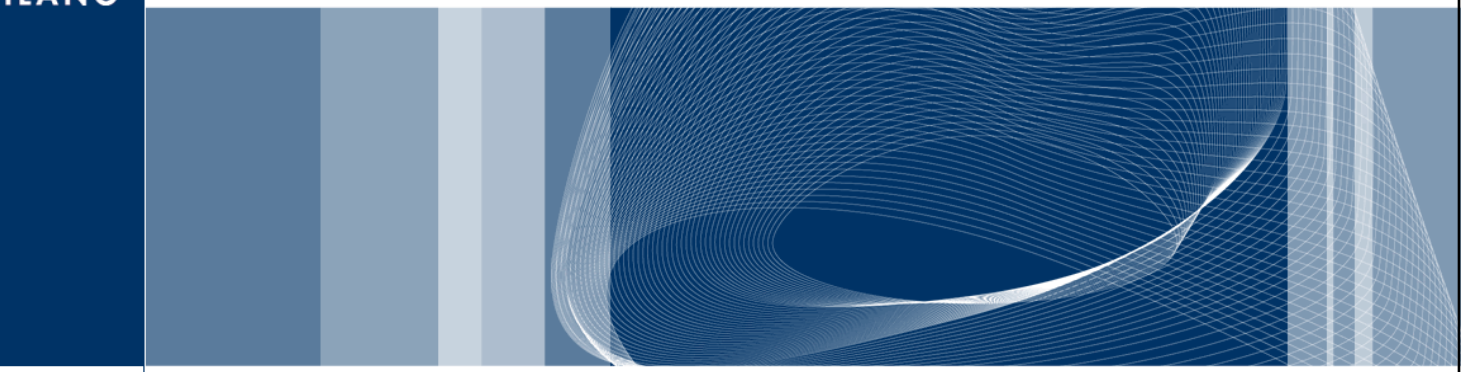
POLITECNICO DI MILANO



Decision Trees

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

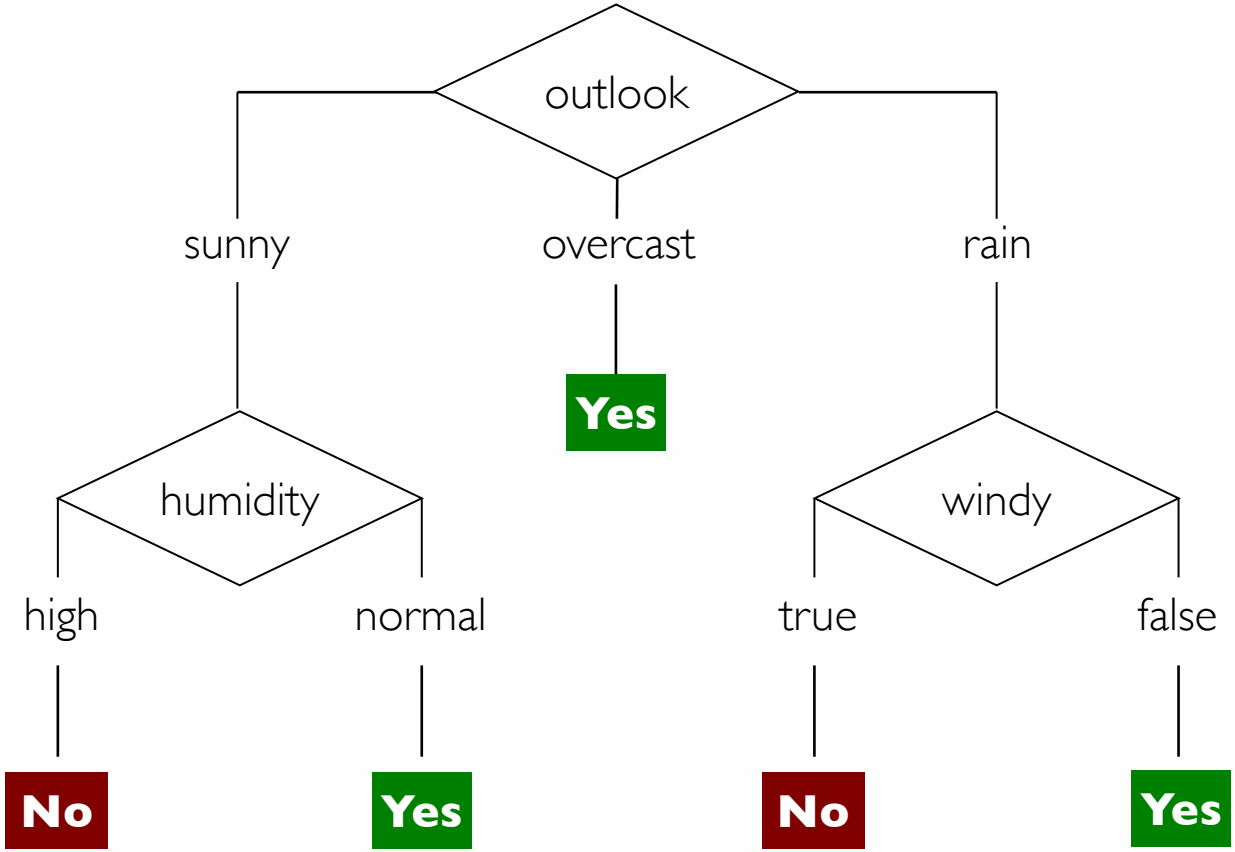
Prof. Pier Luca Lanzi

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

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

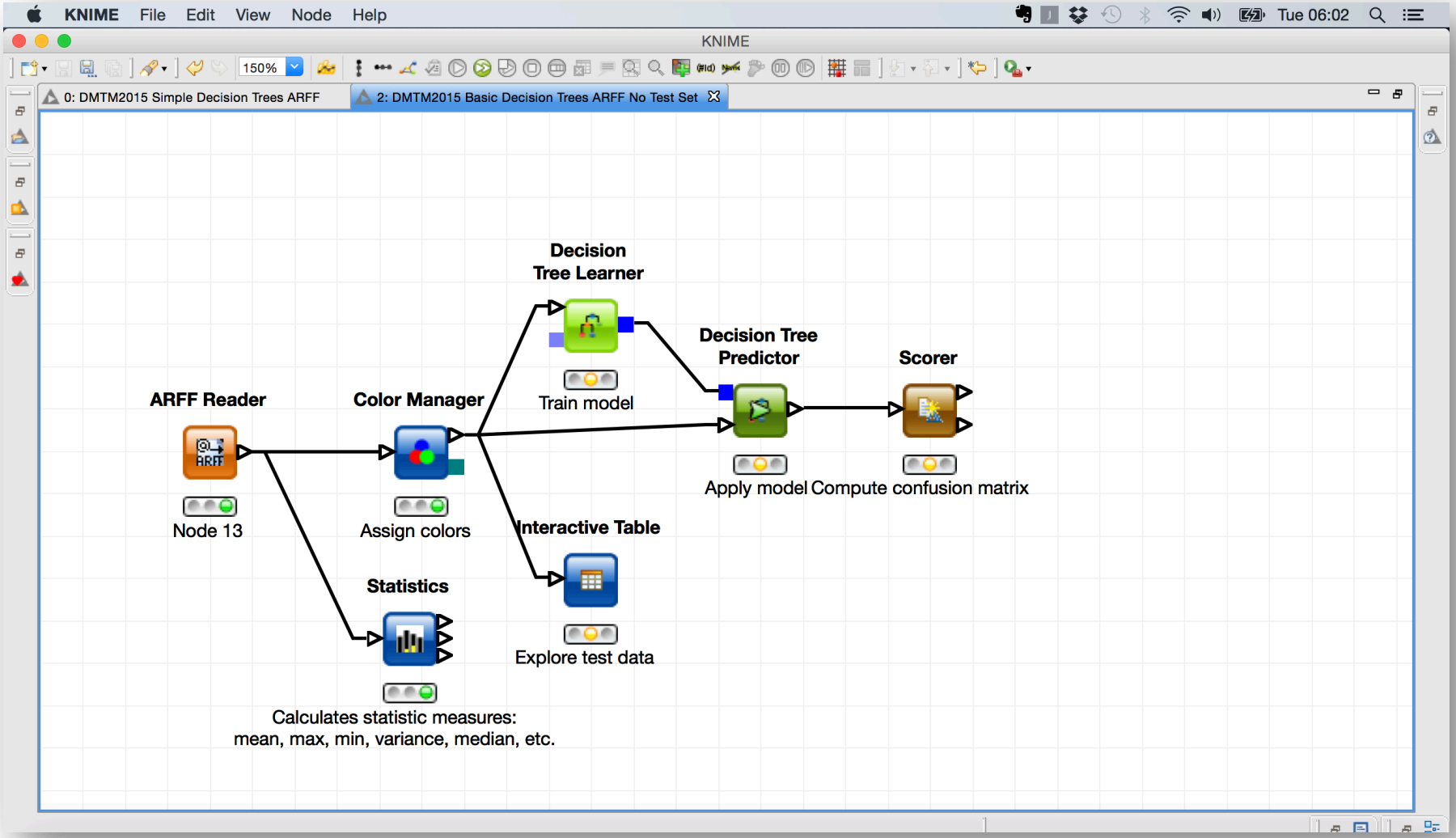
The Decision Tree for the Weather Dataset

