Learning Algorithm Implementations Inferring Rudimentary Rules and Decision Trees

Natasha Balac, Ph.D.

MAS DSE 220

April 2016

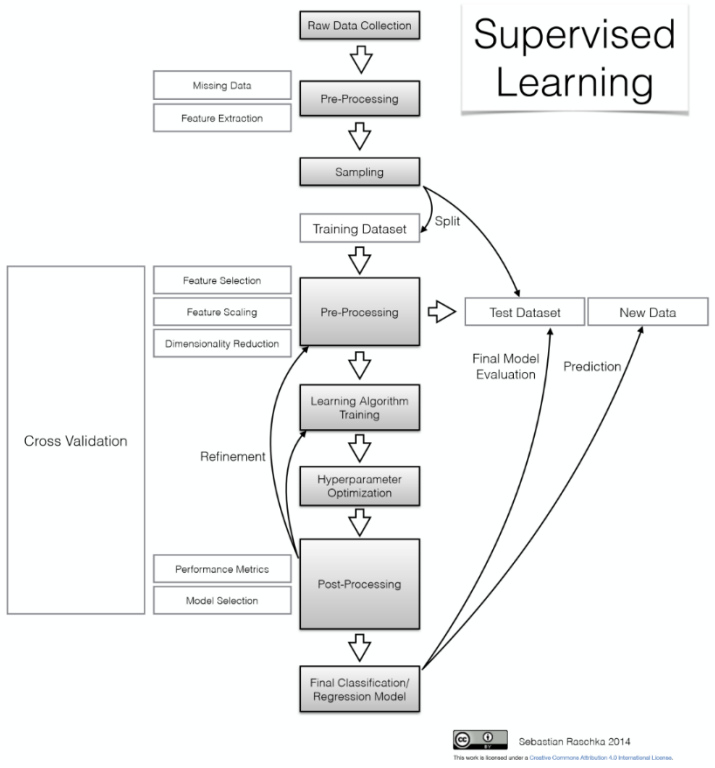
Learning About Outcomes

- Functions

  Outcome = Function of Attributes

- Sometimes functions are hard to interpret and describe
- Sometimes we can describe and interpret functions as concepts or rules
- Sometimes we can more directly learn simple descriptions

Supervised Learning



Basic Methods

- Nearest Neighbor (instance based or lazy learner)
- 1R (rule learner)
- Naïve Bayes (a statistical rule)
- Decision Tree (divide and conquer)
- Hands-on Decision Trees

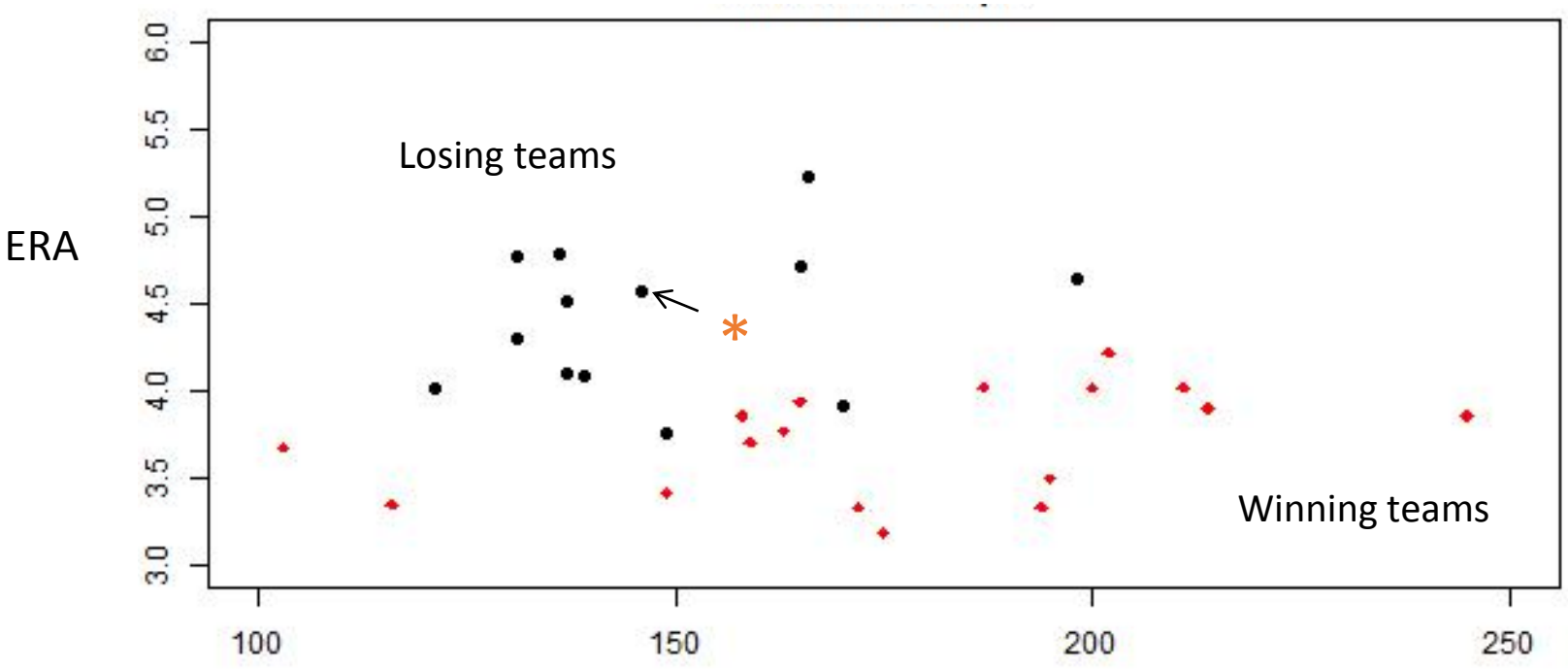
Simplicity first

- Simple algorithms often work surprisingly well
- Different kinds of strategies exist:
  - One attribute
  - All attributes - equal importance
  - A linear combination (ie linear function)
- Different kinds of conceptual descriptions
  - An instance-based representation
  - Simple logical structures

1 Nearest Neighbor

Instance Based Learning: Given new data point, classify by Nearest Neighbor's class

