Chapter 4 – Classification

These slides are based on material and slides from:

1) Dunham, Data Mining – Introductory and Advanced Topics
2) Han, Kamber and Pei, Data Mining – Concepts and Techniques
3) Tan, Stienback and Kumar, Introduction to Data Mining
4) Zaki and Meira JR., Data Mining and Analysis – Fundamental Concepts and Algorithms
5) Liu, Web Data Mining – Exploring Hyperlinks, Contents, and Usage Data

Introduction

- Given a predefined set of classes and a training dataset, build a model to help classify objects into the different classes
- Applications
  - Classifying an image as a cat or a dog
  - Identify mushrooms as poisonous or edible
  - Identifying fake news
  - Predict when a river will flood
  - Pattern recognition – recognize hand written digits
  - Medical diagnosis
  - Fraud and spam detection
  - Identify individuals with credit risks
  - Speech recognition

Classification vs. Prediction

- Estimation and prediction can be considered classification
  - Estimating a person's age
  - Predicting stock market behavior
- Prediction models a continuous function while classification models a discrete-valued function
- Classification
  - predicts categorical class labels (discrete or nominal)
- Prediction
  - predicts unknown or missing values (models continuous-valued functions)

Formal Definition

- Given a database $D=\{t_1,t_2,\ldots,t_n\}$ and a set of classes $C=\{C_1,\ldots,C_m\}$, the ***Classification Problem*** is to define a mapping $f:D\rightarrow C$ where each t_i is assigned to one class.
- Classes are **disjoint** and form a **partition** of D.

Supervised vs. Unsupervised Learning

- **Supervised learning (classification)**
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- **Unsupervised learning (clustering)**
  - The class labels of training data is unknown
  - Given a set of data, the task is to establish the existence of classes or clusters in the data

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.

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 model can be represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known labels of test samples are compared with the classified results from the model
    - Test set is independent of training set
- If the accuracy is acceptable, use the model to classify new tuples

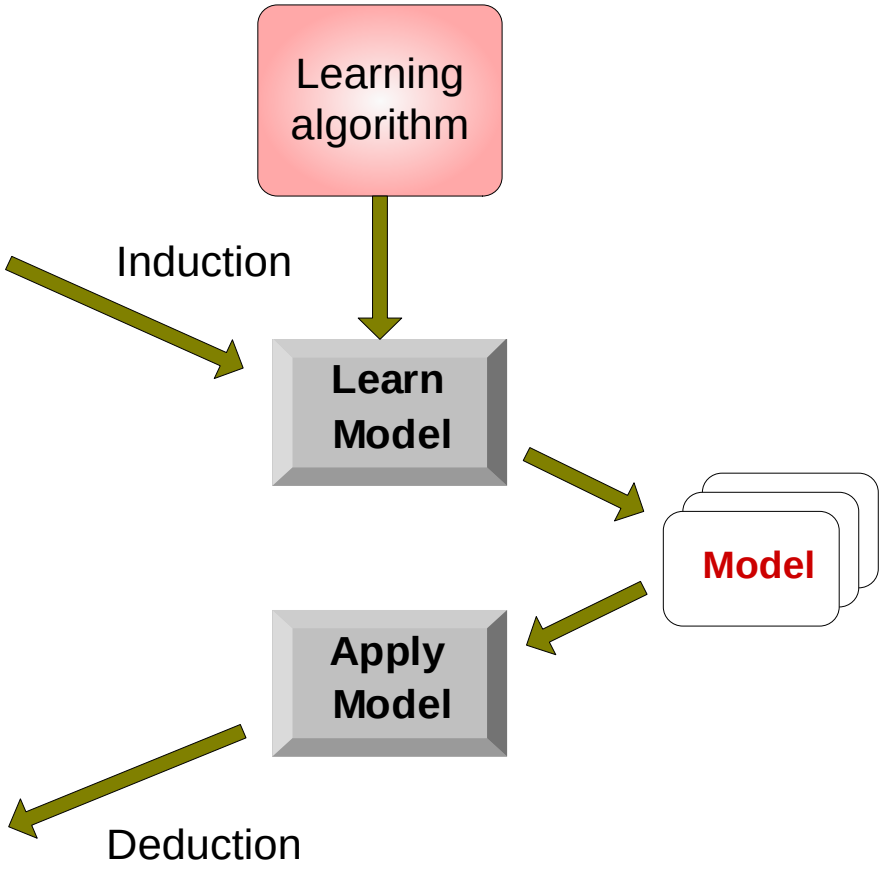
Illustrating Classification Task

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	?

