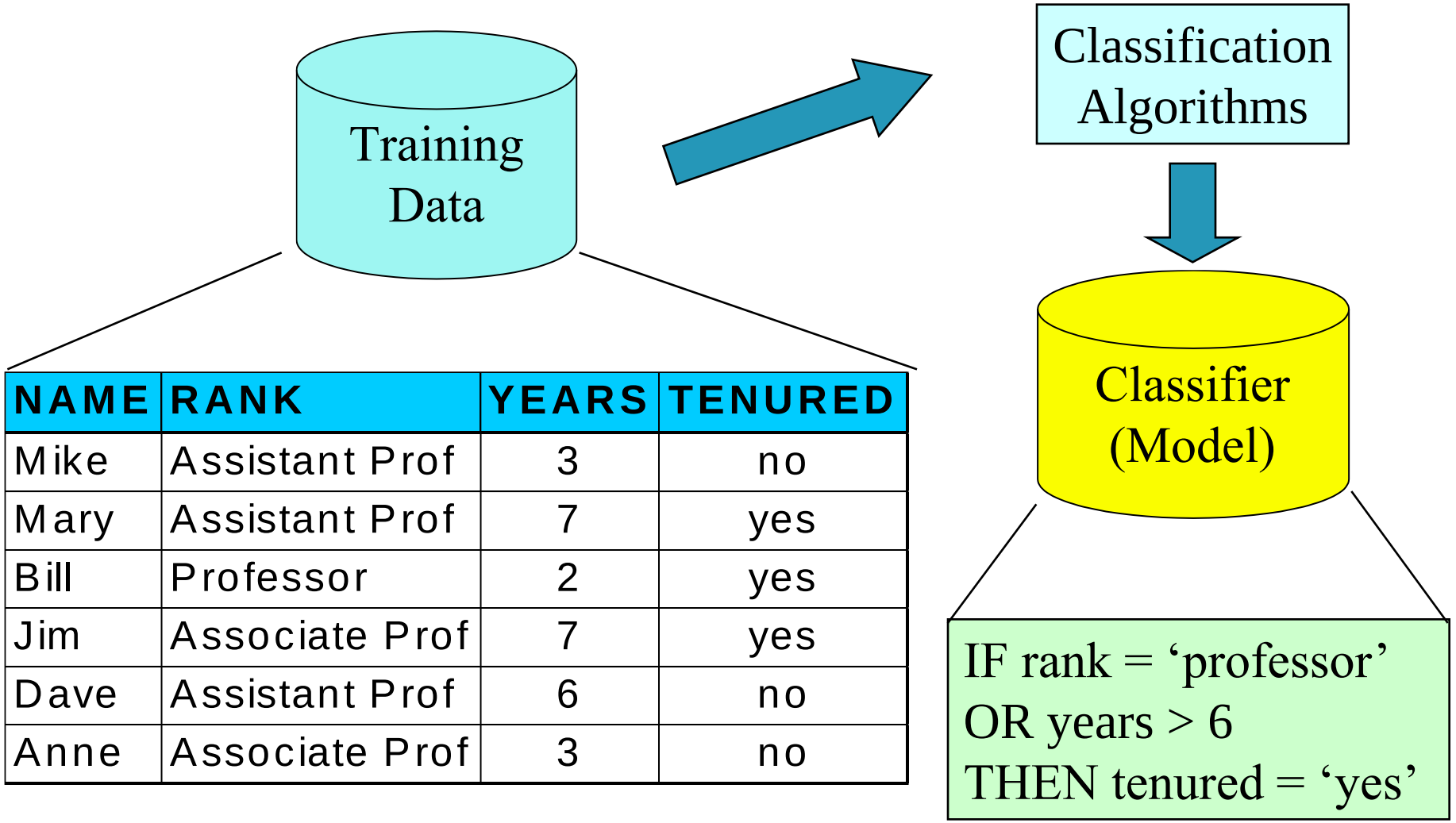
# Supervised Learning

# Classification—A Two-Step Process

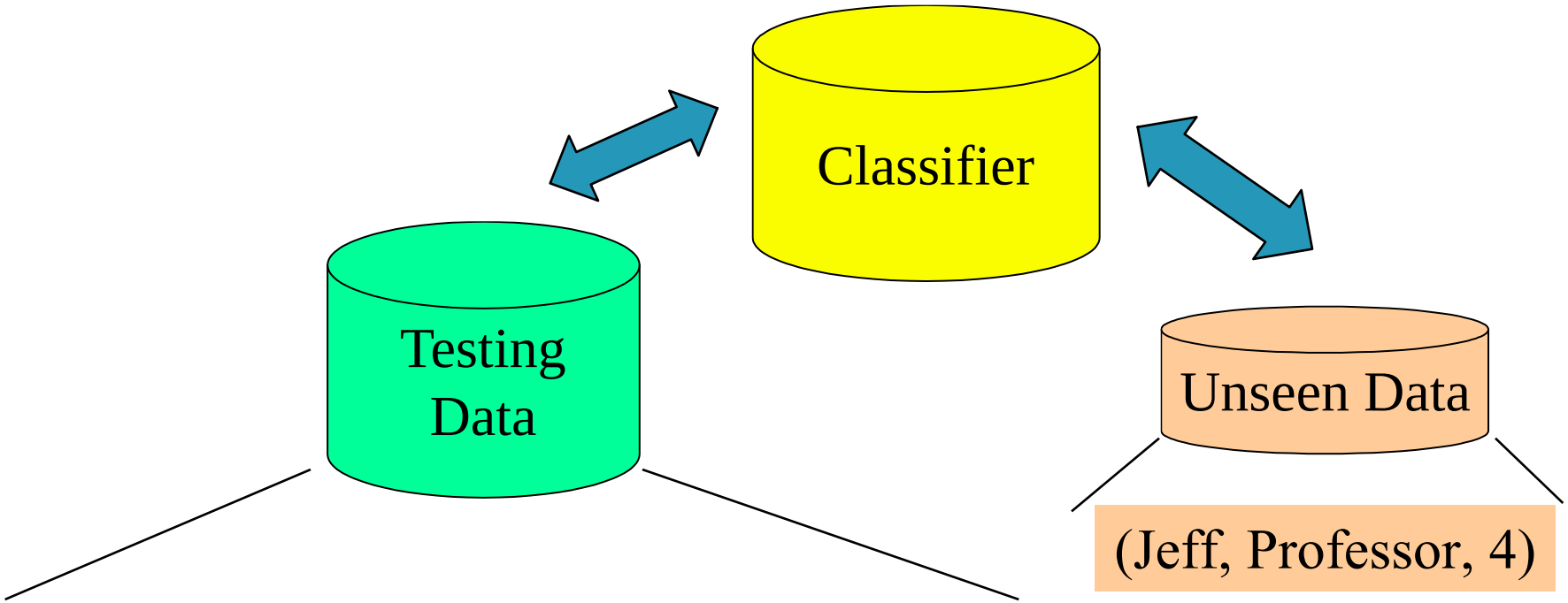
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision 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
- Note: If *the test set* is used to select models, it is called validation (test) set

# Process (1): Model Construction




| NAME | RANK | YEARS | TENURED |
|---|---|---|---|
| Mike | Assistant Prof | 3 | no |
| Mary | Assistant Prof | 7 | yes |
| Bill | Professor | 2 | yes |
| Jim | Associate Prof | 7 | yes |
| Dave | Assistant Prof | 6 | no |
| Anne | Associate Prof | 3 | no |

# Process (2): Using the Model in Prediction




| NAME | RANK | YEARS | TENURED |
|---|---|---|---|
| Tom | Assistant Prof | 2 | no |
| Merlisa | Associate Prof | 7 | no |
| George | Professor | 5 | yes |
| Joseph | Assistant Prof | 7 | yes |

# Decision Tree

- A decision tree is a tree-like structure where each internal node tests on attribute, each branch corresponds to attribute value and each leaf node represents the final decision or prediction.
- The decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.