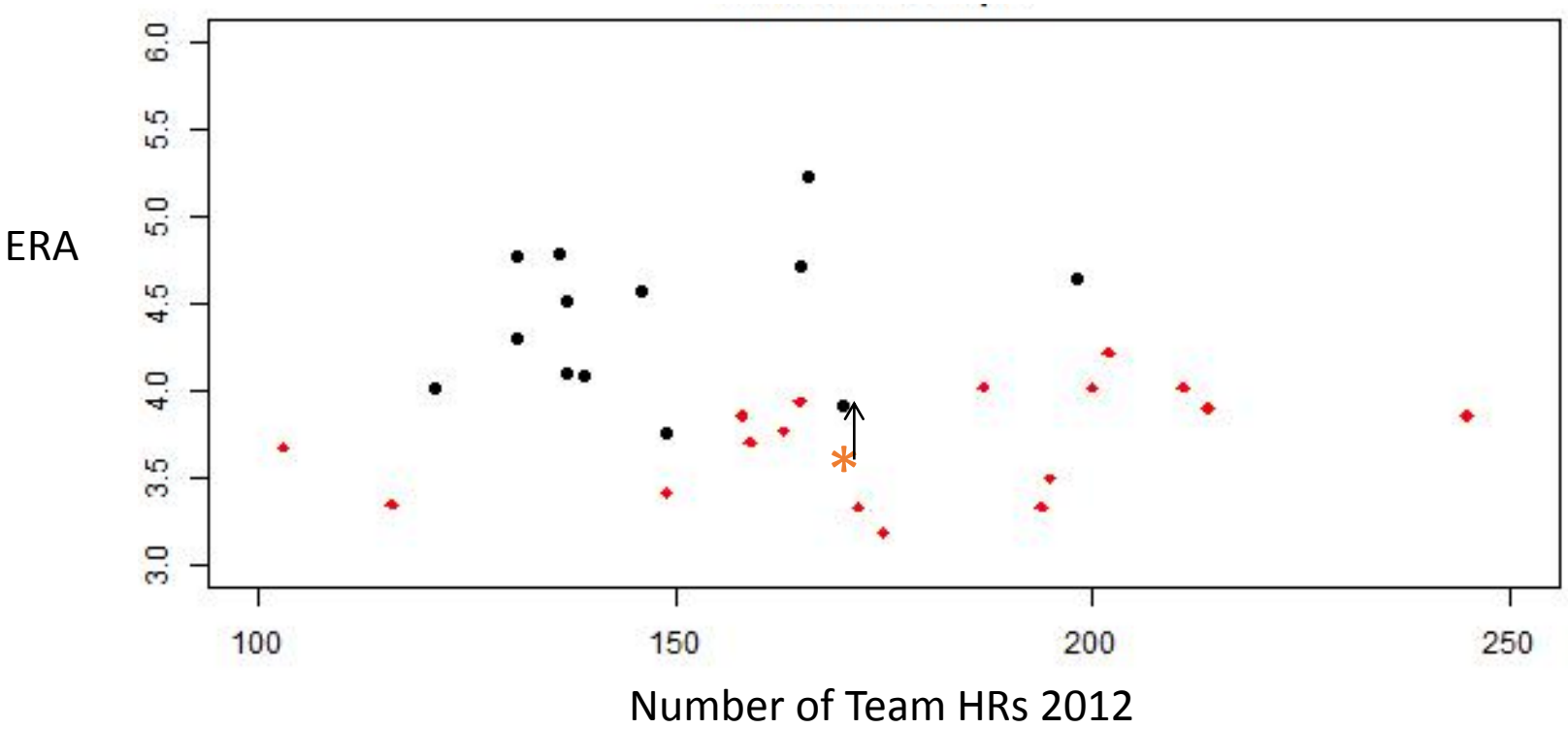
(depends on distance metric, so scale variables)



(not well scaled, dominates ERA)

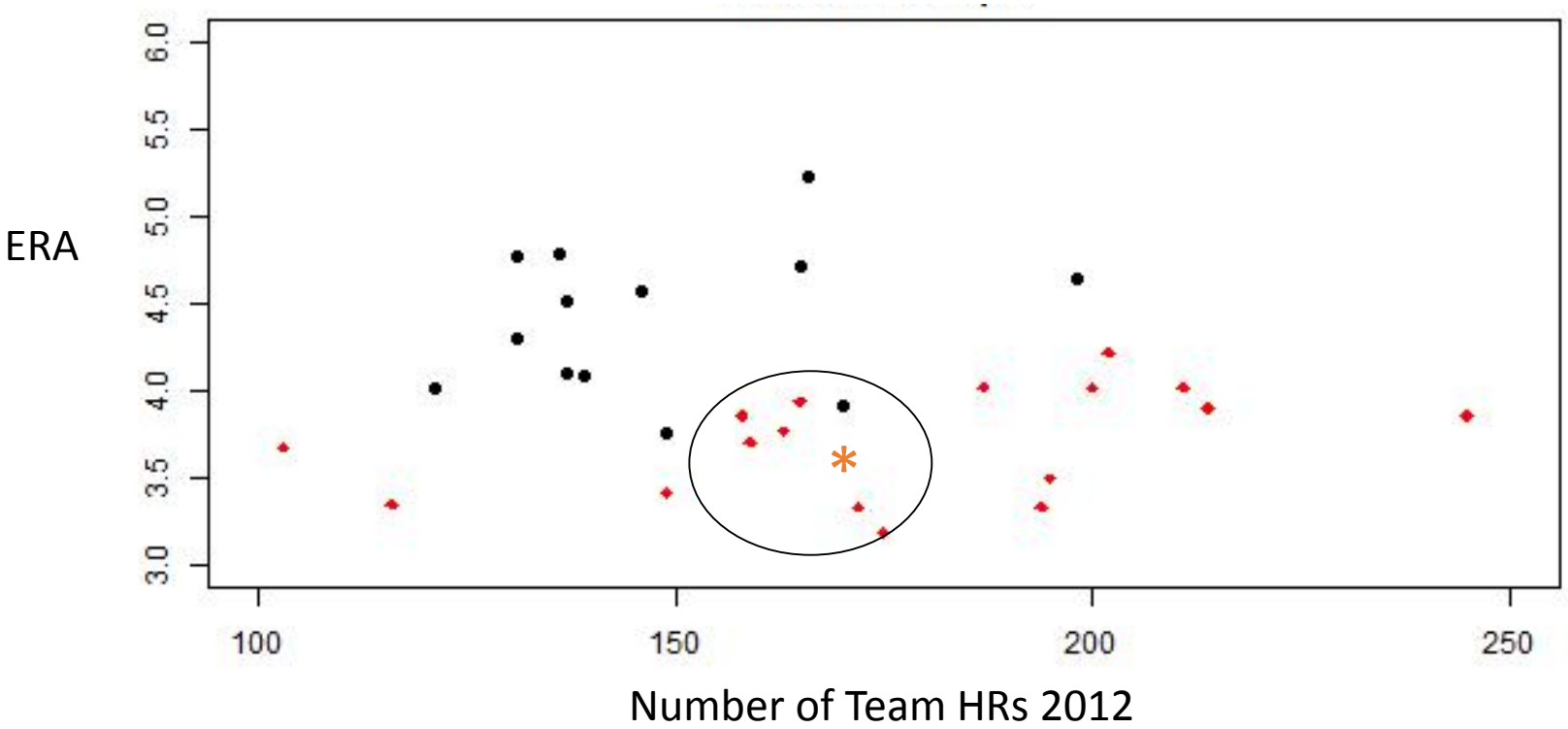
1NN issue

Idea: 1 NN might be too sensitive, maybe use k NN



1NN issue

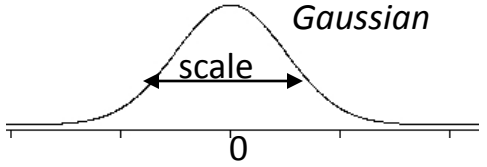
Idea: take most common outcome from k NNs
(note: k=all points is just mean)



kNN

- kNN uses k points as neighbors, but how to choose k?
  - Cross-validation

- Kernel method: weight all points inversely by distance
  - e.g. weight ~ exp(distance($\boldsymbol{x_i}, \boldsymbol{xj}$)$/scale$)
  - Need to choose scale
  - Can be more robust, not as interpretable



