

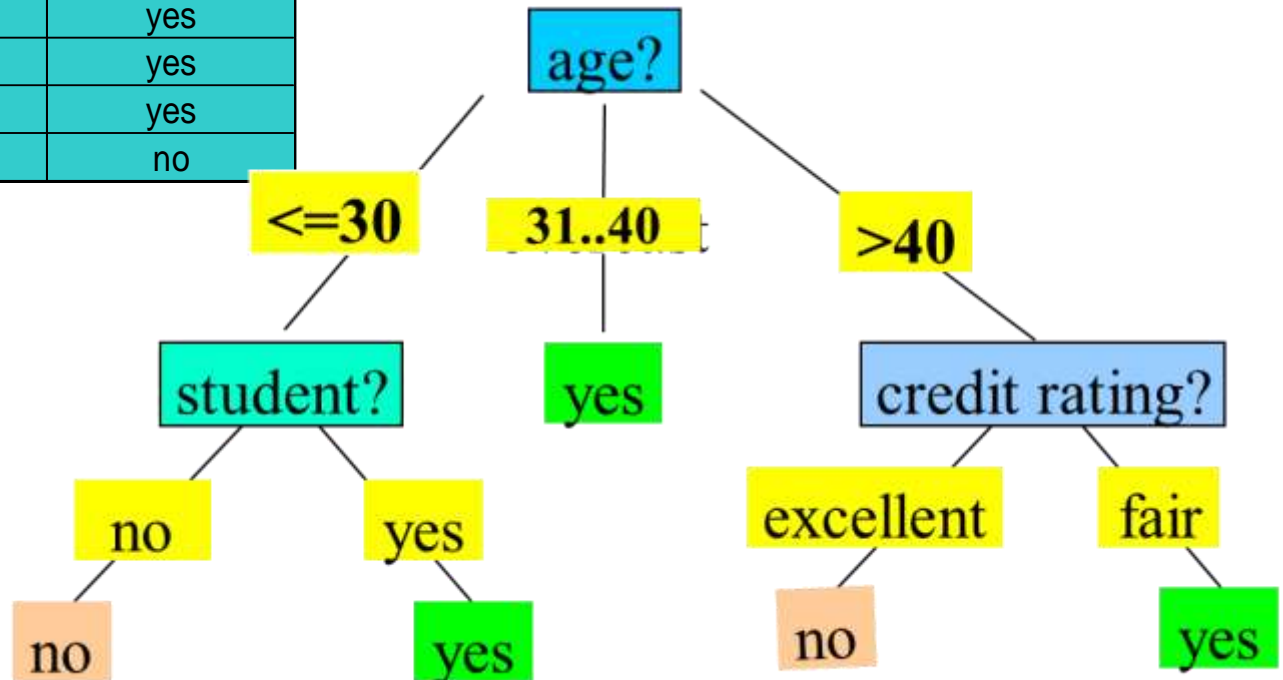
# Classification : Decision Tree

**Decision Trees** are a popular supervised machine learning algorithm used for both **classification** and regression tasks. They work by learning simple decision rules inferred from the data features to predict a target label.

- A Decision Tree is structured as a tree where each node represents a decision based on a feature of the data, and each branch represents an outcome of that decision.
- The root node is the topmost node, representing the first decision.
- The internal nodes are decision points, splitting based on feature values.
- Leaf nodes are the endpoints representing the final prediction or output.

# Classification : Decision Tree

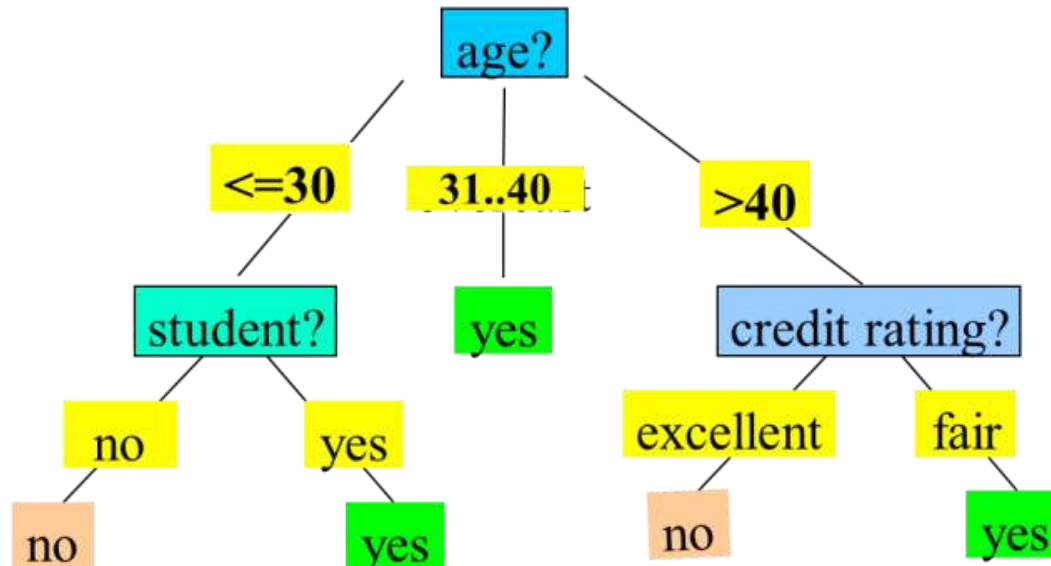
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes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



