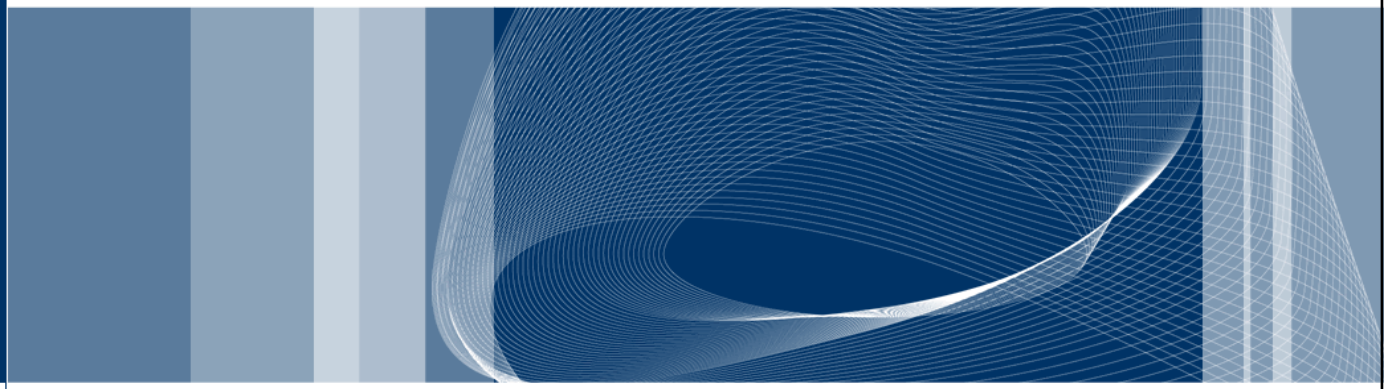
POLITECNICO DI MILANO



Decision Trees

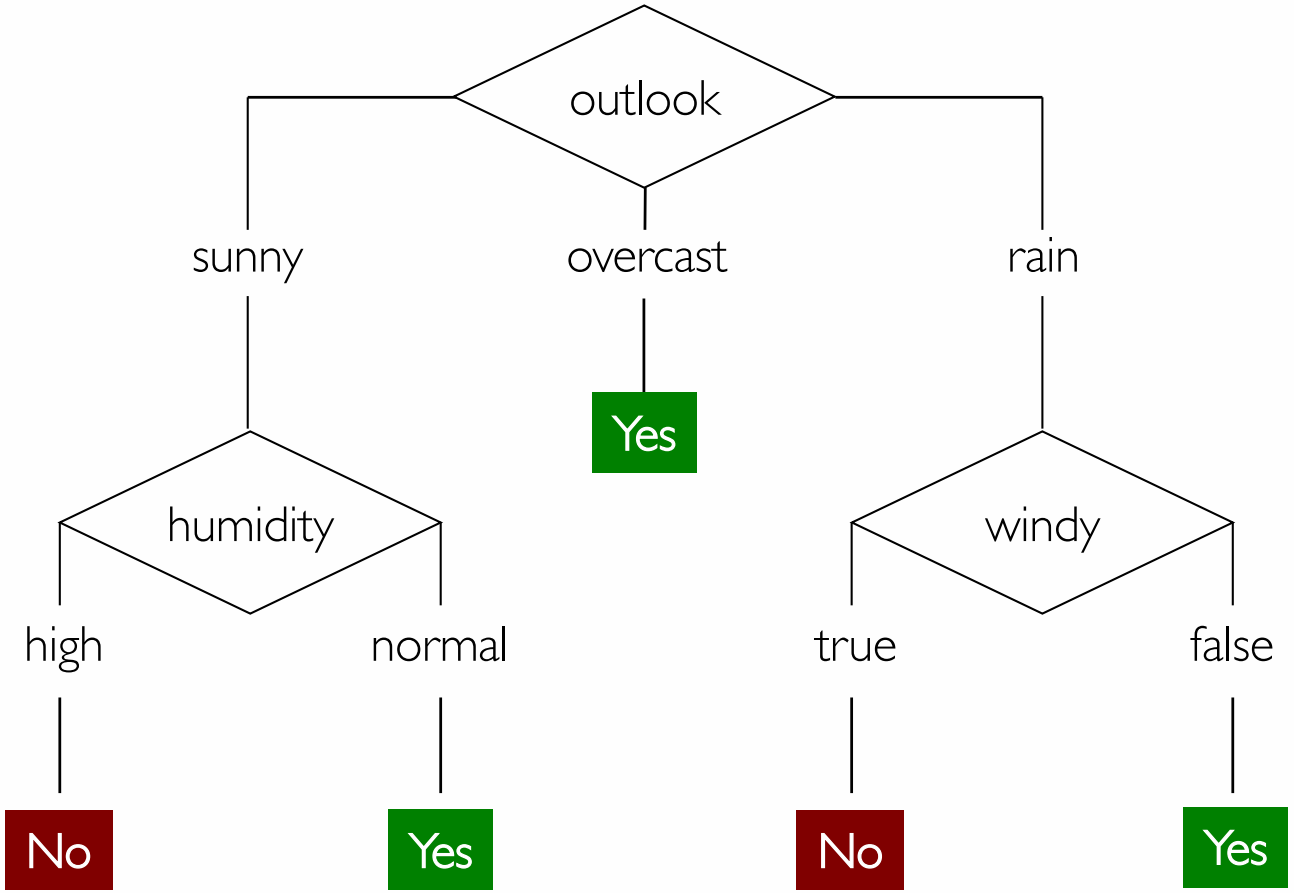
Data Mining and Text Mining

Prof. Pierluca Lanzi

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

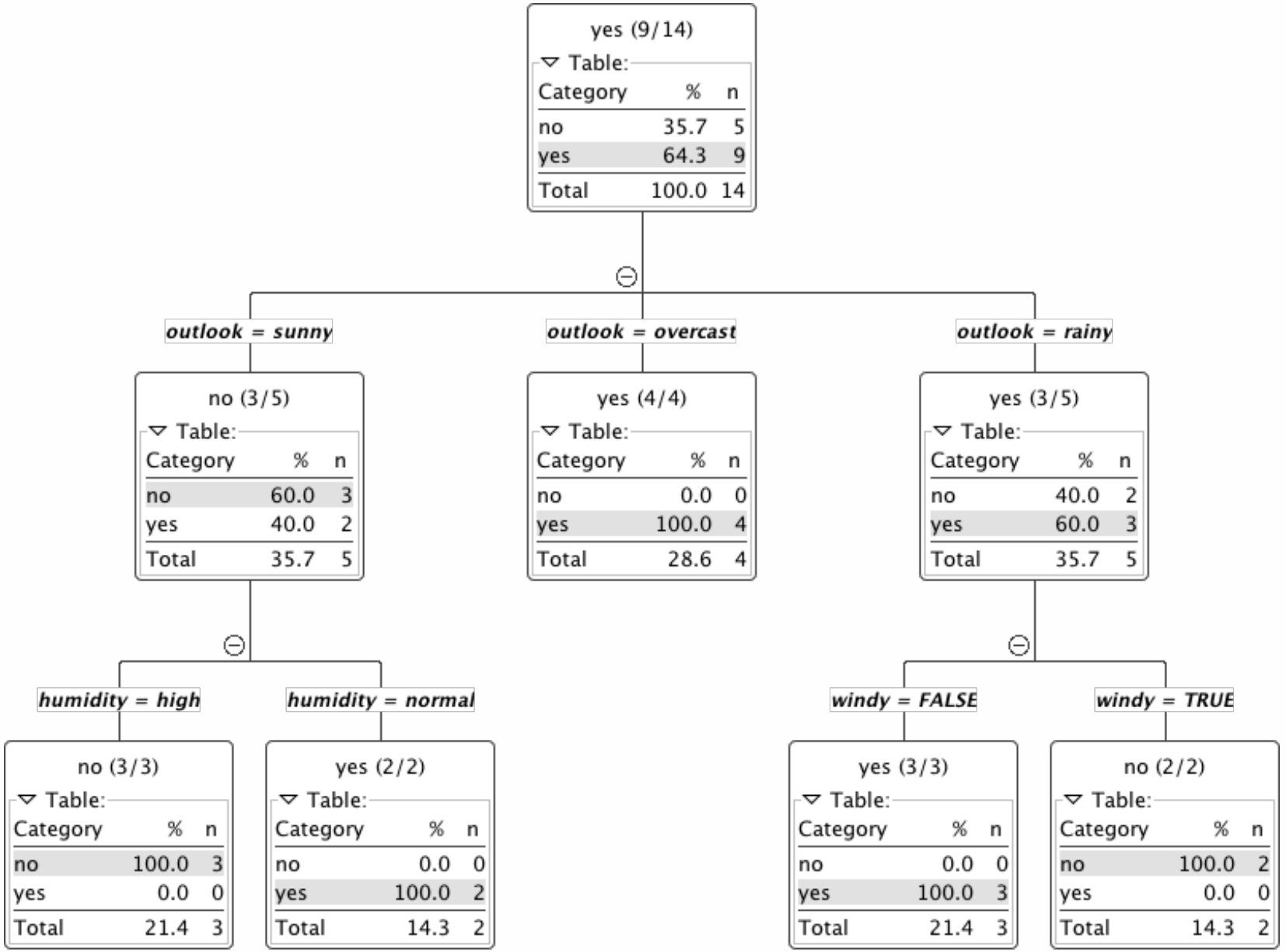
The Decision Tree for the Weather Dataset



