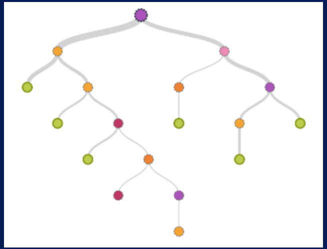
Decision Tree Induction

Method for approximating discrete-valued functions

- Robust to noisy/missing data
- Learn non-linear relationships
- Inductive bias towards shorter trees

Decision Trees



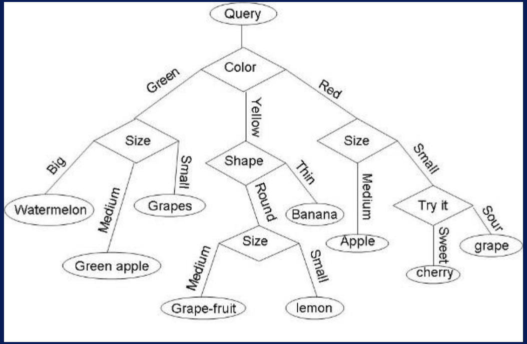
“Divide-and-conquer” approach

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

Decision Trees



Leaves assign
- classification
- set of classifications
- probability distribution to instances

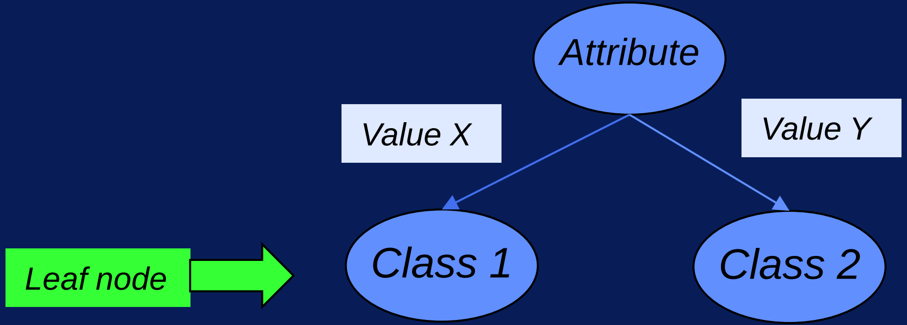
Unknown instance is routed down the tree

