Classification

- **Decision Tree Induction**

Classification and Prediction

- **Classification** and **prediction** are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends.
- **Classification** predicts categorical (discrete, unordered) labels,
- **Prediction** models continuous valued functions.
- Many classification and prediction methods have been proposed
  - **Some Classifiers:** decision tree classifiers, Bayesian classifiers, Bayesian belief networks, rule based classifiers, neural network technique, k-nearest-neighbor classifiers, support vector machines, ...
  - **Some Prediction Methods:** linear regression, nonlinear regression, neural network technique, ...
- Classification and prediction have numerous applications, including fraud detection, target marketing, performance prediction, manufacturing, and medical diagnosis.

Classification - A Two-Step Process

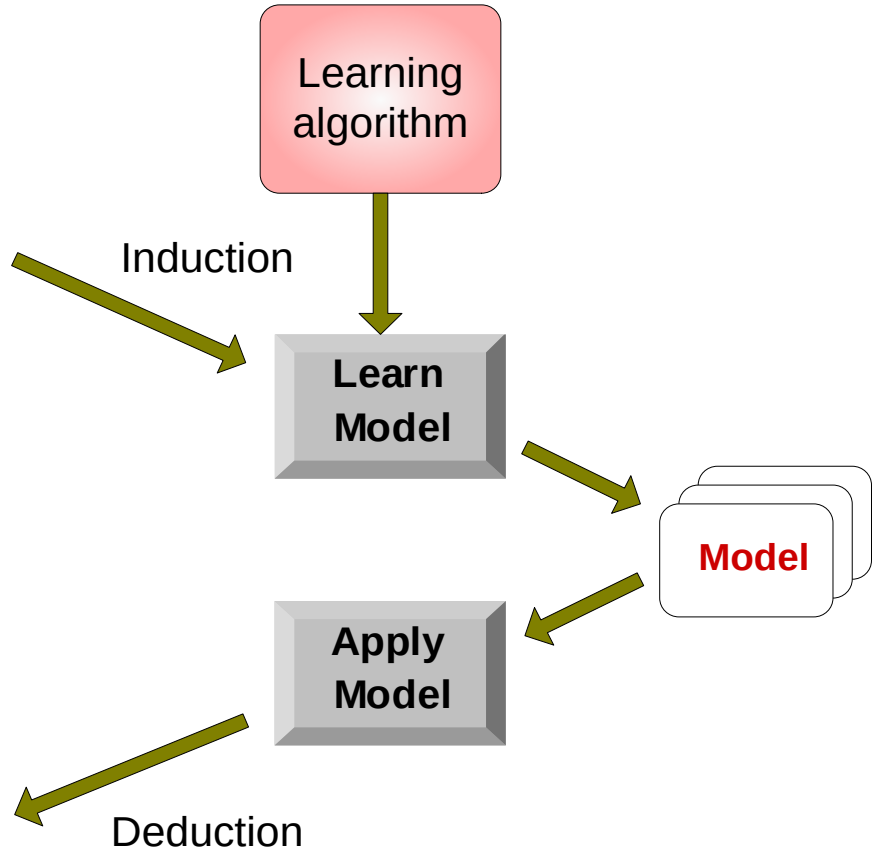
- **Model construction**: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae

- **Model usage**: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data

General Approach for Building Classification Model

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

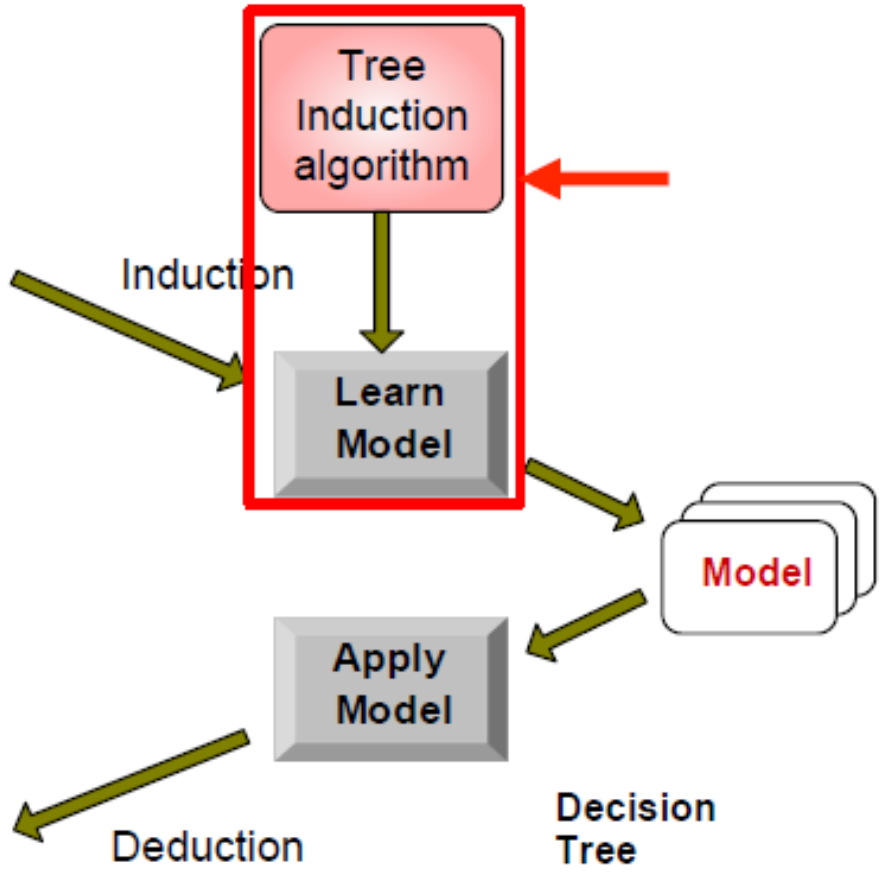
Decision Tree Classification

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

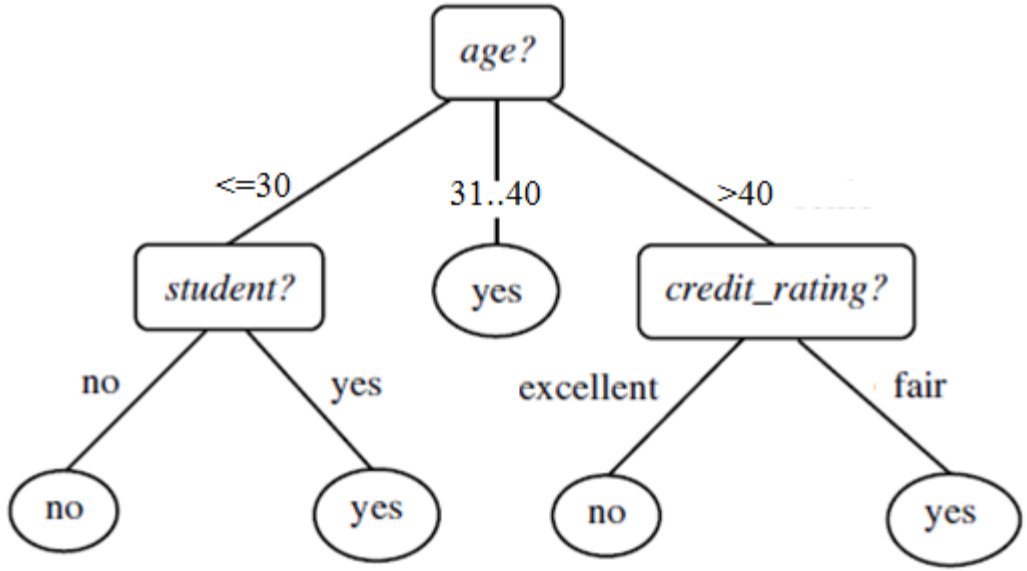
Test Set



Decision Tree Induction

- **Decision tree induction** is the learning of decision trees (which represents discrete-valued functions) from class-labeled training tuples.
- A **decision tree** is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label.

A decision tree for concept ***buys_computer*** ➔



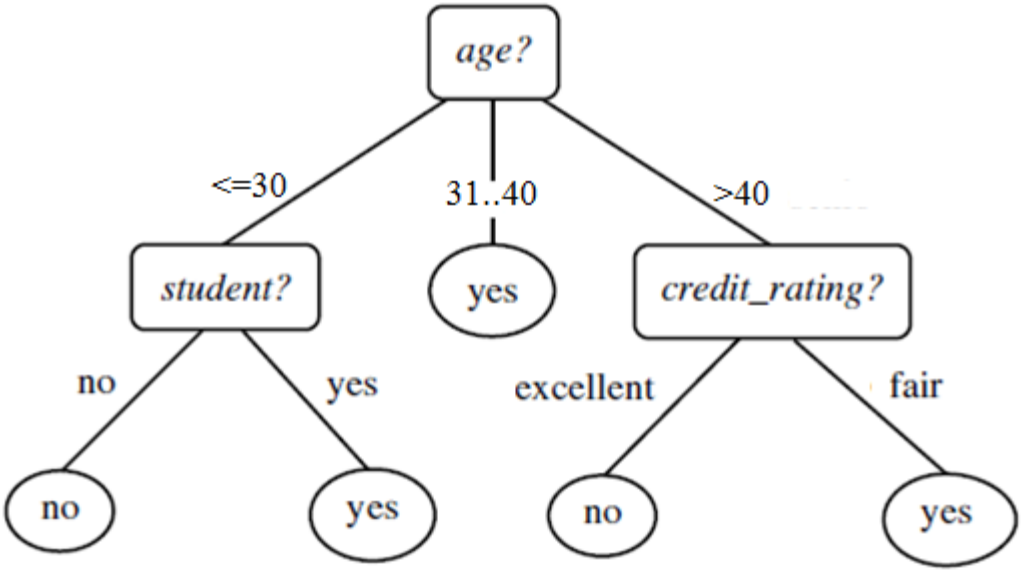
Decision Tree

- **Decision trees** represent a disjunction of conjunctions of constraints on the attribute values of instances.
- Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and
- The tree itself is a disjunction of these conjunctions.

