Chapter X: Classification

Information Retrieval & Data Mining
Universität des Saarlandes, Saarbrücken
Winter Semester 2011/12

Chapter X: Classification*

1. Basic idea
2. Decision trees
3. Naïve Bayes classifier
4. Support vector machines
5. Ensemble methods

* Zaki & Meira: Ch. 24, 26, 28 & 29; Tan, Steinbach & Kumar: Ch. 4, 5.3–5.6

X.1 Basic idea

1. Definitions

1.1. Data

1.2. Classification function

1.3. Predictive vs. descriptive

1.4. Supervised vs. unsupervised

Definitions

- *Data* for classification comes in tuples ($\boldsymbol{x}$, y)
  - Vector $\boldsymbol{x}$ is the **attribute (feature) set**
    - Attributes can be binary, categorical or numerical
  - Value y is the **class label**
    - We concentrate on binary or nominal class labels
    - Compare classification with regression!
- A **classifier** is a function that maps attribute sets to class labels, $f(\boldsymbol{x}) = y$

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

Definitions

- *Data* for classification comes in tuples ($\boldsymbol{x}$, y)
  - Vector $\boldsymbol{x}$ is the **attribute (feature) set**
    - Attributes can be binary, categorical or numerical
  - Value y is the **class label**
    - We concentrate on binary or nominal class labels
    - Compare classification with regression!
- A **classifier** is a function that maps attribute sets to class labels, $f(\boldsymbol{x}) = y$

attribute set

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

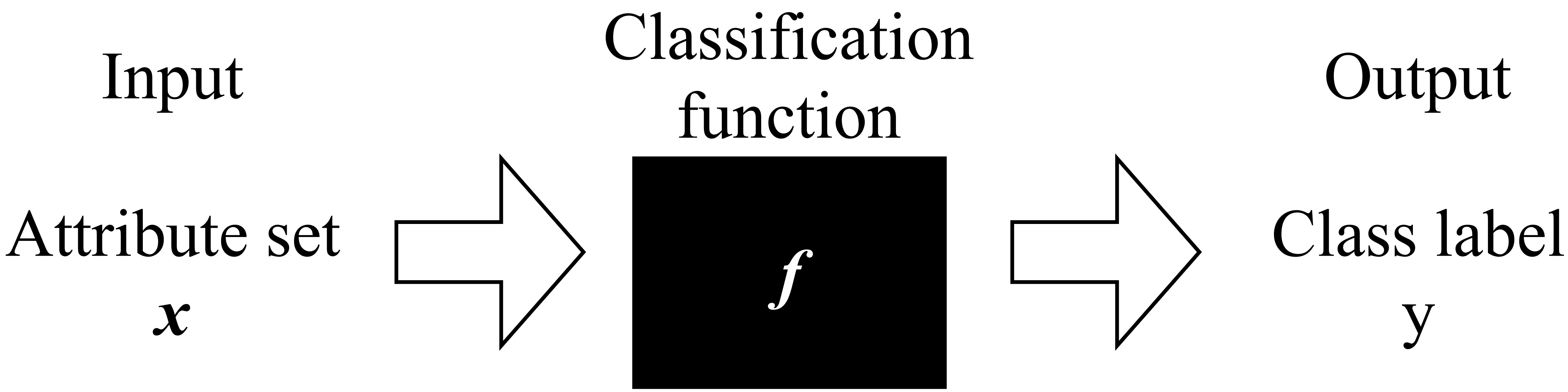
Definitions

- *Data* for classification comes in tuples ($\boldsymbol{x}$, y)
  - Vector $\boldsymbol{x}$ is the **attribute (feature) set**
    - Attributes can be binary, categorical or numerical
  - Value y is the **class label**
    - We concentrate on binary or nominal class labels
    - Compare classification with regression!
- A **classifier** is a function that maps attribute sets to class labels, $f(\boldsymbol{x}) = y$

class

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

Classification function as a black box



Descriptive vs. predictive

- In **descriptive** data mining the goal is to give a description of the data
  - Those who have bought diapers have also bought beer
  - These are the clusters of documents from this corpus
- In **predictive** data mining the goal is to predict the future
  - Those who will buy diapers will also buy beer
  - If new documents arrive, they will be similar to one of the cluster centroids
- The difference between predictive data mining and machine learning is hard to define

Descriptive vs. predictive classification

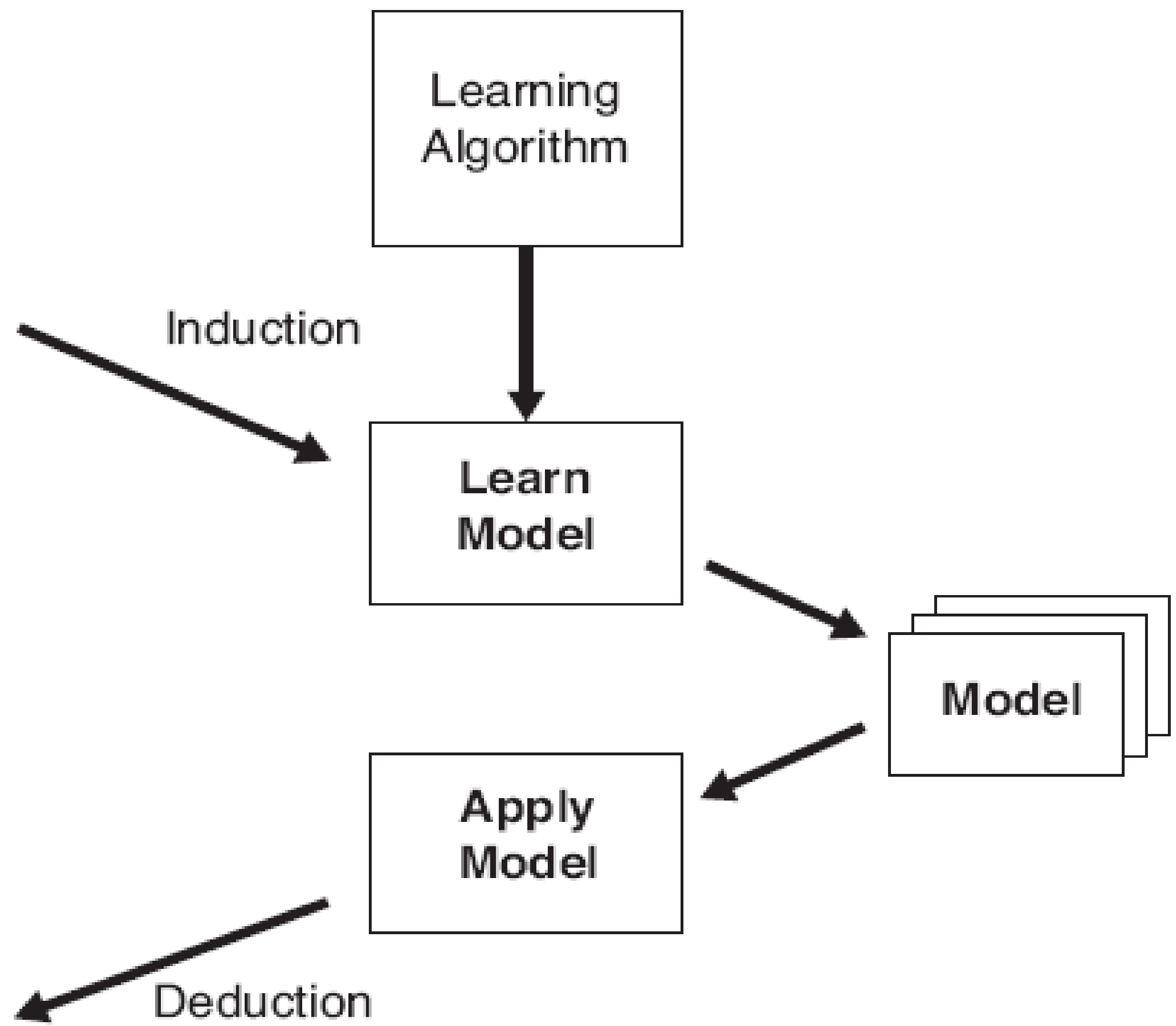
- Who are the borrowers that will default?
  - Descriptive
- If a new borrower comes, will they default?
  - Predictive
- Predictive classification is the usual application
  - What we will concentrate on

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

General classification framework

Training Set

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



Test 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	?

Classification model evaluation

- Recall the *confusion matrix*:

		Predicted class	
		Class = 1	Class = 0
Actual class	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

- Much the same measures as with IR methods
  - Focus on *accuracy* and *error rate*

$$\text{Accuracy} = \frac{f_{11} + f_{00}}{f_{11} + f_{00} + f_{10} + f_{01}}$$

$$\text{Error rate} = \frac{f_{10} + f_{01}}{f_{11} + f_{00} + f_{10} + f_{01}}$$

  - But also precision, recall, F-scores, …

Supervised vs. unsupervised learning

- In **supervised learning**
  - Training data is accompanied by class labels
  - New data is classified based on the training set
    - Classification
- In **unsupervised learning**
  - The class labels are unknown
  - The aim is to establish the existence of classes in the data based on measurements, observations, etc.
    - Clustering

