# Chapter 3:
# Supervised Learning

# Road Map

- **Basic concepts**
- Decision tree induction
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- K-nearest neighbor
- Ensemble methods: Bagging and Boosting
- Summary

# An example application

- An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.
- A decision is needed: whether to put a new patient in an intensive-care unit.
- Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.
- Problem: to predict high-risk patients and discriminate them from low-risk patients.

# Another application

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,
  - age
  - Marital status
  - annual salary
  - outstanding debts
  - credit rating
  - etc.
- Problem: to decide whether an application should approved, or to classify applications into two categories, approved and not approved.

# Machine learning and our focus

- Like human learning from past experiences.
- A computer does not have "experiences".
- A computer system learns from data, which represent some "past experiences" of an application domain.
- Our focus: learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- The task is commonly called: Supervised learning, classification, or inductive learning.

# The data and the goal

- Data: A set of data records (also called examples, instances or cases) described by
  - *k* attributes: $A_1, A_2, \ldots A_k$.
  - a class: Each example is labelled with a pre-defined class.
- Goal: To learn a classification model from the data that can be used to predict the classes of new (future, or test) cases/instances.

# An example: data (loan application)

Approved or not

| ID | Age | Has_Job | Own_House | Credit_Rating | Class |
|---|---|---|---|---|---|
| 1 | young | false | false | fair | **No** |
| 2 | young | false | false | good | **No** |
| 3 | young | true | false | good | **Yes** |
| 4 | young | true | true | fair | **Yes** |
| 5 | young | false | false | fair | **No** |
| 6 | middle | false | false | fair | **No** |
| 7 | middle | false | false | good | **No** |
| 8 | middle | true | true | good | **Yes** |
| 9 | middle | false | true | excellent | **Yes** |
| 10 | middle | false | true | excellent | **Yes** |
| 11 | old | false | true | excellent | **Yes** |
| 12 | old | false | true | good | **Yes** |
| 13 | old | true | false | good | **Yes** |
| 14 | old | true | false | excellent | **Yes** |
| 15 | old | false | false | fair | **No** |

# An example: the learning task

- Learn a classification model from the data
- Use the model to classify future loan applications into
  - Yes (approved) and
  - No (not approved)
- What is the class for following case/instance?

| Age | Has_Job | Own_house | Credit-Rating | Class |
|---|---|---|---|---|
| young | false | false | good | ? |

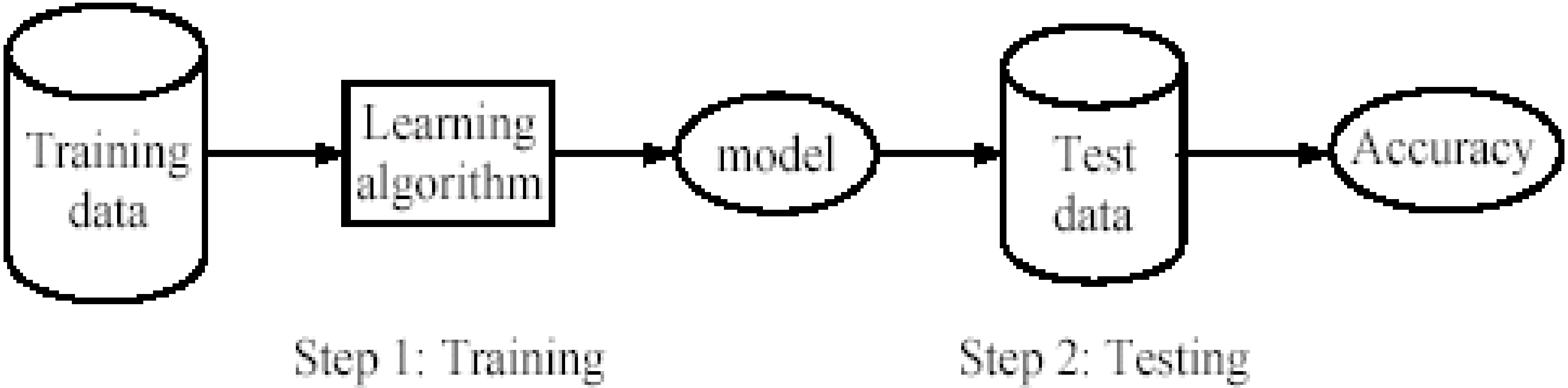
# Supervised vs. unsupervised Learning

- Supervised learning: classification is seen as supervised learning from examples.
  - Supervision: The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a “teacher” gives the classes (supervision).
  - Test data are classified into these classes too.
- Unsupervised learning (clustering)
  - Class labels of the data are unknown
  - Given a set of data, the task is to establish the existence of classes or clusters in the data

# Supervised learning process: two steps

- Learning (training): Learn a model using the training data
- Testing: Test the model using unseen test data to assess the model accuracy

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



# What do we mean by learning?

- Given
  - a data set *D*,
  - a task *T*, and
  - a performance measure *M*,

  a computer system is said to **learn** from *D* to perform the task *T* if after learning the system's performance on *T* improves as measured by *M*.
- In other words, the learned model helps the system to perform *T* better as compared to no learning.

# An example

- Data: Loan application data
- Task: Predict whether a loan should be approved or not.
- Performance measure: accuracy.