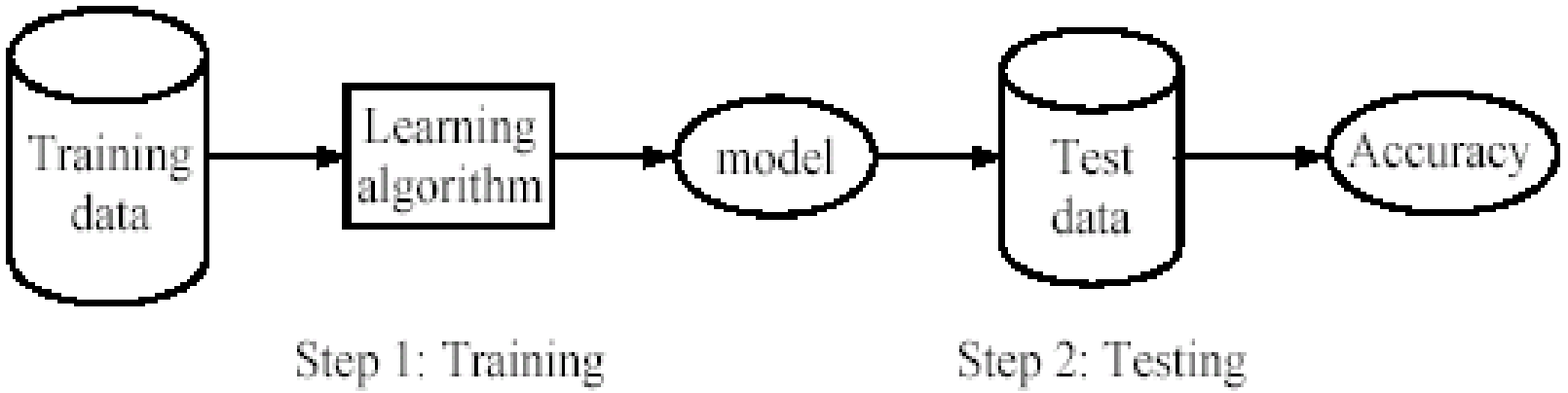
Test Set



Accuracy

- Other name: accuracy rate, classification accuracy, prediction accuracy: percentage of test samples that are correctly classified

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



Comparing Classification Methods

- Prediction accuracy
- Speed
  - time to construct the model
  - time to use the model
- Robustness
  - handling noise and missing values
- Scalability
  - ability to construct model efficiently given large data
- Interpretability
  - understanding and insight provided by the model
- Other measures
  - e.g., goodness of rules; decision tree size; compactness of classification rules

Categorization of Classification Algorithms

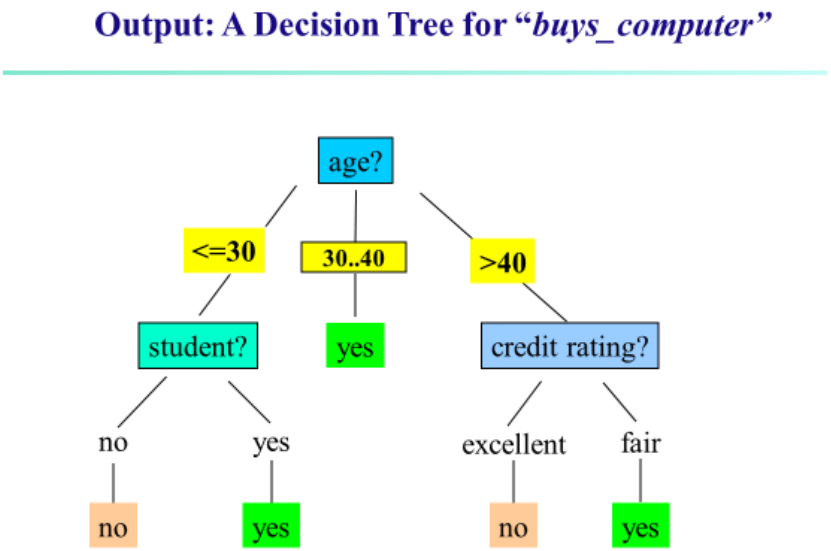
- Classification Techniques
  - Decision Trees
  - Statistical
  - Distance
  - Rules
  - Neural Networks

Decision Tree-Based Algorithms

- Decision trees are among the most widely used techniques for classification
  - classification accuracy is competitive with other methods, and
  - very efficient
- The classification model is a tree, called decision tree (DT)
- C4.5 by Ross Quinlan is perhaps the best known system

Decision Tree-Based Algorithms

- A tree structure where:
  - each internal node represents a test on an attribute,
  - each branch represents an outcome of the test,
  - and leaf nodes reflect decision outcomes
    - different classes and class distributions



