# Decision Trees

Adopted from Machine Learning, Mitchell

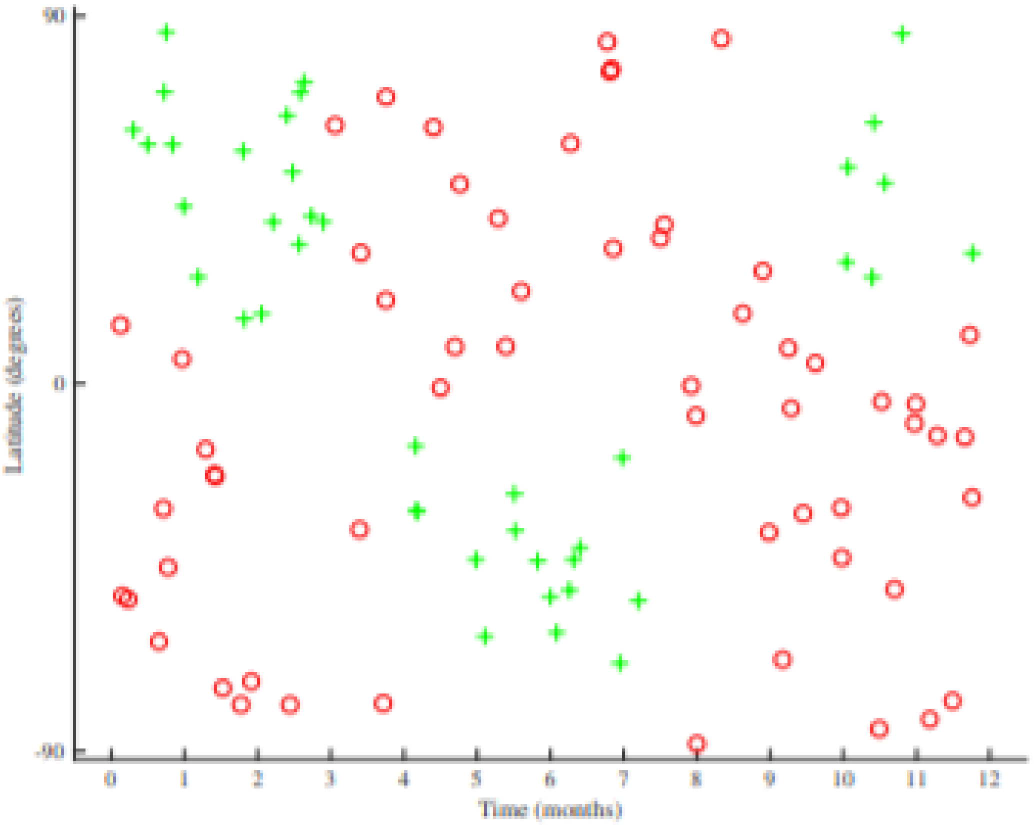
# Decision Trees

- It approximates discrete-valued target function.

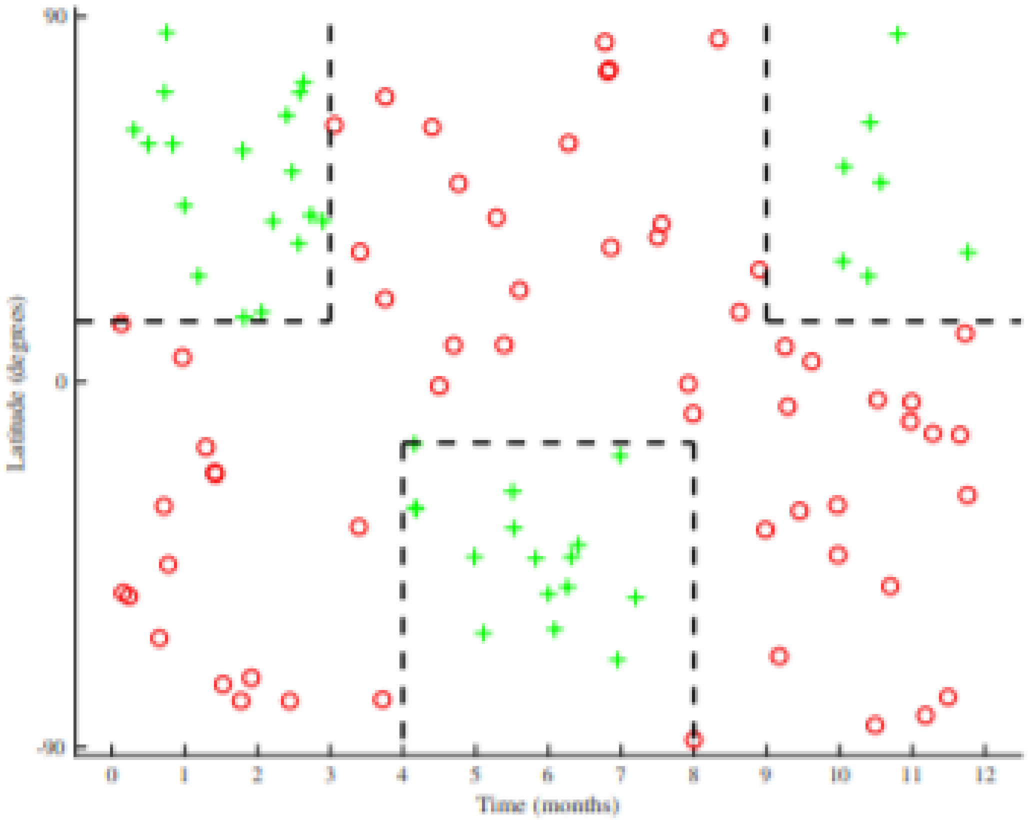
- Learning function is a set of *if-then* rules to improve human readability.

- Highly interpretable

# I want to ski

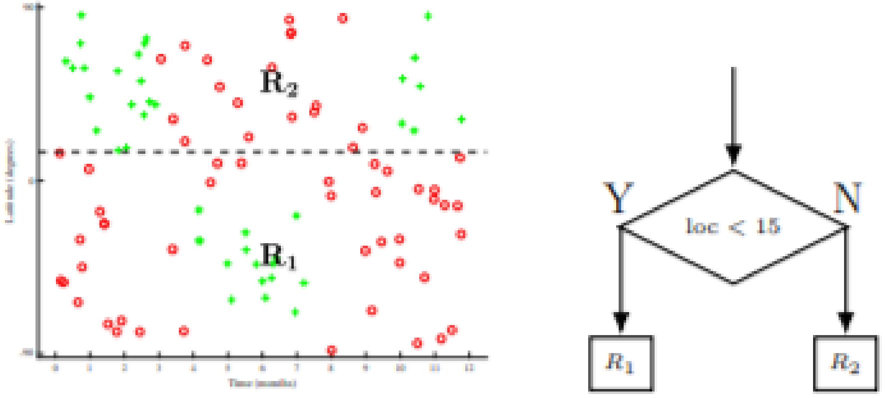


90
0
-90
Latitude (degrees)
0 1 2 3 4 5 6 7 8 9 10 11 12
Time (months)