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

Decision Tree Representations (Graphical)



Computing Decision Trees

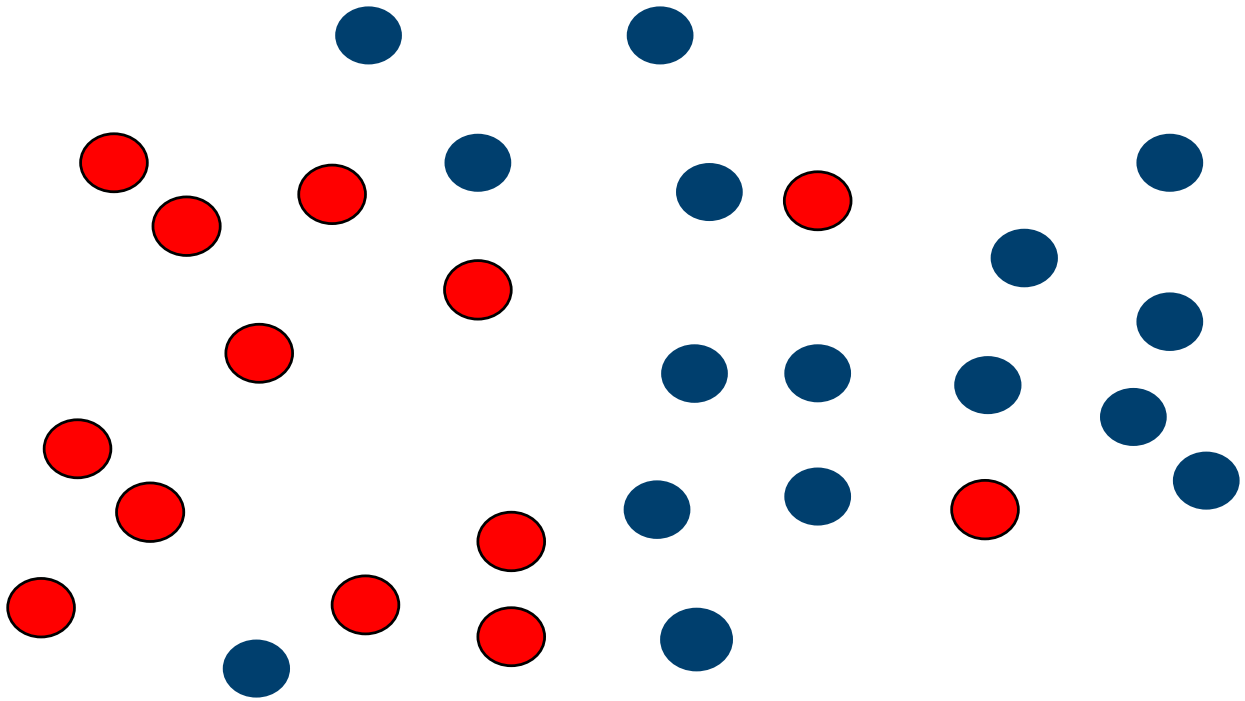
Computing Decision Trees

- 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

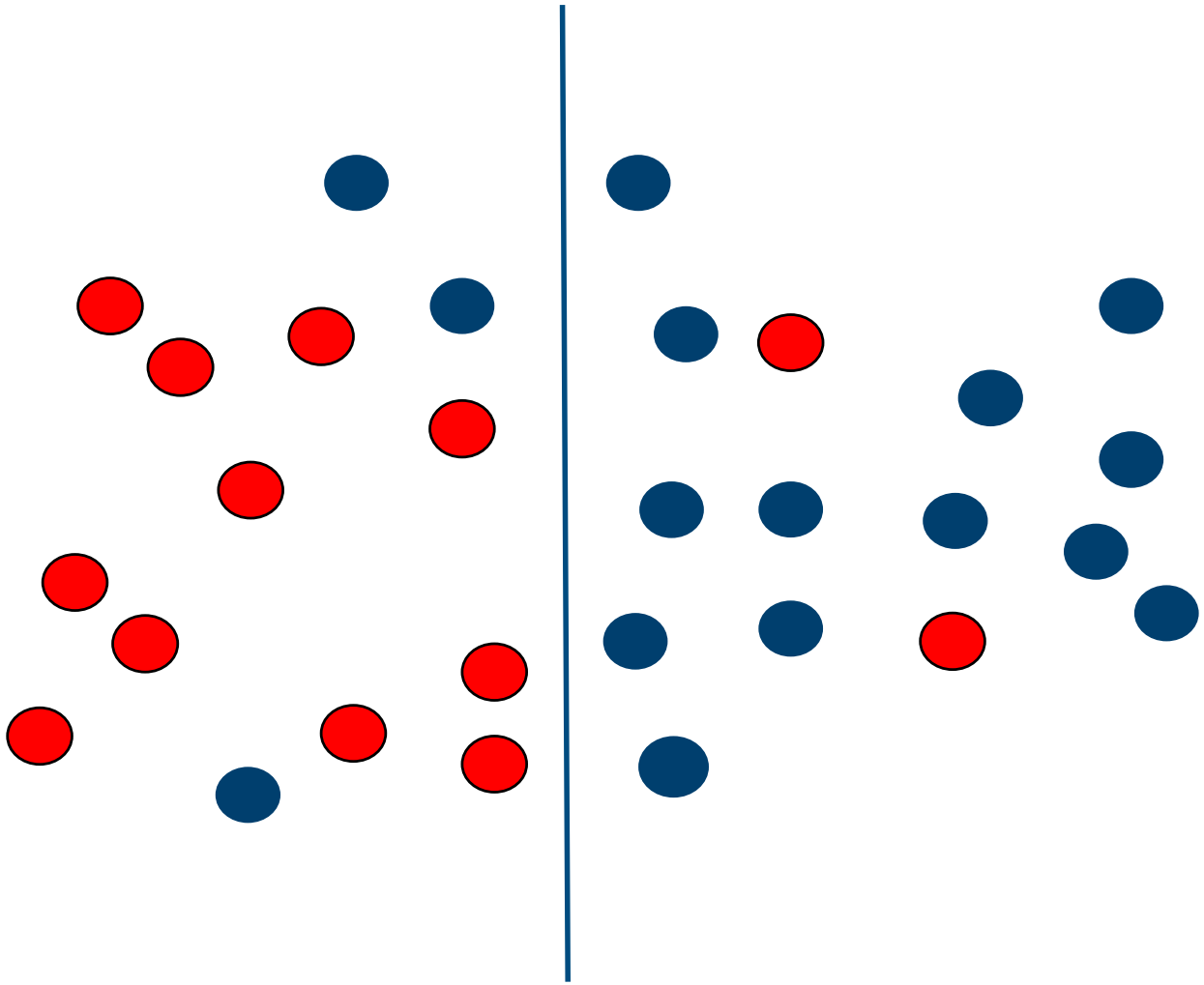
- Bottom-up Tree Pruning
  - Remove subtrees or branches, in a bottom-up manner, to improve the estimated accuracy on new cases.

What Splitting Attribute?

We are looking for a way to separate example to create areas that are “pure” and contain mainly examples of one class



There are 13 red labels and 16 blue labels, using the majority class unlabeled point would be labeled as blue. Training accuracy for baseline classifier is 16/29 that is, 55%

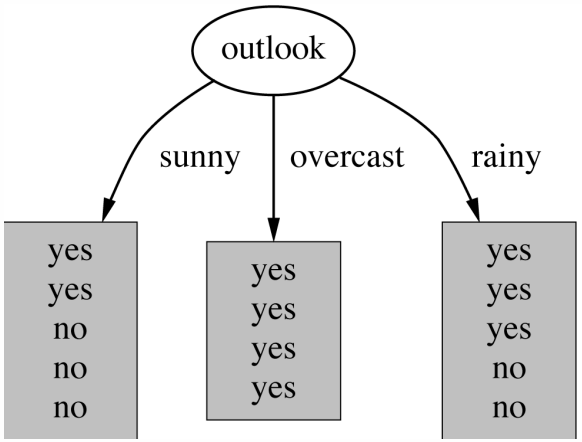


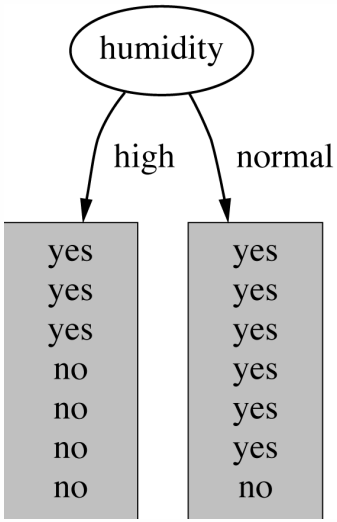
By building this simple model, using majority class on the two subspaces we would have a training accuracy of 11/14 (78.5%) on the left and 13/15 (86.6%) on the right. Much better! We can combine in an overall performance of 78.5x14/29 + 86.6%x15/29 that is 82.6%

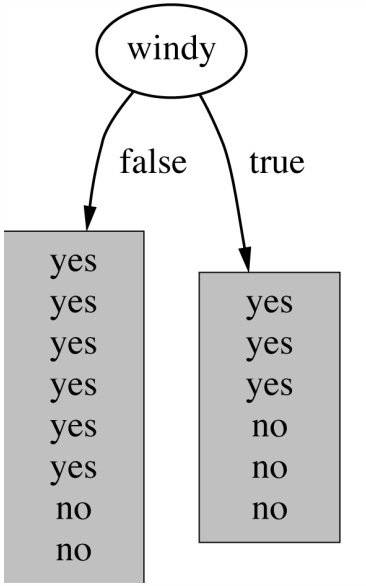
Which Attribute for Splitting?

