Decision Tree

[Slides from Fardina Fathmiul Alam]

Before We Start, Remember:

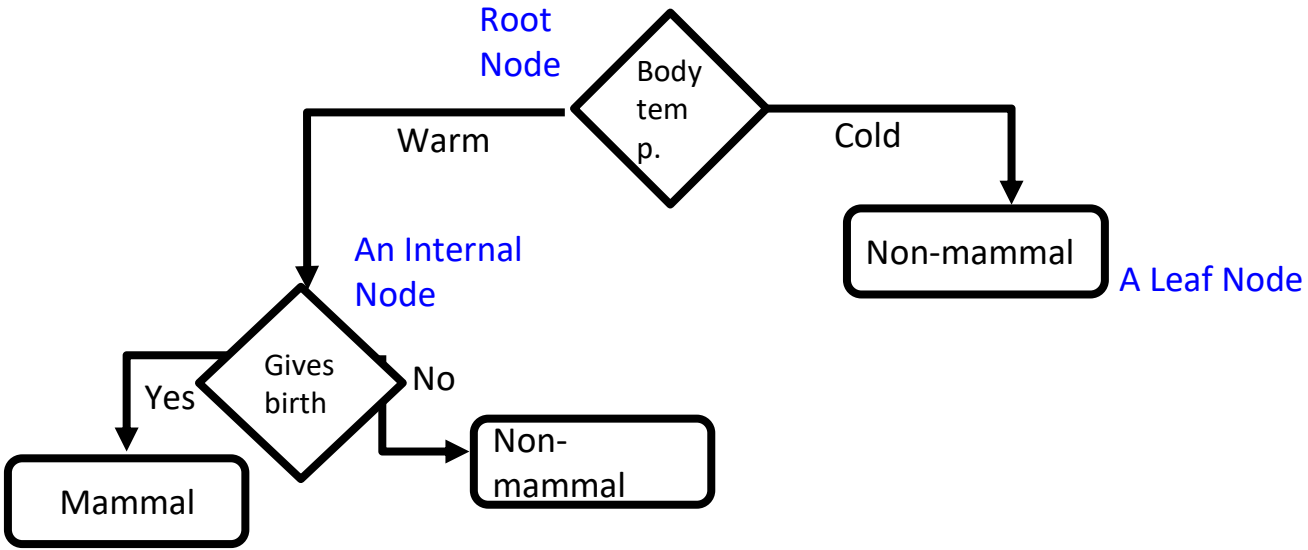
No free lunch theorem

Much of ML is concerned with devising different models and different algorithms to fit them.

*However, there's **no single best model** that works optimally for all ends of problems.*

Decision Trees

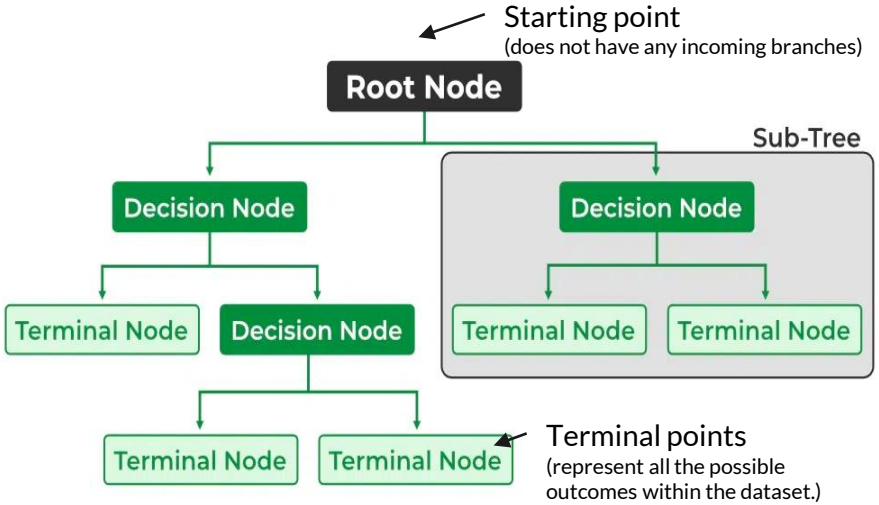
- A Decision Tree (DT) defines a hierarchy of rules to make a prediction



- Root and internal nodes test rules. Leaf nodes make predictions

- Decision Tree (DT) learning is about learning such a tree from labeled data

Terminology: Decision Tree



A binary decision tree (BST) is a decision tree in which each node has two branches.

The parts of a decision tree are:

- **Decision node:** where a specific feature of the data is tested. Based on the value of this feature, the tree decides which branch to follow.
- **Root node:** the first/initial decision node at the top of the tree. It represents the starting point for decision-making.
- **Branch/Sub-Tree:** a connection between decision nodes in the tree. Each branch represents a possible outcome or decision based on the feature being tested at the decision node.
- **Layer:** all of the decision nodes that are the same distance from the root node.
- **Leaf/Terminal node:** a node that does not test a feature and so is where a decision is made.
- **Depth:** the number of layers in a decision tree.

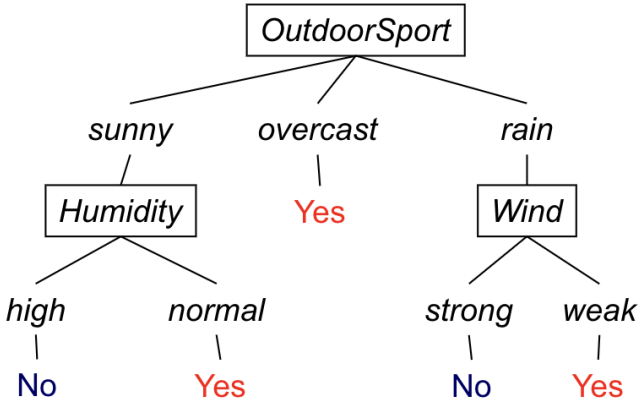
Decision Trees

Concept Learning: Involves a conjunction of attributes.