Decisions Trees Representation

Each internal node tests an attribute

Each branch corresponds to attribute value

Each leaf node assigns a classification



Decision Tree Applications

Medical diagnosis – ex. heart disease

Analysis of complex chemical compounds

Classifying equipment malfunction

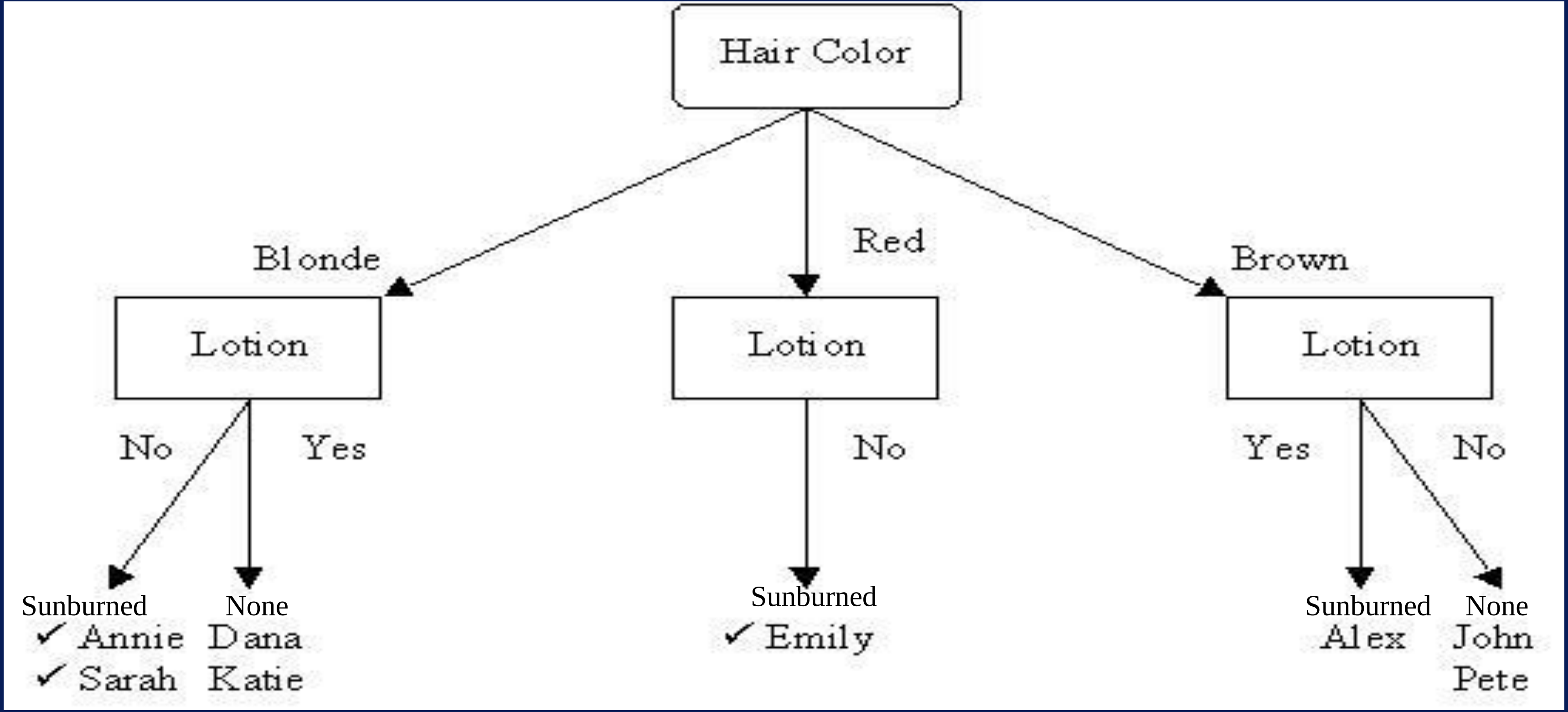
Risk of loan applicants

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*"



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”