**How is Decision Tree formed?**

- The process of forming a decision tree involves recursively partitioning the data based on the values of different attributes.
- The algorithm selects the best attribute to split the data at each internal node, based on certain criteria such as information gain or Gini impurity.
- This splitting process continues until a stopping criterion is met, such as reaching a maximum depth or having a minimum number of instances in a leaf node.

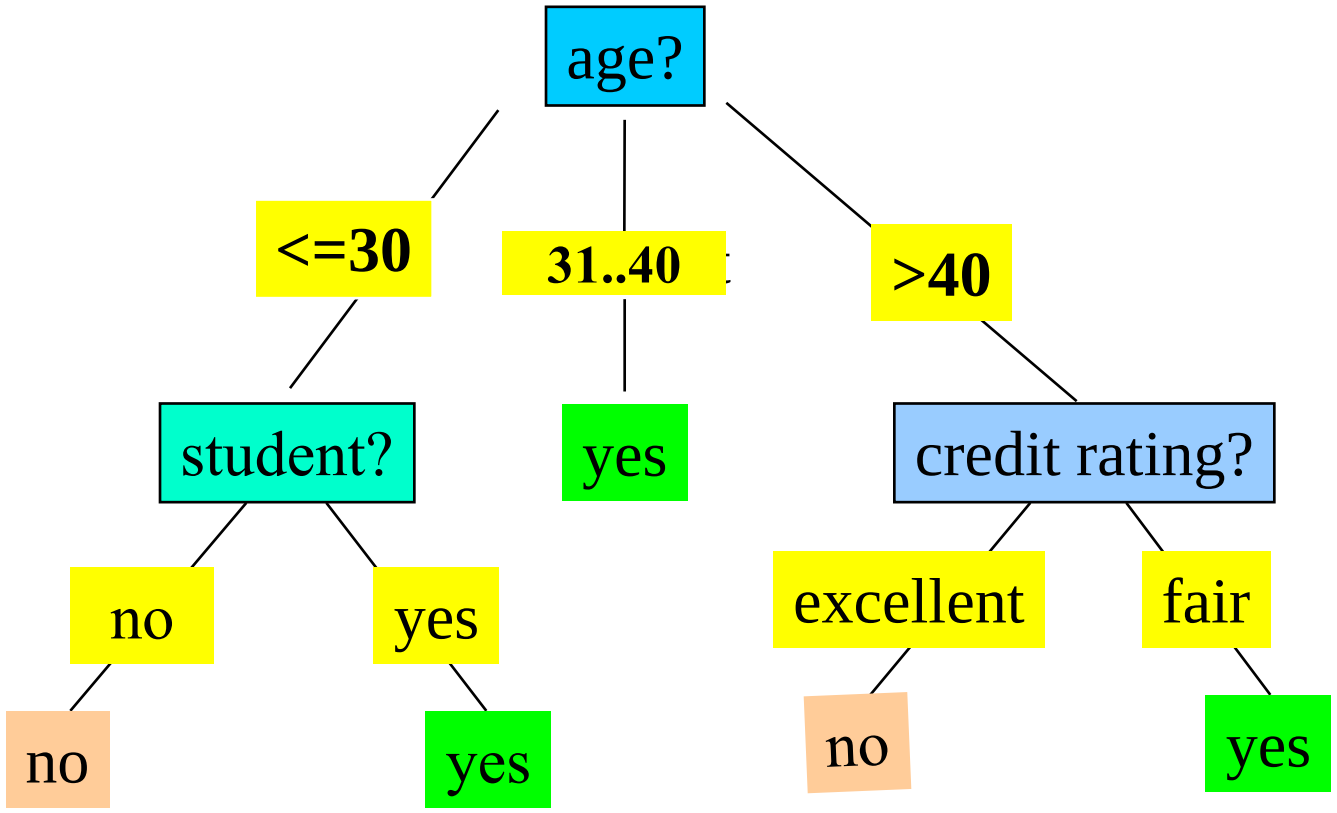
# Decision Tree

There are specialized terms associated with decision trees that denote various components and facets of the tree structure and decision-making procedure. :