X.2 Decision trees

1. Basic idea
2. Hunt's algorithm
3. Selecting the split
4. Combatting overfitting

Zaki & Meira: Ch. 24; Tan, Steinbach & Kumar: Ch. 4

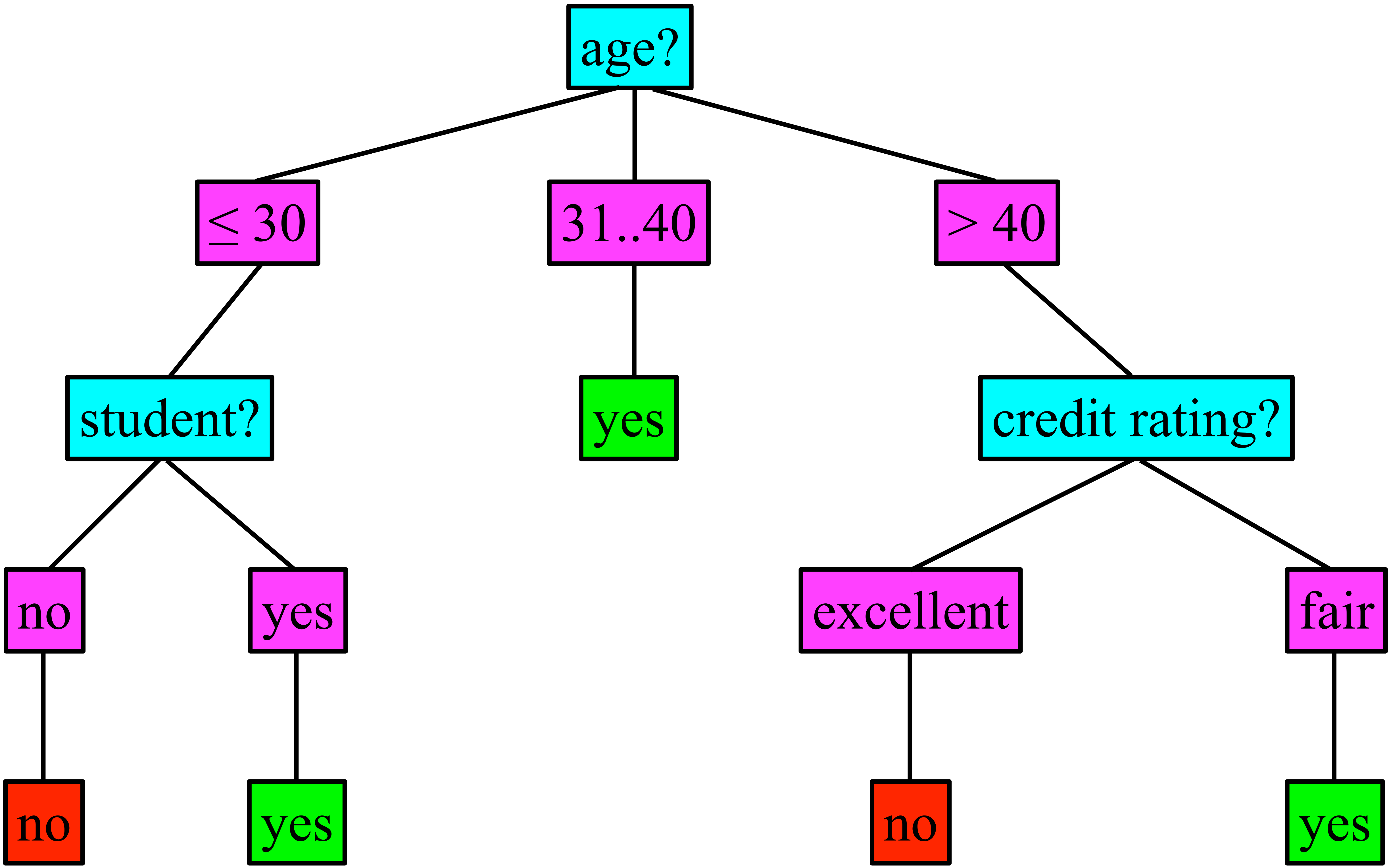
Basic idea

- We define the label by asking series of questions about the attributes
  - Each question depends on the answer to the previous one
  - Ultimately, all samples with satisfying attribute values have the same label and we're done
- The flow-chart of the questions can be drawn as a tree
- We can classify new instances by following the proper edges of the tree until we meet a leaf
  - Decision tree leafs are always class labels

Example: training data

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

Example: decision tree



Hunt’s algorithm

- The number of decision trees for a given set of attributes is exponential
- Finding the the most accurate tree is NP-hard
- Practical algorithms use *greedy heuristics*
  - The decision tree is grown by making a series of locally optimum decisions on which attributes to use
- Most algorithms are based on Hunt’s algorithm

Hunt's algorithm

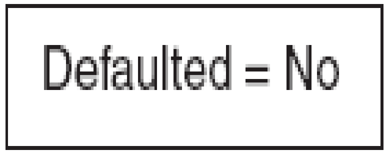
- Let X_t be the set of training records for node t
- Let $y = \{y_1, \ldots y_c\}$ be the class labels
- **Step 1**: If all records in X_t belong to the same class y_t, then t is a leaf node labeled as y_t
- **Step 2:** If X_t contains records that belong to more than one class
  - Select *attribute test condition* to partition the records into smaller subsets
  - Create a *child node* for each outcome of test condition
  - Apply algorithm recursively to each child

Example decision tree construction

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

Example decision tree construction

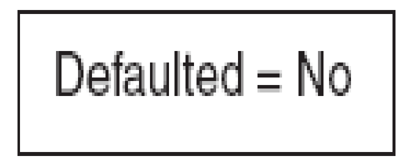
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



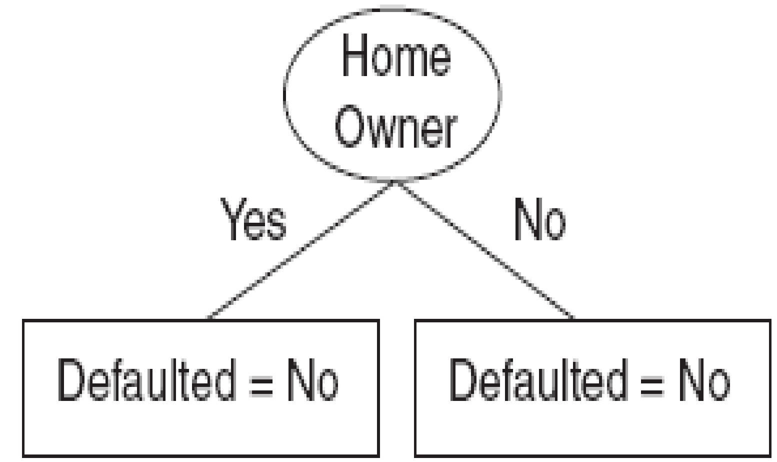
Has multiple labels

Example decision tree construction

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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



Has multiple labels

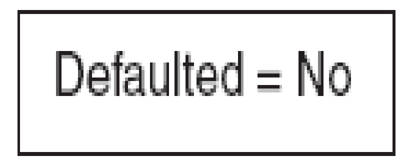


Only one label

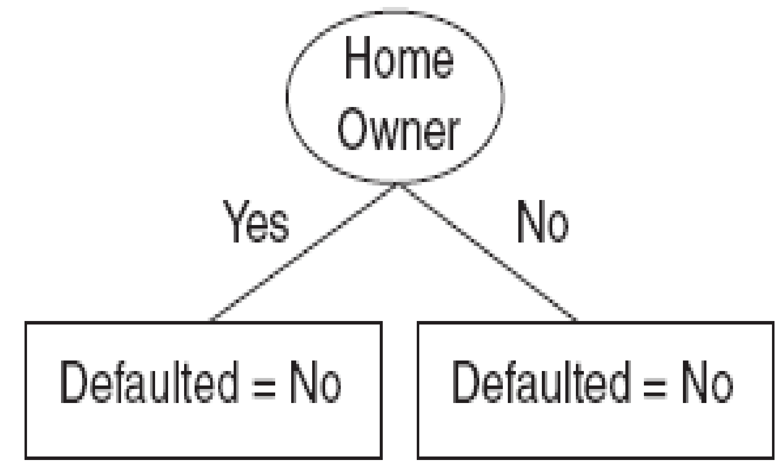
Has multiple labels

Example decision tree construction

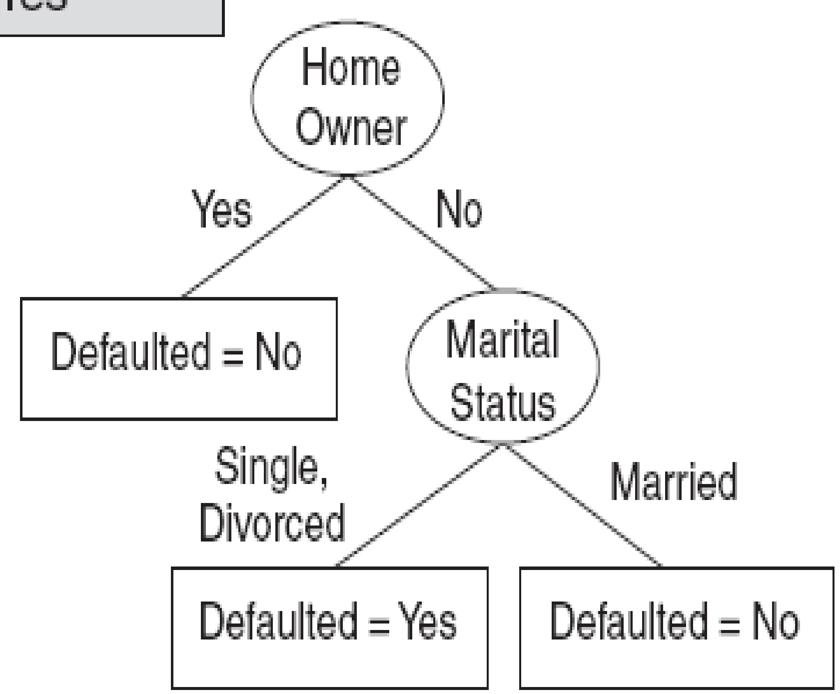
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Has multiple labels



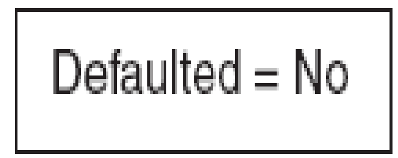
Only one label Has multiple labels



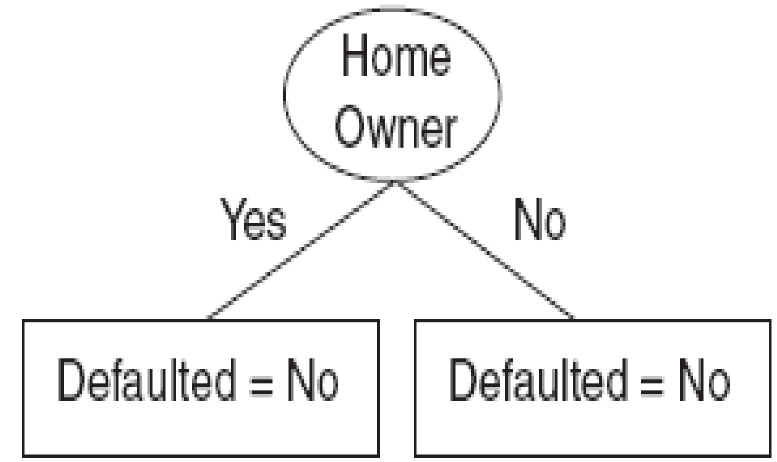
Has multiple labels Only one label

Example decision tree construction

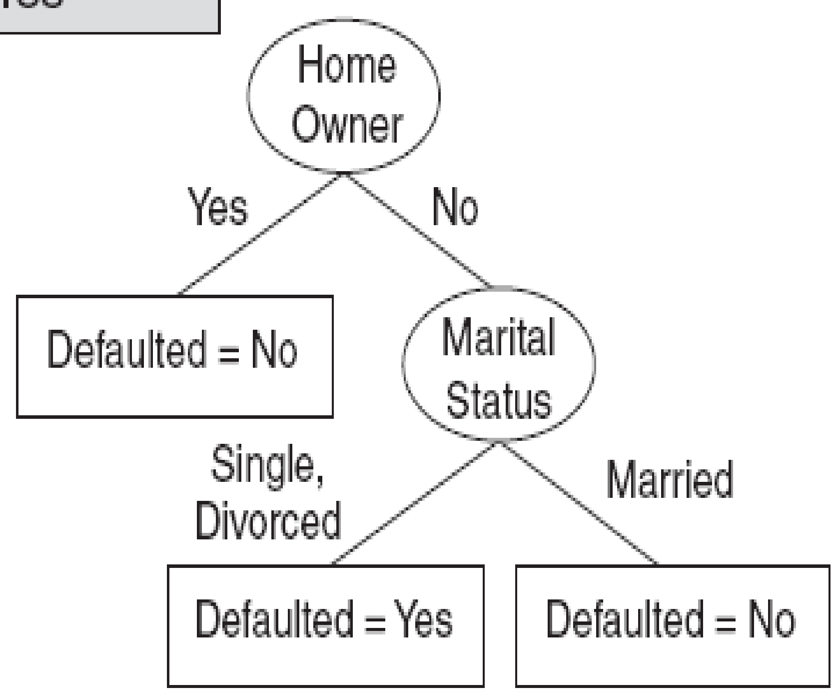
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