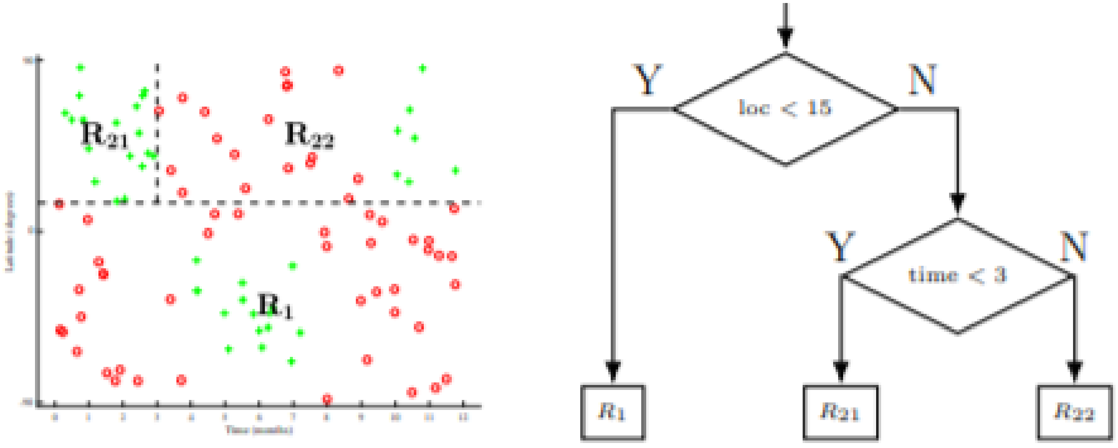



90
0
-90
Latitude (degrees)
0 1 2 3 4 5 6 7 8 9 10 11 12
Time (months)

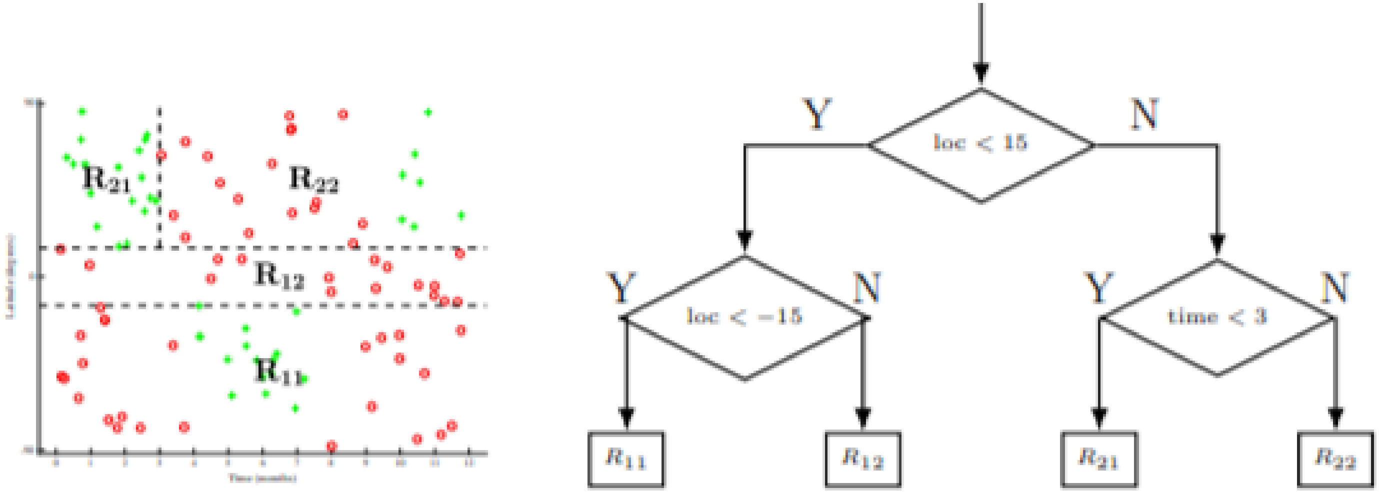
# I want to ski




(a)