No learning: classify all future applications (test data) to the majority class (i.e., Yes):

Accuracy = 9/15 = 60%.

- We can do better than 60% with learning.

# Fundamental assumption of learning

Assumption: The distribution of training examples is identical to the distribution of test examples (including future unseen examples).

- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.

# Road Map

- Basic concepts
- **Decision tree induction**
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- K-nearest neighbor
- Ensemble methods: Bagging and Boosting
- Summary

# Introduction

- Decision tree learning is one of the most widely used techniques for classification.
  - Its classification accuracy is competitive with other methods, and
  - it is very efficient.
- The classification model is a tree, called decision tree.
- C4.5 by Ross Quinlan is perhaps the best known system. It can be downloaded from the Web.

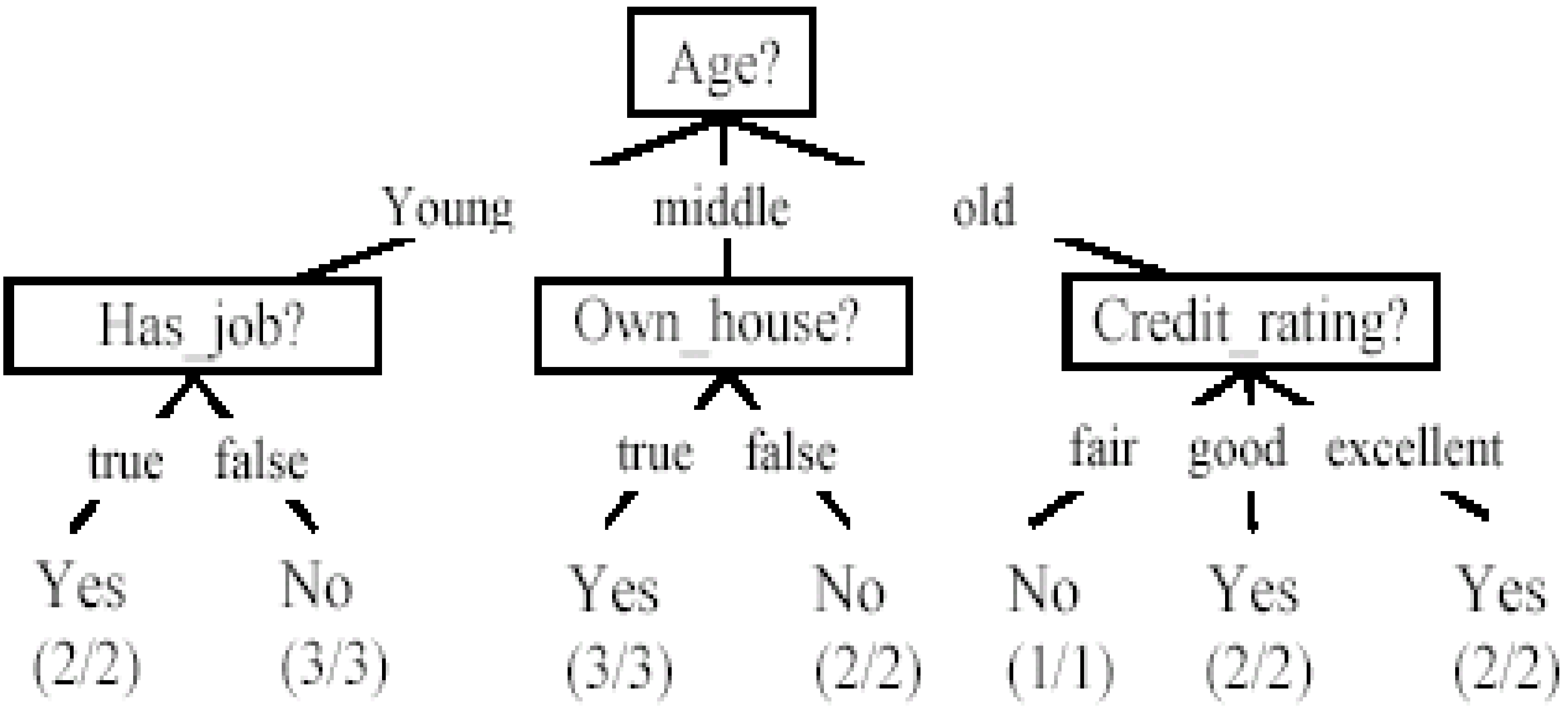
# The loan data (reproduced)

Approved or not

| ID | Age | Has_Job | Own_House | Credit_Rating | Class |
|---|---|---|---|---|---|
| 1 | young | false | false | fair | **No** |
| 2 | young | false | false | good | **No** |
| 3 | young | true | false | good | **Yes** |
| 4 | young | true | true | fair | **Yes** |
| 5 | young | false | false | fair | **No** |
| 6 | middle | false | false | fair | **No** |
| 7 | middle | false | false | good | **No** |
| 8 | middle | true | true | good | **Yes** |
| 9 | middle | false | true | excellent | **Yes** |
| 10 | middle | false | true | excellent | **Yes** |
| 11 | old | false | true | excellent | **Yes** |
| 12 | old | false | true | good | **Yes** |
| 13 | old | true | false | good | **Yes** |
| 14 | old | true | false | excellent | **Yes** |
| 15 | old | false | false | fair | **No** |