Rudimentary rules

- 1R: learns a 1-level decision tree
  - Set of rules that all test on one particular attribute
- Basic version
  - One branch for each of the attribute's values
  - Each branch assigns most frequent class
  - Error rate: proportion of instances that don't belong to the majority class of their corresponding branch
  - Choose attribute with lowest error rate

Pseudo-code for 1R

```
For each attribute
    For each value of the attribute make a rule:
            count how often each class appears
            find the most frequent class
            assign that class to this attribute-value
    Calculate the error rate of the rules
Choose the rules with the smallest error rate
```

Weather Data Set

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

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

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temperature	Hot → No*	2/4	5/14
	Mild → Yes	2/6	
	Cool → Yes	1/4	
Humidity	High → No	3/7	4/14
	Normal → Yes	1/7	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

Summary

- 1R was described in a paper by Holte (1993)
- 16 datasets
- Minimum number of instances was set to 6
- 1R's simple rules performed not much worse than much more complex decision trees
- "Simplicity first" pays off!

Statistical modeling

- “Opposite” of 1R: use all the attributes
- Two assumptions: Attributes are
  - equally important
  - statistically independent
- Knowledge about the value of a particular attribute doesn’t tell us anything about the value of another attribute (if the class is known)
- Assumptions that are almost never correct
  - scheme works well in practice!

Weather Data Set

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Weather Data Counts and Probabilities

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

A new day to be classified

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Likelihood of the New Day Outcome

Likelihood of the two classes

For "yes" = $2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$

For "no" = $3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0206$

Conversion into a probability by normalization:

P("yes") = 0.0053 / (0.0053 + 0.0206) = 0.205

P("no") = 0.0206 / (0.0053 + 0.0206) = 0.795

Bayes's rule

- Probability of event H given evidence E:

$$\Pr[H \mid E] = \frac{\Pr[E \mid H]\Pr[H]}{\Pr[E]}$$

- A priori probability of H:
  - Probability of event before evidence has been seen

$$\Pr[H]$$

- A posteriori probability of H:
  - Probability of event after evidence has been seen

$$\Pr[H \mid E]$$

Naïve Bayes for classification

- Classification learning: what's the probability of the class given an instance?
- Evidence E = instance
- Event H = class value for instance
- Naïve Bayes assumption: evidence can be split into independent parts

$$\Pr[H \mid E] = \frac{\Pr[E_1 \mid H]\Pr[E_1 \mid H]\ldots\Pr[E_n \mid H]\Pr[H]}{\Pr[E]}$$

Evidence:

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Pr[*yes*|E]=

Pr[*Outlook=Sunny*|yes] x

Pr[*Temperature=Cool*|*yes*] x Pr[*Humidity=High*|*yes*]

x Pr[*Windy=True*|*yes*] x $\frac{\Pr[yes]}{\Pr[E]}$

Probabilities for class YES

$$= \frac{2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14}{\Pr[E]}$$

Summary

- Naïve Bayes works amazingly well
  - Violated independence assumption
- Because classification doesn't require accurate probability estimates as long as maximum probability is assigned to correct class
- Problem: Adding too many redundant attributes
  - Example: identical attributes
- Conditional Probability visualization:

  http://setosa.io/conditional/

DECISION TREE INDUCTION

- Method for approximating discrete-valued functions
  - robust to noisy/missing data
  - can learn non-linear relationships
  - inductive bias towards shorter trees

Decision trees

- “Divide-and-conquer” approach
- Nodes involve testing a particular attribute
- Attribute value is compared to
  - Constant
  - Comparing values of two attributes
  - Using a function of one or more attributes
- Leaves assign classification, set of classifications, or probability distribution to instances
- Unknown instance is routed down the tree