- **Root Node:** A decision tree's root node, which represents the original choice or feature from which the tree branches, is the highest node.
- **Internal Nodes (Decision Nodes):** Nodes in the tree whose choices are determined by the values of particular attributes. There are branches on these nodes that go to other nodes.
- **Leaf Nodes (Terminal Nodes):** The branches' termini, when choices or forecasts are decided upon. There are no more branches on leaf nodes.
- **Branches (Edges):** Links between nodes that show how decisions are made in response to particular circumstances.
- **Splitting:** The process of dividing a node into two or more sub-nodes based on a decision criterion. It involves selecting a feature and a threshold to create subsets of data.
- **Decision Criterion:** The rule or condition used to determine how the data should be split at a decision node. It involves comparing feature values against a threshold.
- **Pruning:** The process of removing branches or nodes from a decision tree to improve its generalization and prevent overfitting.

# Decision Tree Induction: An Example

- Training data set: Buys_computer
- The data set follows an example of Quinlan's ID3 (Playing Tennis)
- Resulting tree:




| 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 |

# Algorithm for Decision Tree Induction

- 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 based on a heuristic or statistical measure (e.g., information gain, GINI Index)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

# Attribute Selection Measures

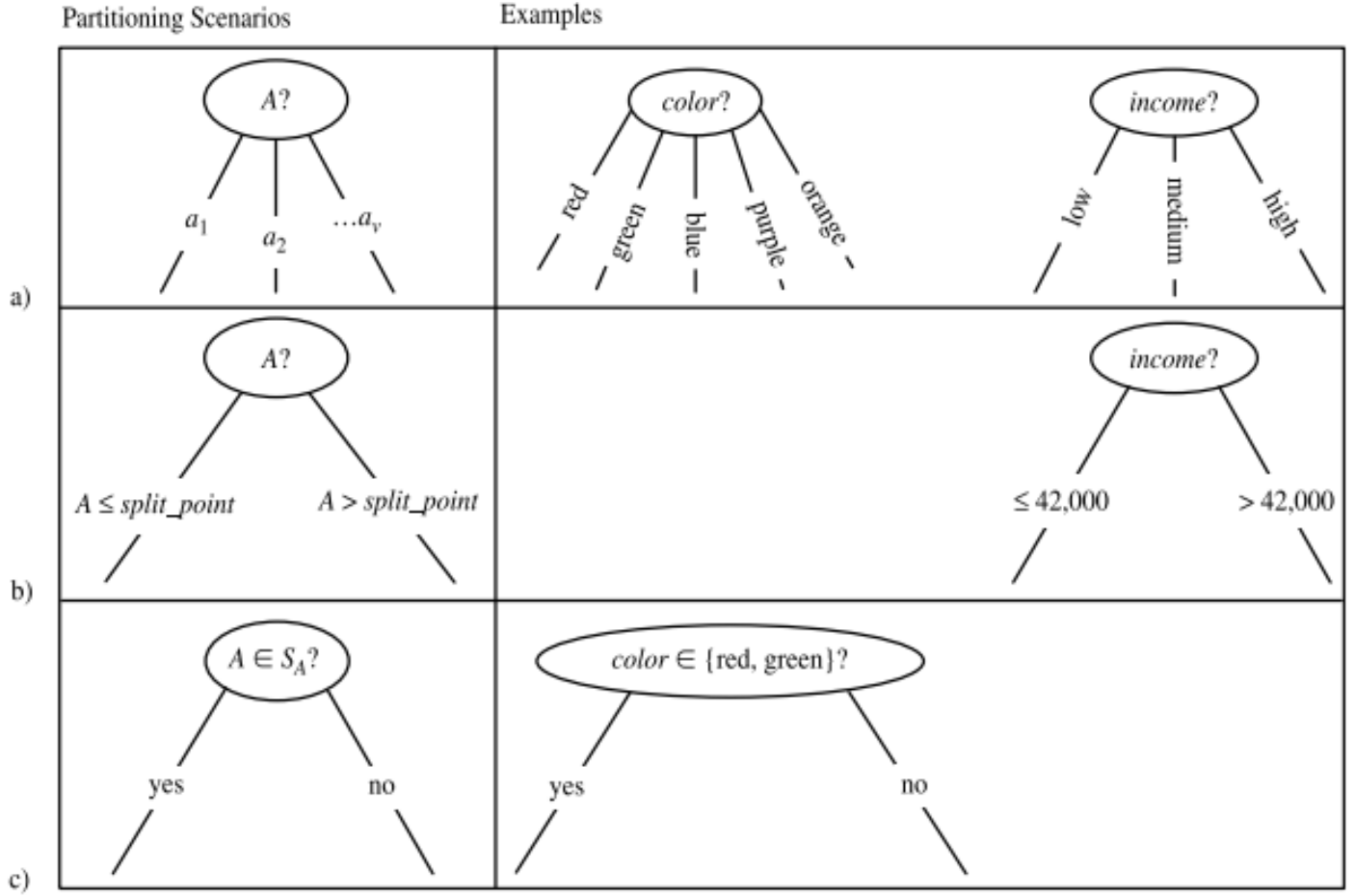
An attribute selection measure is a heuristic for selecting the splitting criterion that "best" separates a given data partition, D, of class-labeled training tuples into individual classes.

