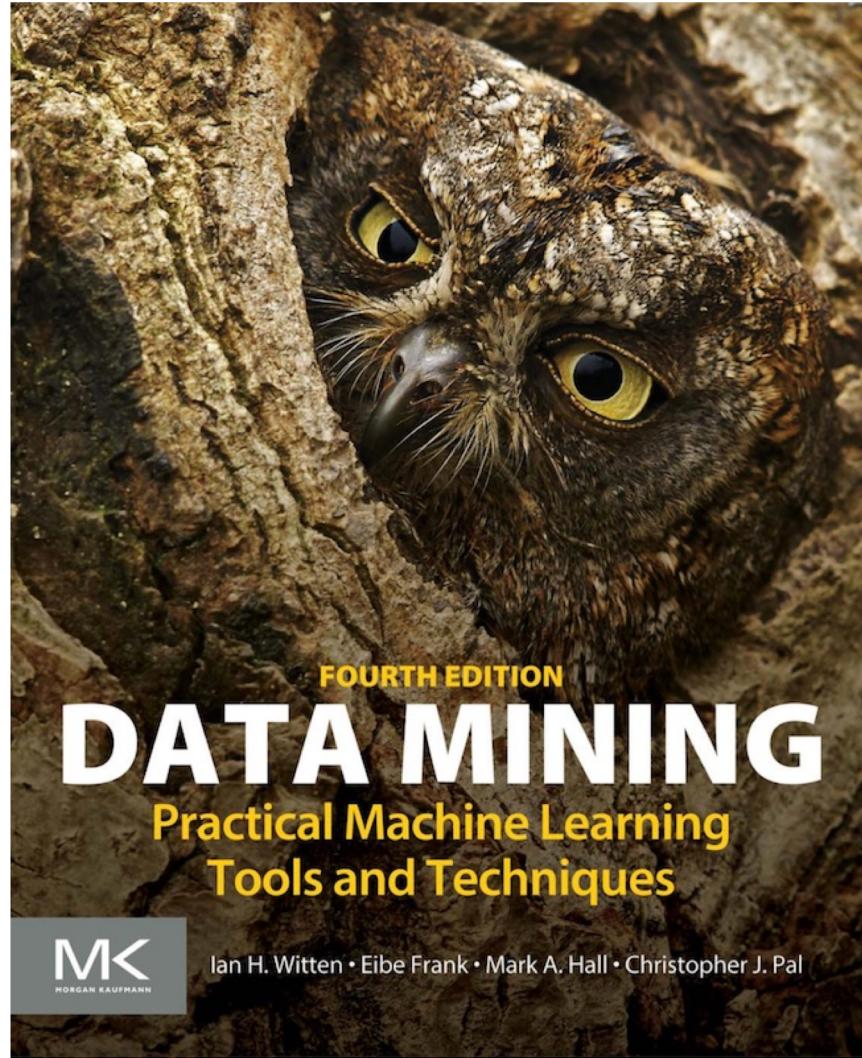


# Decision Trees

Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)

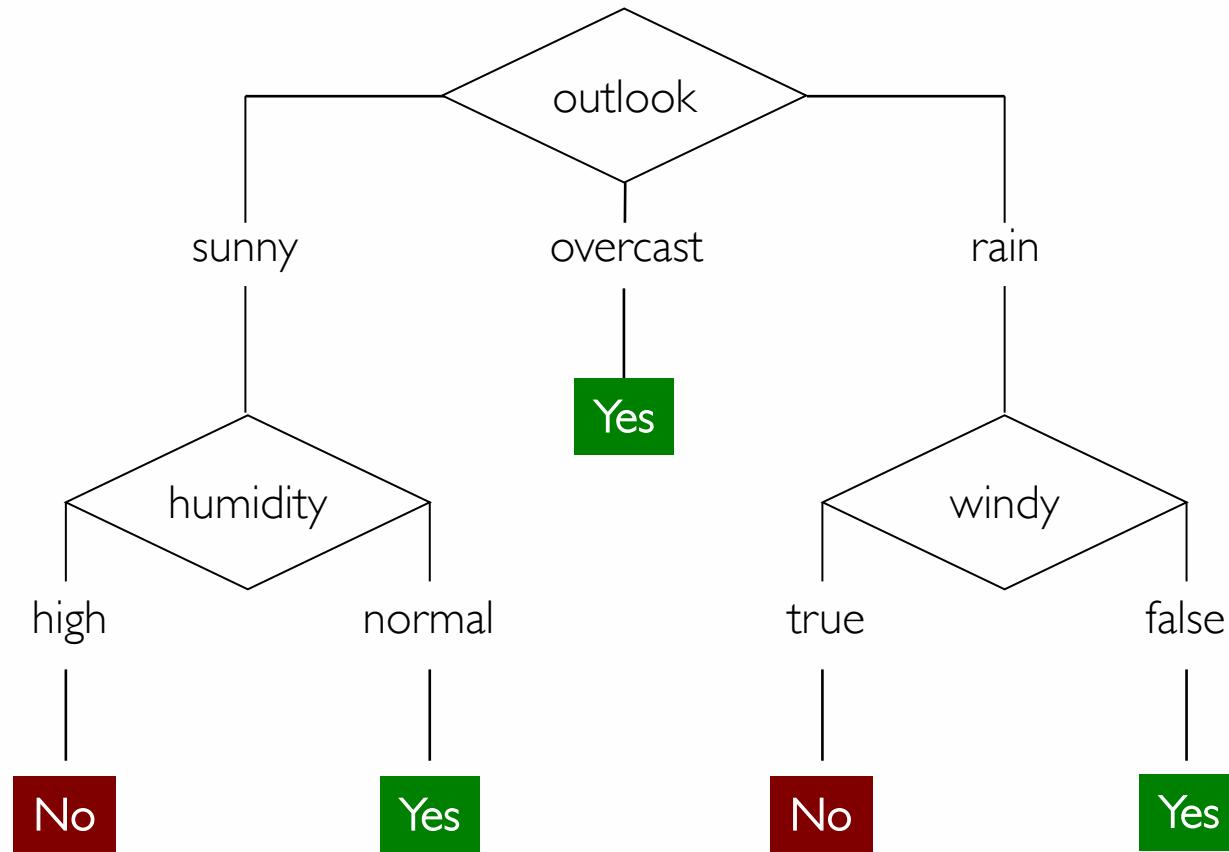


Data Mining: Practical Machine Learning Tools and Techniques (Morgan Kaufmann Series in Data Management Systems) 4th Edition by Ian H. Witten, Eibe Frank, Mark A. Hall, Christopher J. Pal

# The Weather Dataset

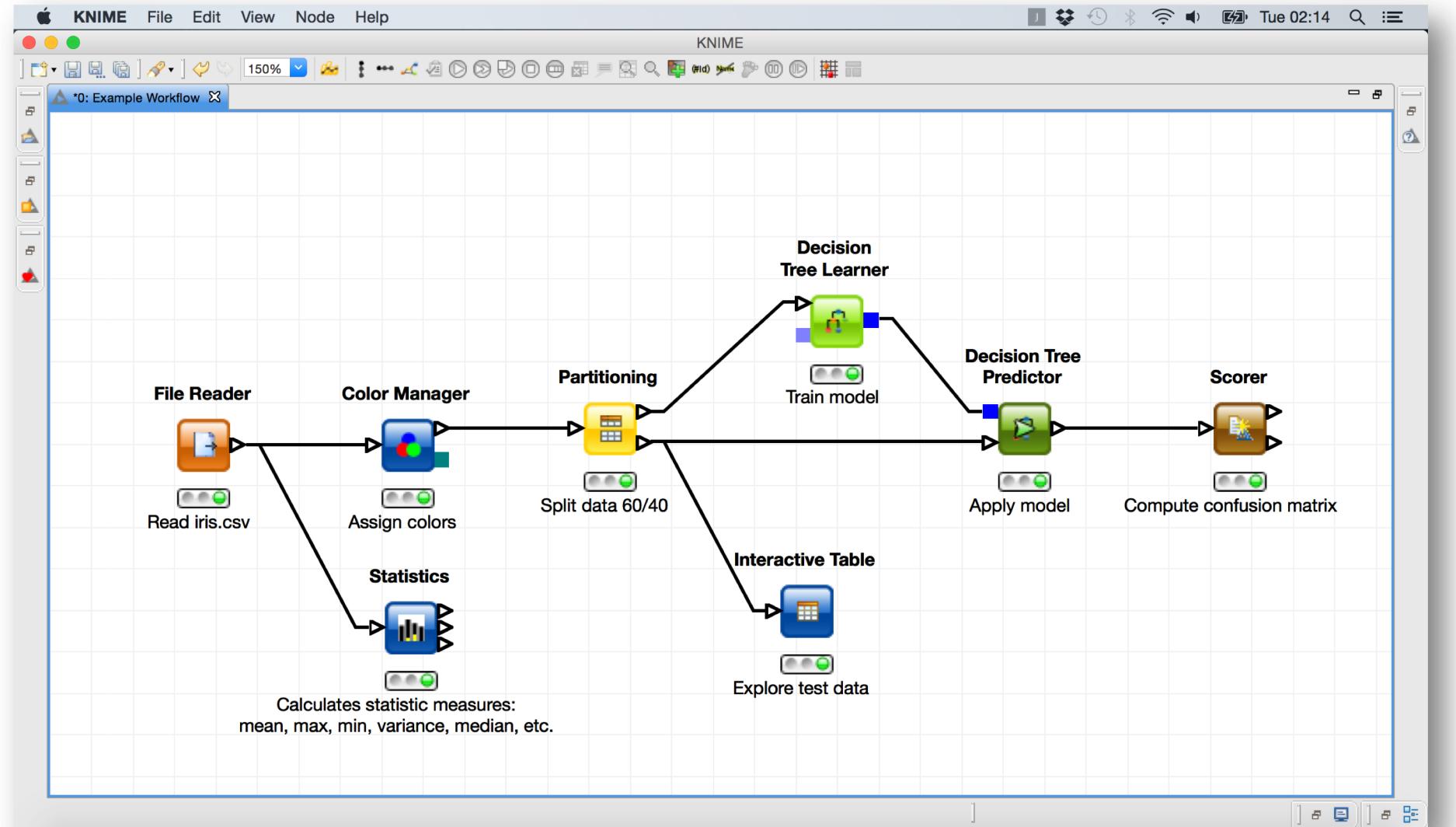
Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny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