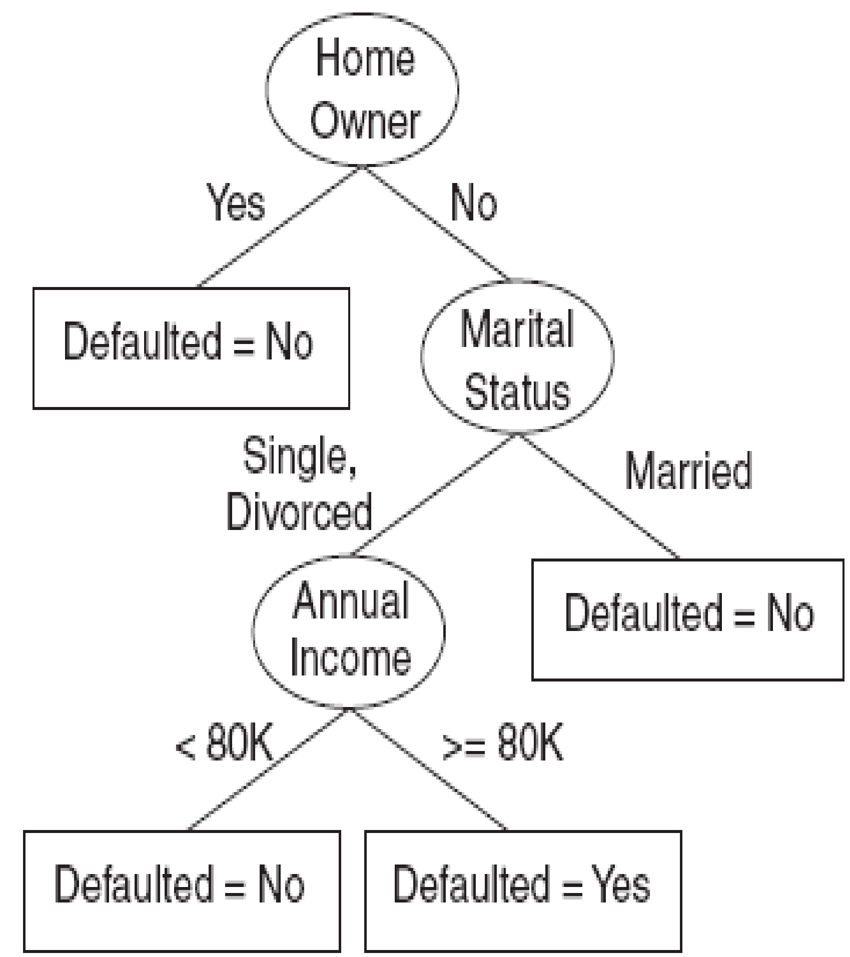
Has multiple labels



Only one label Has multiple labels



Has multiple labels Only one label



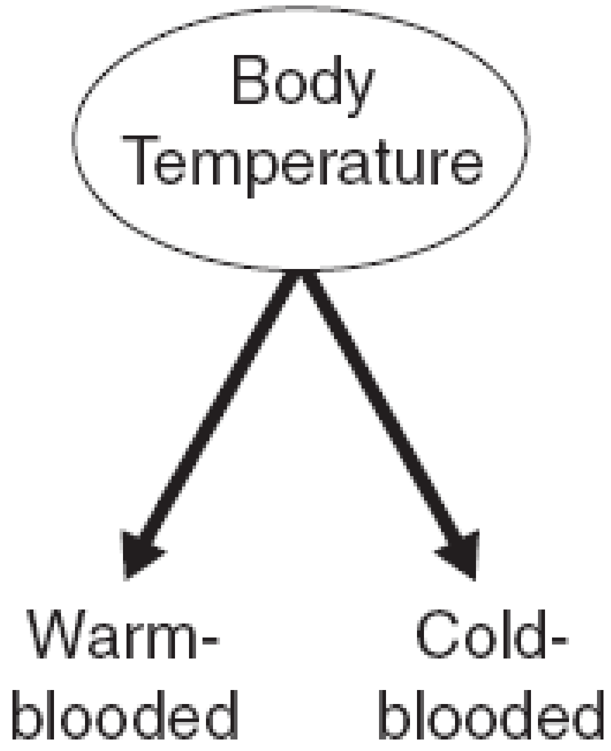
Only one label Only one label

Selecting the split

- Designing a decision-tree algorithm requires answering two questions
  1. How should the training records be split?
  2. How should the splitting procedure stop?