Three popular attribute selection measures are :

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

Let,

- D be the data partition, be a training set of class-labeled tuples.
- Suppose the class label attribute has m distinct values defining m distinct classes, $C_i$ (for i = 1,..., m).
- Let $C_{i,D}$ be the set of tuples of class $C_i$ in D.
- Let |D| and | $C_{i,D}$ | denote the number of tuples in D and $C_{i,D}$, respectively.




Three possibilities for partitioning tuples based on the splitting criterion, shown with examples. Let $A$ be the splitting attribute. (a) If $A$ is discrete-valued, then one branch is grown for each known value of $A$. (b) If $A$ is continuous-valued, then two branches are grown, corresponding to $A \leq split\_point$ and $A > split\_point$. (c) If $A$ is discrete-valued and a binary tree must be produced, then the test is of the form $A \in S_A$, where $S_A$ is the splitting subset for $A$.

# Attribute Selection Measures

**1. Information Gain:**
ID3 uses information gain as its attribute selection measure. When we use a node in a decision tree to partition the training instances into smaller subsets the information/entropy changes.
Information gain is a measure of this change in entropy.

- Expected information/entropy needed to classify a tuple in D is given by

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

Where, where $p_i$ is the probability that an arbitrary tuple in D belongs to class $C_i$ and is estimated by $|C_i,D|/|D|$.

- Now, suppose the tuples in D are partitioned on some attribute A having *v* distinct values, $\{a_1, a_2,..., a_v\}$.

D is thus split into *v* partitions $\{D_1, D_2,..., D_v\}$, where $D_j$ contains those tuples in D that have outcome $a_j$ of A.

- We then calculate How much more information would we still need (after the partitioning) in order to arrive at an exact classification? This amount is measured by

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

- Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on A). That is,
- The attribute A with the highest information gain, (Gain(A)), is chosen as the splitting attribute at node N.

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

# Attribute Selection Measures

Class-labeled training tuples from the *AllElectronics* customer database.