When to Consider Decision Trees

Instances can be represented as attribute--value pairs

Target function has discrete output value

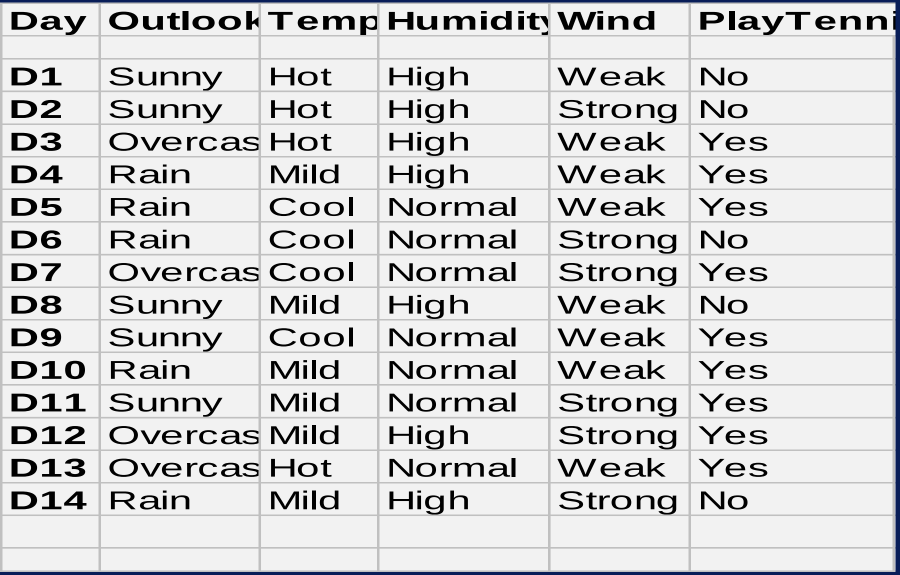
Possibly noisy training data

Weather Data Set

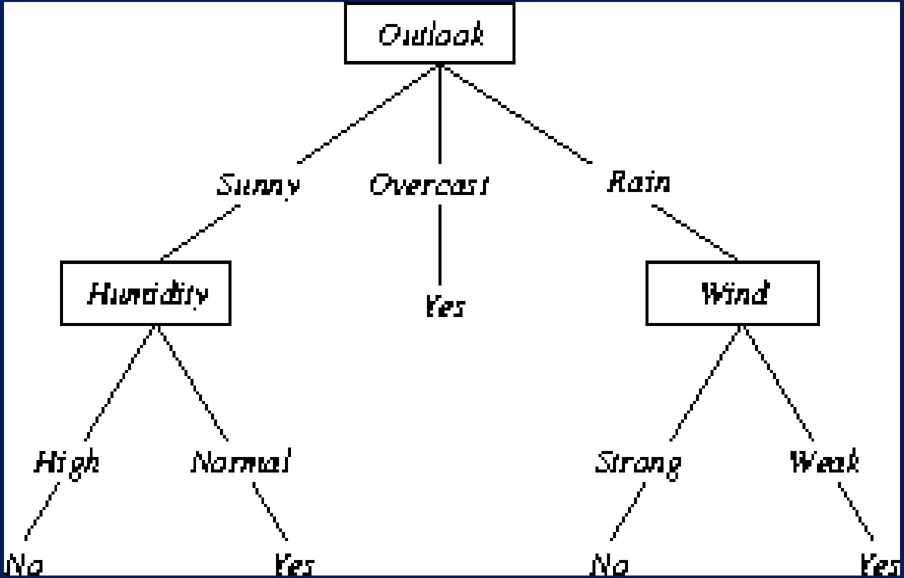
Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes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

DECISION TREE FOR THE CONCEPT

"Play Tennis"



