▸ Evaluate impact on validation set when pruning sub-tree rooted at each node
    - ▸ Temporarily remove sub-tree rooted at node
    - ▸ Replace it with a leaf labeled with the current majority class at that node
    - ▸ Measure and record error on validation set
  - ▸ Greedily remove the one that most improves validation set accuracy (if any).

Produces smallest version of the most accurate sub-tree.

Continuous attributes

- Tests on continuous variables as boolean ?

- Either use threshold to turn into binary or discretize

- Its possible to compute information gain for all possible thresholds (there are a finite number of training samples)

- Harder if we wish to assign more than two values (can be done recursively)

Decision tree advantages

▶ Simple to understand and interpret

▶ Requires little data preparation and also can handle both numerical and categorical data

▶ Time efficiency of learning decision tree classifier

   ▶ Cab be used on large datasets

▶ Robust: Performs well even if its assumptions are somewhat violated

Reference

- T. Mitchell, "Machine Learning", 1998. [Chapter 3]

- Machine Learning, Dr. Soleymani, Sharif University