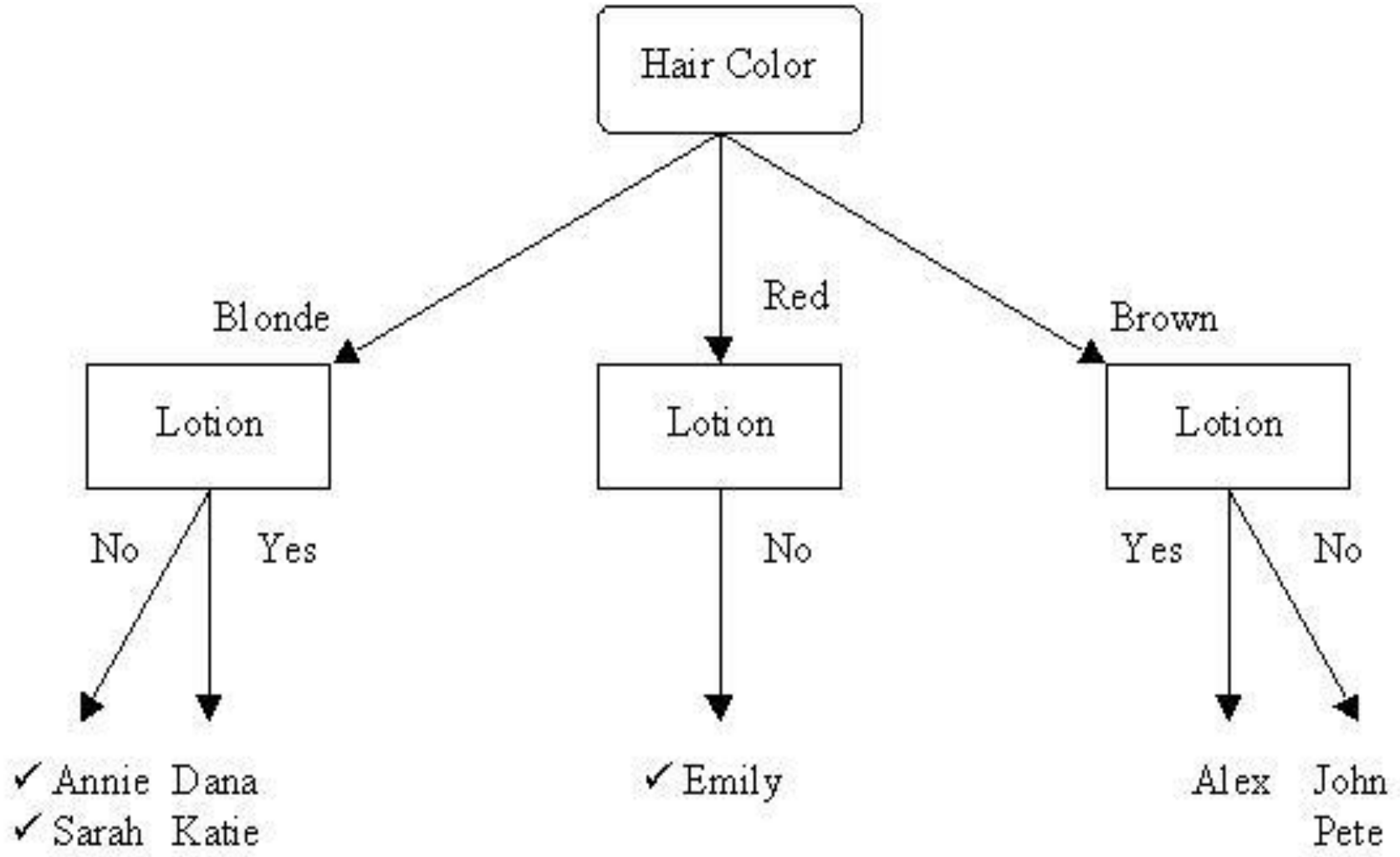
Decision Tree Learning

- Applications:
  - medical diagnosis – ex. heart disease
  - analysis of complex chemical compounds
  - classifying equipment malfunction
  - risk of loan applicants
  - Boston housing project – price prediction

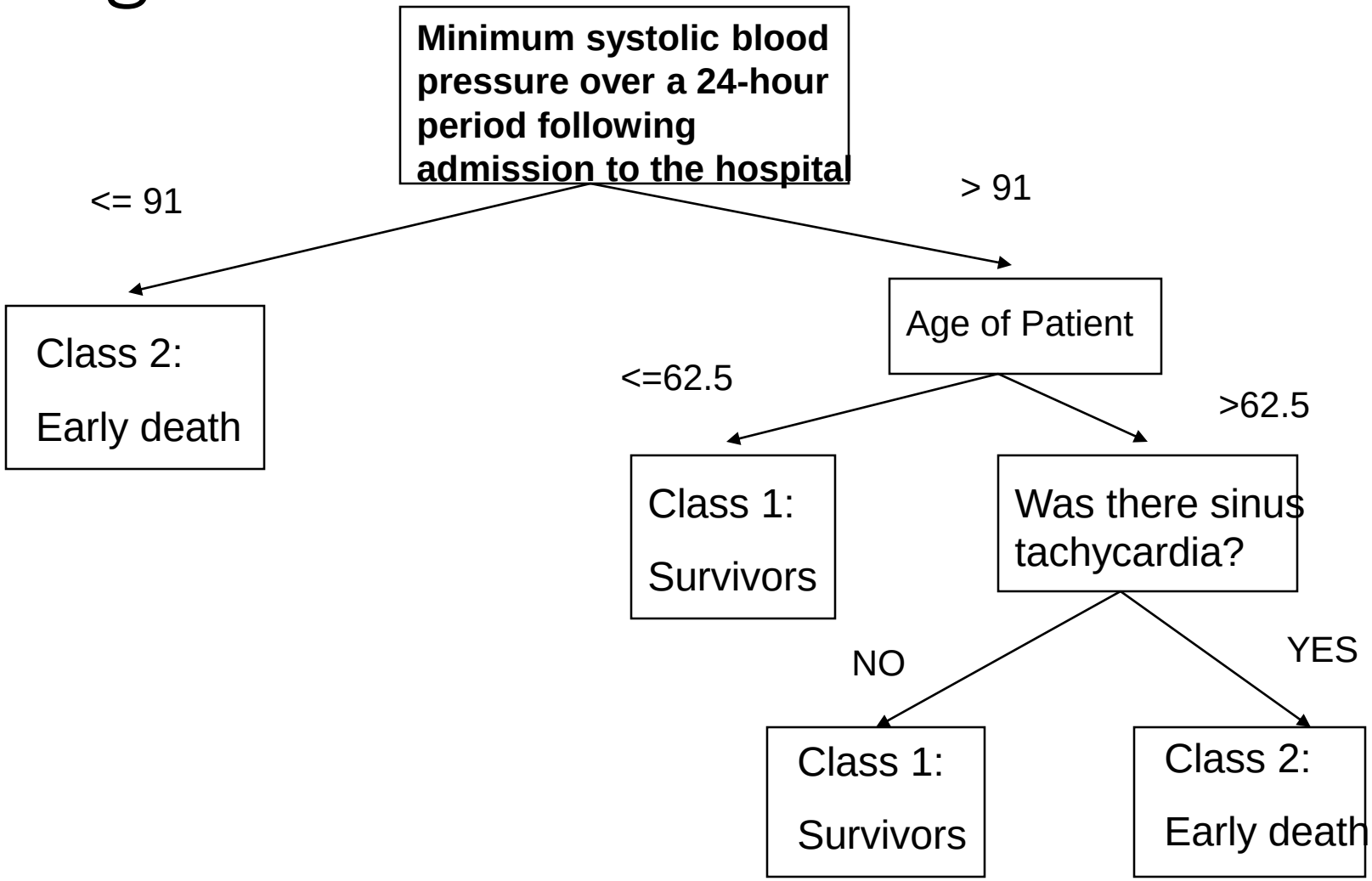
DECISION TREE FOR THE CONCEPT *"Sunburn"*

Name	Hair	Height	Weight	Lotion	Result
Sarah	blonde	average	light	no	sunburned (positive)
Dana	blonde	tall	average	yes	none (negative)
Alex	brown	short	average	yes	none
Annie	blonde	short	average	no	sunburned
Emily	red	average	heavy	no	sunburned
Pete	brown	tall	heavy	no	none
John	brown	average	heavy	no	none
Katie	blonde	short	light	yes	none

DECISION TREE FOR THE CONCEPT *"Sunburn"*



DT for Medical Diagnosis and Prognosis Heart Disease



Beriman et. al, 1984

Occam's Razor

- "The world is inherently simple. Therefore the smallest decision tree that is consistent with the samples is the one that is most likely to identify unknown objects correctly"

Decisions Trees Representation

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

When to Consider Decision Trees

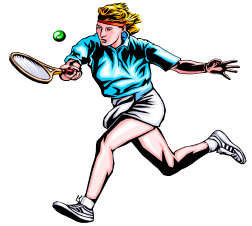
- Instances describable by attribute--value pairs
  - each attribute takes a small number of disjoint possible values
- Target function has discrete output value
- Possibly noisy training data
  - may contain errors
  - may contain missing attribute values

Weather Data Set-Make the Tree!