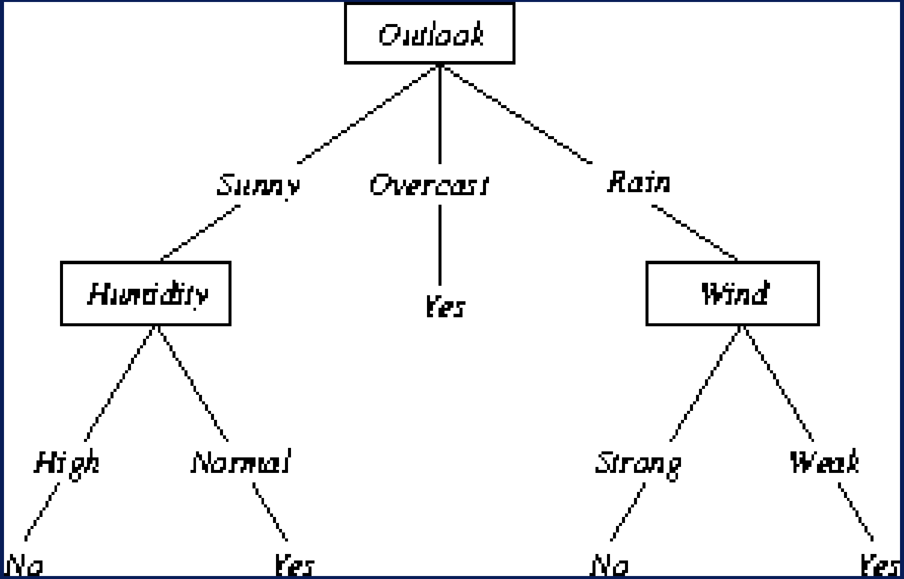
Day	Outlook	Temp	Humidity	Wind	PlayTenni
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcas	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcas	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes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

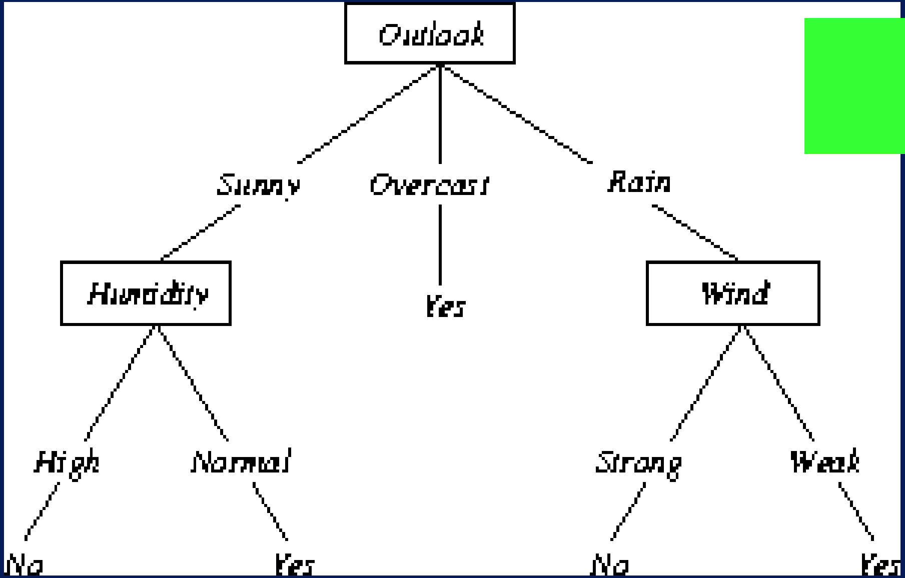
In Lecture Question

Did you notice anything particular about the tree?