# Classification : Decision Tree

## 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**)



# Classification : Decision Tree

**ID3 (Iterative Dichotomiser 3)** is a classic algorithm used to construct a Decision Tree, primarily for classification tasks. It was developed by Ross Quinlan and serves as one of the earliest algorithms in Decision Tree-based models.

- ID3 aims to create a Decision Tree by selecting the most **significant feature** at each step that best separates the data for classification.
- It does this by using **information gain** (based on entropy) as the criterion to determine the best feature to split on.

## **Conditions for stopping partitioning:**

- All records have the same target class (pure leaf node).
- No more features to split on (terminate as a leaf node).
- Reaching a specified maximum depth or minimum samples threshold to prevent overfitting.

# Classification : Decision Tree

**Attribute Selection Measure:**  
**Information Gain**

$D$  = set of data instances

$|D|$  = number of data instances (14)

$p_i$  = the probability that an arbitrary tuple in  $D$  belongs to class  $C_i$

$$p_i = \frac{|C_{i,D}|}{|D|}$$

Example: class yes is 1

$$p_1 = \frac{|C_{1,D}|}{|D|} = \frac{9}{14}$$

$$p_2 = \frac{|C_{2,D}|}{|D|} = \frac{5}{14}$$

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

# Classification : Decision Tree

**Expected information (entropy) needed to classify a tuple in D:**

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

For the given data

$$Info(D) = - \sum_{i=1}^2 p_i \log_2 p_i$$

$$= -p_1 \log_2 p_1 - p_2 \log_2 p_2$$

$$= -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940$$

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

# Classification : Decision Tree

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

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

**Example**

$$Info_{age}(D) = \sum_{j=1}^3 \frac{|D_j|}{|D|} \times Info(D_j)$$

$$= \frac{5}{14} \times I(2,3) + \frac{4}{14} \times I(4,0) + \frac{5}{14} \times I(2,3) = 0.694$$

$\frac{5}{14} \times I(2,3)$  means “age  $\leq 30$ ” has 5 out of 14 samples, with 2 yes’es and 3 no’s.

age	income	student	credit_rating	buys_computer
$\leq 30$	high	no	fair	no
$\leq 30$	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
$\leq 30$	medium	no	fair	no
$\leq 30$	low	yes	fair	yes
>40	medium	yes	fair	yes
$\leq 30$	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

# Classification : Decision Tree

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

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

**Example**

$$Info_{age}(D) = \sum_{j=1}^3 \frac{|D_j|}{|D|} \times Info(D_j)$$

$$= \frac{5}{14} \times I(2,3) + \frac{4}{14} \times I(4,0) + \frac{5}{14} \times I(2,3) = 0.694$$

$\frac{5}{14} \times I(2,3)$  means “age  $\leq 30$ ” has 5 out of 14 samples, with 2 yes’es and 3 no’s.

age	income	student	credit_rating	buys_computer
$\leq 30$	high	no	fair	no
$\leq 30$	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
$\leq 30$	medium	no	fair	no
$\leq 30$	low	yes	fair	yes
$> 40$	medium	yes	fair	yes
$\leq 30$	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
$> 40$	medium	no	excellent	no



# Classification : Decision Tree

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

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

**Example**

$$Info_{age}(D) = \sum_{j=1}^3 \frac{|D_j|}{|D|} \times Info(D_j)$$

$$= \frac{5}{14} \times I(2,3) + \frac{4}{14} \times I(4,0) + \frac{5}{14} \times I(3,2) = 0.694$$

$\frac{5}{14} \times I(2,3)$  means “age  $\leq 30$ ” has 5 out of 14 samples, with 2 yes’es and 3 no’s.

age	income	student	credit_rating	buys_computer
$\leq 30$	high	no	fair	no
$\leq 30$	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
$\leq 30$	medium	no	fair	no
$\leq 30$	low	yes	fair	yes
>40	medium	yes	fair	yes
$\leq 30$	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