- Example: (Sunny AND Hot AND Humid AND Windy).

Decision Trees: Use a disjunction of conjunctions of attributes.

- Example: (Sunny AND Normal) OR (Overcast) OR (Rain AND Weak).

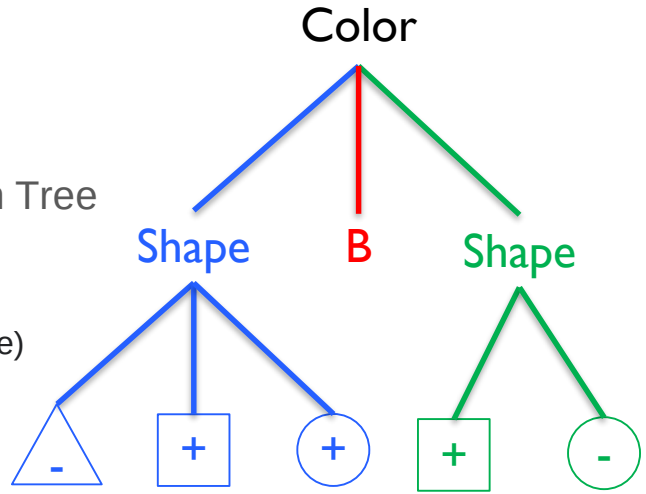


Advantages of Decision Trees:

- More Powerful Representation: Can represent more complex concepts.
- Larger Hypothesis Space (H): Can explore a broader range of possibilities.
- Tree Representation: Concepts can be visually represented as a tree.
- Human-Like Decision Making: Mirrors common forms of human decision-making.

Expressivity of Decision Trees

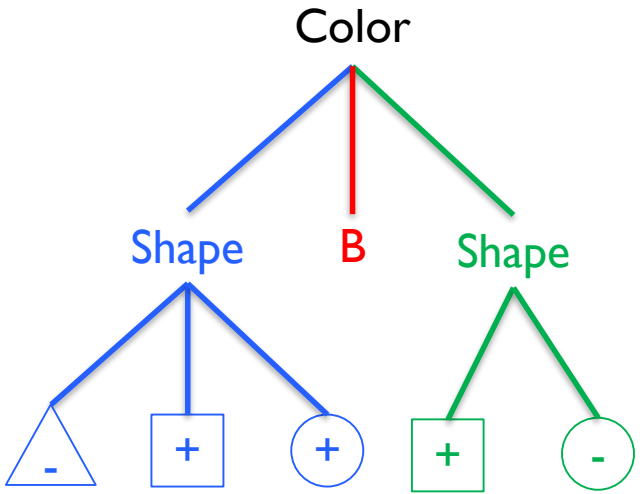
- As Boolean functions they can represent any Boolean function.
- Can be rewritten as rules in Disjunctive Normal Form (DNF)
  - Green ∧ Square → positive
  - Blue ∧ Circle → positive
  - Blue ∧ Square → positive
- The disjunction of these rules is equivalent to the Decision Tree
- What did we show? What is the hypothesis space here?
  - 2 dimensions: color and shape
  - 3 values each: color(red, blue, green), shape(triangle, square, circle)
  - |X| = 9: (red, triangle), (red, circle), (blue, square) …
  - |Y| = 2: + and -
  - $|H| = 2^9$
- And, all these functions can be represented as decision trees.



Output of Decision Trees

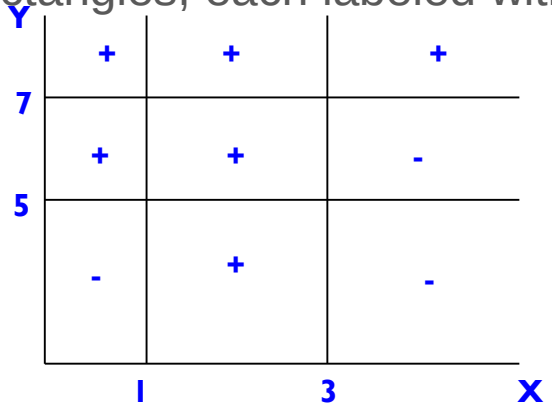
- Output is a **discrete category**. Real valued outputs are possible (regression trees)

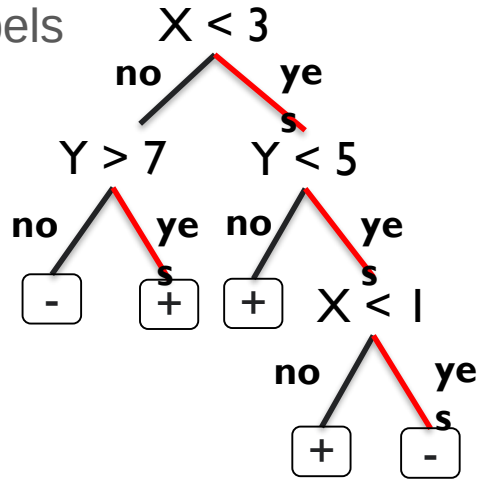
- There are efficient algorithms for processing large amounts of data (but not too many features)



Decision Boundaries

- Usually, instances are represented as attribute-value pairs (color=blue, shape = square, +)
- Numerical values can be used either by discretizing or by using thresholds for splitting nodes
- In this case, the tree divides the features space into axis-parallel rectangles, each labeled with one of the labels