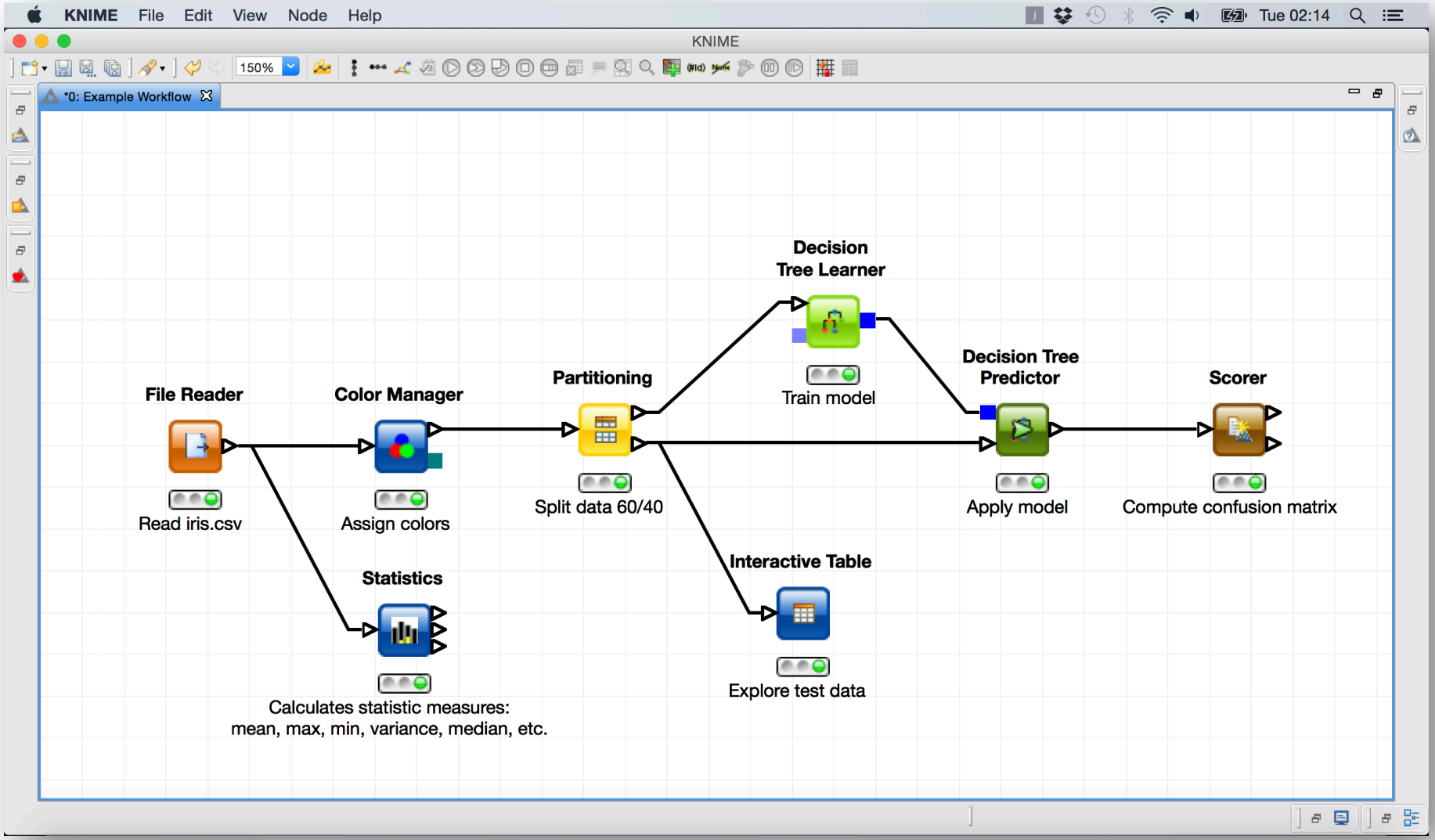


What is a Decision Tree?

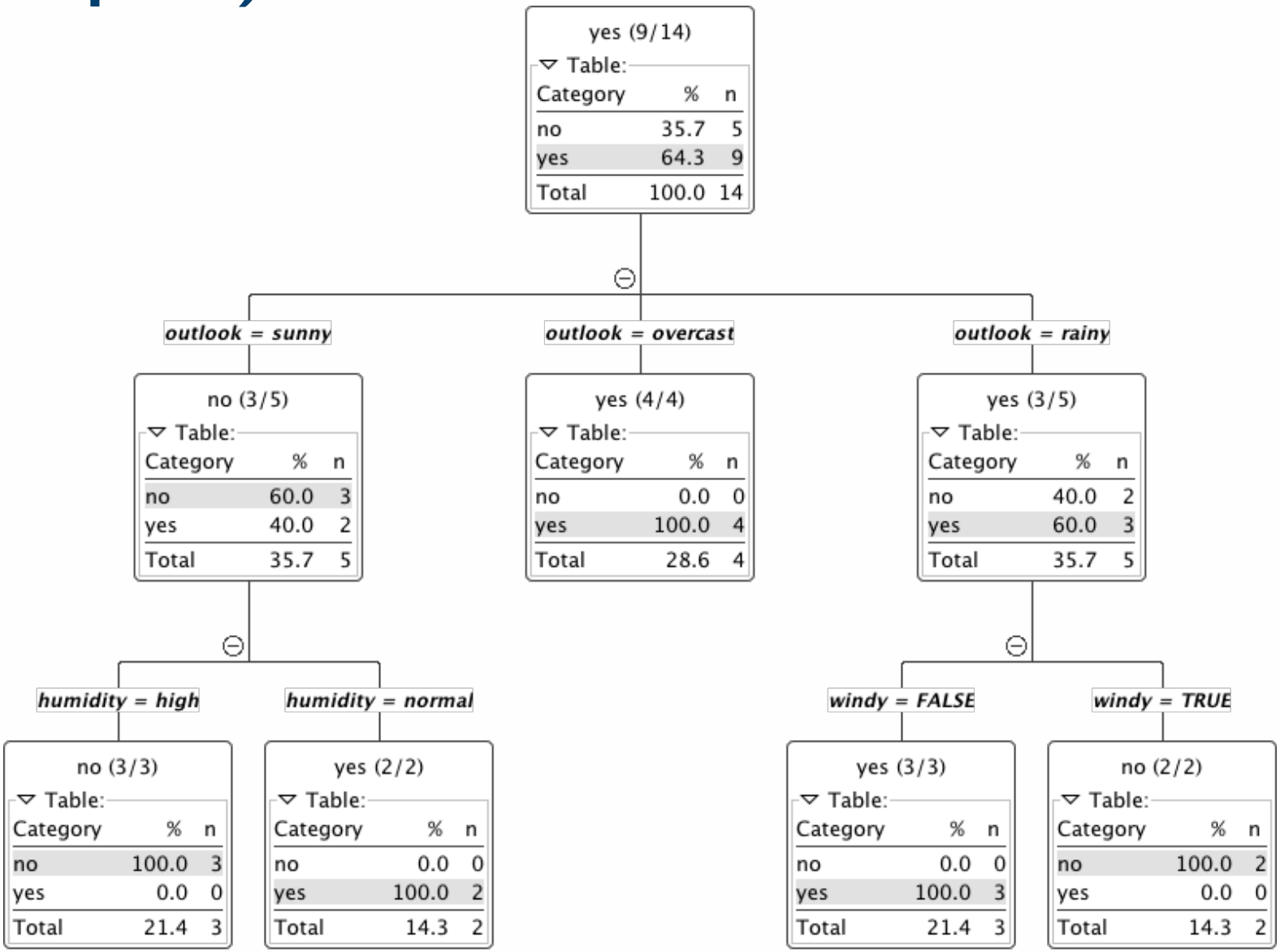
- An internal node is a test on an attribute
- A branch represents an outcome of the test, e.g., outlook=windy
- A leaf node represents a class label or class label distribution
- At each node, one attribute is chosen to split training examples into distinct classes as much as possible
- A new case is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

Building a Decision Tree with KNIME


KNIME File Edit View Node Help
Tue 06:02
KNIME
150%
0: DMTM2015 Simple Decision Trees ARFF
2: DMTM2015 Basic Decision Trees ARFF No Test Set
Decision Tree Learner
Train model
Decision Tree Predictor
Scorer
ARFF Reader
Color Manager
Node 13
Assign colors
Apply model
Compute confusion matrix
Interactive Table
Explore test data
Statistics
Calculates statistic measures:
mean, max, min, variance, median, etc.


KNIME File Edit View Node Help
Tue 02:14
KNIME
150%
*0: Example Workflow
File Reader
Read iris.csv
Color Manager
Assign colors
Partitioning
Split data 60/40
Decision Tree Learner
Train model
Decision Tree Predictor
Apply model
Scorer
Compute confusion matrix
Statistics
Calculates statistic measures:
mean, max, min, variance, median, etc.
Interactive Table
Explore test data

Decision Tree Representations (Graphical)



Decision Tree Representations (Text)

outlook = overcast: yes {no=0, yes=4}

outlook = rainy

| windy = FALSE: yes {no=0, yes=3}

| windy = TRUE: no {no=2, yes=0}

outlook = sunny

| humidity = high: no {no=3, yes=0}

| humidity = normal: yes {no=0, yes=2}

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.

Top Down Induction of Decision Trees

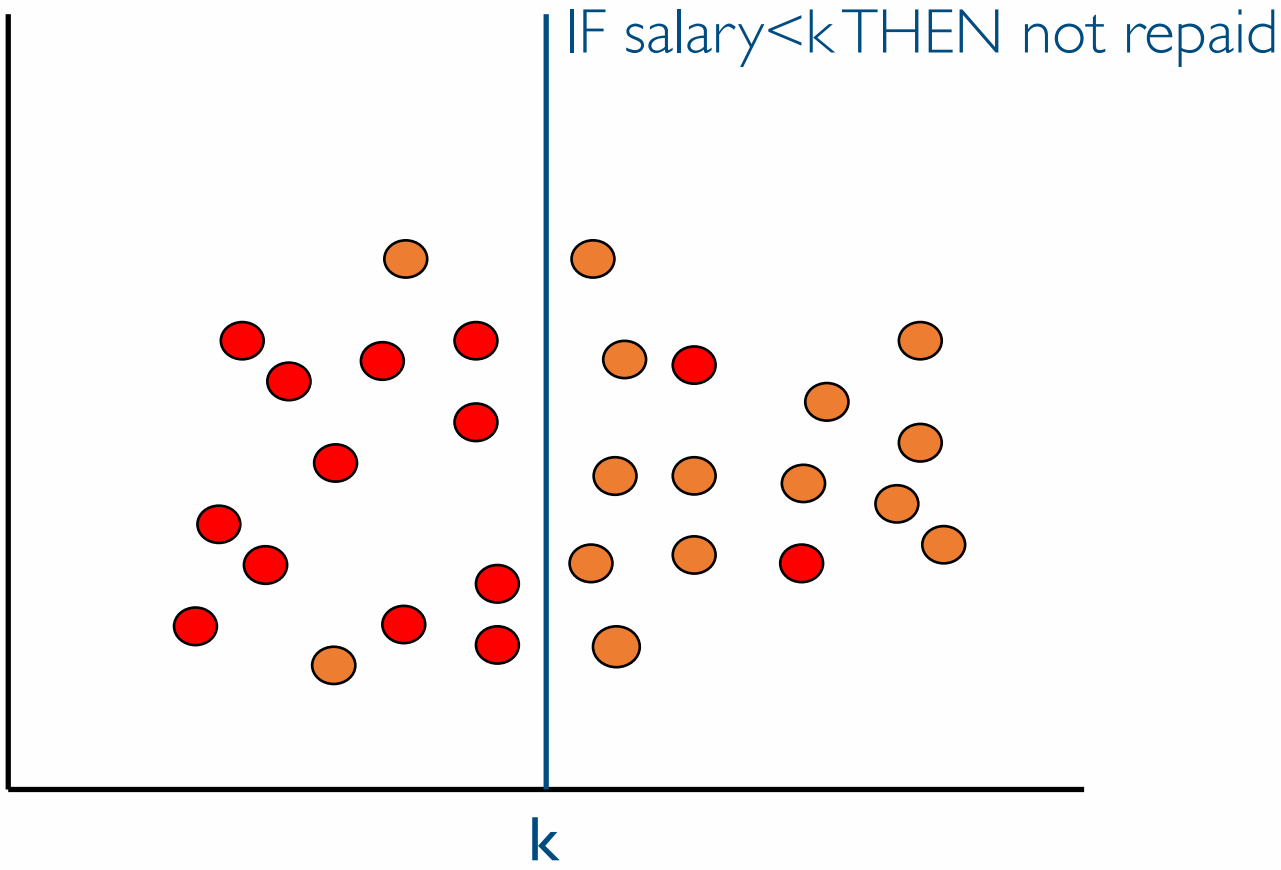

```
function TDIDT(S)      // S, a set of labeled examples
Tree = new empty node
if (all examples have the same class c
    or no further splitting is possible)
then // new leaf labeled with the majority class c
     Label(Tree) = c
else // new decision node
    (A,T) = FindBestSplit(S)
    foreach test t in T do
       St = all examples that satisfy t
       Nodet = TDIDT(St)
       AddEdge(Tree -> Nodet)
    endfor
endif
return Tree
```

When Should Building Stop?