# Classification : Decision Tree

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

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

**Similarly**

$$Info_{income}(D) = \sum_{j=1}^3 \frac{|D_j|}{|D|} \times Info(D_j)$$

$$= \frac{4}{14} \times I(2,2) + \frac{6}{14} \times I(4,2) + \frac{4}{14} \times I(3,1) = 0.911$$

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

# Classification : Decision Tree

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

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

**Similarly**

$$Info_{income}(D) = \sum_{j=1}^3 \frac{|D_j|}{|D|} \times Info(D_j)$$

$$= \frac{4}{14} \times I(2,2) + \frac{6}{14} \times I(4,2) + \frac{4}{14} \times I(3,1) = 0.911$$

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

# Classification : Decision Tree

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

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

**Similarly**

$$Info_{income}(D) = \sum_{j=1}^3 \frac{|D_j|}{|D|} \times Info(D_j)$$

$$= \frac{4}{14} \times I(2,2) + \frac{6}{14} \times I(4,2) + \frac{4}{14} \times I(3,1) = 0.911$$

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

# Classification : Decision Tree

Information gained by branching on attribute A

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

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

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

Similarly,

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

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

# Classification : Decision Tree

Information gained by branching on attribute A

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


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

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

Similarly,

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

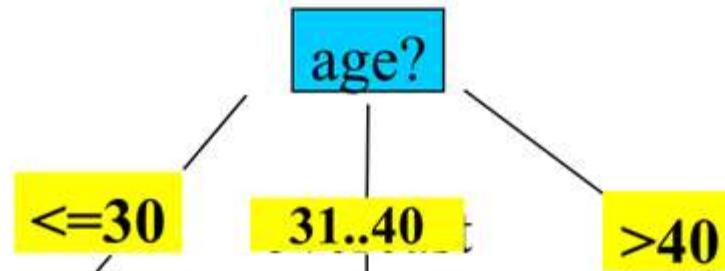


age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes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

# Classification : Decision Tree

So, the first split is on the **age** attribute

Next, repeat the same process with the other nodes with smaller dataset.



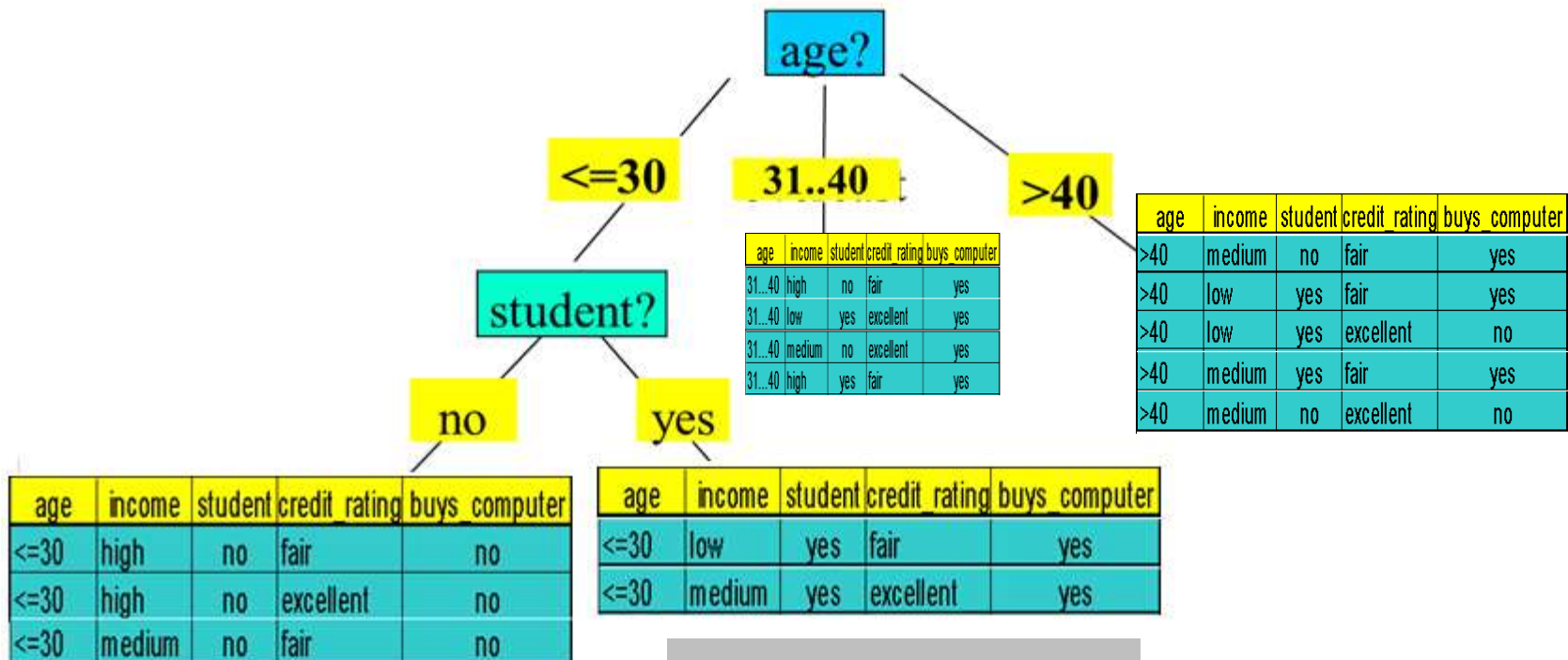
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