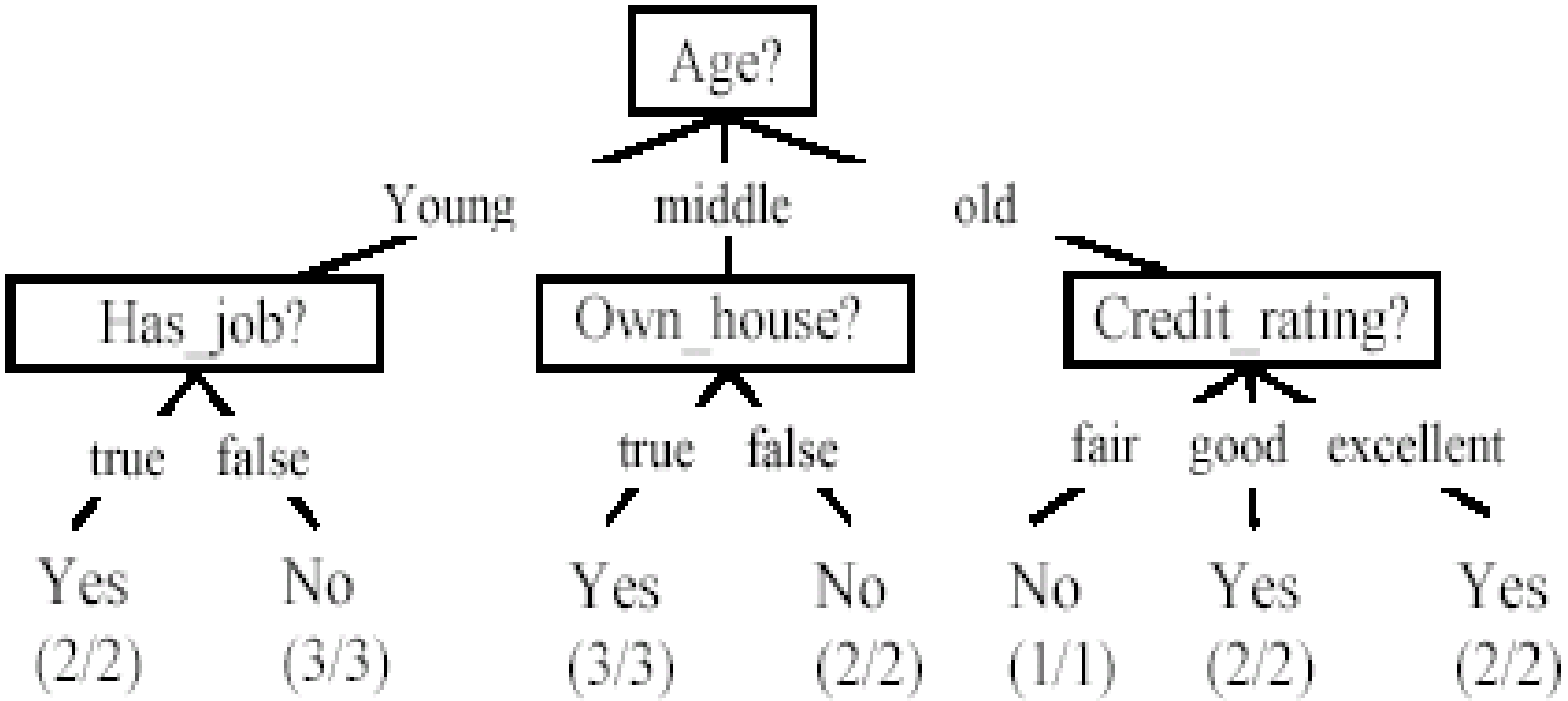
DT- Loan Dataset

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 Dataset

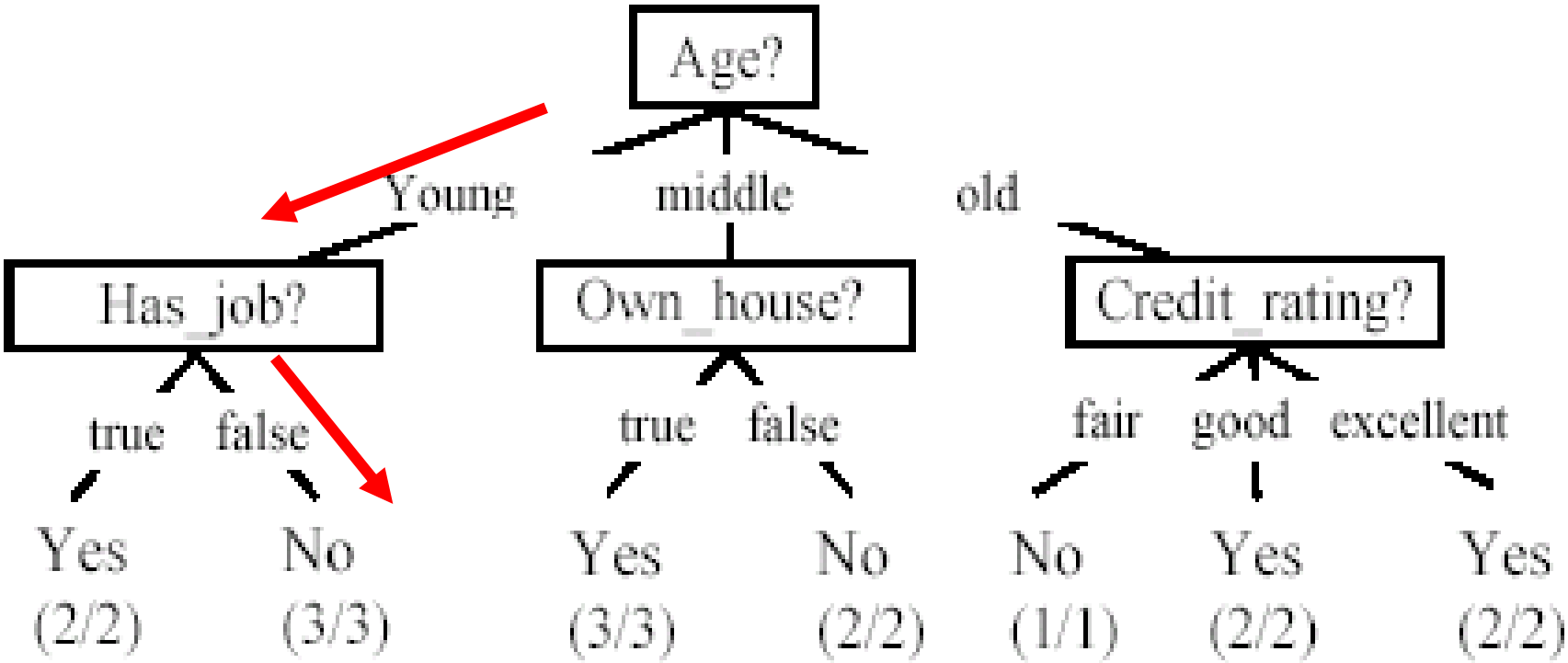
- Decision nodes and leaf nodes (classes)