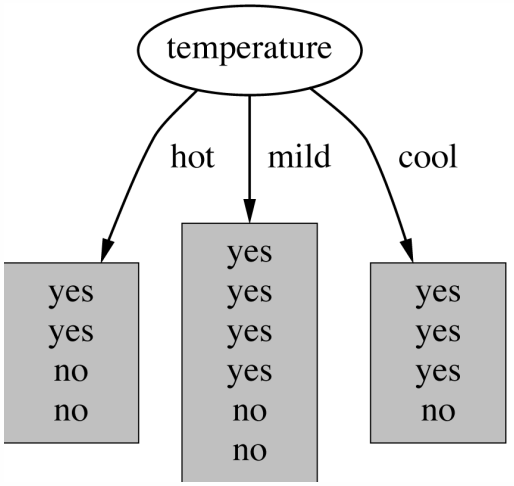
- At each node, available attributes are evaluated based on separating the classes of the training examples using either a purity or impurity measure
- Typical measures used are the information gain (ID3), information gain ratio (C4.5), gini index (CART)
- Information Gain increases with the average purity of the subsets that an attribute produces. It selects the attribute with the highest information gain

Which Attribute Should We Select?









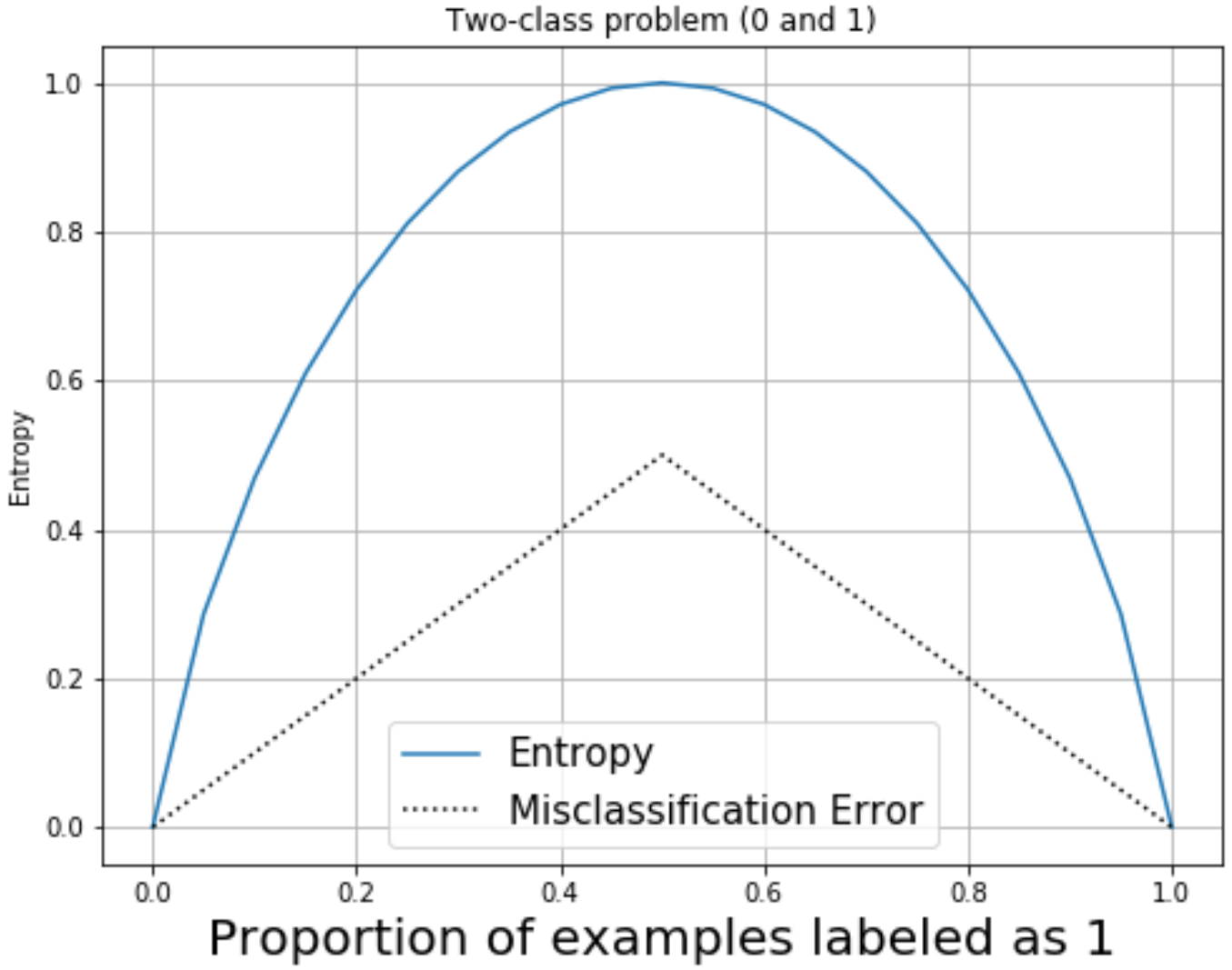
Computing Information

- 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!)

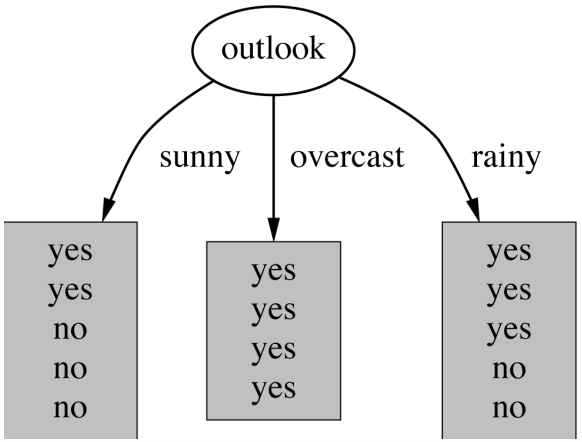
- Formula for computing the entropy

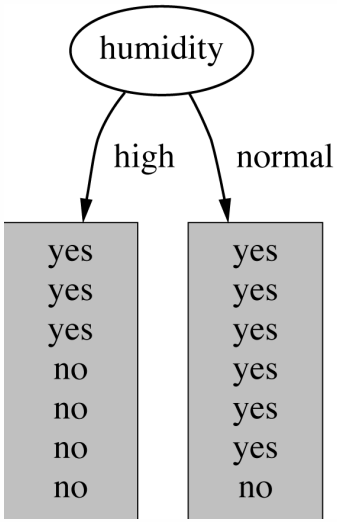
$$entropy(p_1, p_2, \ldots, p_n) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 \cdots - p_n \log_2 p_n$$

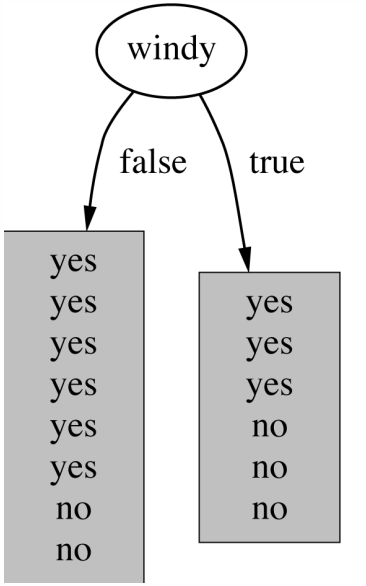
- Note that, since entropy refers to the amount of information expressed in bits, it uses the logarithm with base 2

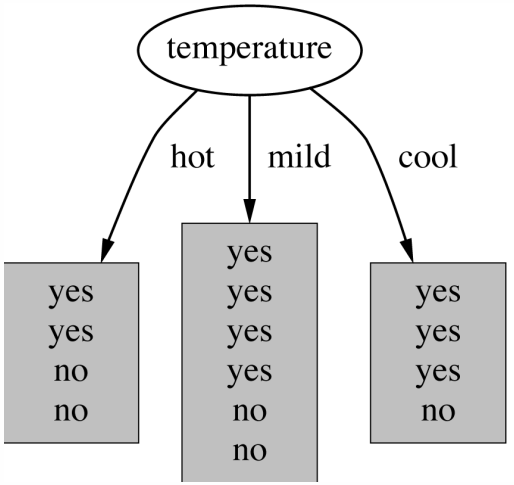

Two-class problem (0 and 1)
Entropy
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Entropy
Misclassification Error
Proportion of examples labeled as 1

Which Attribute Should We Select?









