Supervised Learning

Example

- 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

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

Focus

- 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 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**.

Data and Goal

Data: A set of data records (also called examples, instances or cases) described by

k attributes: A_1, A_2, … 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.

Example

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**

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

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.

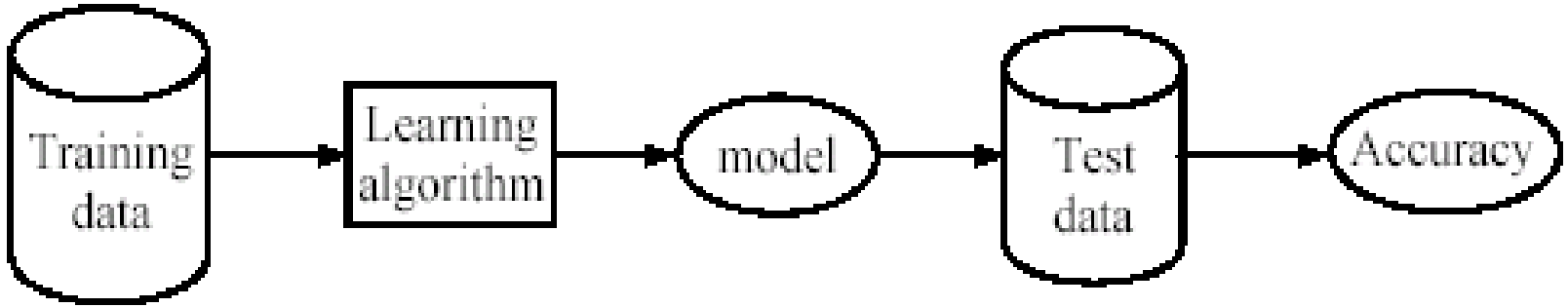
Unsupervised learning (clustering)

- 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

Steps

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

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



Problem

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.

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):

Assumptions

- 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

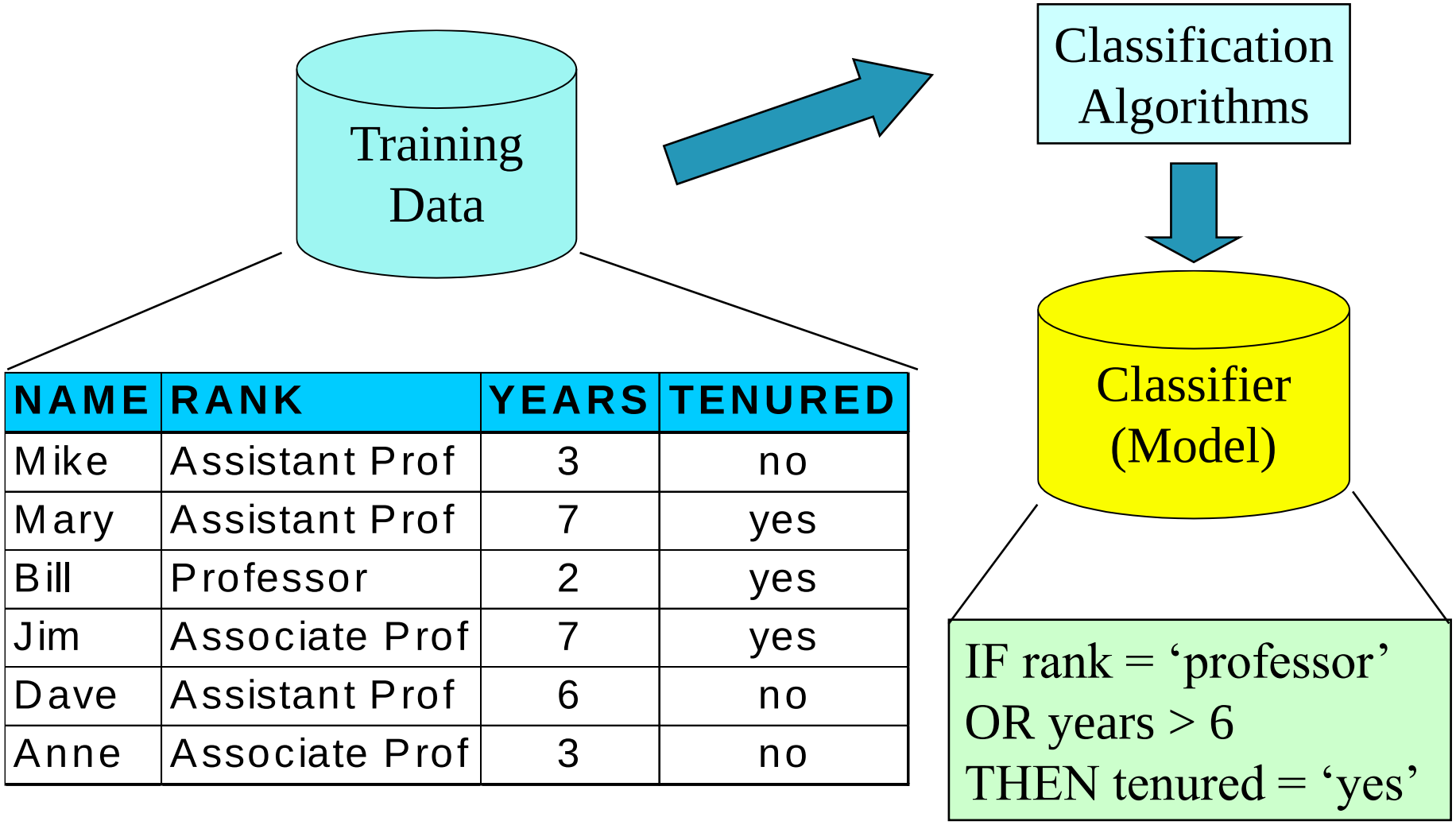
Classification vs Numeric Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Numeric Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