(age= >=30 *and* student = yes)

or (age = 31..40)

or (age = >40 *and* credit_rating = fair)

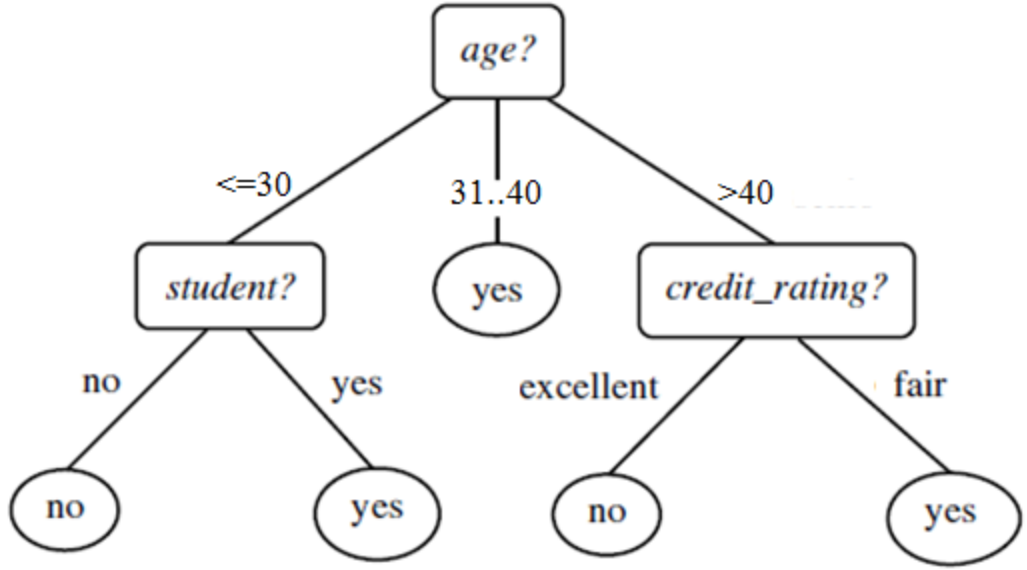


Decision Tree

- Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance.
- Each inner node in the tree specifies a test of some attribute of the instance.
- Each branch descending from a node corresponds to one of the possible values for the attribute.
- Each leaf node assigns a classification.

- The instance
  (age <=30,
  income = low,
  student = no,
  credit_rating = fair)

  is classified as a **negative** instance.



Decision Tree Induction: Example

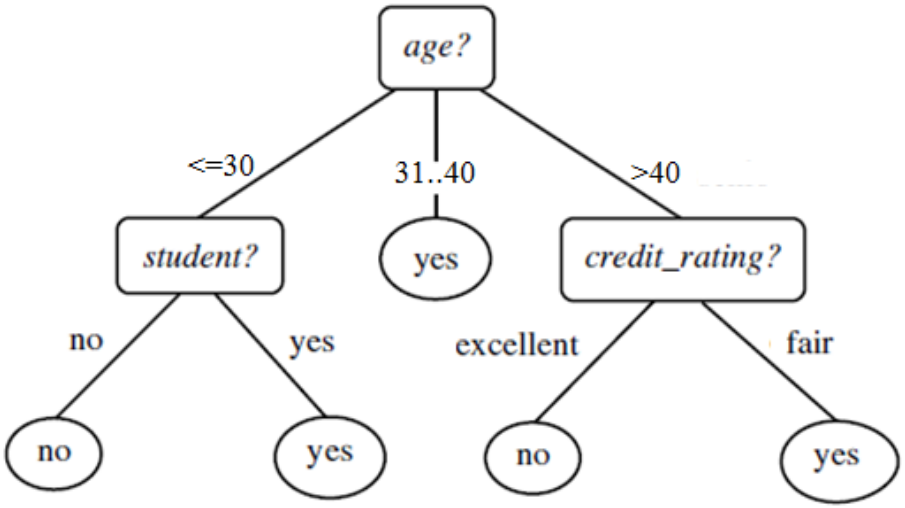
Training Data Set: ***buys_computer***

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

attributes

class attribute

decision tree induction algorithm learns this decision tree from this training data set



Decision Tree Induction Algorithm

Generate_Decision_Tree(D,attribute_list)

create a node *N*;
if tuples in *D* are all of the same class *C* **then**
return *N* as a leaf node labeled with the class *C*;
if *attribute list* is empty **then**
return *N* as a leaf node labeled with the majority class in *D*; // majority voting
apply *attribute_selection_method(D, attribute list)* to find the **best *splitting_criterion***;
label node *N* with *splitting criterion*;
if *splitting attribute* is discrete-valued and multiway splits allowed **then**
attribute list ← *attribute list - splitting attribute*; // remove *splitting_attribute*
for each outcome *j* of *splitting_criterion*
// partition the tuples and grow subtrees for each partition
let D_j be the set of data tuples in *D* satisfying outcome *j*; // a partition
if D_j is empty **then**
attach a leaf labeled with the majority class in *D* to node *N*;
else attach the node returned by *Generate_decision_tree*(D_j, *attribute list*) to node *N*;
return *N*;

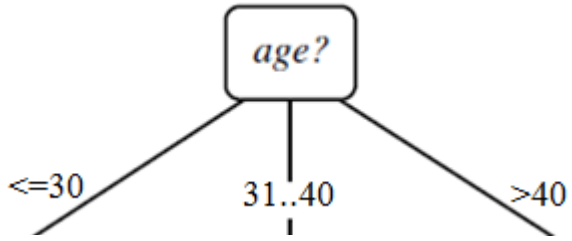
Decision Tree Induction Algorithm: Example

age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

select **best attribute**
that splits this data set.

Decision Tree Induction Algorithm: Example

age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



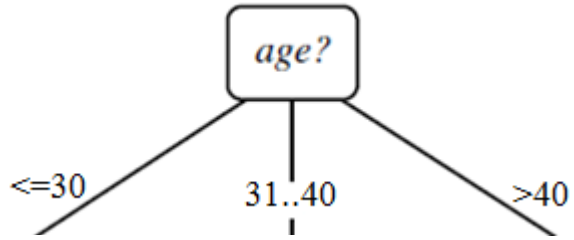
age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

age	income	student	cr	buys
31…40	high	no	fair	yes
31…40	low	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

Decision Tree Induction Algorithm: Example

age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

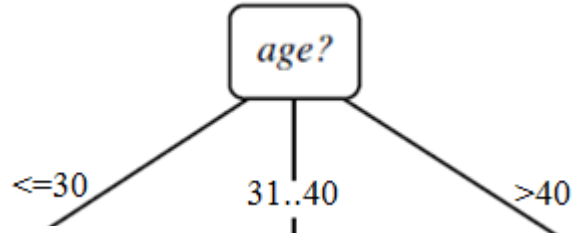
age	income	student	cr	buys
31…40	high	no	fair	yes
31…40	low	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes

yes

all examples are yes

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

Decision Tree Induction Algorithm: Example



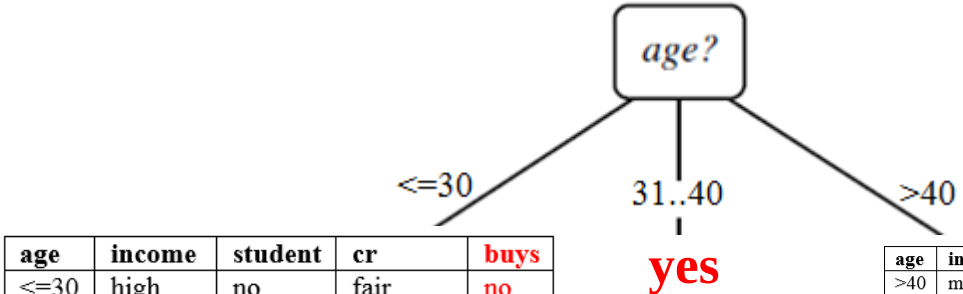
age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

yes

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

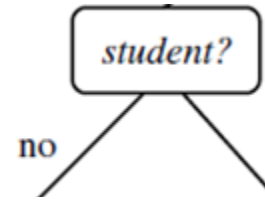
select **best attribute**
that splits this data set.

