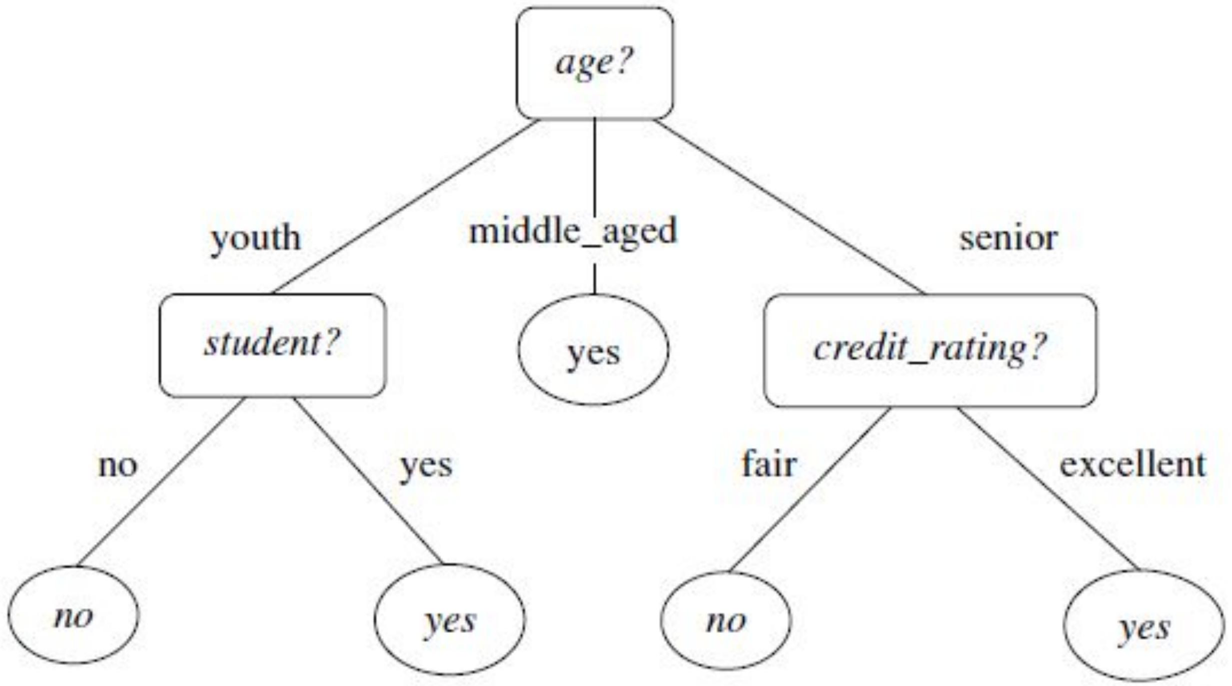
Decision Trees

Decision Tree Learning

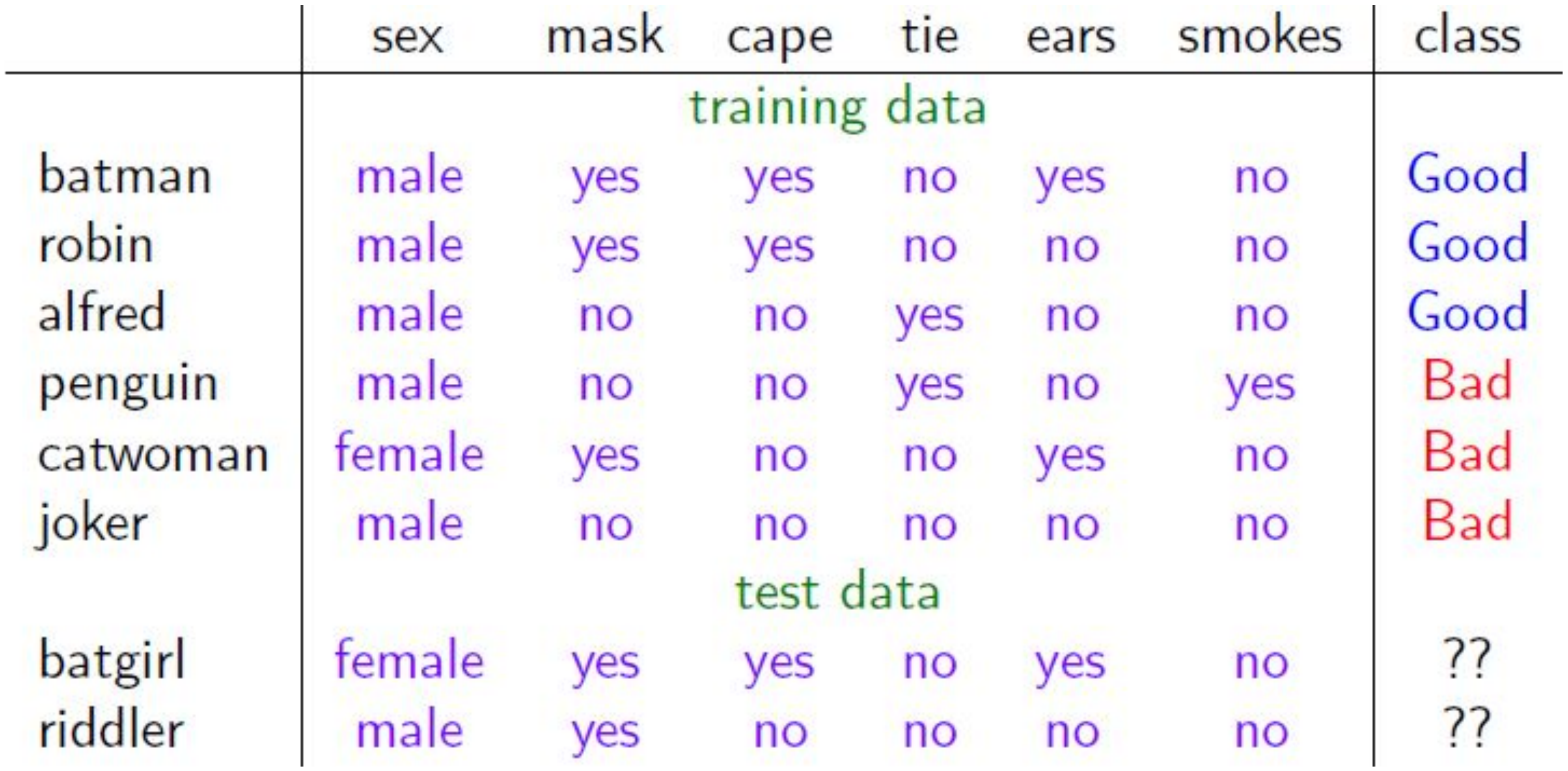
- **Decision tree induction is the learning of decision trees from class-labeled training** tuples.
- A **decision tree is a flowchart-like tree structure, where each internal node (nonleaf** node) denotes a test on an attribute, each **branch represents an outcome of the** test, and each **leaf node (or *terminal node) holds a class label.***
- ***The topmost node in*** a tree is the **root node.**

