Data Mining

Part 4. Prediction

4.2 Decision Tree

Fall 2009

Instructor: Dr. Masoud Yaghini

Outline

- Introduction
- Basic Algorithm for Decision Tree Induction
- Attribute Selection Measures
  - Information Gain
  - Gain Ratio
  - Gini Index
- Tree Pruning
- Scalable Decision Tree Induction Methods
- References

1. Introduction

Decision Tree Induction

- **Classification by Decision tree**
  - the learning of decision trees from class-labeled training instances.
- A **decision tree** is a flowchart-like tree structure, where
  - each internal node (non-leaf node) denotes a test on an attribute
  - each branch represents an outcome of the test
  - each leaf node (or *terminal node)* holds a class label.
  - The topmost node in a tree is the root node.

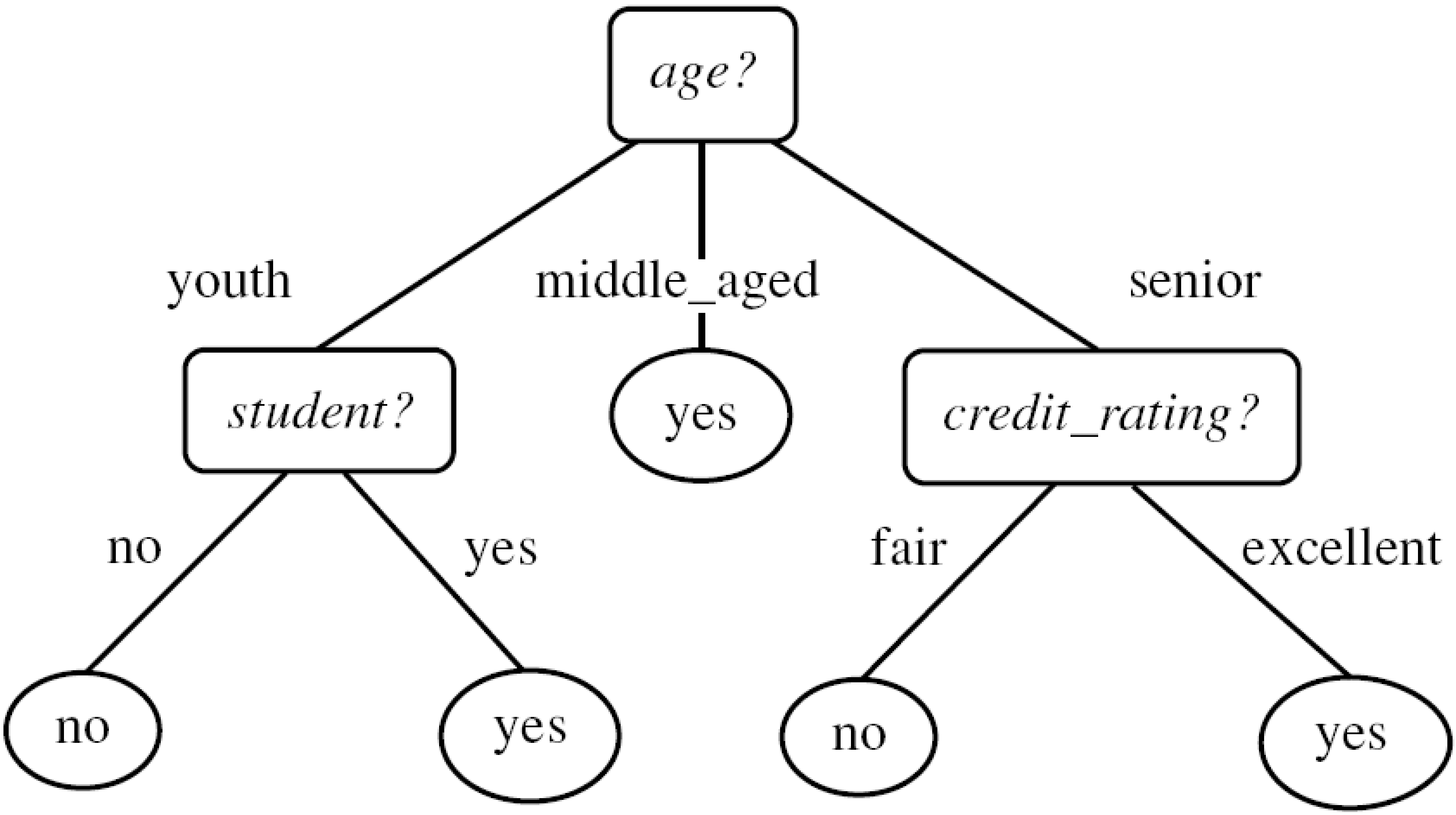
An example

- This example represents the concept *buys_computer*
- It predicts whether a customer at *AllElectronics is likely to purchase a computer.*

An example: Training Dataset

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

An example: A Decision Tree for "*buys_computer*"



Decision Tree Induction

- How are decision trees used for classification?
  - Given an instance, ***X*,** for which the associated class label is unknown,
  - The attribute values of the instance are tested against the decision tree
  - A path is traced from the root to a leaf node, which holds the class prediction for that instance.

Decision Tree Induction

- Advantages of decision tree
  - The construction of decision tree classifiers does not require any domain knowledge or parameter setting.
  - Decision trees can handle high dimensional data.
  - Easy to interpret for small-sized trees
  - The learning and classification steps of decision tree induction are simple and fast.
  - Accuracy is comparable to other classification techniques for many simple data sets
  - Convertible to simple and easy to understand classification rules

Decision Tree

- Decision tree algorithms have been used for classification in many application areas, such as:
  - Medicine
  - Manufacturing and production
  - Financial analysis
  - Astronomy
  - Molecular biology.

2. Basic Algorithm

Decision Tree Algorithms

- ID3 (Iterative Dichotomiser) algorithm
  - Developed by J. Ross Quinlan
  - During the late 1970s and early 1980s
- C4.5 algorithm
  - Quinlan later presented C4.5 (a successor of ID3)
  - Became a benchmark to which newer supervised learning algorithms are often compared.
  - Commercial successor: C5.0
- CART (Classification and Regression Trees) algorithm
  - The generation of binary decision trees
  - Developed by a group of statisticians

Decision Tree Algorithms

- ID3, C4.5, and CART adopt a greedy (i.e., nonbacktracking) approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner.
- Most algorithms for decision tree induction also follow such a top-down approach, which starts with a training set of instances and their associated class labels.
- The training set is recursively partitioned into smaller subsets as the tree is being built.

Basic Algorithm

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

Basic Algorithm

- Algorithm: *Generate_decision_tree*
- Parameters:
  - *D,* a data set
  - *Attribute_list* : a list of attributes describing the instances
  - *Attribute_selection_method* : a heuristic procedure for selecting the attribute

Basic Algorithm

- Step 1
  - The tree starts as a single node, *N,* representing the training instances in *D*
- Steps 2
  - If the instances in *D* are all of the same class, then node *N* becomes a leaf and is labeled with that class.
- Steps 3
  - if *attribute_list* is empty then return N as a leaf node labeled with the majority class in D
  - Steps 3 is terminating conditions.