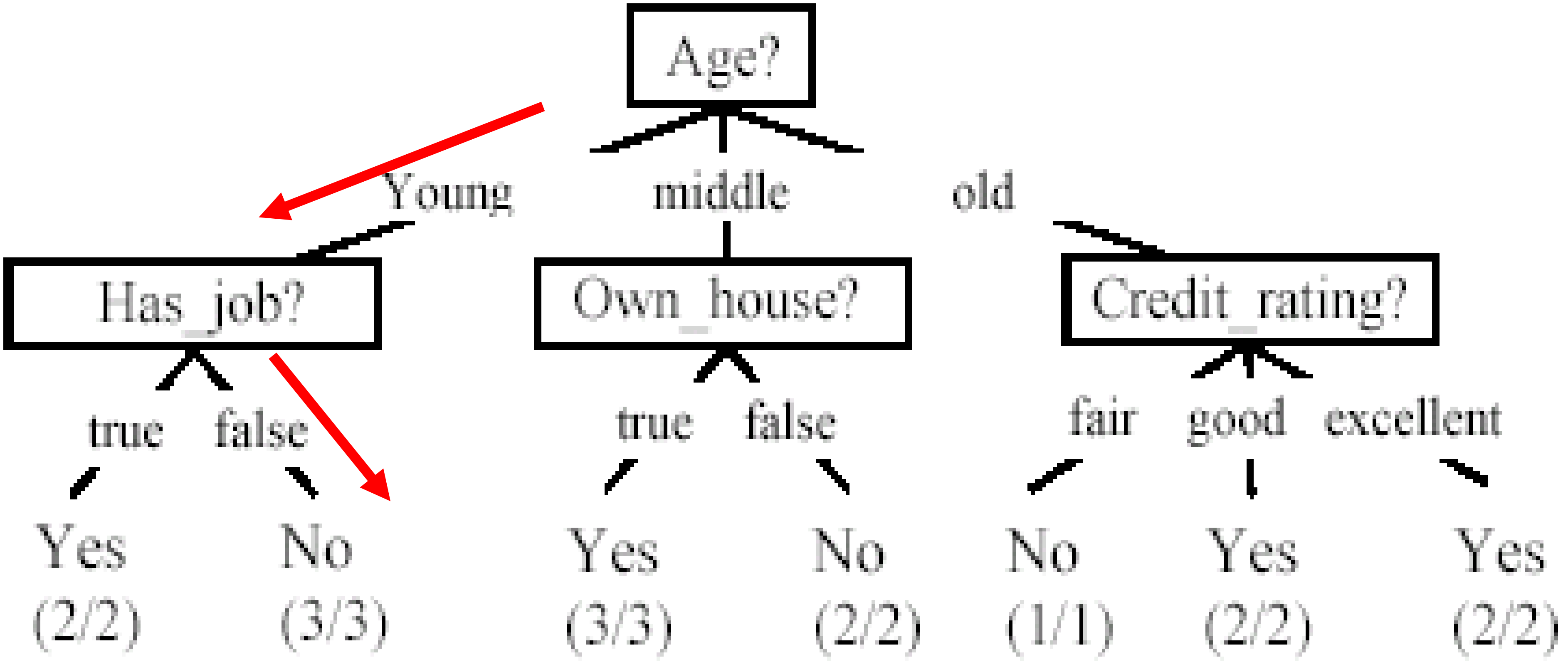
# A decision tree from the loan data

- Decision nodes and leaf nodes (classes)



# Use the decision tree

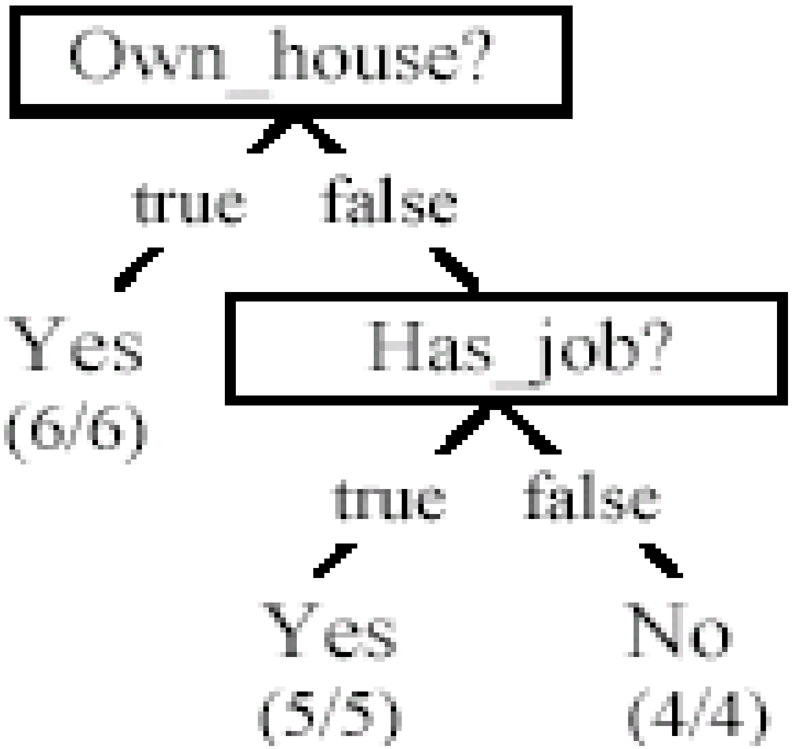
| Age | Has_Job | Own_house | Credit-Rating | Class |
|---|---|---|---|---|
| young | false | false | good | ? No |