The Attribute "outlook"

- "outlook" = "sunny" $info([2,3]) = entropy(2/5, 3/5) = 0.971$
- "outlook" = "overcast" $info([4,0]) = entropy(1,0) = 0.000$
- "outlook" = "rainy" $info([3,2]) = entropy(3/5, 2/5) = 0.971$
- Expected information for attribute

$$info([2,3][4,0][3,2]) = 5/14 \times 0.971 + 4/14 \times 0 + 5/14 \times 0.971$$

- As previously noted, since entropy refers to the amount of information expressed in bits, it uses the logarithm with base 2

Information Gain

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

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

- The information before the split is the entropy of the class distribution in D,

$$info(D) = -p_1 \log_2 p_1 \cdots - p_n \log_2 p_n$$

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

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

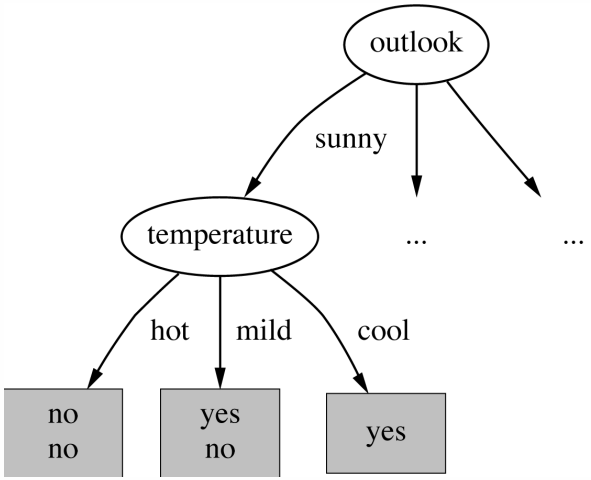
Information Gain

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

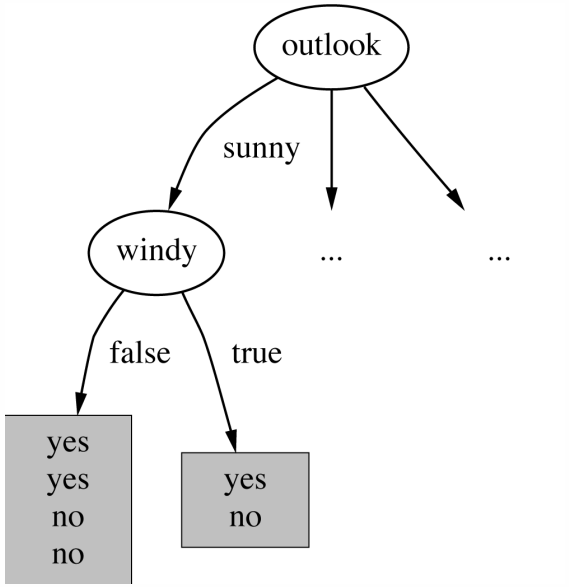
$$\begin{aligned} gain(outlook) &= info([9,5]) - info([2,3][4,0][3,2]) \\ &= 0.940 - 0.693 \\ &= 0.247 \end{aligned}$$

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

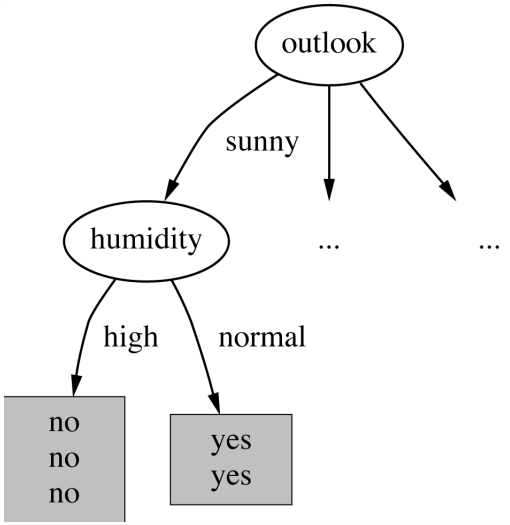
Going Further



$$gain(temperature) = 0.571$$



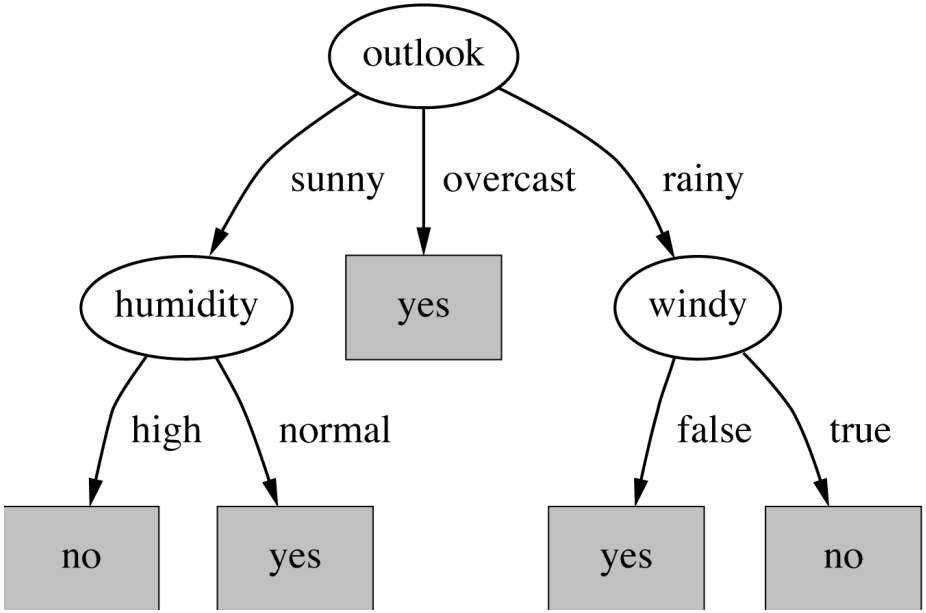
$$gain(windy) = 0.020$$



$$gain(humidity) = 0.971$$

The Final Decision Tree

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

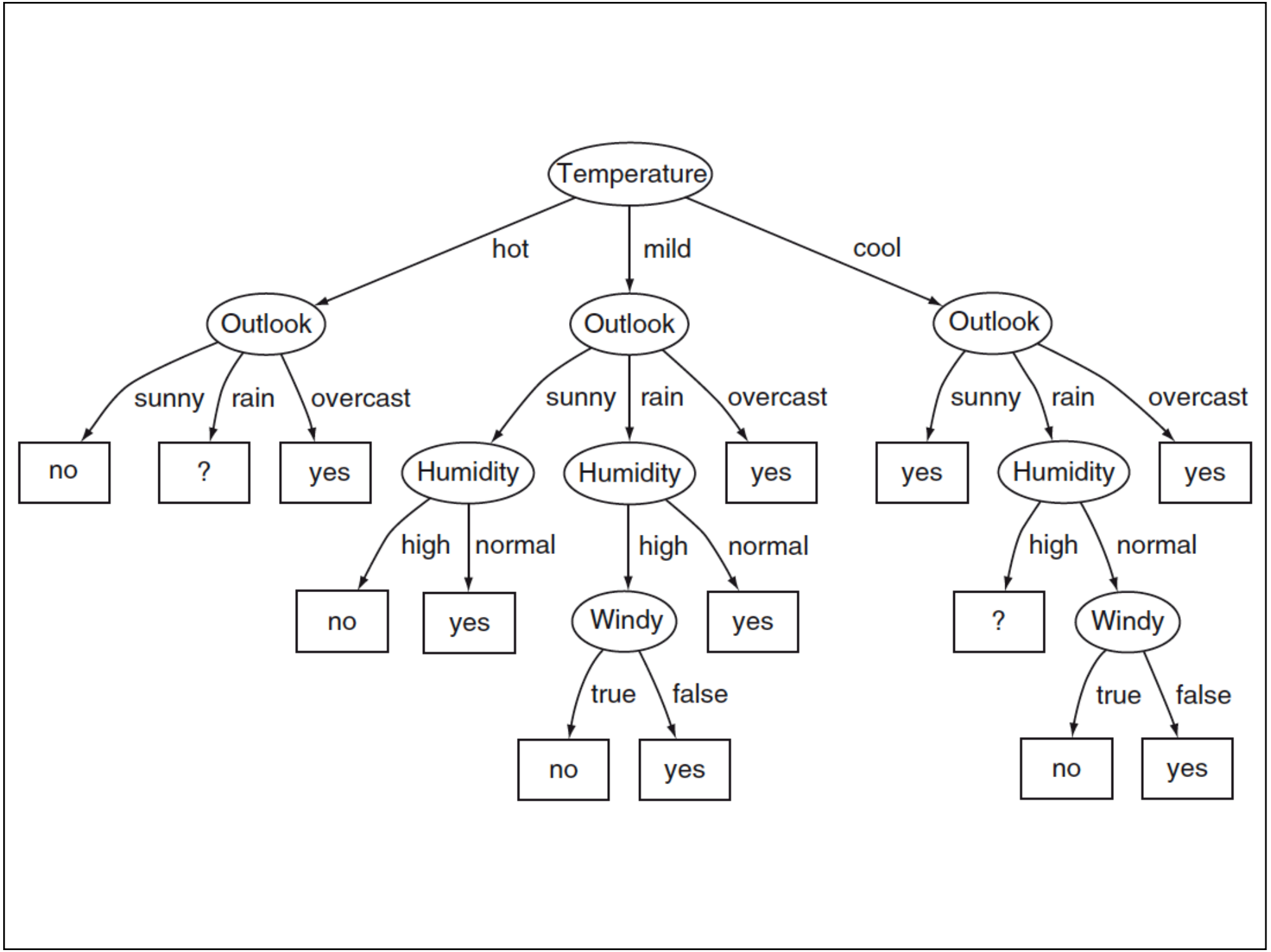


When Should Building Stop?