Learning Decision Trees with Supervision

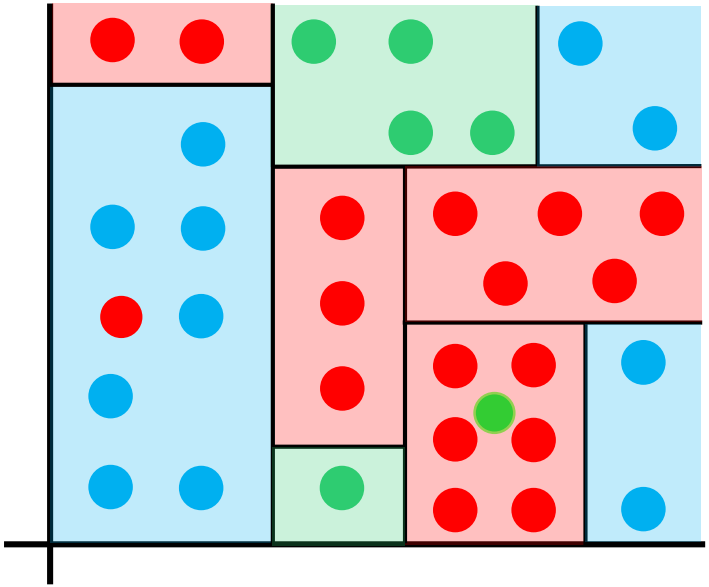
- The basic idea is very simple

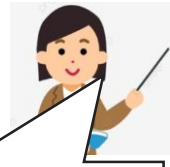
- Recursively partition the training data into homogeneous regions
  - Within each group, fit a simple supervised learner (e.g., predict the majority label)

Learning Decision Trees with Supervision

What do you mean by “homogeneous” regions?

A homogeneous region will have all (or a majority of) training inputs with the same/similar outputs

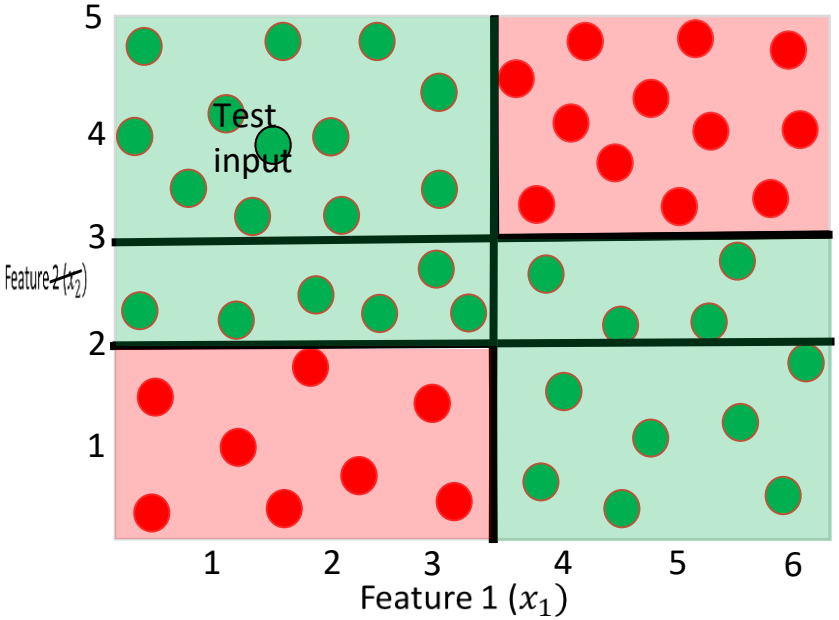


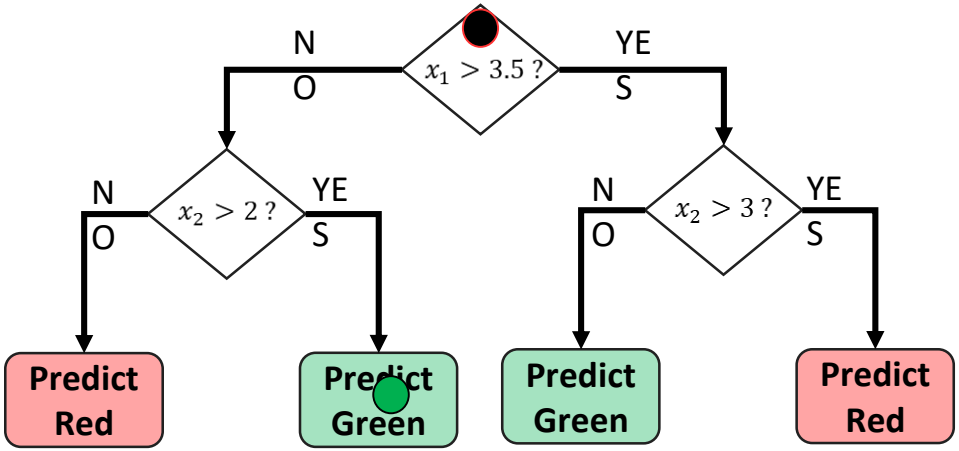


Even though the rule within each group is simple, we are able to learn a fairly sophisticated model overall.

In this example, each rule is a simple horizontal/vertical classifier but the overall decision boundary is rather sophisticated)