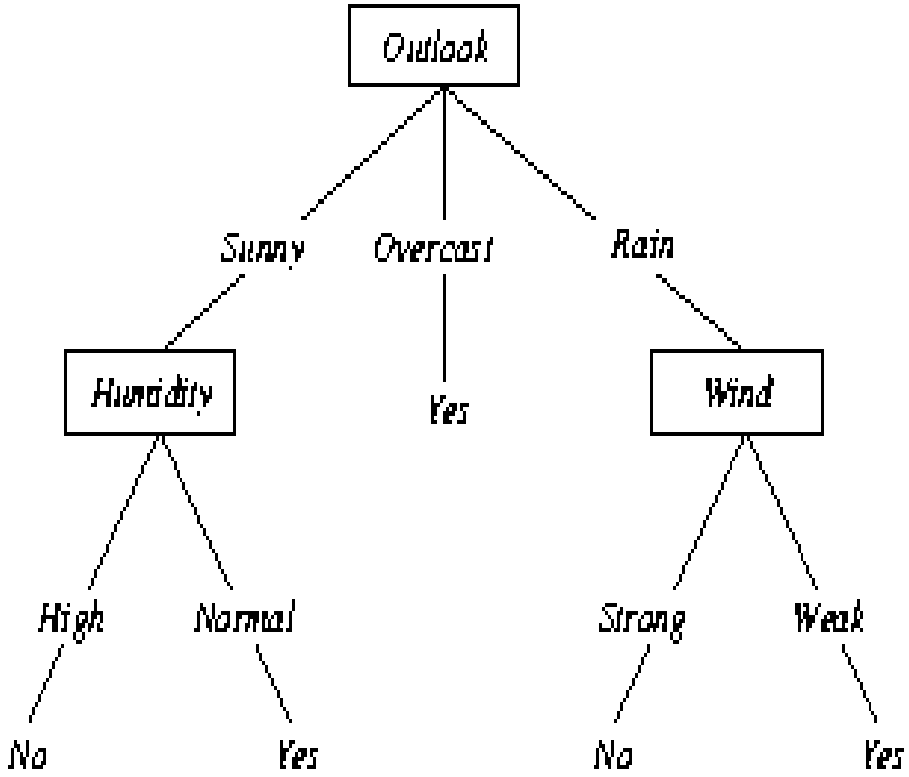
Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

DECISION TREE FOR THE CONCEPT

"Play Tennis"



Day	Outlook	Temp	Humidity	Wind	PlayTenni
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcas	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcas	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcas	Mild	High	Strong	Yes
D13	Overcas	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



[Mitchell,1997]

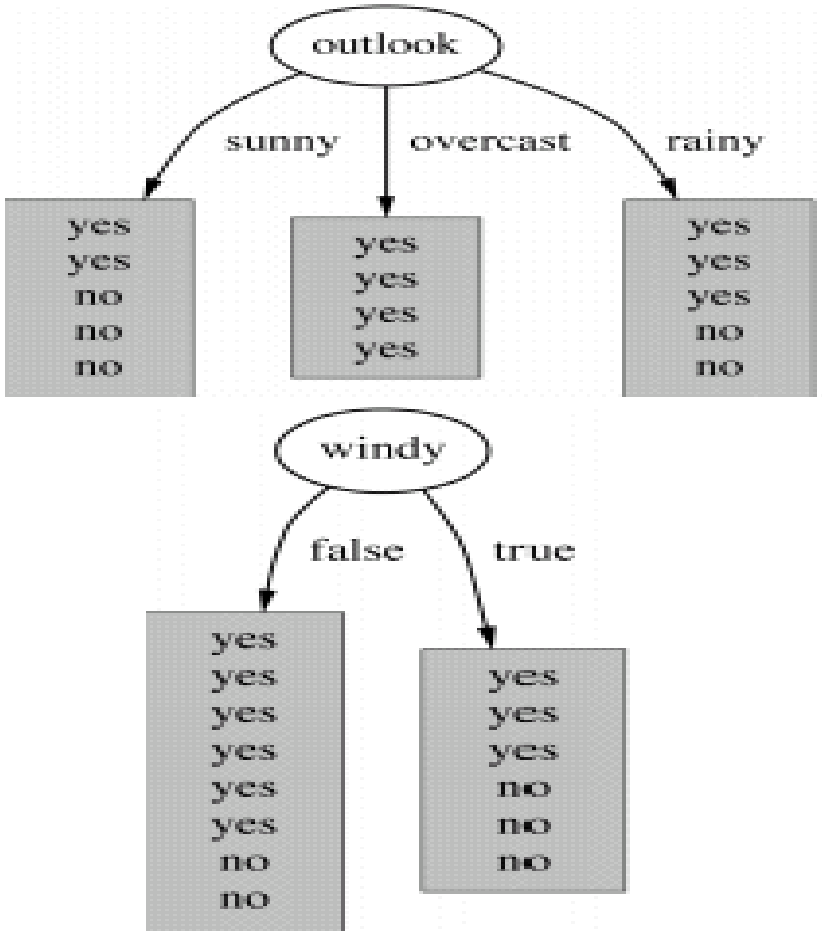
Constructing Decision Trees

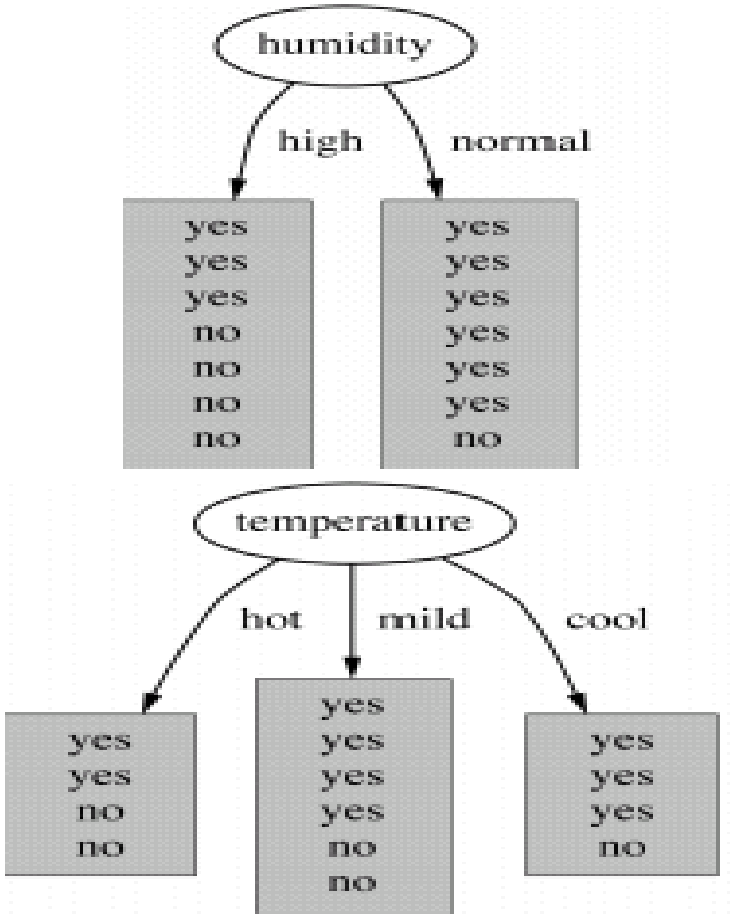
- Normal procedure: top down in recursive divide-and-conquer fashion
- First: attribute is selected for root node and branch is created for each possible attribute value
- Then: the instances are split into subsets (one for each branch extending from the node)
- Finally: procedure is repeated recursively for each branch, using only instances that reach the branch
- Process stops if all instances have the same class

Induction of Decision Trees

- Main Recursive loop:
  - Pick the “best'' attribute to split the data at current decision node, according to some measure
  - For each value of attribute, create new leaf node descendants of current node
  - Sort data to new leaf nodes
  - For each leaf node:
    - If training examples perfectly classified,
    - Then STOP
    - Else Iterate at current leaf node as next decision node

Which is the best attribute?