- 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

What Splitting Attribute?

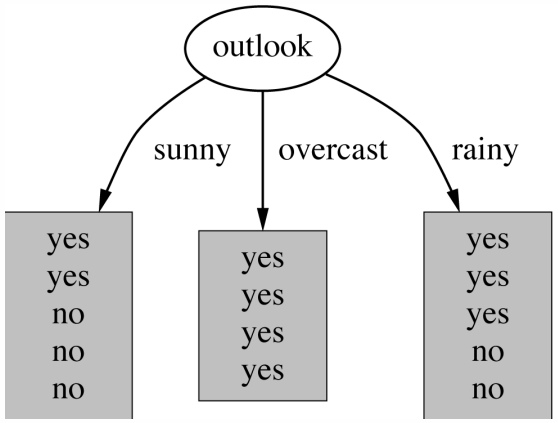
What Do We Need?

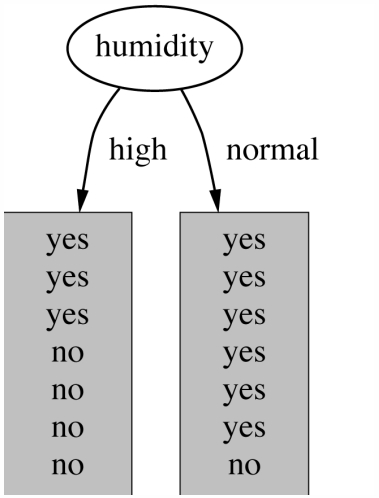


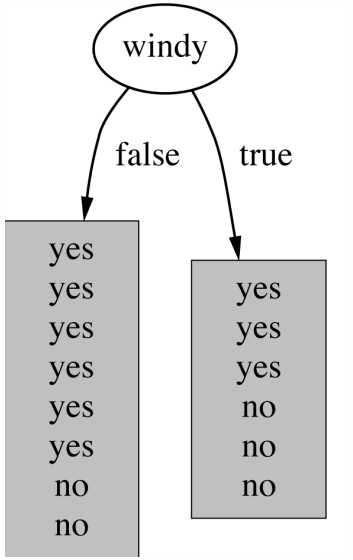
Which Attribute for Splitting?

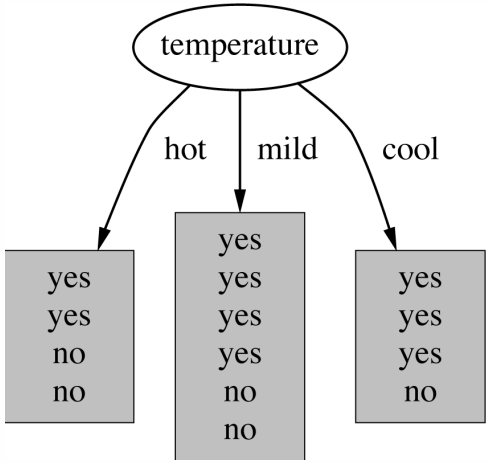
- At each node, available attributes are evaluated on the basis of separating the classes of the training examples
- A purity or impurity measure is used for this purpose
- Information Gain: increases with the average purity of the subsets that an attribute produces
- Splitting Strategy: choose the attribute 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?









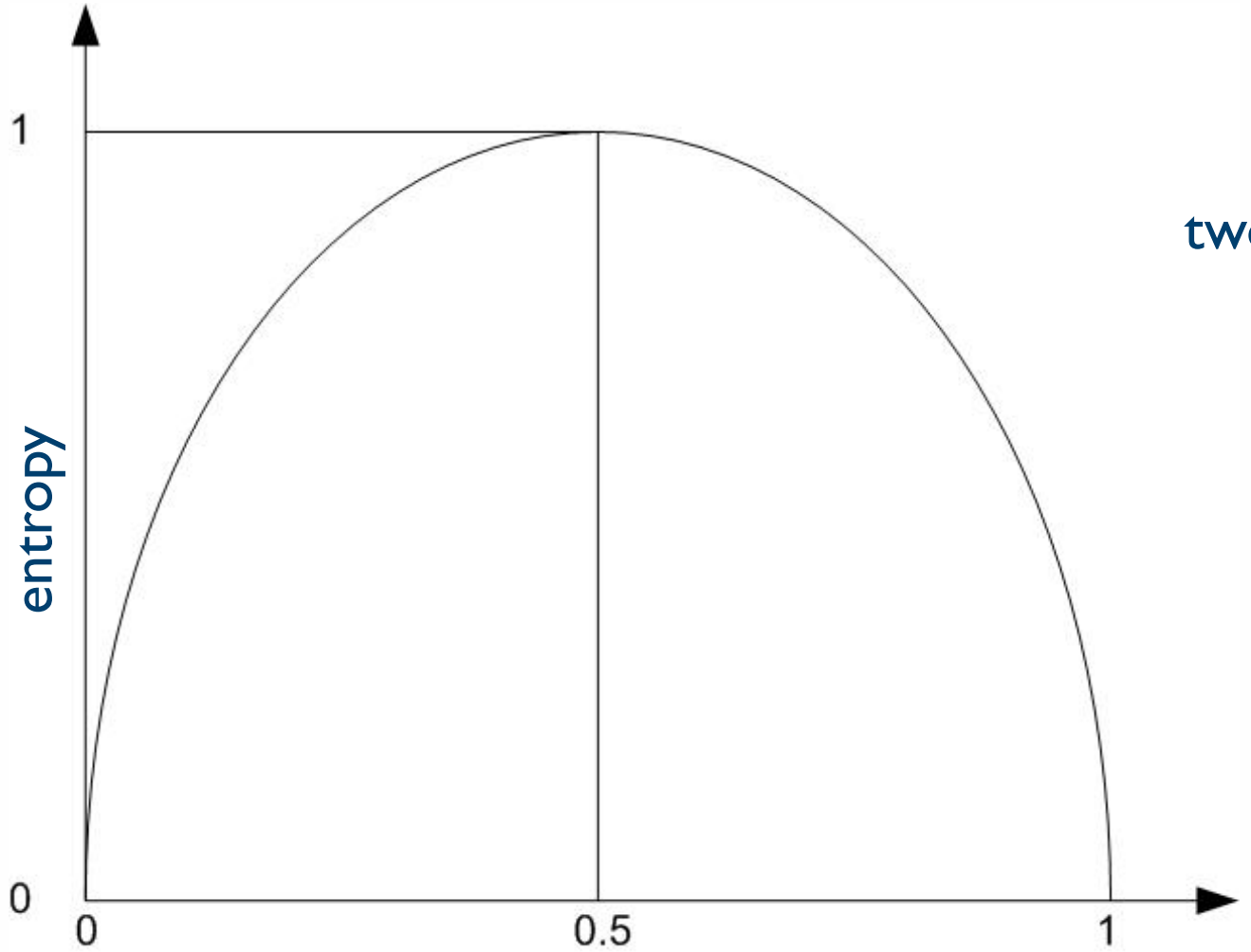
Computing Information

- 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

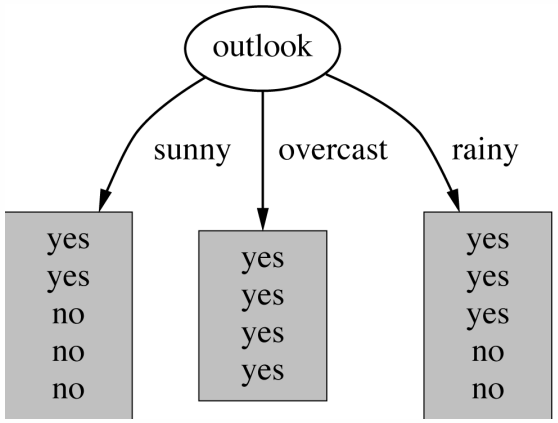
$$entropy(p_1, p_2, \ldots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n$$