Decision Tree Induction Algorithm: Example



age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

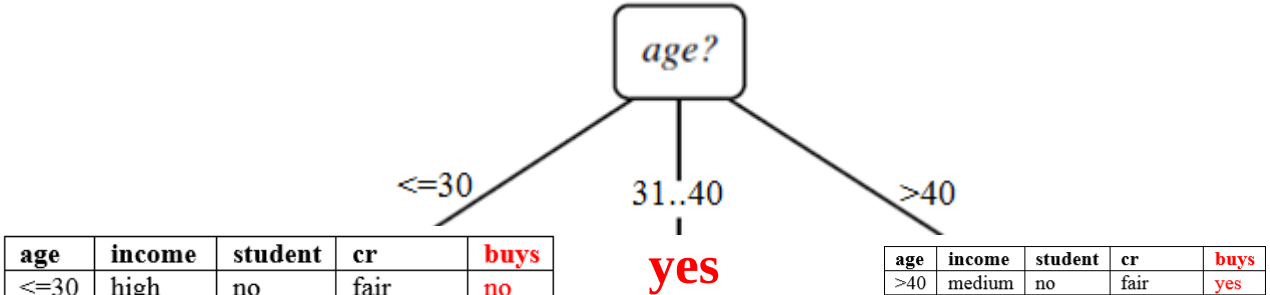
age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no



age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no

age	income	student	cr	buys
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

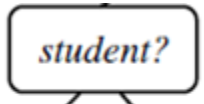
Decision Tree Induction Algorithm: Example



age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

yes

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no



age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no

no

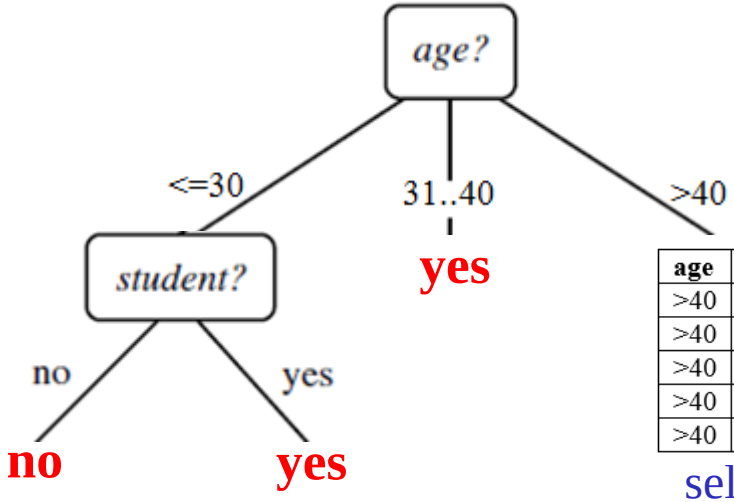
all examples are no

age	income	student	cr	buys
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

yes

all examples are yes

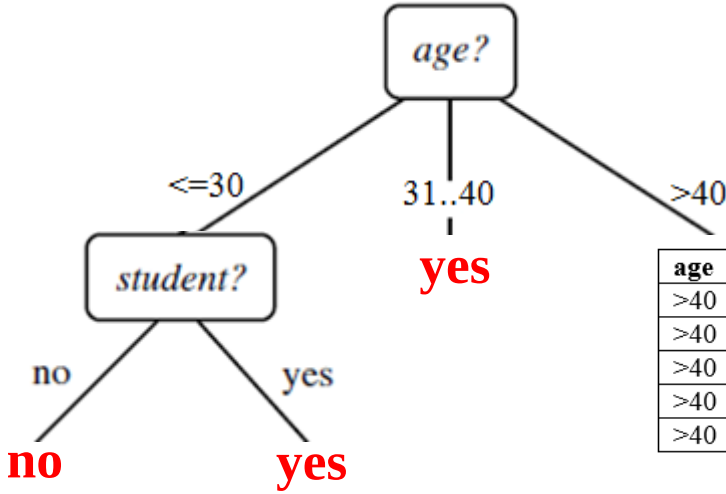
Decision Tree Induction Algorithm: Example



age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

select **best attribute**
that splits this data set.

Decision Tree Induction Algorithm: Example



age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

credit_rating?

excellent fair

age	income	student	cr	buys
>40	low	yes	excellent	no
>40	medium	no	excellent	no

no

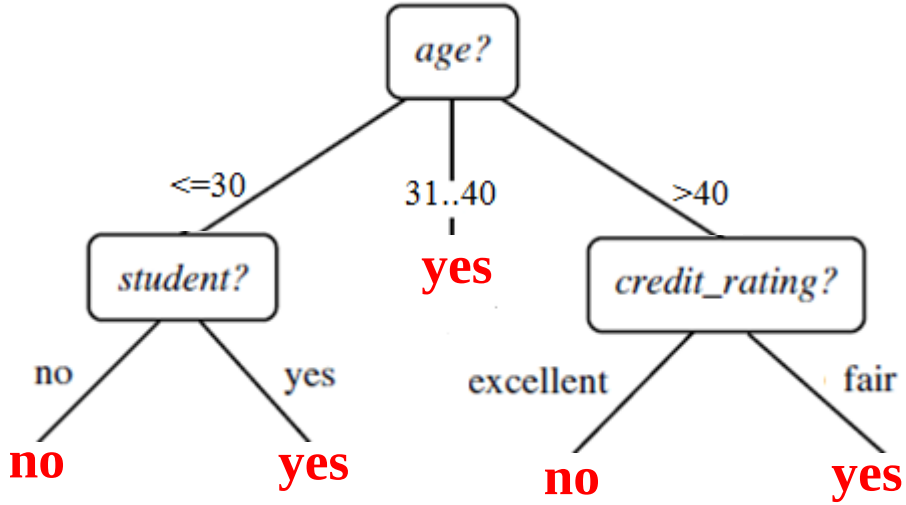
all examples are no

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	medium	yes	fair	yes

yes

all examples are yes

Decision Tree Induction Algorithm: Example



Design Issues of Decision Tree Induction

- **How should training examples be split?**
  - Method for specifying test condition (**Splitting Criterion**)
    - depending on attribute types
  - Measure for evaluating the goodness of a test condition (**Selecting Best Attribute**)
    - information gain, gain ratio, Gini index, misclassification error, statistical test, …

- **How should the splitting procedure stop?**
  - Stop splitting if all the examples belong to the same class
  - Early termination depending on the results of a statistical test.

Methods for Expressing Test Conditions
Splitting Criterion

- **Depends on attribute types**
  - Binary
  - Nominal
  - Ordinal
  - Continuous

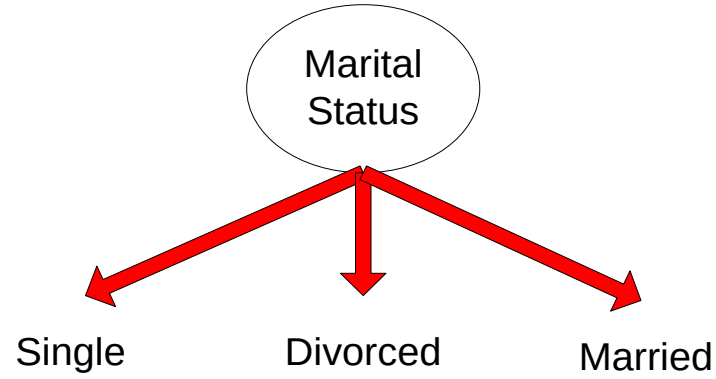
- **Depends on number of ways to split**
  - 2-way split (Binary split)
  - Multi-way split

Test Condition for Nominal Attributes

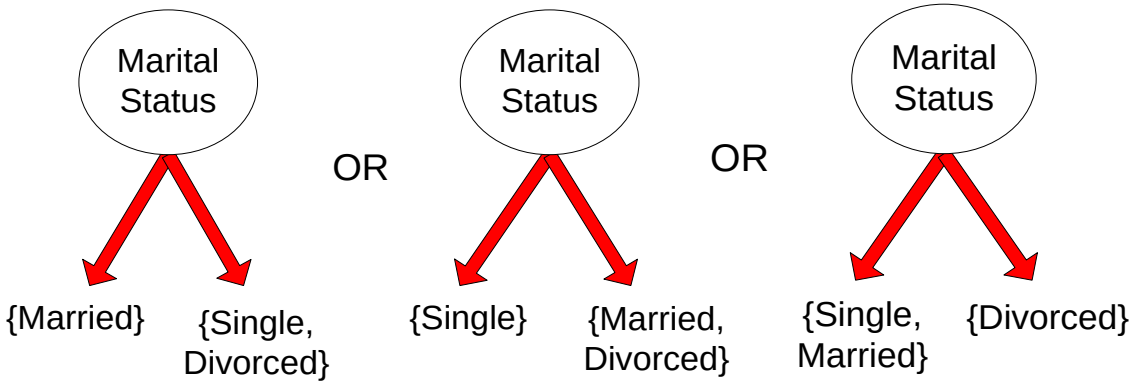
- Multi-way split:
  - Use as many partitions as distinct values of the attribute.

Example:

MaritalStatus: {Single,Divorced,Married}



- Binary split:
  - Divide attribute values into two non-empty subsets.

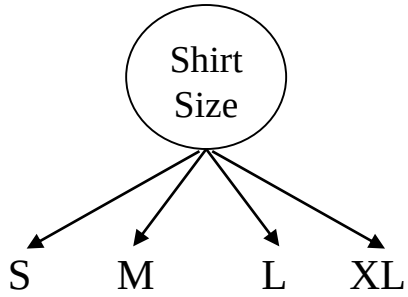


Test Condition for Ordinal Attributes

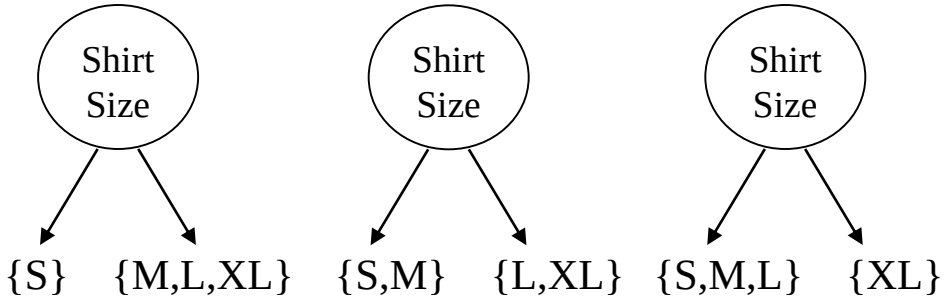
- Multi-way split:
  - Use as many partitions as distinct values of the attribute.