A decision tree for the concept *buys_computer, indicating whether an AllElectronics customer* is likely to purchase a computer. Each internal (nonleaf) node represents a test on an attribute. Each leaf node represents a class (either *buys_computer = yes or buys_computer = no).*



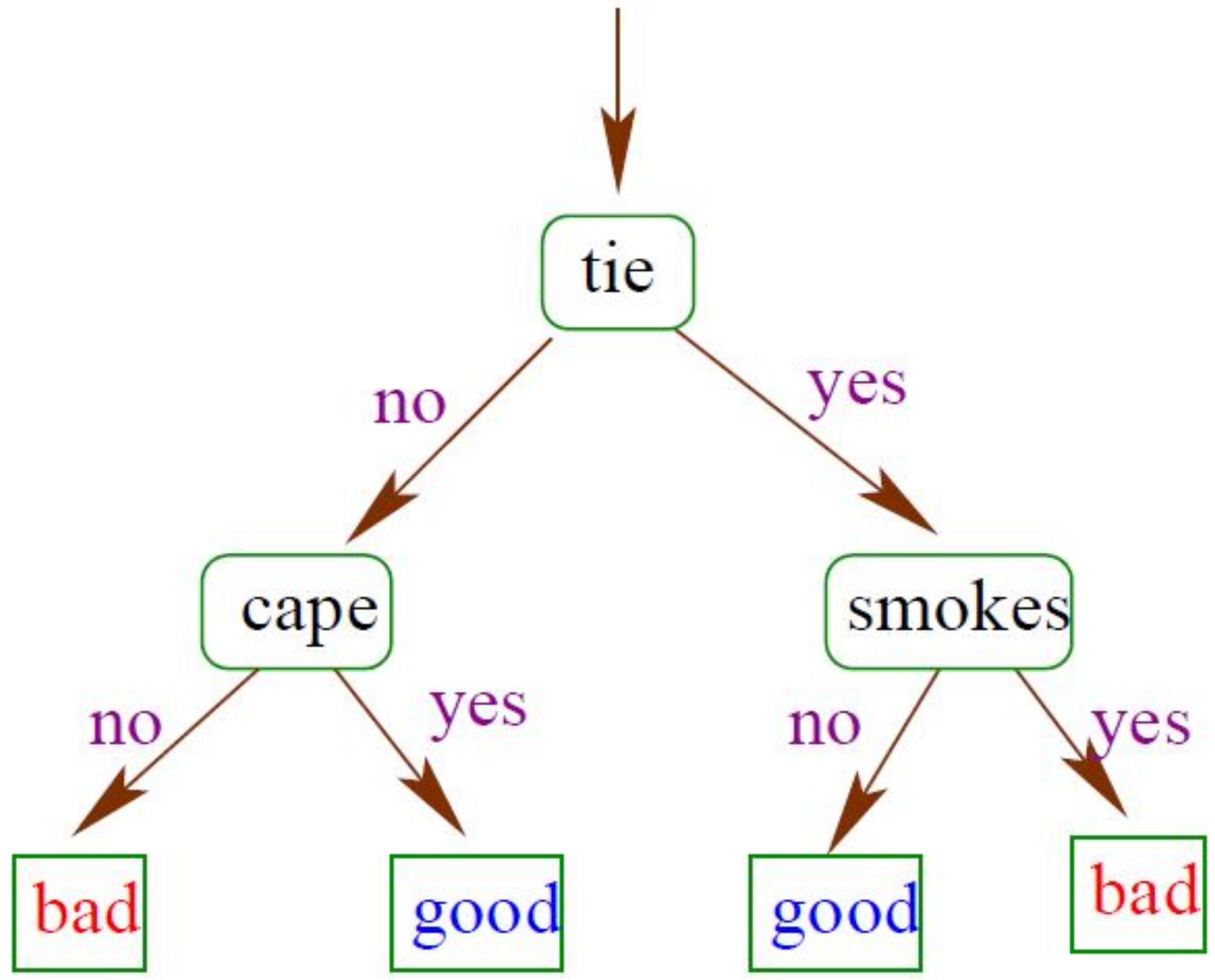
Example: Good versus Evil

- **problem**: identify people as good or bad from their appearance



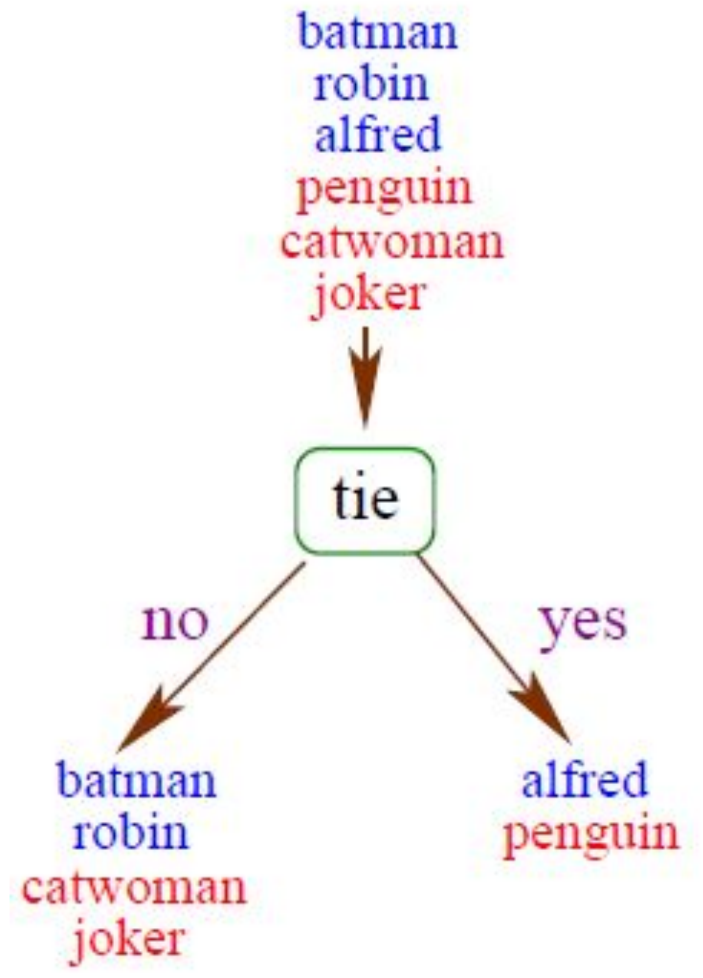
	sex	mask	cape	tie	ears	smokes	class
			training data				
batman	male	yes	yes	no	yes	no	Good
robin	male	yes	yes	no	no	no	Good
alfred	male	no	no	yes	no	no	Good
penguin	male	no	no	yes	no	yes	Bad
catwoman	female	yes	no	no	yes	no	Bad
joker	male	no	no	no	no	no	Bad
			test data				
batgirl	female	yes	yes	no	yes	no	??
riddler	male	yes	no	no	no	no	??