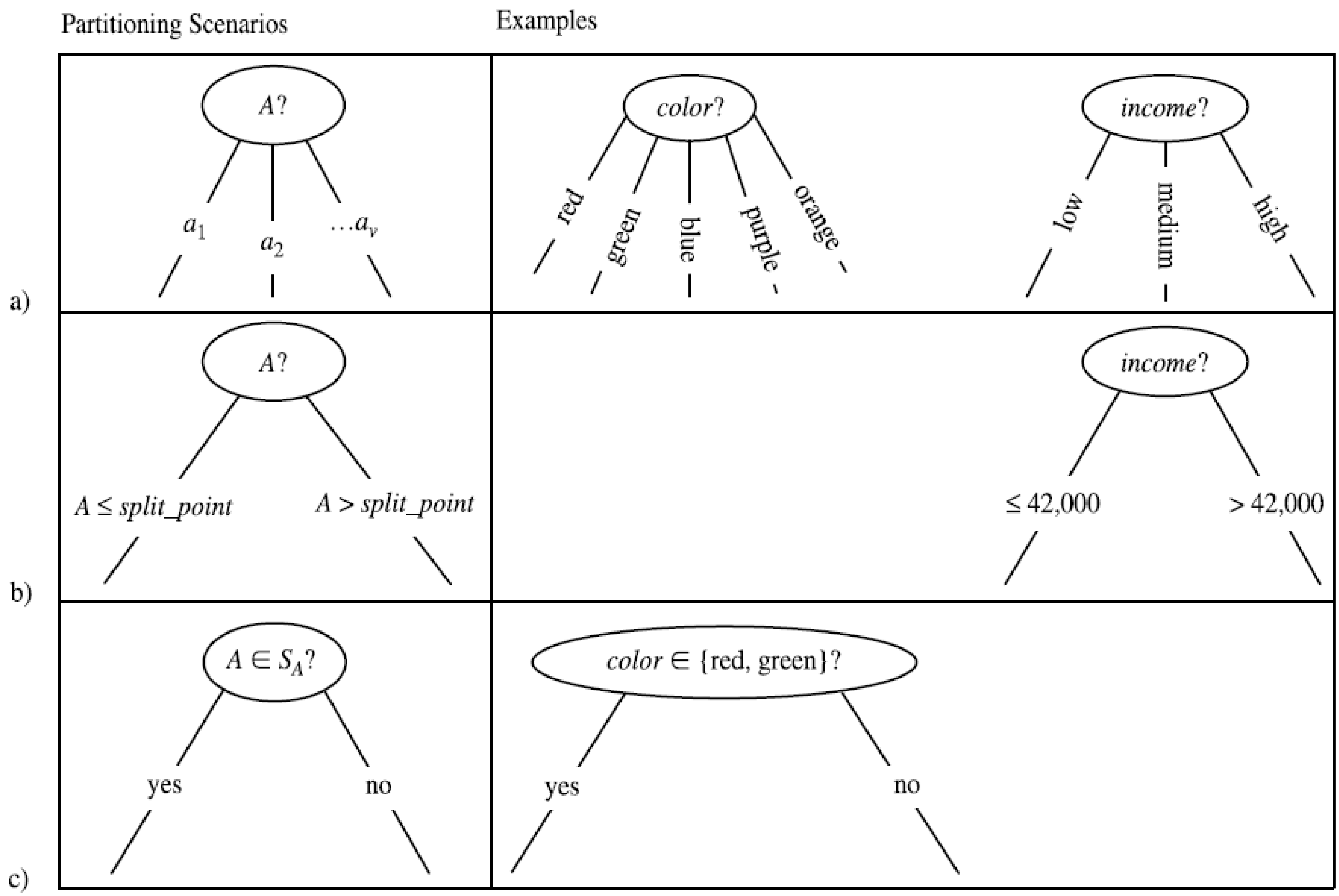
Basic Algorithm

- Step 4
  - the algorithm calls *Attribute_selection_method* to determine the splitting criterion.
  - The splitting criterion tells us which attribute to test at node *N* by determining the "best" way to separate or partition the instances in *D* into individual classes
  - The splitting criterion indicates the splitting attribute and may also indicate either a split-point or a splitting subset.

Basic Algorithm

- Step 5
  - The node *N* is labeled with the splitting criterion, which serves as a test at the node
- Steps 6
  - A branch is grown from node *N* for each of the outcomes of the splitting criterion.
  - The instances in *D* are partitioned accordingly
  - Let *A be the splitting_attribute,* there are three possible scenarios for branching:
    - ***A*** is discrete-valued
    - ***A*** is continuous-valued
    - ***A*** is discrete-valued and a binary treemust be produced

Basic Algorithm



Basic Algorithm

- In scenario a (A is discrete-valued)
  - the outcomes of the test at node N correspond directly to the known values of A.
  - Because all of the instances in a given partition have the same value for A, then A need not be considered in any future partitioning of the instances.
  - Therefore, it is removed from attribute_list.

Basic Algorithm

- In scenario b (A is continuous-valued)
  - the test at node N has two possible outcomes, corresponding to the conditions A ≤ split_point and A > split_point, respectively.
  - where split_point is the split-point returned by Attribute_selection_method as part of the splitting criterion.
  - The instances are partitioned such that D1 holds the subset of class-labeled instances in D for which A ≤ split_point, while D2 holds the rest.

Basic Algorithm

- In scenario c (A is discrete-valued and a binary tree must be produced)
  - The test at node N is of the form "$A \in S_A$?".
  - S_A is the splitting subset for A, returned by Attribute_selection_method as part of the splitting criterion.

Basic Algorithm

- Step 7
  - for each outcome j of *splitting_criterion*
    - let *Dj* be the set of data tuples in D satisfying outcome j
    - if Dj is empty
      - then attach a leaf labeled with the majority class in D to node N;
    - Else
      - attach the node by Generate_decision_tree(*Dj, attribute list)* to node N
- Step 8
  - The resulting decision tree is returned.

Basic Algorithm

- The algorithm stops only when any one of the following terminating conditions is true:
  1. All of the instances in partition *D* (represented at node N) belong to the same class (steps 2)
  2. There are no remaining attributes for further partitioning (step 3).
  3. There are no instances for a given branch, that is, a partition Dj is empty (step 7).

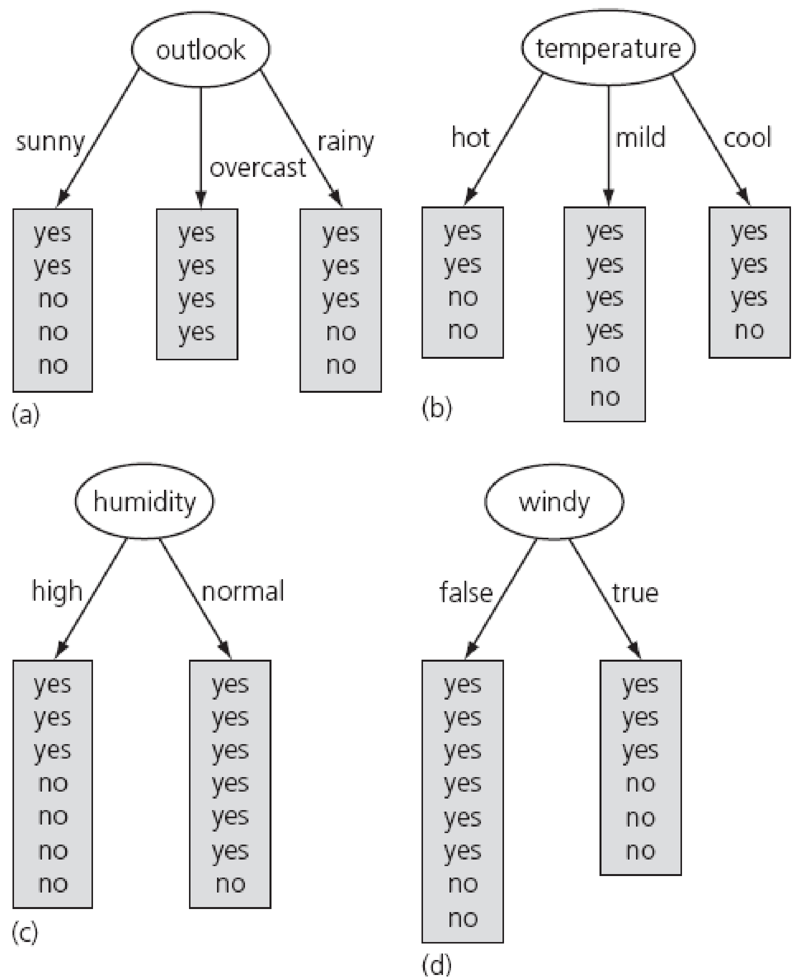
Decision Tree Issues

- Attribute selection measures
  - During tree construction, **attribute selection measures** are used to select the attribute that best partitions the instances into distinct classes.
- Tree pruning
  - When decision trees are built, many of the branches may reflect noise or outliers in the training data.
  - Tree pruning attempts to identify and remove such branches, with the goal of improving classification accuracy on unseen data.
- Scalability
  - Scalability issues related to the induction of decision trees from large databases.