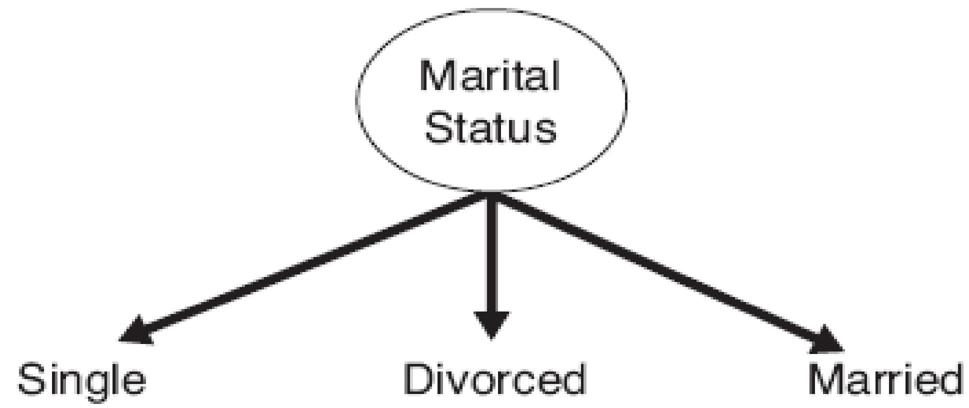
Splitting methods

Binary attributes

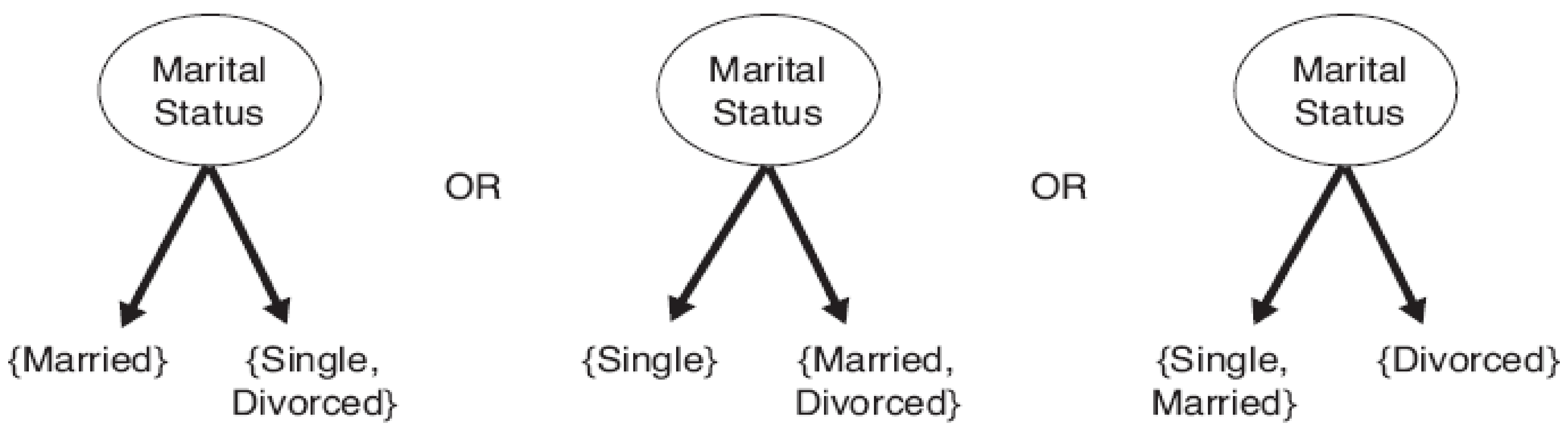


Splitting methods

Nominal attributes



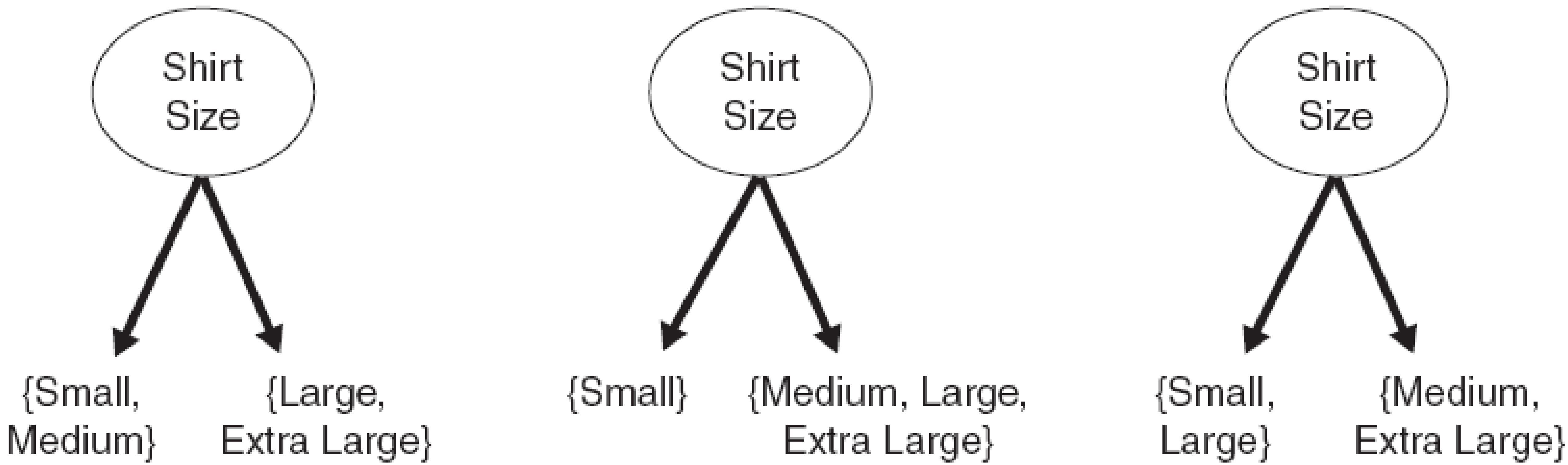
Multiway split



Binary split

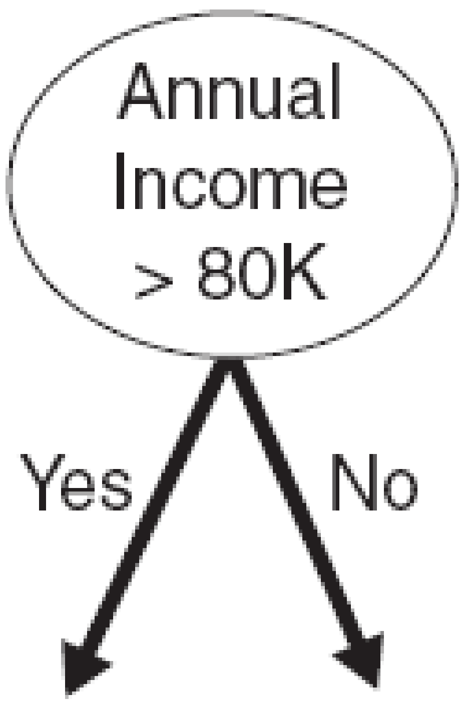
Splitting methods

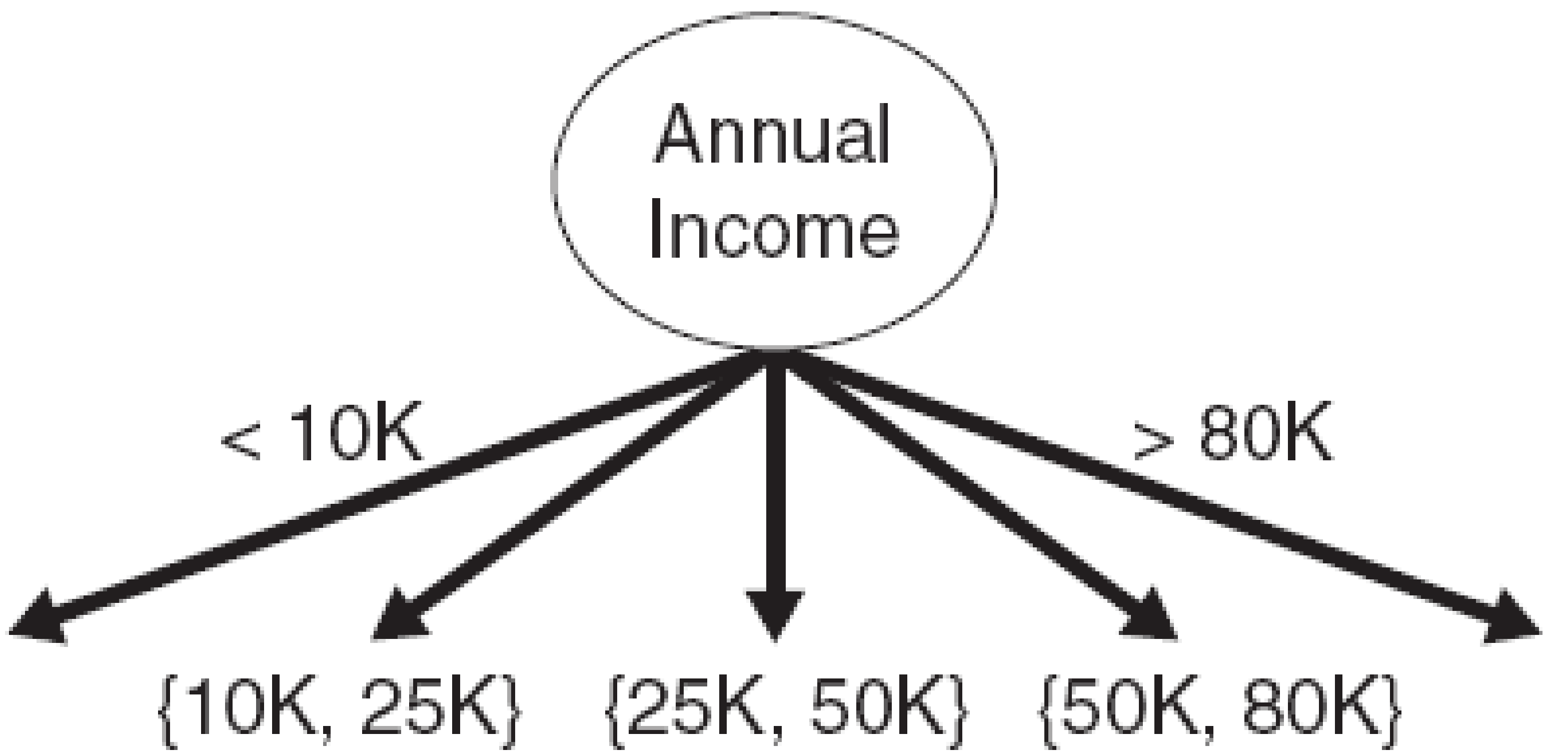
Ordinal attributes



Splitting methods

Continuous attributes

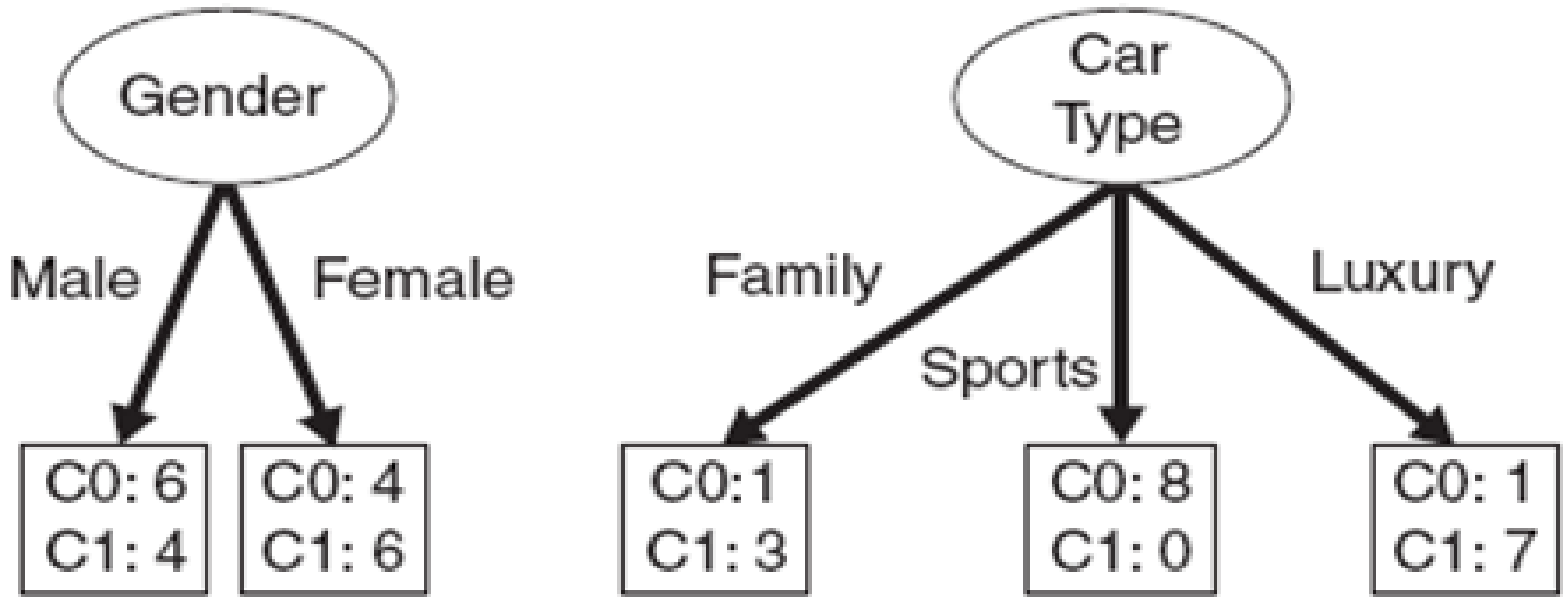




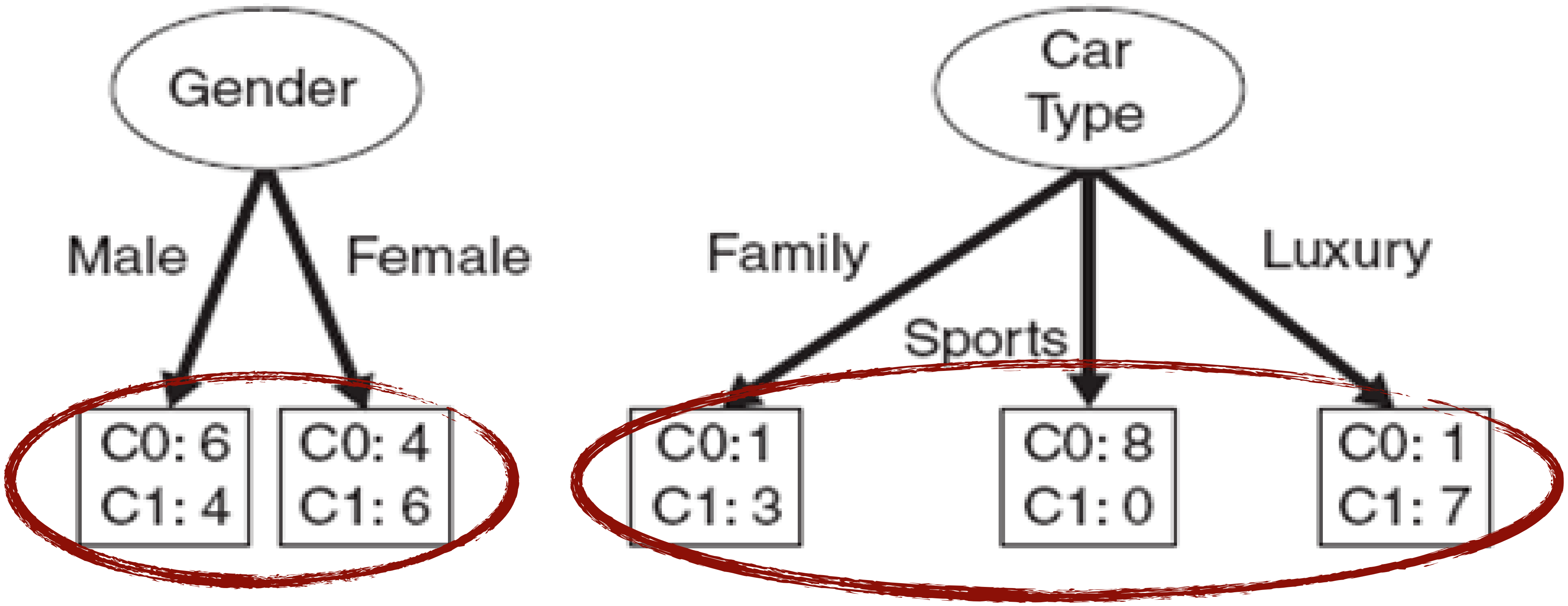
Selecting the best split

- Let $p(i \mid t)$ be the fraction of records belonging to class i at node t
- *Best split* is selected based on the degree of **impurity** of the child nodes
  - $p(0 \mid t) = 0$ and $p(1 \mid t) = 1$ has *high purity*
  - $p(0 \mid t) = 1/2$ and $p(1 \mid t) = 1/2$ has the *smallest purity* (*highest impurity*)
- Intuition: high purity $\Rightarrow$ small value of impurity measures $\Rightarrow$ better split