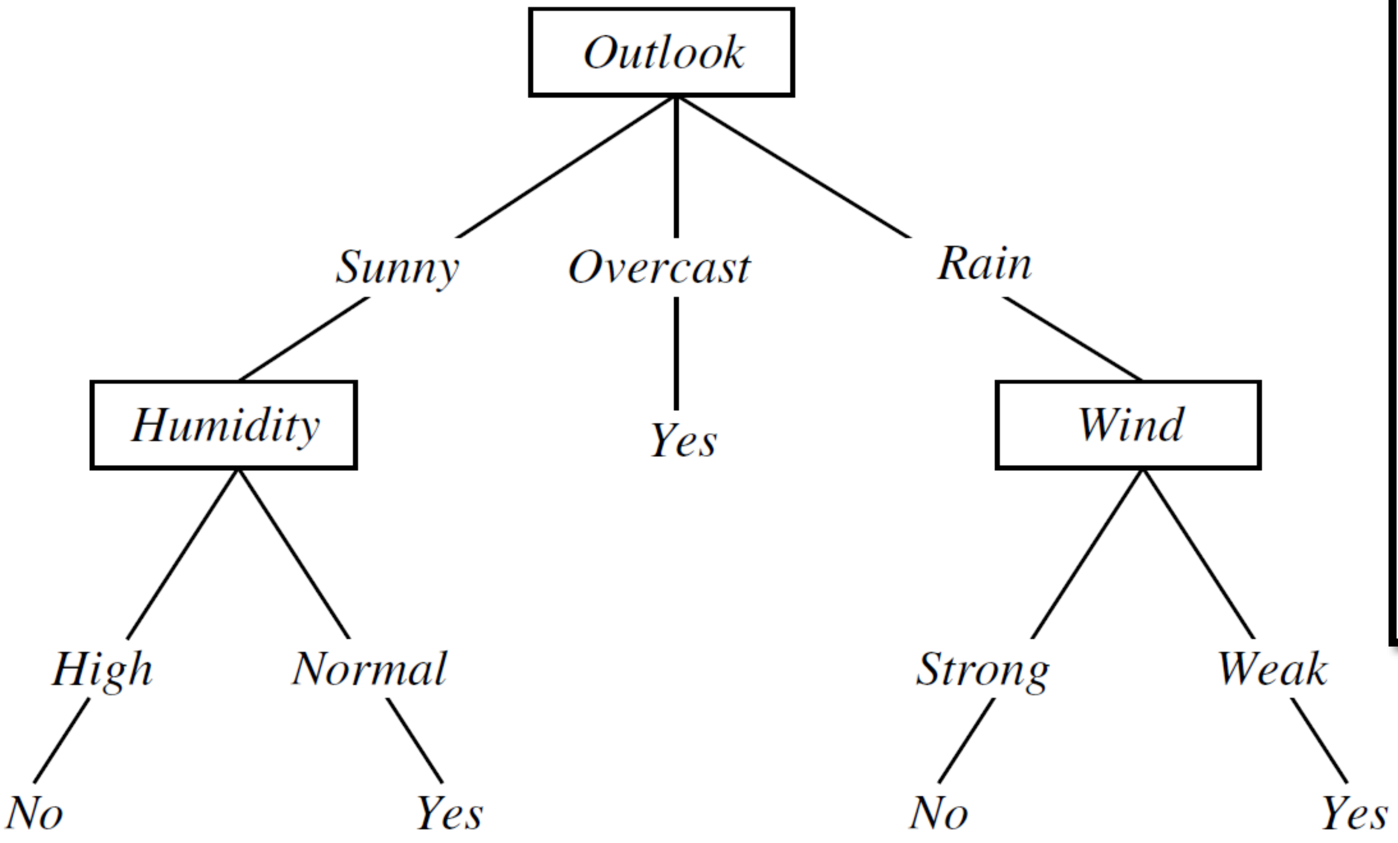



(b)




(c)

# Decision tree for *PlayTennis*




Decision tree representation:

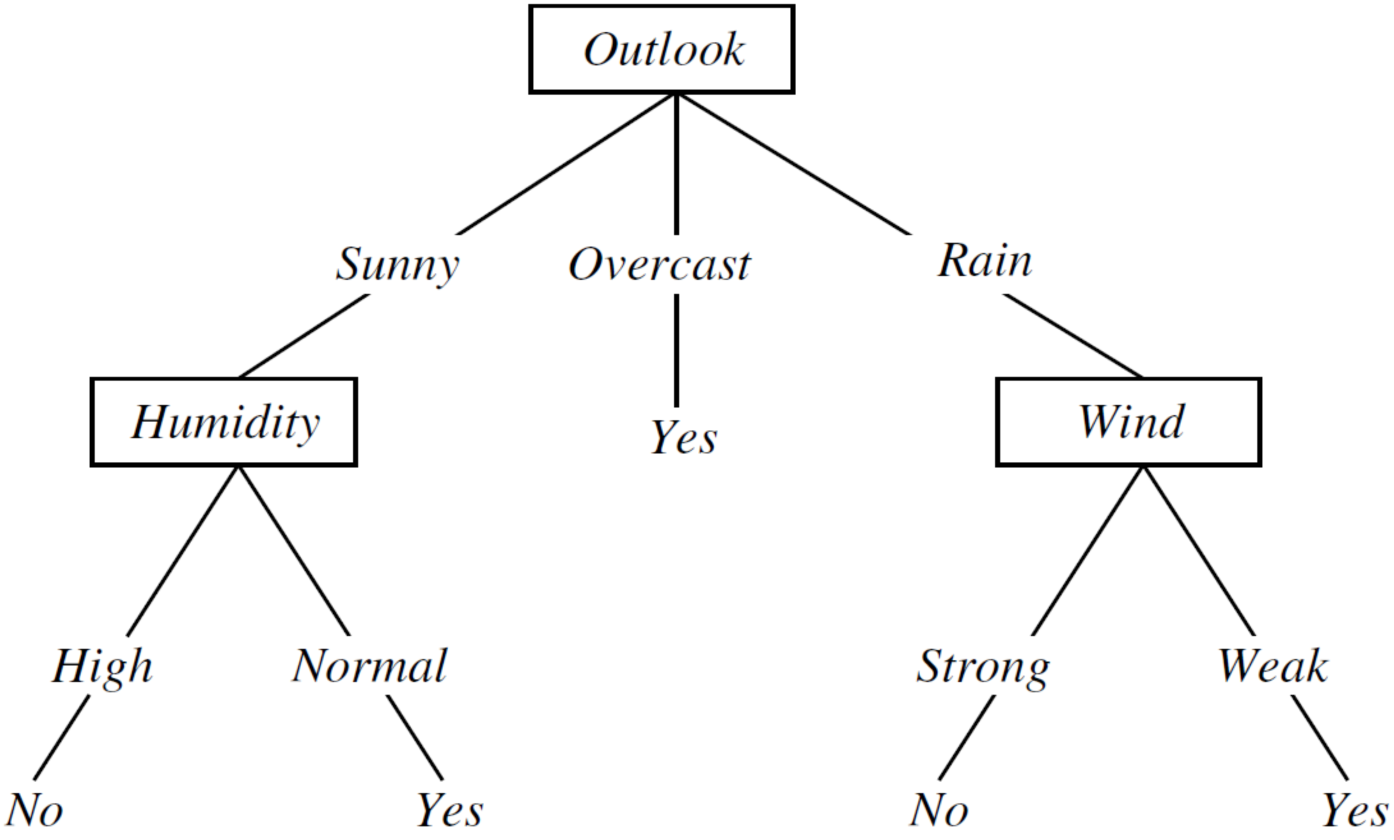
- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent:

- $\wedge, \vee$, XOR
- $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
- $M$ of $N$

$\langle Outlook = Sunny,\ Temperature = Hot,\ Humidity = High,\ Wind = Strong \rangle$

# Decision Tree




A disjunction of conjunctions of constraints on attribute values of instances.

$$(Outlook = Sunny \wedge Humidity = Normal)$$
$$\vee \quad (Outlook = Overcast)$$
$$\vee \quad (Outlook = Rain \wedge Wind = Weak)$$

# When to use Decision Tress

1. Instances are represented by attribute-value pairs.
2. The target function has discrete output values.
3. Disjunctive descriptions may be required.
4. The training data may contain error.
5. The training data may contain missing attribute values.

Examples:

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

# Types of Decision Trees

- **ID3:** Categorical feature that will yield the **largest information gain** for categorical targets

- **C4.5:** Successor to ID3 and removes the restriction that features must be categorical by dynamically defining a discrete attribute (based on numerical variables) that partitions the continuous attribute value into a discrete set of intervals.

- **CART (Classification and Regression Trees):** Similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets.

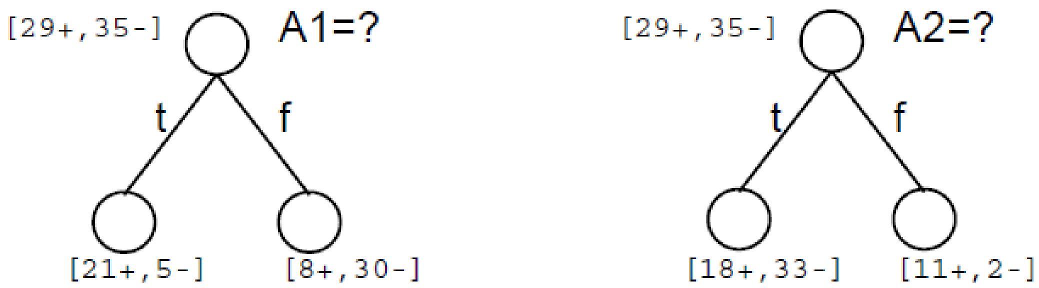
  - https://www.quora.com/What-are-the-differences-between-ID3-C4-5-and-CART (ID: Iterative Dichotomiser)
  - https://medium.com/datadriveninvestor/tree-algorithms-id3-c4-5-c5-0-and-cart-413387342164

# Inductive Learning of Decision Tree

Main loop:

1. $A \leftarrow$ the "best" decision attribute for next *node*
2. Assign $A$ as decision attribute for *node*
3. For each value of $A$, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which attribute is best?