Steps

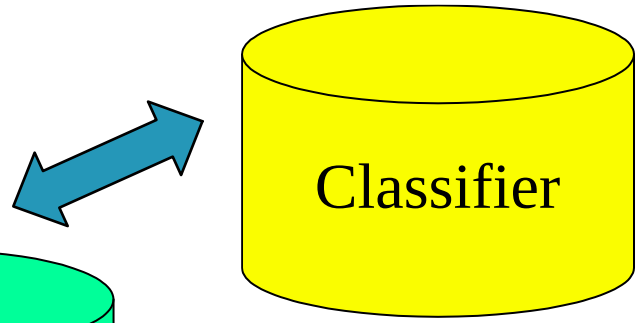
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision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

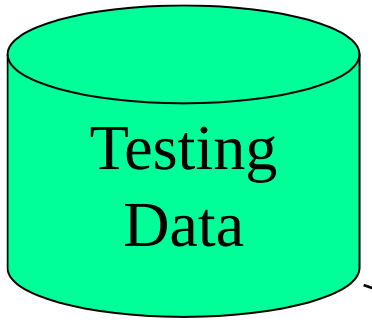
Model Construction

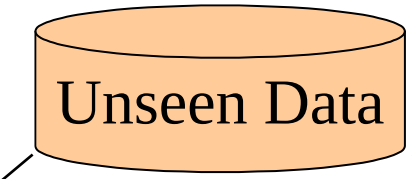


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Using the Model







(Jeff, Professor, 4)

Tenured?

Yes

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

Decision Trees

Partially adopted from CS583 University of Illinois & ‘Data Mining Concepts and Techniques’

Introduction

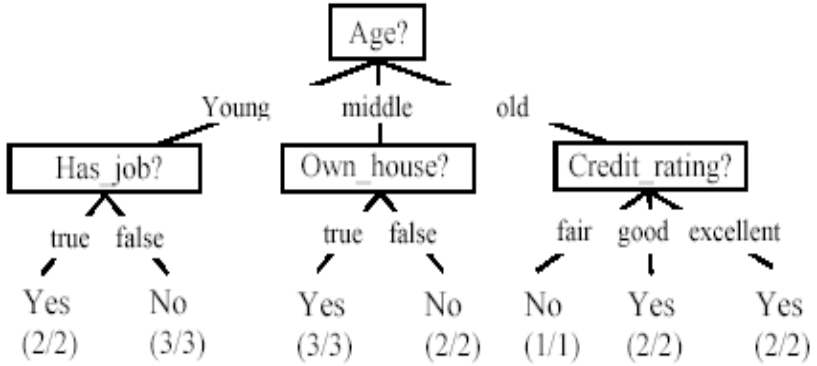
- Decision tree learning is one of the most widely used techniques for classification.
  - Its classification accuracy is competitive with other methods, and it is very efficient.

- The classification model is a tree, called decision tree.

- C4.5 by Ross Quinlan is perhaps the best known system.