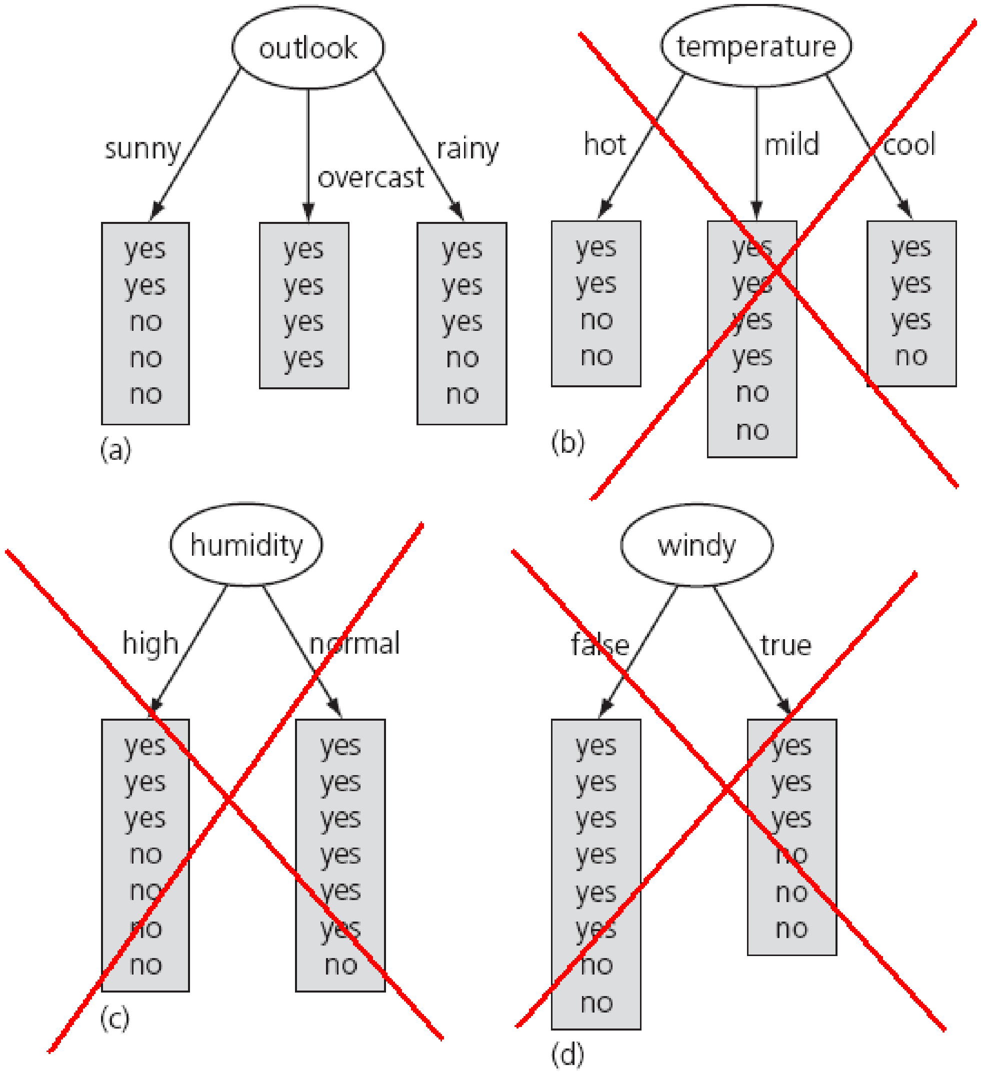
3. Attribute Selection Measures

Attribute Selection Measures

- Which attribute to select?



Attribute Selection Measures



Attribute Selection Measures

- Which is the best attribute?
  - Want to get the smallest tree
  - choose the attribute that produces the "purest" nodes
- Attribute selection measure
  - a heuristic for selecting the splitting criterion that "best" separates a given data partition, *D,* of class-labeled training instances into individual classes.
  - If we were to split *D* into smaller partitions according to the outcomes of the splitting criterion, ideally each partition would be **pure** (i.e., all of the instances that fall into a given partition would belong to the same class).

Attribute Selection Measures

- Attribute selection measures are also known as splitting rules because they determine how the instances at a given node are to be split.
- The attribute selection measure provides a ranking for each attribute describing the given training instances.
- The attribute having the best score for the measure is chosen as the splitting attribute for the given instances.

Attribute Selection Measures

- If the splitting attribute is continuous-valued or if we are restricted to binary trees then, respectively, either a split point or a splitting subset must also be determined as part of the splitting criterion.
- Three popular attribute selection measures:
  - Information gain
  - Gain ratio
  - Gini index

Attribute Selection Measures

- The notation used herein is as follows.
  - Let *D*, the data partition, be a training set of class-labeled instances.
  - Suppose the class label attribute has *m* distinct values defining *m* distinct classes, C_i (for $i = 1, \ldots, m$)
  - *Let* $C_{i,D}$ *be the set of instances of class* C_i *in D.*
  - *Let* $|D|$ and $|C_{i,D}|$ *denote the number of instances in* *D* *and* $C_{i,D}$*, respectively.*

Information Gain

Attribute Selection Measures

- Select the attribute with the highest **information gain** as the splitting attribute
- This attribute minimizes the information needed to classify the instances in the resulting partitions and reflects the **least impurity** in these partitions.
- **ID3** uses **information gain** as its attribute selection measure.
- ***Entropy (impurity)***
  - *High Entropy* means X is from a uniform (boring) distribution
  - *Low Entropy* means X is from a varied (peaks and valleys) distribution

Attribute Selection Measures

- Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, **High degree of impurity**	**Homogeneous,** **Low degree of impurity**

Attribute Selection Measures

- Let pi be the probability that an arbitrary instance in D belongs to class C_i, estimated by $|C_i, D|/|D|$
- **Expected information (entropy)** needed to classify an instance in D is given by:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Info(D) (entropy of D)
  - the average amount of information needed to identify the class label of an instance in D.
  - The smaller information required, the greater the purity.

Attribute Selection Measures

- At this point, the information we have is based solely on the **proportions of instances of each class**.
- A log function to the base 2 is used, because the information is encoded in bits (It is measured in bits).

Attribute Selection Measures

- Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, **High degree of impurity**	**Homogeneous,** **Low degree of impurity**
$Info(D) = 1$	$Info(D) = 0.469$

Attribute Selection Measures

- Suppose attribute A can be used to split D into v partitions or subsets, {D1, D2, … , Dv}, where Dj contains those instances in D that have outcome aj of A.
- Information needed (after using A to split D) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- The smaller the expected information (still) required, the greater the purity of the partitions.

Attribute Selection Measures

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

- Information gain increases with the average purity of the subsets
- Information gain: information needed before splitting – information needed after splitting
  - The attribute that has the highest information gain among the attributes is selected as the splitting attribute.

Example: *AllElectronics*

- This table presents a training set, *D.*

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Example: *AllElectronics*

- The class label attribute, *buys_computer,* has two distinct values (namely, {yes, no}); therefore, there are two distinct classes (that is, $m = 2$).
- Let class *C1* correspond to *yes* and class *C2* correspond to *no*.
- The expected information needed to classify an instance in *D*:

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

Example: ***AllElectronics***

- Next, we need to compute the expected information requirement for each attribute.
- Let's start with the attribute *age.* We need to look at the distribution of *yes* and *no* instances for each category of *age.*
  - For the age category *youth*,
    - there are two yes instances and three *no* instances*.*
  - For the category *middle_aged,*
    - there are four *yes* instances and zero *no* instances*.*
  - For the category *senior,*
    - there are three *yes* instances and two *no* instances.

Example: *AllElectronics*

- The expected information needed to classify an instance in *D* if the instances are partitioned according to *age* is

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2)$$

$$\begin{aligned} Info_{age}(D) &= \frac{5}{14} \times (-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}) \\ &+ \frac{4}{14} \times (-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}) \\ &+ \frac{5}{14} \times (-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}) \\ &= 0.694 \text{ bits.} \end{aligned}$$

Example: *AllElectronics*

- The gain in information from such a partitioning would be

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits}$$

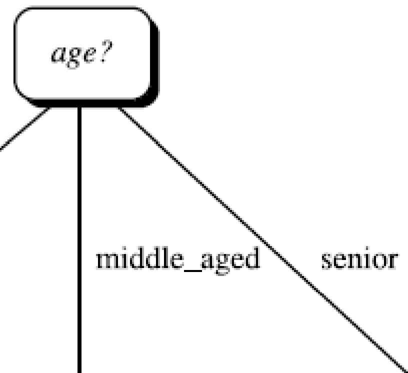
- Similarly

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

- Because *age* has the highest information gain among the attributes, it is selected as the splitting attribute.