Using the Decision Tree

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

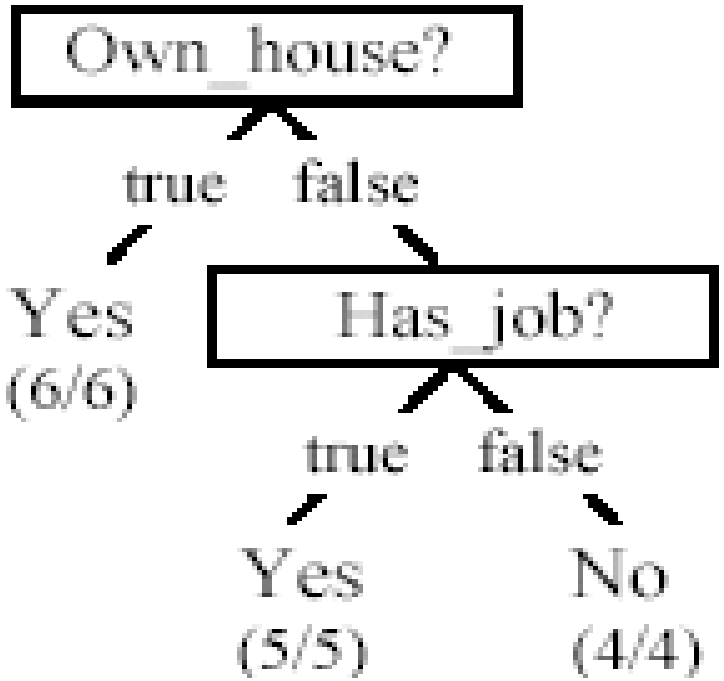
No



Is the DT Unique?

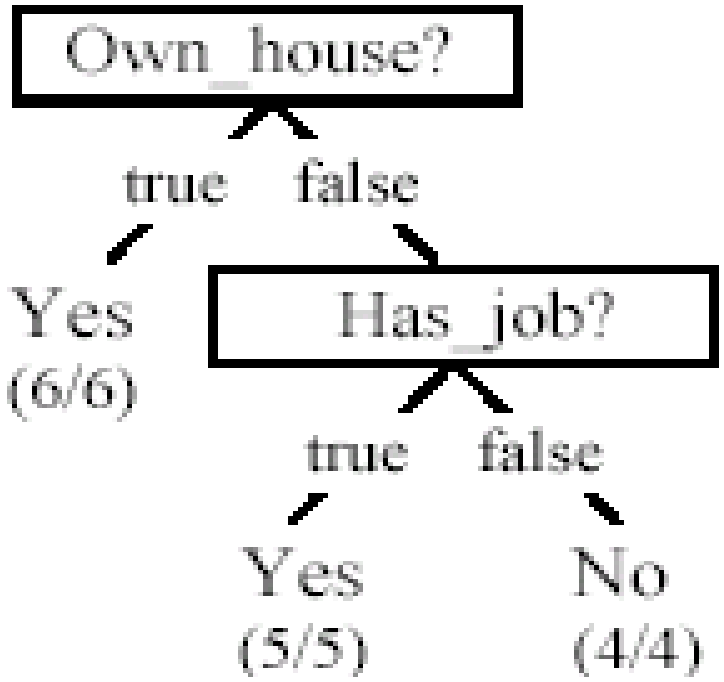
- No. Here is a simpler tree
- We want smaller tree and accurate tree
  - Easy to understand and perform better

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



From a DT to Classification 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]

Constructing a Decision Tree

- Constructing a DT is aka decision tree induction
- Many Algorithms:
  - ID3
  - C4.5
  - C5.0
  - CART
  - SLIQ,SPRINT

The Basic Algorithm – ID3

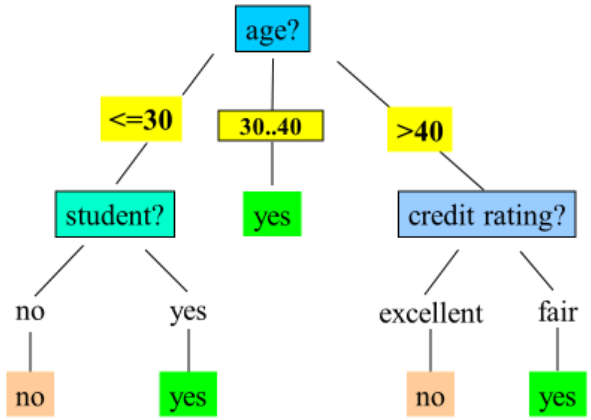
- Assume attributes are categorical
  - Continuous attributes can be handled too
- A greedy divide-and-conquer algorithm
- Tree is constructed in a top-down recursive manner
- At start, all the training examples are at the root
- Examples are partitioned recursively based on selected attributes
- Attributes are selected on the basis of a measure called information gain

ID3

- A greedy divide-and-conquer algorithm
- 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