- Greedy
- Top-down
- Recursive partitioning

# Loss Function

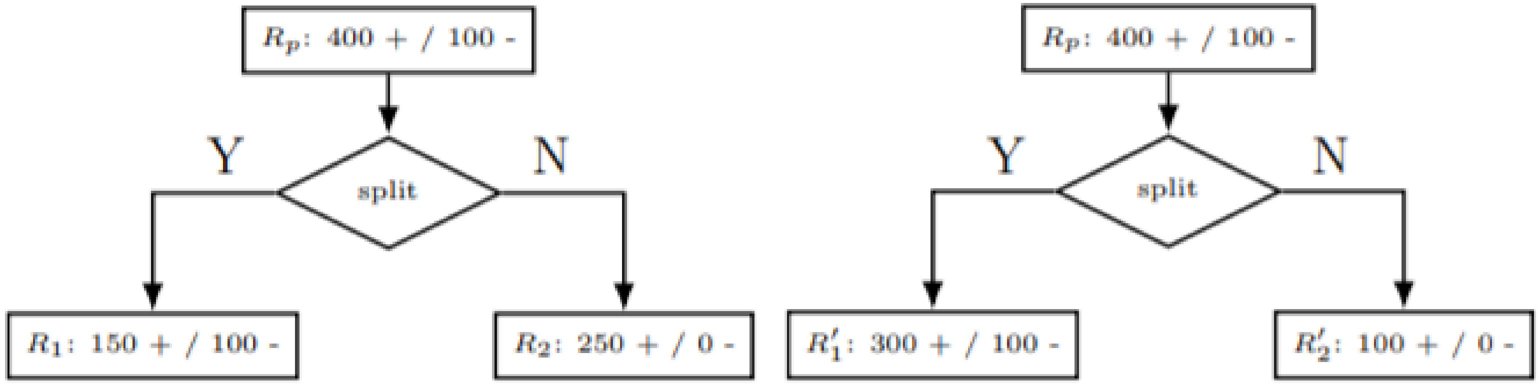
- We calculate the loss of the parent $L(R_p)$ as well as the cardinality-weighted loss of the children

- Select an attribute greedily that maximizes the decrease in loss

$$L(R_p) - \frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|}$$

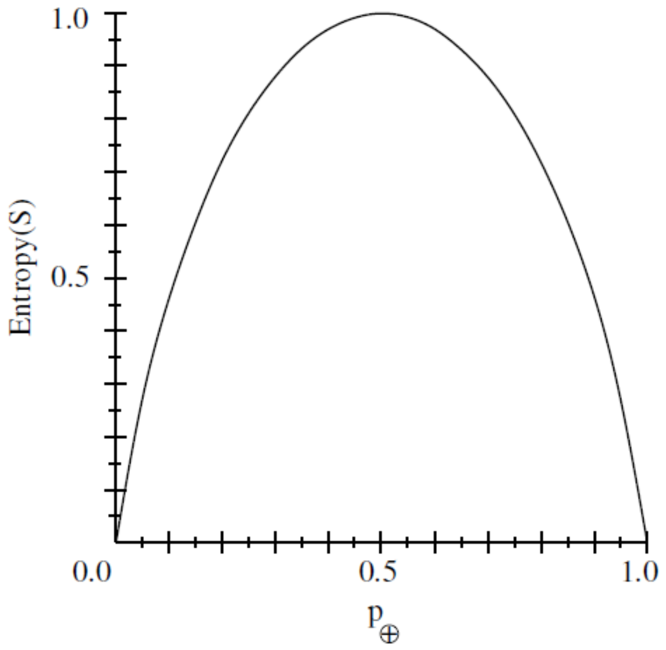
- Misclassification loss: $L_{misclass}(R) = 1 - \max_c(\hat{p}_c)$

# Loss Function: Misclassification loss




$$L(R_p) = \frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|} = \frac{|R_1'|L(R_1') + |R_2'|L(R_2')}{|R_1' + |R_2'|} = 100$$

# (Shannon's) Entropy




Entropy measures homogeneity of examples.

- $S$ is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in $S$
- $p_{\ominus}$ is the proportion of negative examples in $S$
- Entropy measures the impurity of $S$

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

# (Shannon's) Entropy

$Entropy(S)$ = expected number of bits needed to encode class ($\oplus$ or $\ominus$) of randomly drawn member of $S$ (under the optimal, shortest-length code)

Why?

Information theory: optimal length code assigns $-\log_2 p$ bits to message having probability $p$.

So, expected number of bits to encode $\oplus$ or $\ominus$ of random member of $S$:

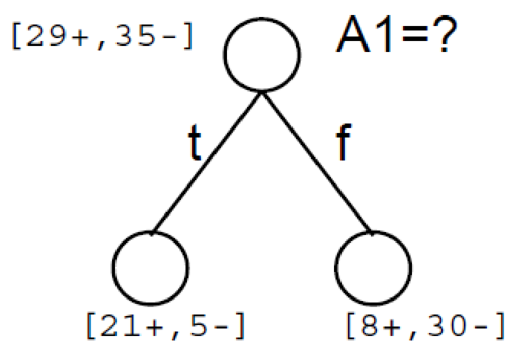
$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

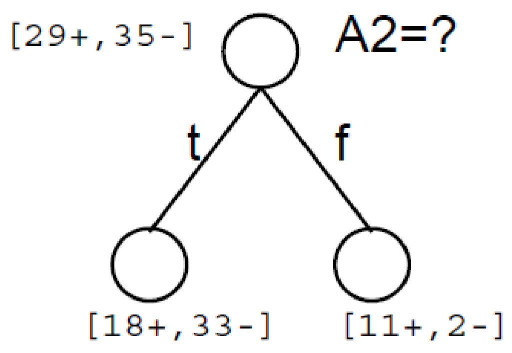
$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