Attribute selection

- How to choose the best attribute?
  - Smallest tree
  - Heuristic: Attribute that produces the “purest” nodes
- Impurity criterion:
  - Information gain
    - Increases with the average purity of the subsets produced by the attribute split
- Choose attribute that results in greatest information gain

Computing information

- Information is measured in bits

- Given a probability distribution, the info required to predict an event is the distribution's entropy
  - e.g. highly predictable events => low entropy
  - e.g. random probabilities => higher entropy

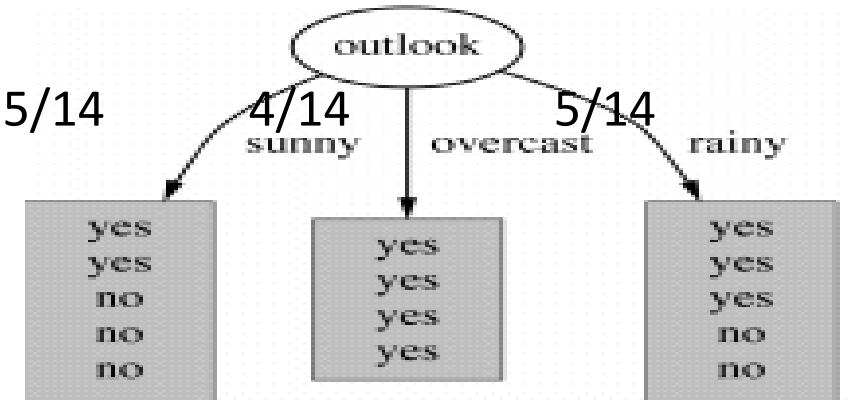
- Formula for computing the entropy for n classes:

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

Expected information for attribute *"Outlook"*

- "*Outlook*" = "Sunny"
- "*Outlook*" ="Overcast"
- "*Outlook*" ="Rainy"
- Total expected information: is weighted avg. of entropy at each node branch,

Branch probs
are the weights: 5/14



Class probs
used for each Entropy: 2/5,3/5 4/4,0/4 3/5,2,5

Computing the information gain

- Information gain: information before splitting – information after splitting

- Test the Info Gain for outlook at root node:

Gain("Outlook")=entropy of classes –

avg. entropy of classes at new leaf nodes

= Ent(9/14,5/14) –

5/14*Ent(2/5,3/5) + 4/14*Ent(4/4,0/4)+

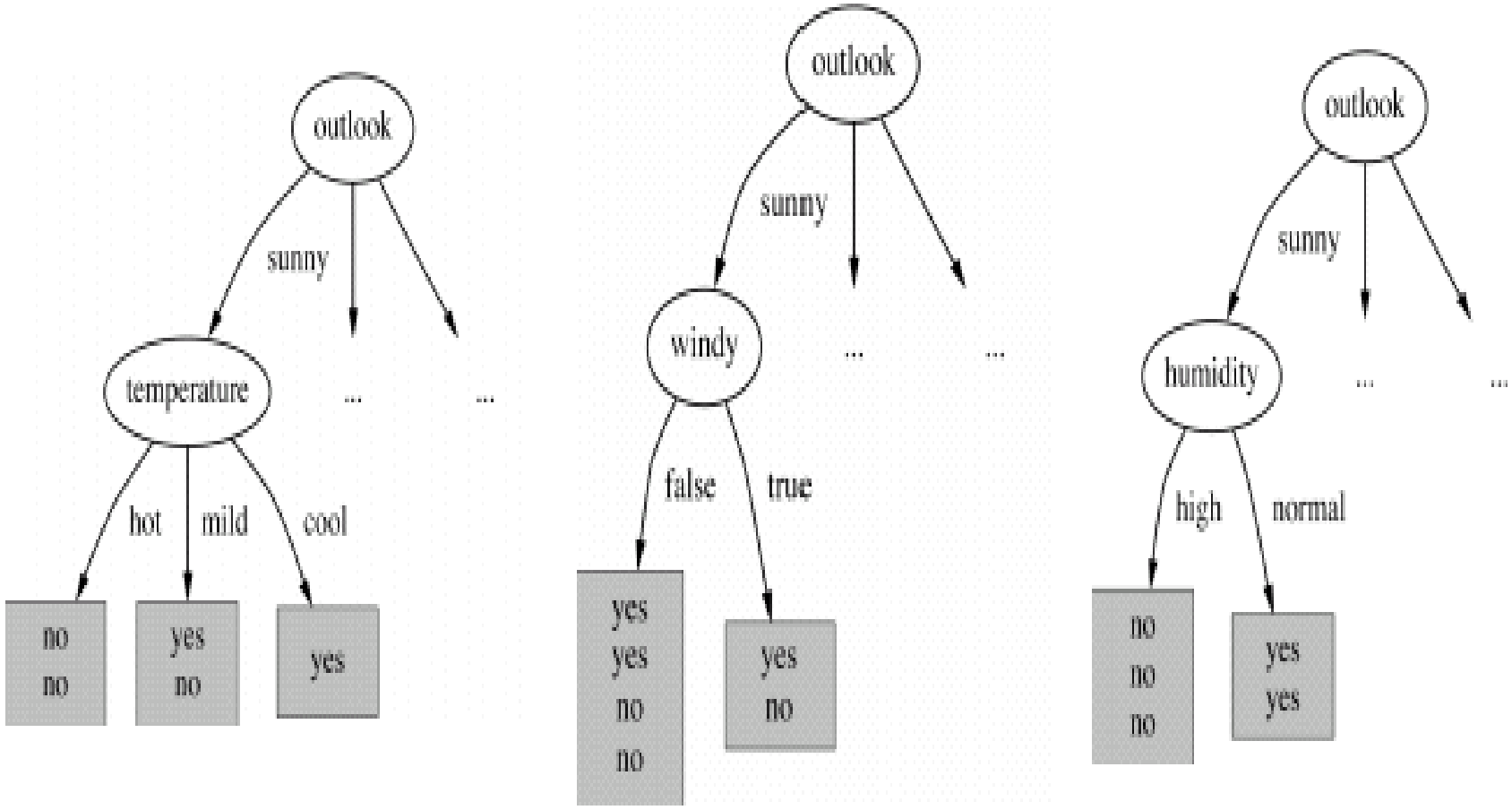
5/14*Ent(3/5,2/5)