age	income	student	credit_rating	buys_computer
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	credit_rating	buys_computer
>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

# Classification : Decision Tree

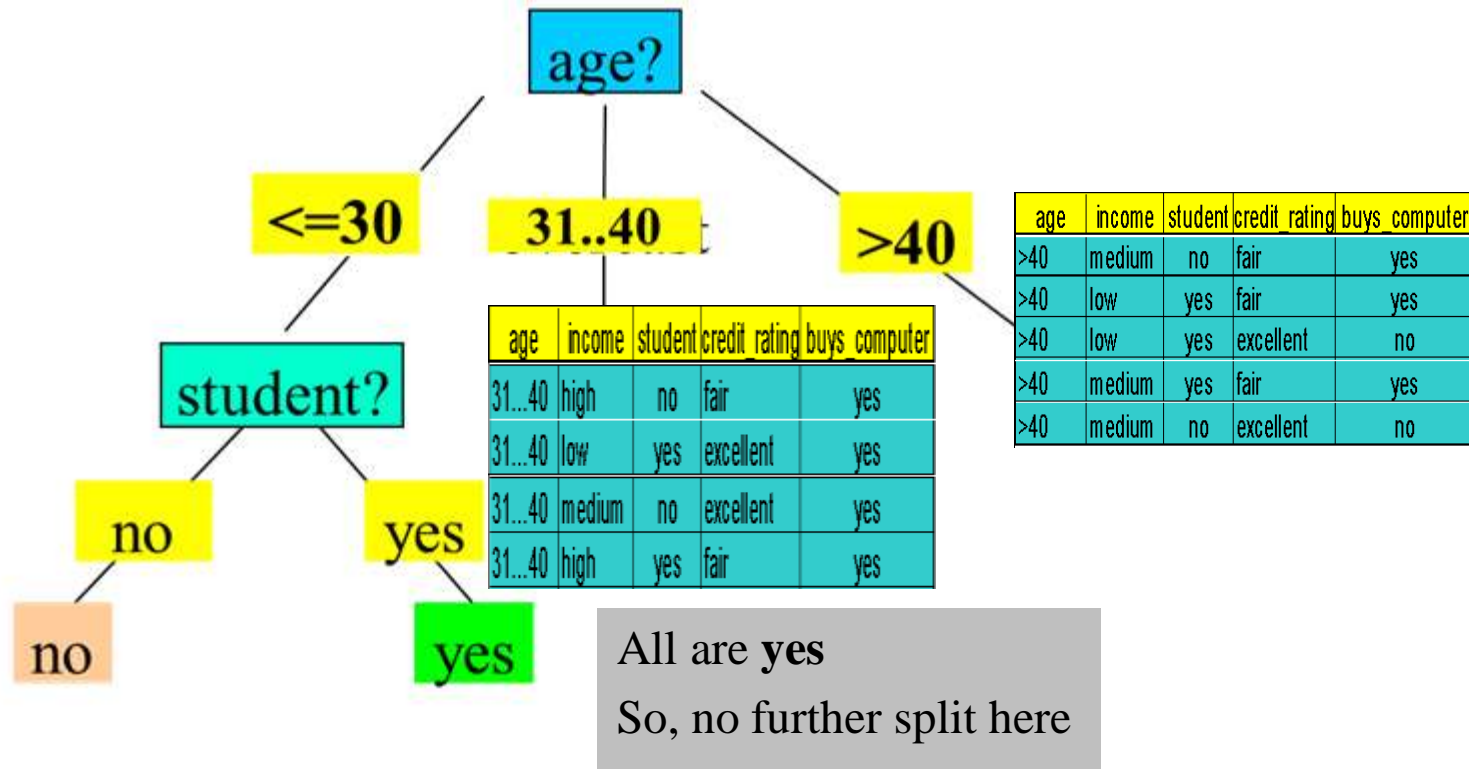
Next split is on **student** at node  $\leq 30$