# The Decision Tree for the Weather Dataset



## What is a Decision Tree?

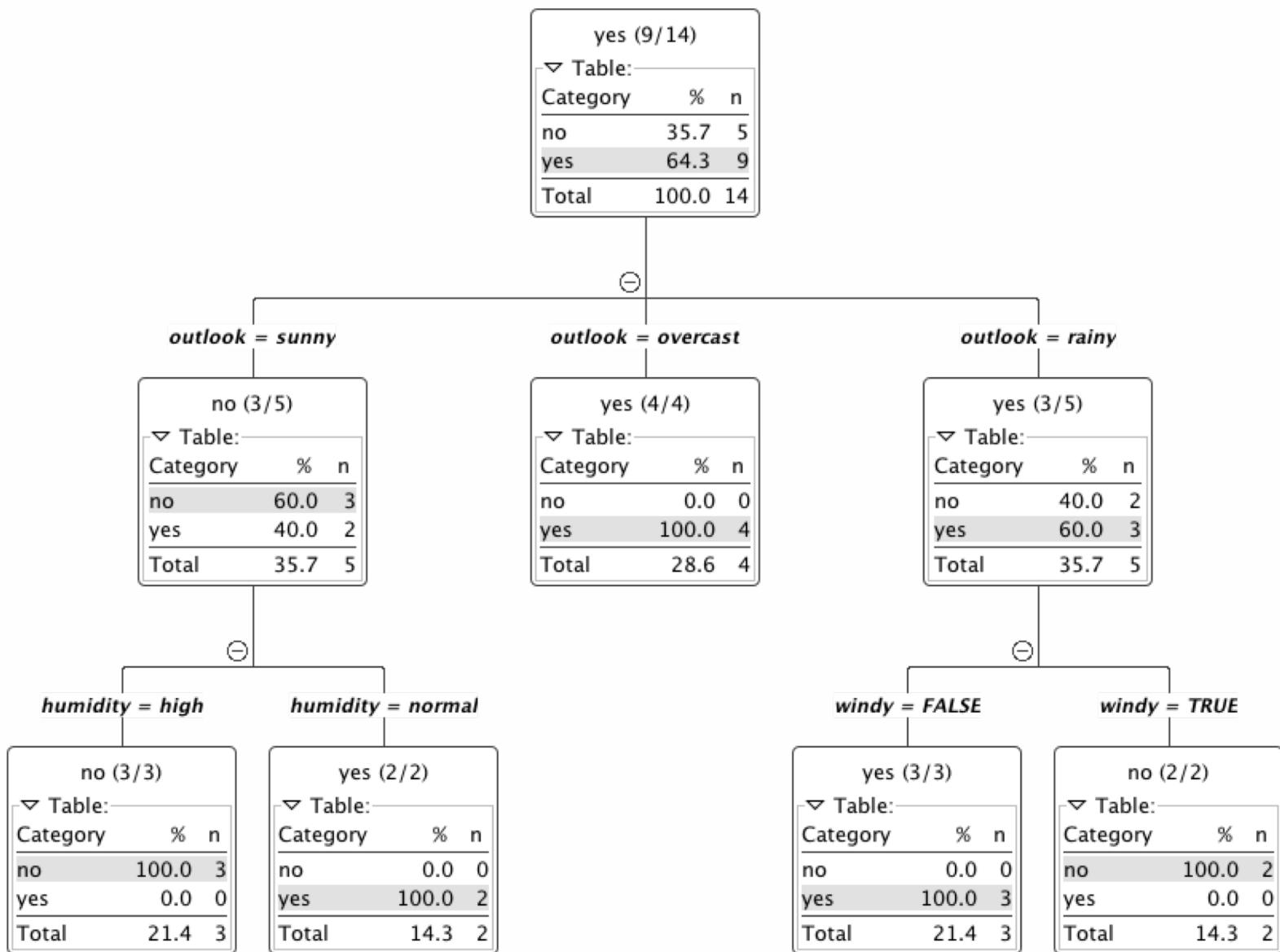
- An internal node is a test on an attribute
- A branch represents an outcome of the test, e.g., outlook=windy
- A leaf node represents a class label or class label distribution
- At each node, one attribute is chosen to split training examples into distinct classes as much as possible
- A new case is classified by following a matching path to a leaf node



Building a Decision Tree with KNIME

# Decision Tree Representations (Graphical)

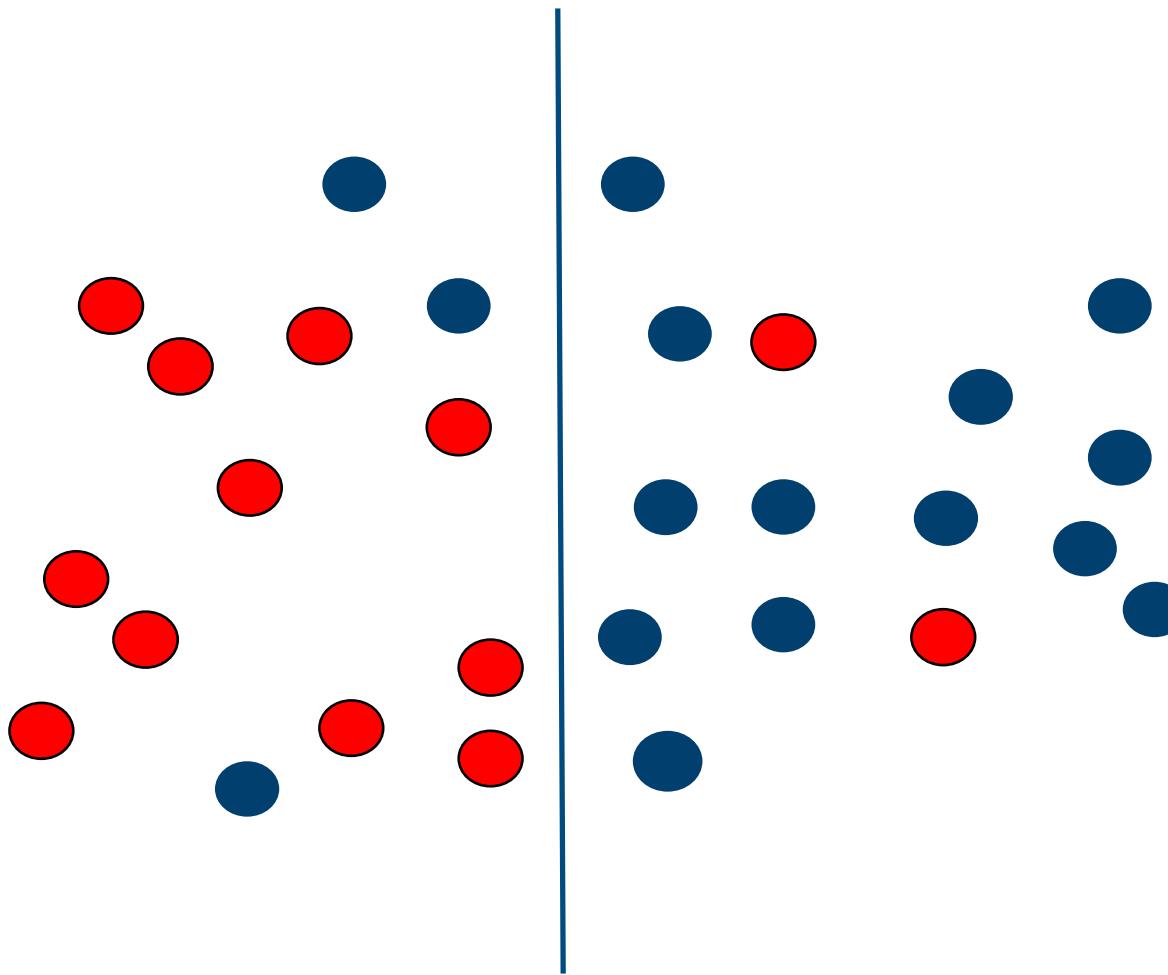
7



# Computing Decision Trees

- Top-down Tree Construction
  - Initially, all the training examples are at the root
  - Then, the examples are recursively partitioned, by choosing one attribute at a time
- Bottom-up Tree Pruning
  - Remove subtrees or branches, in a bottom-up manner, to improve the estimated accuracy on new cases.