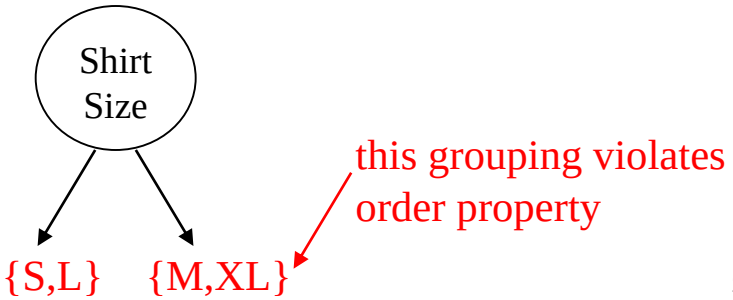
Example:

ShirtSize: {S,M,L,XL}



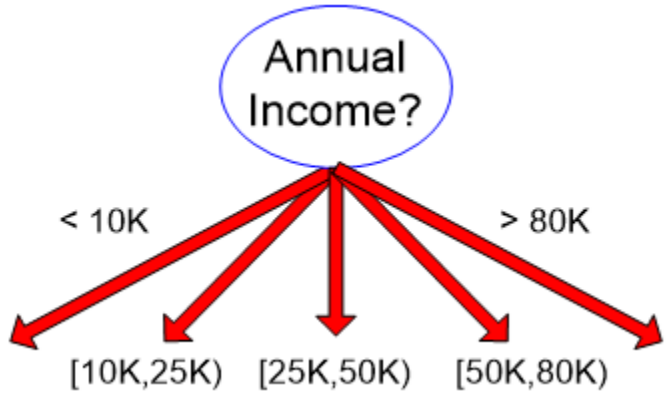
- Binary split:
  - Divide attribute values into two non-empty subsets.
  - Preserve order property among attribute values



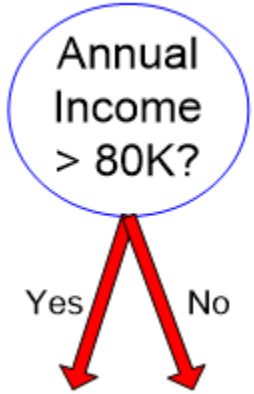


Test Condition for Continuous Attributes

- Multi-way split:
  - Discretize the continuous attribute.
  - Discretization to form an ordinal categorical attribute



- Binary split:
  - Divide attribute values into two non-empty subsets from a cut-off point.
  - Binary Decision: (A≤v) or (A> v)
    - consider all possible splits and finds the best cut

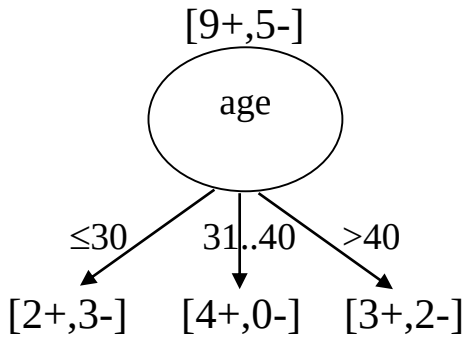


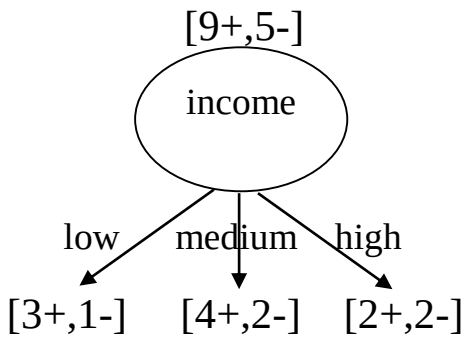
How to Determine the Best Split

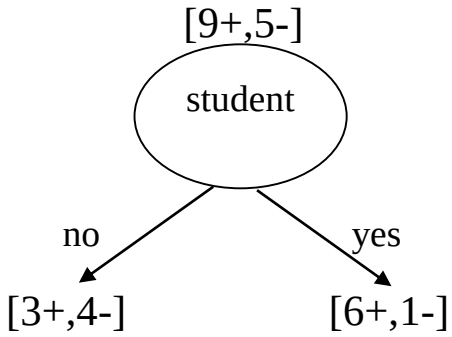
- There are 9 positive examples and 5 negative examples.

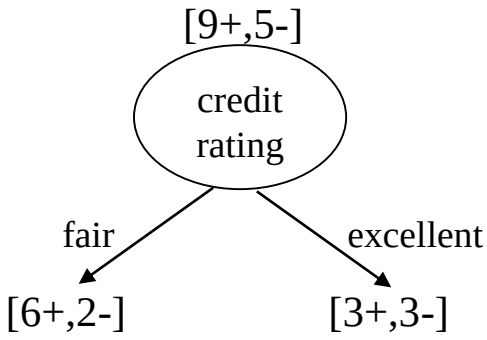
- **Which test condition (attribute) is the best?**

age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no









How to Determine the Best Split

- Greedy approach:
  - Nodes with **homogeneous (purer)** class distribution are preferred

- Need a **measure of node impurity**:

[5+,5-]	**[9+,1-]**
Non-homogeneous	Homogeneous
High degree of impurity	Low degree of impurity

- **Measures of Node Impurity:**
  - **entropy, Gini index, misclassification error, …**

Finding the Best Split

1. Compute impurity measure (P) before splitting

2. Compute impurity measure (M) after splitting
   - Compute impurity measure of each child node
   - M is the weighted impurity of children

3. Choose the attribute test condition that produces the highest gain

   Gain = P - M

   or equivalently, lowest impurity measure after splitting (M)

Attribute Selection Measures

- An **attribute selection measure** is a heuristic for *selecting the splitting criterion* that **best** separates a given data set D of class-labeled training tuples into individual classes.
  - 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 tuples that fall into a given partition would belong to the same class).

- The **attribute having the best score for the measure** is chosen as the splitting attribute for the given tuples.

- Three popular **attribute selection measures:**

  information gain, ***gain ratio***, and ***gini index***.

Measure of Impurity: Entropy

- Given a collection S, containing positive and negative examples of some target concept, the ***entropy of S*** relative to this boolean classification is:

$$\mathbf{Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-}$$

- **S** is a sample of training examples
- $\mathbf{p_+}$ is the proportion of positive examples
- $\mathbf{p_-}$ is the proportion of negative examples

Measure of Impurity: Entropy

Entropy([12+,5-] =

– (12/17) $\log_2$(12/17) – (5/17) $\log_2$(5/17) = 0.874

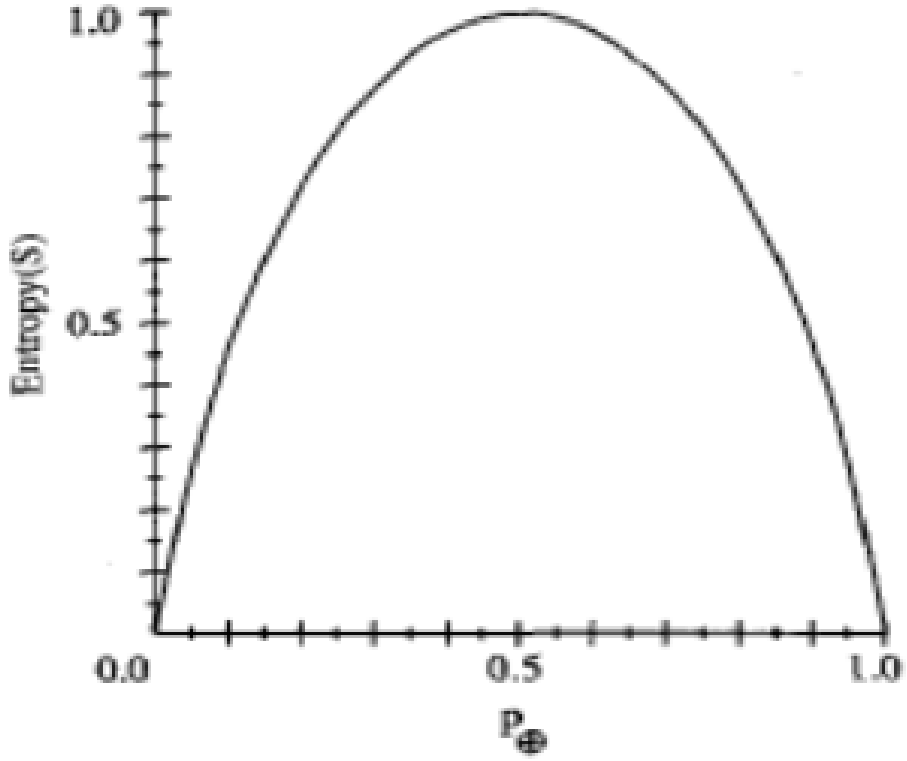
Entropy([8+,8-] =

– (8/16) $\log_2$(8/16) – (8/16) $\log_2$(8/16) = 1.0

Entropy([8+,0-] =

– (8/8) $\log_2$(8/8) – (0/8) $\log_2$(0/8) = 0.0

– It is assumed that $\log_2$(0) is 0



Entropy – Non-Boolean Target Classification

- If the target attribute can take on c different values, then the entropy of S relative to this c-wise classification is defined as

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

- p_i is the proportion of S belonging to class i.
- The logarithm is still base 2 because entropy is a measure of the expected encoding length measured in bits.
- If the target attribute can take on c possible values, the entropy can be as large as $\log_2 c$.