Decision Trees for Classification

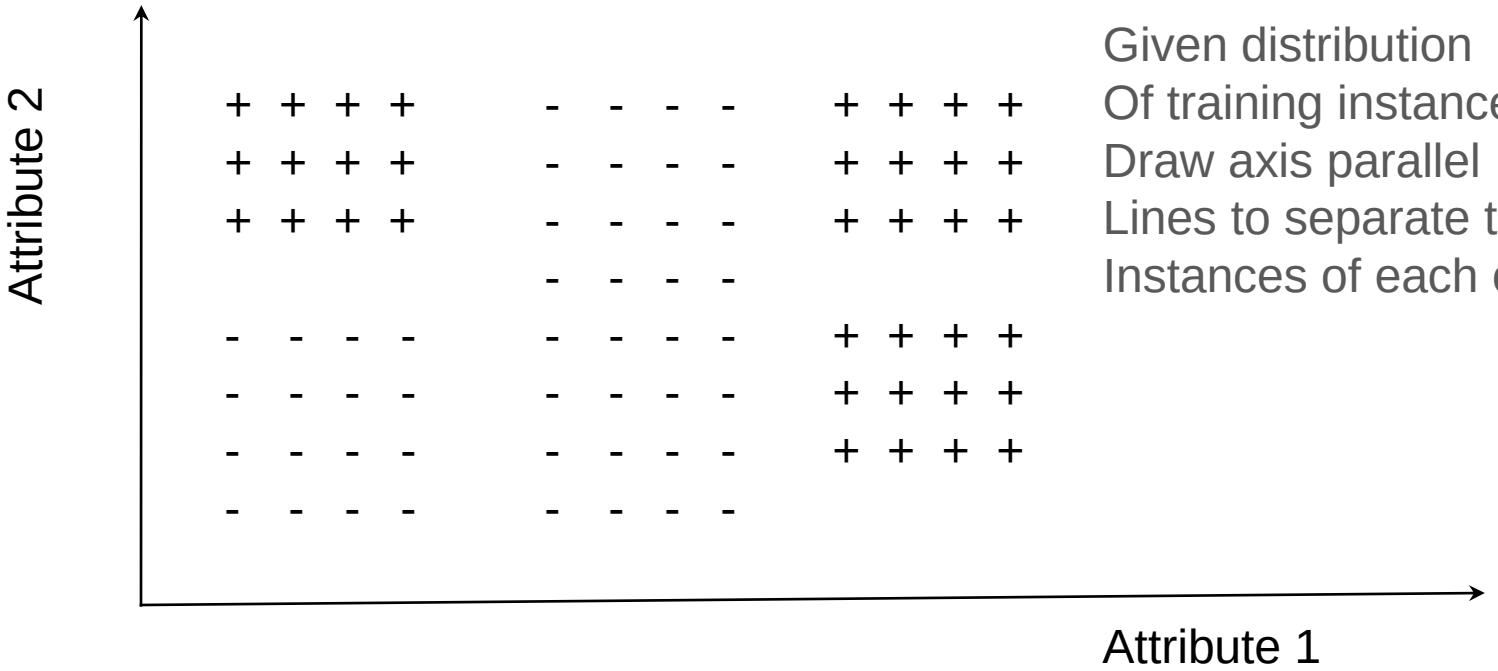




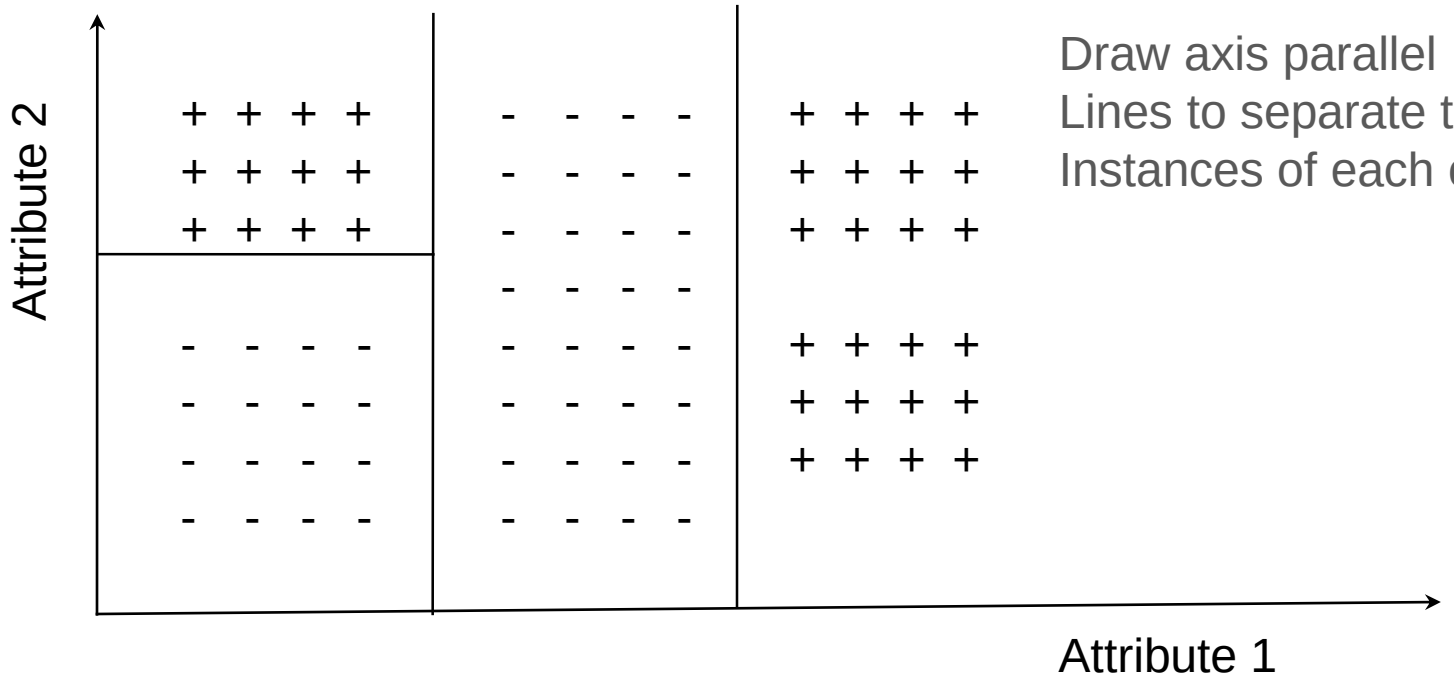
DT is very efficient at test time: DT predicts the label by doing just 2 feature-value comparisons! Way faster!