# Information Gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on $A$

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



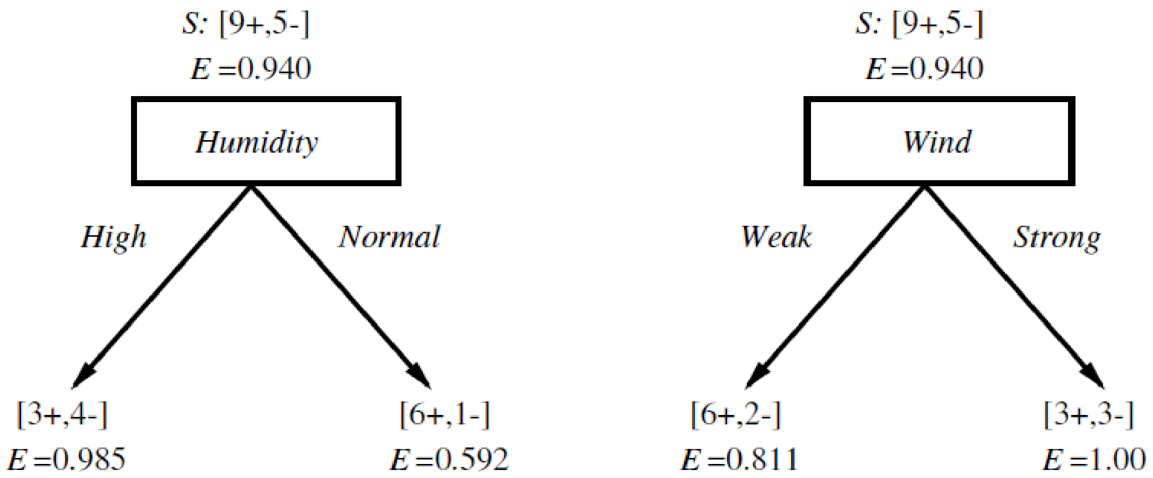




**Gain(S,A):** Number of bits saved when encoding the target value of an arbitrary member of S, by knowing the value of attribute A.

# Training examples

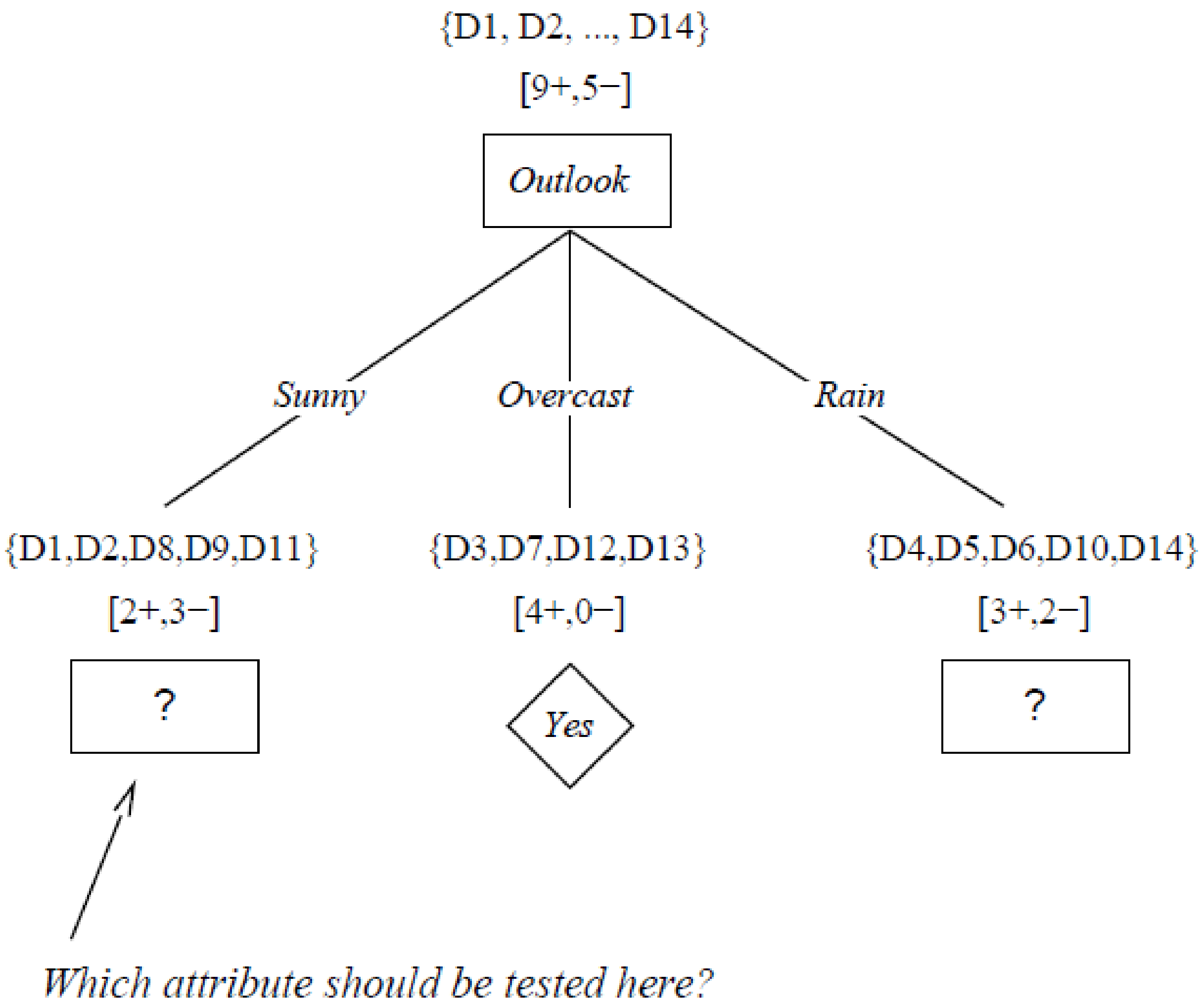
| Day | Outlook | Temperature | Humidity | Wind | PlayTennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| 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 |

**Which attribute is the best classifier?**




*Gain (S, Humidity )*
= .940 - (7/14).985 - (7/14).592
= .151

*Gain (S, Wind)*
= .940 - (8/14).811 - (6/14)1.0
= .048




| Day | Outlook | Temperature | Humidity | Wind | PlayTenni |
|---|---|---|---|---|---|
| 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 |