A Decision Tree Classifier



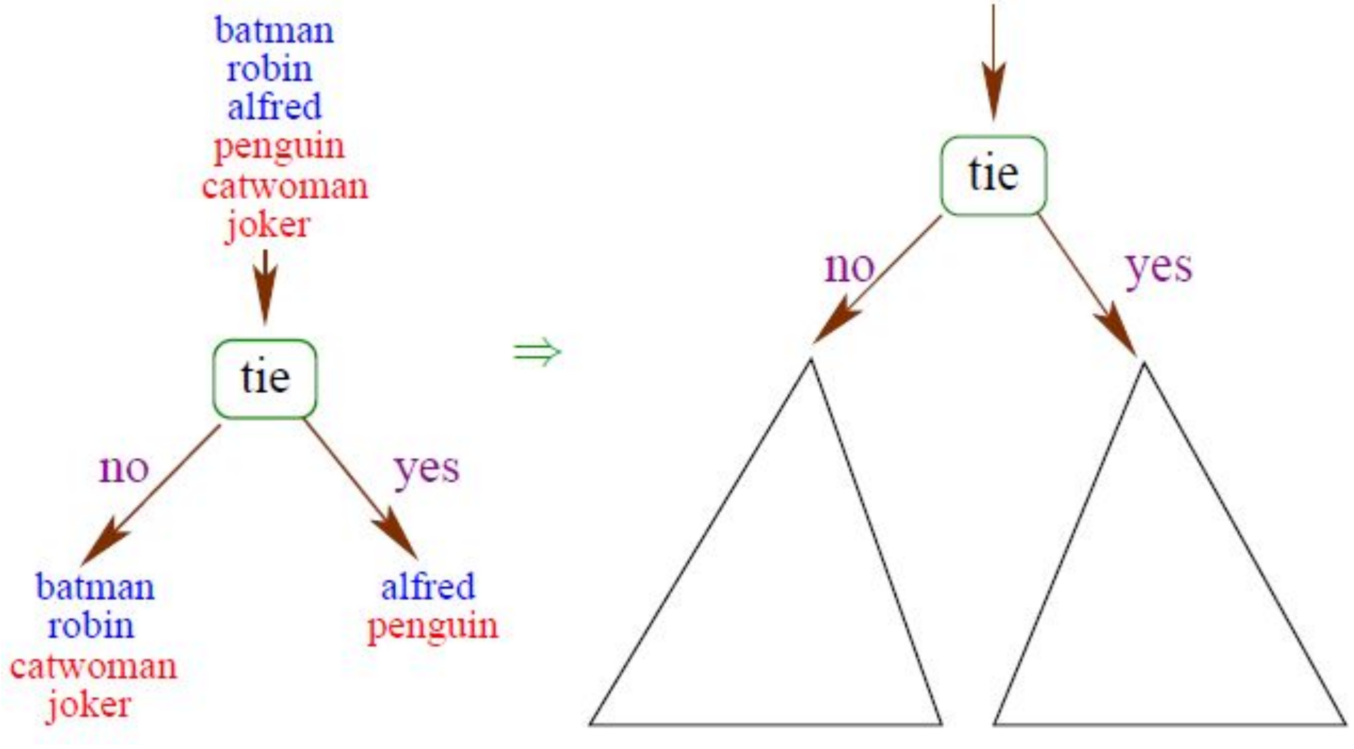
How to Build Decision Trees

- choose rule to split on
- divide data using splitting rule into disjoint subsets