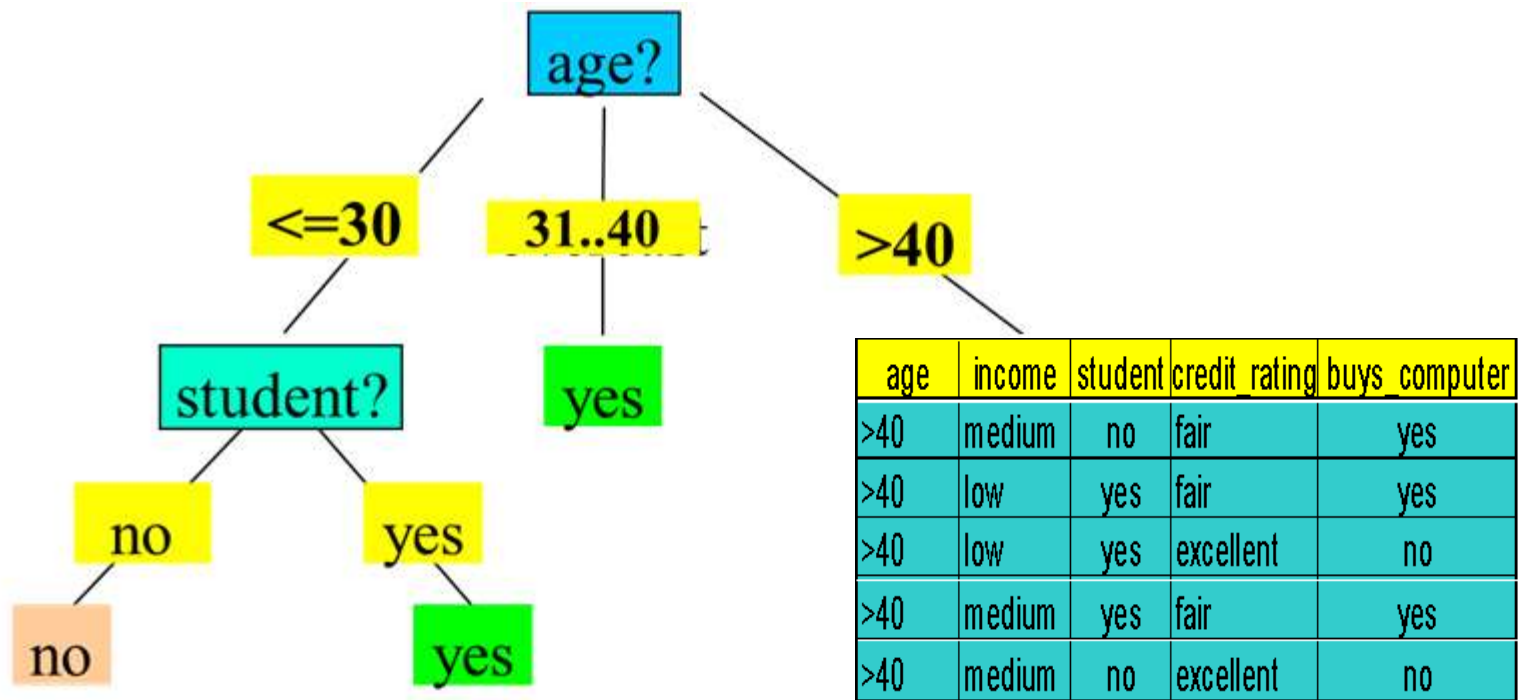


# Classification : Decision Tree

No split is required at node **31..40**

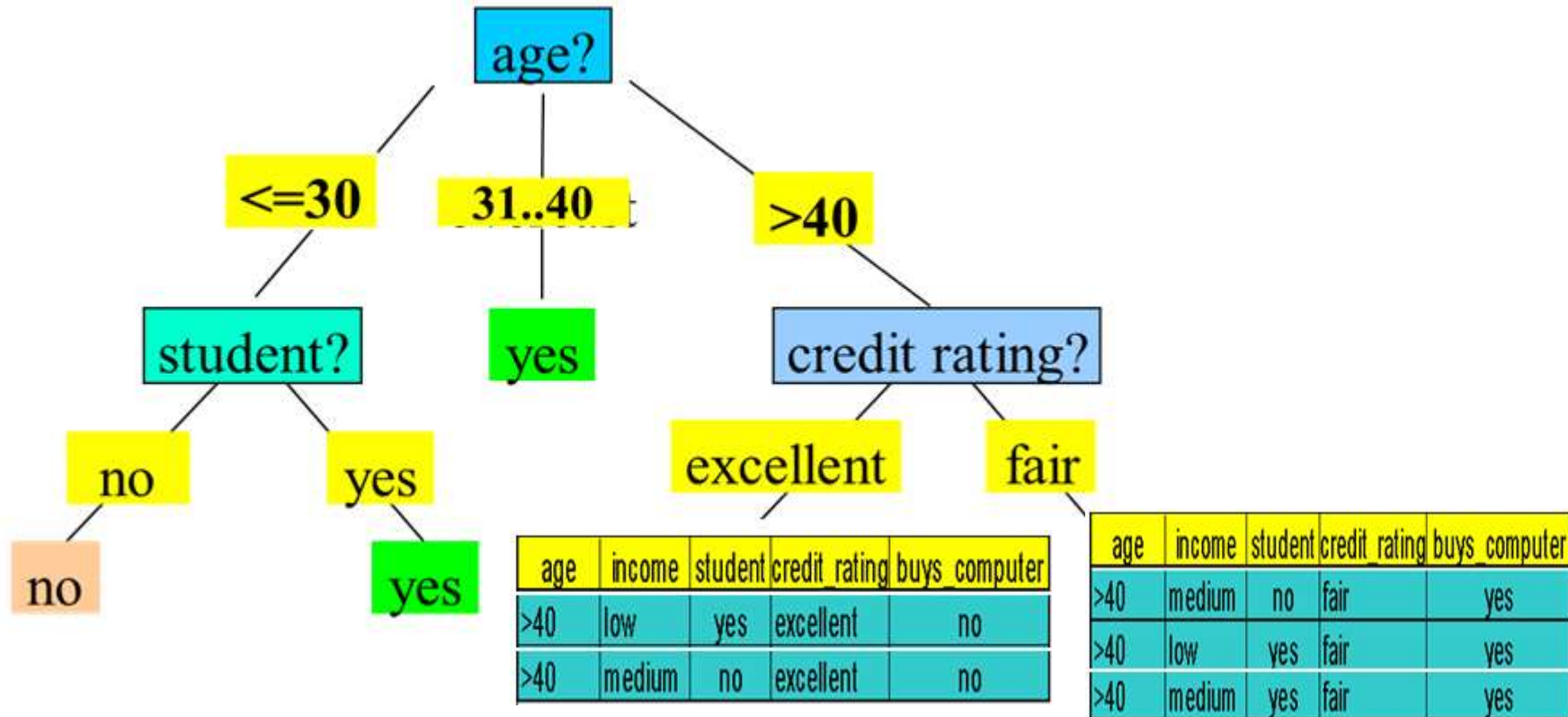


# Classification : Decision Tree

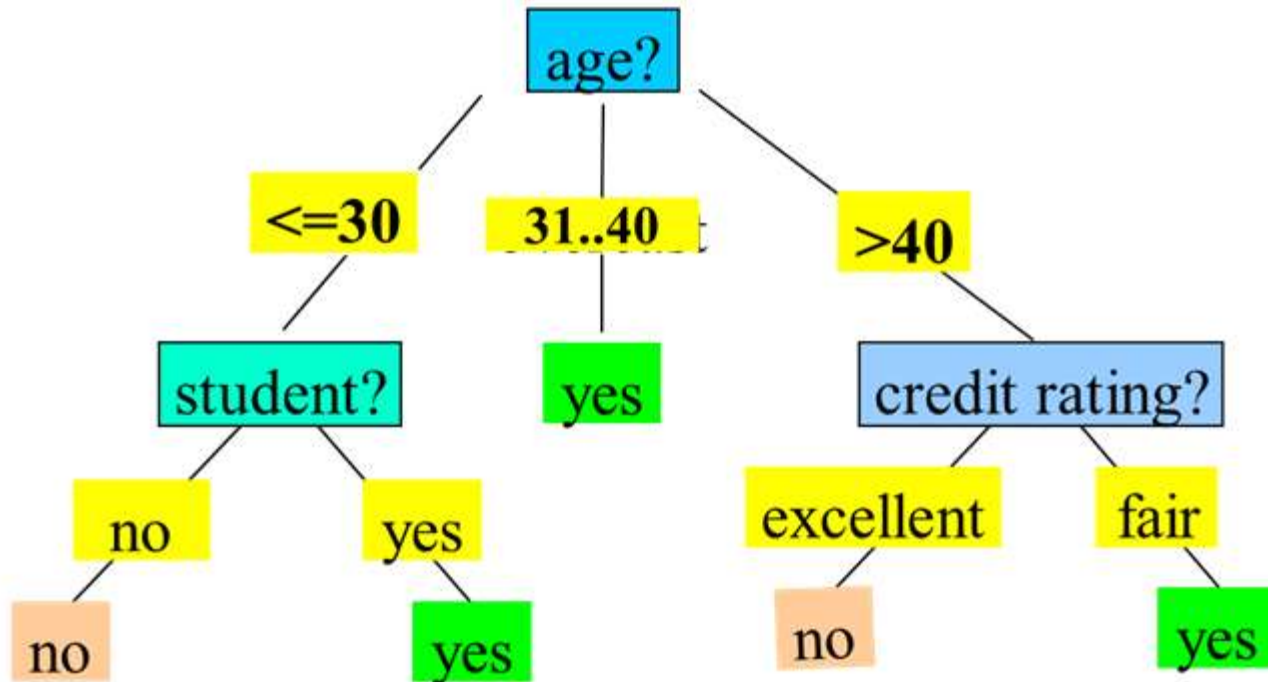


# Classification : Decision Tree

Next split is on **credit\_rating** at node **>40**



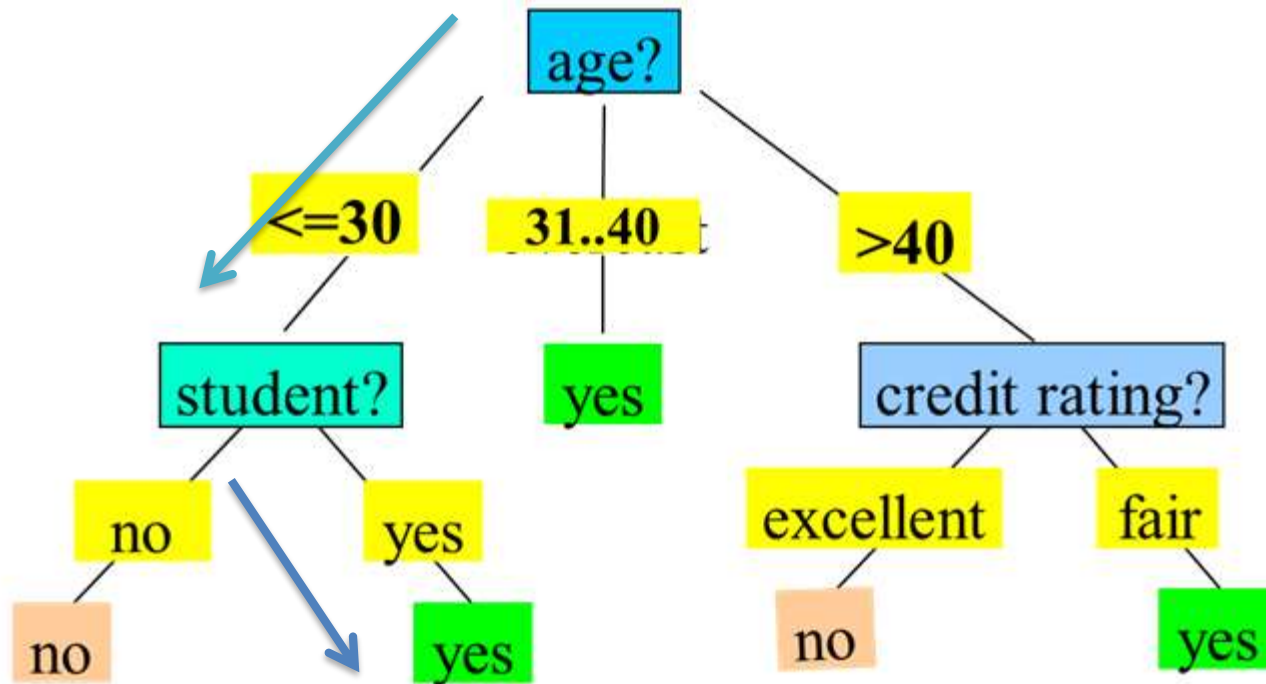
# Classification : Decision Tree



# Classification : Decision Tree

**Classify a new instance**

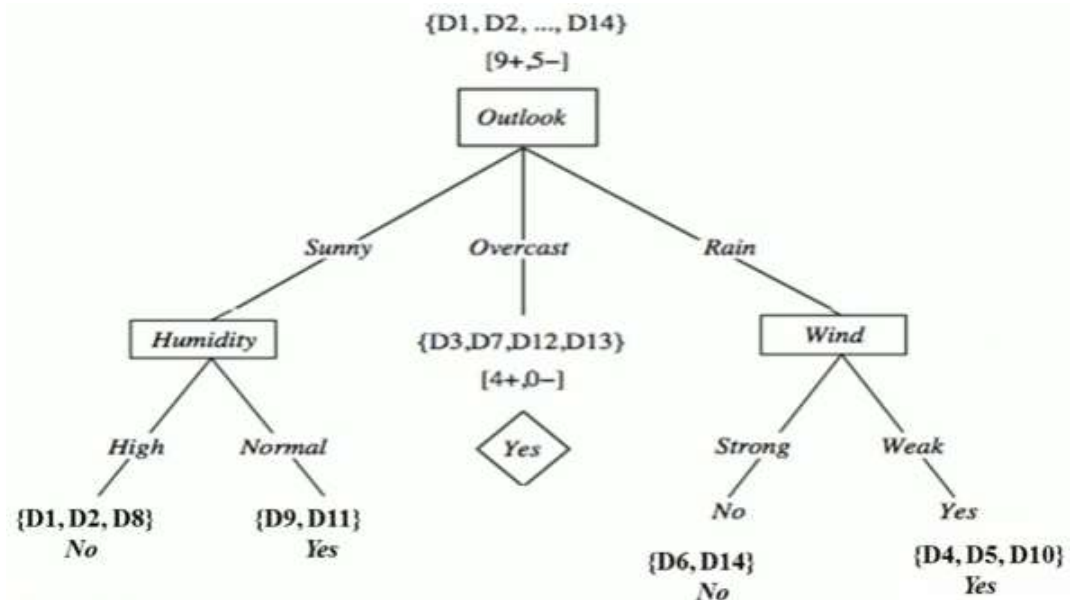
(age:  $\leq 30$ , income: low, student: yes, credit\_rating: fair) = buys\_computer?



# Classification : Decision Tree

## Exercise

Day	Outlook	Temp	Humidity	Wind	Play Tennis
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



# Classification : Decision Tree

[https://colab.research.google.com/drive/1YsHFA55DZ5iQBNq7S11qoS-\\_WVYtxUYH?usp=sharing](https://colab.research.google.com/drive/1YsHFA55DZ5iQBNq7S11qoS-_WVYtxUYH?usp=sharing)