Remember: Root node contains all training inputs
Each leaf node receives a subset of training inputs

Decision Trees

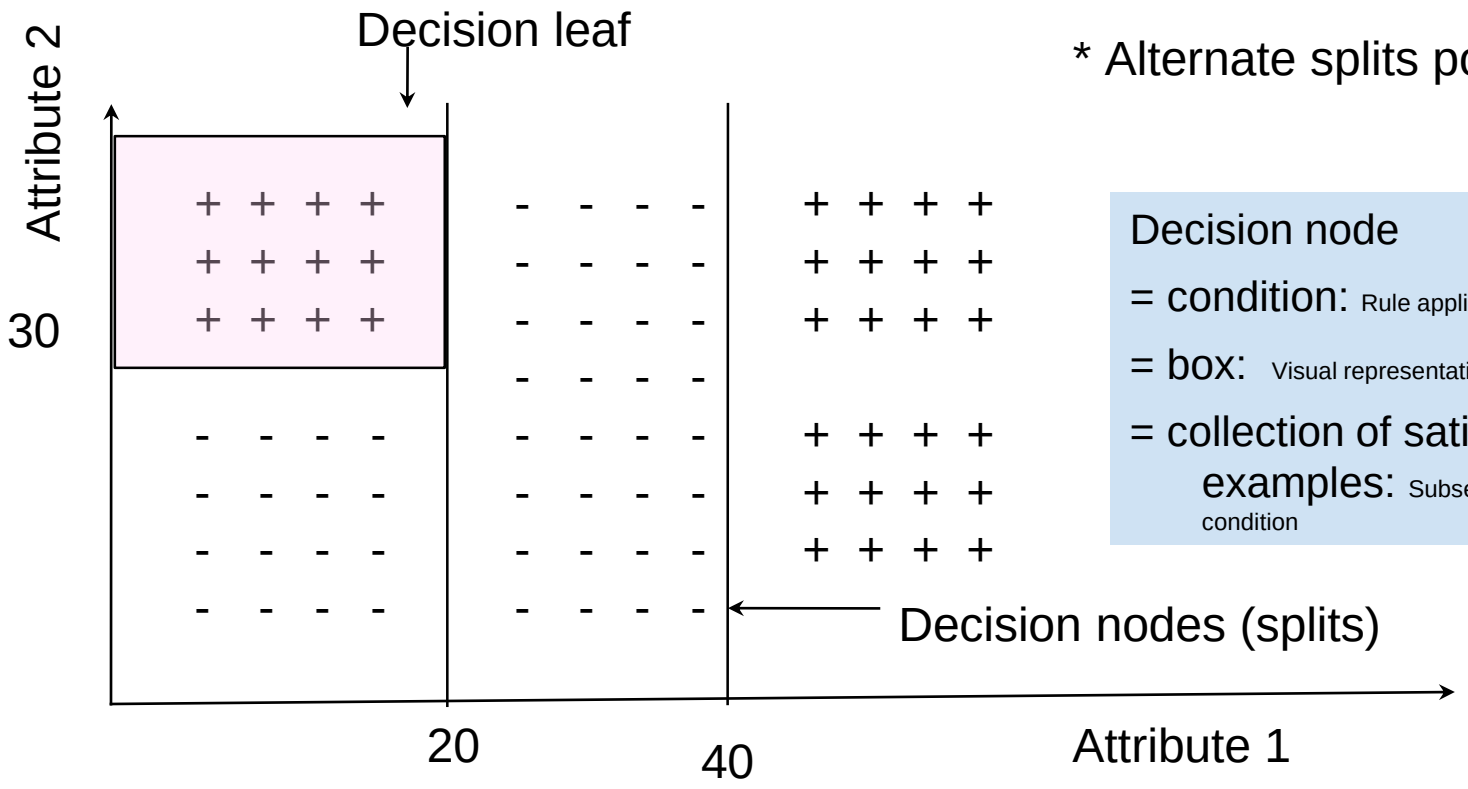


Decision Tree Structure



Draw axis parallel
Lines to separate the
Instances of each class

Decision Tree Structure



* Alternate splits possible

Decision node

= condition: Rule applied at a decision node.

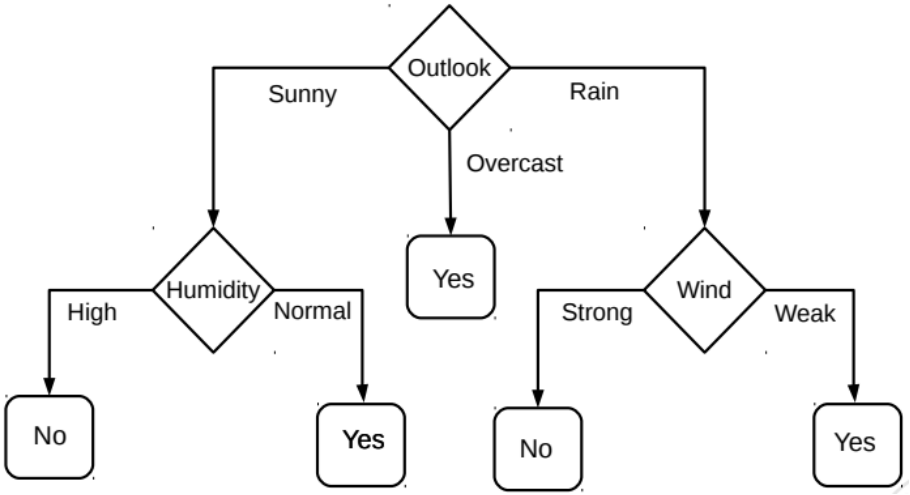
= box: Visual representation in a tree

= collection of satisfying examples: Subset of data meeting the condition

Decision Trees for Classification: Another Example

- Deciding whether to play or not to play Tennis on a Saturday
- Each input (Saturday) has 4 categorical features: Outlook, Temp., Humidity, Wind
- A binary classification problem (play vs no-play)
- Below Left: Training data, Below Right: A decision tree constructed using this data