Example: *AllElectronics*

- Branches are grown for each outcome of *age.* The instances are shown partitioned accordingly.



income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

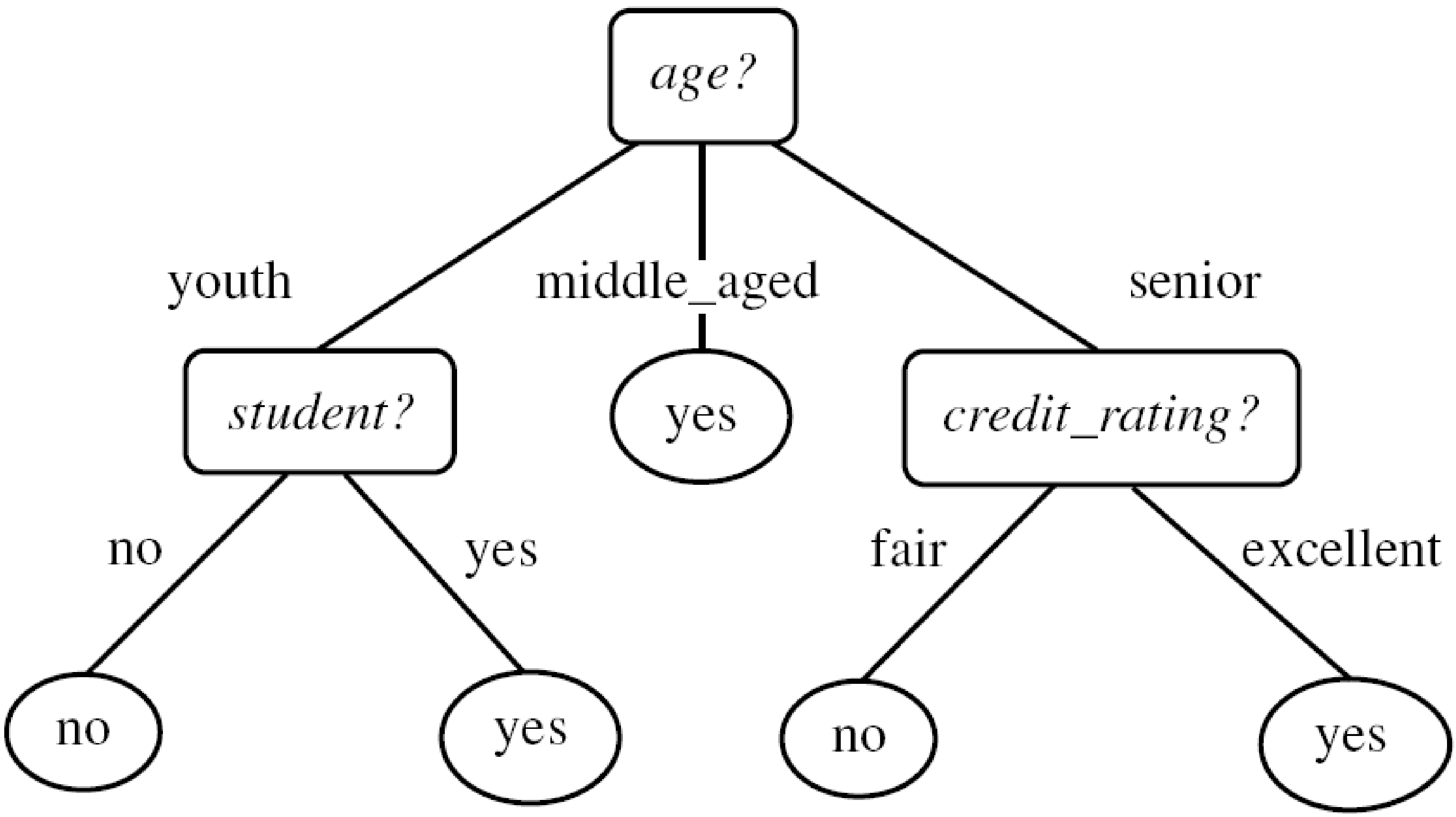
income	*student*	*credit_rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

Example: *AllElectronics*

- Notice that the instances falling into the partition for *age = middle_aged* all belong to the same class.
- Because they all belong to class *"yes," a* leaf should therefore be created at the end of this branch and labeled with *"yes."*

Example: *AllElectronics*

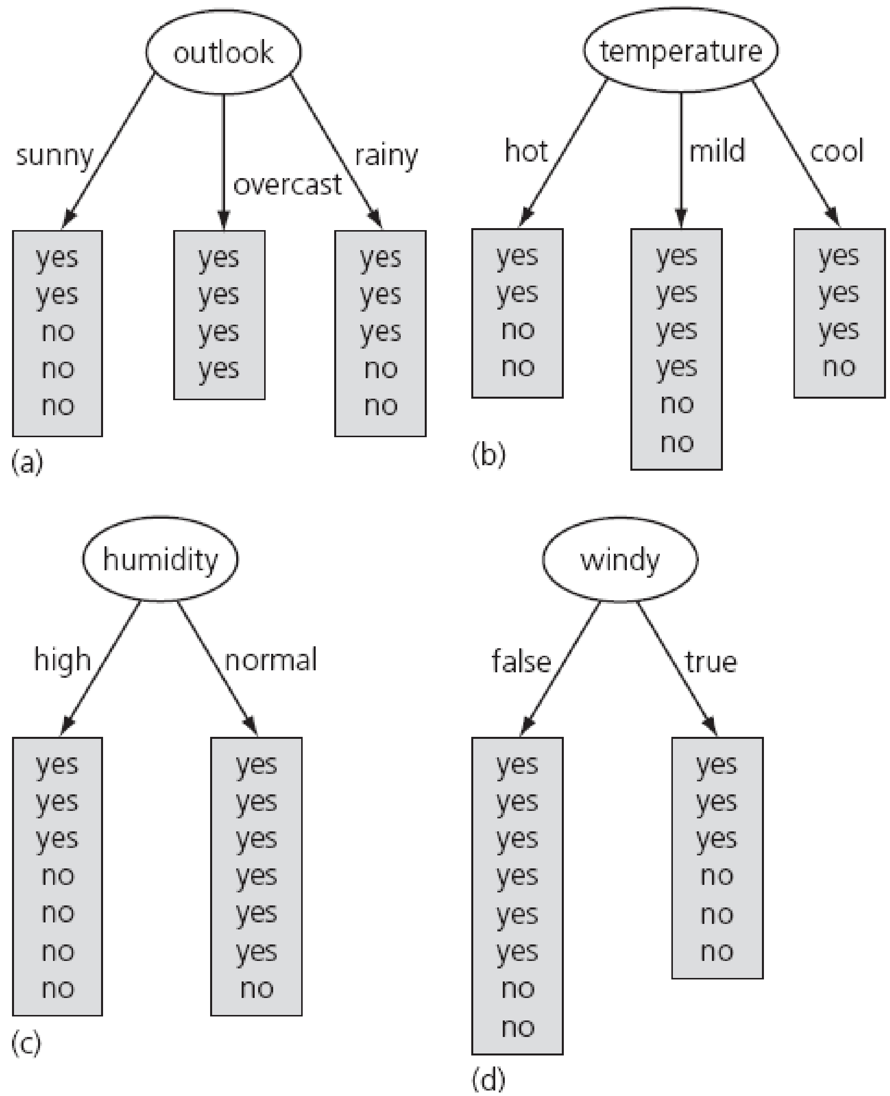
- The final decision tree returned by the algorithm



Example: Weather Problem

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Example: Weather Problem



Example: Weather Problem

- attribute *Outlook:*

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

$$Info_{outlook}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.693$$

Example: Weather Problem

- Information gain: information before splitting – information after splitting:

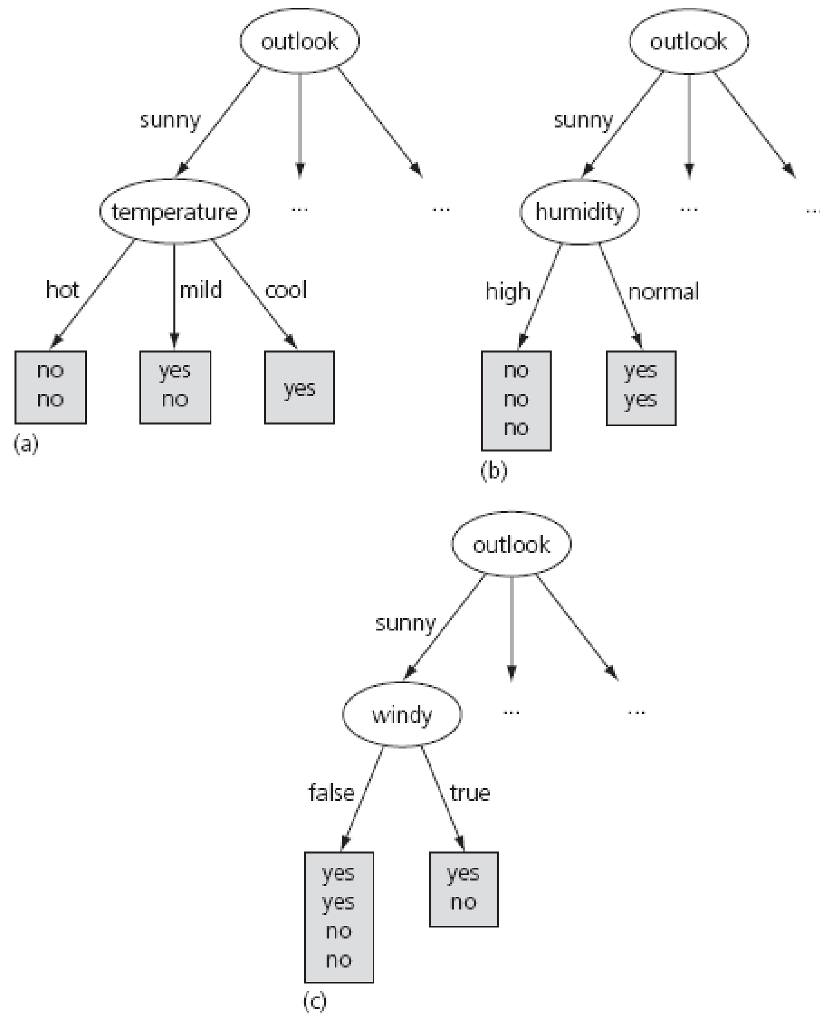
  gain(*Outlook*) = 0.940 – 0.693
  = 0.247 bits

- Information gain for attributes from weather data:

  gain(*Outlook*) = 0.247 bits
  gain(*Temperature*) = 0.029 bits
  gain(*Humidity*) = 0.152 bits
  gain(*Windy*) = 0.048 bits

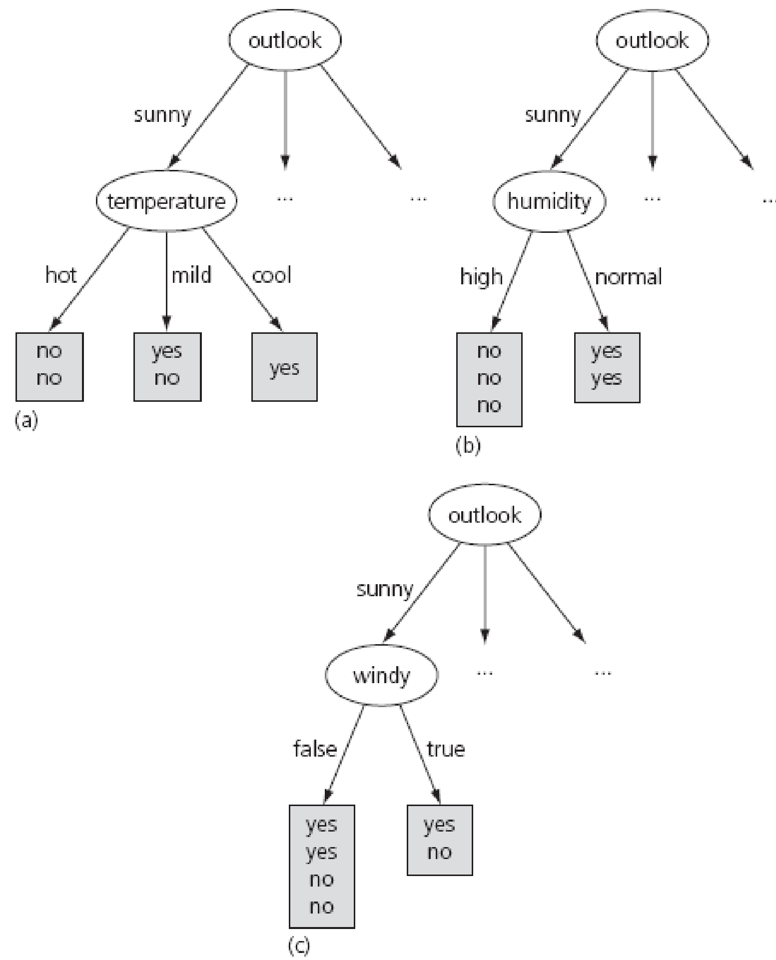
Example: Weather Problem

- Continuing to split



Example: Weather Problem

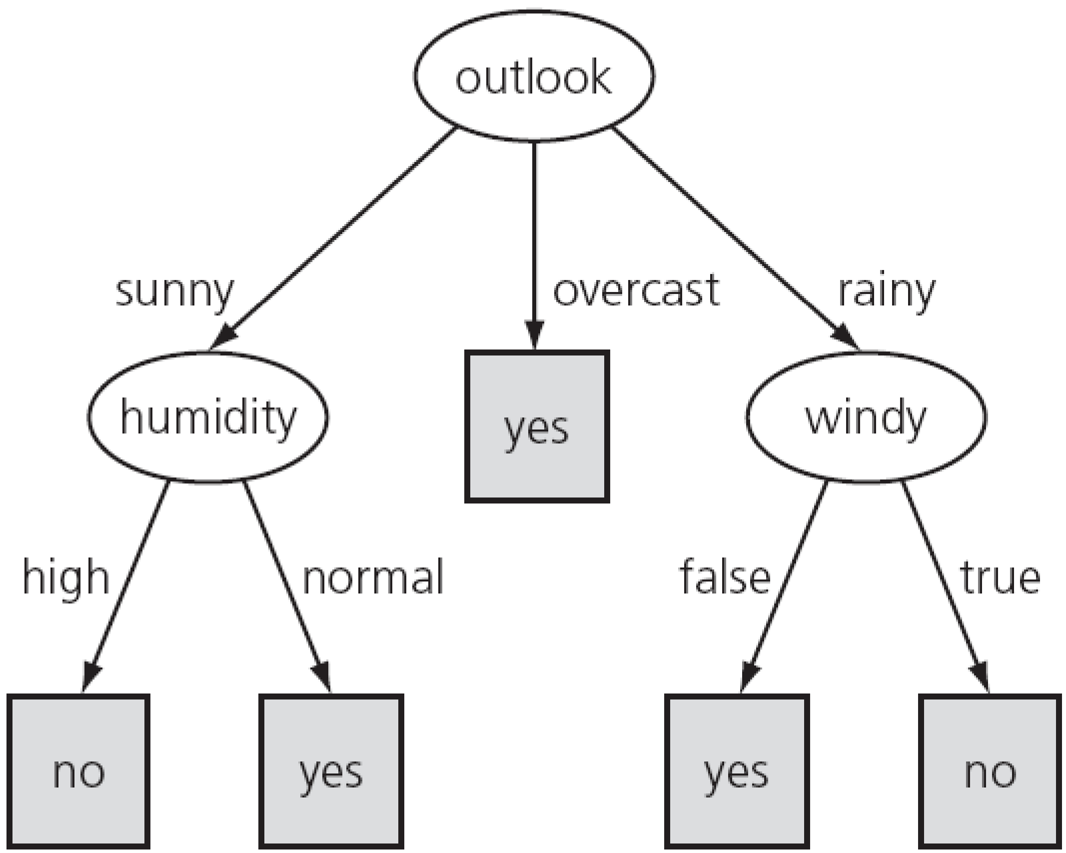
- Continuing to split



$$\text{gain}(\textit{temperature}) = 0.571 \text{ bits}$$
$$\text{gain}(\textit{humidity}) = 0.971 \text{ bits}$$
$$\text{gain}(\textit{windy}) = 0.020 \text{ bits}$$