Example of purity



Example of purity



high impurity

high purity

Impurity measures

$0 \times \log_2(0) = 0$

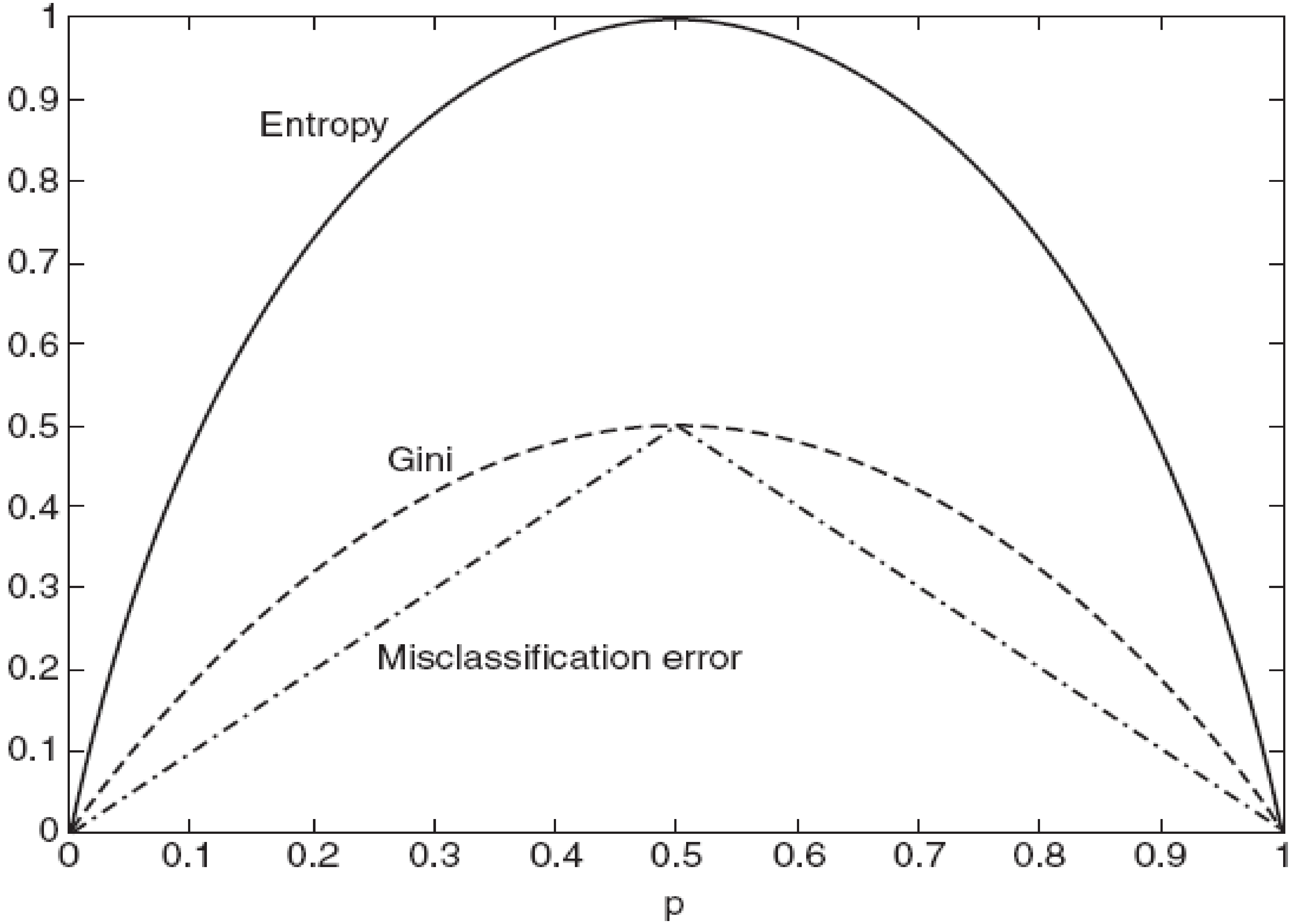
$$\text{Entropy}(t) = -\sum_{i=0}^{c-1} p(i \mid t) \log_2 p(i \mid t)$$

≤ 0

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} \big(p(i \mid t)\big)^2$$

$$\text{Classification error}(t) = 1 - \max_i \{p(i \mid t)\}$$

Comparing impurity measures



Comparing conditions

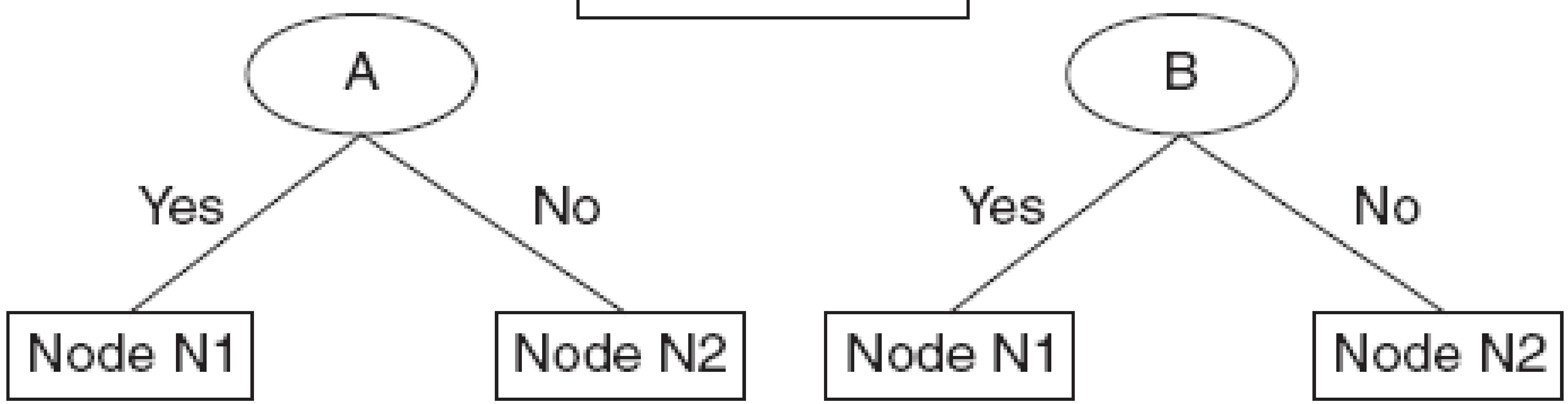
- The quality of the split: the change in the impurity
  - Called the **gain** of the test condition

$$\Delta = I(p) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

    - $I(\,)$ is the impurity measure
    - k is the number of attribute values
    - p is the parent node, v_j is the child node
    - N is the total number of records at the parent node
    - $N(v_j)$ is the number of records associated with the child node
- Maximizing the gain ⇔ minimizing the weighted average impurity measure of child nodes
- If $I() = \text{Entropy}()$, then $\Delta = \Delta_{\text{info}}$ is called **information gain**