# Is the decision tree unique?

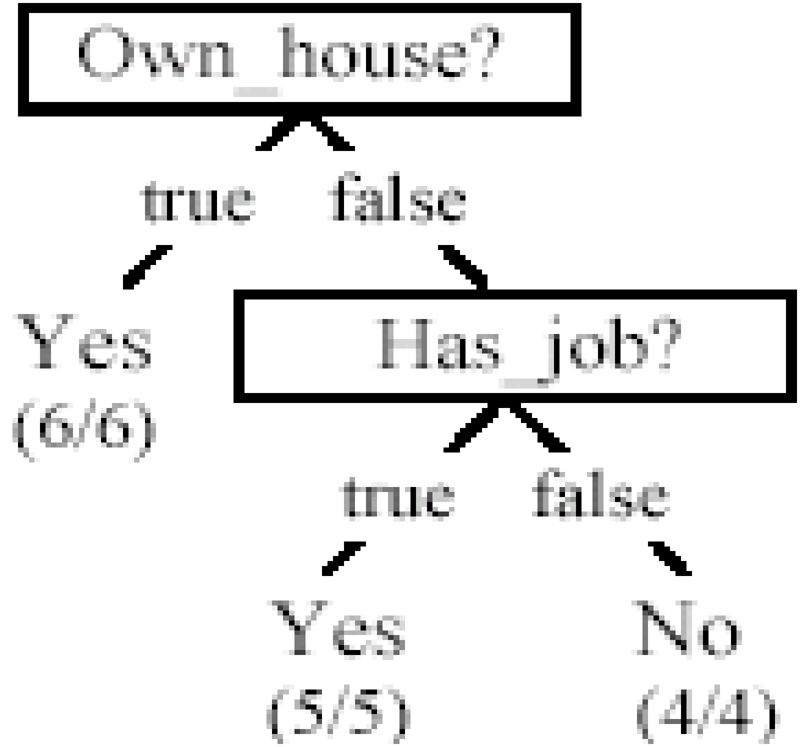
- No. Here is a simpler tree.
- We want smaller tree and accurate tree.
  - Easy to understand and perform better.

- Finding the best tree is NP-hard.
- All current tree building algorithms are heuristic algorithms



# From a decision tree to a set of rules

- A decision tree can be converted to a set of rules
- Each path from the root to a leaf is a rule.




Own_house = true → Class =Yes [sup=6/15, conf=6/6]
Own_house = false, Has_job = true → Class = Yes [sup=5/15, conf=5/5]
Own_house = false, Has_job = false → Class = No [sup=4/15, conf=4/4]

# Algorithm for decision tree learning

- Basic algorithm (a greedy **divide-and-conquer** algorithm)
  - Assume attributes are categorical now (continuous attributes can be handled too)
  - Tree is constructed in a top-down recursive manner
  - At start, all the training examples are at the root
  - Examples are partitioned recursively based on selected attributes
  - Attributes are selected on the basis of an impurity function (e.g., information gain)
- Conditions for stopping partitioning
  - All examples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority class is the leaf
  - There are no examples left

# Decision tree learning algorithm

```
. Algorithm decisionTree(D, A, T)
    if D contains only training examples of the same class c_j ∈ C then
        make T a leaf node labeled with class c_j;
    elseif A = ∅ then
        make T a leaf node labeled with c_j, which is the most frequent class in D
    else  // D contains examples belonging to a mixture of classes. We select a single
          // attribute to partition D into subsets so that each subset is purer
        p_0 = impurityEval-1(D);
        for each attribute A_i ∈ {A_1, A_2, …, A_k} do
            p_i = impurityEval-2(A_i, D)
        end
        Select A_g ∈ {A_1, A_2, …, A_k} that gives the biggest impurity reduction,
            computed using p_0 – p_i;
        if p_0 – p_g < threshold then     // A_g does not significantly reduce impurity p_0
          make T a leaf node labeled with c_j, the most frequent class in D.
        else                               // A_g is able to reduce impurity p_0
            Make T a decision node on A_g;
            Let the possible values of A_g be v_1, v_2, …, v_m. Partition D into m
                disjoint subsets D_1, D_2, …, D_m based on the m values of A_g.
            for each D_j in {D_1, D_2, …, D_m} do
                if D_j ≠ ∅ then
                    create a branch (edge) node T_j for v_j as a child node of T;
                    decisionTree(D_j, A-{A_g}, T_j)// A_g is removed
                end
            end
        end
    end
```