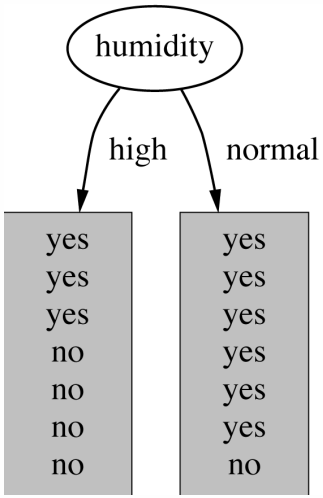
Entropy

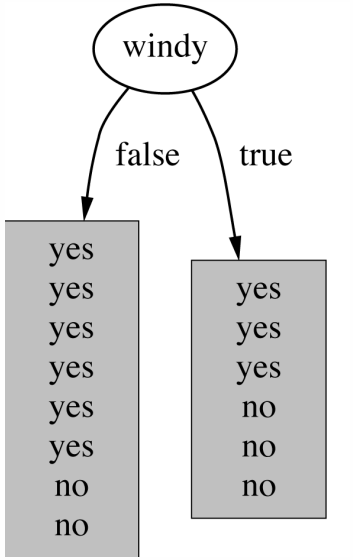


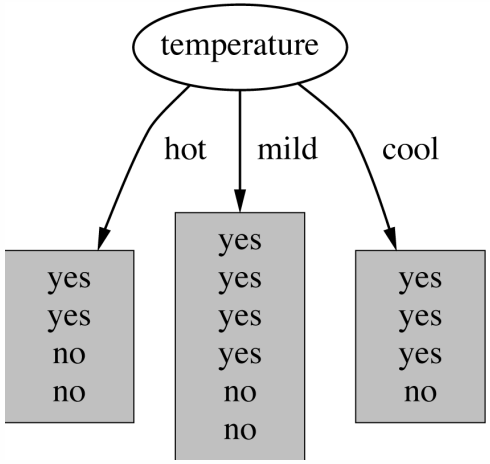
two classes
0/1

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

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

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, info(D), is the entropy,

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

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

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

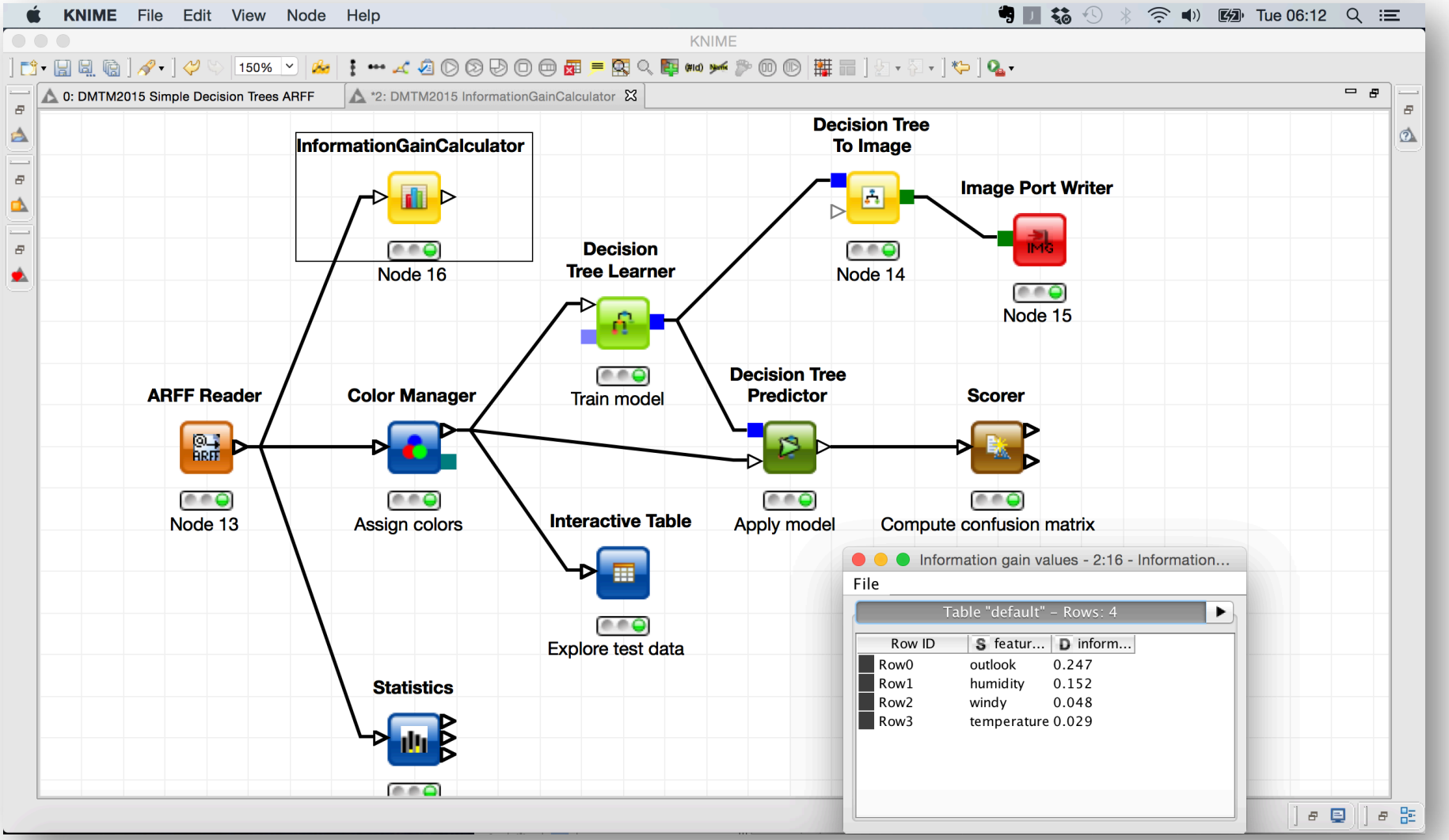
Information Gain

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

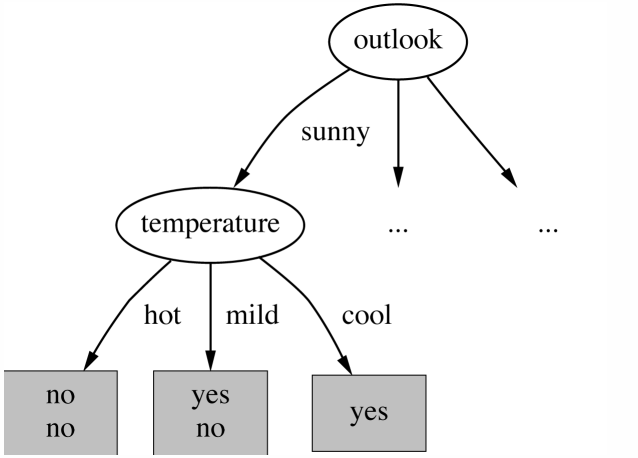
$$\begin{aligned} gain(outlook) &= info([9,5]) - info([2,3][4,0][3,2]) \\ &= 0.940 - 0.693 \\ &= 0.247 \end{aligned}$$

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

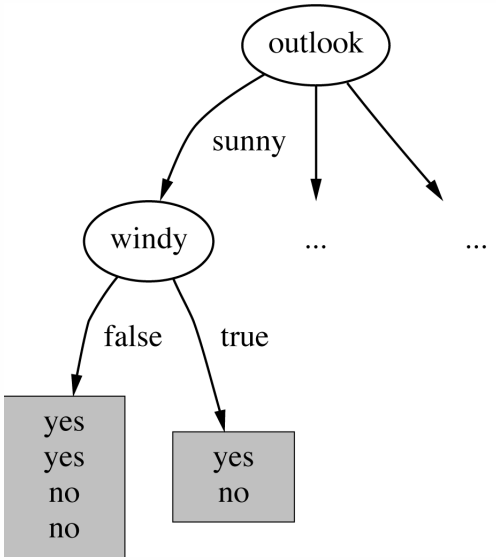
We can easily check the math


KNIME File Edit View Node Help
Tue 06:12
KNIME
150%
0: DMTM2015 Simple Decision Trees ARFF
*2: DMTM2015 InformationGainCalculator
InformationGainCalculator
Node 16
Decision Tree To Image
Node 14
Image Port Writer
Node 15
Decision Tree Learner
Train model
ARFF Reader
Node 13
Color Manager
Assign colors
Decision Tree Predictor
Apply model
Scorer
Compute confusion matrix
Interactive Table
Explore test data
Statistics
Information gain values - 2:16 - Information...
File
Table "default" - Rows: 4
Row ID
featur...
inform...
Row0 outlook 0.247
Row1 humidity 0.152
Row2 windy 0.048
Row3 temperature 0.029