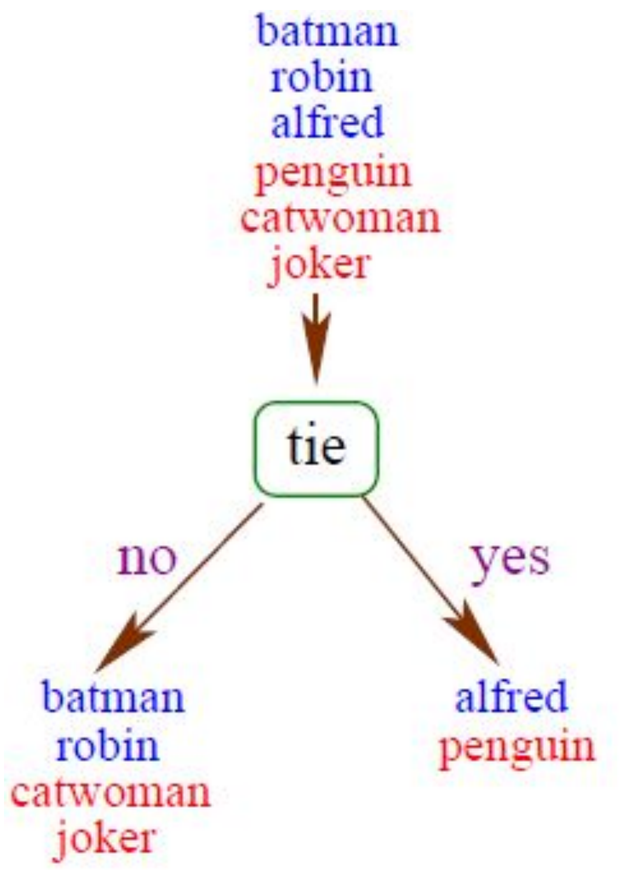
How to Build Decision Trees

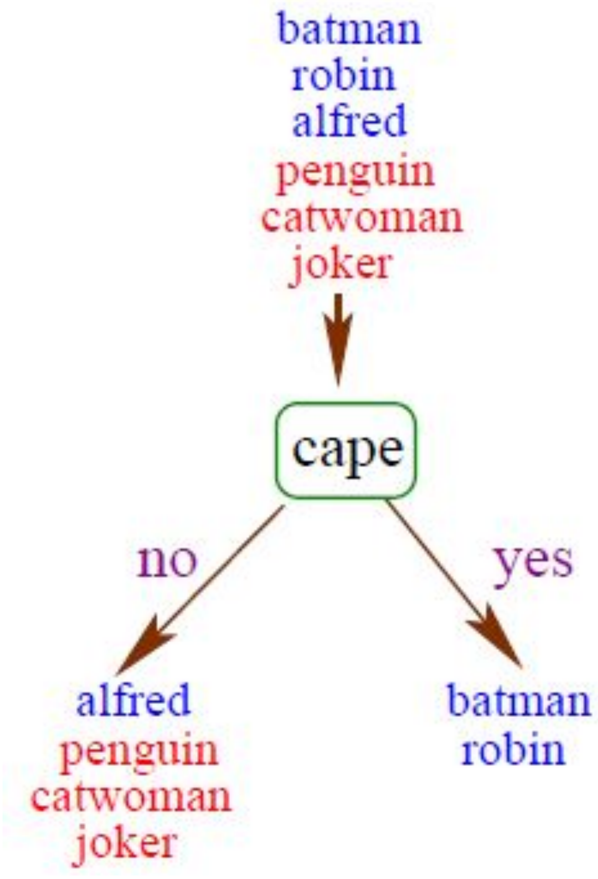
- choose rule to split on
- divide data using splitting rule into disjoint subsets
- repeat recursively for each subset
- stop when leaves are (almost) "pure"



How to Choose the Splitting Rule

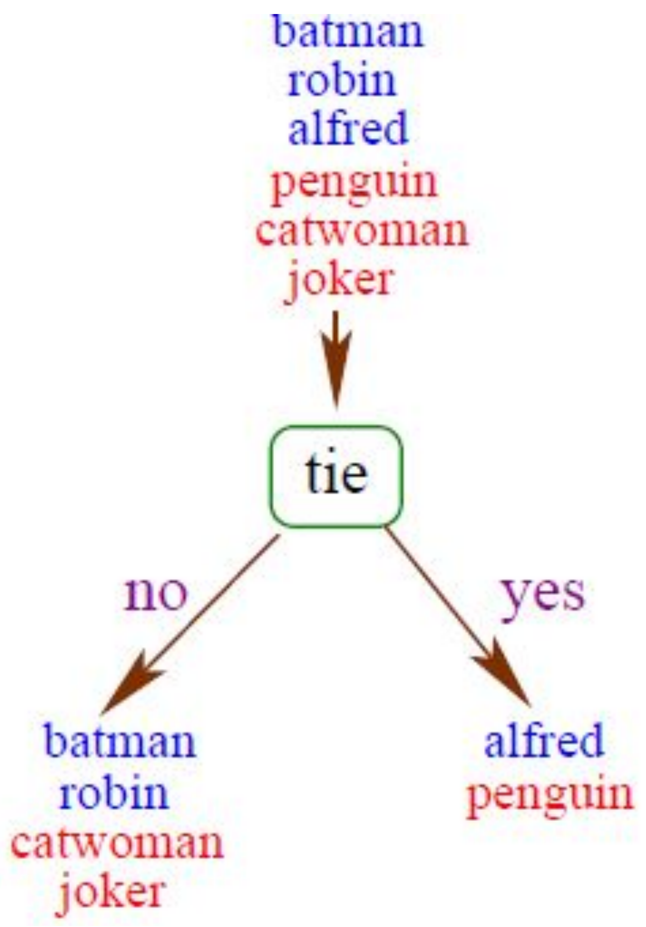
- key problem: choosing best rule to split on:

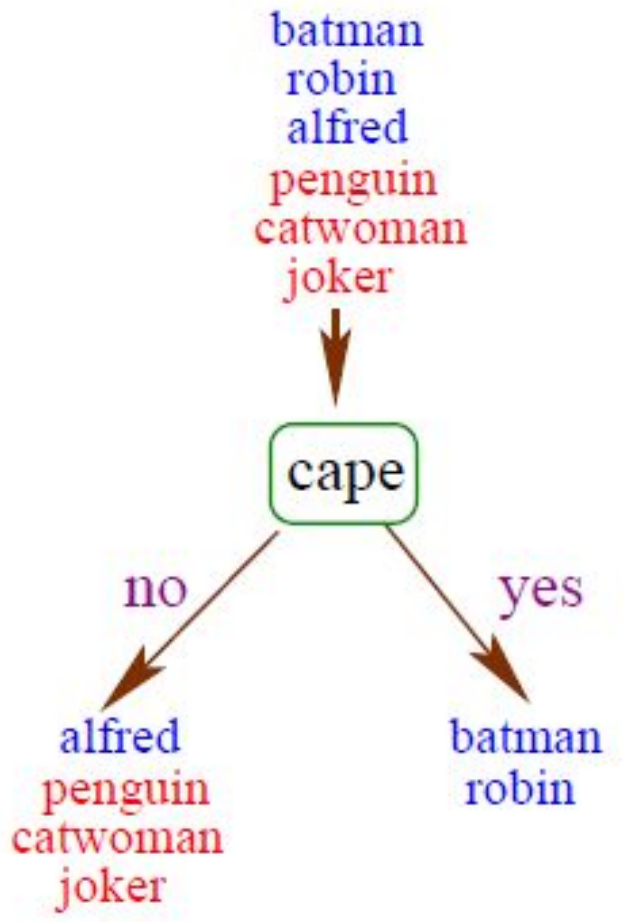




How to Choose the Splitting Rule

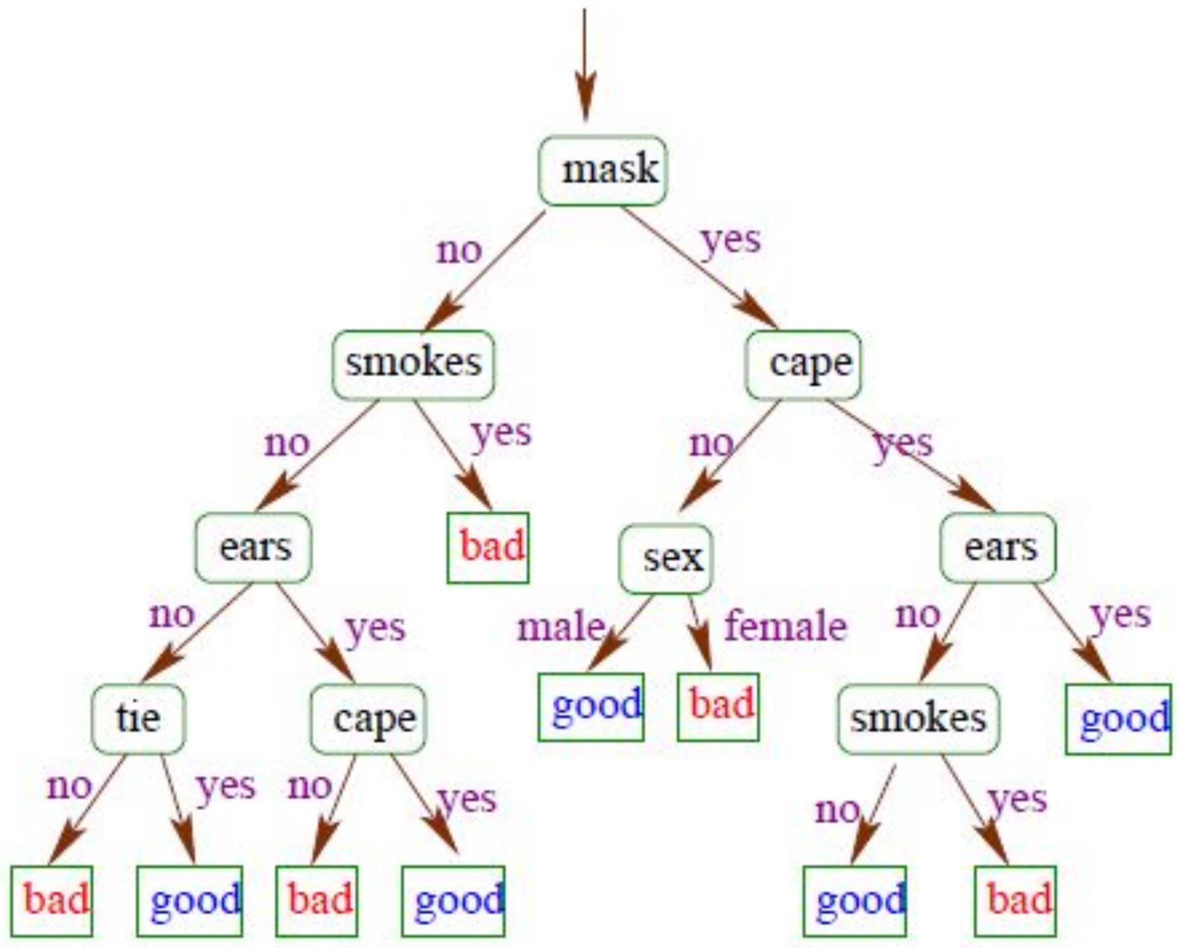
key problem: choosing best rule to split on:





idea: choose rule that leads to greatest increase in "purity"

A Possible Classifier



How to Measure Purity

- Information gain
- Gini Index
- Gain Ratio

Information Gain

- 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

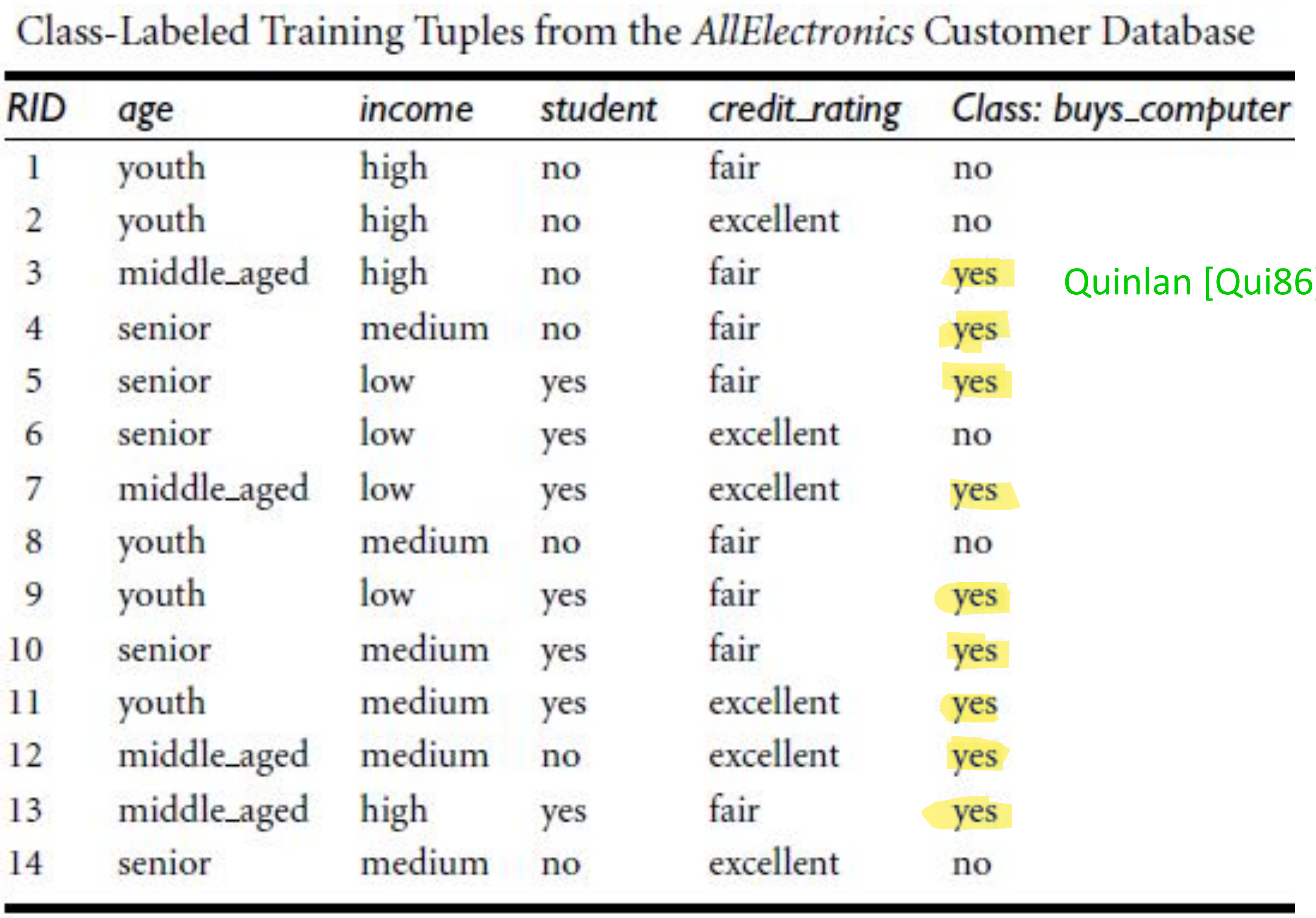
$$Gain(A) = Info(D) - Info_A(D)$$

$$\boxed{Gain(A) = Info(D) - Info_A(D)}$$

- **Gain(A)** tells us how much would be gained by branching on A

- The attribute A with the highest **information gain, Gain (A),** is chosen as the splitting attribute at node N.