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

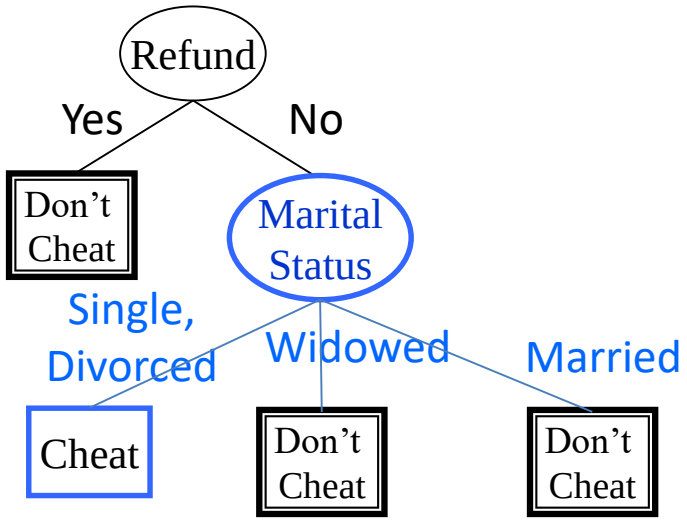
Output: A Decision Tree for "*buys_computer*"



Decision Tree Induction

- Let D_t be the set of training samples that reach a node t
- General Procedure:
  - If all the samples in D_t belong to the same class C, then t is a leaf node labeled as C
  - If D_t is an empty set, then t is a leaf node labeled by the most common class at the parent
  - If attribute set is empty, then t is a leaf node labeled by the majority class
  - If D_t contains samples that belong to more than one class,
    - use an attribute test to split the data into smaller subsets
    - recursively apply the procedure to each subset

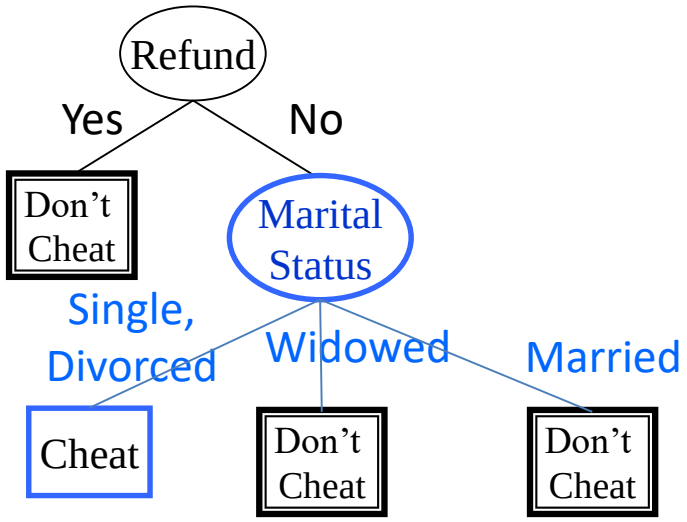
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Decision Tree Induction

- Let D_t be the set of training samples that reach a node t
- General Procedure:
  - If all the samples in D_t belong to the same class C, then t is a leaf node labeled as C
  - If D_t is an empty set, then t is a leaf node labeled by the most common class at the parent
  - If attribute set is empty, then t is a leaf node labeled by the majority class
  - If D_t contains samples that belong to more than one class,
    - use an attribute test to split the data into smaller subsets
    - recursively apply the procedure to each subset

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	Yes	Married	120K	No
3	Yes	Divorced	220K	No
4	No	Married	100K	No
5	No	Married	75K	No
6	No	Married	60K	No
7	No	Single	70K	No
8	No	Single	85K	Yes
9	No	Single	90K	Yes
10	No	Divorced	95K	Yes



The Basic Algorithm – ID3

- 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

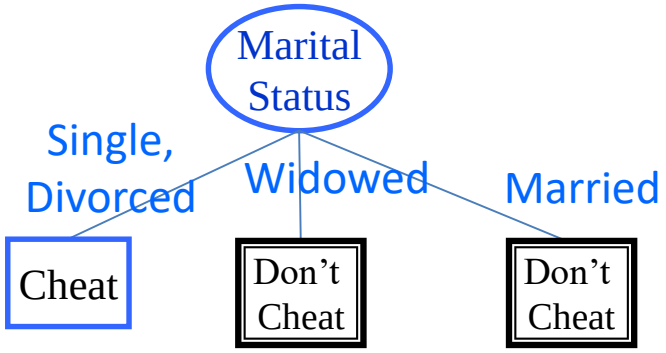
ID3 Algorithm in More Details

- Algorithm Generate_DT(D, attribute_list)
- Input: D, attribute_list
- Output: a DT

1. Create a node N
2. If samples are all of the same class C, then
   return N as a leaf node labeled with class C
3. If attribute_list is empty, then
   return N as a leaf node labeled with the majority class
4. Select test_attribute, the attribute among attribute_list with the most information gain

ID3 Algorithm in More Details (cont.)

5. Label node N with test_attribute
6. For each known value a_i of test_attribute
   i. grow a branch from node N for the condition test_attribute = a_i
   ii. let S_i be the set of samples in D for which test_attribute = a_i
   iii. if S_i is empty, then
      attach a leaf node labeled with the majority class in D
   iv. else attach the node returned by Generate_DT(S_i, attribute_list - test_attribute)



When does Algorithm Stop?