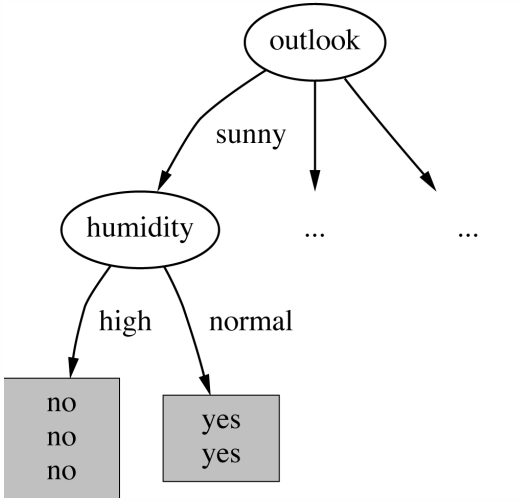
Going Further



$$gain(temperature) = 0.571$$



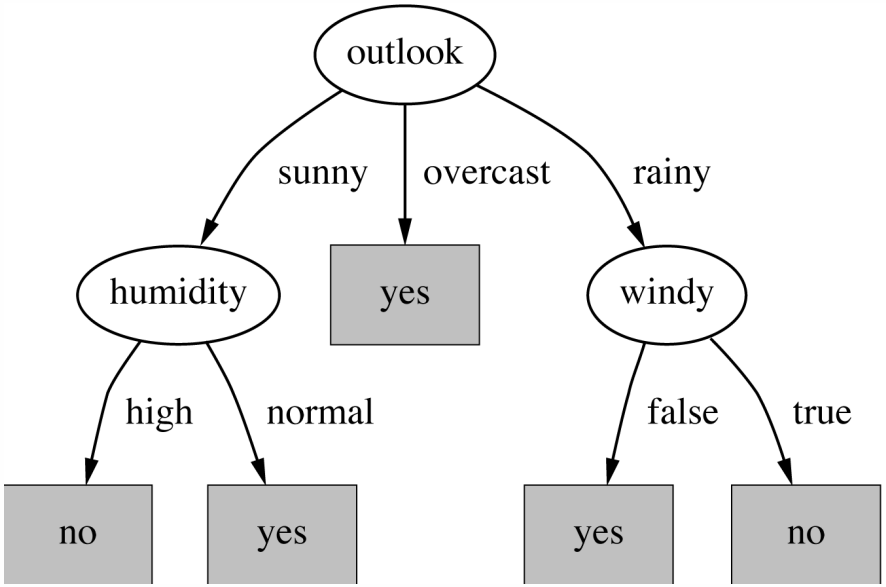
$$gain(windy) = 0.020$$

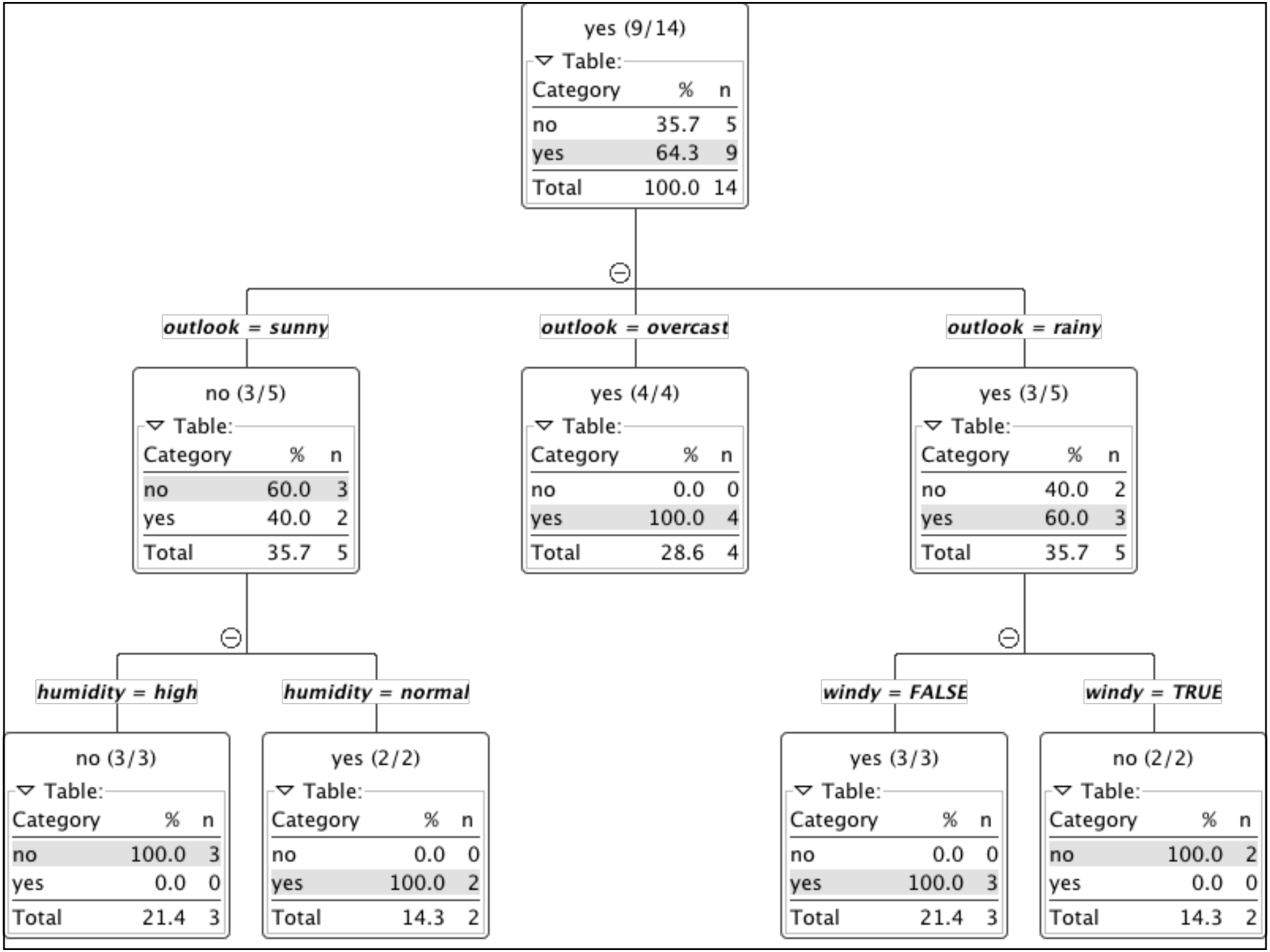


$$gain(humidity) = 0.971$$

The Final Decision Tree

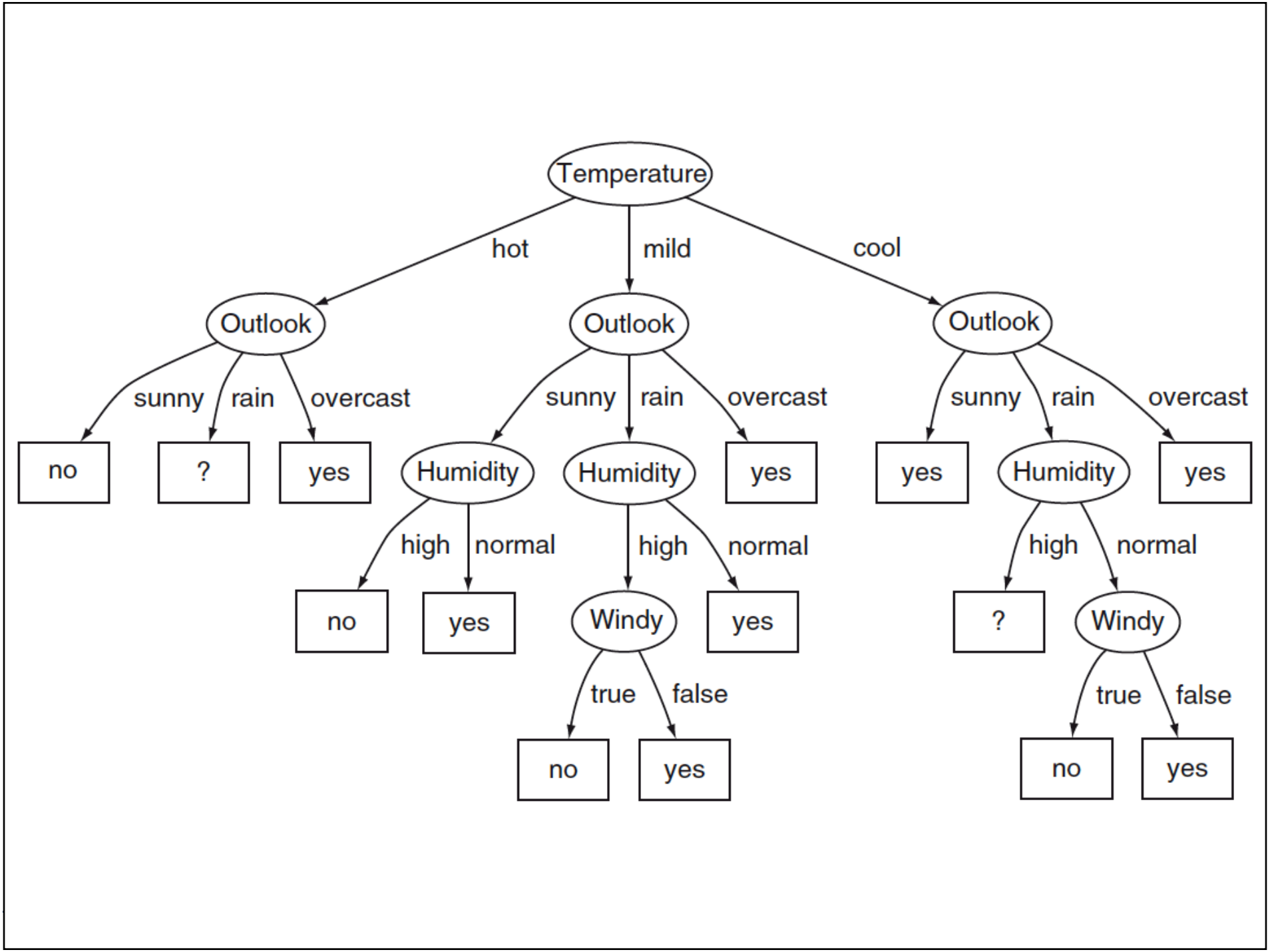
- Not all the leaves need to be pure
- Splitting stops when data can not be split any further




yes (9/14)
Table:
Category % n
no 35.7 5
yes 64.3 9
Total 100.0 14
outlook = sunny
outlook = overcast
outlook = rainy
no (3/5)
Table:
Category % n
no 60.0 3
yes 40.0 2
Total 35.7 5
yes (4/4)
Table:
Category % n
no 0.0 0
yes 100.0 4
Total 28.6 4
yes (3/5)
Table:
Category % n
no 40.0 2
yes 60.0 3
Total 35.7 5
humidity = high
humidity = normal
windy = FALSE
windy = TRUE
no (3/3)
Table:
Category % n
no 100.0 3
yes 0.0 0
Total 21.4 3
yes (2/2)
Table:
Category % n
no 0.0 0
yes 100.0 2
Total 14.3 2
yes (3/3)
Table:
Category % n
no 0.0 0
yes 100.0 3
Total 21.4 3
no (2/2)
Table:
Category % n
no 100.0 2
yes 0.0 0
Total 14.3 2

We Can Always Build
a 100% Accurate Tree

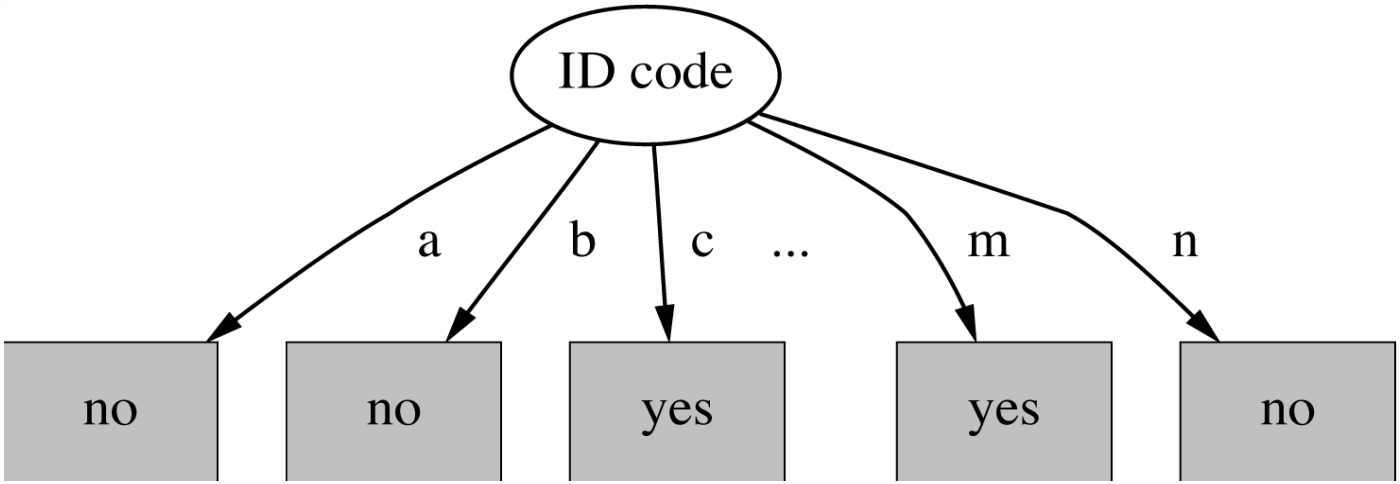
But do we want it?


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

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 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

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

- 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

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

- 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

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

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

More on Information Gain Ratio

- "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

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 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

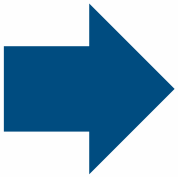
- E.g. temperature ≤71.5: yes/4, no/2
  temperature > 71.5: yes/5, no/3

- Info([4,2],[5,3]) = 6/14 info([4,2]) + 8/14 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

compute the gain for every possible split

what is the information gain if we split here?

Information Gain for Humidity

Humidity	Play	# of Yes	% of Yes	# of No	% of No	Weight	Entropy Left	# of Yes	% of Yes	# of No	% of No	Weight	Entropy Right	Information Gain
65	Yes	1	100.00%	0	0.00%	7.14%	0.00	8.00	0.62	5.00	0.38	92.86%	0.96	0.0477
70	No	1	50.00%	1	50.00%	14.29%	1.00	8.00	0.67	4.00	0.33	85.71%	0.92	0.0103
70	Yes	2	66.67%	1	33.33%	21.43%	0.92	7.00	0.64	4.00	0.36	78.57%	0.95	0.0005
70	Yes	3	75.00%	1	25.00%	28.57%	0.81	6.00	0.60	4.00	0.40	71.43%	0.97	0.0150
75	Yes	4	80.00%	1	20.00%	35.71%	0.72	5.00	0.56	4.00	0.44	64.29%	0.99	0.0453
78	Yes	5	83.33%	1	16.67%	42.86%	0.65	4.00	0.50	4.00	0.50	57.14%	1.00	0.0903
80	Yes	6	85.71%	1	14.29%	50.00%	0.59	3.00	0.43	4.00	0.57	50.00%	0.99	0.1518
80	Yes	7	87.50%	1	12.50%	57.14%	0.54	2.00	0.33	4.00	0.67	42.86%	0.92	0.2361
80	No	7	77.78%	2	22.22%	64.29%	0.76	2.00	0.40	3.00	0.60	35.71%	0.97	0.1022
85	No	7	70.00%	3	30.00%	71.43%	0.88	2.00	0.50	2.00	0.50	28.57%	1.00	0.0251
90	No	7	63.64%	4	36.36%	78.57%	0.95	2.00	0.67	1.00	0.33	21.43%	0.92	0.0005
90	Yes	8	66.67%	4	33.33%	85.71%	0.92	1.00	0.50	1.00	0.50	14.29%	1.00	0.0103
95	No	8	61.54%	5	38.46%	92.86%	0.96	1.00	1.00	0.00	0.00	7.14%	0.00	0.0477
96	Yes	9	64.29%	5	35.71%	100.00%	0.94	0.00	0.00	0.00	0.00	0.00%	0.00	0.0000