# Classification : Decision Tree

## 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  $A \leq \text{split-point}$ , and **D2** is the set of tuples in **D** satisfying  $A > \text{split-point}$



# Classification : Decision Tree

## Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with many values
- C4.5** (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

- $\text{GainRatio}(A) = \text{Gain}(A) / \text{SplitInfo}(A)$

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

$$\text{SplitInfo}_{\text{income}}(D) = -\frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) - \frac{6}{14} \times \log_2 \left( \frac{6}{14} \right) - \frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) = 1.557$$

- $\text{gain\_ratio}(\text{income}) = 0.029 / 1.557 = 0.019$

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

# Classification : Decision Tree

## Gini Index (CART-Classification and Regression Trees)

**Information Gain** and Gain Ratio can sometimes exhibit a bias toward features with many unique values. For example, if a feature has many unique categories, Information Gain might prefer it because it creates more distinct subsets.

To address this, **Gain Ratio** (an adaptation of Information Gain) is used in algorithms like C4.5 to counter this bias, but this additional step adds complexity.

**Gini Index**, on the other hand, is less sensitive to the number of categories in a feature, reducing the need for an additional adjustment like Gain Ratio.

For a node  $S$  with  $C$  possible classes, the Gini Index is calculated as:

$$\text{Gini}(S) = 1 - \sum_{i=1}^C p_i^2$$

For a split into two nodes,  $S_1$  and  $S_2$ :

$$\text{Gini}_{\text{split}} = \frac{|S_1|}{|S|} \text{Gini}(S_1) + \frac{|S_2|}{|S|} \text{Gini}(S_2)$$

# Classification : Decision Tree

## Overfitting in Decision Trees

- Overfitting happens when a Decision Tree model becomes too complex, fitting too closely to the noise and specific details of the training data, resulting in poor generalization to new data.
- Signs of Overfitting: A decision tree that overfits will often have a **high accuracy on the training set but performs poorly on the test set**.
- Causes of Overfitting:
  - Allowing the tree to grow too deep, with many nodes and branches.
  - Having very specific branches (splits) that capture the peculiarities of the training data rather than general patterns.
- Consequences: Overfitting leads to a lack of model generalization, where the model captures random fluctuations instead of meaningful trends.

# Classification : Decision Tree

## **Pre-Pruning (Early Stopping):**

This approach stops the tree from growing once it meets certain conditions, such as reaching a maximum depth, minimum number of samples per leaf, or minimum information gain threshold. This prevents the tree from developing complex branches that might lead to

## **Post-Pruning (Cost Complexity Pruning):**

This technique first grows the tree to its full depth, then prunes back branches that provide minimal value. Post-pruning can be based on measures like cross-validation to determine the optimal structure that minimizes errors on new data.