An Illustrative Example



Class-Labeled Training Tuples from the *AllElectronics* Customer Database

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

Quinlan [Qui86].

How to choose best splitting criterion?

- The class label attribute, buys computer, has two distinct values (namely, {yes, no});
- two distinct classes (i.e., m = 2).
- class C_1 = yes
- class C_2 = no
- 09 tuples of class = yes
- 05 tuples of class = no
- A (root) node N is created for the tuples in D.
- To find the splitting criterion for these tuples, **compute the information gain** of each attribute.

- Expected information needed to classify a tuple in *D:*

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

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

- Next, we need to compute the expected information requirement for each attribute.
- Start with attribute: ***age***
- age category “youth,”: yes = 02 tuples & no = 03 tuples.
- category “middle aged,”: yes = 04 tuples & no = 0 tuples.
- category “senior,”: *yes = 03 tuples & no = 02 tuples.*

- The expected information needed to classify a tuple in *D* if the tuples are partitioned according to **age:**

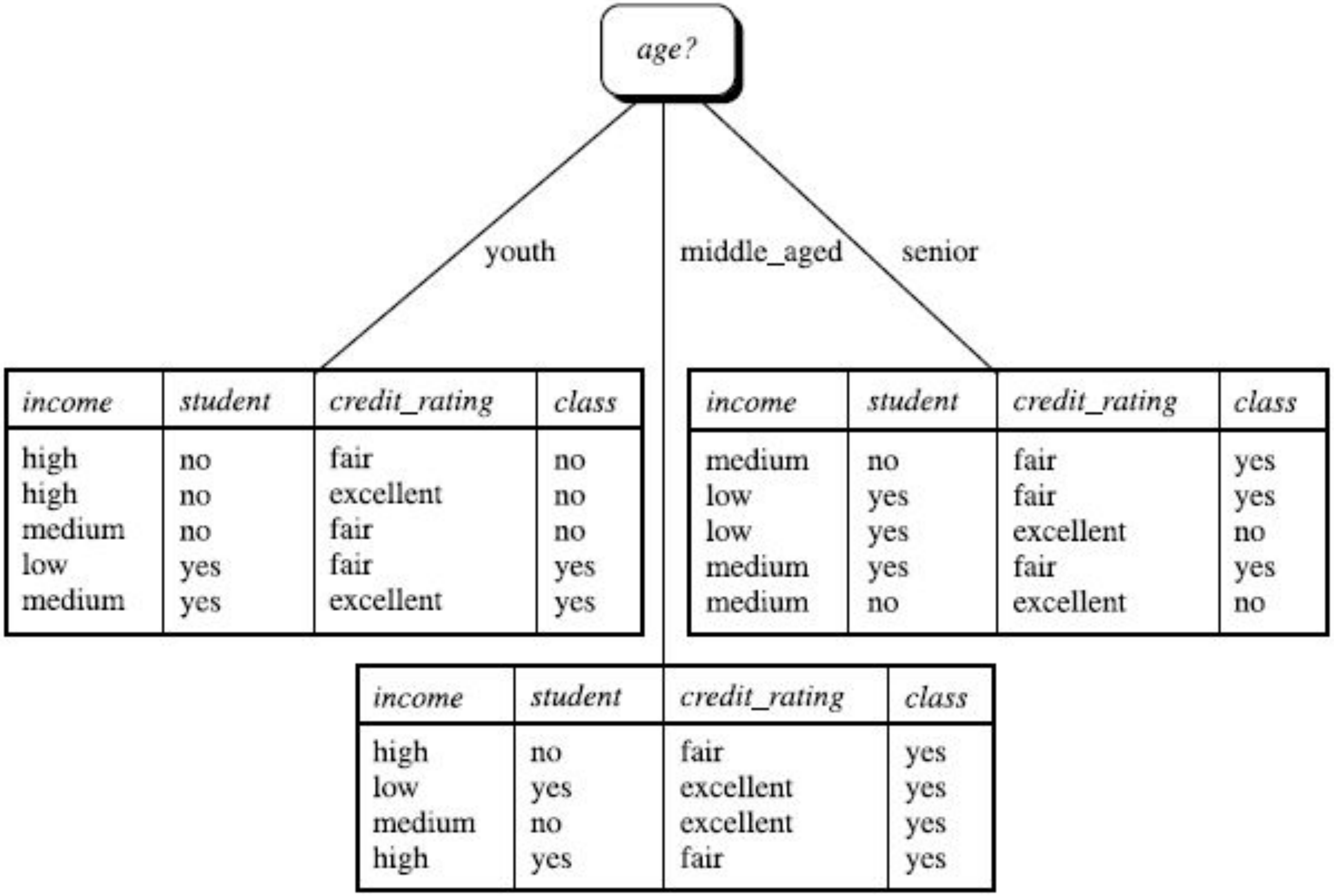
$$\boxed{Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)}$$

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

- Hence, the gain in information from such a partitioning would be

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

Similarly, we can compute Gain(income) = 0.029 bits, Gain(student) = 0.151 bits, and Gain(credit_rating) = 0.048 bits.
Because age has the highest information gain among the attributes, it is selected as the splitting attribute.



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*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	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

Gain Ratio

- The information gain measure is **biased** toward tests with many outcomes.
- That is, it prefers to select attributes having a large number of values.
- For example, consider an attribute that acts as a unique identifier such as *product_ID.*
- *A split on product_ID would* result in a large number of partitions (as many as there are values), each one containing just one tuple.
- Because each partition is pure, the information required to classify data set *D based on this partitioning would be* ***$Info_{product\_ID}(D) = 0$****.*

 information gained by partitioning on this attribute is maximal.

- Such a partitioning is **useless f**or classification.