Entropy – Informaton Theory

- Entropy(S) = ***expected number of bits needed to encode*** class (+ or -) of randomly drawn members of S (under the optimal, shortest length-code)
  - if p+ is 1, the receiver knows the drawn example will be positive, so no message need be sent, and the entropy is zero.
  - if p+ is 0.5, one bit is required to indicate whether the drawn example is positive or negative.
  - if p+ is 0.8, then a collection of messages can be encoded using on average less than 1 bit per message by assigning shorter codes to collections of positive examples and longer codes to less likely negative examples.
- Information theory optimal length code assign $\mathbf{-\log_2 p}$ bits to messages having probability p.
- So the ***expected number of bits to encode*** (+ or -) of random member of S:

$$\mathbf{- p_+ \log_2 p_+ - p_- \log_2 p_-}$$

Attribute Selection Measure: Information Gain

- **Select the attribute with the highest information gain**
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- **Expected information (entropy)** needed to classify a tuple in D:

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

- **Information needed** (after using A to split D into v partitions) to classify D:

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

- **Information gained by branching on attribute A**

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

Attribute Selection Measure: Information Gain

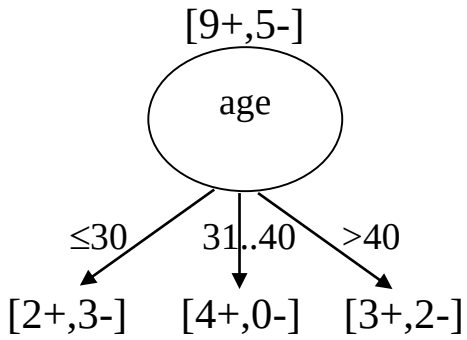
- There are 9 positive examples and 5 negative examples.

  D: [9+,5-]

age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Info(D) = Entropy(D) =

$$-(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$$

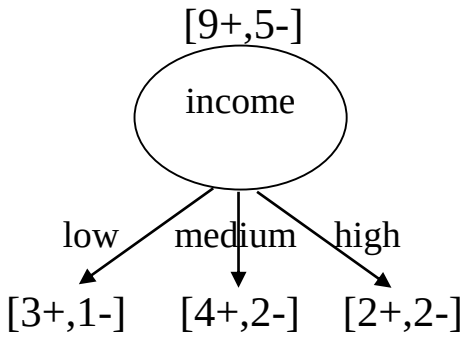


$$\text{Info}_{age}(D) = 5/14 * \text{Info}([2+,3-]) + 4/14 * \text{Info}([4+,0-]) + 5/14 * \text{Info}([3+,2-])$$
$$= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971$$
$$= 0.694$$

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

Attribute Selection Measure: Information Gain

$$Info(D) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$$

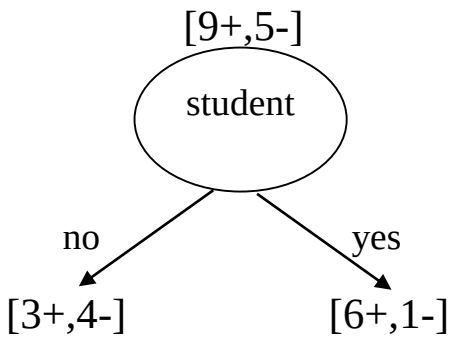


$$Info_{income}(D) = 4/14 * Info([3+,1-]) + 6/14 * Info([4+,2-]) + 4/14 * Info([2+,2-])$$
$$= (4/14)*0.811 + (6/14)*0.911 + (4/14)*1.0$$
$$= 0.911$$

$$Gain(income) = Info(D) - Info_{income}(D)$$
$$= 0.940 - 0.911 = 0.029$$

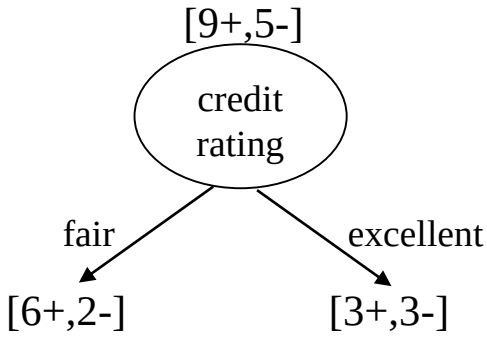
Attribute Selection Measure: Information Gain

$$Info(D) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$$



$$Info_{student}(D) = 7/14 * Info([3+,4-]) + 7/14 * Info([6+,1-])$$
$$= (7/14)*0.985 + (7/14)*0.592 = 0.789$$

$$Gain(student) = Info(D) - Info_{student}(D)$$
$$= 0.940 - 0.789 = 0.151$$



$$Info_{cr}(D) = 8/14 * Info([6+,2-]) + 6/14 * Info([3+,3-])$$
$$= (8/14)*0.811 + (6/14)*1.0 = 0.892$$

$$Gain(cr) = Info(D) - Info_{cr}(D)$$
$$= 0.940 - 0.892 = 0.048$$

Computing Information Gain for Continuous Valued Attributes

- Let attribute A be a continuous-valued attribute
- 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}
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Continuous Valued Attributes - Example

Temperature:	40	48	60	72	80	90
ClassAttribute :	No	No	Yes	Yes	Yes	No

Two candidate thresholds: (48+60)/2=54 (80+90)/2=85

Check the information gain for new boolean attributes:

$Temperature_{>54}$ $Temperature_{>85}$

Use these new boolean attributes same as other discrete valued attributes.

Gain Ratio for Attribute Selection

- **Information gain** measure is biased towards attributes with a large number of values
- Some decision tree induction algorithms (such as C4.5) uses **gain ratio** to overcome the problem (normalization to information gain)

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

GainRatio(A) = Gain(A) / $SplitInfo_A(D)$

- Ex.

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557.$$

GainRatio(income) = 0.029 / 1.557 = 0.019

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

Measure of Impurity: GINI Index

- **Gini Index** for a given node t :

$$\mathbf{GINI(t) = 1 - \sum_{j} [p(j|t)]^2}$$

(NOTE: **p(j|t)** is the relative frequency of class j at node t).

  - Maximum (1 - 1/n_c) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Measure of Impurity: GINI Index

Computing Gini Index of a Single Node

$$\mathbf{GINI(t) = 1 - \sum_{j} [p(j|t)]^2}$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – P(C1)² – P(C2)² = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – (1/6)² – (5/6)² = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – (2/6)² – (4/6)² = 0.444

Measure of Impurity: GINI Index

Computing Gini Index for a Collection of Nodes

- When a node **parent** is split into k partitions (children)

$$\mathbf{GINI_{split}} = \sum_{\mathbf{i=1}}^{\mathbf{k}} \frac{\mathbf{n_i}}{\mathbf{n}} \mathbf{GINI(i)}$$

where, n_i = number of examples at child i,
n = number of examples at parent node p,
GINI(i) = GINI Index value of child i.

Computing GAIN based on GINI Index:

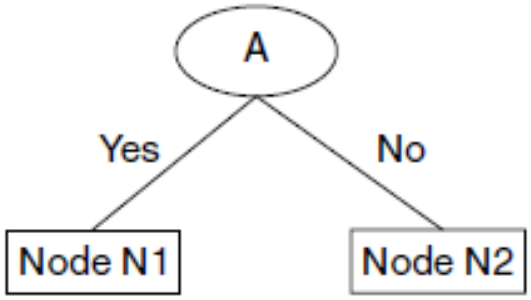
- The **gain** is a criterion that can be used to determine the goodness of a split:

$$\mathbf{GAIN = GINI(parent) - GINI_{split}}$$

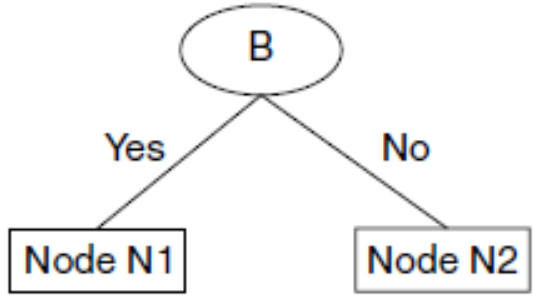
- Choose the attribute that minimizes weighted average Gini Index of the children

- **Gini Index** is used in decision tree algorithms such as CART, SLIQ, SPRINT

Binary Attributes: Computing Gini Index



	Parent
C0	6
C1	6
Gini = 0.500	



	N1	N2
C0	4	2
C1	3	3

	N1	N2
C0	1	5
C1	4	2

Gini(N1) = $1 - (4/7)^2 - (3/7)^2 = 0.49$

Gini(N2) = $1 - (2/5)^2 - (3/5)^2 = 0.48$

$Gini_{splitA}$ = 7/12 * Gini(N1) + 5/12 * Gini(N2)

= 0.486

Gini(N1) = $1 - (1/5)^2 - (4/5)^2 = 0.32$

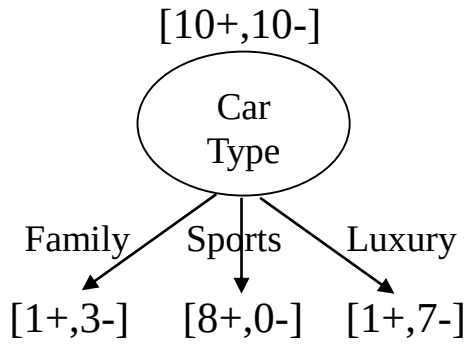
Gini(N2) = $1 - (5/7)^2 - (2/7)^2 = 0.408$