# Choose an attribute to partition data

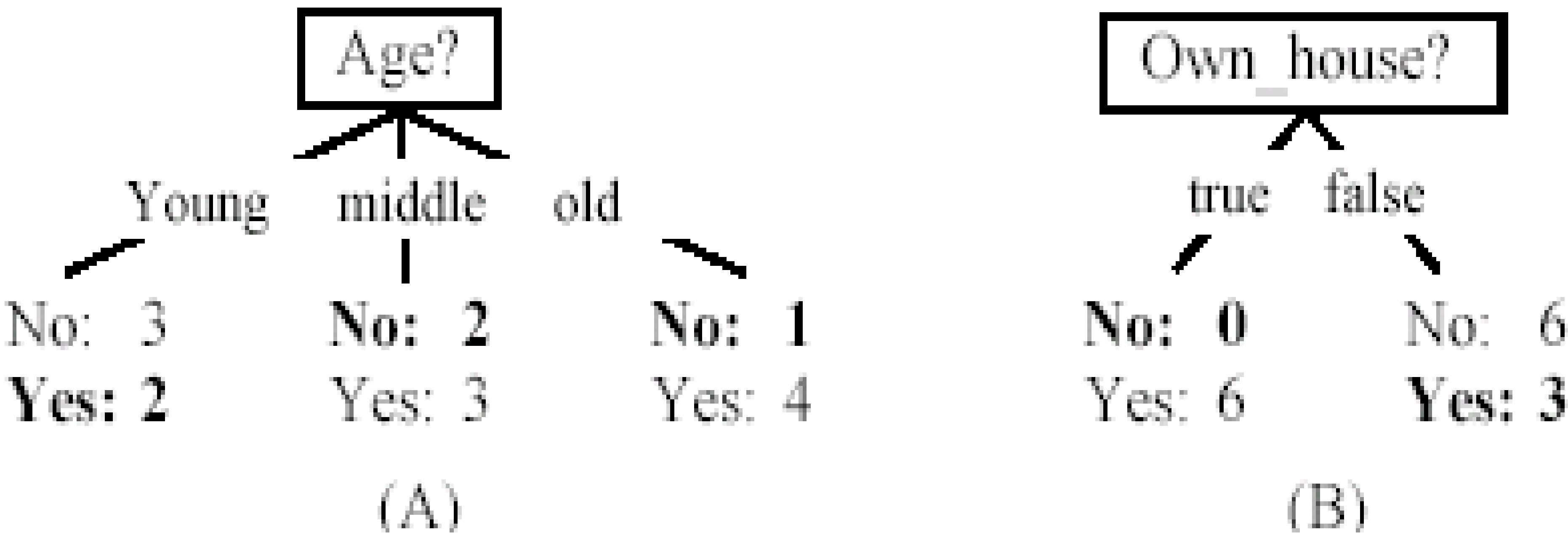
- The *key* to building a decision tree - which attribute to choose in order to branch.
- The objective is to reduce impurity or uncertainty in data as much as possible.
  - A subset of data is pure if all instances belong to the same class.
- The *heuristic* in C4.5 is to choose the attribute with the maximum Information Gain or Gain Ratio based on information theory.

# The loan data (reproduced)

Approved or not

| ID | Age | Has_Job | Own_House | Credit_Rating | Class |
|---|---|---|---|---|---|
| 1 | young | false | false | fair | **No** |
| 2 | young | false | false | good | **No** |
| 3 | young | true | false | good | **Yes** |
| 4 | young | true | true | fair | **Yes** |
| 5 | young | false | false | fair | **No** |
| 6 | middle | false | false | fair | **No** |
| 7 | middle | false | false | good | **No** |
| 8 | middle | true | true | good | **Yes** |
| 9 | middle | false | true | excellent | **Yes** |
| 10 | middle | false | true | excellent | **Yes** |
| 11 | old | false | true | excellent | **Yes** |
| 12 | old | false | true | good | **Yes** |
| 13 | old | true | false | good | **Yes** |
| 14 | old | true | false | excellent | **Yes** |
| 15 | old | false | false | fair | **No** |

# Two possible roots, which is better?




- Fig. (B) seems to be better.

# Information theory

- Information theory provides a mathematical basis for measuring the information content.
- To understand the notion of information, think about it as providing the answer to a question, for example, whether a coin will come up heads.
  - If one already has a good guess about the answer, then the actual answer is less informative.
  - If one already knows that the coin is rigged so that it will come with heads with probability 0.99, then a message (advanced information) about the actual outcome of a flip is worth less than it would be for a honest coin (50-50).

# Information theory (cont …)

- For a fair (honest) coin, you have no information, and you are willing to pay more (say in terms of $) for advanced information - less you know, the more valuable the information.
- Information theory uses this same intuition, but instead of measuring the value for information in dollars, it measures information contents in **bits**.
- One bit of information is enough to answer a yes/no question about which one has no idea, such as the flip of a fair coin

# Information theory: Entropy measure

- The entropy formula,

$$entropy(D) = -\sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

$$\sum_{j=1}^{|C|} \Pr(c_j) = 1,$$

- $\Pr(c_j)$ is the probability of class $c_j$ in data set $D$
- We use entropy as a measure of impurity or disorder of data set $D$. (Or, a measure of information in a tree)

# Entropy measure: let us get a feeling

1. The data set $D$ has 50% positive examples ($\Pr(positive) = 0.5$) and 50% negative examples ($\Pr(negative) = 0.5$).

$$entropy(D) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$$

2. The data set $D$ has 20% positive examples ($\Pr(positive) = 0.2$) and 80% negative examples ($\Pr(negative) = 0.8$).

$$entropy(D) = -0.2 \times \log_2 0.2 - 0.8 \times \log_2 0.8 = 0.722$$

3. The data set $D$ has 100% positive examples ($\Pr(positive) = 1$) and no negative examples, ($\Pr(negative) = 0$).

$$entropy(D) = -1 \times \log_2 1 - 0 \times \log_2 0 = 0$$

- As the data become purer and purer, the entropy value becomes smaller and smaller. This is useful to us!