- Algorithm stops at steps 2, 3 and 6-III
  - 2) all samples for a given node belong to the same class
  - 3) attribute_list is empty
  - 6-III) S_i is empty, i.e., there are no samples left

Choosing an Attribute to Partition Data

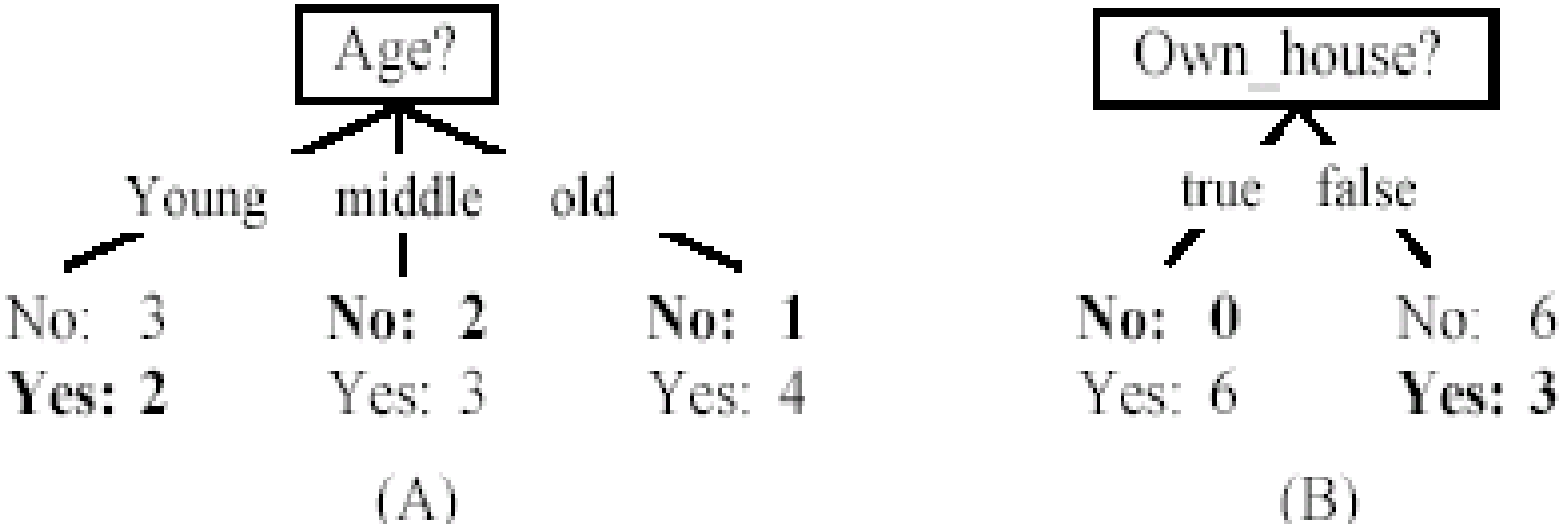
- The key to building a decision tree - which attribute to choose in order to branch
- The objective is to reduce impurity or uncertainty in data as much as possible
  - A subset of data is pure if all instances belong to the same class.
- The *heuristic* in ID3 is to choose the attribute with the maximum Information Gain based on entropy reduction

The Loan Dataset

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

Entropy Measure

- Entropy is a measure of impurity, randomness, uncertainty
- We use entropy as a measure of impurity or disorder of dataset *D*. (Or, a measure of information in a tree)
- 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*

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!

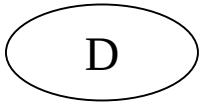
Entropy

- Entropy is a measure of impurity, randomness, uncertainty "in a set of data"
- Entropy of node T is minimum when all samples are of the same class
- Entropy of node T is maximum when there is equal representation of all classes

Information Gain

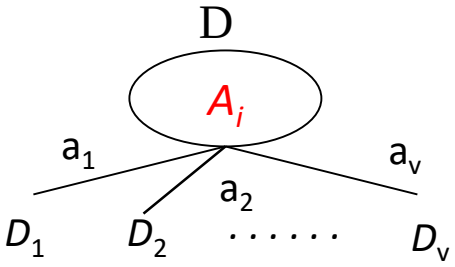
- Given a set of samples *D*, we first compute its entropy:

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



- If we select attribute A_i, with v values, as the test attribute at the current node, this will partition *D* into v subsets D_1, D_2 …, D_v
- The expected entropy if A_i is used as the current test attribute:

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

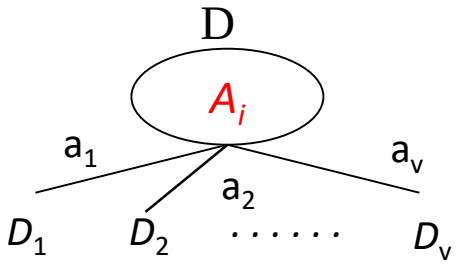


Information Gain (cont.)

- 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.