What Splitting Attribute?  
(ID3)

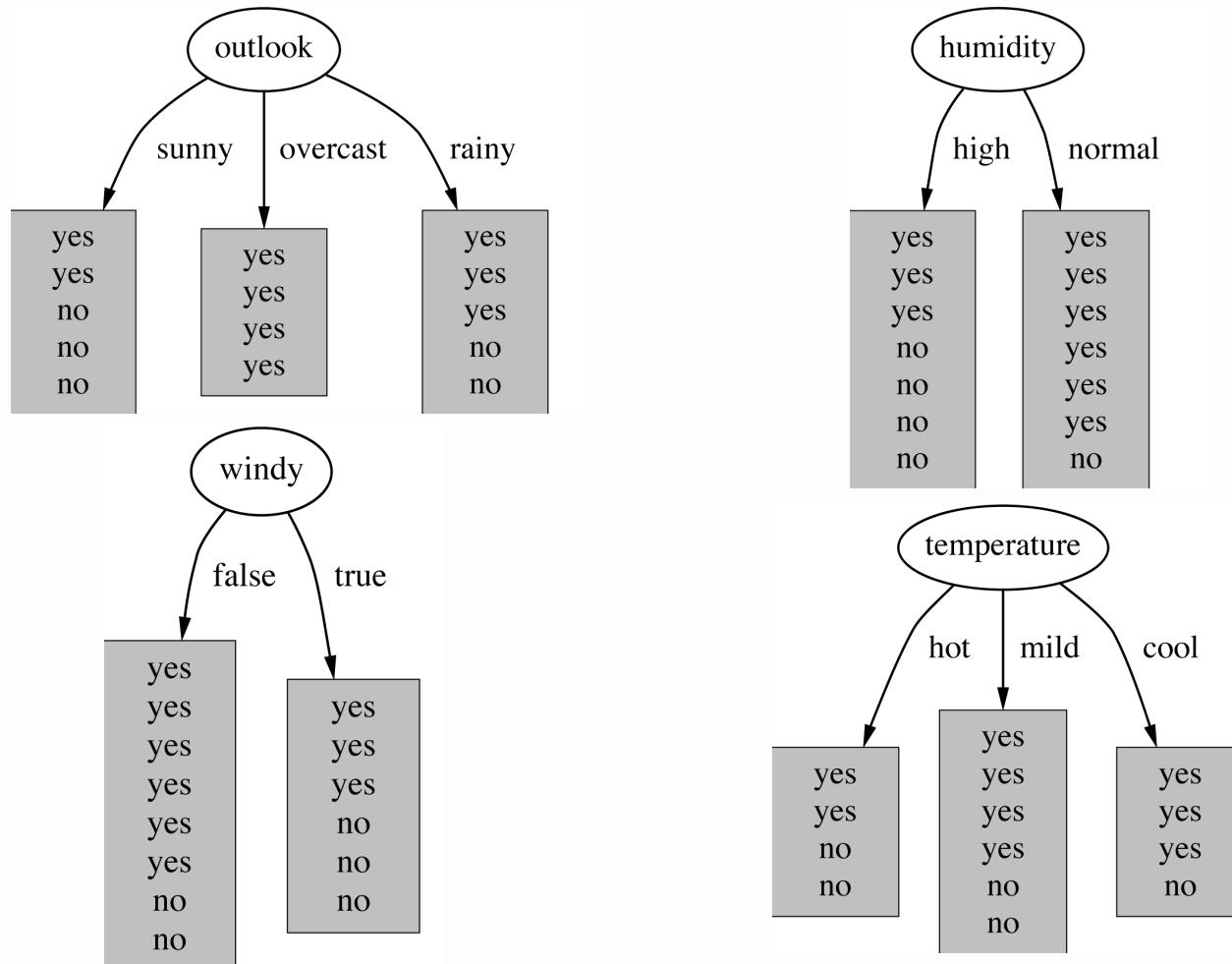


We are looking for a way to separate classes so as to create areas that are “pure” in that they contain mainly examples of one class

- At each node, available attributes are evaluated on the basis of separating the classes of the training examples
- A purity or impurity measure is used for this purpose
- Information Gain: increases with the average purity of the subsets that an attribute produces
- Splitting Strategy: choose the attribute that results in greatest information gain
- Typical goodness functions: information gain (ID3), information gain ratio (C4.5), gini index (CART)

# Which Attribute Should We Select?

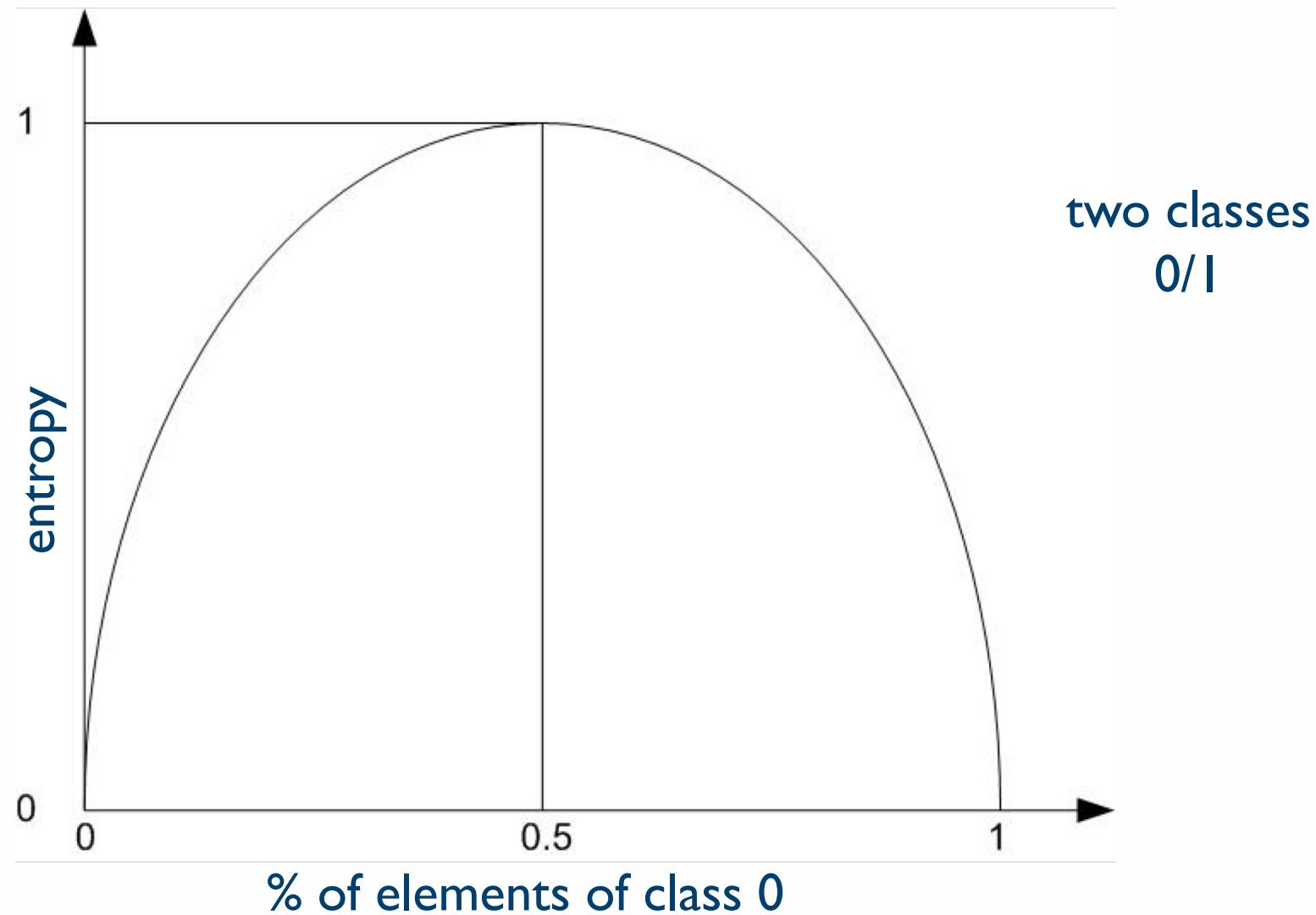
13



- Information is measured in bits
  - Given a probability distribution, the info required to predict an event is the distribution's entropy
  - Entropy gives the information required in bits (this can involve fractions of bits!)
- Formula for computing the entropy