Computing the gain: example

	Parent
C0	6
C1	6
Gini = 0.500	

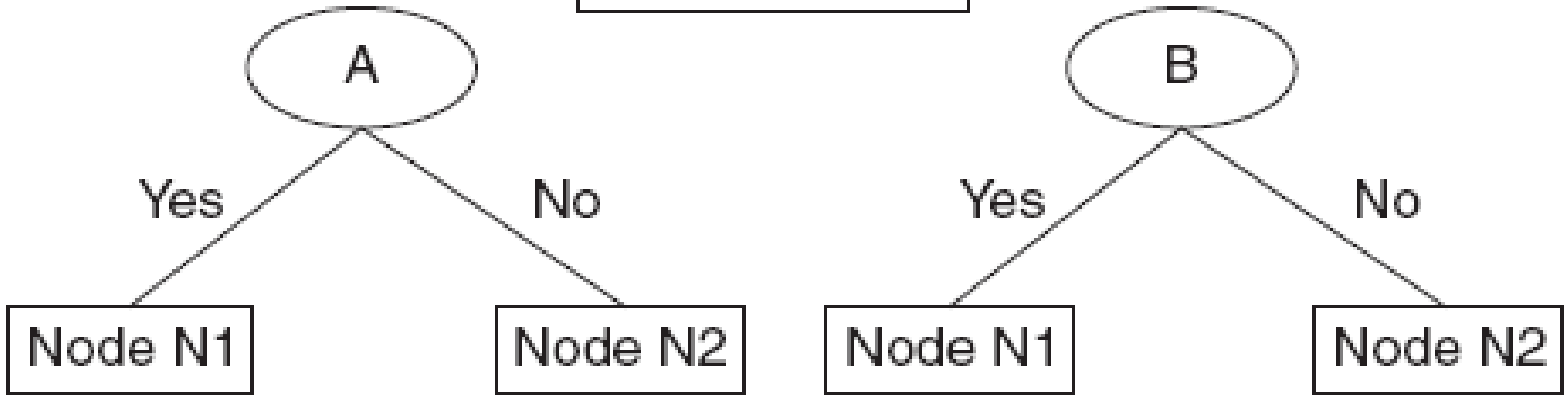


	N1	N2
C0	4	2
C1	3	3
Gini = 0.486		

	N1	N2
C0	1	5
C1	4	2
Gini = 0.375		

Computing the gain: example

	Parent
C0	6
C1	6
Gini = 0.500	



G: 0.4898

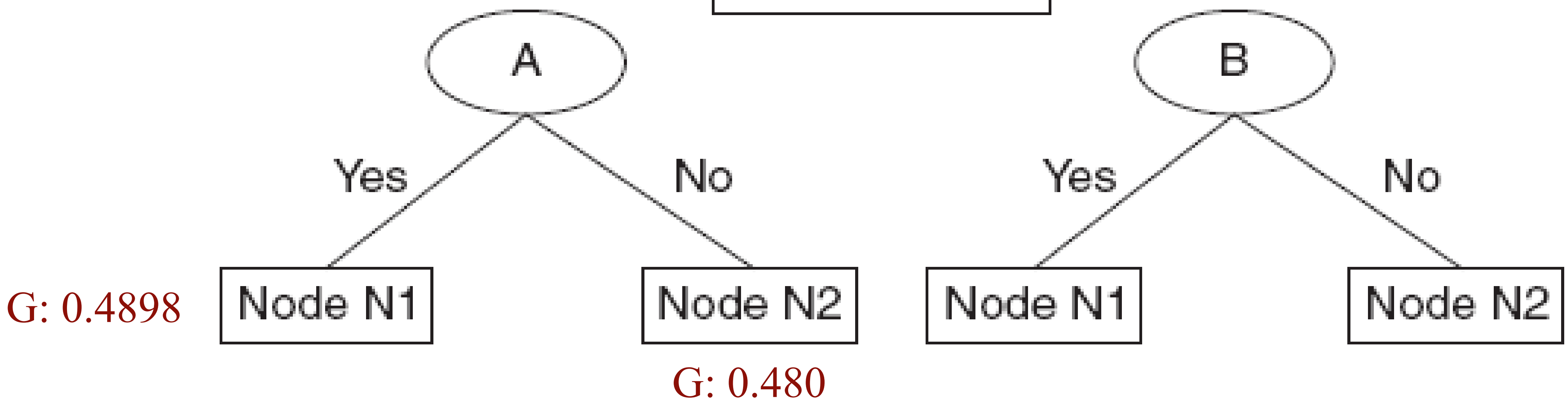
G: 0.480

	N1	N2
C0	4	2
C1	3	3
Gini = 0.486		

	N1	N2
C0	1	5
C1	4	2
Gini = 0.375		

Computing the gain: example

	Parent
C0	6
C1	6
Gini = 0.500	



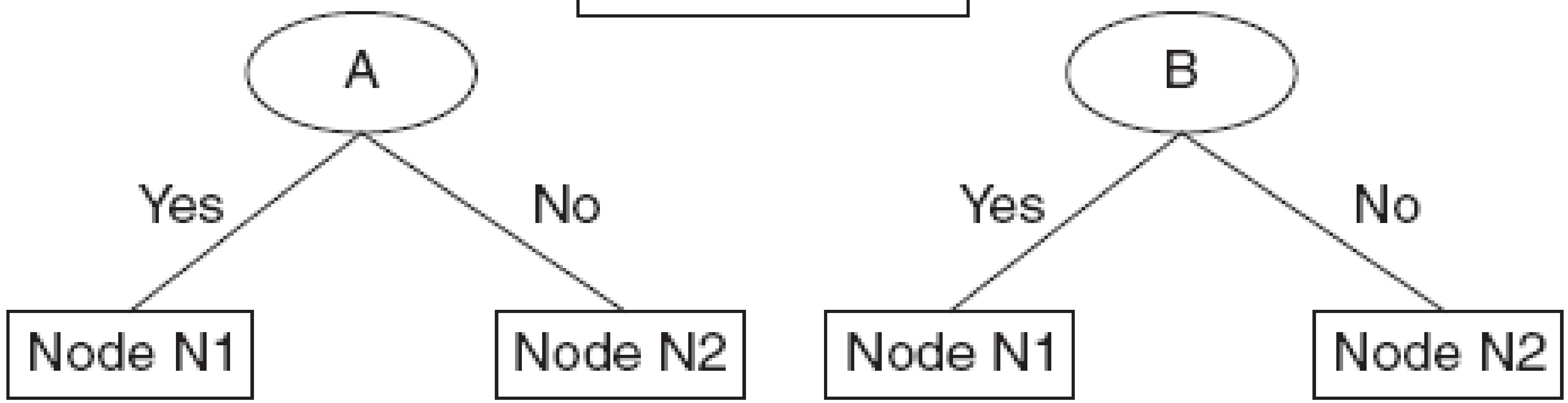
	N1	N2
C0	4	2
C1	3	3
Gini = 0.486		

7

	N1	N2
C0	1	5
C1	4	2
Gini = 0.375		

Computing the gain: example

	Parent
C0	6
C1	6
Gini = 0.500	



G: 0.4898

G: 0.480

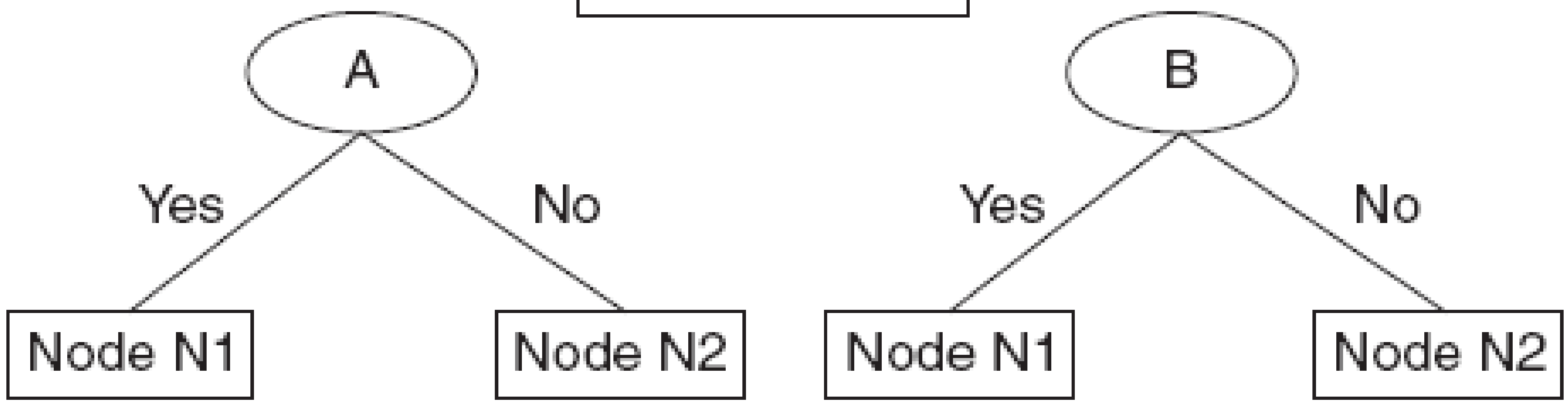
	N1	N2
C0	4	2
C1	3	3
Gini = 0.486		

	N1	N2
C0	1	5
C1	4	2
Gini = 0.375		

7 5

Computing the gain: example

	Parent
C0	6
C1	6
Gini = 0.500	



G: 0.4898

G: 0.480

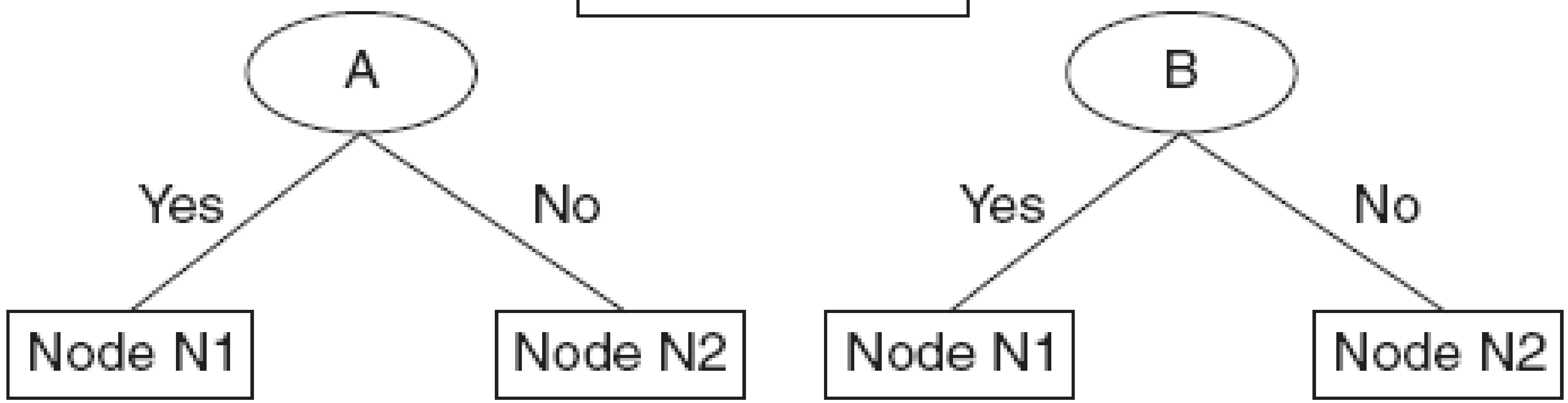
	N1	N2
C0	4	2
C1	3	3
Gini = 0.486		

	N1	N2
C0	1	5
C1	4	2
Gini = 0.375		

7 5

Computing the gain: example

	Parent
C0	6
C1	6
Gini = 0.500	



G: 0.4898

G: 0.480

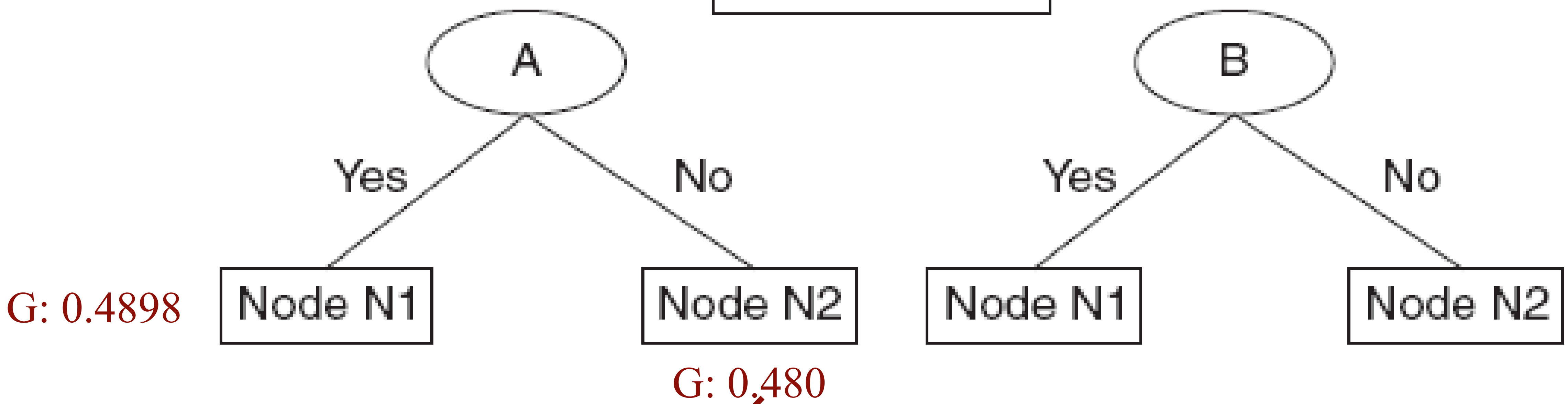
	N1	N2
C0	4	2
C1	3	3
Gini = 0.486		

	N1	N2
C0	1	5
C1	4	2
Gini = 0.375		

$7 \times 0.4898 + 5$

Computing the gain: example

	Parent
C0	6
C1	6
Gini = 0.500	