$S_{sunny} = \{D1,D2,D8,D9,D11\}$

$Gain\ (S_{sunny}, Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$

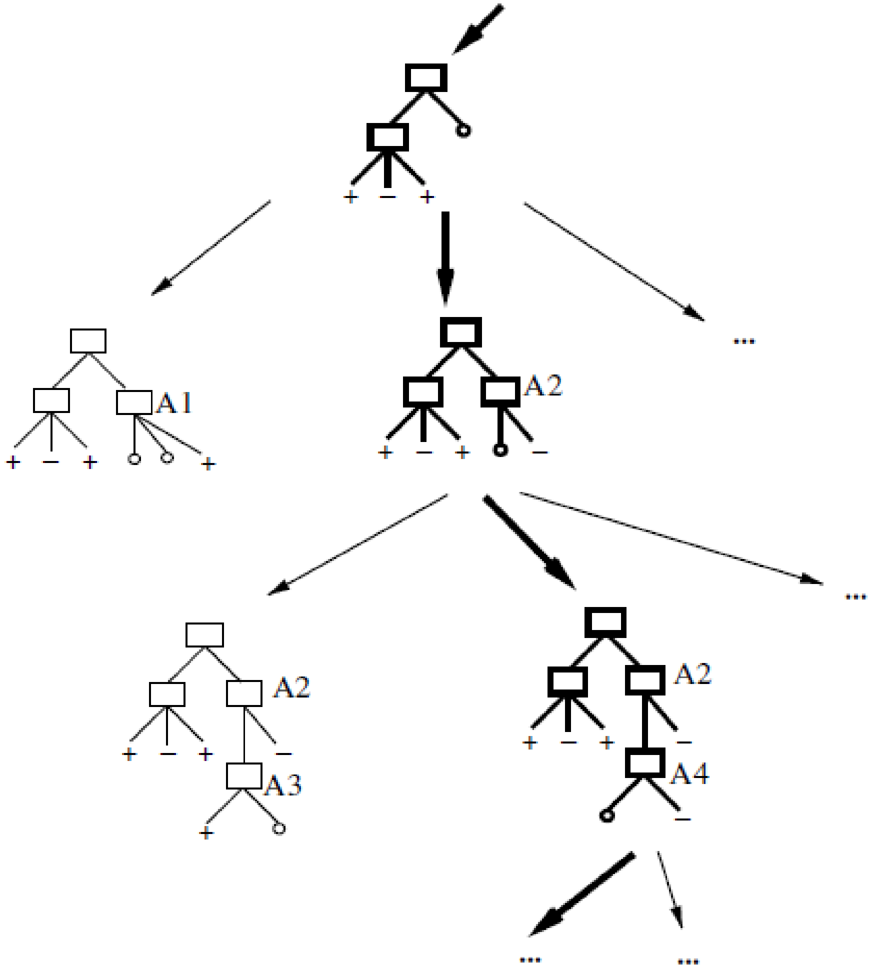
$Gain\ (S_{sunny}, Temperature) = .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .570$

$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

# Hypothesis space search by ID3

- Hypothesis space is complete!
  - Target function surely in there...
- Outputs a single hypothesis (which one?)
  - Can't play 20 questions...
- No back tracking
  - Local minima...
- Statisically-based search choices
  - Robust to noisy data...
- Inductive bias: approx "prefer shortest tree"

# Hypothesis space search in ID3




- ID3: Iterative Dichotomiser 3 !!

# Inductive Bias

Set of assumptions that, together with the training data, justify the classifications assigned by the learner to future instances.

# Inductive Bias in Decision Trees

- Selects in favor of short trees over longer ones
- Select trees that place the attributes with highest information gain closest to the root

**Approximate inductive bias of ID3**: Shorter trees are preferred over longer trees

- Think about a BFS-ID3 algorithm
- ID3 is just a greedy version of BFS-ID3

Shorter trees are preferred over longer trees. that place the attributes with highest information gain closest to the root are preferred over those that do not.

# Inductive bias: ID3 vs CANDIDATE-ELIMINATION

**ID3 (preference bias)**

- **Complete** hypothesis space
- Searches ***incompletely*** *from simple to complex hypotheses*
- Bias is a consequence of the ordering of hypotheses by its search strategy, not due to hypothesis space

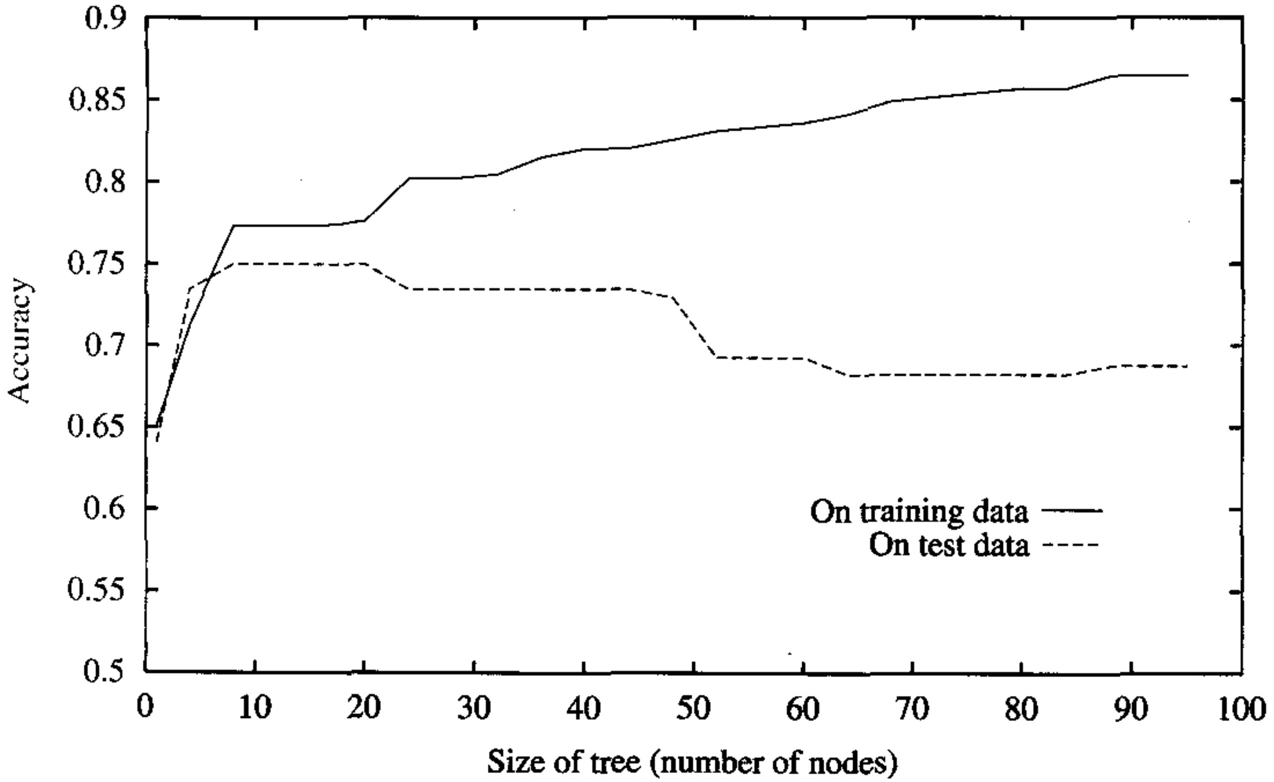
**CE (restriction bias)**

- ***Incomplete*** hypothesis space
- Searches ***completely***
- Bias is a consequence of the ordering of the expressive power of its hypothesis representation, not due to search strategy

Which one is more desirable?

# Overfitting in Decision Trees

***Definition***: Given a hypothesis space $H$, a hypothesis $h \in H$ is said to **overfit** the training data if there exists some alternative hypothesis $h' \in H$, such that $h$ has smaller error than $h'$ over the training examples, but $h'$ has a smaller error than $h$ over the entire distribution of instances.