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

We Can Always Build
a 100% Accurate Tree

But do we want it?

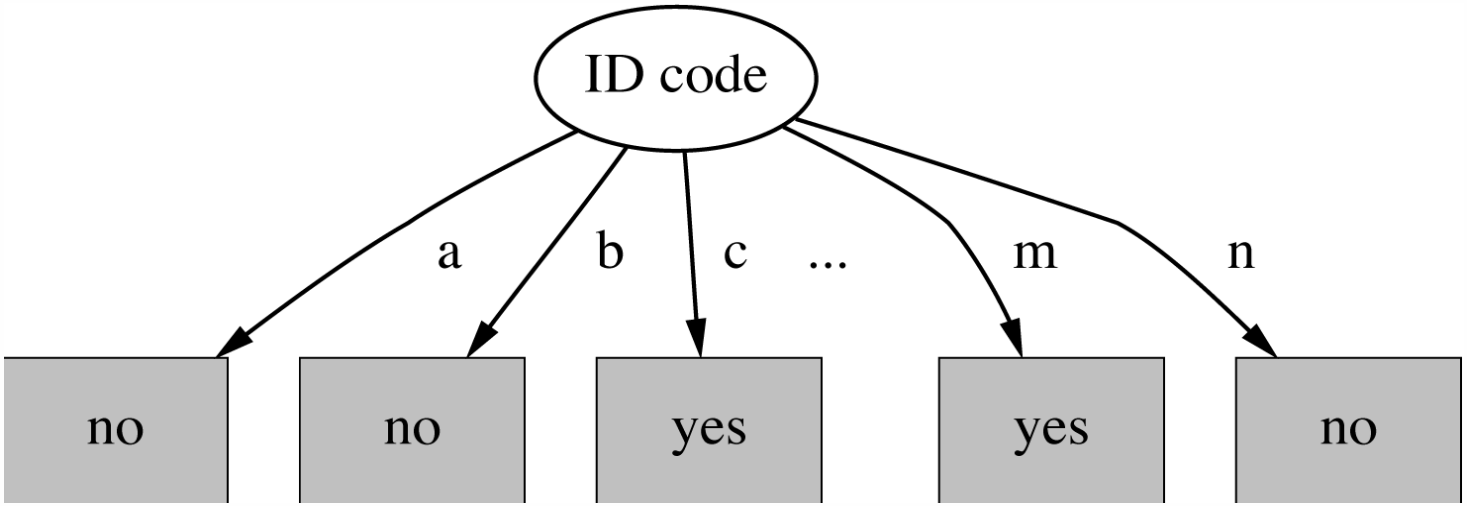

Temperature
hot
mild
cool
Outlook
Outlook
Outlook
sunny
rain
overcast
sunny
rain
overcast
sunny
rain
overcast
no
?
yes
Humidity
Humidity
yes
yes
Humidity
yes
high
normal
high
normal
high
normal
no
yes
Windy
yes
?
Windy
true
false
true
false
no
yes
no
yes

Many-valued Attributes

Another Version of the Weather Dataset

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

Decision Tree for the New Dataset



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

Highly-Branching Attributes

- 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 rewards attributes with many values
- This may result in overfitting (selection of an attribute that is non-optimal for prediction and thus overfits)

Information Gain Ratio

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

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

computes the entropy of distribution of instances into branches

- This is the entropy of A, independent of the class

- Information Gain Ratio normalizes Information Gain by

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

Computing the Information Gain Ratio

- The intrinsic information for ID code is

$$info([1/14, \ldots, 1/14]) = 14 \times (-1/14 \log_2 1/14) = 3.807$$

- 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

Outlook	
Information after split:	0.693
Gain: 0.940-0.693	0.247
Split info: info([5,4,5])	1.577
Gain ratio: 0.247/1.577	0.156

Temperature	
Information after split:	0.911
Gain: 0.940-0.911	0.029
Split info: info([4,6,4])	1.362
Gain ratio: 0.029/1.362	0.021

Humidity	
Information after split:	0.788
Gain: 0.940-0.788	0.152
Split info: info([7,7])	1.000
Gain ratio: 0.152/1	0.152

Windy	
Information after split:	0.892
Gain: 0.940-0.892	0.048
Split info: info([8,6])	0.985
Gain ratio: 0.048/0.985	0.049

More on Information Gain Ratio

- "outlook" still best compared to other original attributes
- however "ID code" has even greater gain ratio
  - despite likely not being a reliable attribute to branch on
  - standard fix is an ad-hoc test to prevent splitting on that type of attribute
- In practice, tree learners often use both IG and IGR:
  - First, only consider attributes with greater than average Information Gain;
  - Then, compare them using the Information Gain Ratio
- Why?
  - Information Gain Ratio may overcompensate and choose an attribute just because its intrinsic information is very low

Gini Index

Another Splitting Criteria: The Gini Index

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

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

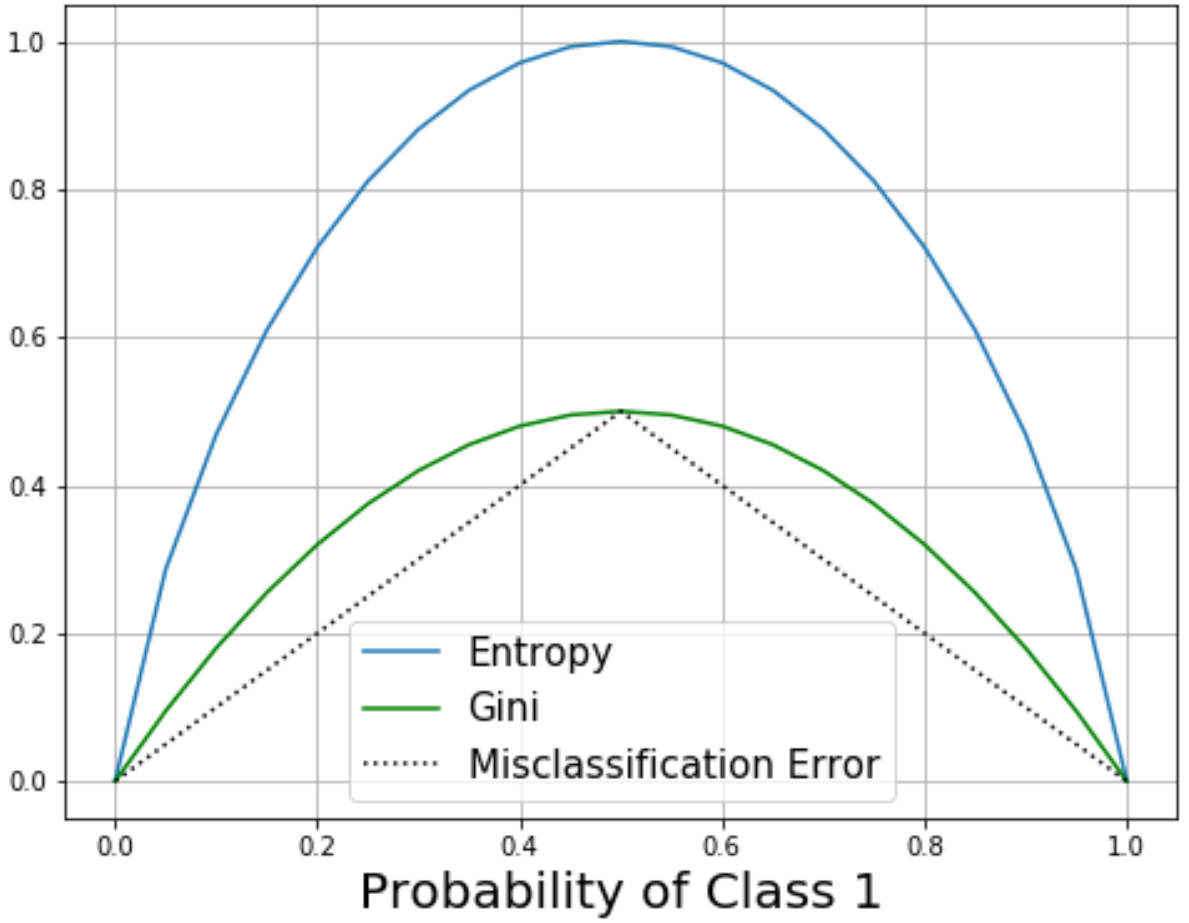
where p_j is the relative frequency of class j in T

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

- gini measures error rate of random classifier that assigns classes to instances according to their prior frequencies:

$$gini(T) = \sum_j p_j(1-p_j) = \sum_j p_j(1-p_j) = \sum_j p_j - p_j^2 = 1 - \sum_j p_j^2$$

The Gini Index vs Entropy



The Gini Index

- 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 Gini Index for the Outlook Attribute

- The dataset has 9 tuples labeled "yes" and 5 labeled "no"