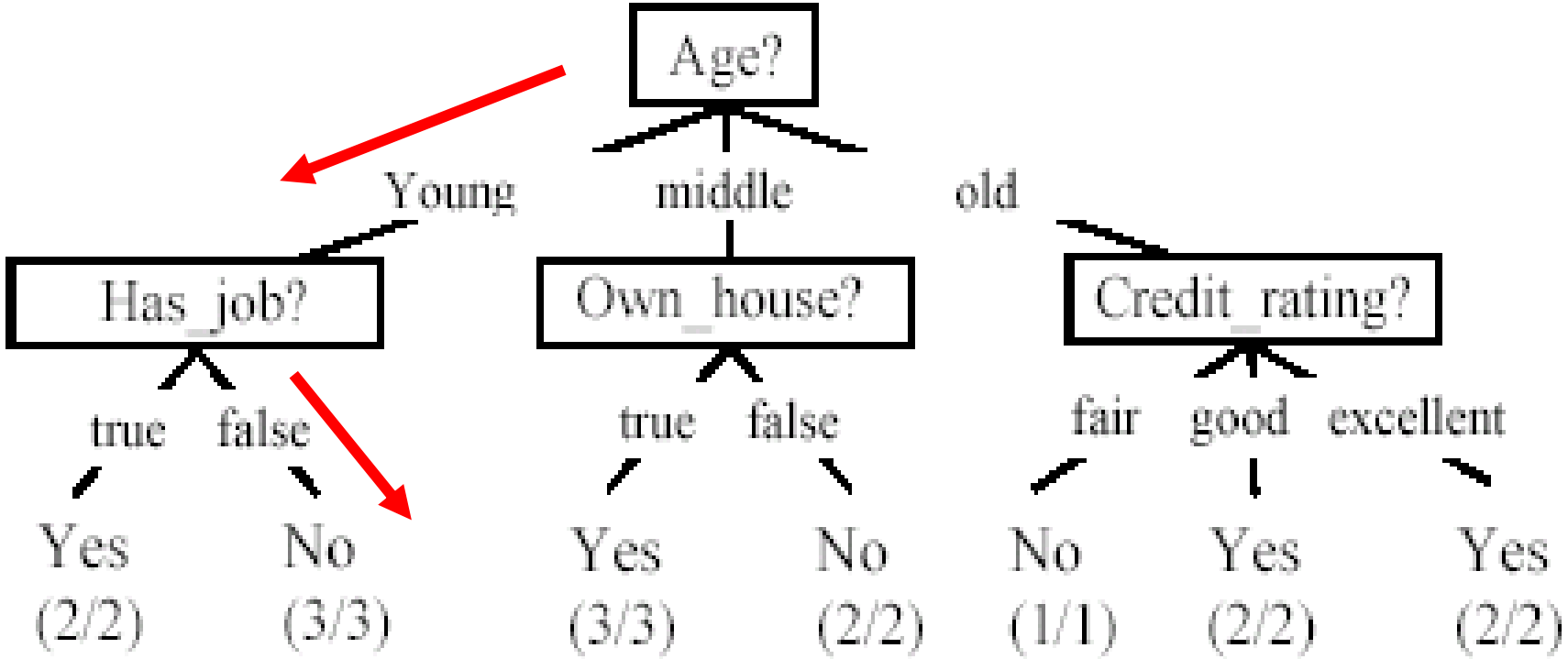
Example

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**



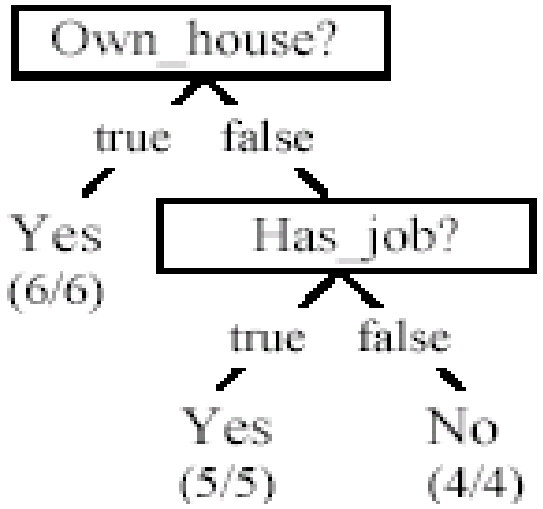
Use the Decision Tree

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



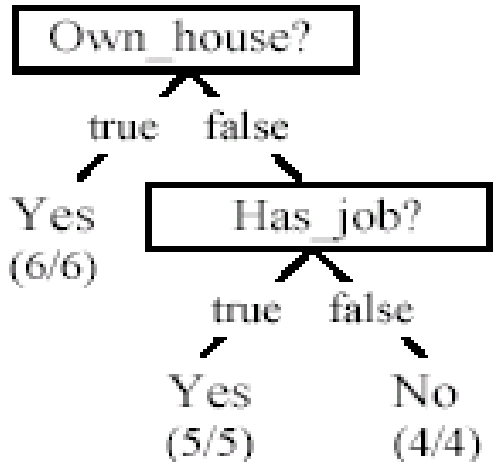
Uniqueness

- 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



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]

Problem

- 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

Purity

- 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.