| 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 |

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

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits}.$$

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

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

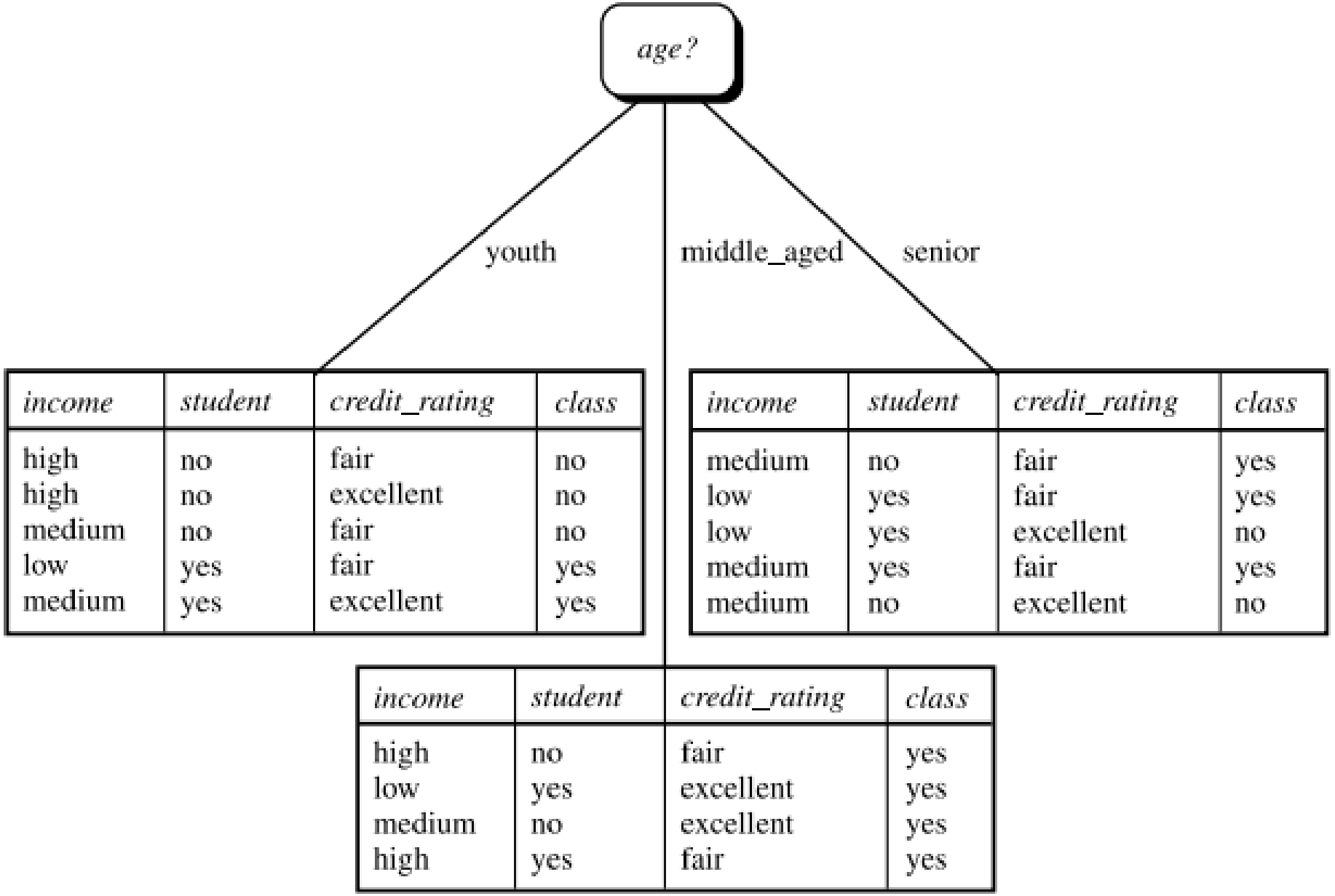
$$Gain(A) = Info(D) - Info_A(D).$$

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

Gain(income) = 0.029 bits, Gain(student) = 0.151 bits,
Gain(credit rating) = 0.048 bits.

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

# Attribute Selection Measures




| *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 |

# Attribute Selection Measures

**2. Gain Ratio:** The information gain measure is biased toward tests with many outcomes. That is, it prefers to select attributes having a large number of values.
C4.5, a successor of ID3, uses an extension to information gain known as gain ratio, which attempts to overcome this bias. It applies a kind of normalization to information gain using a "split information" value defined analogously with Info(D) as

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

The gain ratio is defined as

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

# Attribute Selection Measures

**3. Gini Index**

- Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.
- It means an attribute with a lower Gini index should be preferred.
- The Gini index is used in CART algorithm.
- The Gini value or D is calculated by:

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

Where, where $p_i$ is the probability that an arbitrary tuple in D belongs to class $C_i$ and is estimated by $|C_i,D|/|D|$.

- The Gini index considers a binary split for each 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.
- If A has v possible values, then there are $2^v$ possible subsets.
    - For example, if income has three possible values, namely {low, medium, high}, then the possible subsets are {low, medium, high}, {low, medium}, {low, high}, {medium, high}, {low}, {medium}, {high}, and {}.
    - We exclude the power set, {low, medium, high}, and the empty set from consideration since, conceptually, they do not represent a split.
    - Therefore, there are $2^v - 2$ possible ways to form two partitions of the data, D, based on a binary split on A.