Example: Weather Problem

- Final decision tree



Continuous-Value Attributes

- Let attribute A be a continuous-valued attribute
- Standard method: binary splits
- Must determine the best split point for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - Therefore, given *v* values of *A,* then *v-1* possible splits are evaluated.
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A

Continuous-Value Attributes

- Split:
  - D1 is the set of instances in D satisfying A ≤ split-point, and D2 is the set of instances in D satisfying A > split-point
- Split on temperature attribute:

64	65	68	69	70	71	72	75	80	81	83	85
yes	no	yes	yes	yes	no	no yes	yes yes	no	yes	yes	no

  - E.g. temperature < 71.5: yes/4, no/2
    temperature > 71.5: yes/5, no/3
  - Info = 6/14 info([4,2]) + 8/14 info([5,3])
    = 0.939 bits

Gain Ratio

Gain ratio

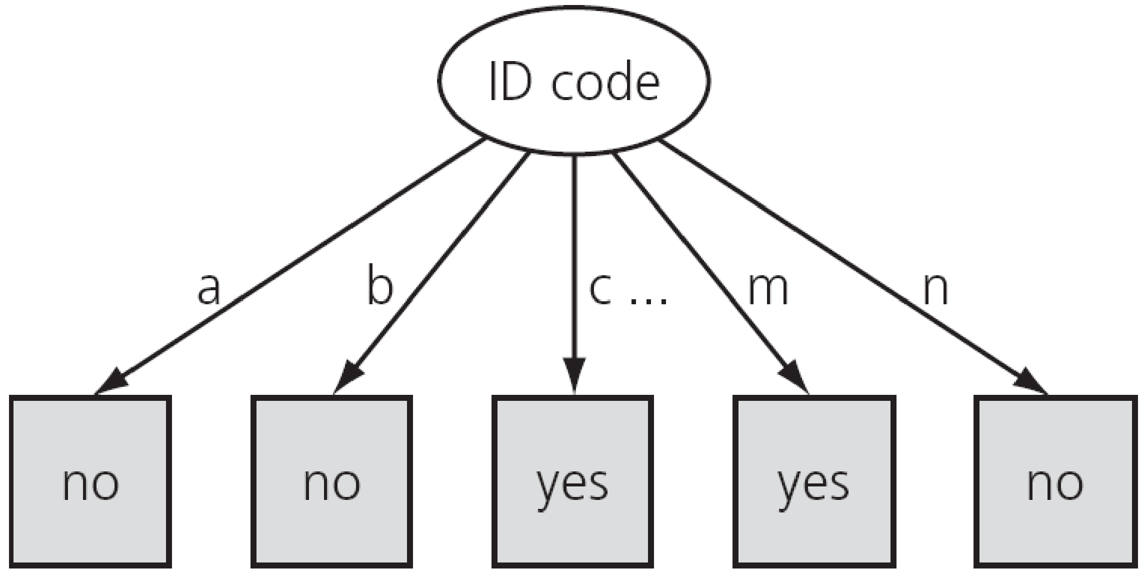
- Problem of information gain
  - When there are attributes with a large number of values
  - Information gain measure is biased towards attributes with a large number of values
  - This may result in selection of an attribute that is non-optimal for prediction

Gain ratio

- Weather data with *ID code*

ID code	Outlook	Temperature	Humidity	Windy	Play
a	sunny	hot	high	false	no
b	sunny	hot	high	true	no
c	overcast	hot	high	false	yes
d	rainy	mild	high	false	yes
e	rainy	cool	normal	false	yes
f	rainy	cool	normal	true	no
g	overcast	cool	normal	true	yes
h	sunny	mild	high	false	no
i	sunny	cool	normal	false	yes
j	rainy	mild	normal	false	yes
k	sunny	mild	normal	true	yes
l	overcast	mild	high	true	yes
m	overcast	hot	normal	false	yes
n	rainy	mild	high	true	no

Gain ratio



- Information gain is maximal for ID code (namely 0.940 bits)

Gain ratio

- *Gain ratio*
  - a modification of the information gain
  - C4.5 uses gain ratio to overcome the problem
- Gain ratio applies a kind of normalization to information gain using a **split information**

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

- The attribute with the maximum gain ratio is selected as the splitting attribute.

Example: Various Partition Numbers

		Class Lable		
		Yes	No	Total
Attribute 1	Value 1	4	8	12
	Value 2	4	8	12
Attribute 2	Value 1	2	4	6
	Value 2	2	4	6
	Value 3	2	4	6
	Value 4	2	4	6

	Gain	SplitInfo	Gain Ratio	
Attribute 1	0.082	1.000	0.082	
Attribute 2	0.082	2.000	0.041	

Example: Unbalanced Partitions

		Class Label		
		Yes	No	Total
Attribute 1	Value 1	2	4	6
	Value 2	6	12	18
Attribute 2	Value 1	4	8	12
	Value 2	4	8	12

$$SplitInfo_1(D) = -\frac{6}{24}\times\log_2(\frac{6}{24}) - \frac{18}{24}\times\log_2(\frac{18}{24}) = 0.811$$

$$SplitInfo_2(D) = -\frac{12}{24}\times\log_2(\frac{12}{24}) - \frac{12}{24}\times\log_2(\frac{12}{24}) = 1$$

	Gain	SplitInfo	Gain Ratio
Attribute 1	0.082	0.811	0.101
Attribute 2	0.082	1	0.082

Gain ratio

- Example
  - Computation of gain ratio for the attribute *income.*
  - A test on *income* splits the data into three partitions, namely *low, medium, and high,* containing four, six, and four instances, respectively.
  - Computation of the gain ratio of *income:*

$$SplitInfo_A(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 0.926$$

  - Gain(income) = 0.029
  - GainRatio(income) = 0.029/0.926 = 0.031

Gain ratio

- Gain ratios for weather data

Outlook		Temperature		Humidity		Windy	
info:	0.693	info:	0.911	info:	0.788	info:	0.892
gain: 0.940–0.693	0.247	gain: 0.940–0.911	0.029	gain: 0.940–0.788	0.152	gain: 0.940–0.892	0.048
split info: info([5,4,5])	1.577	split info: info([4,6,4])	1.557	split info: info ([7,7])	1.000	split info: info([8,6])	0.985
gain ratio: 0.247/1.577	0.157	gain ratio: 0.029/1.557	0.019	gain ratio: 0.152/1	0.152	gain ratio: 0.048/0.985	0.049

Gini Index

Gini Index

- Gini index
  - is used in CART algorithm.
  - measures the impurity of *D*
  - considers a binary split for each attribute.
- If a data set *D* contains examples from *m* classes, gini index, *gini*(*D*) is defined as

$$gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

  - where p_i is the relative frequency of class *i* in *D*

Gini Index of a Discrete-Valued Attribute

- To determine the best binary split on A, we examine all of the possible subsets that can be formed using known values of A.
- Need to enumerate all the possible splitting points for each attribute
- If A is a discrete-valued attribute having v distinct values, then there are 2v – 2 possible subsets.

Gini Index

- When considering a binary split, we compute a weighted sum of the impurity of each resulting partition.
- If a data set D is split on A into two subsets D_1 and D_2, the *gini* index *gini*(D) is defined as

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

- First we calculate Gini index for all subsets of an attribute, then the subset that gives the minimum Gini index for **that attribute** is selected.