- C4.5, a successor of ID3, uses an extension to information gain known as ***gain ratio***, which attempts to overcome this bias.
- It applies a kind of normalization to information gain using a "**split information**" value defined analogously with *Info(D):*

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

This value represents the potential information generated by splitting the training data set, *D, into v partitions, corresponding to the v outcomes of a test on attribute A.*

- The gain ratio is defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}.$$

- The attribute with the **maximum gain ratio** is selected as the splitting attribute.
- Note, however, that as the split information approaches 0, the ratio becomes unstable.

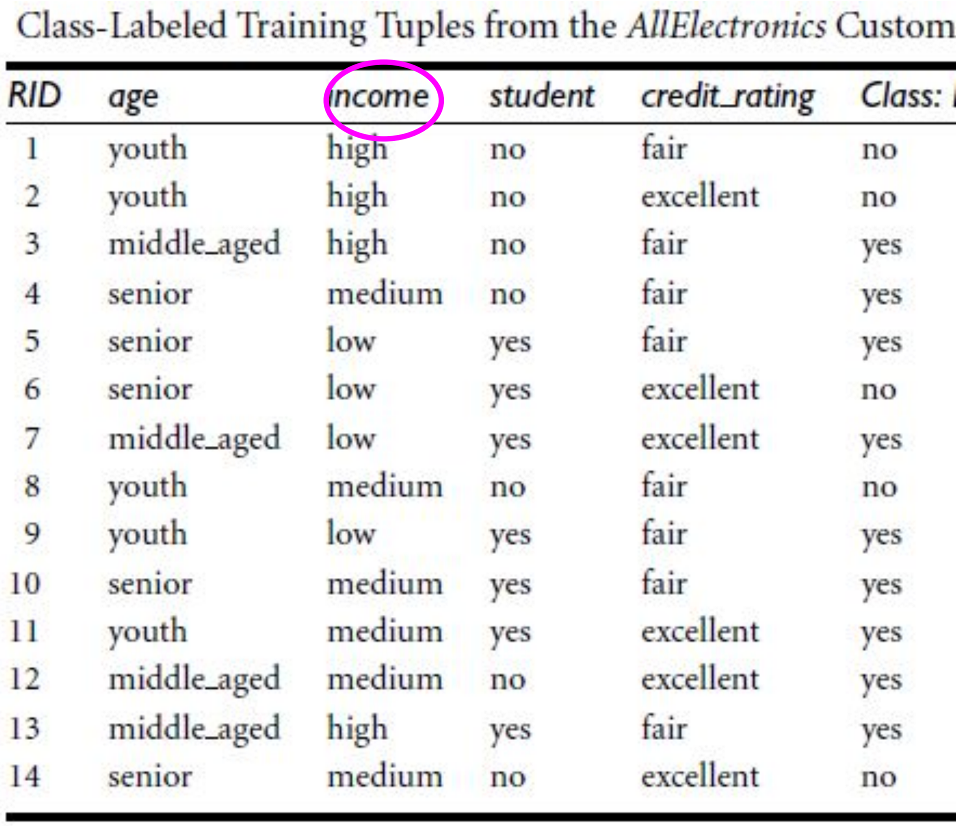
Example: Computation of gain ratio for the attribute *income*

- A test on ***income*** *splits the data of* into 03 partitions:

✔ *Low = 04 tuples*

✔ *Medium = 06 tuples*

✔ *High = 04 tuples*



Class-Labeled Training Tuples from the *AllElectronics* Custom

RID	*age*	*income*	*student*	*credit_rating*	*Class:*
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

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

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

$$= 1.557.$$

$$\boxed{Gain(age) = Info(D) - Info_{age}(D)}$$

$$\boxed{GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}.}$$

Gain(income) = 0.029
[See previous example]

$$= \frac{0.029}{1.557}$$

$$= 0.019$$

Gini Index

- The Gini index is used in CART (Classification & Regression Tree).
- Gini index measures the impurity of *D, a data partition or set of training tuples*

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

- where p_i *is the probability that a tuple in D belongs to class* C_i *and is estimated by* $|C_{i,D}|/|D|$.
- *The sum is computed over m classes.*

- The Gini index considers a binary split for each attribute.
- To determine the best binary split on *A, we examine all the possible subsets* that can be formed using known values of *A.*
- Each subset, S_A, *can be considered as a* binary test for attribute *A of the form "*$A \in S_A$*?"*
- *Given a tuple, this test is satisfied if* the value of *A for the tuple is among the values listed in* S_A*.*
- *If A has v possible values,* then there are 2^v *possible subsets.*

- For example, if ***income*** *has three possible values: low, medium, high,*
- *possible subsets are {low, medium, high}, {low, medium}, {low, high}, {medium, high}, {low}, {medium}, {high}, and {}.*
- *We exclude the* power set, {*low, medium, high}, and the empty set from consideration since, conceptually,* they do not represent a split.
- Therefore, there are 2^{v} *-2 possible ways to form two* partitions of the data, *D, based on a binary split on A.*

- When considering a binary split, we compute a weighted sum of the impurity of each resulting partition.
- For example, if a binary split on *A partitions D into D_1 and D_2, the* Gini index of *D given that partitioning is*

$$Gini_A(D) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2).$$

For each attribute, each of the possible binary splits is considered.

For a discrete-valued attribute, the subset that gives the **minimum Gini index for that attribute is selected as its splitting subset.**