Gini Index

Another Splitting Criteria: The Gini Index

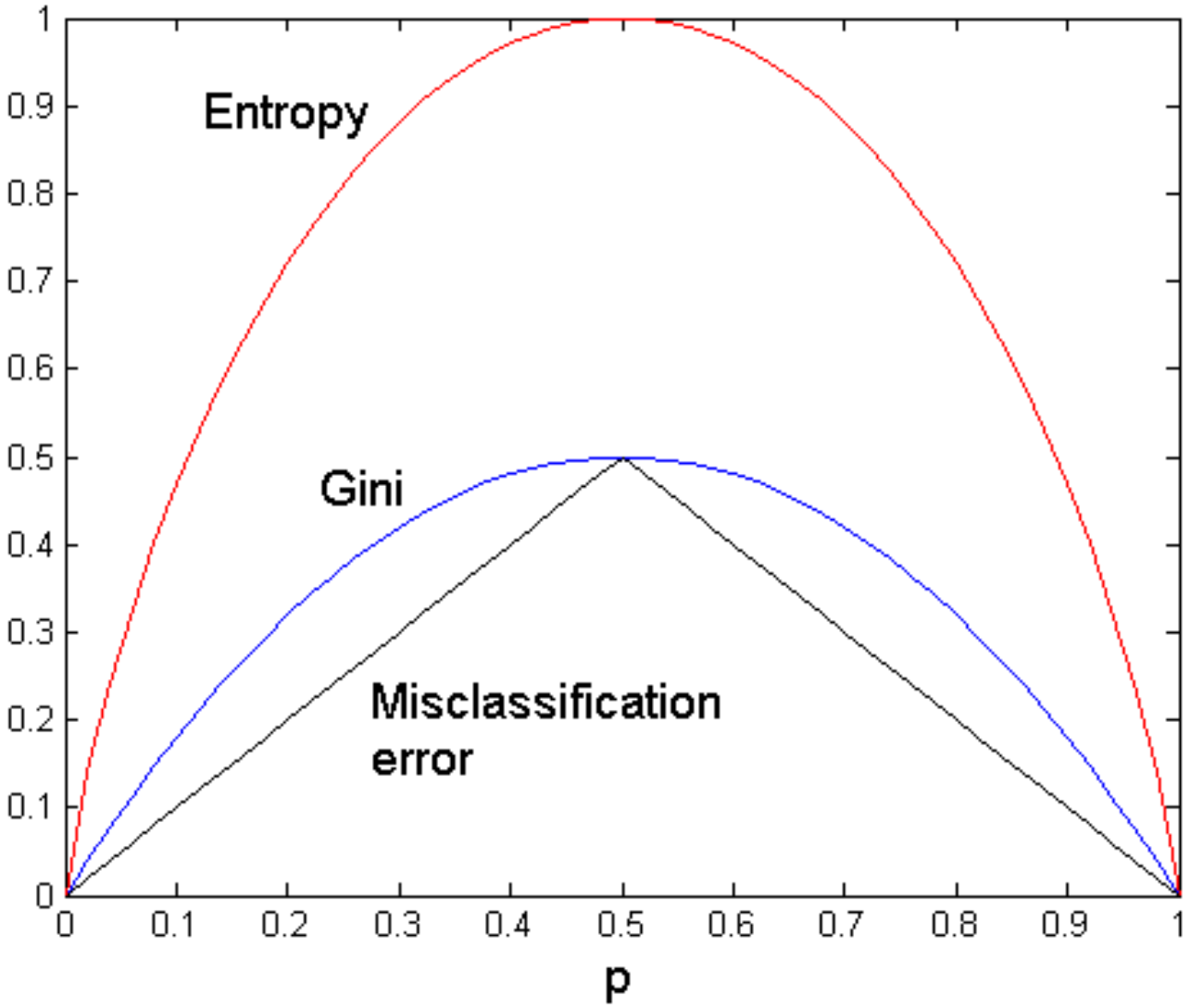
- The Gini index, for a data set T contains examples from n classes, is defined as

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

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



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: Example

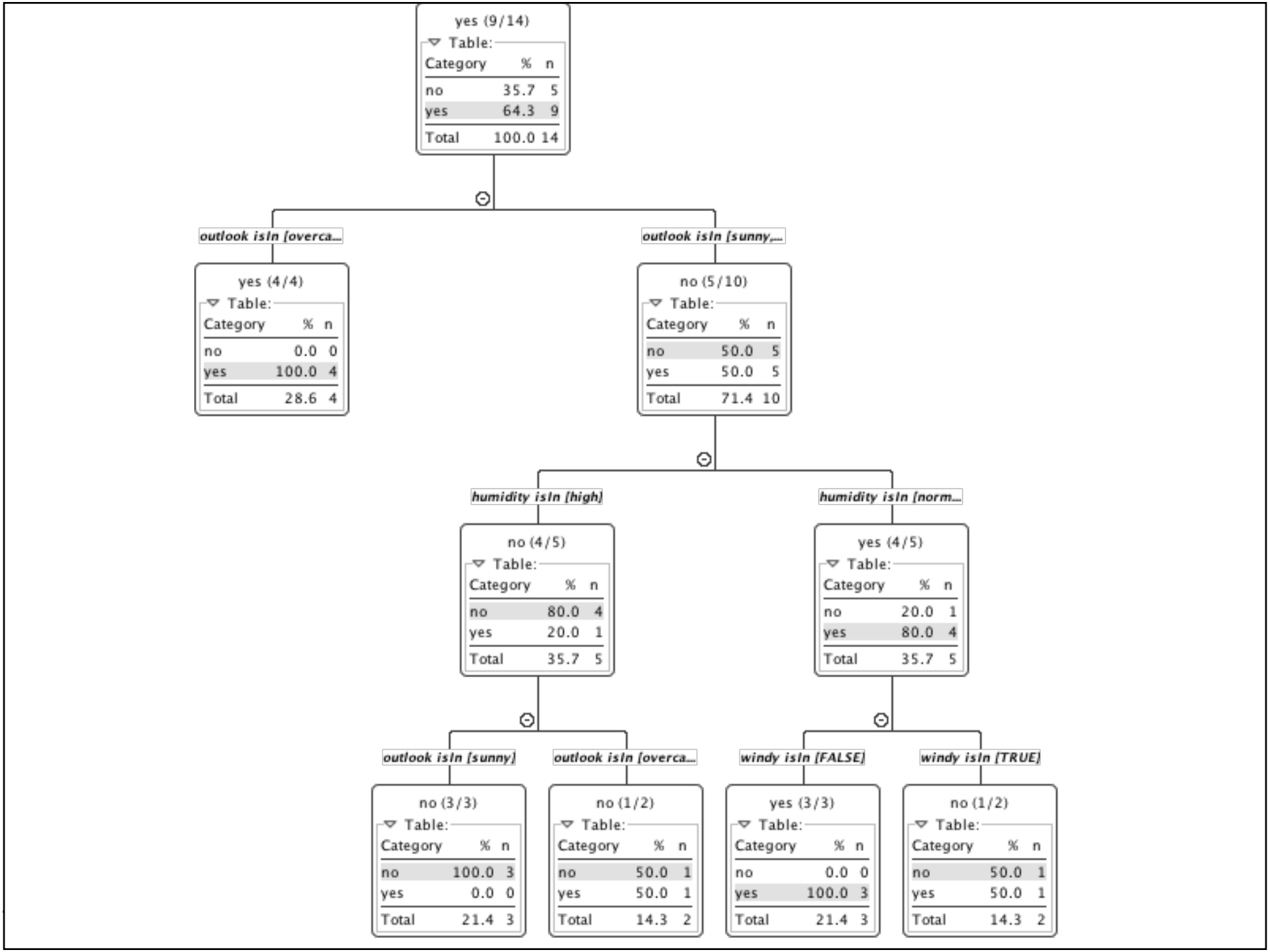
- D has 9 tuples labeled "yes" and 5 labeled "no"

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

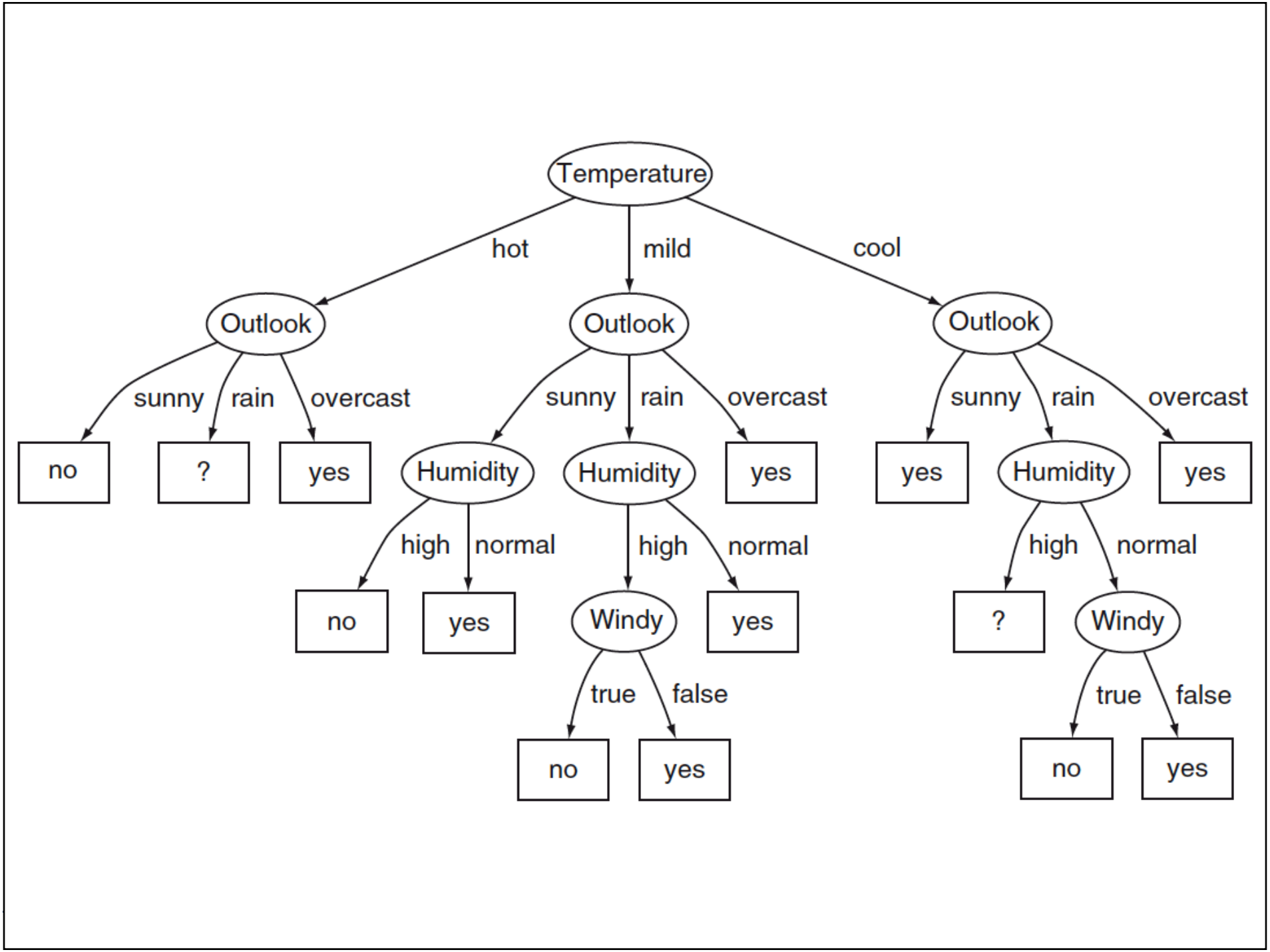
- Suppose the attribute income partitions D into 10 in D_1 branching on low and medium and 4 in D_2

$$\begin{aligned} gini(D)_{\{l,m\}} &= \left(\frac{10}{14}\right) gini(D_1) + \left(\frac{4}{14}\right) gini(D_2) \\ &= \left(\frac{10}{14}\right)\left(1 - (\frac{6}{10})^2 - (\frac{4}{10})^2\right) + \\ &+ \left(\frac{4}{14}\right)\left(1 - (\frac{1}{4})^2 - (\frac{3}{4})^2\right) = 0.450 \end{aligned}$$