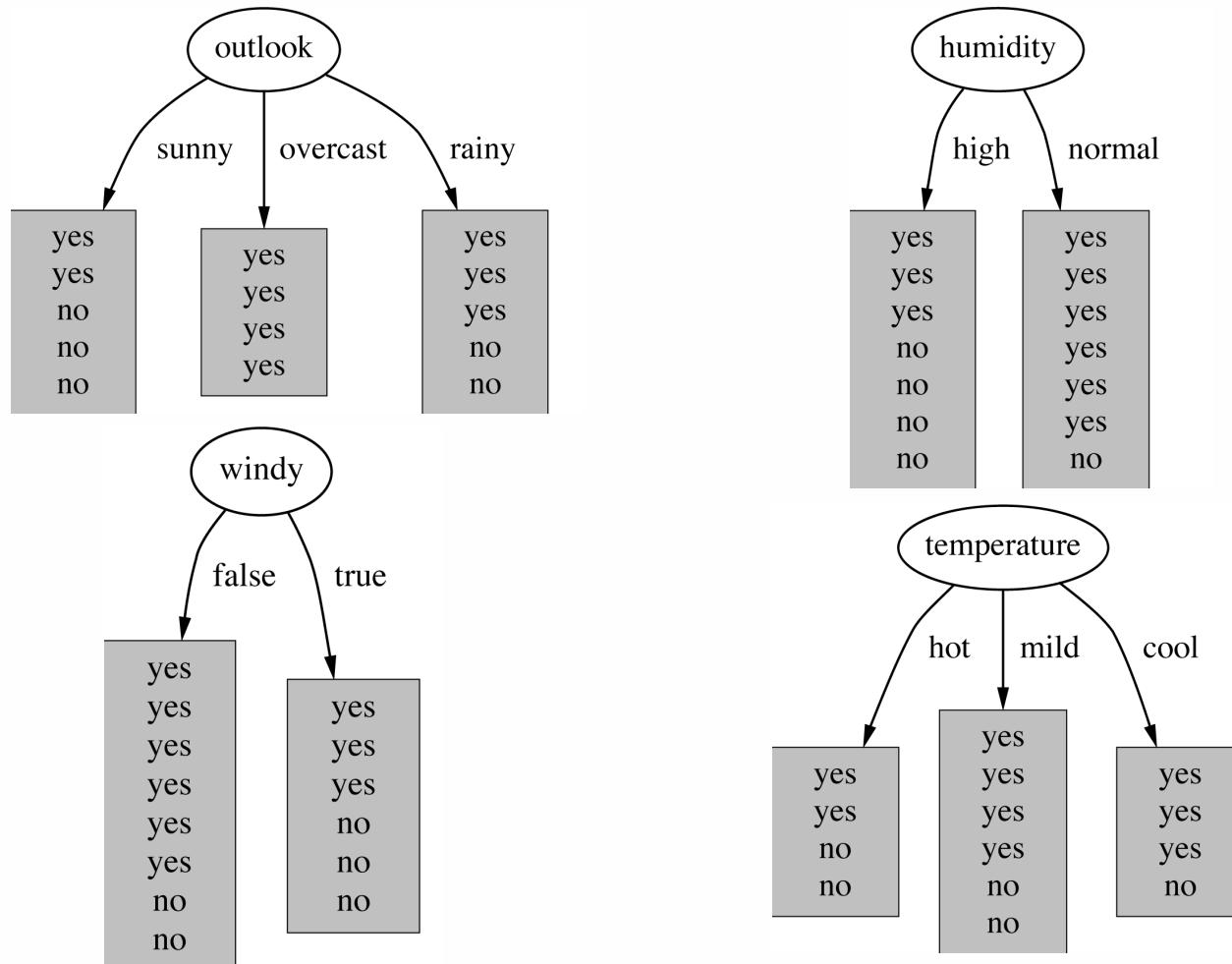


$$\text{entropy}(p_1, p_2, \dots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \dots - p_n \log p_n$$



# Which Attribute Should We Select?

16



- “outlook” = “sunny”

$$\text{info}([2, 3]) = \text{entropy}(2/5, 3/5) = 0.971$$

- “outlook” = “overcast”

$$\text{info}([4, 0]) = \text{entropy}(1, 0) = 0.000$$

- “outlook” = “rainy”

$$\text{info}([3, 2]) = \text{entropy}(3/5, 2/5) = 0.971$$

- Expected information for attribute

$$\begin{aligned}\text{info}([2, 3][4, 0][3, 2]) &= 5/14 \times 0.971 + \\ &4/14 \times 0 + 5/14 \times 0.971\end{aligned}$$

- Difference between the information before split and the information after split

$$gain(A) = info(D) - info_A(D)$$

- The information before the split,  $info(D)$ , is the entropy,

$$info(D) = -p_1 \log p_1 - \dots - p_n \log p_n$$

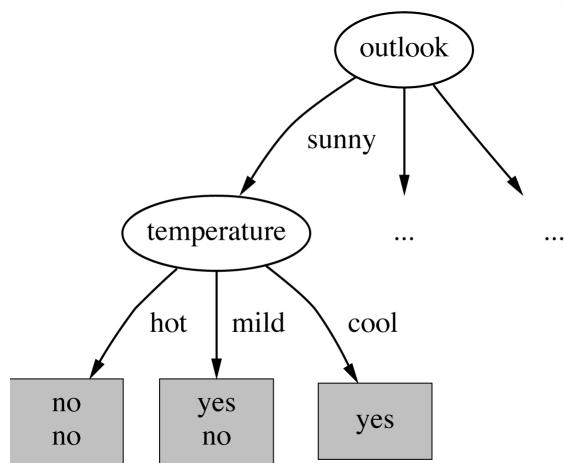
- The information after the split using attribute A is computed as the weighted sum of the entropies on each split, given n splits,

$$info_A(D) = \frac{|D_1|}{|D|}info(D_1) + \dots + \frac{|D_n|}{|D|}info(D_n)$$

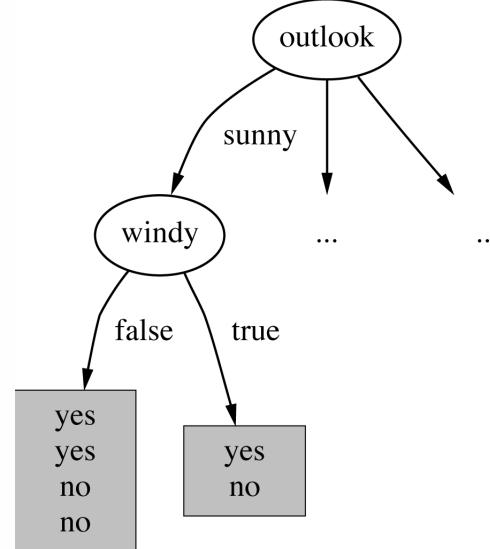
- Difference between the information before split and the information after split

$$\begin{aligned} \text{gain}(\text{outlook}) &= \text{info}([9, 5]) - \text{info}([2, 3][4, 0][3, 2]) \\ &= 0.940 - 0.693 \\ &= 0.247 \end{aligned}$$

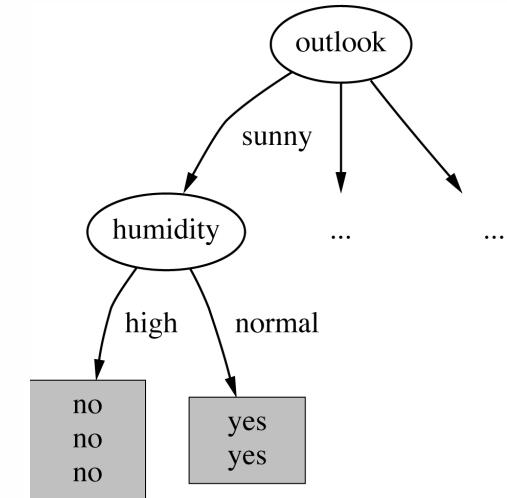
- Information gain for the attributes from the weather data:
  - $\text{gain}(\text{"outlook"}) = 0.247$
  - $\text{gain}(\text{"temperature"}) = 0.029$
  - $\text{gain}(\text{"humidity"}) = 0.152$
  - $\text{gain}(\text{"windy"}) = 0.048$