# Attribute Selection Measures

**3. Gini Index**

The Gini value or D is calculated by:

Where, where $p_i$ is the probability that an arbitrary tuple in D belongs to class $C_i$ and is estimated by $|C_i,D|/|D|$.

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

- The Gini index considers a binary split for each 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.
- When considering a binary split, we compute a weighted sum of the impurity of each resulting partition. For example, if a binary split on A partitions D into D1 and D2, the gini index of D given that partitioning is

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

The reduction in impurity that would be incurred by a binary split on a discrete- or continuous-valued 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.

# Attribute Selection Measures

## Example of Gini index

Let D be the training data that has 9 tuples belonging to the class buys_computer = yes and the remaining 5 tuples belong to the class buys_computer = no.

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

Let's start with the attribute income, and its subset {low, medium}. The Gini index value computed based on this partitioning is

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

Similarly, the Gini index values for splits on the remaining subsets are: 0.315 (for the subsets {low, high} and {medium}) and 0.300 (for the subsets {medium, high} and {low}). Therefore, the best binary split for attribute income is on {medium, high} (or {low}) because it minimizes the gini index.

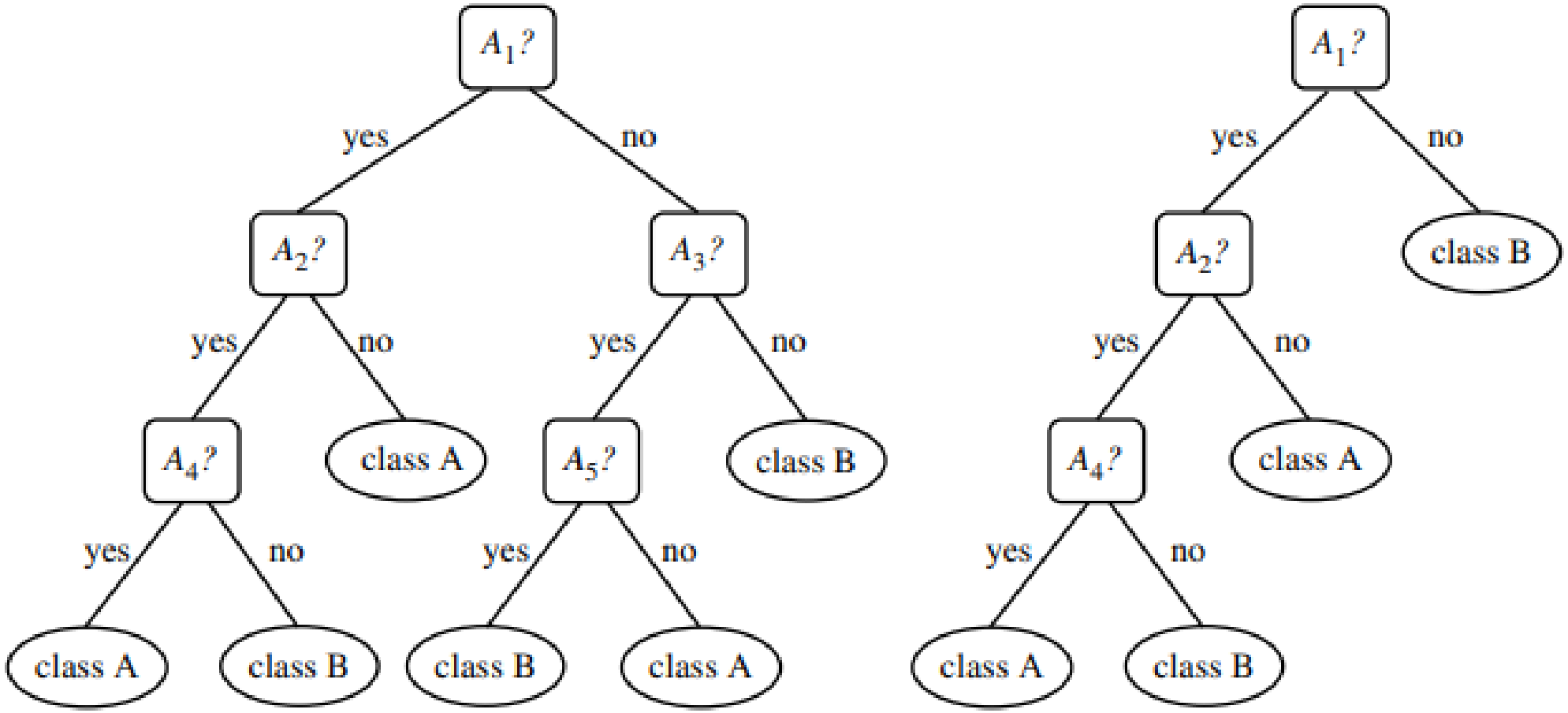
Evaluating the attribute, we obtain {youth, senior} (or {middle aged}) as the best split for age with a Gini index of 0.375; the attributes {student} and {credit rating} are both binary, with Gini index values of 0.367 and 0.429, respectively.