Example

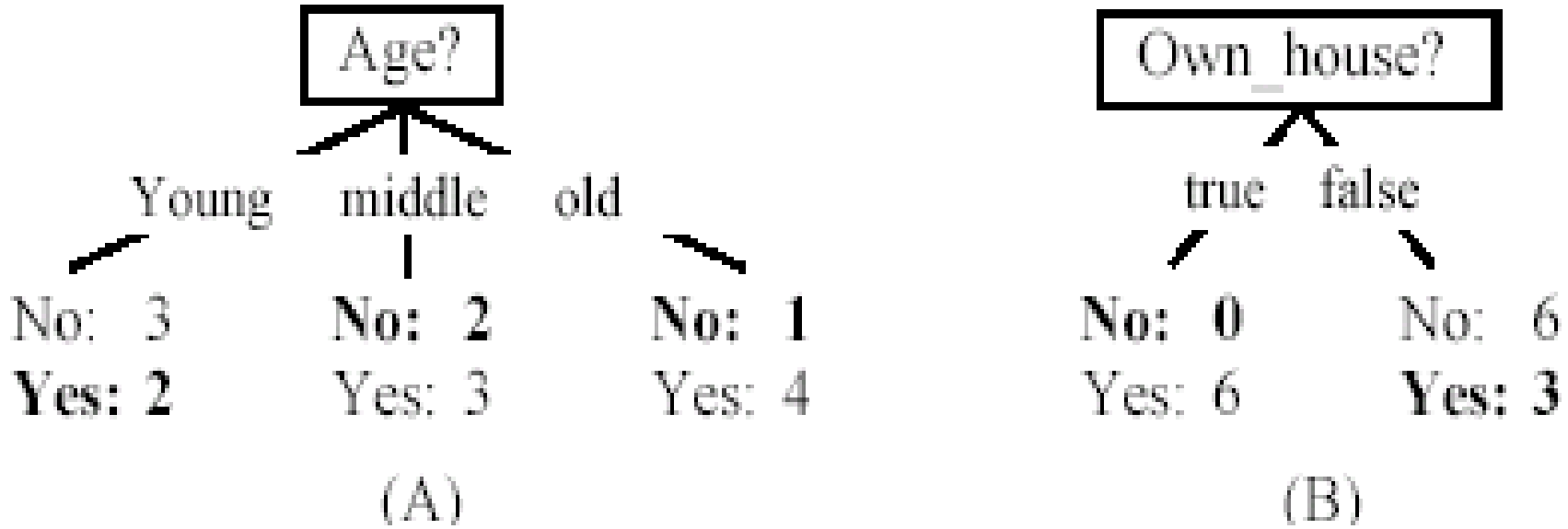


Fig. (B) seems to be better.

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

- 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

Entropy

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)

Example

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

- 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 make attribute A_i, with v values, the root of 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

- 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.

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

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

Example

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

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

$$= \frac{6}{15} \times 0 + \frac{9}{15} \times 0.918$$

$$= 0.551$$

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

$$= \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.722$$

$$= 0.888$$

- Own_house is the best choice for the root.

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

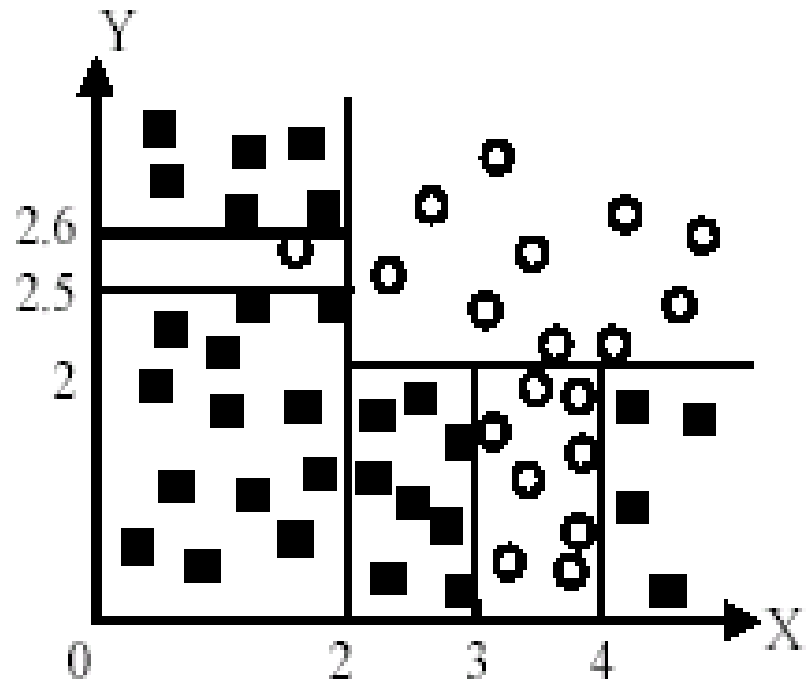
Age	Yes	No	entropy(Di)
young	2	3	0.971
middle	3	2	0.971
old	4	1	0.722