In Lecture Question

Did you notice anything particular about the tree?



Where is Temperature?

Lesson #2

Constructing Decision Trees

Constructing Decision Trees

Regular procedure

top down in recursive divide-and-conquer fashion

Constructing Decision Trees

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)

Constructing Decision Trees

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

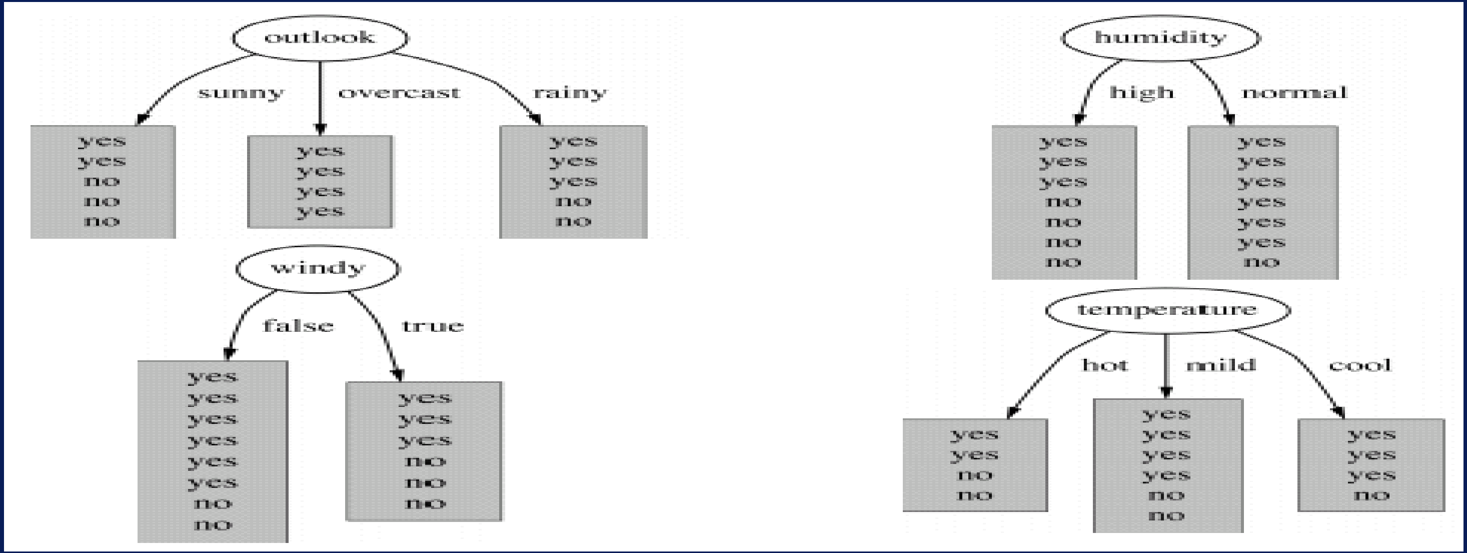
For each value of attribute, create new leaf node descendants of current node

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

Attribute Selection

- **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

highly predictable events => low entropy

random probabilities => higher entropy

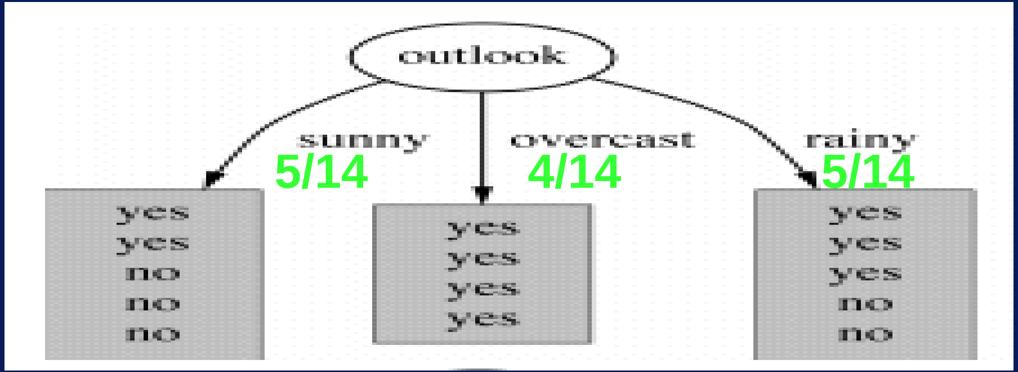
Formula for computing the entropy for n classes:

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

Expected information for attribute "*Outlook*"

Total expected information: is weighted avg. of entropy at each node branch

Branch probabilities are the weights:



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

Information Gain for Outlook at Root Node

Calculating Information gain
information before splitting – information after splitting

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

Outlook = Sunny

Outlook = Overcast

Outlook = Rainy

Computing the Information Gain for Each Attribute

Information gain for attributes from weather data:

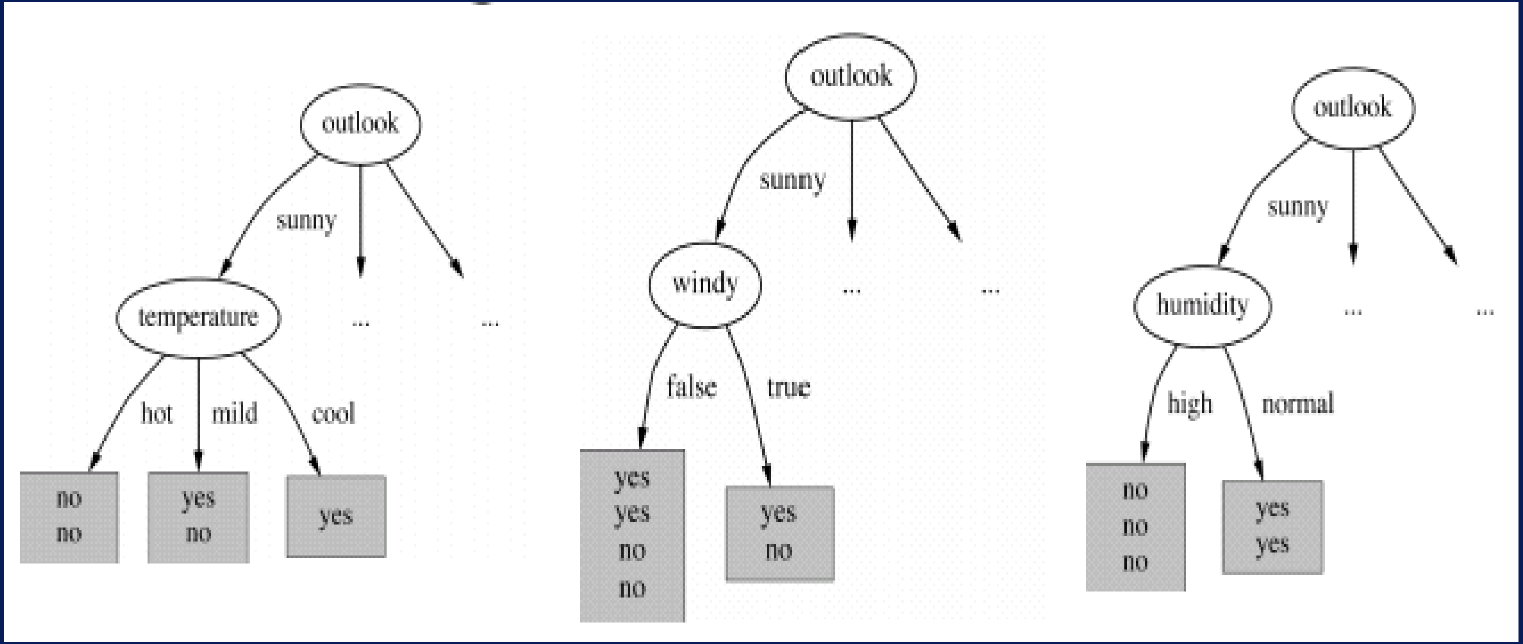
➡ Gain (“Outlook”) = 0.247 bits

Gain (“Temp”) = 0.029 bits

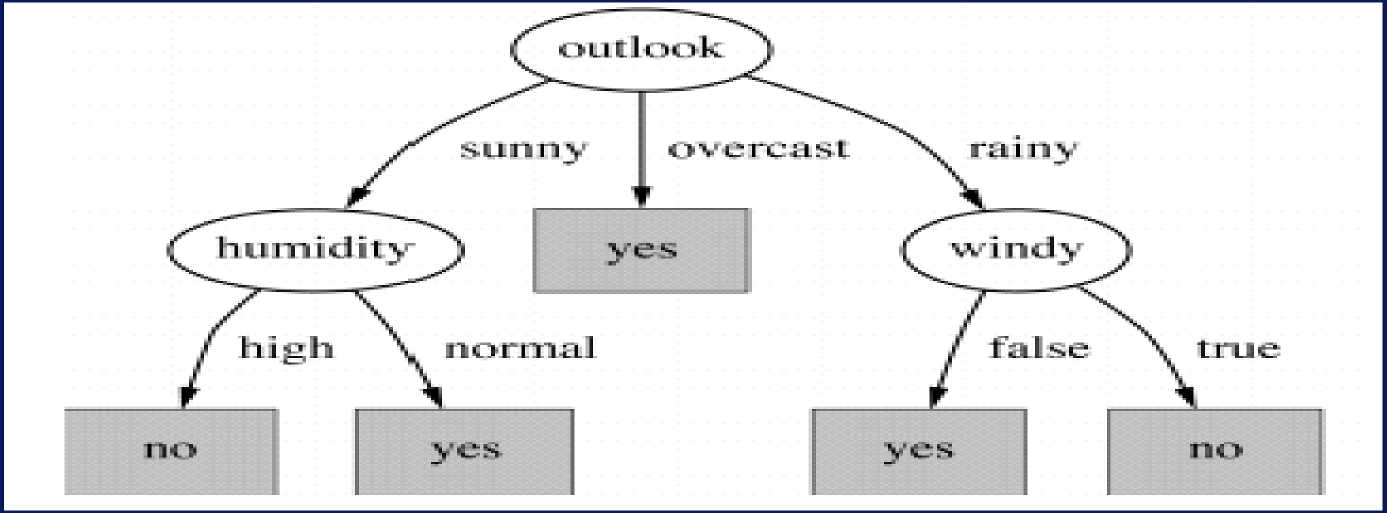
Gain (“Humidity”) = 0.152 bits

Gain (“Windy”) = 0.048 bits

Further splits



Final Product



Purity Measure

Desirable properties

- Pure Node -> measure = zero
- Impurity maximal -> measure = maximal
- Multistage property
  - decisions can be made in several stages
  - measure([2 ,3,4])= measure([2,7])+(7/9) ´ measure([3,4])

Entropy is the 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:

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

Lesson #3

Overfitting and Other concerns with DTs

Highly-branching attributes

Attributes with a large number of values

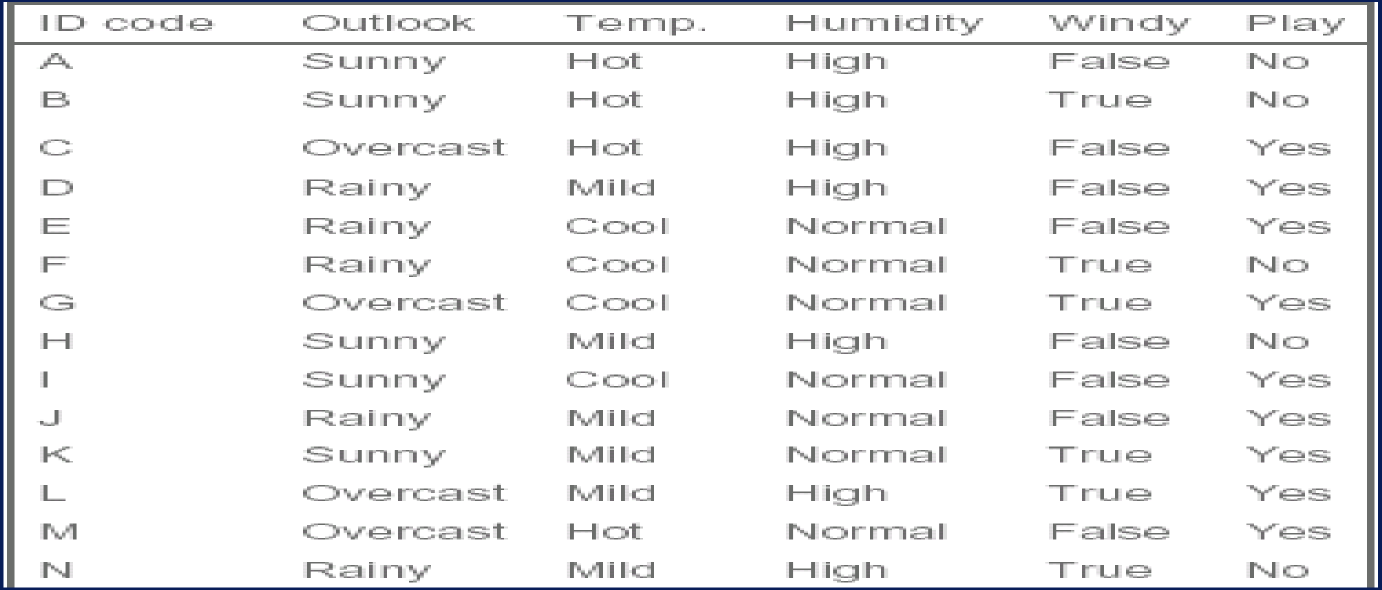
- example: ID code

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

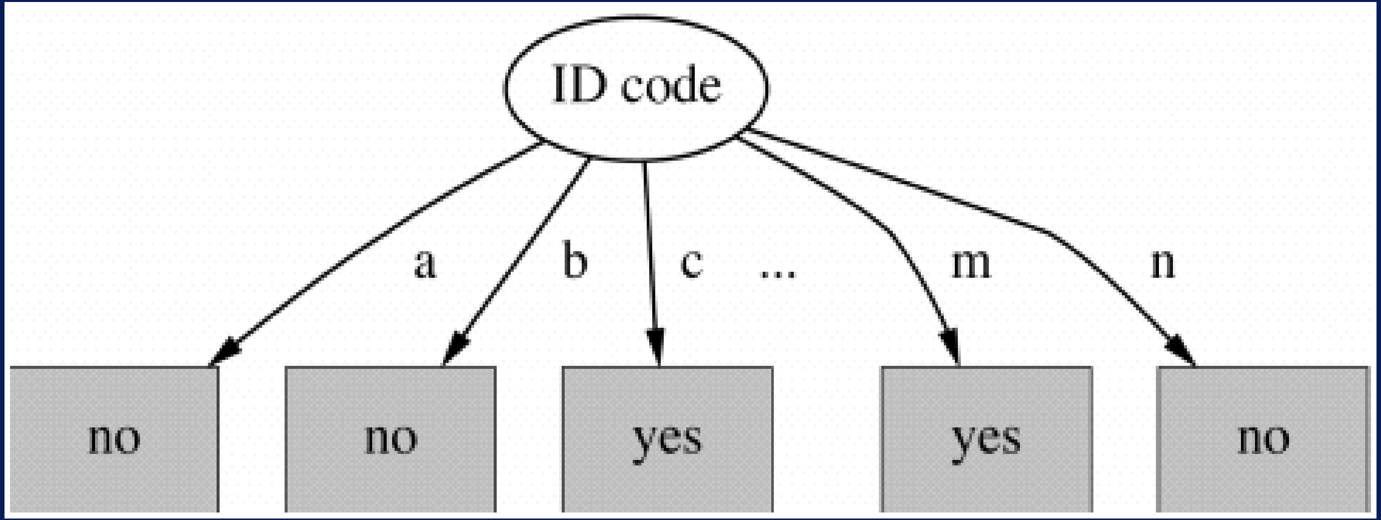
- Overfitting

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)