0.940-0.693 = 0.247 bits

Computing the information gain

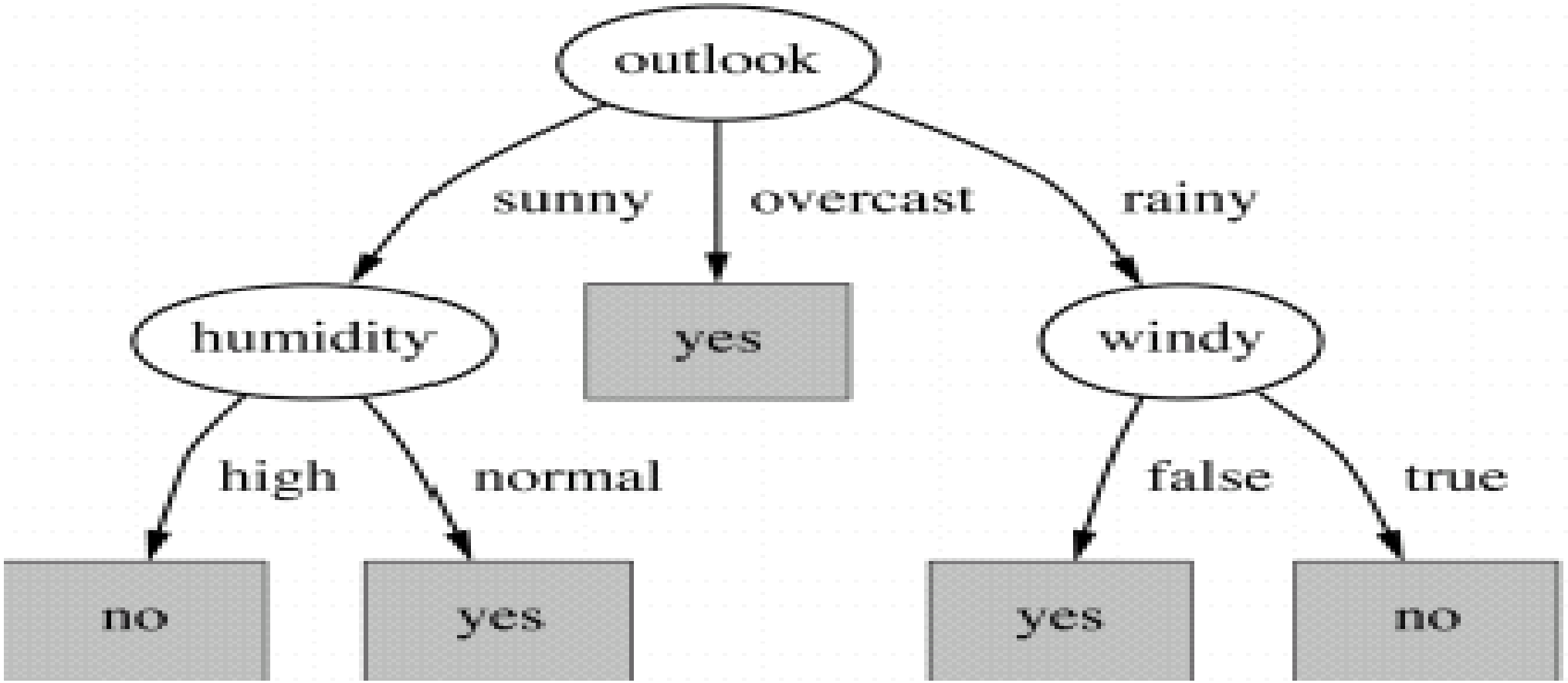
- Information gain for attributes from weather data:
  - Gain (“Outlook”) = 0.247 bits
  - Gain (“Temp”) = 0.029 bits
  - Gain (“Humidity”) = 0.152 bits
  - Gain (“Windy”) = 0.048 bits

Further splits



Gain (“Temp”)=0.571 bits Gain (“Humidity”)=0.971 Gain(“Windy”)=0.020 bits

Final product



Purity measure

- Desirable properties
  - Pure Node -> measure = zero
  - Impurity maximal -> measure = maximal
  - Multistage property
    - decisions can be made in several stages
    - $\text{measure}([2,3,4]) = \text{measure}([2,7]) + (7/9) \times \text{measure}([3,4])$
- Entropy is the only function that satisfies all the properties

Other Heuristics for Node Selection

- Gini Index over data S: $(1 - \sum_j p_j^2)$

  Similar to behavior to entropy, maximal for random p, minimal for high p

  Average Gini index for new split in S_i subsets:

$$=\sum_i |S_i|/|S| \cdot (1 - \sum_j p_j)$$

- Conditional Tests

  Test error before/after each new possible node for statistical significance (pre-prune tree)