# Information gain

- Given a set of examples $D$, we first compute its entropy:

$$entropy(D) = -\sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

- If we m the current tree, this will partition $D$ into v subsets $D_1$, $D_2$ …, $D_v$ . The expected entropy if $A_i$ is used as the current root:

$$entropy_{A_i}(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times entropy(D_j)$$

# Information gain (cont …)

- Information gained by selecting attribute $A_i$ to branch or to partition the data is

$$gain(D, A_i) = entropy(D) - entropy_{A_i}(D)$$

- We choose the attribute with the highest gain to branch/split the current tree.

# An example

$$entropy(D) = -\frac{6}{15}\times\log_2\frac{6}{15} - \frac{9}{15}\times\log_2\frac{9}{15} = 0.971$$

| ID | Age | Has_Job | Own_House | Credit_Rating | Class |
|---|---|---|---|---|---|
| 1 | young | false | false | fair | No |
| 2 | young | false | false | excellent | No |
| 3 | young | true | false | good | Yes |
| 4 | young | true | true | good | Yes |
| 5 | young | false | false | fair | No |
| 6 | middle | false | false | fair | No |
| 7 | middle | false | false | good | No |
| 8 | middle | true | true | good | Yes |
| 9 | middle | false | true | excellent | Yes |
| 10 | middle | false | true | excellent | Yes |
| 11 | old | false | true | excellent | Yes |
| 12 | old | false | true | good | Yes |
| 13 | old | true | false | good | Yes |
| 14 | old | true | false | excellent | Yes |
| 15 | old | false | false | fair | No |

$$entropy_{Own_{house}}(D) = -\frac{6}{15}\times entropy(D_1) - \frac{9}{15}\times entropy(D_2)$$
$$= \frac{6}{15}\times 0 + \frac{9}{15}\times 0.918$$
$$= 0.551$$

$$entropy_{Age}(D) = -\frac{5}{15}\times entropy(D_1) - \frac{5}{15}\times entropy(D_2) - \frac{5}{15}\times entropy(D_3)$$
$$= \frac{5}{15}\times 0.971 + \frac{5}{15}\times 0.971 + \frac{5}{15}\times 0.722$$
$$= 0.888$$

| Age | Yes | No | entropy(Di) |
|---|---|---|---|
| young | 2 | 3 | 0,97 |
| middle | 3 | 2 | 0,97 |
| old | 4 | 1 | 0,72 |
| | | | |

$gain(D, \text{Age}) = 0.971 - 0.888 = 0.083$

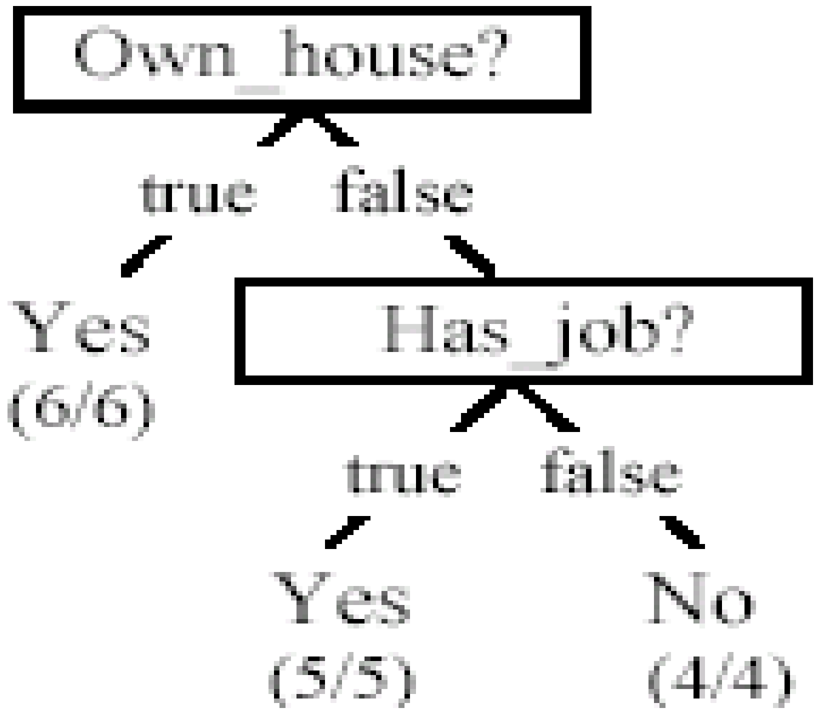
$gain(D, \text{Own\_house}) = 0.971 - 0.551 = 0.420$

$gain(D, \text{Has\_Job}) = 0.971 - 0.647 = 0.324$

$gain(D, \text{Credit\_Rating}) = 0.971 - 0.608 = 0.363$

- Own_house is the best choice for the root.