$$gain(\text{temperature}) = 0.571$$

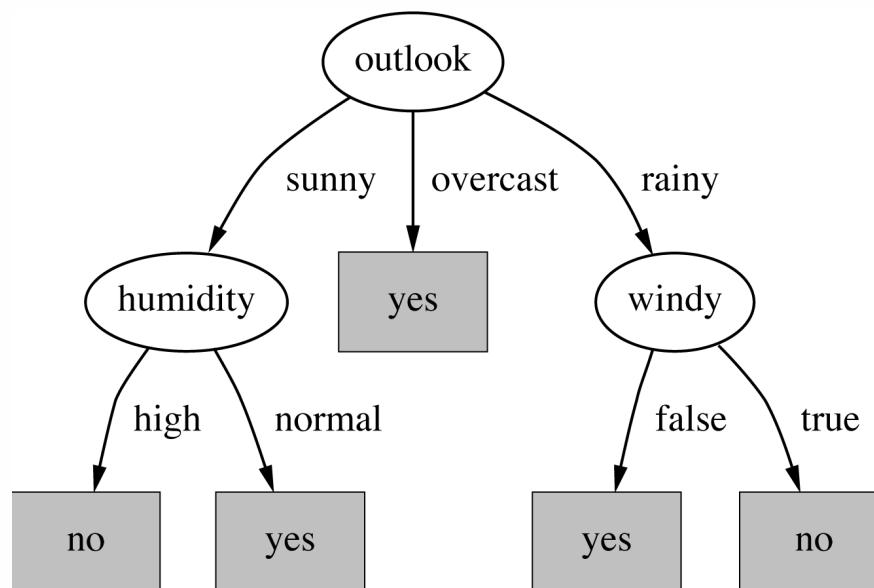


$$gain(\text{windy}) = 0.020$$



$$gain(\text{humidity}) = 0.971$$

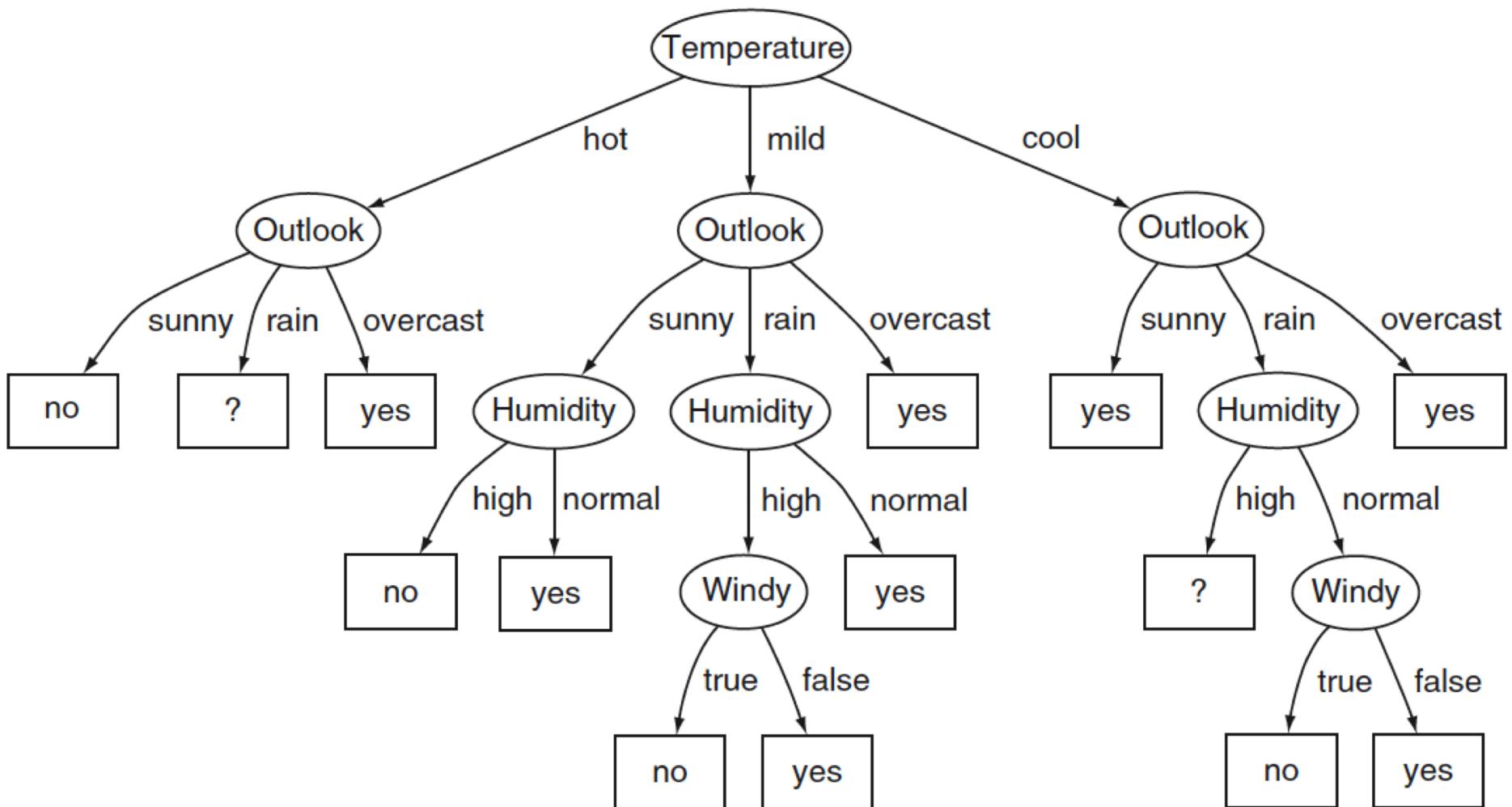
- Not all the leaves need to be pure
- Splitting stops when data can not be split any further



- There are several possible stopping criteria
- All samples for a given node belong to the same class
- If there are no remaining attributes for further partitioning, majority voting is employed
- There are no samples left
- Or there is nothing to gain in splitting

We Can Always Build  
a 100% Accurate Tree

But do we want it?

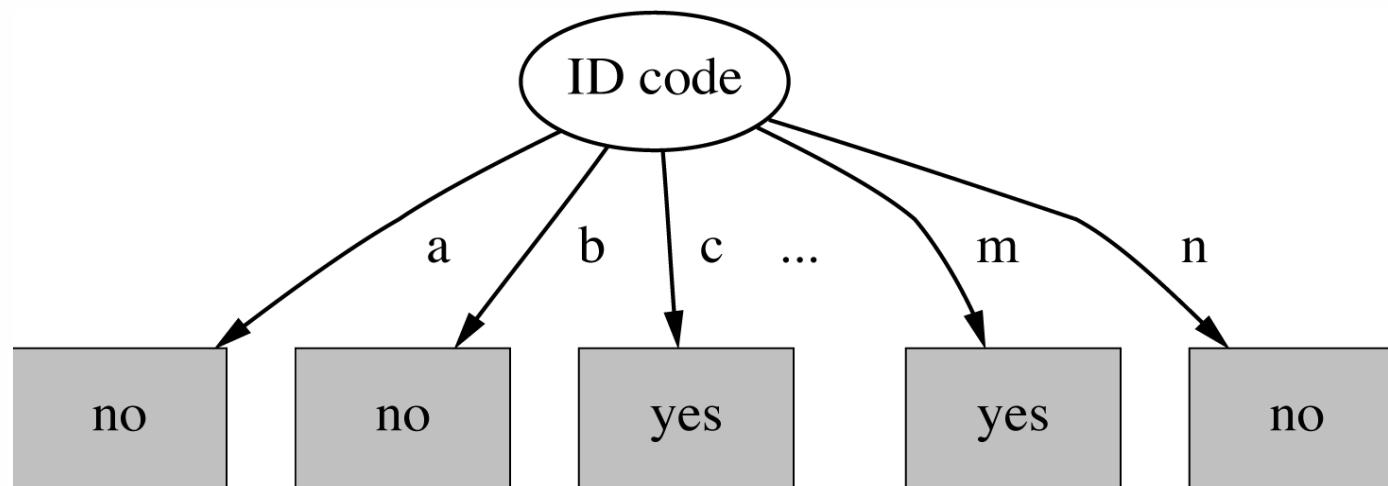


# Another Version of the Weather Dataset

25



ID Code	Outlook	Temp	Humidity	Windy	Play
A	Sunny	Hot	High	False	No
B	Sunny	Hot	High	True	No
C	Overcast	Hot	High	False	Yes
D	Rainy	Mild	High	False	Yes
E	Rainy	Cool	Normal	False	Yes
F	Rainy	Cool	Normal	True	No
G	Overcast	Cool	Normal	True	Yes
H	Sunny	Mild	High	False	No
I	Sunny	Cool	Normal	False	Yes
J	Rainy	Mild	Normal	False	Yes
K	Sunny	Mild	Normal	True	Yes
L	Overcast	Mild	High	True	Yes
M	Overcast	Hot	Normal	False	Yes
N	Rainy	Mild	High	True	No



- Entropy for splitting using “ID Code” is zero, since each leaf node is “pure”
- Information Gain is thus maximal for ID code

- Attributes with a large number of values are usually problematic
- Examples: id, primary keys, or almost primary key attributes
- Subsets are likely to be pure if there is a large number of values
- Information Gain is biased towards choosing attributes with a large number of values
- This may result in overfitting (selection of an attribute that is non-optimal for prediction)

# Information Gain Ratio

- Modification of the Information Gain that reduces the bias toward highly-branching attributes
- Information Gain Ratio should be
  - Large when data is evenly spread
  - Small when all data belong to one branch
- Information Gain Ratio takes number and size of branches into account when choosing an attribute
- It corrects the information gain by taking the intrinsic information of a split into account

## Information Gain Ratio and Intrinsic information

- Intrinsic information

$$\text{IntrinsicInfo}(S, A) = - \sum \frac{|S_i|}{|S|} \log \frac{|S_i|}{|S|}$$

computes the entropy of distribution of instances into branches

- Information Gain Ratio normalizes Information Gain by

$$\text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\text{IntrinsicInfo}(S, A)}$$

- The intrinsic information for ID code is
$$info([1, 1, \dots, 1]) = 14 \times (-1/14 \times \log 1/14) = 3.807$$
- Importance of attribute decreases as intrinsic information gets larger
- The Information gain ratio of “ID code”,

$$GainRatio(ID\_code) = \frac{0.940}{3.807} = 0.246$$

# Information Gain Ratio for Weather Data

32

Outlook	Temperature
Information after split:	0.693
Gain: 0.940-0.693	0.247
Split info: $\text{info}([5,4,5])$	1.577
Gain ratio: $0.247/1.577$	0.156
Information after split:	0.911
Gain: 0.940-0.911	0.029
Split info: $\text{info}([4,6,4])$	1.362
Gain ratio: $0.029/1.362$	0.021

Humidity	Windy
Information after split:	0.788
Gain: 0.940-0.788	0.152
Split info: $\text{info}([7,7])$	1.000
Gain ratio: $0.152/1$	0.152
Information after split:	0.892
Gain: 0.940-0.892	0.048
Split info: $\text{info}([8,6])$	0.985
Gain ratio: $0.048/0.985$	0.049

- “outlook” still comes out top, however “ID code” has greater Information Gain Ratio
- The standard fix is an ad-hoc test to prevent splitting on that type of attribute
- First, only consider attributes with greater than average Information Gain; Then, compare them using the Information Gain Ratio
- Information Gain Ratio may overcompensate and choose an attribute just because its intrinsic information is very low

# Numerical Attributes (C4.5)

# The Weather Dataset (Numerical)

35

Outlook	Temp	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	78	False	Yes
Rainy	70	96	False	Yes
Rainy	68	80	False	Yes
Rainy	65	70	True	No
Overcast	64	65	True	Yes
Sunny	72	95	False	No
Sunny	69	70	False	Yes
Rainy	75	80	False	Yes
Sunny	75	70	True	Yes
Overcast	72	90	True	Yes
Overcast	81	75	False	Yes
Rainy	71	80	True	No

- First, sort the temperature values, including the class labels
- Then, check all the cut points and choose the one with the best information gain



64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

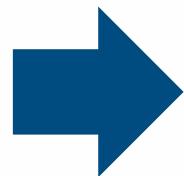
- E.g. temperature  $\leq 71.5$ : yes/4, no/2  
temperature  $> 71.5$ : yes/5, no/3
- $\text{Info}([4,2],[5,3]) = 6/14 \text{ info}([4,2]) + 8/14 \text{ info}([5,3]) = 0.939$
- Place split points halfway between values
- Can evaluate all split points in one pass!