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



Example credit: Tom Mitchell

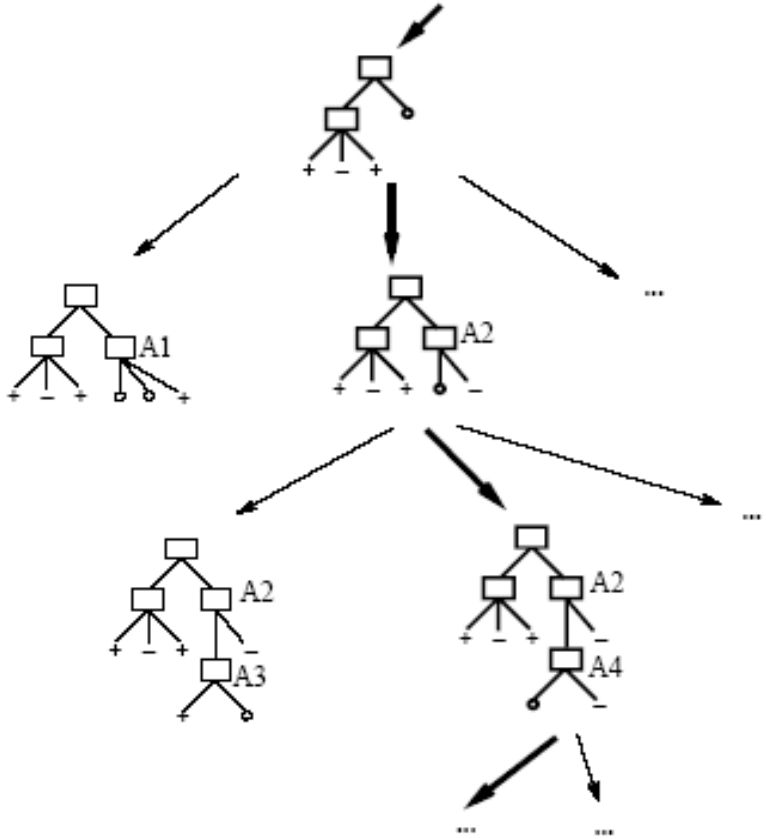
Decision Tree Induction

Many Algorithms:

- ID3 (Iterative Dichotomiser 3) → **We will focus only on this one**
- C4.5
- Hunt’s Algorithm
- CART
- SLIQ,SPRINT

Hypothesis Space

- Clearly, given data, there are many ways to represent it as a decision tree.
- Learning a good representation from data is the challenge.



Ockham's Razor

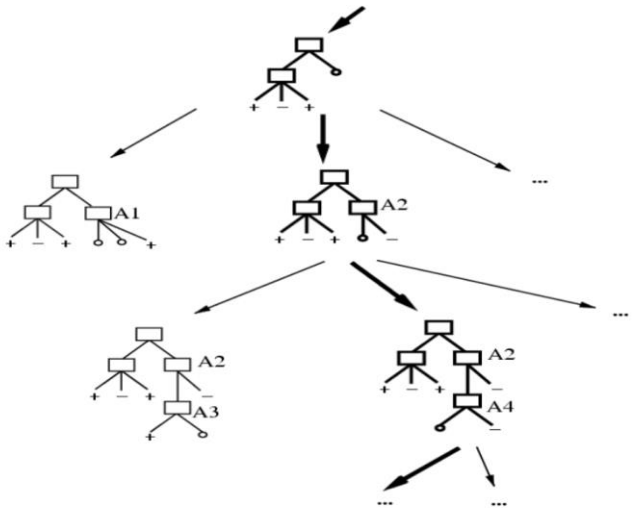
The goal is to have the resulting decision tree as small as possible (Occam's Razor)

- But, finding the minimal decision tree consistent with the data is NP-hard;
- The recursive algorithm is a greedy heuristic search for a simple tree, but cannot guarantee optimality.
- The main decision in the algorithm is the selection of the next attribute to condition on.

- Therefore, the smallest decision tree that correctly classifies all of the training examples is best
  - Finding the provably smallest decision tree is NP-hard
  - ...So instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small

Ockham's Razor

Which Tree Should We Output?



- ID3 performs heuristic search through space of decision trees
- It stops at smallest acceptable tree. Why?

Occam's razor: prefer the simplest hypothesis that fits the data

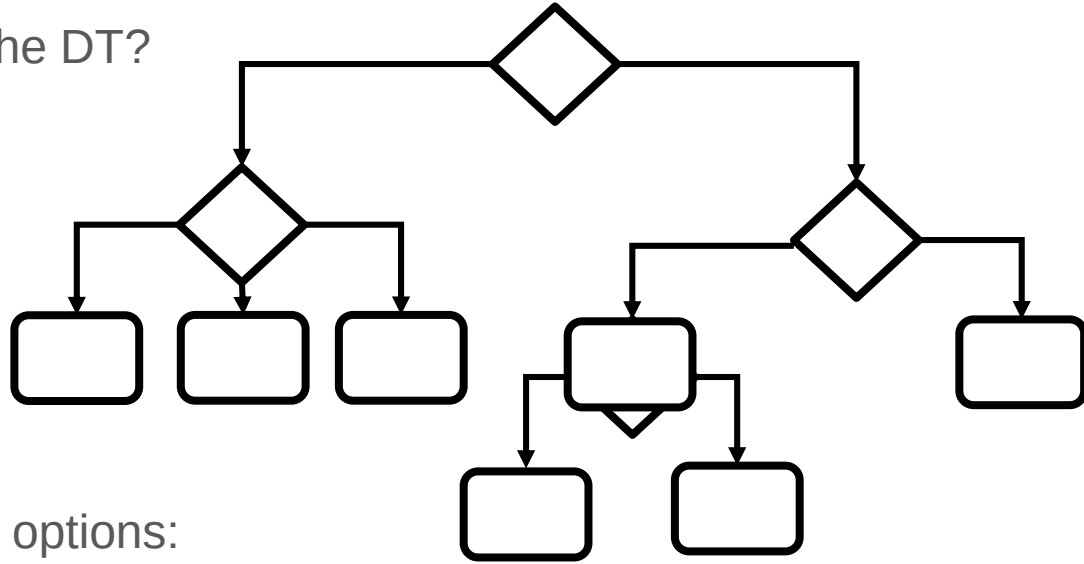
Decision Tree Construction: Top Down Approach