$Gini_{splitB}$ = 5/12 * Gini(N1) + 7/12 * Gini(N2)

= 0.371

Categorical Attributes: Computing Gini Index

- **Multiway split:**

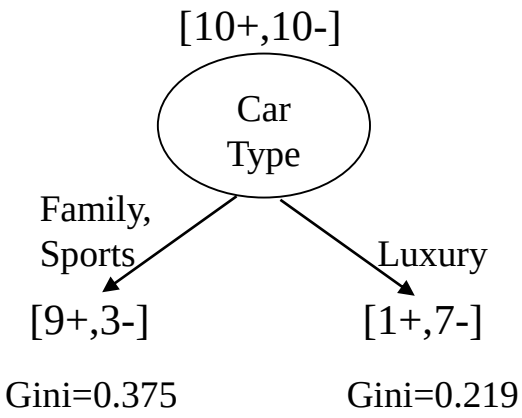


Gini(N1) = 1 - $(1/4)^2$ - $(3/4)^2$ = 0.375
Gini(N2) = 1 - $(8/8)^2$ - $(0/8)^2$ = 0
Gini(N3) = 1 - $(1/8)^2$ - $(7/8)^2$ = 0.219

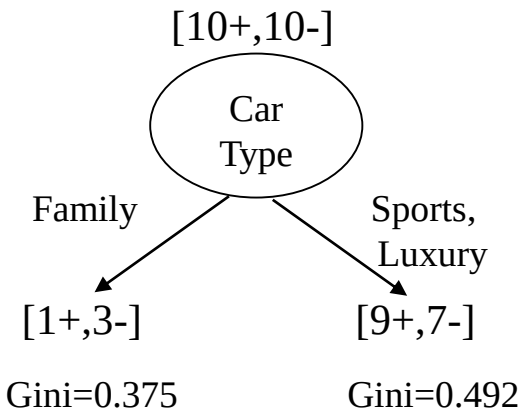
$Gini_{splitA}$ = 4/20 * Gini(N1) + 8/20 * Gini(N2) + 8/20 *Gini(N3) = 0.163

Categorical Attributes: Computing Gini Index

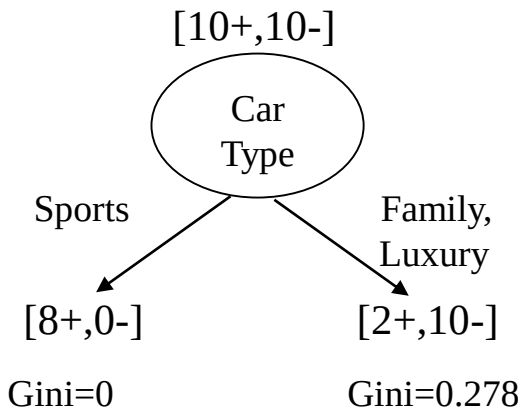
- **Binary split:** find best partition of values (least Gini value)



$Gini_{split}$ = 0.313



$Gini_{split}$ = 0.469



$Gini_{split}$ = 0.168

Continuous Attributes: Computing Gini Index

- Sort the attribute on values
- Choose the split position that has the **least Gini index**

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual Income										
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72	80	87	92	97	110	122	172	230

	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		*0.300*		0.343		0.375		0.400		0.420	

best split

Measure of Impurity: Misclassification Error

- Classification error at a node t :

$$\mathbf{Error(t) = 1 - \max_i P(i|t)}$$

- Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information
- Minimum (0) when all records belong to one class, implying most interesting information

Measure of Impurity: Misclassification Error

Computing Error of a Single Node

$$\mathbf{Error(t) = 1 - \max_i P(i|t)}$$

C1	0
C2	6

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	1
C2	5

P(C1) = 1/6 P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

C1	2
C2	4

P(C1) = 2/6 P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparing Attribute Selection Measures

- All selection measures, in general, return good results but all measures have some bias.
  - **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
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- G-statistic: has a close approximation to χ^2 distribution
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.

- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others.

Model Overfitting

Training error: Train a model on the training dataset, then test the model on the same training set. The error rate is called “training error,” which evaluates how well the model fits the training data.

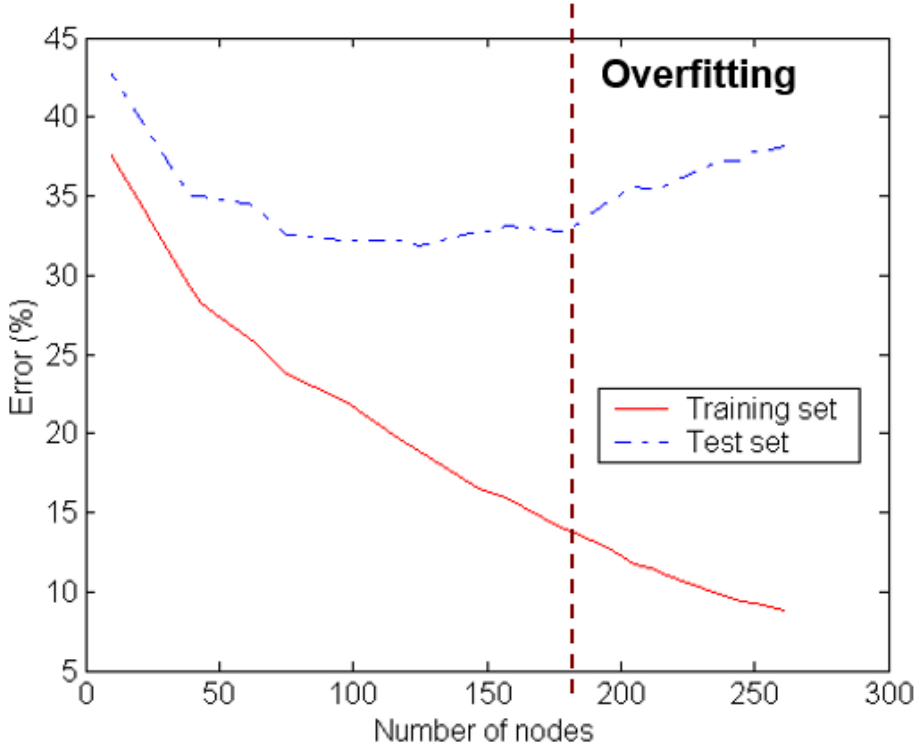
Test error: Test the model on a test dataset that is different from the training set. The error rate is called “test error,” which evaluates how well the model generalizes to unseen data.

Overfitting means a model fits the training data very well but generalizes to unseen data poorly.

How do I know if my model is overfitting?

- Your model is overfitting if its training error is small (fits well with training data) but the test error is large (generalizes poorly to unseen data).

Underfitting and Overfitting



Overfitting due to
- Noise
- Insufficient Examples

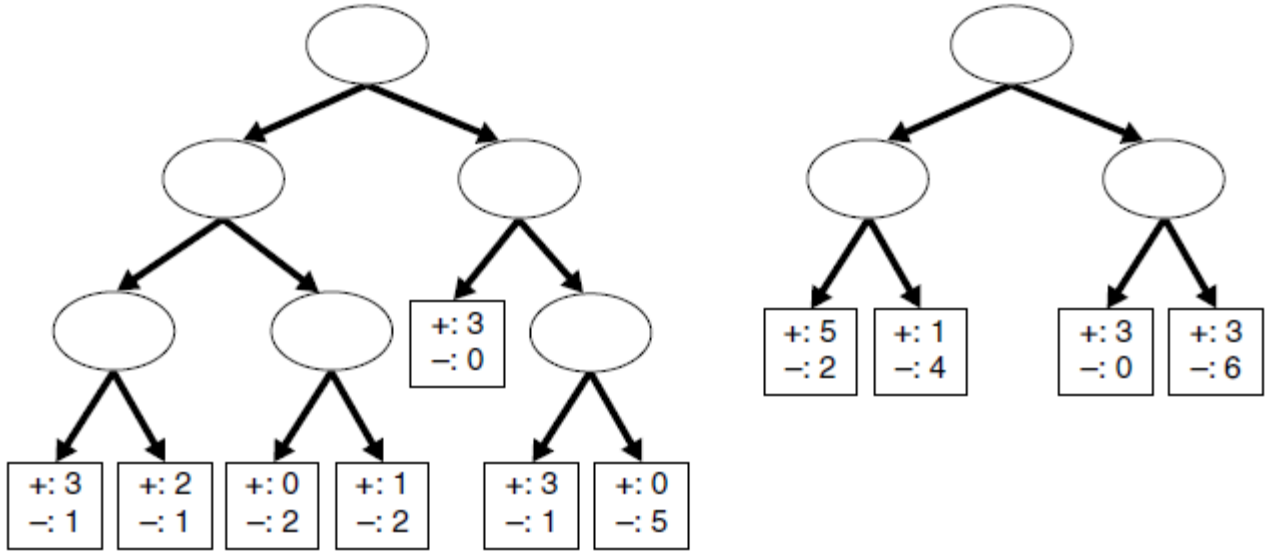
Two approaches to **avoid overfitting**

- ***Prepruning***: Halt tree construction early‑do not split a node if this would result in the goodness measure falling below a threshold
- ***Postpruning***: Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees