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

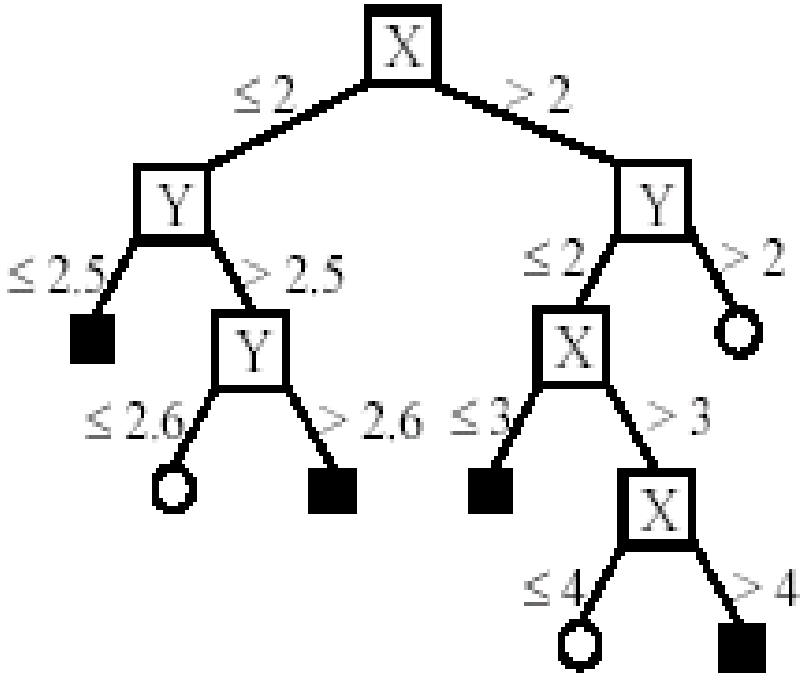
Continuous Attributes

- 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 Attributes

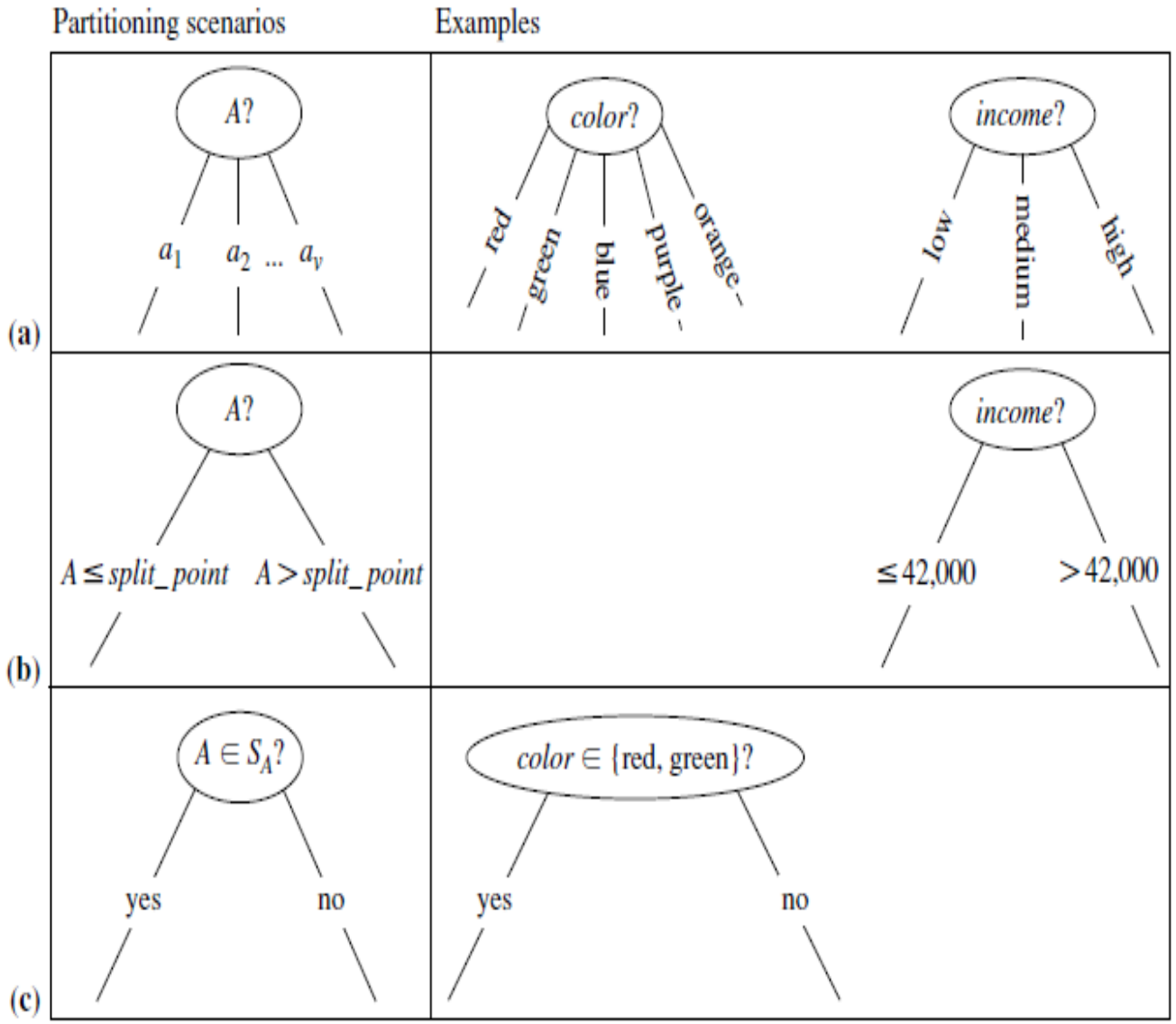


(A) A partition of the data space



(B). The decision tree

Examples


Partitioning scenarios
Examples
(a)
A?
a_1 a_2 ... a_v
color?
red
green
blue
purple
orange
income?
low
medium
high
(b)
A?
$A \leq split\_point$ $A > split\_point$
income?
≤ 42,000
> 42,000
(c)
$A \in S_A$?
yes
no
color ∈ {red, green}?
yes
no