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

Post-pruning

Pruning

Pre-pruning

- stops growing a branch when information becomes unreliable

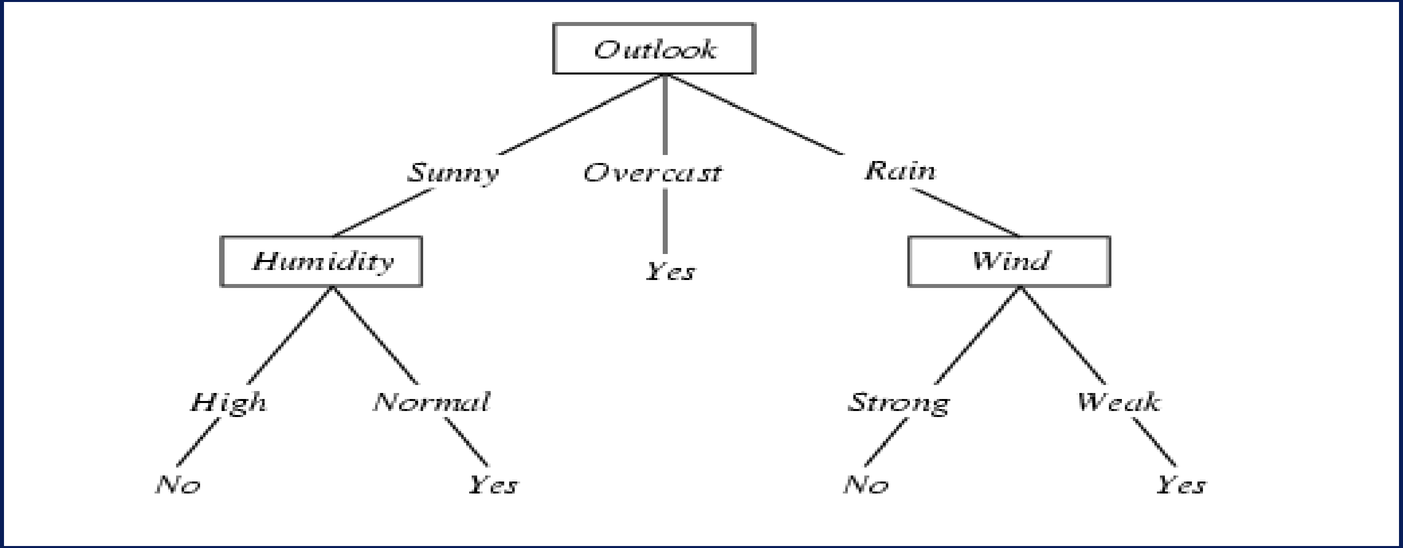
Post-pruning

- discard subtrees if they don’t improve error estimates
- Error estimates are derived from cross-validation or statistical assumptions

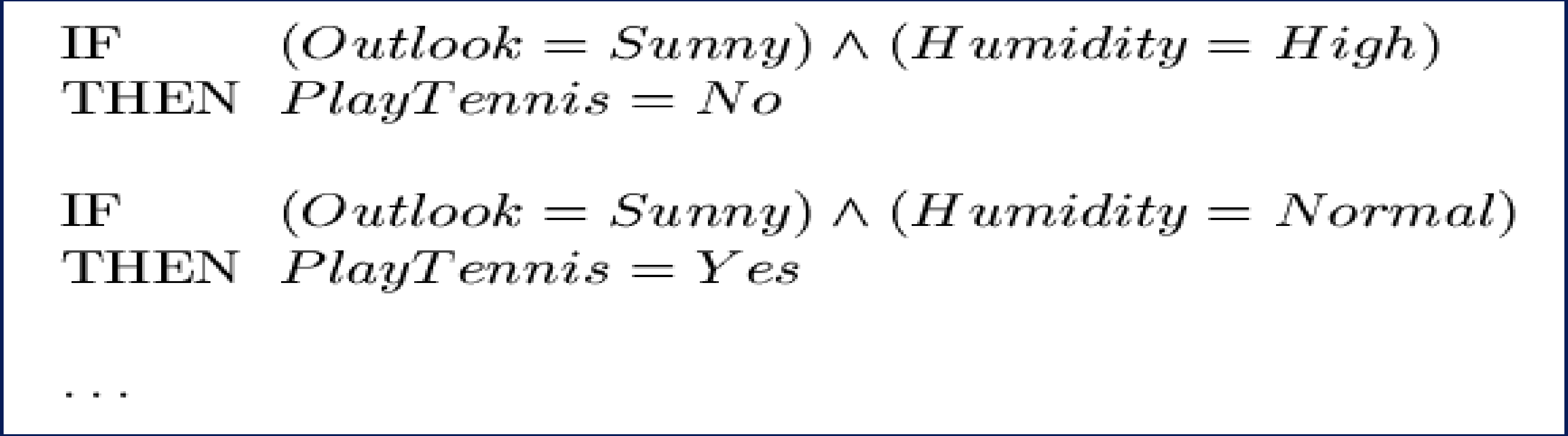
Pruning

Post-pruning preferred in practice because of early stopping in pre-pruning

Converting Tree to rules



Converting Tree to rules



IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$

IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$

$\ldots$

Decision Tree 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 by Breiman, Friedman, Olshen, Stone

Many other variations including Random Forests, etc.

Thank you!