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	0.971
middle	3	2	0.971
old	4	1	0.722

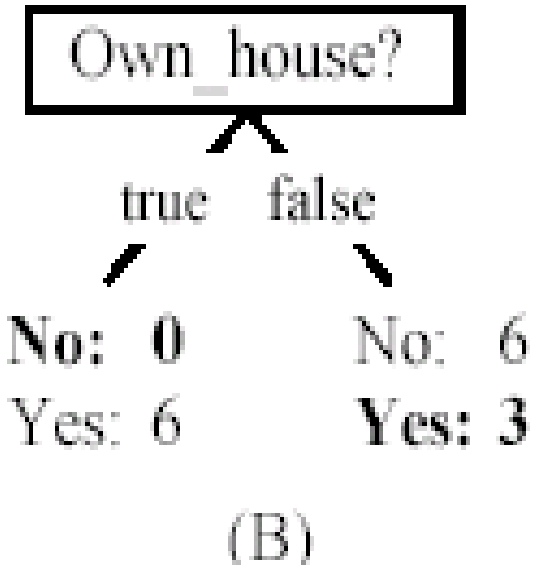
Own_house is the best choice for the root.

$gain(D, \text{Age}) = 0.971 - 0.888 = 0.083$
$gain(D, \text{Own\_house}) = 0.971 - 0.551 = 0.420$
$gain(D, \text{Has\_Job}) = 0.971 - 0.647 = 0.324$
$gain(D, \text{Credit\_Rating}) = 0.971 - 0.608 = 0.363$

Calculating entropy(D_2)

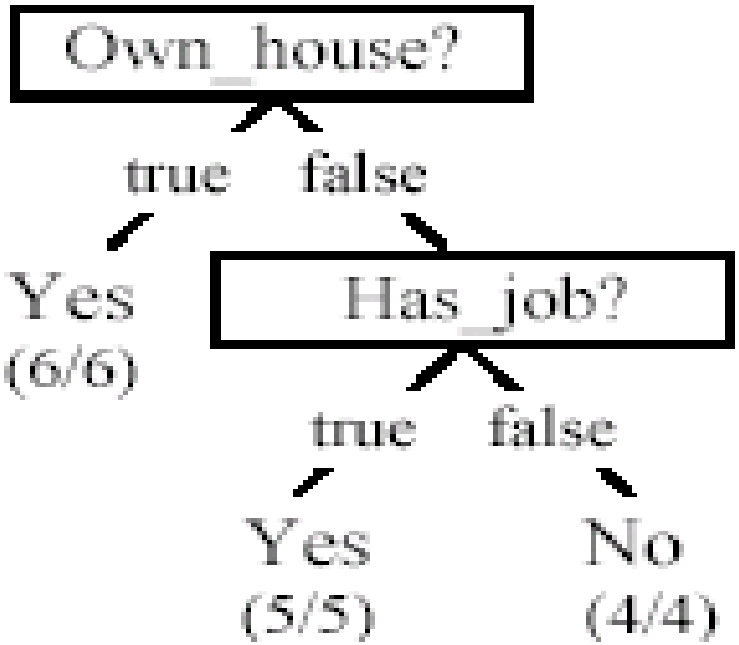
$$entropy_{Own\_house}(D) = \frac{6}{15} \times entropy(D_1) + \frac{9}{15} \times entropy(D_2)$$
$$= \frac{6}{15} \times 0 + \frac{9}{15} \times 0.918$$
$$= 0.551$$

entropy(D_2) = -Pr(Class=Yes)*log(Pr(Class = Yes))
-Pr(Class=No)*log(Pr(Class = No))
= - $\frac{6}{9}$log($\frac{6}{9}$) - $\frac{3}{9}$log($\frac{3}{9}$) = 0.918



(B)