Example

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

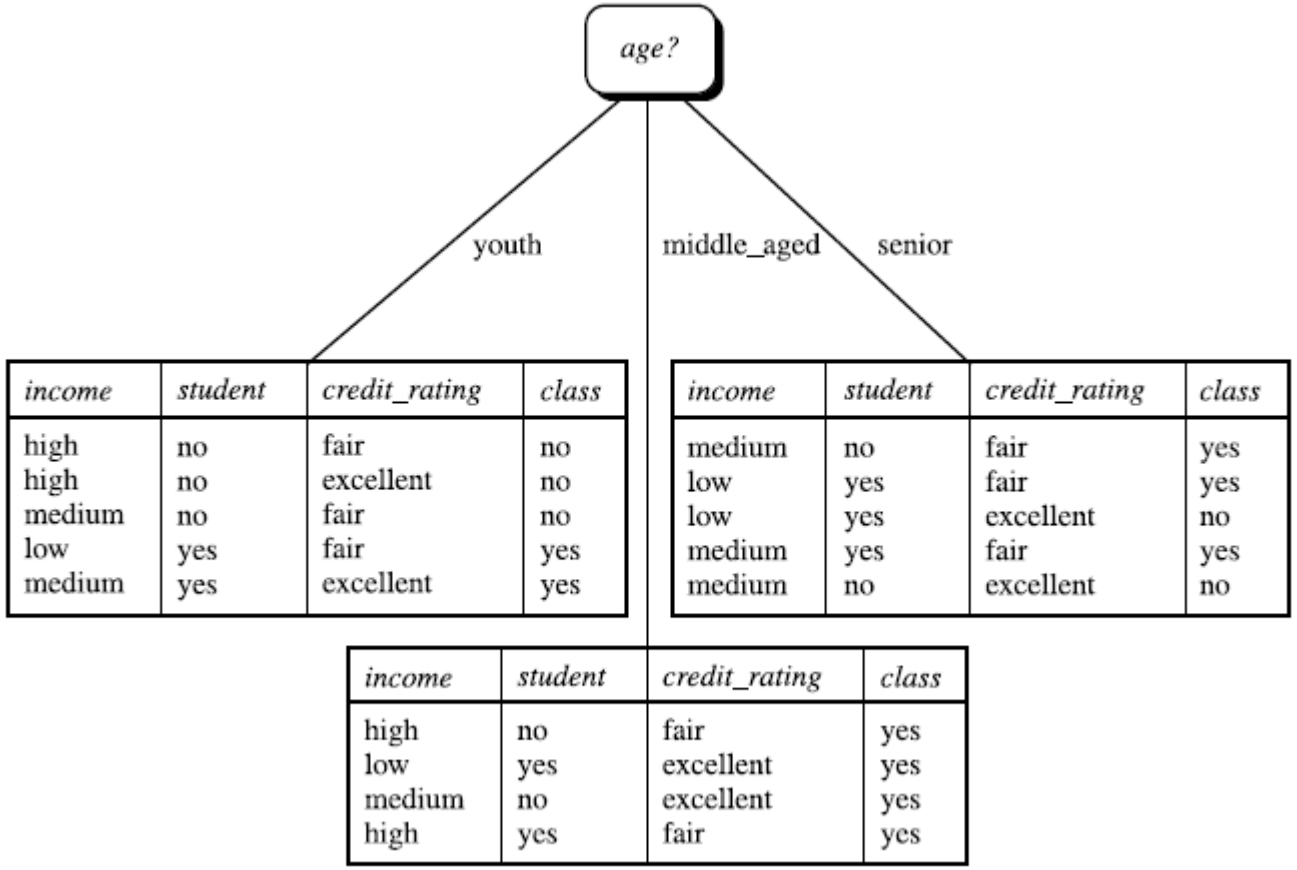
$$\begin{aligned} Info_{age}(D) = {} & \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) \\ & + \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4}\right) \\ & + \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right)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

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

$$Gain(income) = 0.029 \text{ bits} \quad Gain(student) = 0.151 \text{ bits}$$

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

Example



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

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

Gain Ratio

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) 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)

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

- Example

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

- gain_ratio(income) = 0.029/1.557 = 0.019
- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini Index

- If a data set *D* contains examples from *n* classes, gini index, *gini*(*D*) is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class *j* in *D*

- If a data set *D* is split on A into two subsets D_1 and D_2, the *gini* index *gini*(*D*) is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Example

- 9 tuples in buys_computer = “yes” and 5 in “no”
- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

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

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

$$= Gini_{income \in \{high\}}(D).$$

$Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450.

- Split on the {low,medium} (and {high}) since it has the lowest Gini index

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

Comparison

The three measures, in general, return good results but

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
- has difficulty when # of classes is large
- tends to favor tests that result in equal-sized partitions and purity in both partitions

Other Attribute Selection Methods

- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistic: has a close approximation to χ^2 distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- 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

Overfitting

- 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
- **Overfitting is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably"**
- 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"

Underfitting

- Underfitting refers to a model that can neither model the training data nor generalize to new data.

- An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data.

- Underfitting is often not discussed as it is easy to detect given a good performance metric.

Enhancements

- Allow for **continuous-valued attributes**
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle **missing attribute values**
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- **Attribute construction**
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	High	True	***No***

Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	*No*
Sunny	Hot	High	True	*No*
Overcast	Hot	High	False	*Yes*
Rainy	Mild	High	False	*Yes*
Rainy	Cool	Normal	False	*Yes*
Rainy	Cool	Normal	True	*No*
Overcast	Cool	Normal	True	*Yes*
Sunny	Mild	High	False	*No*
Sunny	Cool	Normal	False	*Yes*
Rainy	Mild	Normal	False	*Yes*
Sunny	Mild	Normal	True	*Yes*
Overcast	Mild	High	True	*Yes*
Overcast	Hot	Normal	False	*Yes*
Rainy	Mild	High	True	*No*