ID3 is a specific algorithm for building decision trees using a top-down induction approach

- Start with empty tree
- Main loop:
  1. Split the “best” decision attribute (*A)* 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, STOP,
     Else iterate over new leaf nodes
- Grow tree just deep enough for perfect classification
  - If possible (or can approximate at chosen depth)
- Which attribute is best?

Decision Trees: Some Considerations

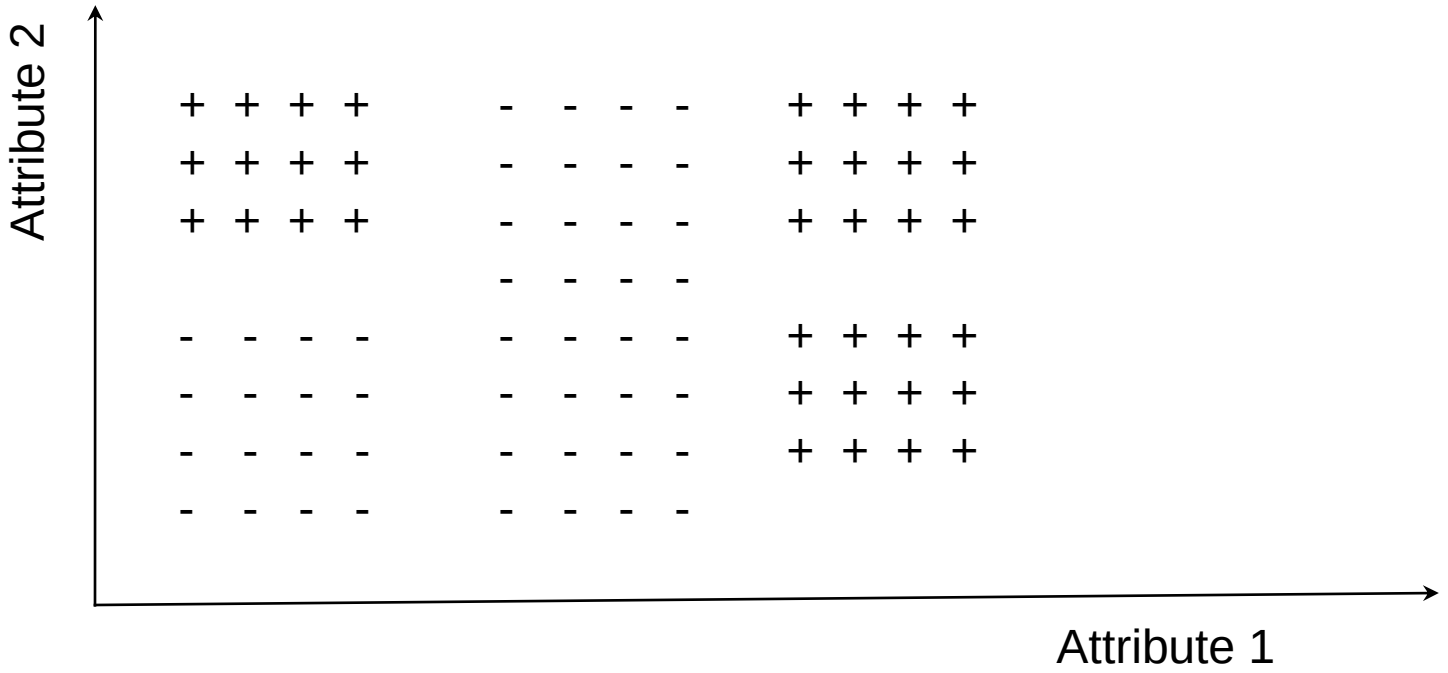
- What should be the size/shape of the DT?
  - Number of internal and leaf nodes
  - Branching factor of internal nodes
  - Depth of the tree