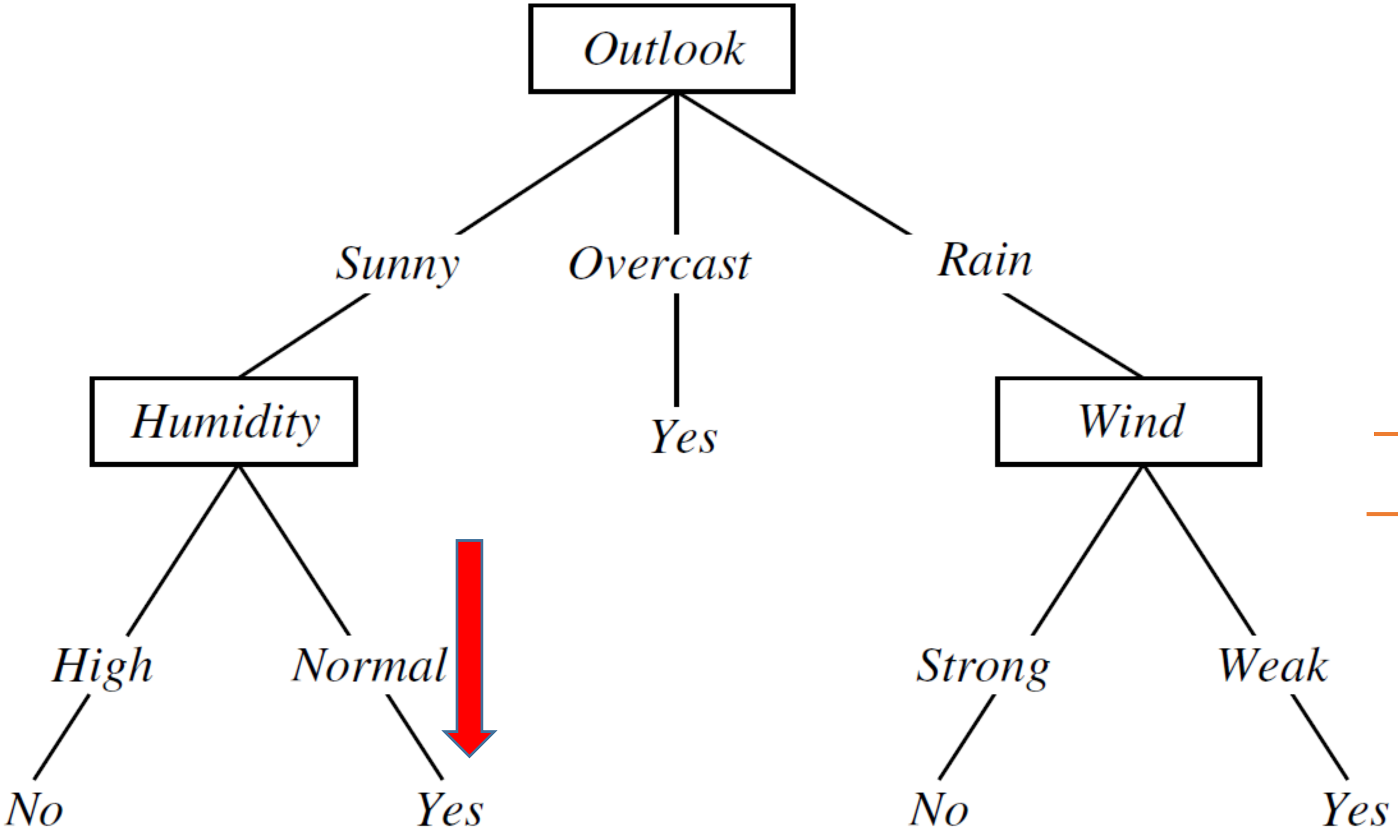



- Noise in the training set
- No. of training samples is too small

# Overfitting in Decision Trees

The following positive example, incorrectly marked as negative

$$\langle Outlook = Sunny, Temperature = Hot, Humidity = Normal,$$
$$Wind = Strong, PlayTennis = No\rangle$$




| Day | Outlook | Temperature | Humidity | Wind | PlayTenni |
|---|---|---|---|---|---|
| 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 |

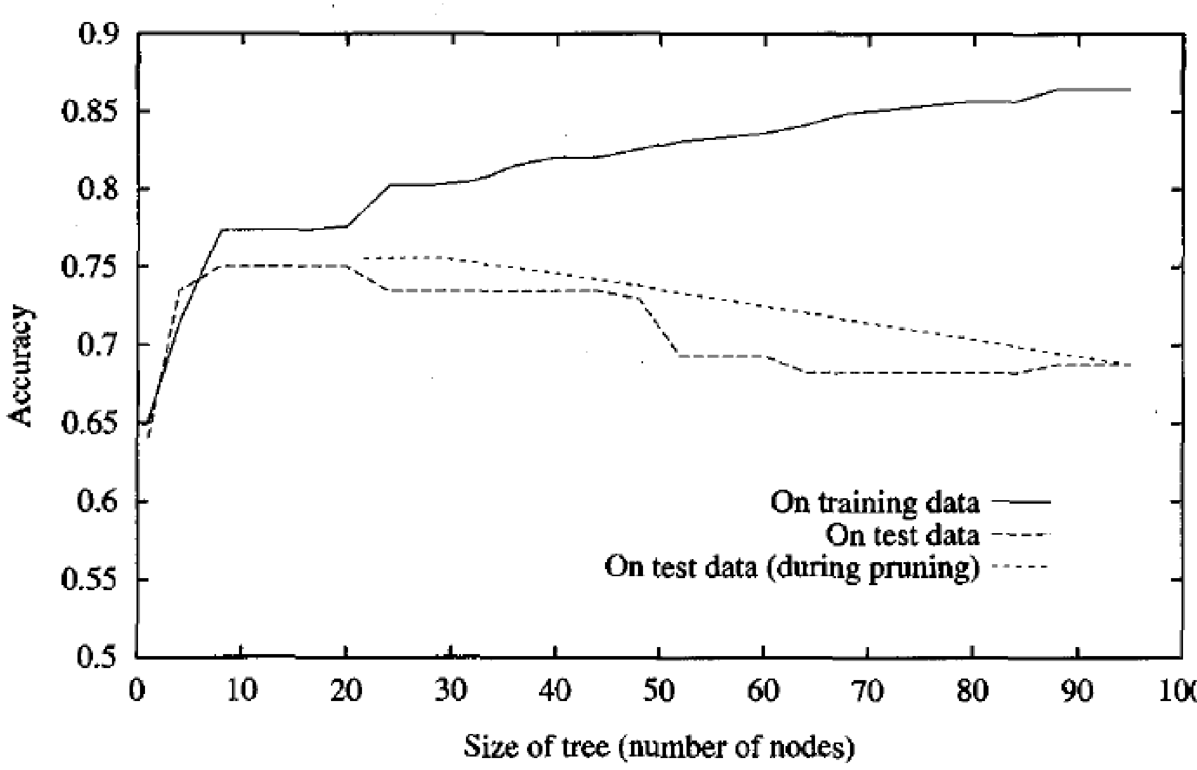
# Avoiding Overfitting

- Stop growing the tree earlier
- **Allow the tree to overfit the data and then post-prune the tree**