$$H(Y) = -\sum_{i=1}^{K} p_k \log_2 p_k$$

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

$$= 0.94$$

Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	*No*
Sunny	Hot	High	True	*No*
Overcast	Hot	High	False	*Yes*
Rainy	Mild	High	False	*Yes*
Rainy	Cool	Normal	False	*Yes*
Rainy	Cool	Normal	True	*No*
Overcast	Cool	Normal	True	*Yes*
Sunny	Mild	High	False	*No*
Sunny	Cool	Normal	False	*Yes*
Rainy	Mild	Normal	False	*Yes*
Sunny	Mild	Normal	True	*Yes*
Overcast	Mild	High	True	*Yes*
Overcast	Hot	Normal	False	*Yes*
Rainy	Mild	High	True	*No*

$$InfoGain(\text{Humidity}) = H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$

$$= 0.94 - \frac{7}{14} H_L - \frac{7}{14} H_R$$

Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	**Normal**	False	***Yes***
Rainy	Cool	**Normal**	True	***No***
Overcast	Cool	**Normal**	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	**Normal**	False	***Yes***
Rainy	Mild	**Normal**	False	***Yes***
Sunny	Mild	**Normal**	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	**Normal**	False	***Yes***
Rainy	Mild	High	True	***No***

$$InfoGain(\text{Humidity}) = H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$

$$0.94 - \frac{7}{14} H_L - \frac{7}{14} H_R$$

$$H_L = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7}$$

Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	**High**	False	*No*
Sunny	Hot	**High**	True	*No*
Overcast	Hot	**High**	False	*Yes*
Rainy	Mild	**High**	False	*Yes*
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	**High**	False	*No*
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	**High**	True	*Yes*
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	**High**	True	*No*

$$InfoGain(\text{Humidity}) = H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$

$$0.94 - \frac{7}{14} H_L - \frac{7}{14} H_R$$

$$H_L = -\frac{\mathbf{6}}{7}\log_2\frac{\mathbf{6}}{7} - \frac{\mathbf{1}}{7}\log_2\frac{\mathbf{1}}{7} = 0.592$$

$$H_R = -\frac{\mathbf{3}}{7}\log_2\frac{\mathbf{3}}{7} - \frac{\mathbf{4}}{7}\log_2\frac{\mathbf{4}}{7} = 0.985$$

Example

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	High	True	***No***

$$InfoGain(\text{Humidity}) = H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$

$$0.94 - \frac{7}{14} 0.592 - \frac{7}{14} 0.985$$

$$= 0.94 - 0.296 - 0.4925$$

$$= 0.1515$$

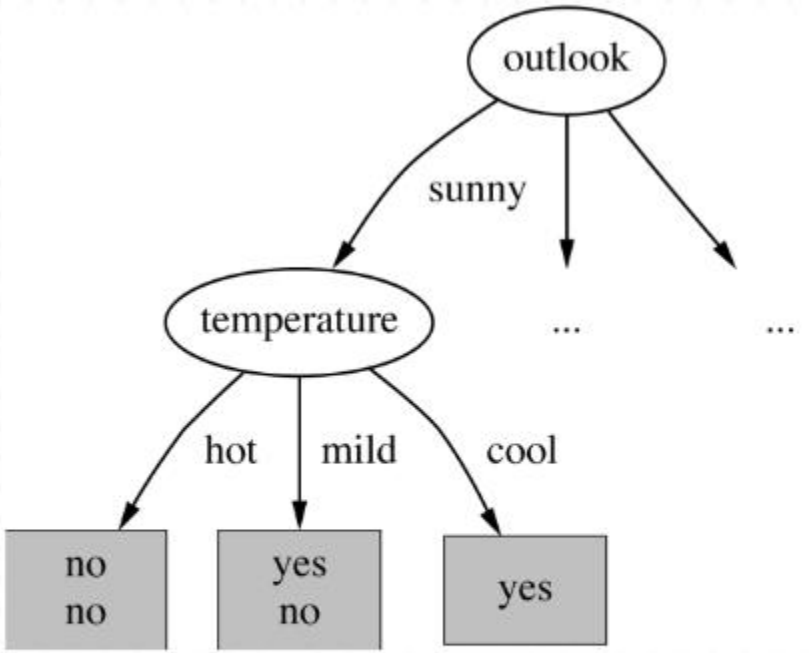
Example

Information gain for each feature:

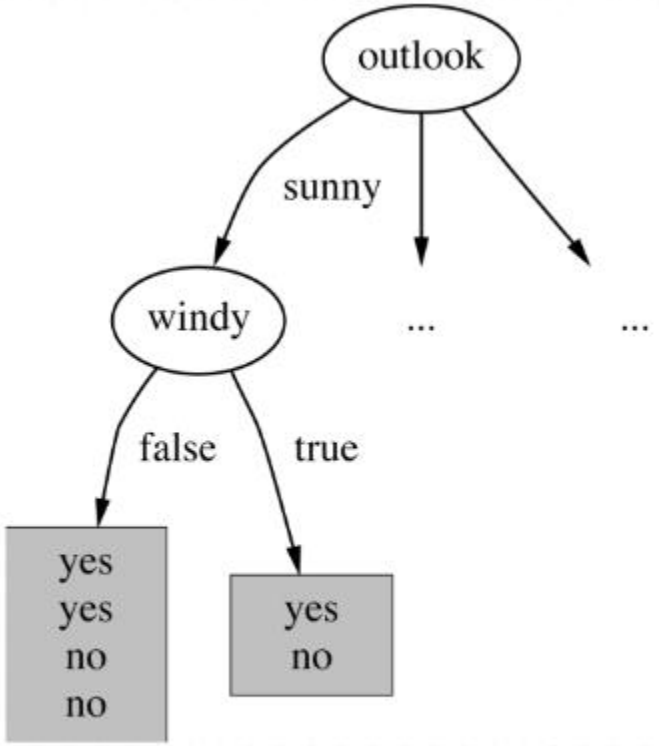
- Outlook = 0.247
- Temperature = 0.029
- Humidity = 0.152
- Windy = 0.048

Initial split is on outlook, because it is the feature with the highest information gain.