# The Information Gain for Humidity



Humidity	Play
65	Yes
70	No
70	Yes
70	Yes
75	Yes
78	Yes
80	Yes
80	Yes
80	No
85	No
90	No
90	Yes
95	No
96	Yes

sort the  
attribute  
values



Humidity	Play
65	Yes
70	No
70	Yes
70	Yes
75	Yes
78	Yes
80	Yes
80	Yes
80	No
85	No
90	No
90	Yes
95	No
96	Yes
80	No
85	No
90	No
90	Yes
95	No
96	Yes

compute the gain for  
every possible split

what is the information  
gain if we split here?

- Sort instances by the values of the numeric attribute takes  $O(n \log n)$
- Does this have to be repeated at each node of the tree?
- No, since the sort order for children can be derived from sort order for parent
- The time complexity of derivation is  $O(n)$
- We need to create and store an array of sorted indices for each numeric attribute

- Splitting (multi-way) on a nominal attribute exhausts all information in that attribute
- A nominal attribute is tested (at most) once on any path in the tree
- Numeric attribute may be tested several times along a path in the tree and the tree can become hard to read and interpret
- Possible solutions
  - Pre-discretize numeric attributes, or
  - Use multi-way splits instead of binary ones

# Gini Index

## Another Splitting Criteria: The Gini Index

- The Gini index, for a data set T contains examples from n classes, is defined as

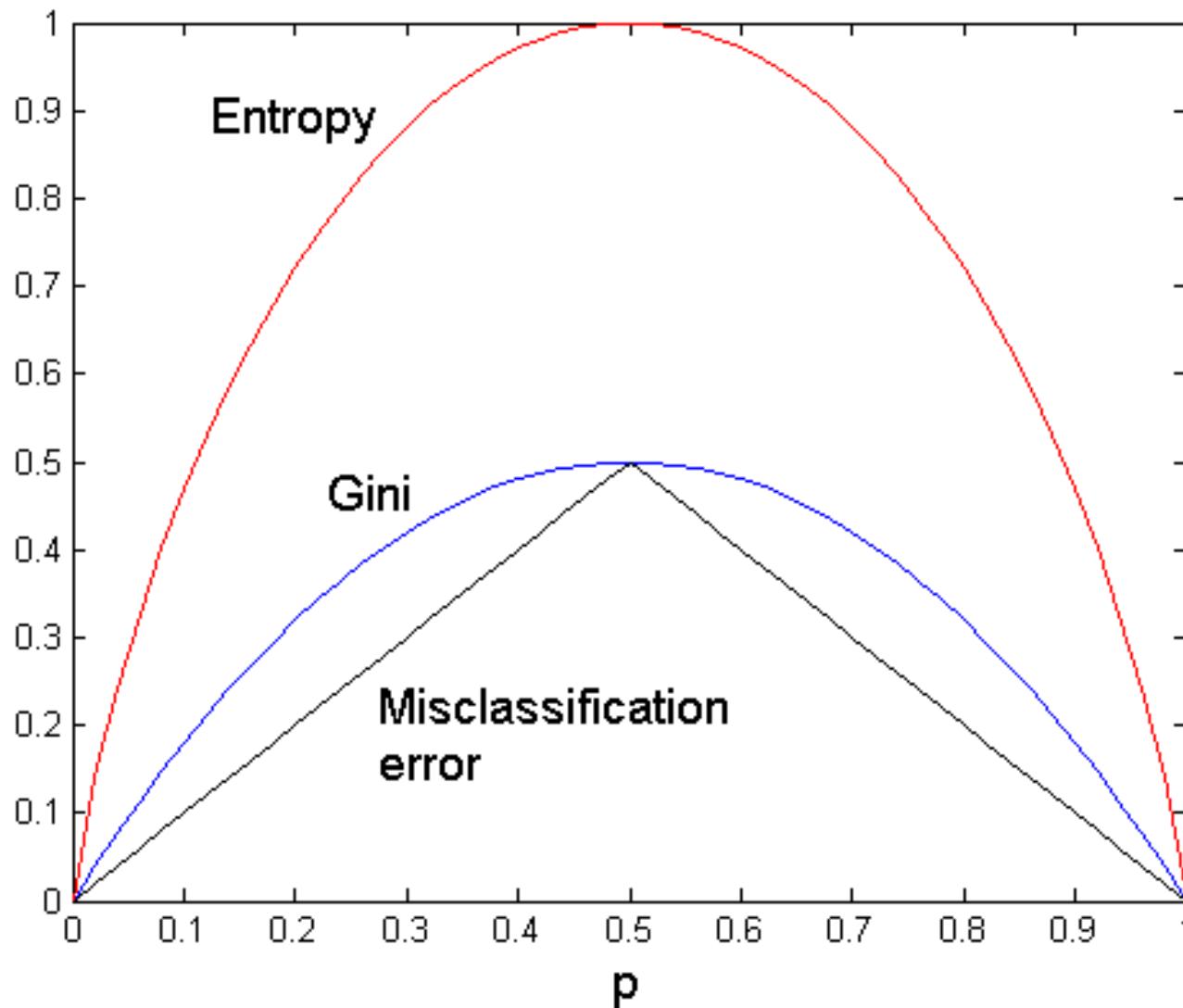
$$gini(T) = 1 - \sum_{j=1}^n p_j^2$$

where  $p_j$  is the relative frequency of class j in T

- $gini(T)$  is minimized if the classes in T are skewed

# The Gini Index vs Entropy

42



- If a data set  $D$  is split on  $A$  into two subsets  $D_1$  and  $D_2$  then,

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

- The reduction of impurity is defined as,

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

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

Gini index is applied  
to produce binary splits

- The dataset has 9 tuples labeled “yes” and 5 labeled “no”

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

- The outlook attribute has three values (overcast, rainy, sunny), thus we have to evaluate three possible partitions
  - {overcast, rainy} and {sunny}
  - {sunny, rainy} and {overcast}
  - {sunny, overcast} and {rainy}

- {overcast, rainy} and {sunny}

$$\begin{aligned}Gini(D_{o,r}, D_s) &= \frac{9}{14}Gini(D_{o,r}) + \frac{5}{14}Gini(D_s) \\&= \frac{9}{14}Gini([7, 2]) + \frac{5}{14}Gini([2, 3]) \\&= \frac{9}{14}0.346 + \frac{5}{14}0.480 \\&= 0.394\end{aligned}$$