Gini Index for Continuous-valued Attributes

- For continuous-valued attributes, each possible split-point must be considered.
- The strategy is similar to that described for information gain.
- The point giving the minimum Gini index for a given (continuous-valued) attribute is taken as the split-point of that attribute.
- For continuous-valued attributes
  - May need other tools, e.g., clustering, to get the possible split values
  - Can be modified for categorical attributes

Gini Index

- The reduction in impurity that would be incurred by a binary split on attribute *A is*

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

- The attribute that maximizes the reduction in impurity (or, equivalently, has the minimum Gini index) is selected as the splitting attribute.

Gini Index

- Example:
  - *D* has 9 instances in buys_computer = “yes” and 5 in “no”
  - The impurity of *D:*

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

  - the attribute *income* partitions:
    - {low, medium} & {high}
    - {low, high} & {medium}
    - {low} & {medium, high}

Gini Index

- Example:
  - Suppose the attribute *income* partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$
\begin{aligned}
& Gini_{income \in \{low, medium\}}(D) \\
&= \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2) \\
&= \frac{10}{14}\left(1-\left(\frac{6}{10}\right)^2-\left(\frac{4}{10}\right)^2\right)+\frac{4}{14}\left(1-\left(\frac{1}{4}\right)^2-\left(\frac{3}{4}\right)^2\right) \\
&= 0.450 \\
&= Gini_{income \in \{high\}}(D).
\end{aligned}
$$

  - Similarly, the Gini index values for splits on the remaining subsets are:
    - For {low, high} and {medium} is 0.315
    - For {low} and {medium, high} is 0.300

Gini Index

- The attribute *income* and splitting subsets {low} and {*medium, high*} *and* give the minimum Gini index overall, with a reduction in impurity of:

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

$$\Delta Gini(income) = 0.459 - 0.300 = 0.159$$

- Now we should calculate ΔGini for other attributes including age, student, and credit rate.
- Then we can choose the best attribute for splitting.

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - Information gain:
    - biased towards multivalued attributes
  - Gain ratio:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - Gini index:
    - biased to multivalued attributes
    - has difficulty when # of classes is large

Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on χ2 test for independence
- C-SEP: performs better than information Gain and Gini index in certain cases
- G-statistics: has a close approximation to χ2 distribution
- MDL (Minimal Description Length) principle: the simplest solution is preferred
- Multivariate splits: partition based on multiple variable combinations
  - CART: can find multivariate splits based on a linear combination of attributes.

Attribute Selection Measures

- Which attribute selection measure is the best?
  - All measures have some bias.
  - Most give good results, none is significantly superior than others
  - It has been shown that the time complexity of decision tree induction generally increases exponentially with **tree height**.
  - Hence, measures that tend to produce shallower trees may be preferred.
    - e.g., with multiway rather than binary splits, and that favor more balanced splits

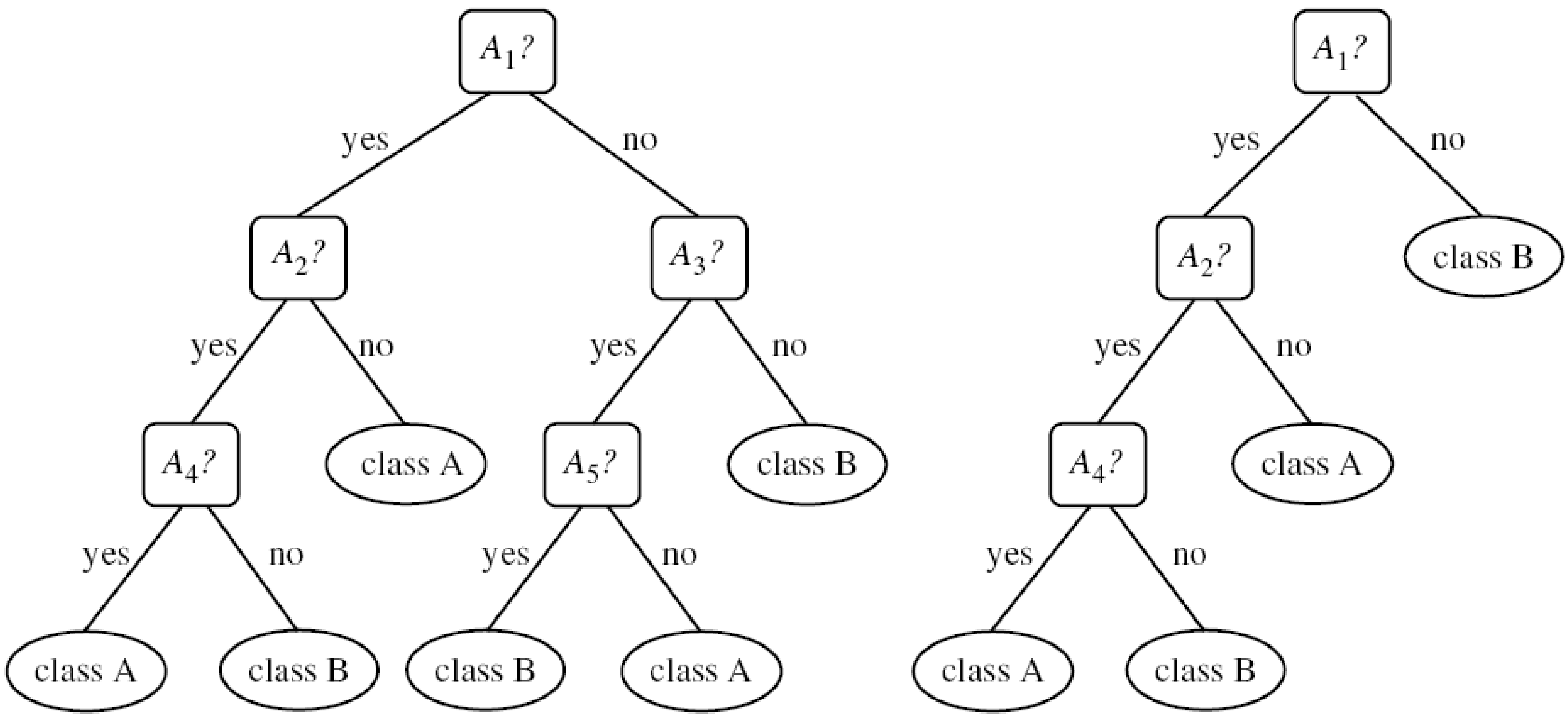
4. Tree Pruning

Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- **Tree Pruning**
  - To prevent overfitting to noise in the data
  - Pruned trees tend to be smaller and less complex and, thus, easier to comprehend.
  - They are usually faster and better at correctly classifying independent test data.

Tree Pruning

- An unpruned decision tree and a pruned version of it.



Tree Pruning

- Two approaches to avoid overfitting
  - Prepruning
    - stop growing a branch when information becomes unreliable
  - Postpruning
    - take a fully-grown decision tree and remove unreliable branches
    - Postpruning preferred in practice

Prepruning

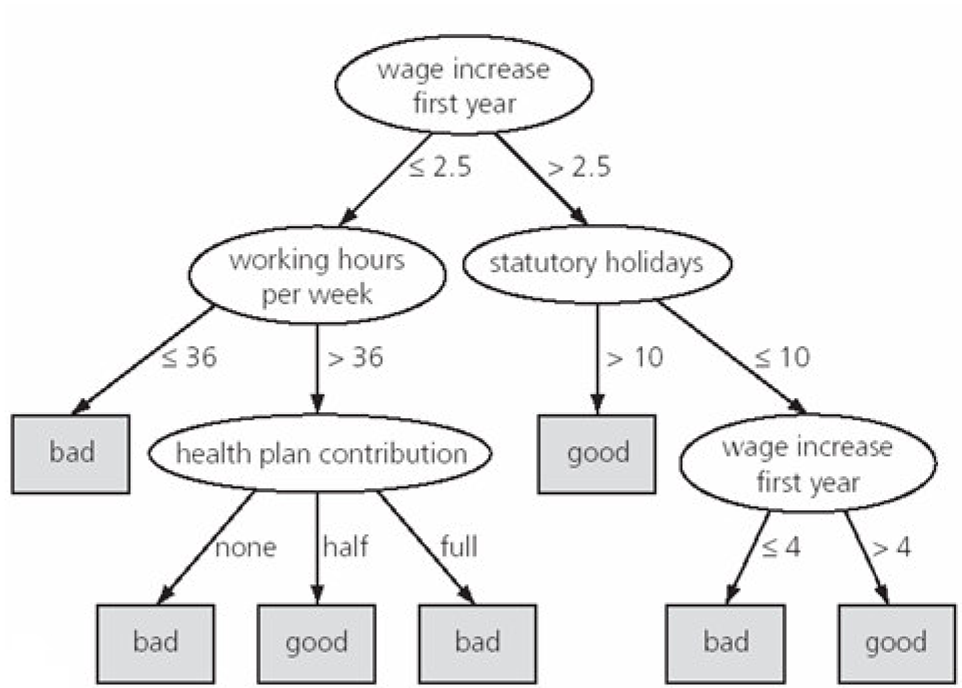
- Based on statistical significance test
  – Stop growing the tree when there is no *statistically significant* association between any attribute and the class at a particular node
- Most popular test: *chi-squared test*
- ID3 used chi-squared test in addition to information gain
  – Only statistically significant attributes were allowed to be selected by information gain procedure