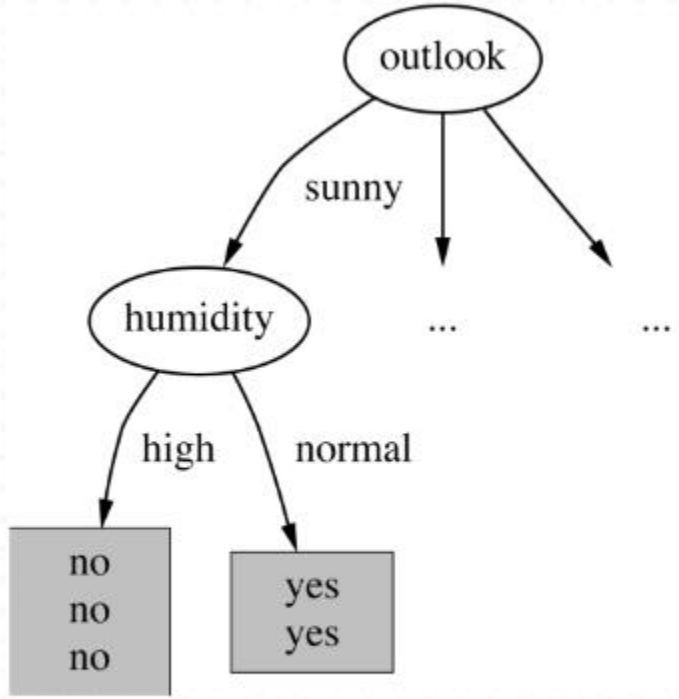
Example



Temperature = 0.571

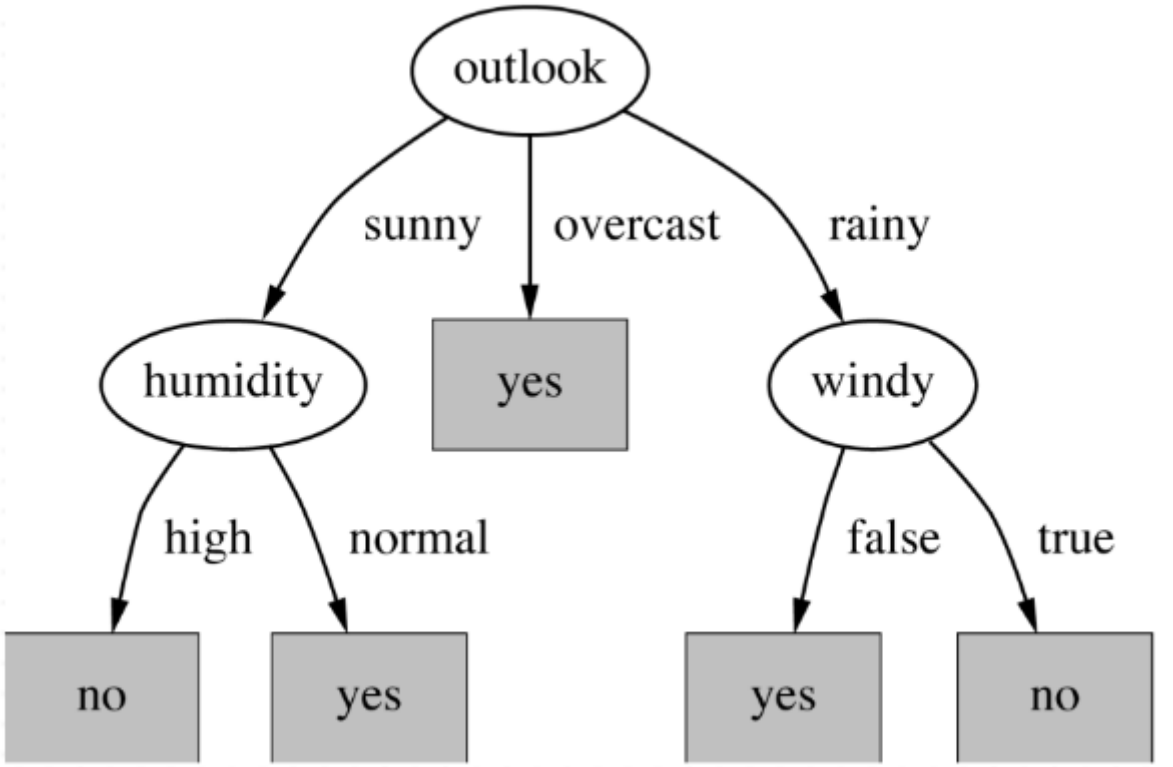


Windy = 0.020



Humidity = 0.971

Example


outlook
sunny
overcast
rainy
humidity
yes
windy
high
normal
false
true
no
yes
yes
no