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

- 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}

The Gini Index for the Outlook Attribute

- {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}

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

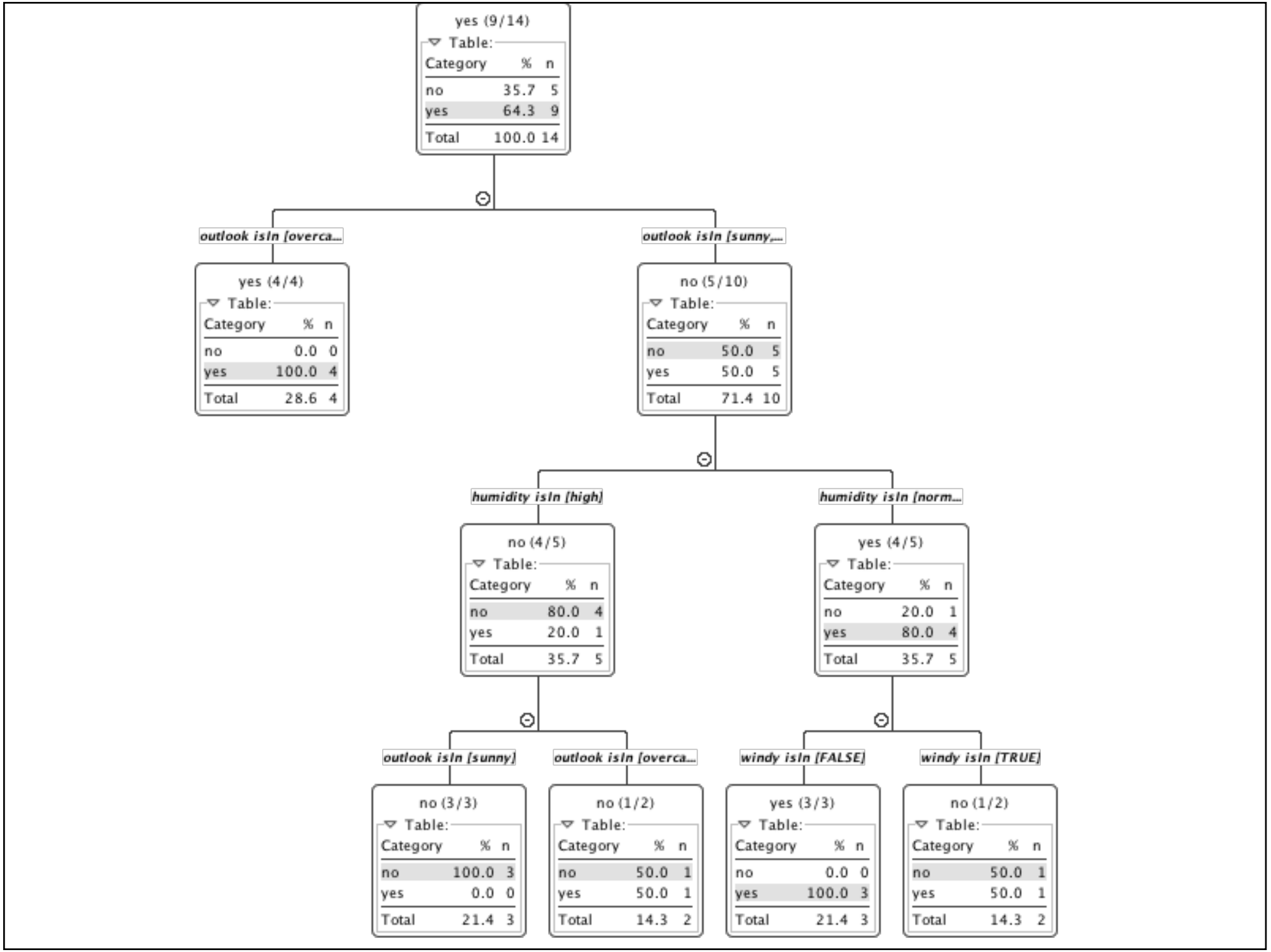
The Gini Index for the Outlook Attribute

- {sunny, overcast} and {rainy}

$$
\begin{aligned}
Gini(D_{s,o}, D_r) &= \frac{9}{14} Gini(D_{s,o}) + \frac{5}{14} Gini(D_r) \\
&= \frac{9}{14} Gini([6,3]) + \frac{5}{14} Gini([3,2]) \\
&= \frac{9}{14} 0.444 + \frac{5}{14} 0.480 \\
&= 0.457
\end{aligned}
$$

- The attribute partition with the largest gain is {rainy, sunny} and {overcast} that correspond to a Gini of 0.357 and an overall gain of

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


yes (9/14)
Table:
Category % n
no 35.7 5
yes 64.3 9
Total 100.0 14
outlook isIn [overca...
outlook isIn [sunny,...
yes (4/4)
Table:
Category % n
no 0.0 0
yes 100.0 4
Total 28.6 4
no (5/10)
Table:
Category % n
no 50.0 5
yes 50.0 5
Total 71.4 10
humidity isIn [high]
humidity isIn [norm...
no (4/5)
Table:
Category % n
no 80.0 4
yes 20.0 1
Total 35.7 5
yes (4/5)
Table:
Category % n
no 20.0 1
yes 80.0 4
Total 35.7 5
outlook isIn [sunny]
outlook isIn [overca...
windy isIn [FALSE]
windy isIn [TRUE]
no (3/3)
Table:
Category % n
no 100.0 3
yes 0.0 0
Total 21.4 3
no (1/2)
Table:
Category % n
no 50.0 1
yes 50.0 1
Total 14.3 2
yes (3/3)
Table:
Category % n
no 0.0 0
yes 100.0 3
Total 21.4 3
no (1/2)
Table:
Category % n
no 50.0 1
yes 50.0 1
Total 14.3 2

Numerical Attributes

The Weather Dataset (Numerical)

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

The Temperature Attribute

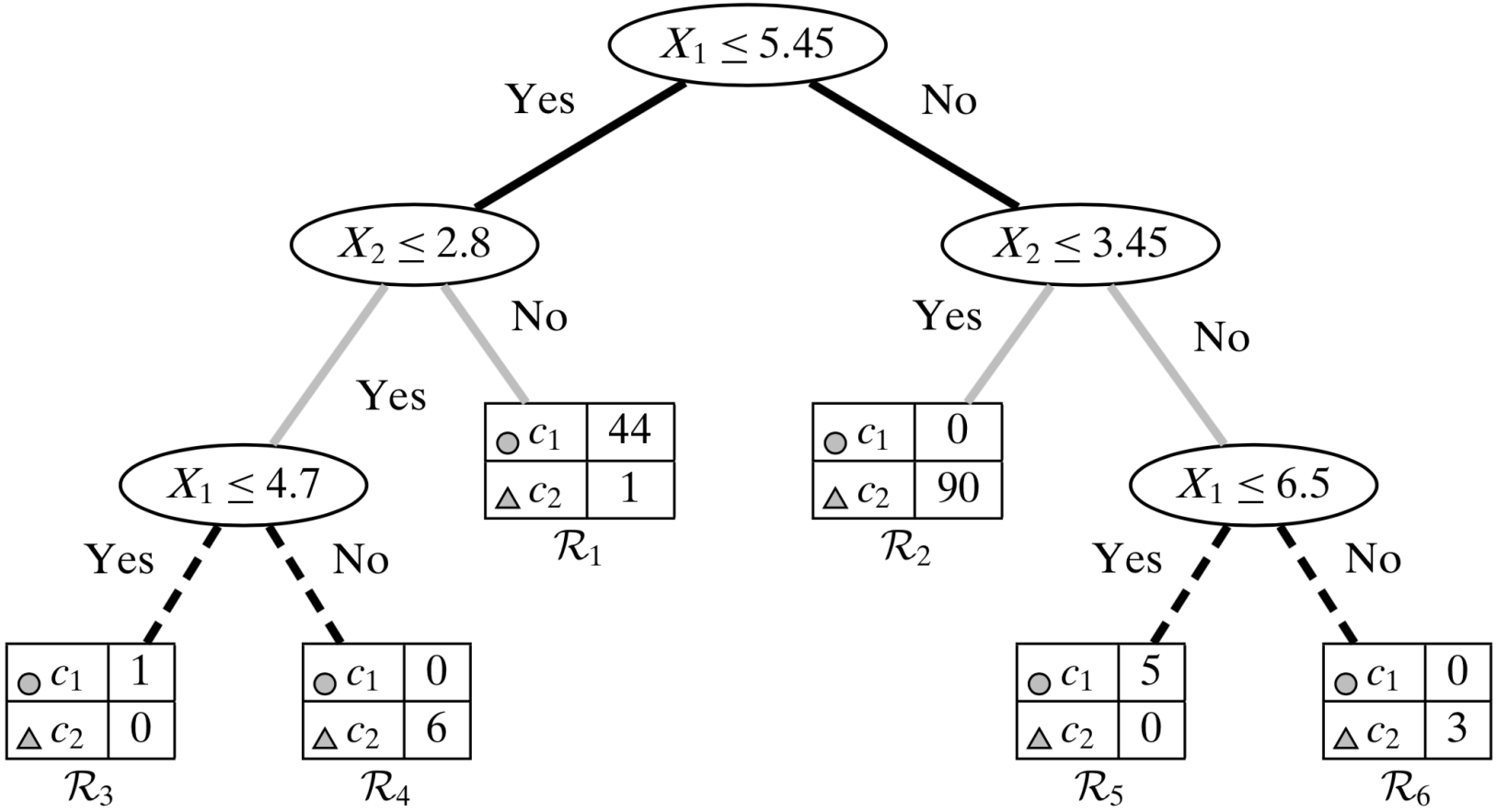
- First, sort the temperature values, including the class labels
- Then, check all the feasible cut points and choose the one with the best information gain