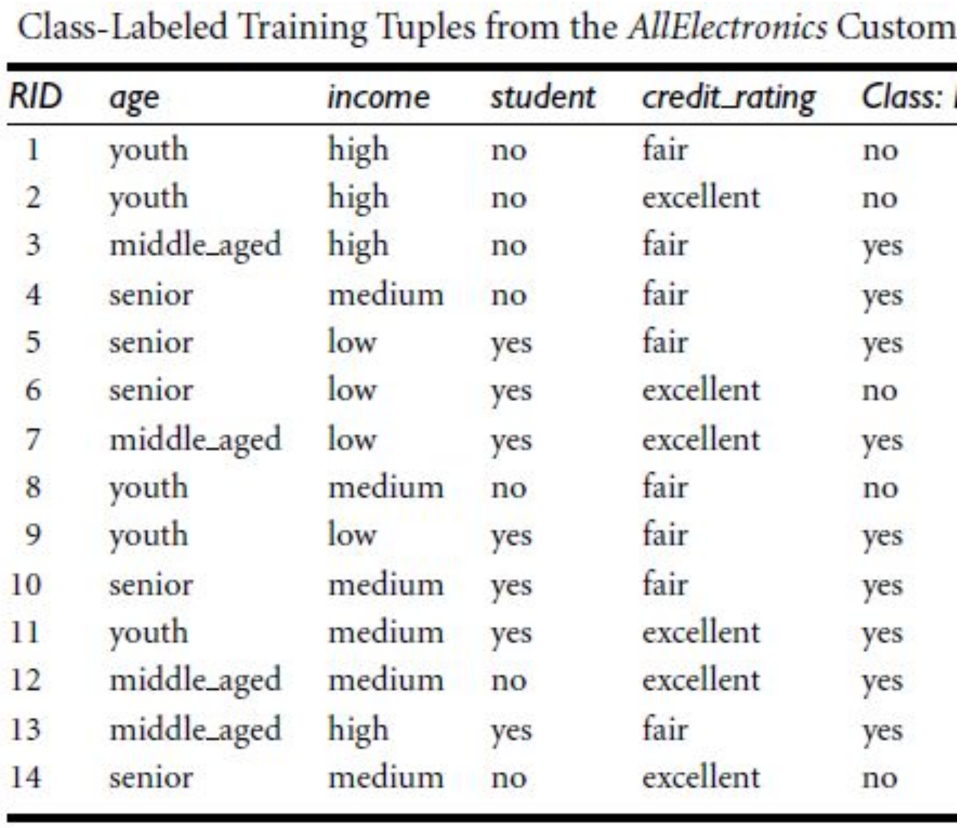
- The reduction in impurity that would be incurred by a binary split on a discrete- or continuous-valued attribute *A*

$$\boxed{\Delta Gini(A) = Gini(D) - Gini_A(D).}$$

- The attribute that maximizes the reduction in impurity (or, equivalently, has the minimum Gini index) is selected as the splitting attribute.
- This attribute & either its splitting subset or split-point together form the splitting criterion.

Induction of a decision tree using the Gini index

- C1= *buys computer =yes = 09*
- *C2 = buys computer = no = 05*



Class-Labeled Training Tuples from the *AllElectronics* Custom

RID	*age*	*income*	*student*	*credit_rating*	*Class:*
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

- the Gini index to compute the impurity of *D*

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

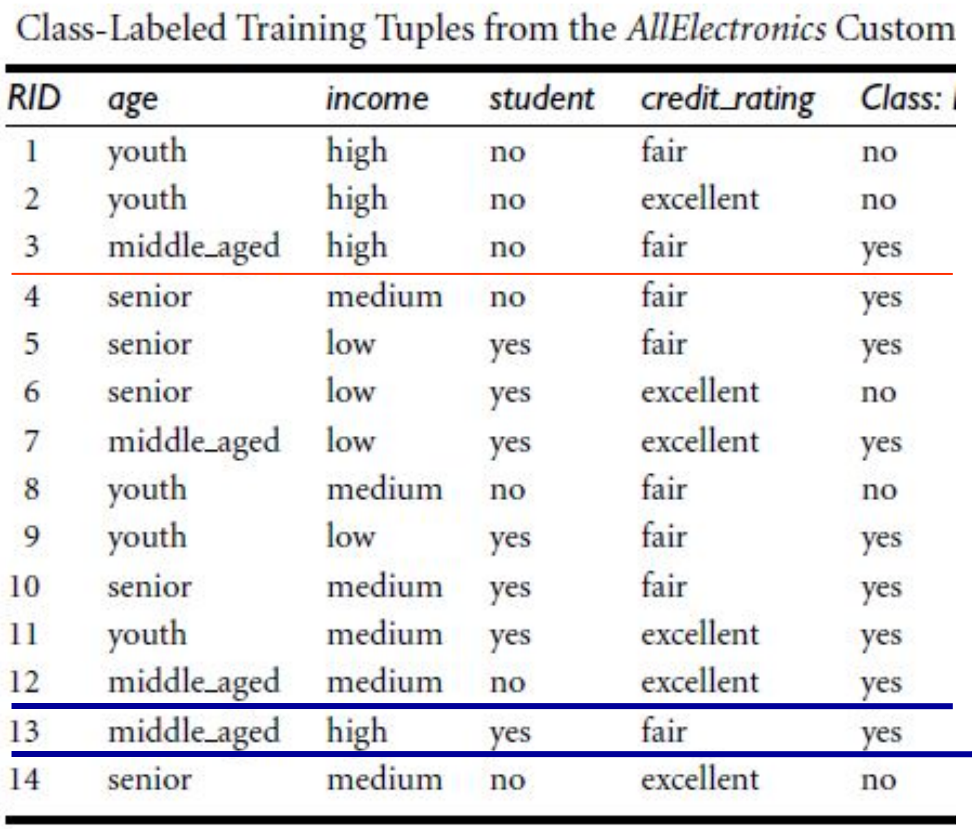
A (root) node *N is created for the tuples in D.*

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

To find the splitting criterion for the tuples in *D, we need to compute the Gini index* for each attribute

- Let's start with the attribute *income* & *consider each of the possible* splitting subsets.
- Consider the subset {*low, medium}.*
- *This would result in 10 tuples in* partition *D_1 satisfying the condition "income $\in$ {low, medium}."*
- *The remaining 04* tuples of *D would be assigned to partition D_2.*

Yes = 07
No = 03


Class-Labeled Training Tuples from the *AllElectronics* Custom

RID	*age*	*income*	*student*	*credit_rating*	*Class:*
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

$$Gini_{income \in \{low, medium\}}(D)$$

$$= \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2)$$

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

$$= 0.443$$

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

- Similarly, the Gini index values for splits on the remaining subsets

✔ 0.458 = {*low, high*} + {*medium*}

✔ *0.450 = {medium, high} + {low}*

- Therefore, the best binary split for attribute ***income*** *is on {low, medium} or {high} = 0.443* because it minimizes the Gini index.

❑ Evaluating *age, we obtain {youth, senior} (or {middle_aged}) as the best split for age with a Gini index of 0.375*

- the attributes *student* and *credit_rating are both binary, with Gini index values of 0.367 & 0.429, respectively.*
- The attribute ***age*** *and splitting subset {youth, senior} therefore give the minimum* Gini index overall, with a reduction in impurity of 0.459-0.357= 0.102.
- The binary split "*age* $\in$ *{youth, senior}" results in the* ***maximum reduction in impurity*** *of the tuples* in *D and is returned as the splitting criterion.*
- Node *N is labeled with the criterion, two* branches are grown from it, and the tuples are partitioned accordingly.

Decision Trees

best known:

- C4.5 (Quinlan)
- CART (Breiman, Friedman, Olshen & Stone)

- very fast to train and evaluate
- relatively easy to interpret
- but: accuracy often not state-of-the-art