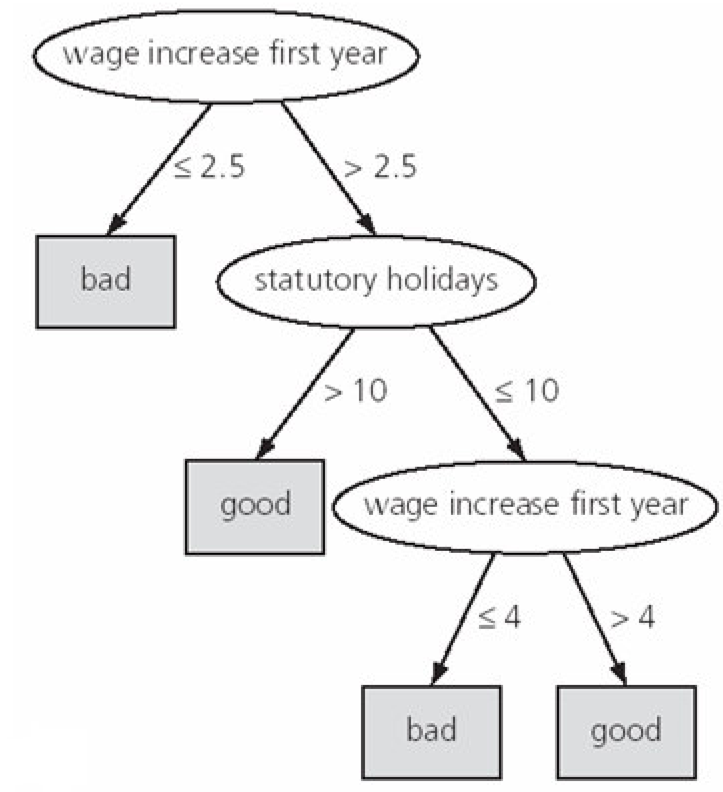
Postpruning

- Postpruning: first, build full tree & Then, prune it
- Two pruning operations:
  - *Subtle replacement*
  - *Subtree raising*
- Possible strategies: error estimation and significance testing

Subtree replacement

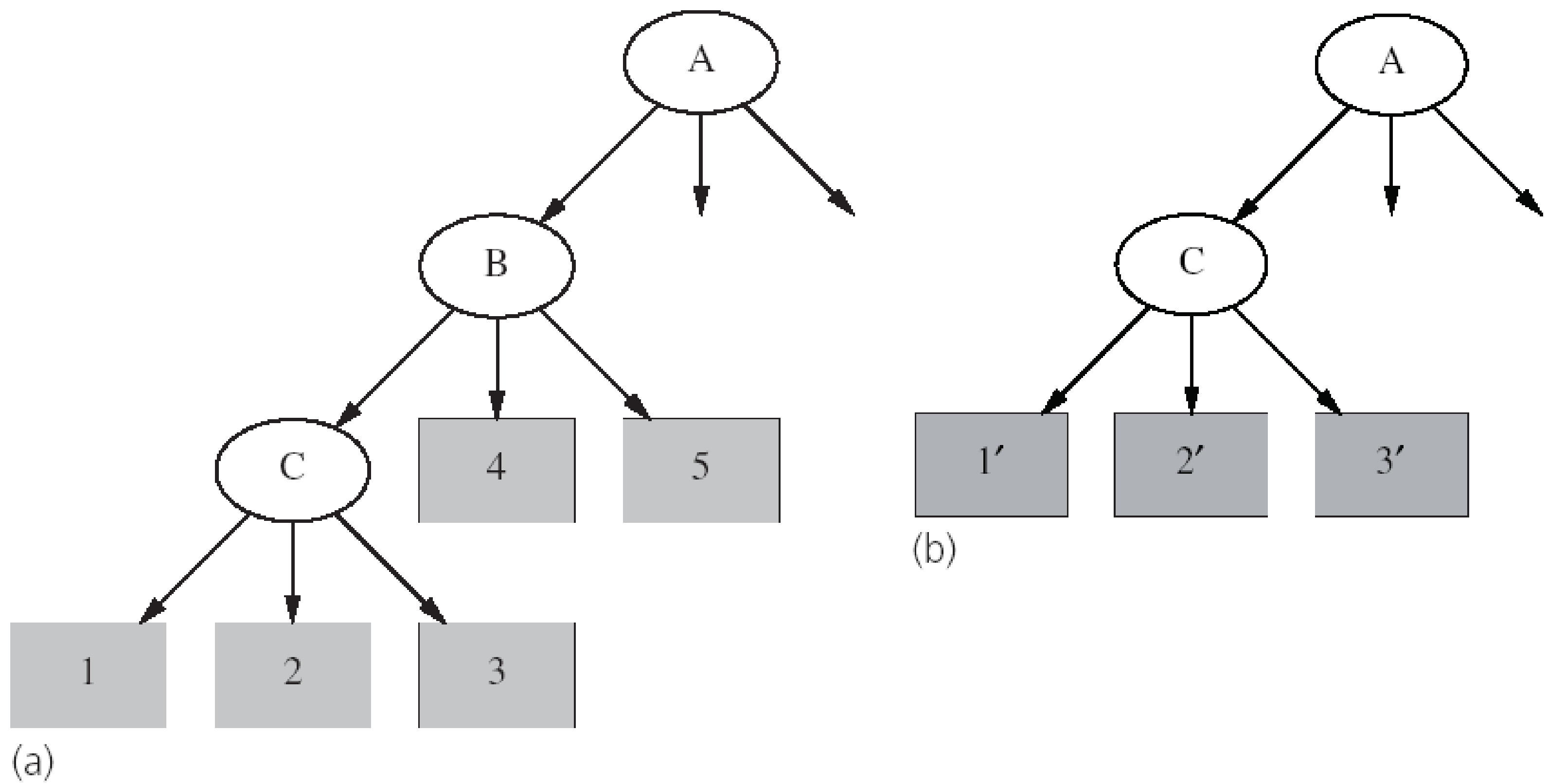
- Subtle replacement*: Bottom-up*
  - To select some subtrees and replace them with single leaves





Subtree raising

- Subtree raising
  - Delete node, redistribute instances
  - Slower than subtree replacement



5. Scalable Decision Tree Induction Methods

Scalable Decision Tree Induction Methods

- Scalability
  - Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- ID3, C4.5, and CART
  - The existing decision tree algorithms has been well established for relatively small data sets.
- The pioneering decision tree algorithms that we have discussed so far have the restriction that the training instances should reside *in memory.*

Scalable Decision Tree Induction Methods

- SLIQ
  - Builds an index for each attribute and only class list and the current attribute list reside in memory
- SPRINT
  - Constructs an attribute list data structure
- PUBLIC
  - Integrates tree splitting and tree pruning: stop growing the tree earlier
- RainForest
  - Builds an AVC-list (attribute, value, class label)
- BOAT
  - Uses bootstrapping to create several small samples

References

References

- J. Han, M. Kamber, **Data Mining: Concepts and Techniques**, Elsevier Inc. (2006). (Chapter 6)
- I. H. Witten and E. Frank, **Data Mining: Practical Machine Learning Tools and Techniques**, 2nd Edition, Elsevier Inc., 2005. (Chapter 6)

The end