- {rainy, sunny} and {overcast}

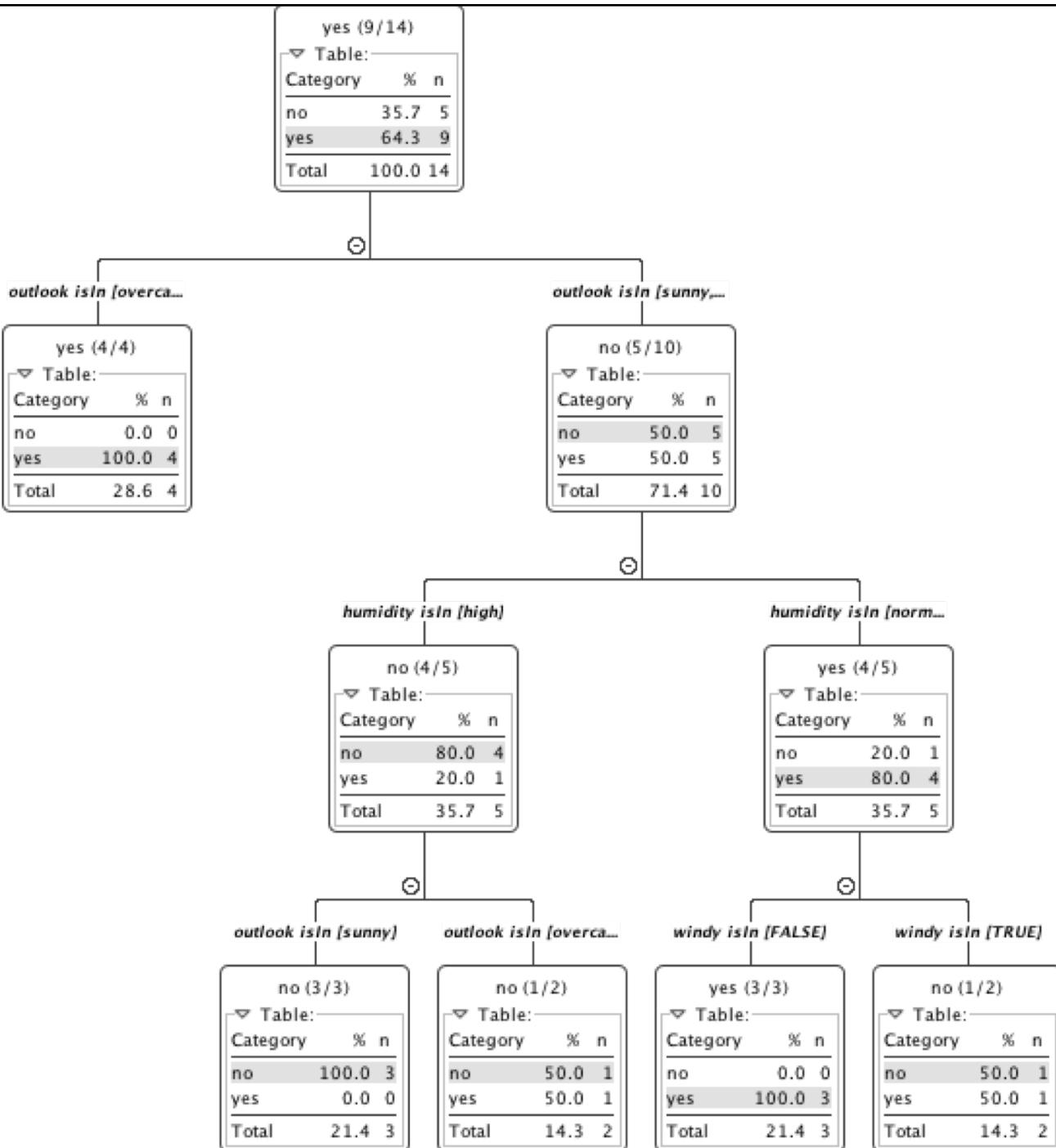
$$\begin{aligned}Gini(D_{r,s}, D_o) &= \frac{10}{14}Gini(D_{r,s}) + \frac{4}{14}Gini(D_o) \\&= \frac{10}{14}Gini([5, 5]) + \frac{4}{14}Gini([4, 0]) \\&= \frac{10}{14}0.5 + \frac{4}{14}0.0 \\&= 0.357\end{aligned}$$

- {sunny, overcast} and {rainy}

$$\begin{aligned}Gini(D_{s,o}, D_r) &= \frac{6}{3}Gini(D_{s,o}) + \frac{3}{2}Gini(D_r) \\&= \frac{6}{3}Gini([6, 3]) + \frac{3}{2}Gini([3, 2]) \\&= \frac{6}{3}0.444 + \frac{3}{2}0.480 \\&= 0.457\end{aligned}$$

- The attribute partition with the largest gain is {overcast, rainy} and {sunny} that correspond to a Gini of 0.394 and an overall gain of

$$Gini(D) - Gini(D_{r,s}, D_o) = 0.459 - 0.357 = 0.102$$



# Generalization and overfitting in trees

- The generated tree may overfit the training data
- Too many branches, some may reflect anomalies due to noise or outliers
- Result is in poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning
  - Postpruning

- 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”

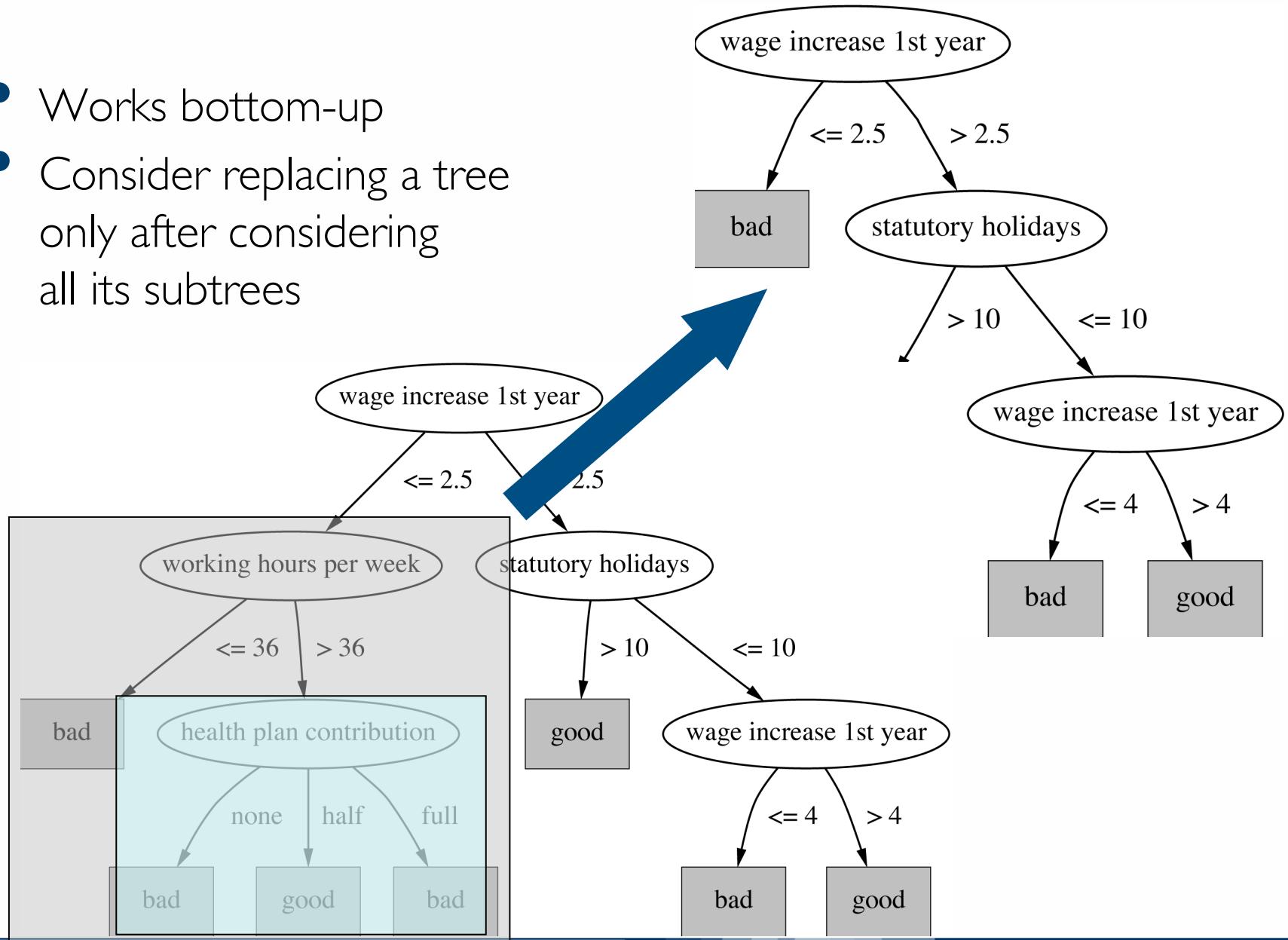
- Based on statistical significance test
- Stop growing the tree when there is no statistically significant association between any attribute and the class at a particular node
- The most popular approach is the chi-squared test
- Quinlan's classic tree learner ID3 used chi-squared test in addition to information gain
- Only statistically significant attributes were allowed to be selected by the information gain procedure

- First, build full tree, then prune it
- Fully-grown tree shows all attribute interactions
- Problem: some subtrees might be due to chance effects
- Two pruning operations
  - Subtree raising
  - Subtree replacement
- Possible strategies
  - Error estimation
  - Significance testing
  - MDL principle

# Subtree Replacement

54

- Works bottom-up
- Consider replacing a tree only after considering all its subtrees