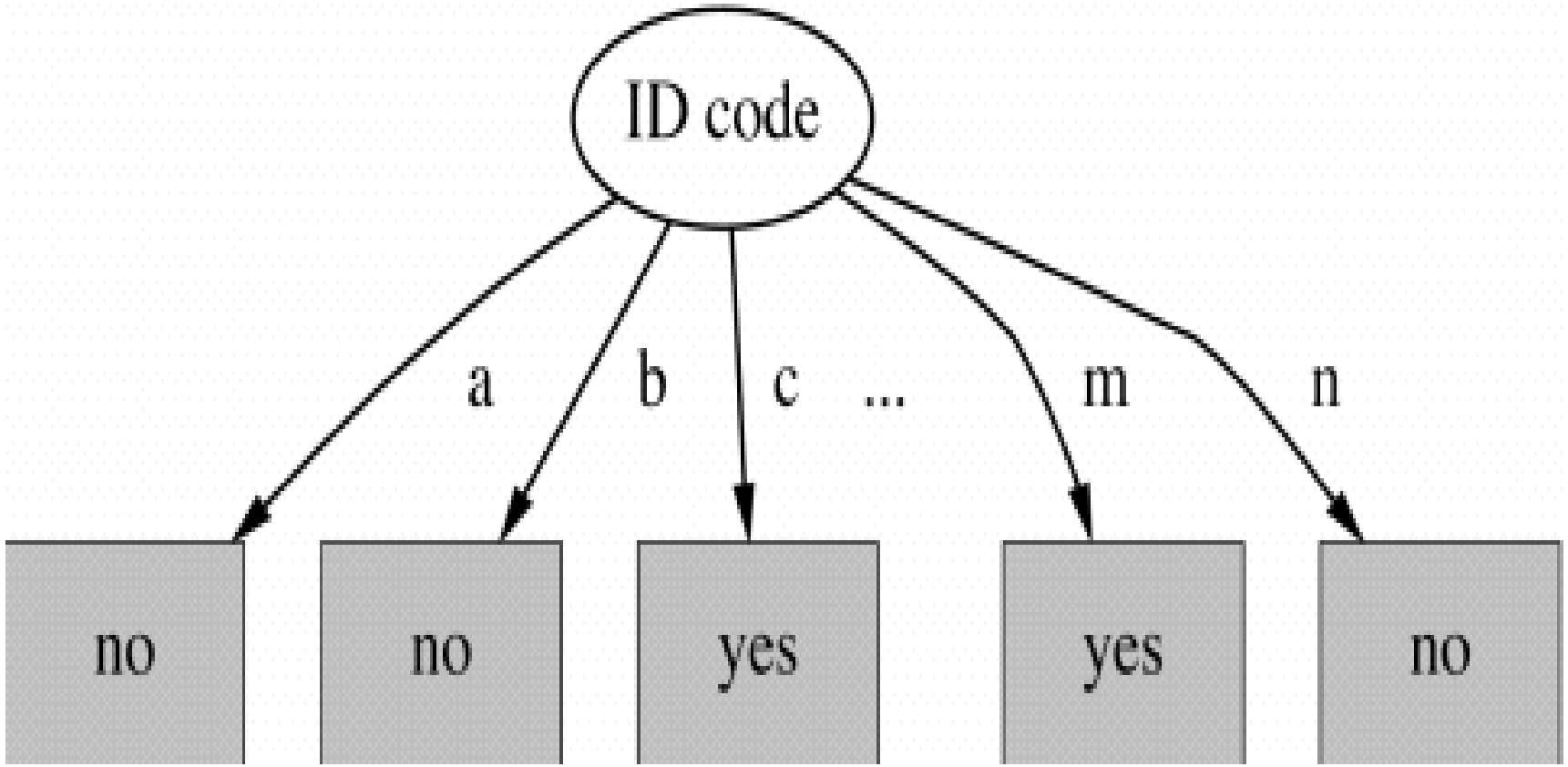
Highly-branching attributes

- Attributes with a large number of values
  - example: ID code
- Subsets more likely to be pure if there is a large number of values
  - Information gain biased towards attributes with a large number of values, (especially as leaf node training examples get smaller)
  - Overfitting

New version of Weather Data

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

ID Code Attribute Split



Info([9,5]) = 0.940 bits

Gain ratio

- Modification that reduces its bias

  Info Gain / Intrinsic Info

- Intrinsic information:

  Entropy of attribute based on sample values at that node

  (not the class values at leaf node)

  e.g. An ID variable will have high intrinsic info, thereby shrinking the Info Gain

  It helps usually (but not necessarily)

Summary

- Algorithm for top-down induction of decision trees
- “ID3” was developed by Ross Quinlan
- C4.5 incorporate
  - numeric attributes, missing values, and noisy data
  - J48 is the WEKA java implementation
- CART Breiman, Friedman, Olshen, Stone
- Many other variations

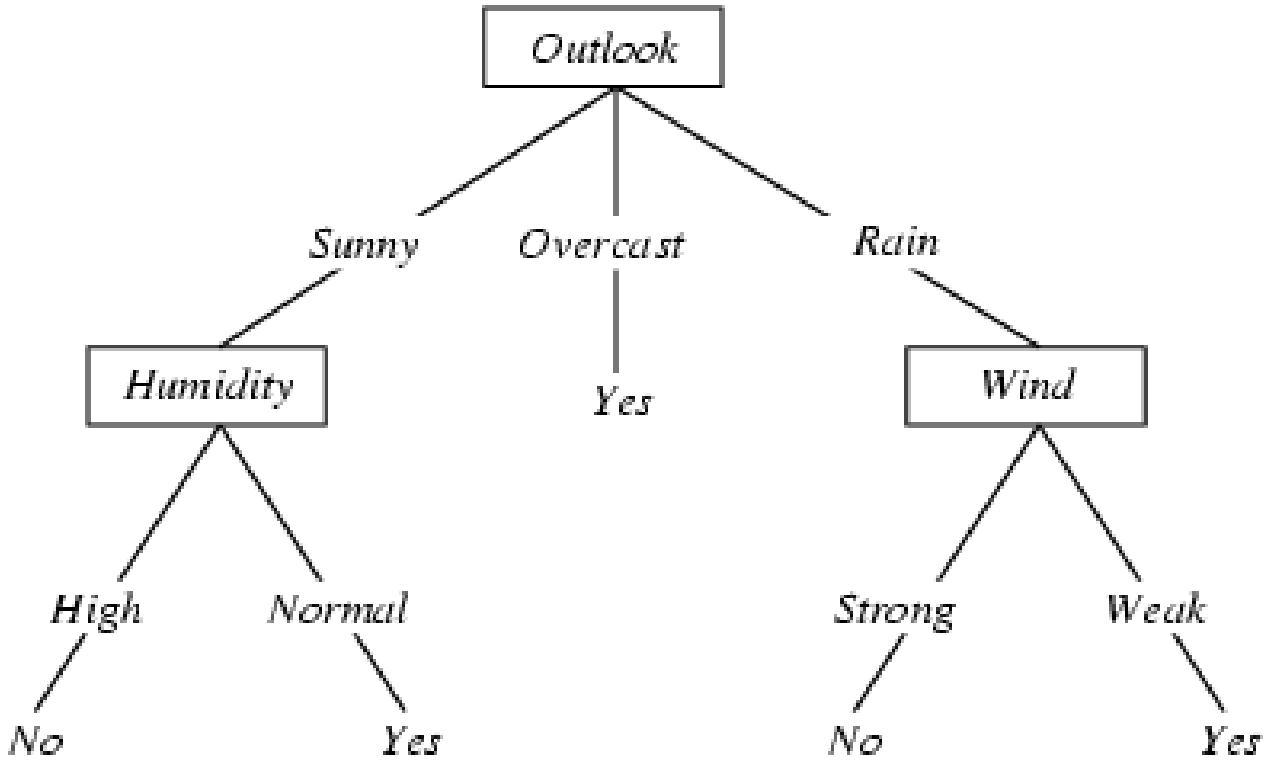
Avoid Overfitting

- How can we avoid Overfitting:
  - Stop growing when data split not statistically significant
  - Grow full tree then post-prune
- How to select best tree?
  - Measure performance over training data
  - Measure performance over separate validation data set

Pruning

- Pruning simplifies a decision tree to prevent overfitting to noise in the data
- Pre-pruning:
  - stops growing a branch when information becomes unreliable
- Post-pruning:
  - discard subtrees if they don’t improve error estimates
  - Error estimates are derived from cross-validation or statistical assumptions
- Post-pruning preferred in practice because of early stopping in pre-pruning

Converting Tree to rules



Converting Tree to rules

$$\begin{array}{ll} \text{IF} & (Outlook = Sunny) \wedge (Humidity = High) \\ \text{THEN} & PlayTennis = No \end{array}$$

$$\begin{array}{ll} \text{IF} & (Outlook = Sunny) \wedge (Humidity = Normal) \\ \text{THEN} & PlayTennis = Yes \end{array}$$

$\ldots$

Thank you!