# We build the final tree

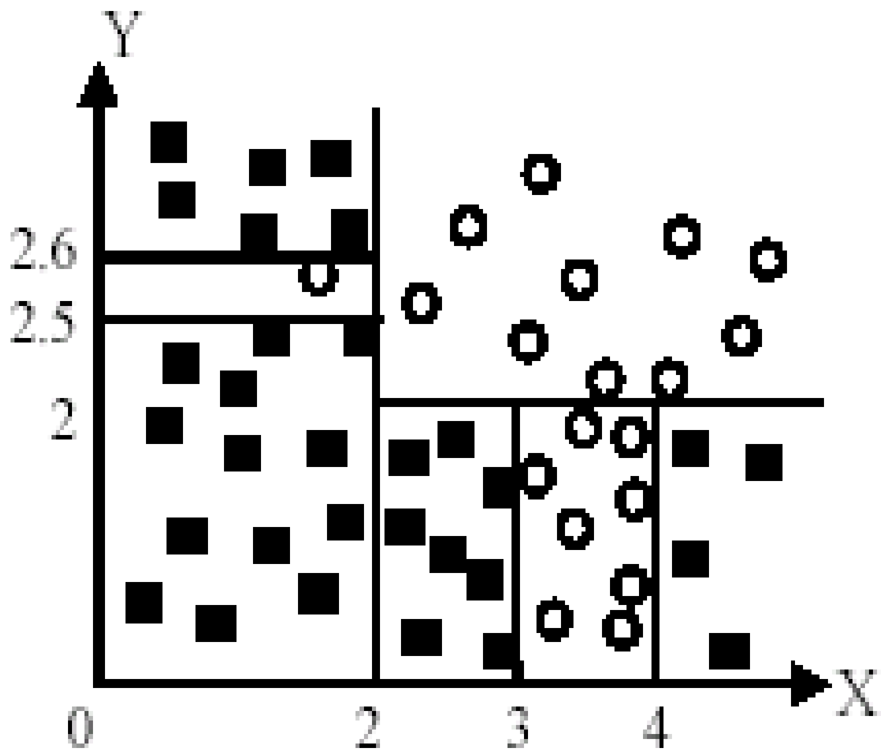



- We can use information gain ratio to evaluate the impurity as well (see the handout)

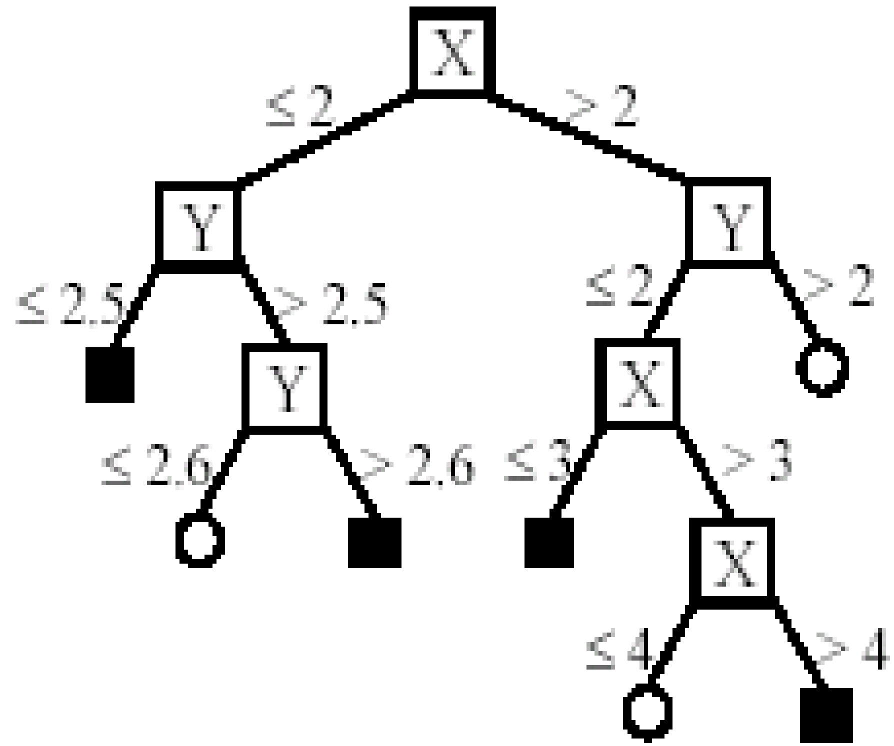
# Handling continuous attributes

- Handle continuous attribute by splitting into two intervals (can be more) at each node.
- How to find the best threshold to divide?
  - Use information gain or gain ratio again
  - Sort all the values of an continuous attribute in increasing order $\{v_1, v_2, \ldots, v_r\}$,
  - One possible threshold between two adjacent values $v_i$ and $v_{i+1}$. Try all possible thresholds and find the one that maximizes the gain (or gain ratio).