- **Overfitting** means a model fits the training data very well but generalizes unseen data poorly.
- **Overfitting:** complex models are more likely to overfit than simple models
- **Underfitting:** when model is too simple, both training and test errors are large

Occam's Razor

- **Occam's Razor: Given two models of similar generalization errors, one should prefer the simpler model over the more complex model.**
  - For complex models, there is a greater chance that they are fitted accidentally by errors in data.
- **Therefore, one should include model complexity when evaluating a model.**



Decision Tree, T_L Decision Tree, T_R

Incorporating Model Complexity

- **Overfitting** results in decision trees that are more complex than necessary

- **Training error** no longer provides a good estimate of how well the tree will perform on previously unseen records

- Need new ways for estimating errors (i.e. we need to incorporate Model Complexity).
  - Estimating Generalization Errors
  - Another way to incorporate model complexity is based on an information-theoretic approach known as the ***minimum description length***

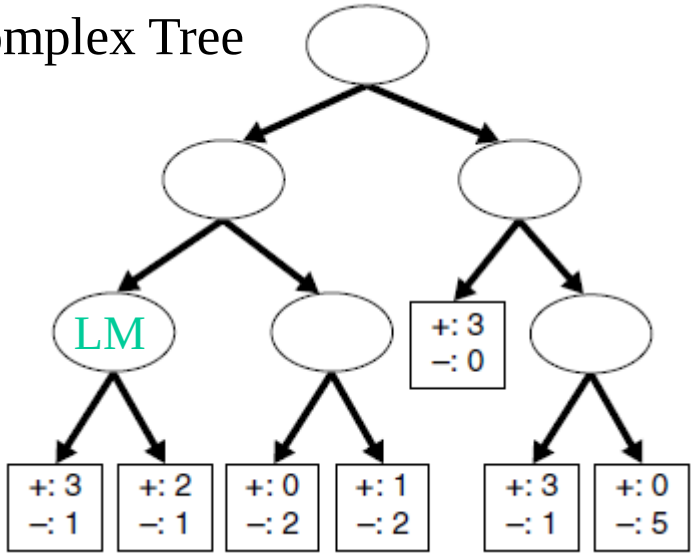
Estimating Generalization Errors

- **Re-substitution errors:** error on training (Σ e(t))
- **Generalization errors:** error on testing (Σ e'(t))

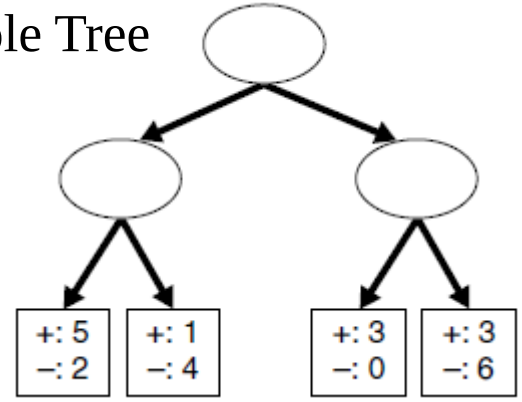
- **Methods for estimating generalization errors:**
  - **Optimistic approach:** e'(t) = e(t)
  - **Pessimistic approach:**
    - For each leaf node: e'(t) = (e(t)+0.5)
    - Total errors: e'(T) = e(T) + N × 0.5 (N: number of leaf nodes)
      - For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):
        Training error = 10/1000 = 1%
        Generalization error = (10 + 30×0.5)/1000 = 2.5%
  - **Reduced error pruning (REP):**
    - uses validation data set to estimate generalization error

Estimating Generalization Errors - Example

Complex Tree



Simple Tree



What are generalization errors for these two decision trees and Complex Tree without **left most inner node** using two approaches?

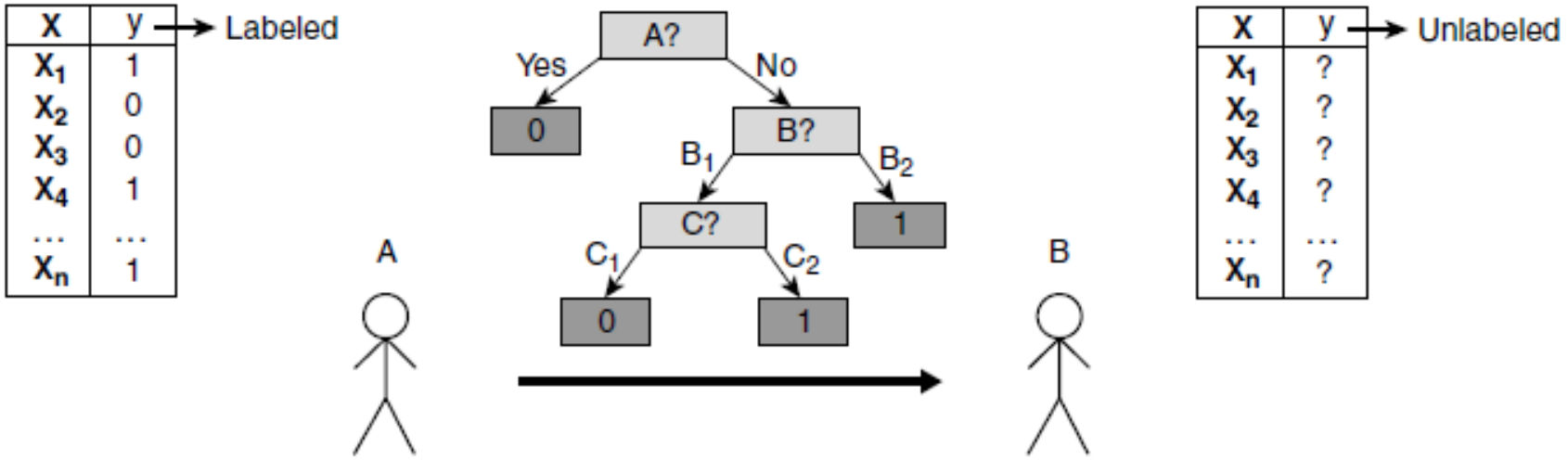
Optimistic approach:

Complex Tree Error Rate: 4/24
Simple Tree Error Rate: 6/24
Complex Tree Without LM: 4/24

Pessimistic approach:

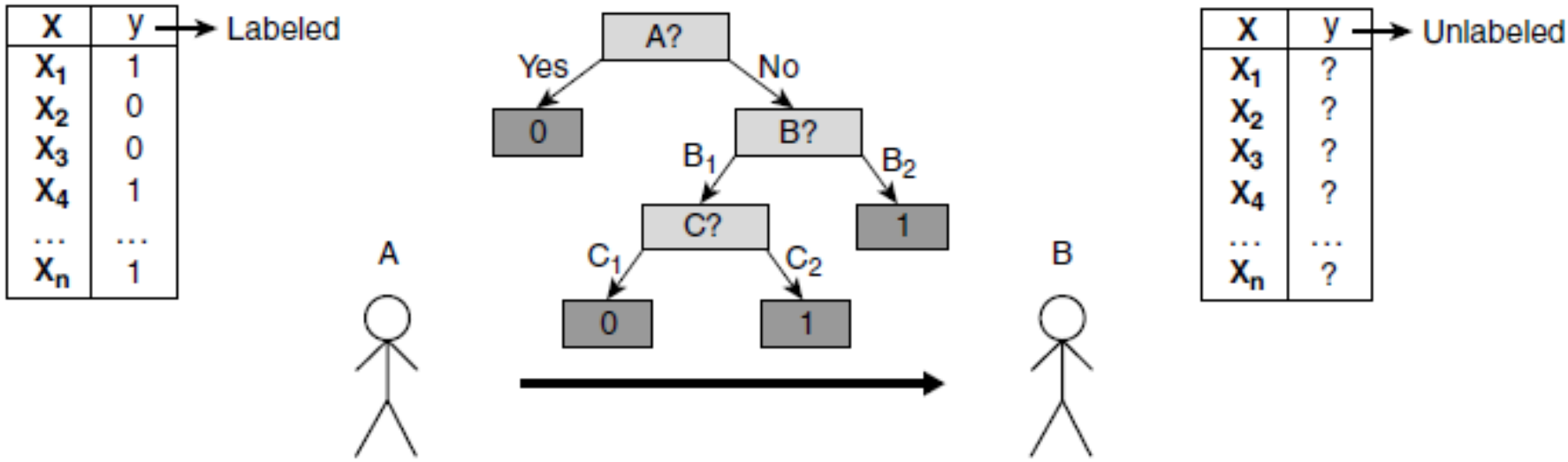
Complex Tree Error Rate: 4/24 + (7*0.5)/24 = 7.5/24
Simple Tree Error Rate: 6/24 + (4*0.5)/24 = 8/24
Complex Tree Without LM: 4/24 + (6*0.5)/24 = 7/24

Minimum Description Length (MDL)



- **A** builds a classification model that summarizes the relationship between **x** and **y**.
- The model can be encoded in a compact form before being transmitted to **B**.
- If the model is 100% accurate, then the cost of transmission is equivalent to the cost of encoding the model.
- Otherwise, **A** must also transmit information about which record is classified incorrectly by the model.

Minimum Description Length (MDL)



Cost(Model,Data) = Cost(Data|Model) + Cost(Model)

- Cost is the number of bits needed for encoding.
- Search for the least costly model.

Cost(Data|Model) encodes the misclassification errors.

Cost(Model) uses node encoding (number of children) plus splitting condition encoding.