- but gini{medium,high} is 0.30 and thus the best since it is the lowest


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

Generalization and overfitting in trees


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

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"

Pre-Pruning

- 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
- Early stopping and halting
  - No individual attribute exhibits any interesting information about the class
  - The structure is only visible in fully expanded tree
  - Pre-pruning won't expand the root node

Post-Pruning

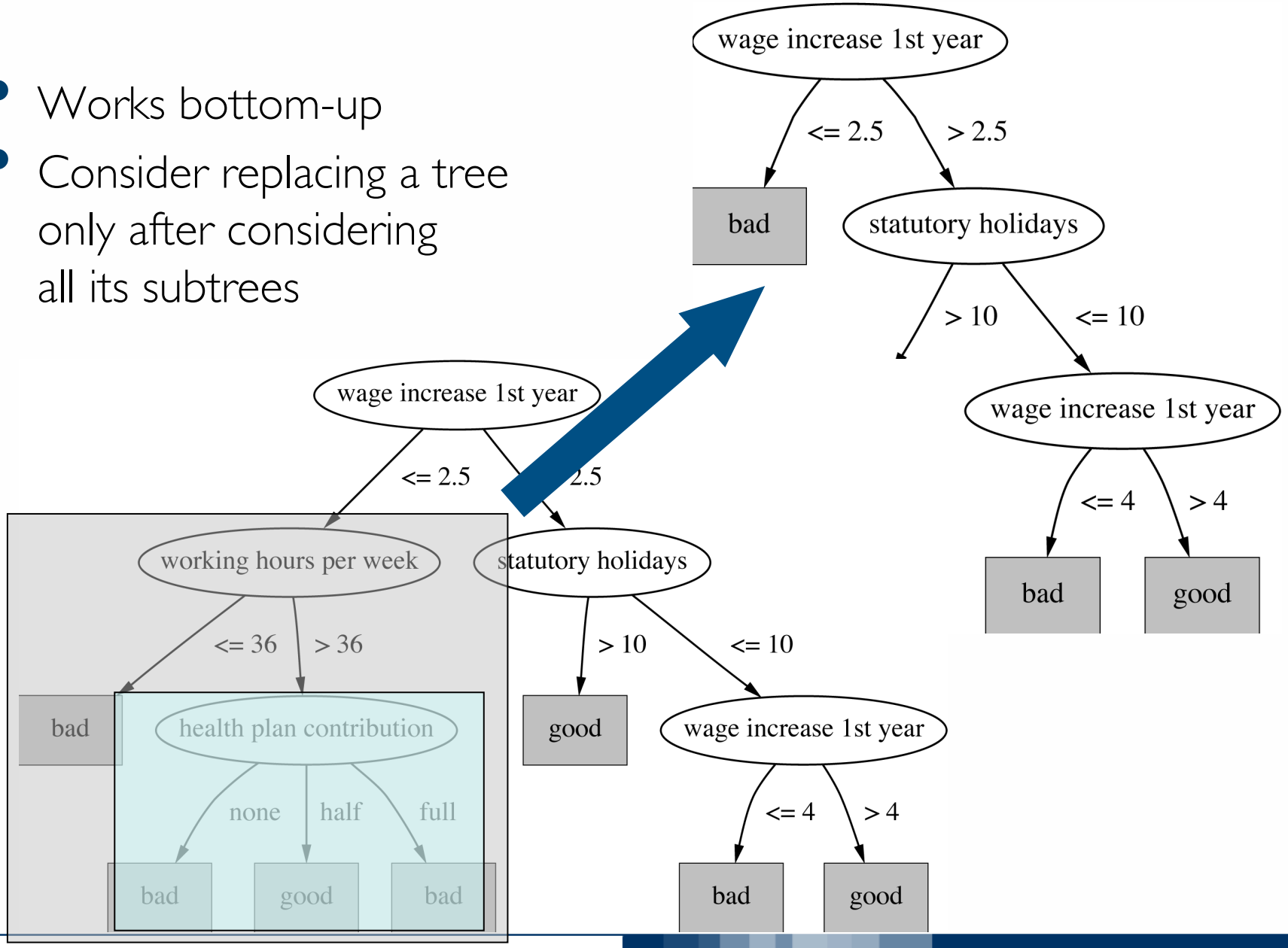
- First, build full tree, then prune it
- Fully-grown tree shows all attribute interactions
- Problem: some subtrees might be due to chance effects

- 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)

Mean and Variance

- Mean and variance for a Bernoulli trial: p, p (1–p)
- Expected error rate f=S/N has mean p and variance p(1–p)/N
- For large enough N, f follows a Normal distribution
- c% confidence interval [–z≤X ≤ z] for random variable with 0 mean is given by:
- With a symmetric distribution,

$$P[-z \leq X \leq z] = c$$
$$= 1 - 2 \times P[X \geq z]$$

Confidence Limits

- Confidence limits for the normal distribution with 0 mean and a variance of 1,

Pr[X ≥ z]	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
25%	0.69
40%	0.25

- Thus,

$$P[-1.65 \leq X \leq 1.65] = 90\%$$

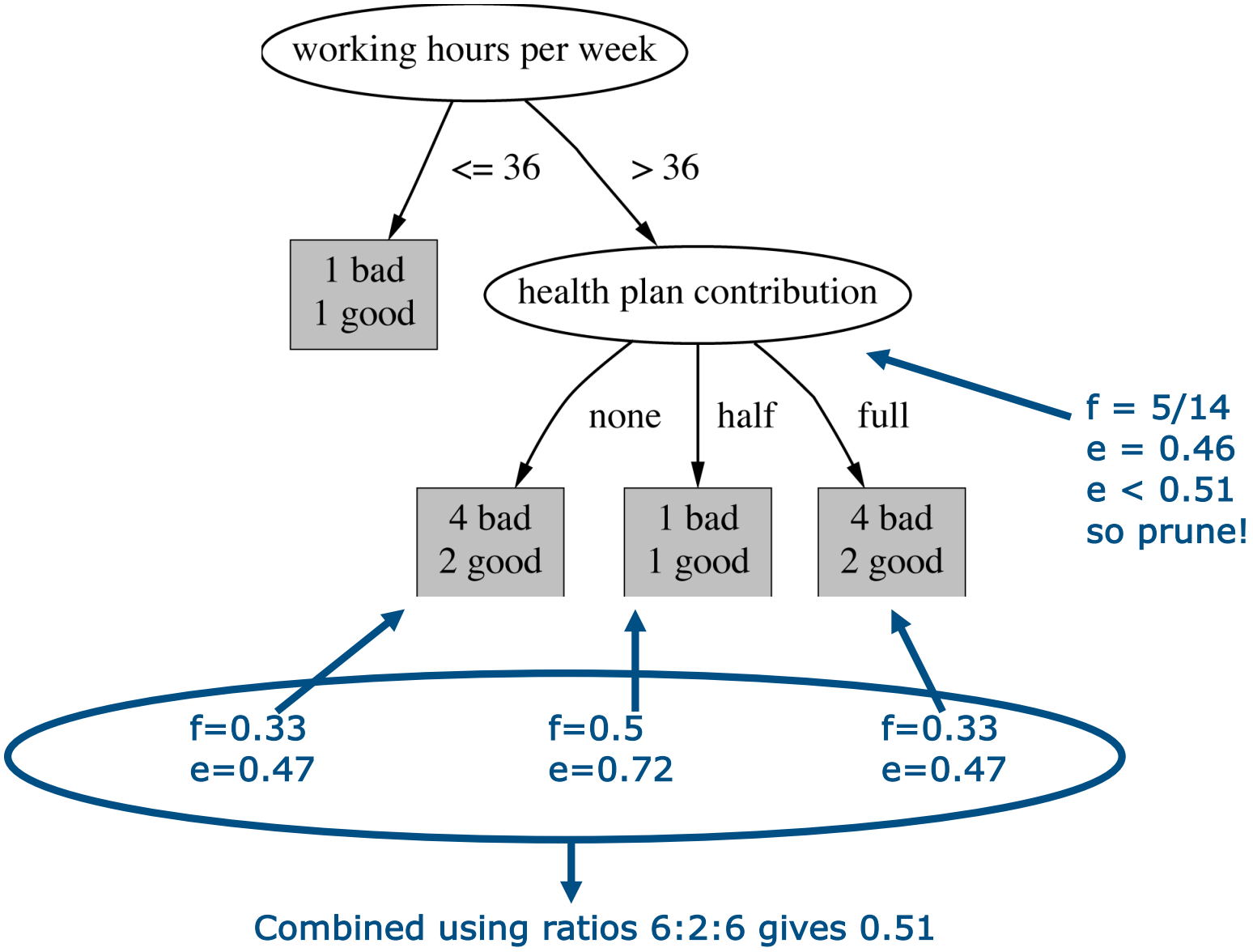
- To use this we have to reduce our random variable f to have 0 mean and unit variance

C4.5's Pruning Method

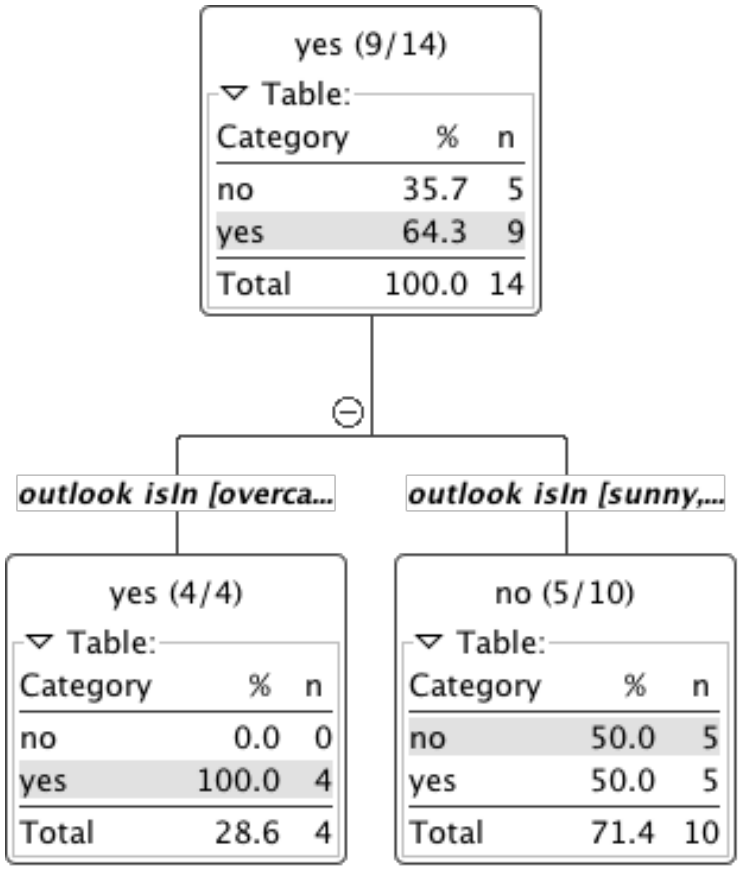
- Given the error f on the training data, the upper bound for the error estimate for a node is computed as