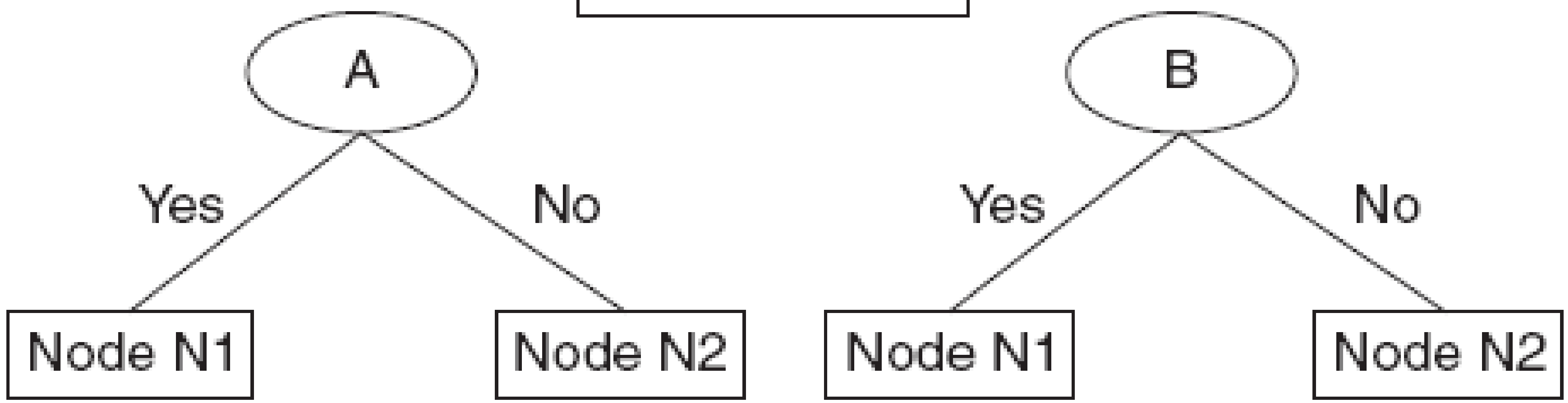
	N1	N2
C0	4	2
C1	3	3
Gini = 0.486		

	N1	N2
C0	1	5
C1	4	2
Gini = 0.375		

$7 \times 0.4898 + 5 \times 0.480$

Computing the gain: example

	Parent
C0	6
C1	6
Gini = 0.500	



G: 0.4898

G: 0.480

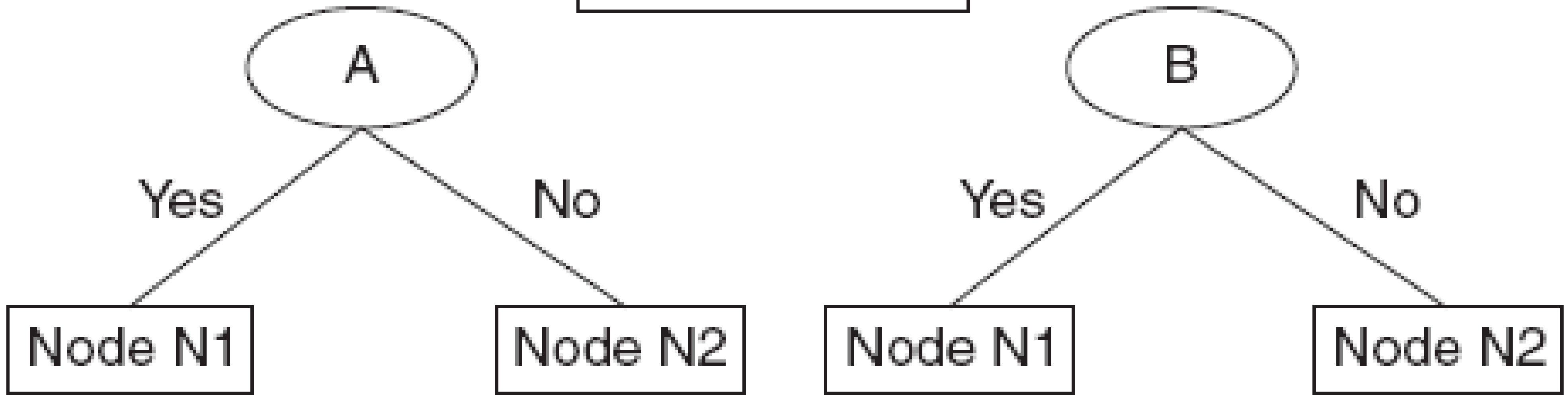
	N1	N2
C0	4	2
C1	3	3
Gini = 0.486		

	N1	N2
C0	1	5
C1	4	2
Gini = 0.375		

$7 \times 0.4898 + 5 \times 0.480$

Computing the gain: example

	Parent
C0	6
C1	6
Gini = 0.500	



G: 0.4898

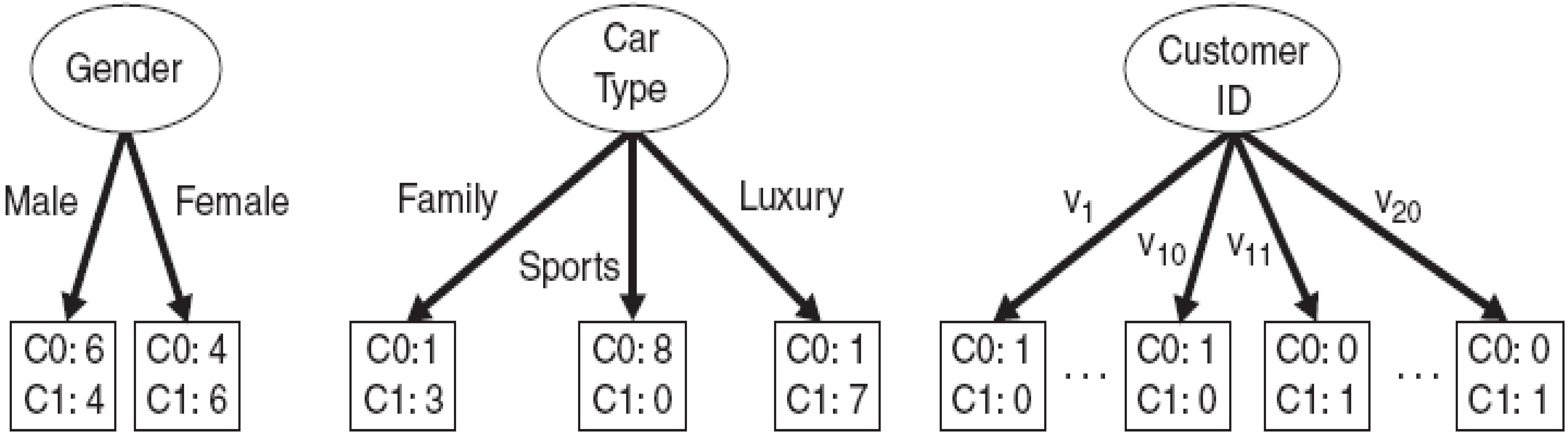
G: 0.480

	N1	N2
C0	4	2
C1	3	3
Gini = 0.486		

	N1	N2
C0	1	5
C1	4	2
Gini = 0.375		

$(7 \times 0.4898 + 5 \times 0.480) / 12 = 0.486$

Problems of maximizing Δ



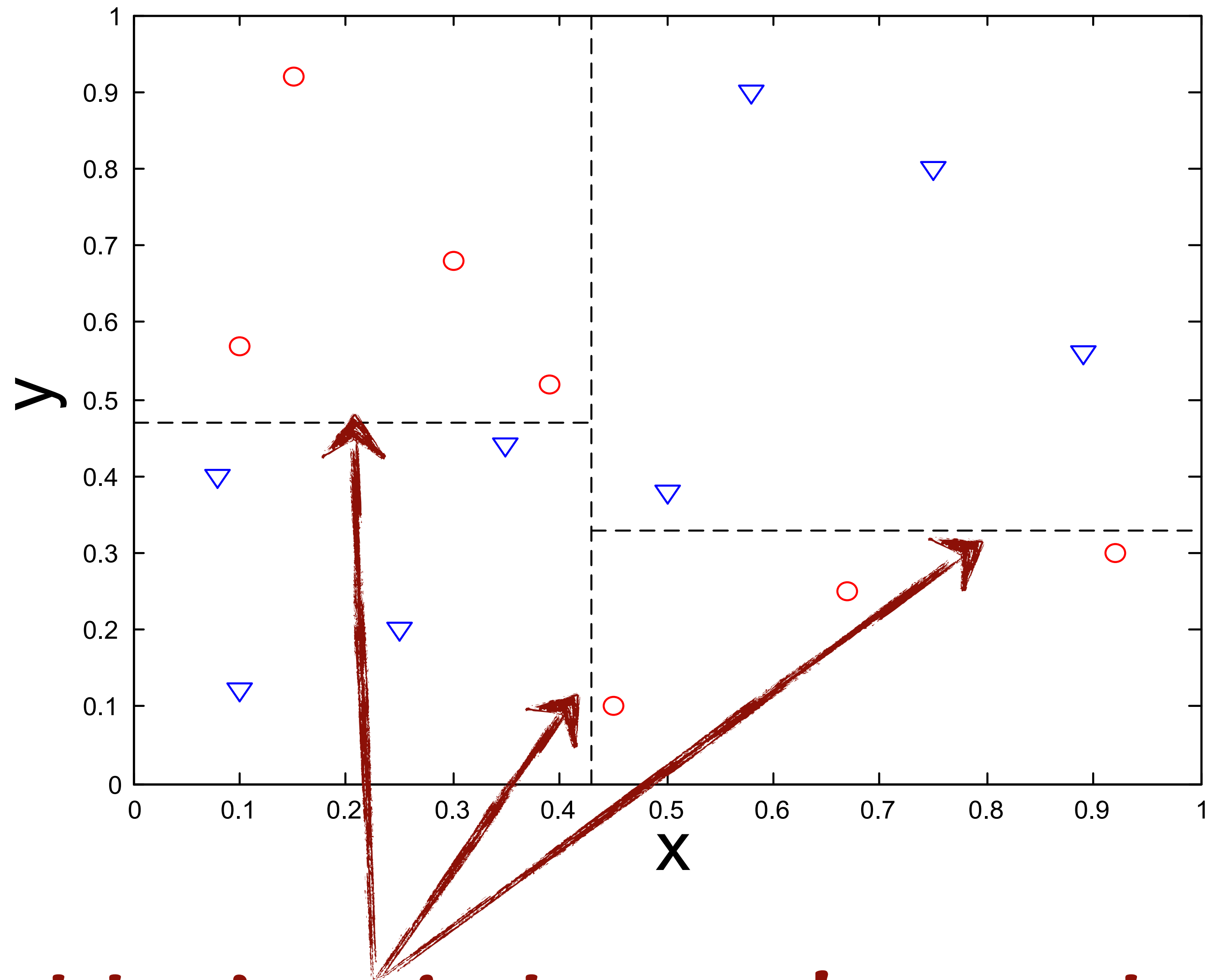
Problems of maximizing Δ

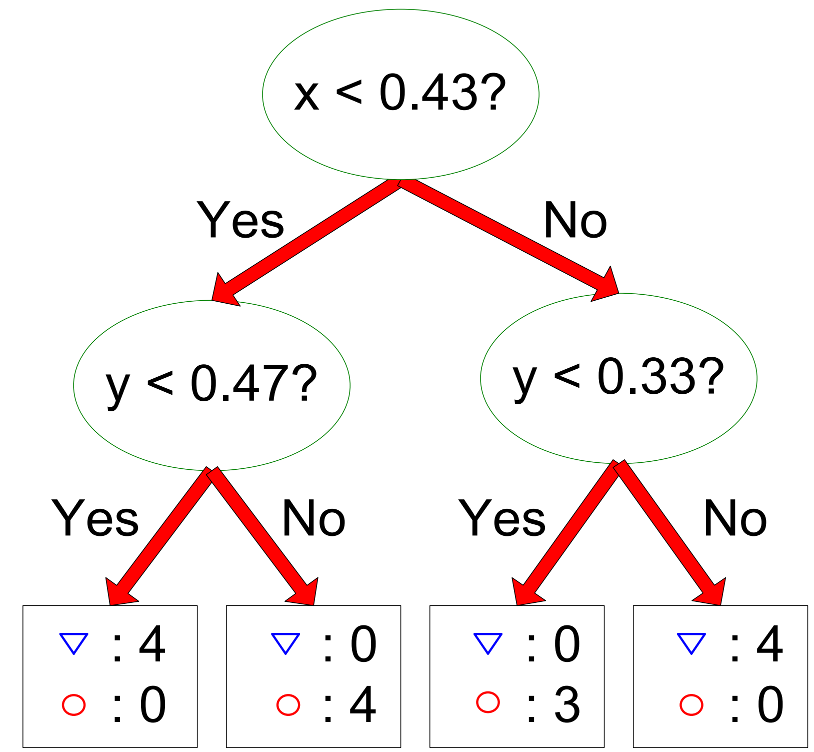
- Impurity measures favor attributes with large number of values
- A test condition with large number of outcomes might not be desirable
  - Number of records in each partition is too small to make predictions
- Solution 1: **gain ratio** = Δ_{info} / SplitInfo
  - $\text{SplitInfo} = -\sum_{i=1}^{k} P(v_i) \log_2(P(v_i))$
    - $P(v_i)$ = the fraction of records at child; k = total number of splits
  - Used e.g. in C4.5
- Solution 2: restrict the splits to binary