Minimum Description Length (MDL)

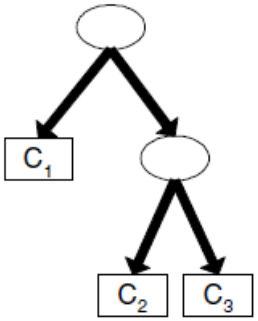
- The **total description length of a tree** is given by:

  Cost(tree, data) = Cost(tree) + Cost(data|tree)

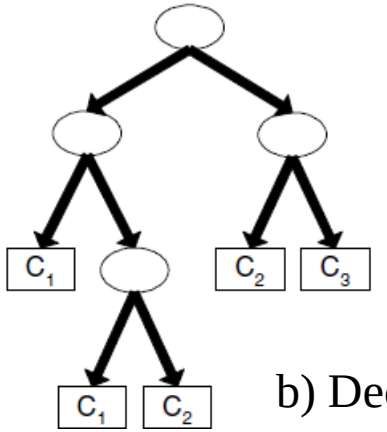
- Each internal node of the tree is encoded by the ID of the splitting attribute. If there are **m** attributes, the cost of encoding each attribute is $\mathbf{\log_2 m}$ bits.
- Each leaf is encoded using the ID of the class it is associated with. If there are **k** classes, the cost of encoding a class is $\mathbf{\log_2 k}$ bits.
- **Cost(tree)** is the cost of encoding all the nodes in the tree.
  - To simplify the computation, we can assume that the total cost of the tree is obtained by adding up the costs of encoding each internal node and each leaf node.
- **Cost(data|tree)** is encoded using the classification errors the tree commits on the training set. Each error is encoded by $\mathbf{\log_2 n}$ bits, where **n** is the total number of training instances.

Minimum Description Length (MDL) - Example

- Consider the following decision trees. Assume they are generated from a data set that contains 16 binary attributes and 3 classes, C_1, C_2, and C_3.



a) Decision tree with 7 errors



b) Decision tree with 4 errors

- Because there are 16 attributes, the cost for each internal node in the decision tree is: $\mathbf{\log_2(m) = \log_2(16) = 4}$
- Furthermore, because there are 3 classes, the cost for each leaf node is: $\mathbf{\log_2(k) = \lceil \log_2(3) \rceil = 2}$
- The cost for each misclassification error is $\mathbf{\log_2(n)}$.
- The overall cost for the decision tree (a) is $\mathbf{2\times4+3\times2+7\times\log_2 n = 14+7\ \log_2 n}$
  and the overall cost for the decision tree (b) is $\mathbf{4\times4+5\times2+4\times5 = 26+4\ \log_2 n.}$
- According to the MDL principle, tree (a) is better than (b) if n < 16 and is worse than (b) if n > 16.

How to Address Overfitting

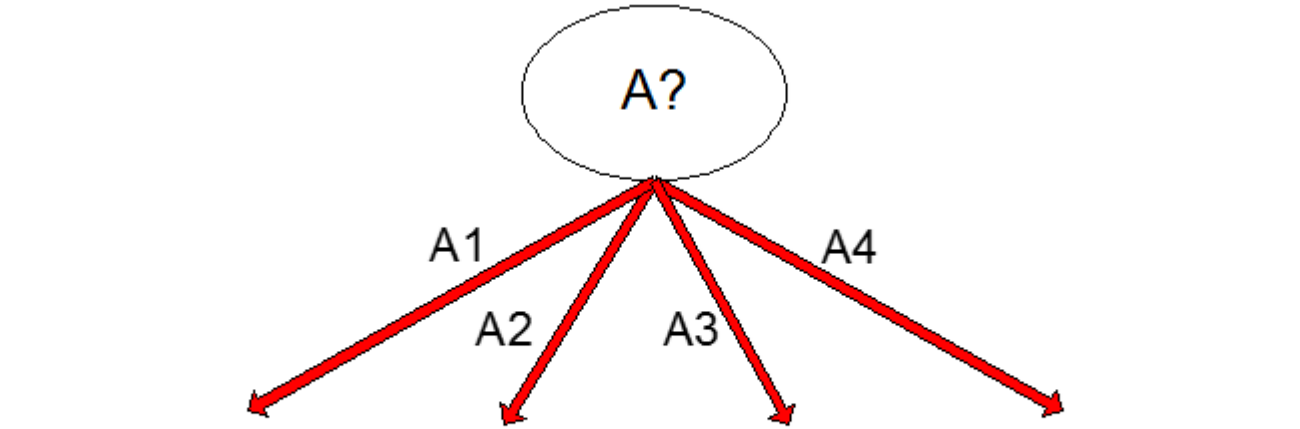
- **Pre-Pruning (Early Stopping Rule)**
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

How to Address Overfitting

- **Post-pruning**
  - Grow decision tree to its entirety
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree
  - Can use MDL for post-pruning

Post-Pruning to avoid overfitting

Class = Yes	20
Class = No	10
Error = 10/30	



Training Error (Before splitting) = 10/30

Pessimistic error = (10 + 0.5)/30 = 10.5/30

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

= (9 + 4 × 0.5)/30 = 11/30

PRUNE!

Classification in Large Databases

- ***Classification*** - a classical problem extensively studied by statisticians and machine learning researchers
- ***Scalability***: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
  - The efficiency of classical decision tree algorithms, such as ID3, C4.5, and CART, has been well established for relatively small data sets.
  - Efficiency becomes an issue of concern when these algorithms are applied to the mining of very large real-world databases.
  - New decision tree algorithms (such as Rainforest, SLIQ and SPRINT) that address scalability issue
- Why is decision tree induction popular?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods

Decision Tree Based Classification

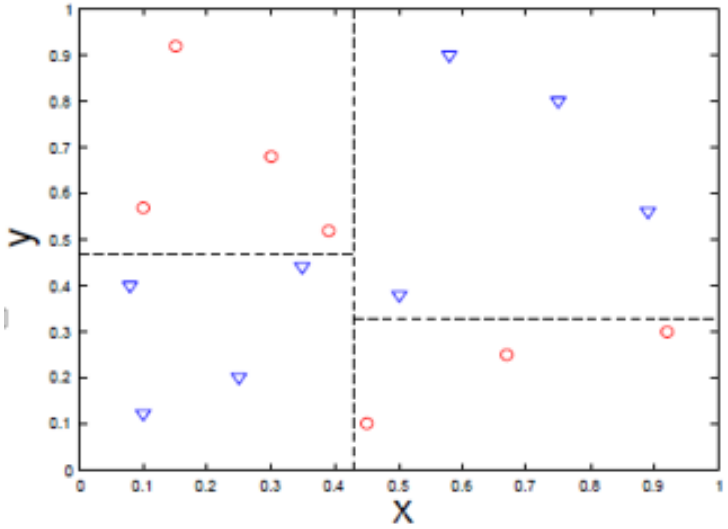
- **Advantages:**
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Robust to noise (especially when methods to avoid overfitting are employed)
  - Can easily handle redundant or irrelevant attributes (unless the attributes are interacting)
- **Disadvantages:**
  - Space of possible decision trees is exponentially large. Greedy approaches are often unable to find the best tree.
  - Does not take into account interactions between attributes
  - Each decision boundary involves only a single attribute

Expressiveness of Decision Tree Representation

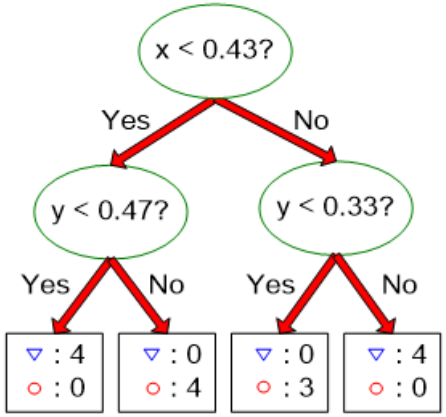
- Test conditions involve using only a ***single attribute at a time***.
  - Tree-growing procedure can be viewed as the process of partitioning the attribute space into disjoint regions until each region contains records of the same class.
  - The border between two neighboring regions of different classes is a **decision boundary**.
  - Since the test condition involves only a single attribute, the ***decision boundaries are rectilinear.***

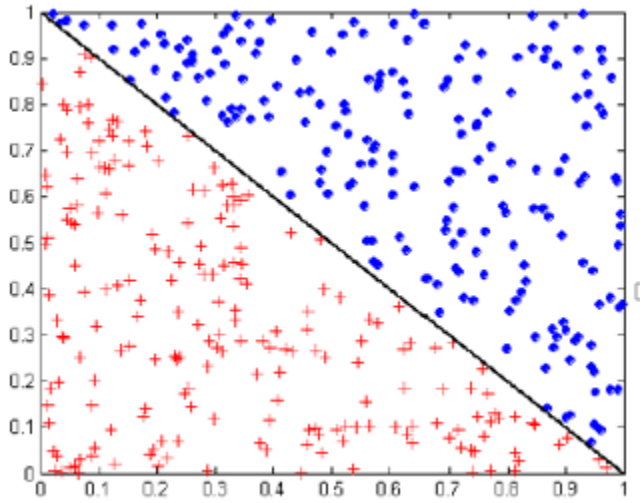
- This limits the expressiveness of the decision tree representation for modeling complex relationships among continuous attributes.

Expressiveness of Decision Tree Representation



- Decision tree induction can handle this kind of data set.





- Test condition may involve multiple attributes
- Decision tree induction cannot handle this kind of data set.