# An example in a continuous space




(A) A partition of the data space



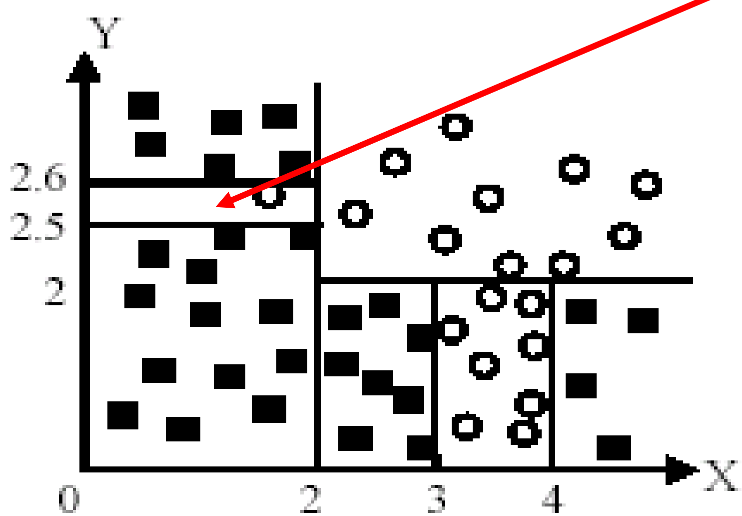

(B). The decision tree

# Avoid overfitting in classification

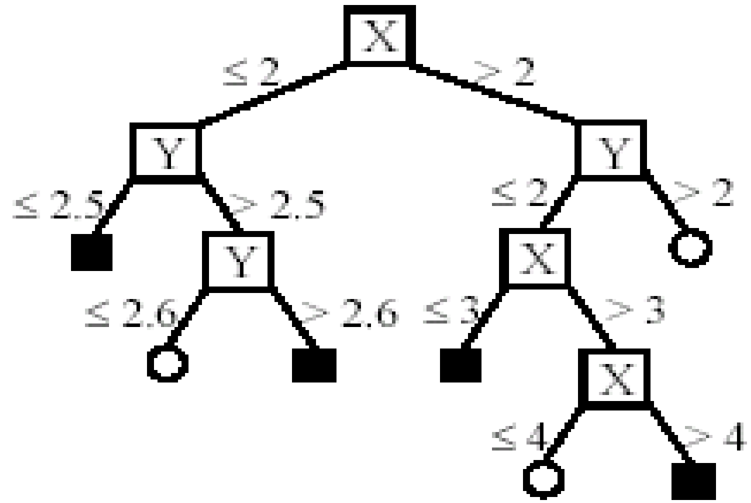
- Overfitting:  A tree may overfit the training data
  - Good accuracy on training data but poor on test data
  - Symptoms: tree too deep and too many branches, some may reflect anomalies due to noise or outliers
- Two approaches to avoid overfitting
  - Pre-pruning: Halt tree construction early
    - Difficult to decide because we do not know what may happen subsequently if we keep growing the tree.
  - Post-pruning: Remove branches or sub-trees from a "fully grown" tree.
    - This method is commonly used. C4.5 uses a statistical method to estimates the errors at each node for pruning.
    - A validation set may be used for pruning as well.

# An example

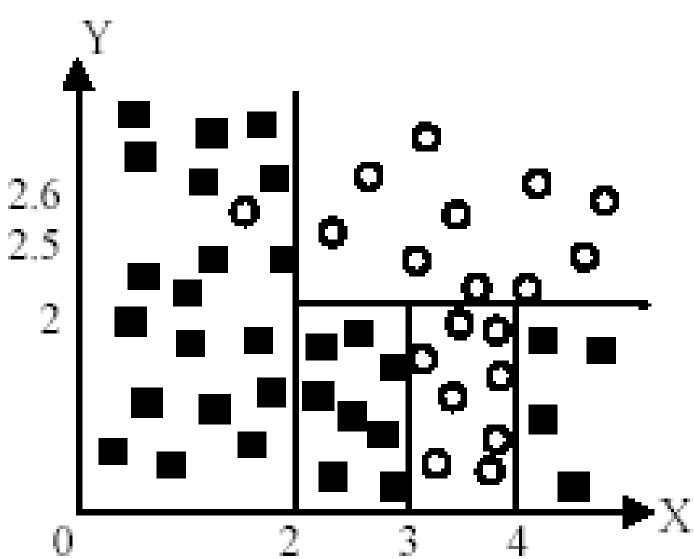
Likely to overfit the data

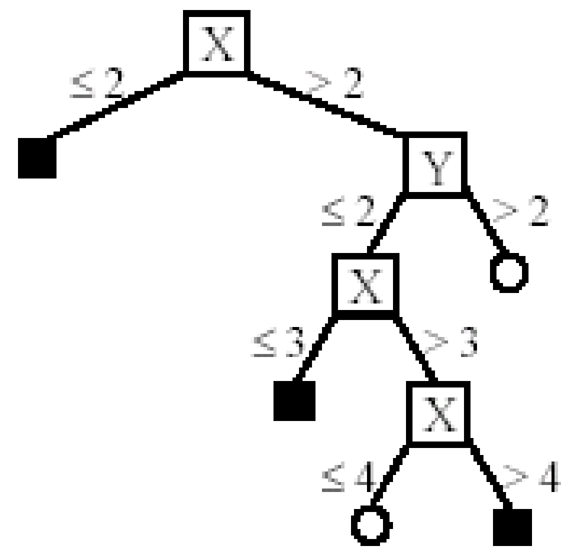



(A) A partition of the data space




(B). The decision tree



# Other issues in decision tree learning

- From tree to rules, and rule pruning
- Handling of miss values
- Handing skewed distributions
- Handling attributes and classes with different costs.
- Attribute construction
- Etc.