Stopping the splitting

- Stop expanding when all records belong to the same class
- Stop expanding when all records have similar attribute values
- Early termination
  - E.g. gain ratio drops below certain threshold
  - Keeps trees simple
  - Helps with overfitting

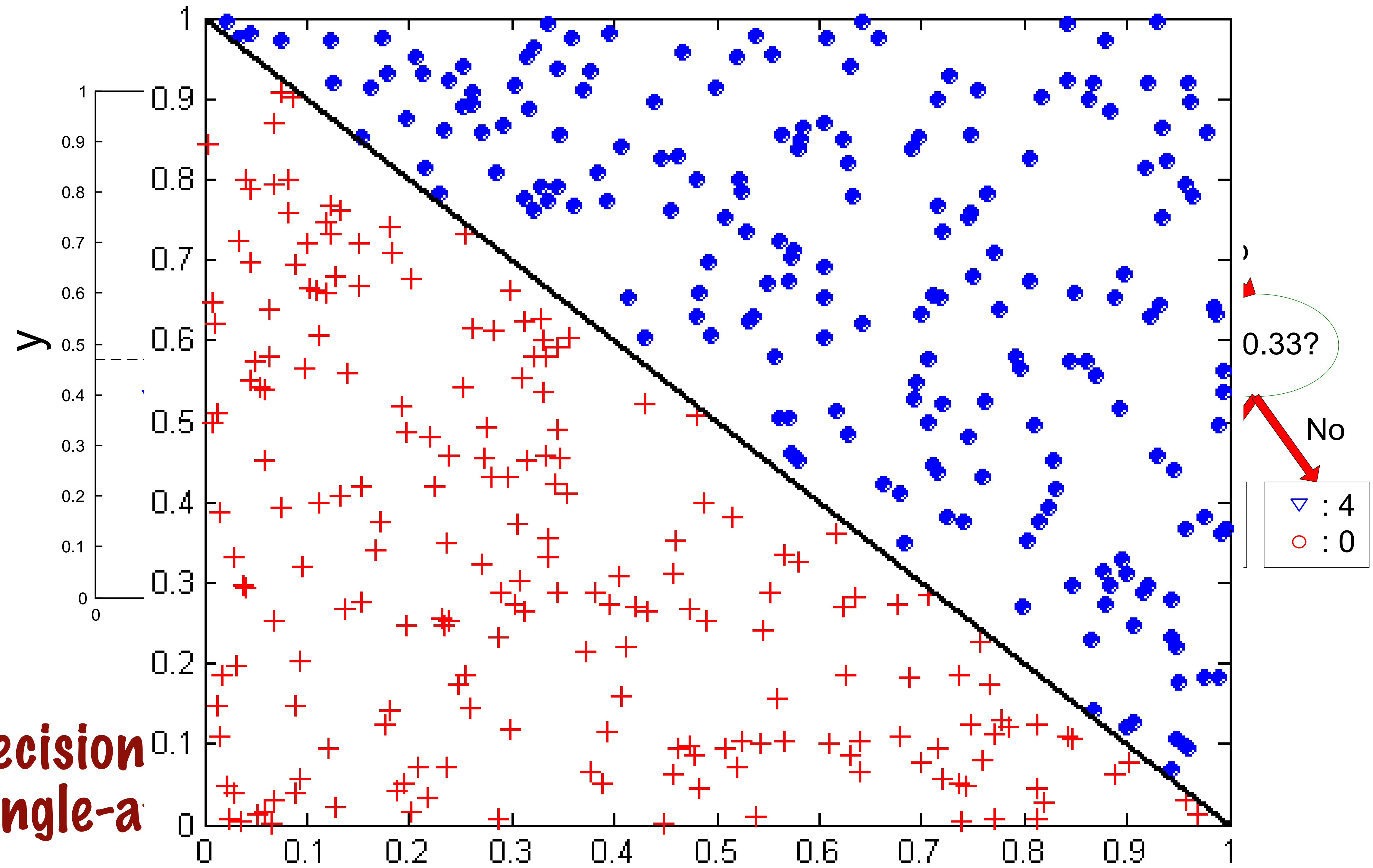
Geometry of single-attribute splits





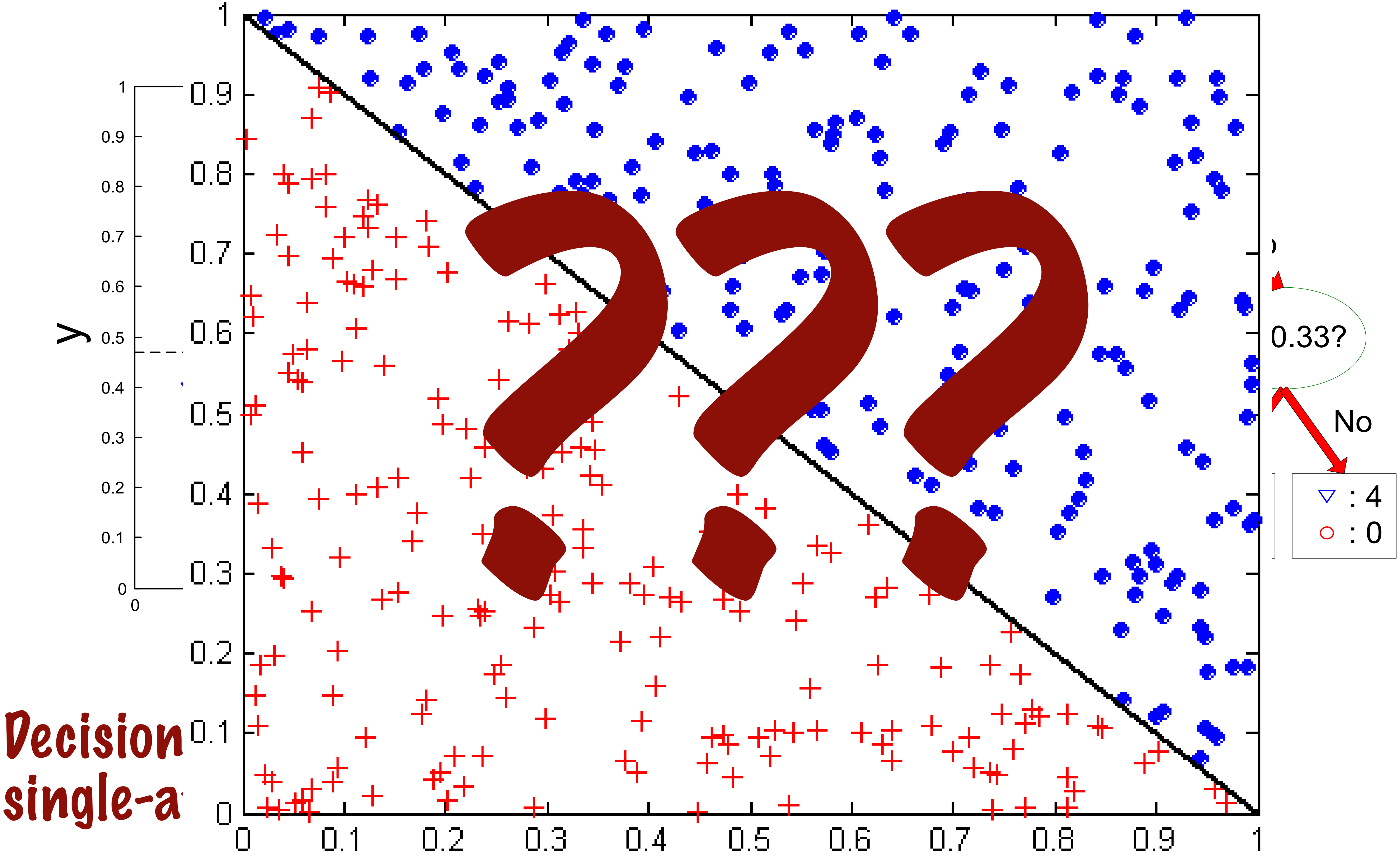
Decision boundaries are always axis-parallel for single-attribute splits

Geometry of single-attribute splits



Decision
single-a

Geometry of single-attribute splits



Combatting overfitting

- Overfitting is a major problem with all classifiers
- As decision trees are parameter-free, we need to stop building the tree before overfitting happens
  - Overfitting makes decision trees overly complex
  - Generalization error will be big
- Let’s measure the generalization error somehow

Estimating the generalization error

- Error on training data is called *re-substitution error*
  - $e(T) = \Sigma e(t) / N$
    - $e(t)$ is the error at leaf node t
    - N is the number of training records
    - $e(T)$ is the error *rate* of the decision tree
- *Generalization error* rate:
  - $e'(T) = \Sigma e'(t) / N$
  - **Optimistic approach**: $e'(T) = e(T)$
  - **Pessimistic approach**: $e'(T) = \Sigma_t(e(t) + \Omega)/N$
    - Ω is a *penalty term*
- Or we can use testing data

Handling overfitting

- In **pre-pruning** we stop building the decision tree when some early stopping criterion is satisfied
- In **post-pruning** full-grown decision tree is trimmed
  - From bottom to up try replacing a decision node with a leaf
  - If generalization error improves, replace the sub-tree with a leaf
    - New leaf node's class label is the majority of the sub-tree
  - We can also use *minimum description length* principle

Minimum description principle (MDL)

- The complexity of a data is made of two parts
  - The complexity of explaining a model for data
  - The complexity of explaining the data given the model
  - $L = L(M) + L(D \mid M)$
- *The model that minimizes L is the optimum for this data*
  - This is the minimum description length principle
  - Computing the least number of bits to produce a data is its *Kolmogorov complexity*
    - Uncomputable!
  - MDL approximates Kolmogorov complexity

MDL and classification

- The model is the classifier (decision tree)
- Given the classifier, we need to tell where it errs
- Then we need a way to encode the classifier and its error
  - Per MDL principle, the better the encoder, the better the results
  - The art of creating good encoders is in the heart of using MDL

Summary of decision trees

- Fast to build
- Extremely fast to use
  - Small ones are easy to interpret
    - Good for domain expert's verification
    - Used e.g. in medicine
- Redundant attributes are not (much of) a problem
- Single-attribute splits cause axis-parallel decision boundaries
- Requires post-pruning to avoid overfitting