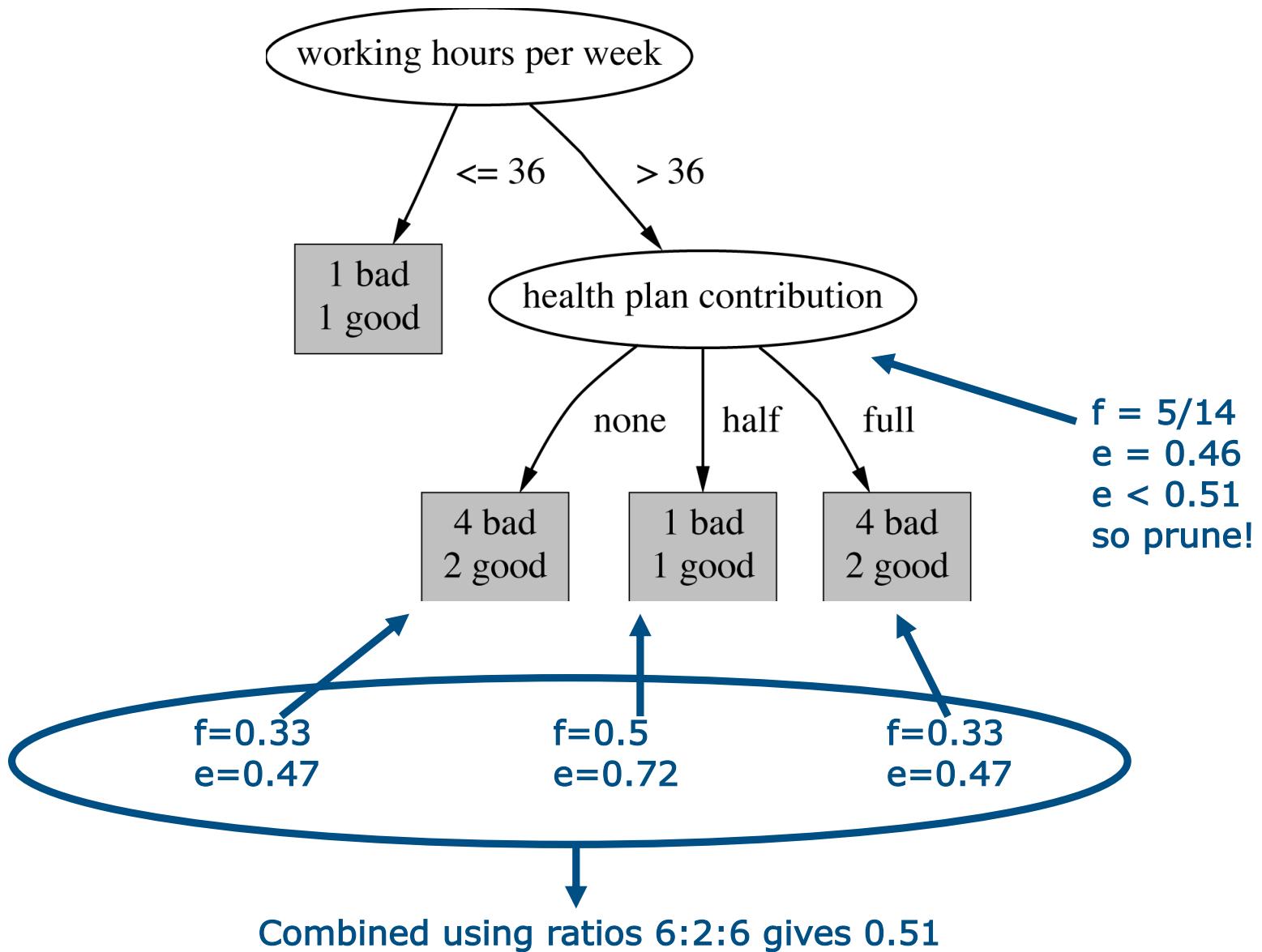


- Prune only if it reduces the estimated error
- Error on the training data is NOT a useful estimator  
(Why it would result in very little pruning?)
- A hold-out set might be kept for pruning  
("reduced-error pruning")
- Example (C4.5's method)
  - Derive confidence interval from training data
  - Use a heuristic limit, derived from this, for pruning
  - Standard Bernoulli-process-based method
  - Shaky statistical assumptions (based on training data)

- Given the error  $f$  on the training data, the upper bound for the error estimate for a node is computed as

$$e = \left( f + \frac{z^2}{2N} + z \sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) \Bigg/ \left( 1 + \frac{z^2}{N} \right)$$

- If  $c = 25\%$  then  $z = 0.69$  
  - $f$  is the error on the training data
  - $N$  is the number of instances covered by the leaf



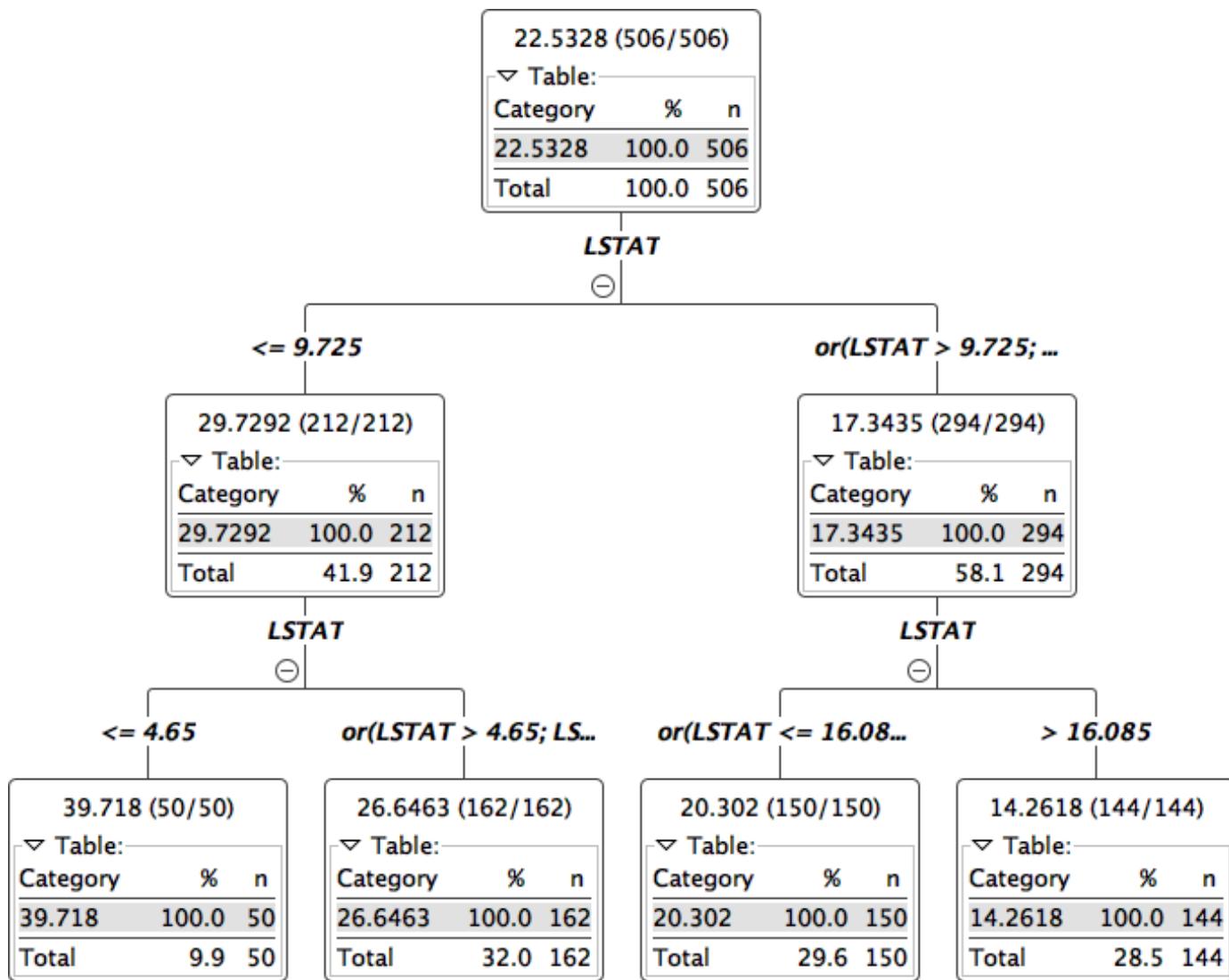
# Regression & Model Trees

Decision trees can also be used to predict the value of a numerical target variable

Regression and model trees work similarly to decision trees

They search for the best split that minimizes an impurity measure

- What prediction?
  - Prediction is computed as the average of numerical target variable in the subspace (regression trees)
  - Alternatively leafs can contain a linear model to predict the target value in the corresponding subspace (model trees)
- What impurity measure?
  - It can be measured as the expected error reduction, or SDR (standard deviation reduction)
$$SDR = \sigma(D) - \sum_i \frac{|D_i|}{|D|} \sigma(D_i)$$
  - D is the original data  $D_i$  are the partitions and  $\sigma$  is the standard deviation of the target attribute in the set



Input variable: LSTAT - % lower status of the population

Output variable: MEDV - Median value of owner-occupied homes in \$1000's

# Decision Stumps

- Decision stumps are one level decision trees
- They are the simplest decision trees possible and also the main building blocks for boosting methods
- Categorical Attributes
  - One branch for each attribute value
  - One branch for one value and one branch for all the others
  - Missing values sometimes are treated as a special value
- Numerical Attributes
  - Two leaves defined by a threshold value selected based on some criterion
  - Multiple splits (rarely used)



- **ADTree**
  - Builds alternating decision trees
- **BFTree**
  - Builds decision trees using a best-first search
- **FT**
  - Builds a functional trees with oblique splits and linear functions at the leaves
- **LADTree**
  - Builds a multiclass alternating decision trees using LogitBoost
- **LMT**
  - Builds logistic model trees
- **NBTree**
  - Builds a decision tree classifier with a Naïve Bayes classifier at the leaves
- ...

run the Python notebooks and the KNIME workflows on decision trees provided

# Summary

- Decision tree construction is a recursive procedure involving
  - The selection of the best splitting attribute
  - (and thus) a selection of an adequate purity measure (Information Gain, Information Gain Ratio, Gini Index)
  - Pruning to avoid overfitting
- Information Gain is biased toward highly splitting attributes
- Information Gain Ratio takes the number of splits that an attribute induces into the picture but might not be enough

- Try building the tree for the nominal version of the Weather dataset using the Gini Index
- Compute the best upper bound you can for the computational complexity of the decision tree building
- Check the problems given in the previous year exams on the use of Information Gain, Gain Ratio and Gini index, solve them and check the results using Weka.