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 ≤ 70.5: yes/4, no/1
  temperature > 70.5: yes/5, no/4

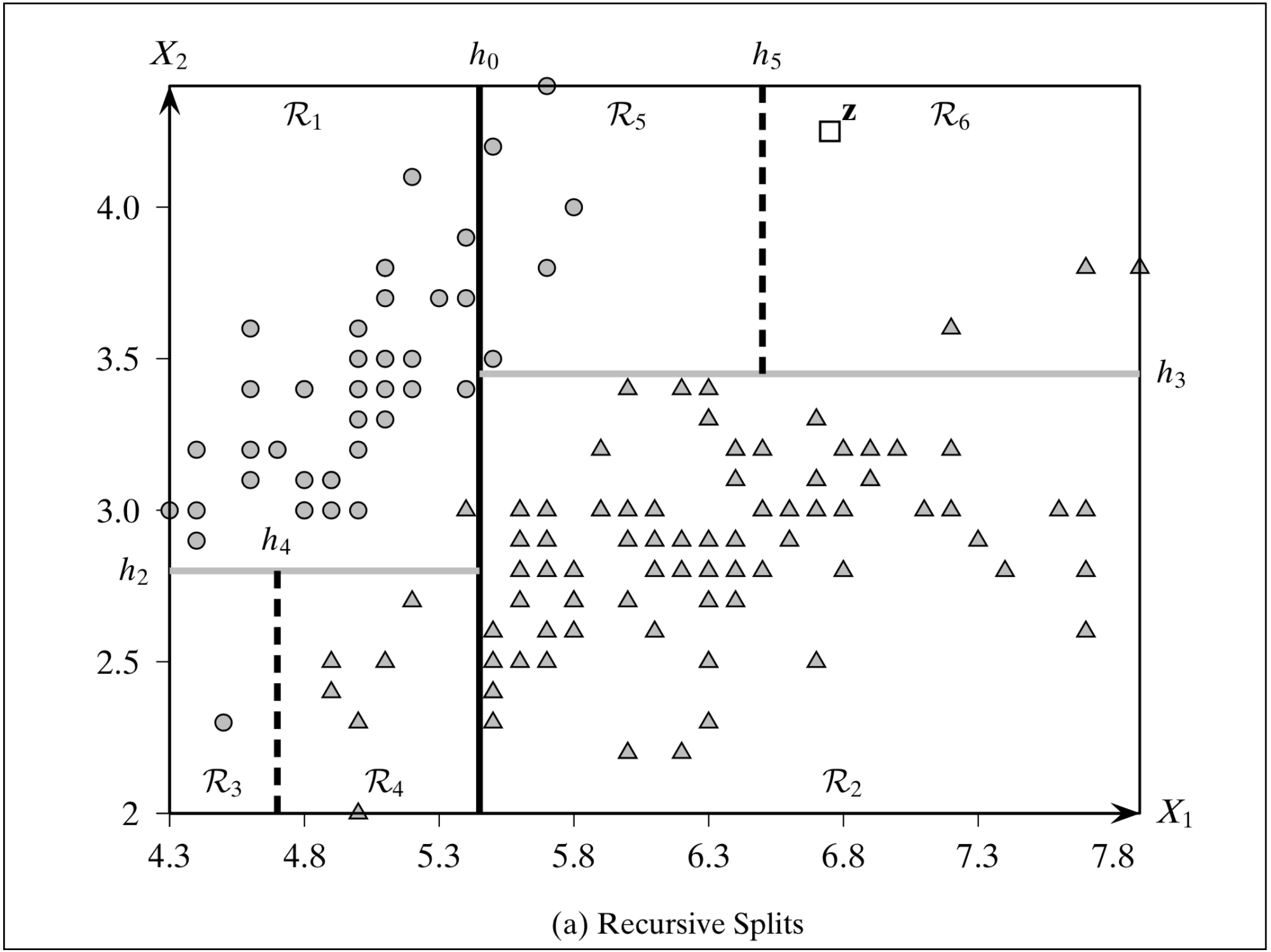
- Info([4,1],[5,4]) = 5/14 info([4,1]) + 9/14 info([5,4]) = 0.894
- Information gain for temperature at 70.5 is thus 0.045

Geometric Interpretation



(b) Decision Tree

Figure 19.1. Decision trees: recursive partitioning via axis-parallel hyperplanes.



(a) Recursive Splits

We can avoid repeated sorting

- 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

Binary vs Multiway splits

- 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

ALGORITHM 19.1. Decision Tree Algorithm

```
DECISIONTREE (D, η, π):
1  n ← |D| // partition size
2  n_i ← |{x_j | x_j ∈ D, y_j = c_i}| // size of class c_i
3  purity(D) ← max_i {n_i / n}
4  if n ≤ η or purity(D) ≥ π then // stopping condition
5      c* ← arg max_{c_i} {n_i / n} // majority class
6      create leaf node, and label it with class c*
7      return
8  (split point*, score*) ← (∅, 0) // initialize best split point
9  foreach (attribute X_j) do
10     if (X_j is numeric) then
11         (v, score) ← EVALUATE-NUMERIC-ATTRIBUTE(D, X_j)
12         if score > score* then (split point*, score*) ← (X_j ≤ v, score)
13     else if (X_j is categorical) then
14         (V, score) ← EVALUATE-CATEGORICAL-ATTRIBUTE(D, X_j)
15         if score > score* then (split point*, score*) ← (X_j ∈ V, score)
   // partition D into D_Y and D_N using split point*, and call
      recursively
16 D_Y ← {x ∈ D | x satisfies split point*}
17 D_N ← {x ∈ D | x does not satisfy split point*}
18 create internal node split point*, with two child nodes, D_Y and D_N
19 DECISIONTREE(D_Y); DECISIONTREE(D_N)
```

Generalization and overfitting in trees

Avoiding Overfitting in Decision 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

Pre-pruning vs Post-pruning

- 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"

Prepruning

- 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

Post-Pruning

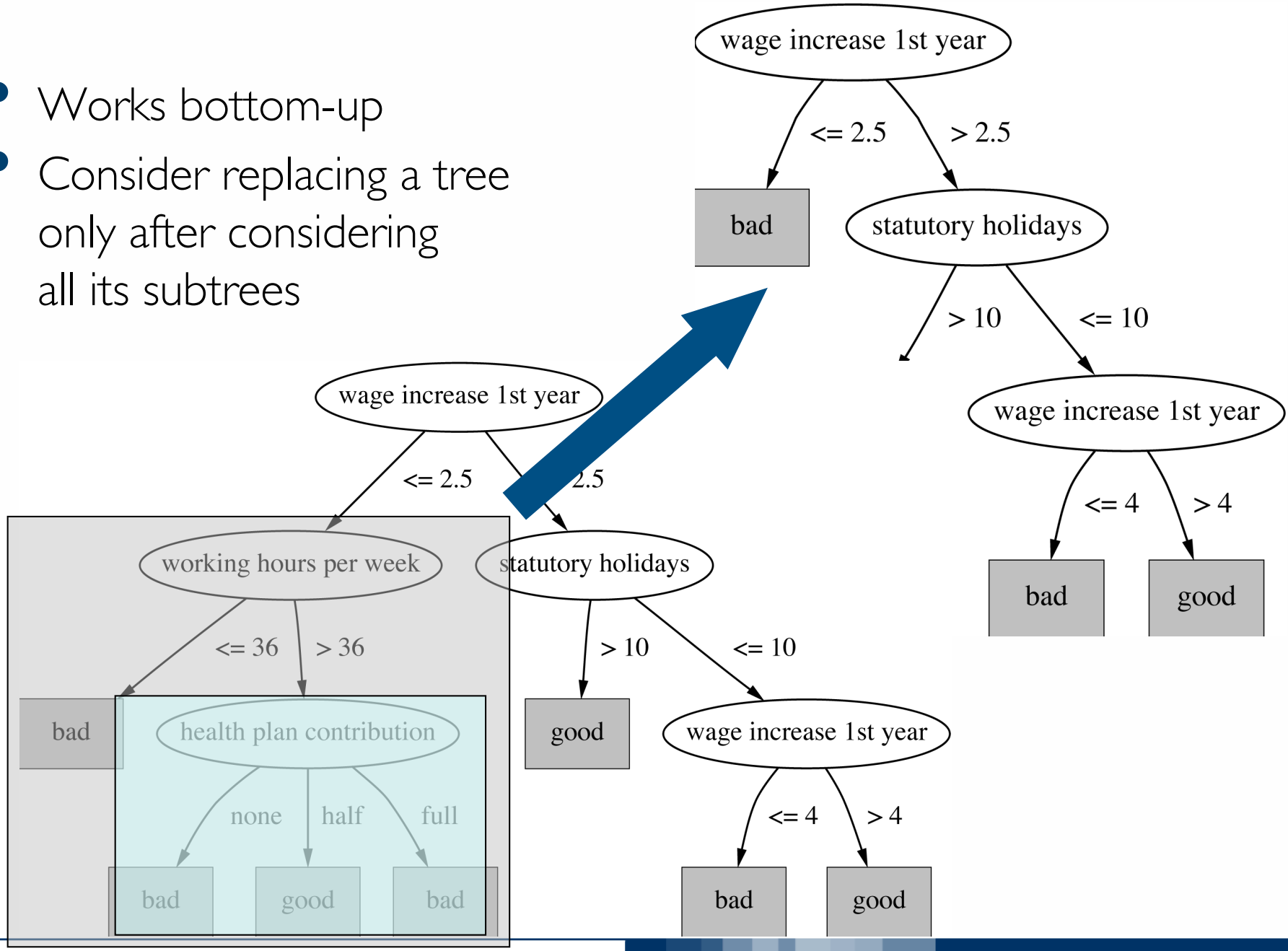
- First, build full tree, then prune it
- Fully-grown tree shows all attribute interactions
- Problem: some subtrees might be due to chance effects