The Final Tree

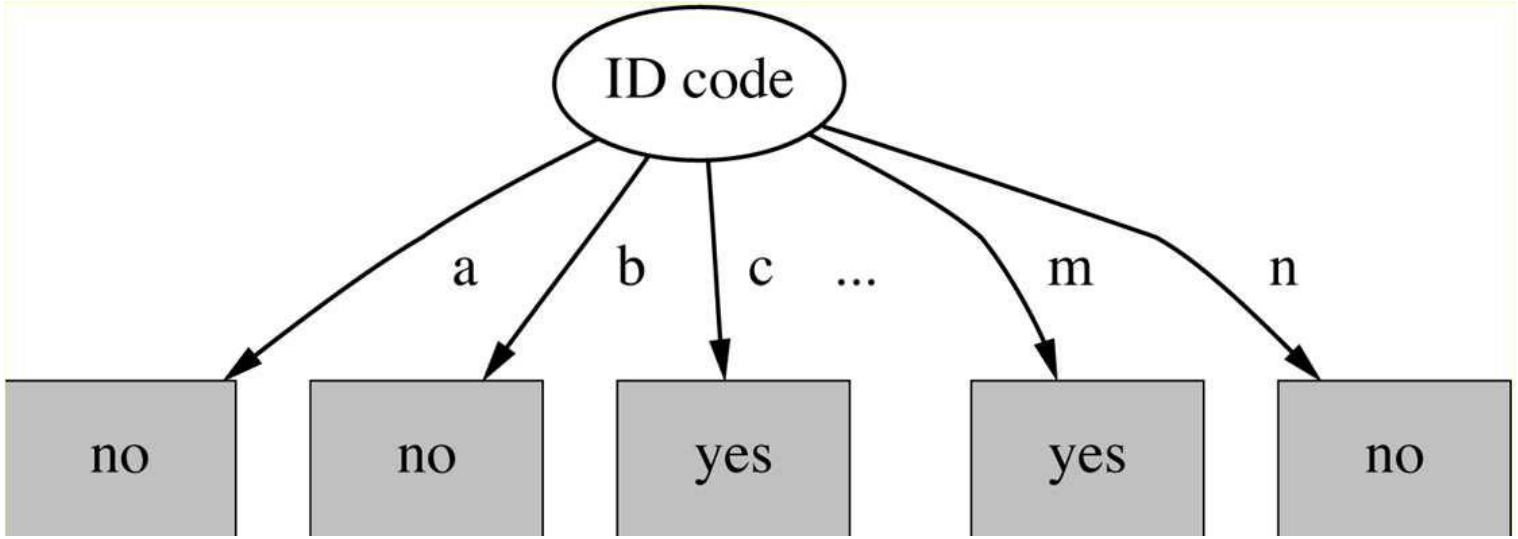


C4.5 and C5.0

- Extensions of ID3
- Details of C5.0 are not public
- C4.5 improves on ID3 as follows:
  - allows for continuous-valued attributes
  - handles missing attribute values
  - uses tree pruning to improve the efficiency of the DT
  - uses a better heuristic (GainRatio vs Information Gain)
  - allows classification via either DT or classification rules
  - simplifies complex rules with long LHS
  - a default rule assigning to the majority class is used

Gain Ratio

- Information Gain is biased towards selecting attributes with many values
- An attribute with many values results in purer partitions, but may be useless for classification



Gain Ratio

- GainRatio: adjusts Information Gain by normalizing by SplitInfo, which takes into consideration the number and sizes of the different partitions of the attribute under consideration

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

$$\text{SplitInfo}(A) = -\sum_{i=1}^{k} \frac{|D_i|}{|D|} \log \frac{|D_i|}{|D|} = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

- C4.5 select the attribute with the maximum GainRatio as the test attribute

SplitInfo

- The split information represents the potential information generated by splitting the dataset into k partitions:

$$\text{SplitInfo}(A) = -\sum_{i=1}^{k} \frac{|D_i|}{|D|} \log \frac{|D_i|}{|D|} = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

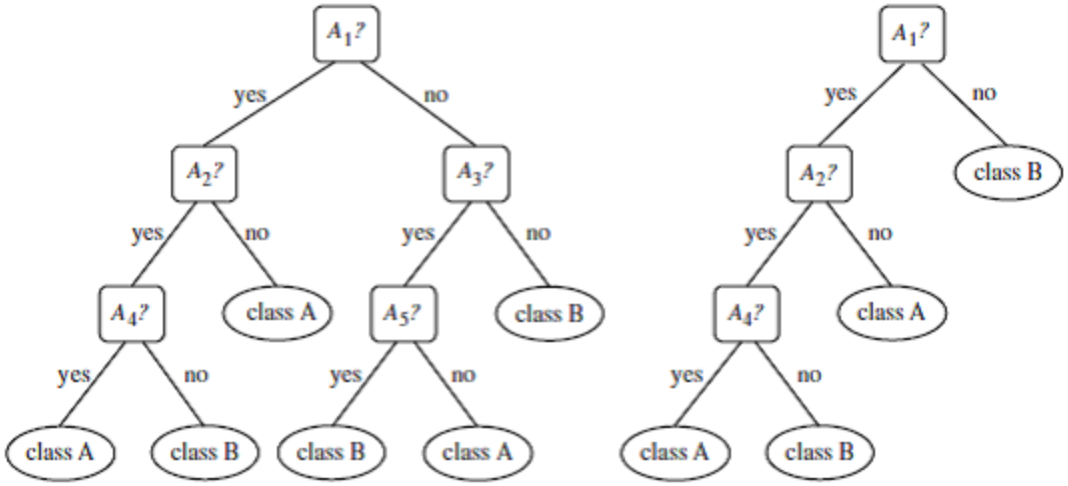
- SplitInfo(A) = entropy(D) based on the values of A
- High split info: many partitions that have about the same size
  - the larger k, the larger splitInfo, which = $\log_2 K$
- Low split info: few partitions hold most of the tuples
  - e., g., [2, 2, 96] is purer than [2, 2, 2, 94] is purer than [5, 5, 5, 85]

Overfitting

- Overfitting: a constructed tree may overfit the training data
  - too many branches, due to noise or outliers
  - poor accuracy for unseen samples
- Definition: A decision tree T overfits the training data if there exists another tree T' such that T performs better than T' over the training samples while T ' has a higher accuracy than T over the entire distribution of the data

Tree Pruning

- Tree pruning removes the least reliable branches in a DT, which usually results in better classification
- A pruned DT is more efficient and easier to understand



Approaches for Tree Pruning

- 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
    - A node is pruned by removing its branches and is labeled by the most common class

Cost Complexity Postpruning Algorithm

- The "cost complexity" of a tree is defined as a function of the number of leaves in the tree and the error rate of the tree – e.g., err(T)/|leaves(T)| or err(T) + α|leaves(T)|
- Construct a full decision tree, T
- Compute the "cost complexity", c, of T
- Internal nodes are removed from T in a bottom-up fashion
- Let T' be the tree after removing an internal node, N, from T
- Compute the "cost complexity", c', of T'
- If c' < c, prune N, otherwise keep N
- The algorithm generates a set of progressively pruned trees
- The smallest decision tree that minimizes the cost complexity is used
- A separate validation set is used when calculating "cost complexity"