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

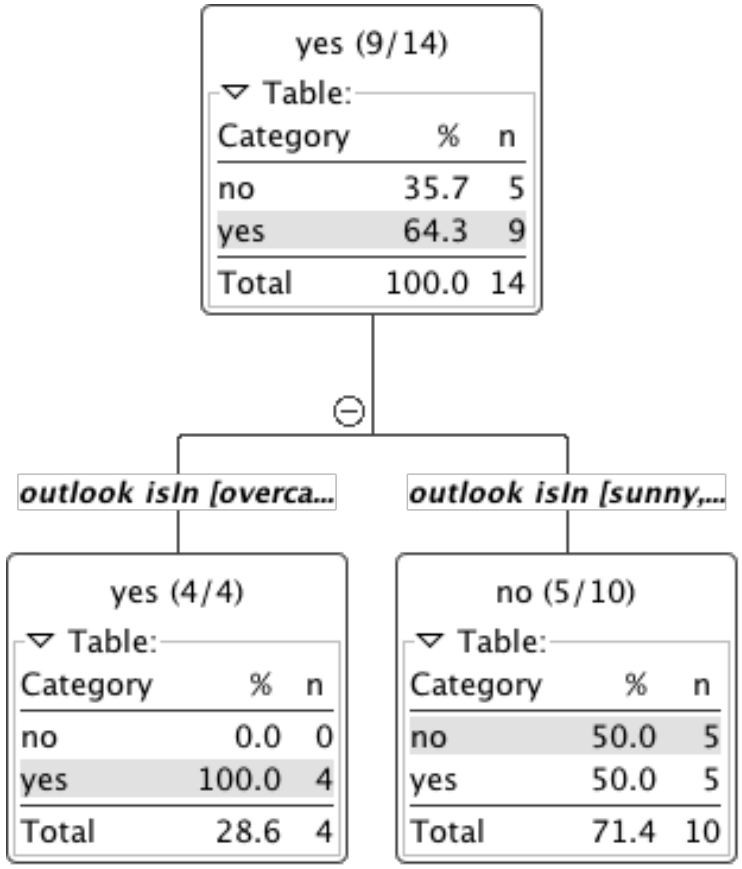
- If c = 25% then z = 0.69 (from normal distribution)
  - f is the error on the training data
  - N is the number of instances covered by the leaf


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

Examples using KNIME

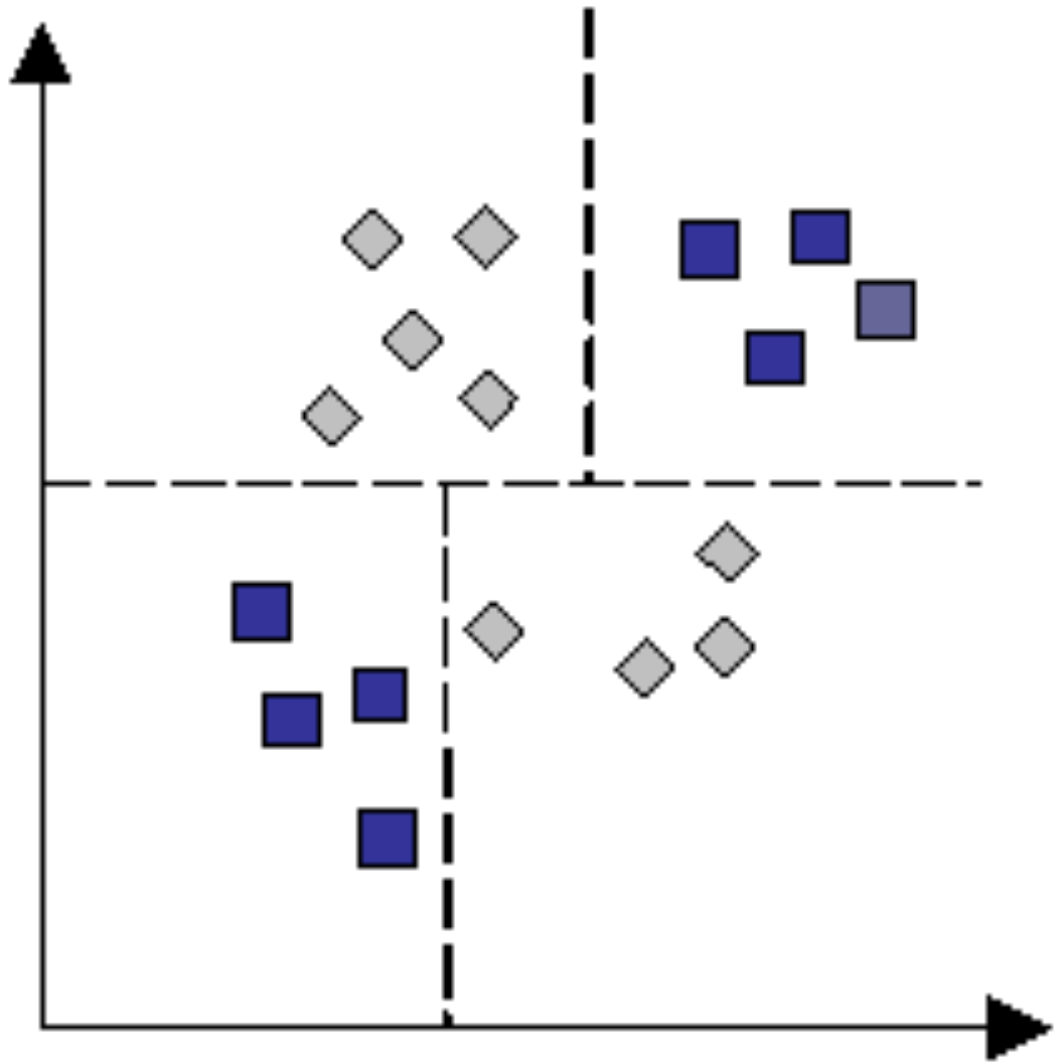


pruned tree (Gain ratio)



pruned tree (Gini index)

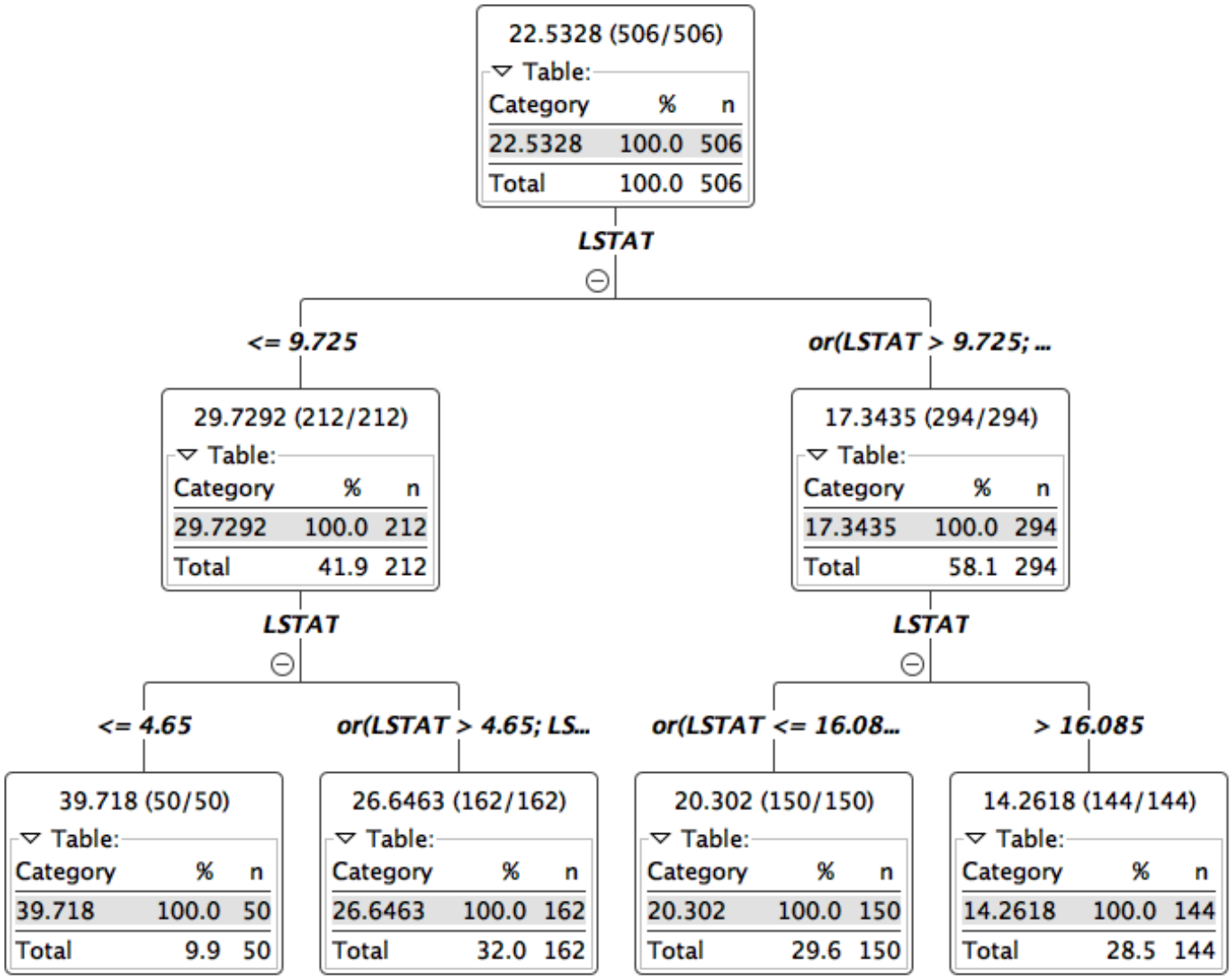
Geometric Interpretation



Regression Trees

Regression Trees for Prediction

- Decision trees can also be used to predict the value of a numerical target variable
- Regression trees work similarly to decision trees, by analyzing several splits attempted, choosing the one that minimizes impurity
- Differences from Decision Trees
  - Prediction is computed as the average of numerical target variable in the subspace (instead of the majority vote)
  - Impurity is measured by sum of squared deviations from leaf mean (instead of information-based measures)
  - Find split that produces greatest separation in $\sum[y - E(y)]^2$
  - Find nodes with minimal within variance and therefore
  - Performance is measured by RMSE (root mean squared error)



Input variable: LSTAT - % lower status of the population

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

Summary

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

Suggested Homework

- 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.