# Overfitting and 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
- 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
    - Difficult to choose an appropriate threshold
  - Postpruning: *Remove branches* from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”

# Overfitting and Tree Pruning

- Example of pruning



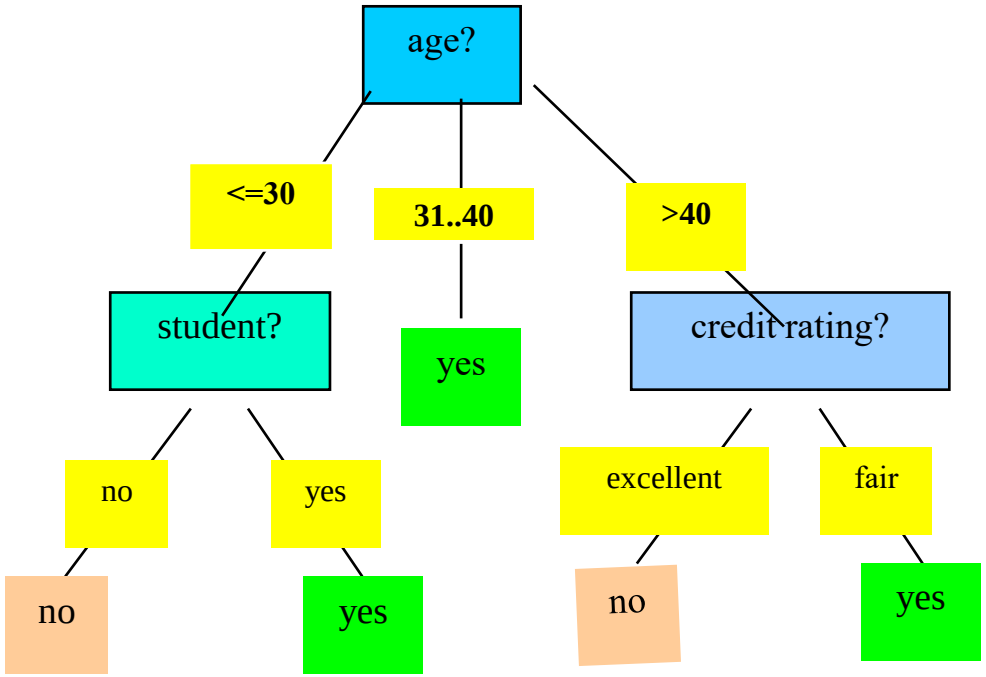
# Practice question

decision-trees-for-classification

| Day | Outlook | Temperature | Humidity | Wind | Play Golf |
|---|---|---|---|---|---|
| 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 |

# Rule Extraction from a Decision Tree

- Rules are *easier to understand* than large trees
- One rule is created *for each path* from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive




- Example: Rule extraction from our *buys_computer* decision-tree

IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*
IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*
IF *age* = mid-age THEN *buys_computer* = *yes*
IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *no*
IF *age* = old AND *credit_rating* = *fair* THEN *buys_computer* = *yes*