- **Criterion to determine the correct final tree size**
  - Separate set of examples (validation set) to determine the utility of post-pruning
  - Apply statistical test to estimate whether the improvement after incorporating a new training instance is statistically significant (chi-square test)
  - Use minimum description length principle

# Reduced Error Pruning

- Use validation set
- Each decision node is candidate for pruning
  - Pruning => removing the subtree rooted at the node, making it a leaf node, assigning it the most common class
  - Pruned if the resultant tree performs no worse than the original
  - Nodes are pruned iteratively
    - Nodes whose removal most increases the accuracy

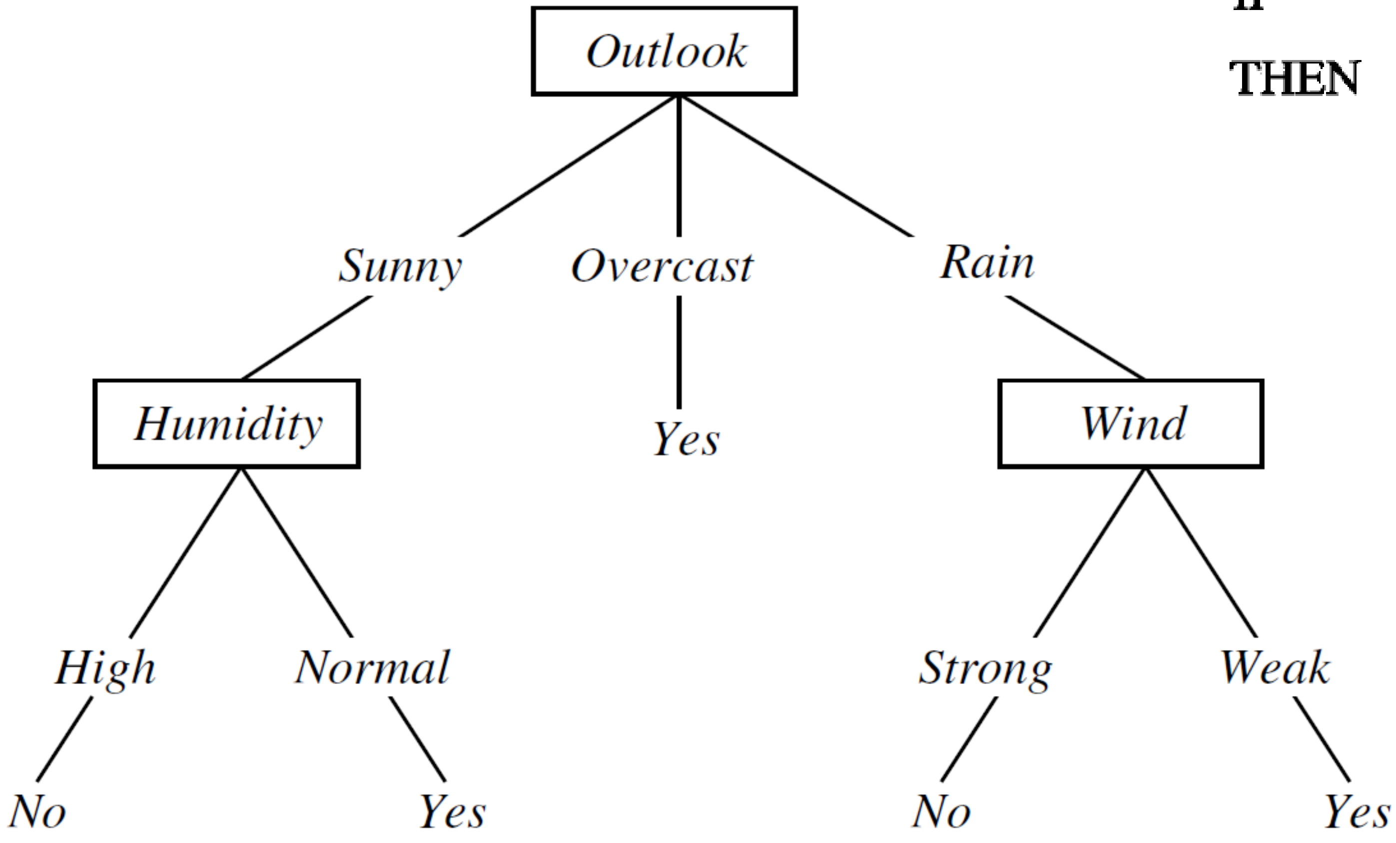



**Drawback**: when data is limited, withholding part of it for validation further reduces the training size

# Rule Post-pruning

- Allow the tree to grow based on the training set and allow to overfit
- Convert one branch (root-> leaf) to an equivalent rule
- Prune (generalize) each rule by removing any preconditions that result in improving the accuracy.
- Sort the pruned rules by their estimated accuracy

# Rule Post-pruning




IF $(Outlook = Sunny) \wedge (Humidity = High)$

THEN $PlayTennis = No$

# Why convert decision tree to rules?

- Allow distinguishing among the different contexts in which the decision node is used
- Remove the distinction between attribute tests that occur near the root vs. near the leaves
- Improve readability

# Incorporating continuous-valued attributes

| | | | | | | |
|---|---|---|---|---|---|---|
| *Temperature*: | 40 | 48 | 60 | 72 | 80 | 90 |
| *PlayTennis*: | No | No | Yes | Yes | Yes | No |

- Sort examples based on continuous attribute
- Decide the candidate thresholds based on the change in attribute values

  (48+60)/2=54; (80+90)/2=85

- Calculate information gain for each of the thresholds separately.

# Alternative measure of selecting attributes

- Information gain favors attributes with many values (e.g., date)

- **Gain ratio:** $\dfrac{Gain(S, A)}{SplitInformation(S, A)}$

$$SplitInformation(S, A) \equiv - \sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

# Issues with Gain Ratio

- It tends to select those attributes whose dominator can be zero or very small ($S_i \sim S$) for one of the $S_i$
- **Heuristics:** first calculate the Information Gain of each attribute, then apply Gain Ratio only to consider those attributes with above average Gain

**Alternative**: Choose an attribute that minimizes the distance between the current data partition and the gold data partition.

# Dealing with missing attribute values

- *Blood-test-result* missing
- Assign the value that is most common among the training examples at a particular node
- Or Assign the value that is most common among the training examples at a particular node that have the class label same as that of the given instance
- **Alternative:** Assign a probability to each of the possible values of A (the missing attribute) and calculate the gain

# Handling attributes with different costs

$$\frac{Gain^2(S, A)}{Cost(A)}$$

$$\frac{2^{Gain(S,A)} - 1}{(Cost(A) + 1)^w}$$