- Two pruning operations
  - Subtree raising
  - Subtree replacement

- Possible strategies
  - Error estimation
  - Significance testing
  - MDL principle

Subtree Replacement

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



Estimating Error Rates

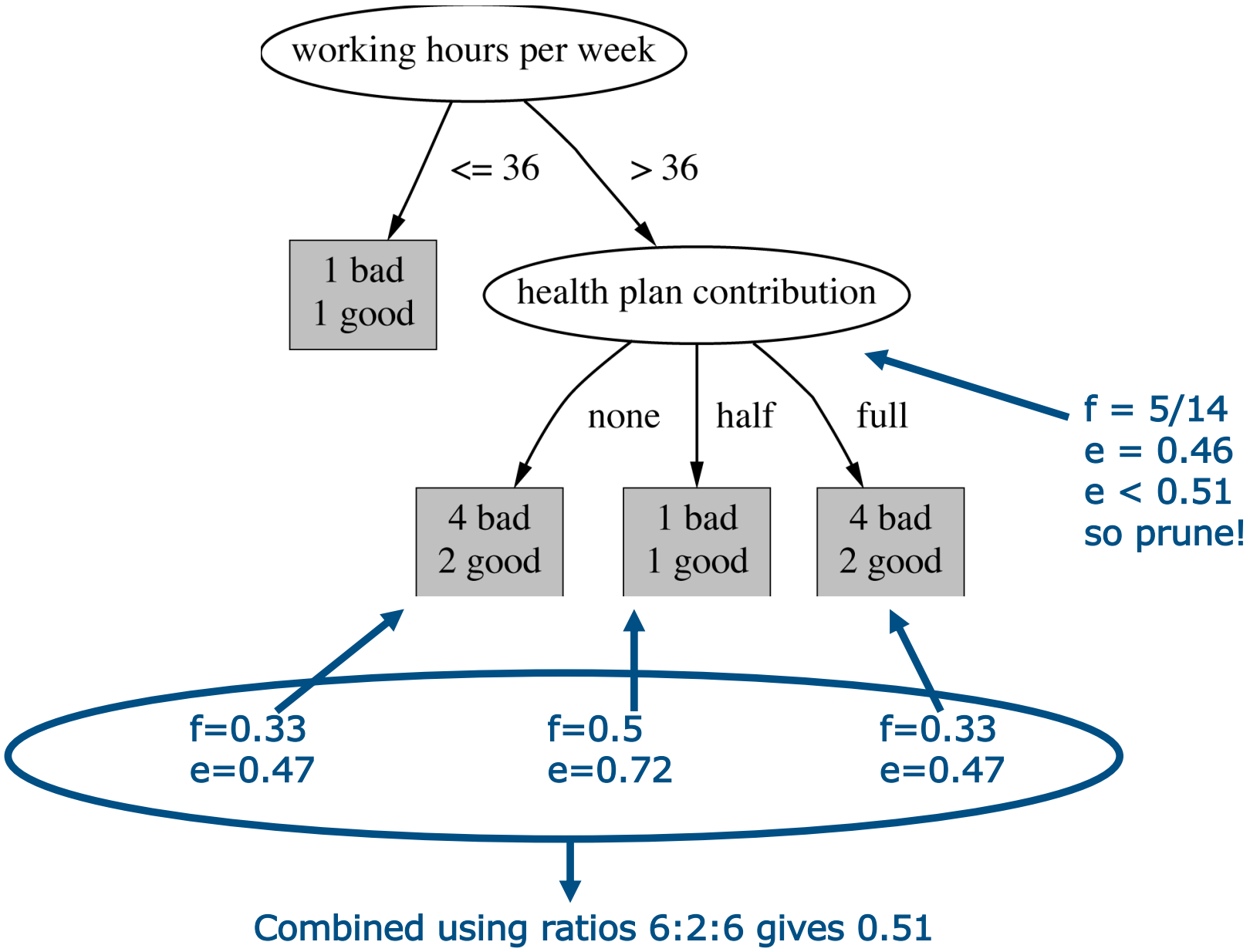
- Prune only if it reduces the estimated error
- Error on the training data is NOT a useful estimator
  (Why it would result in very little pruning?)
- A hold-out set might be kept for pruning
  ("reduced-error pruning")
- Example (C4.5's method)
  - Derive confidence interval from training data
  -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

C4.5's Pruning Method

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

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

- 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


working hours per week
<= 36
> 36
1 bad
1 good
health plan contribution
none
half
full
4 bad
2 good
1 bad
1 good
4 bad
2 good
f = 5/14
e = 0.46
e < 0.51
so prune!
f=0.33
e=0.47
f=0.5
e=0.72
f=0.33
e=0.47
Combined using ratios 6:2:6 gives 0.51

Regression & Model Trees

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

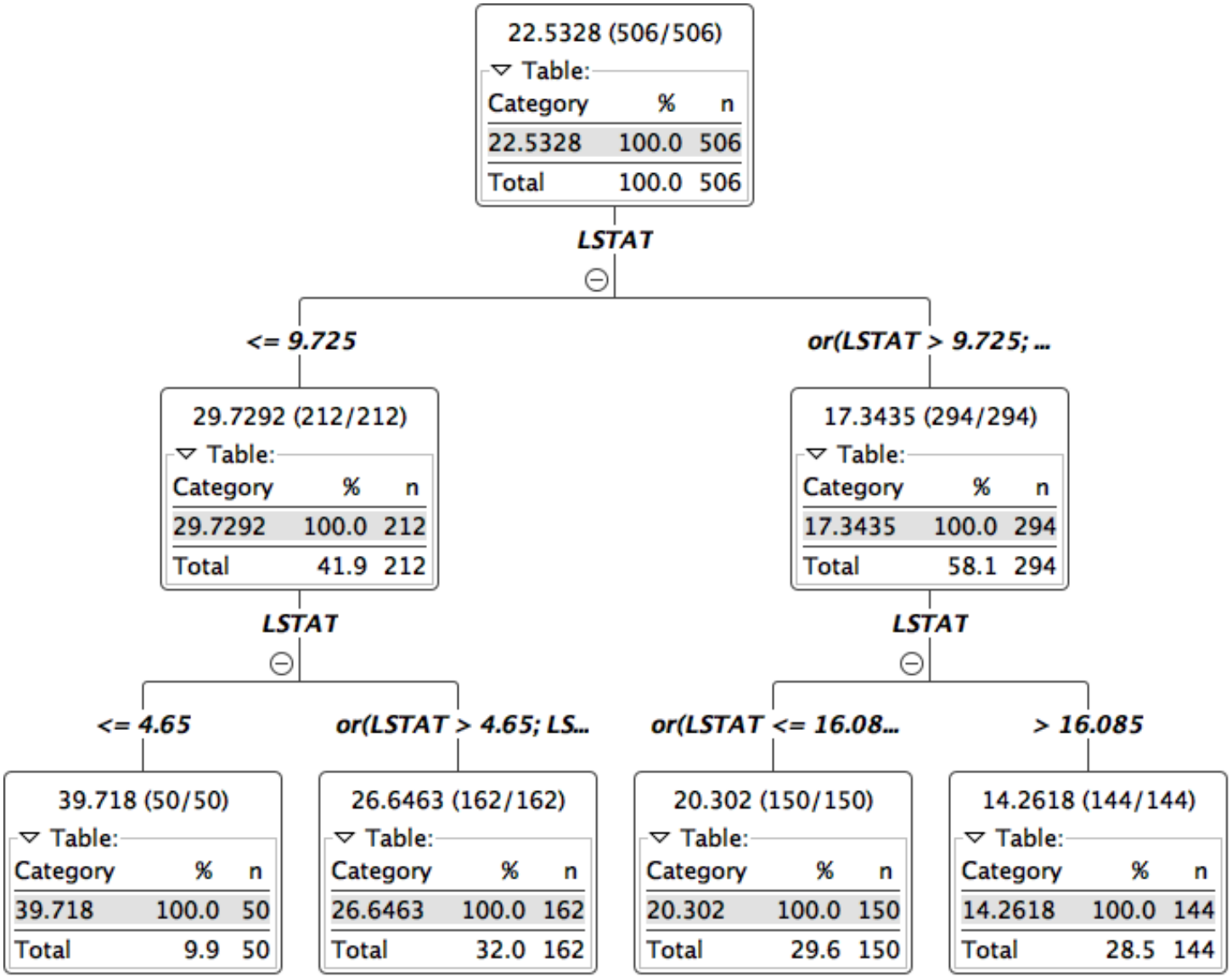
Regression and model trees work similarly to decision trees

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 **σ** 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

- 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)

Other Approaches

- 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
- …