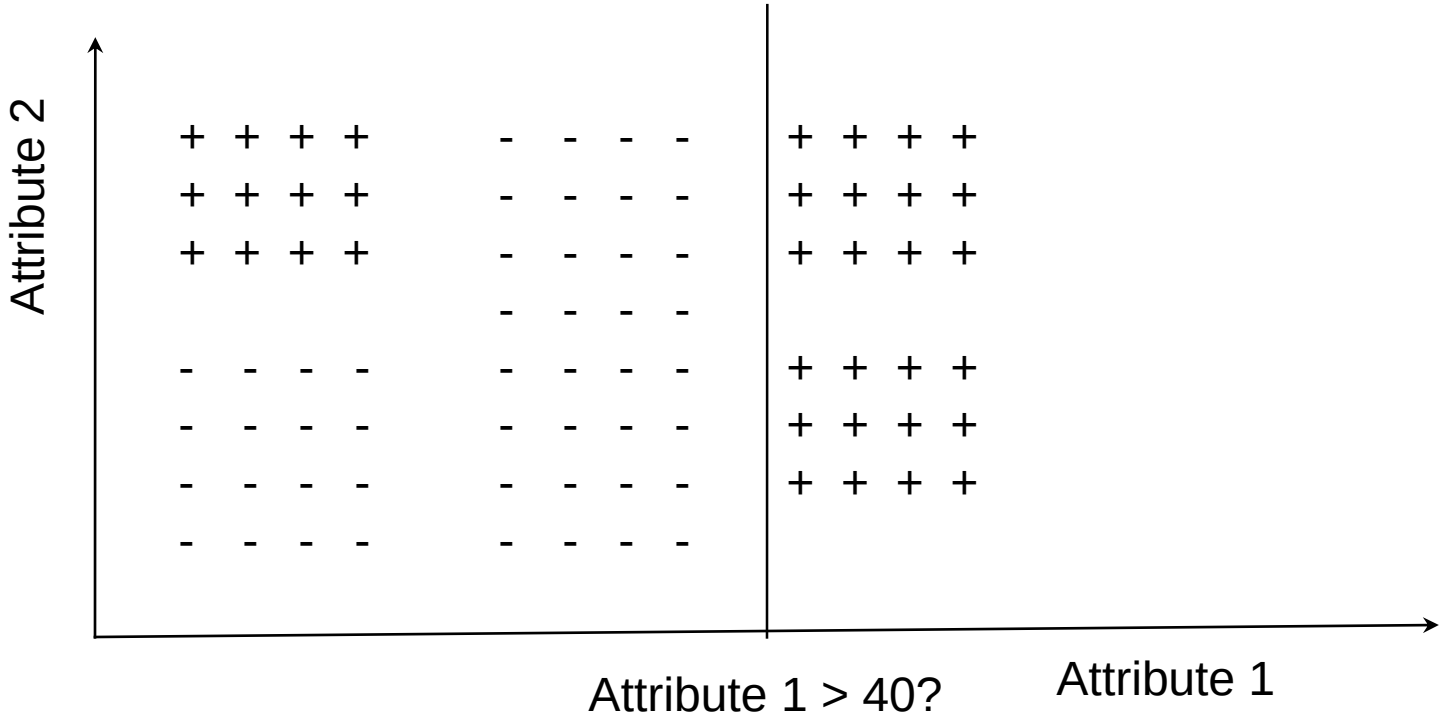
- Split criterion at internal nodes
  - Use another classifier?
  - Or maybe by doing a simpler test?

- What to do at the leaf node? Some options:
  - Make a constant prediction for each test input reaching there (MOST COMMON APPROACH AS IT IS FAST)
  - Use a nearest neighbor based prediction using training inputs at that leaf node
  - Train and predict using some other sophisticated supervised learner on that node

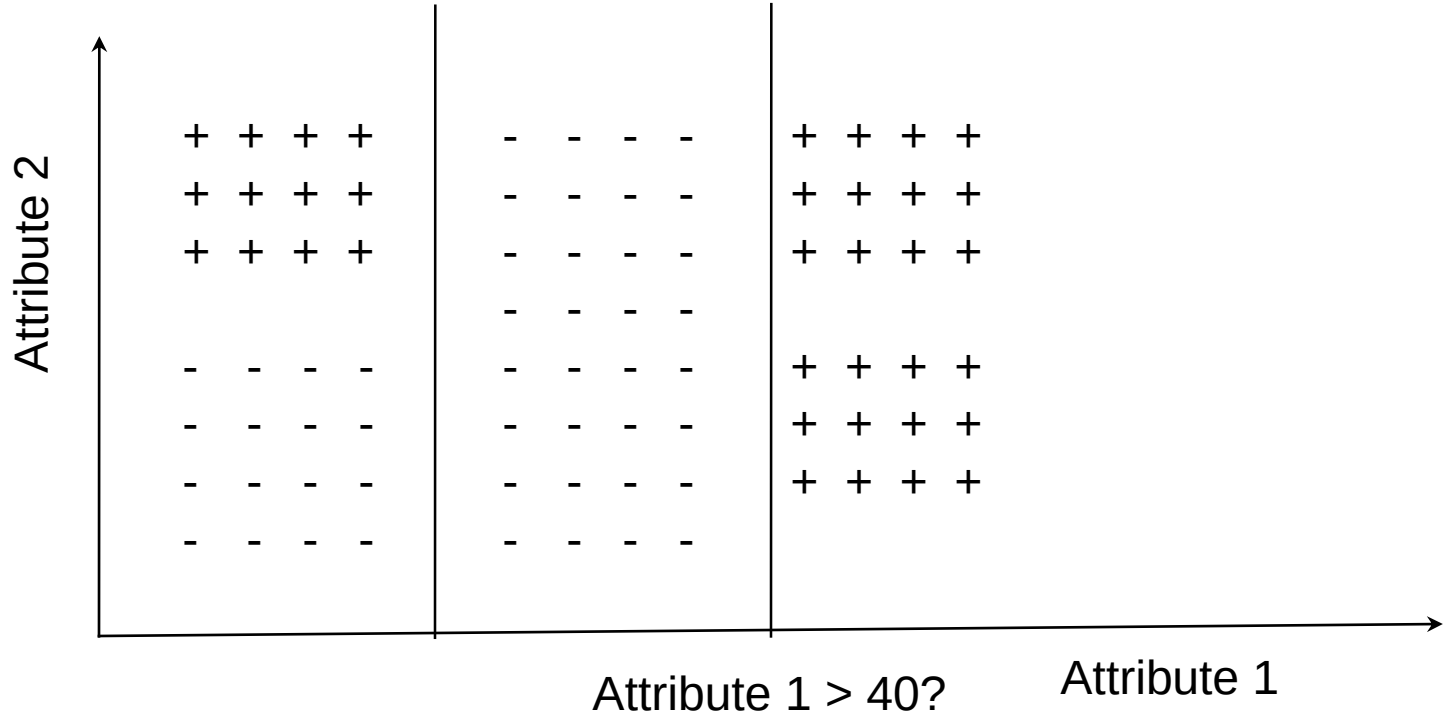
Best Attribute to Split?



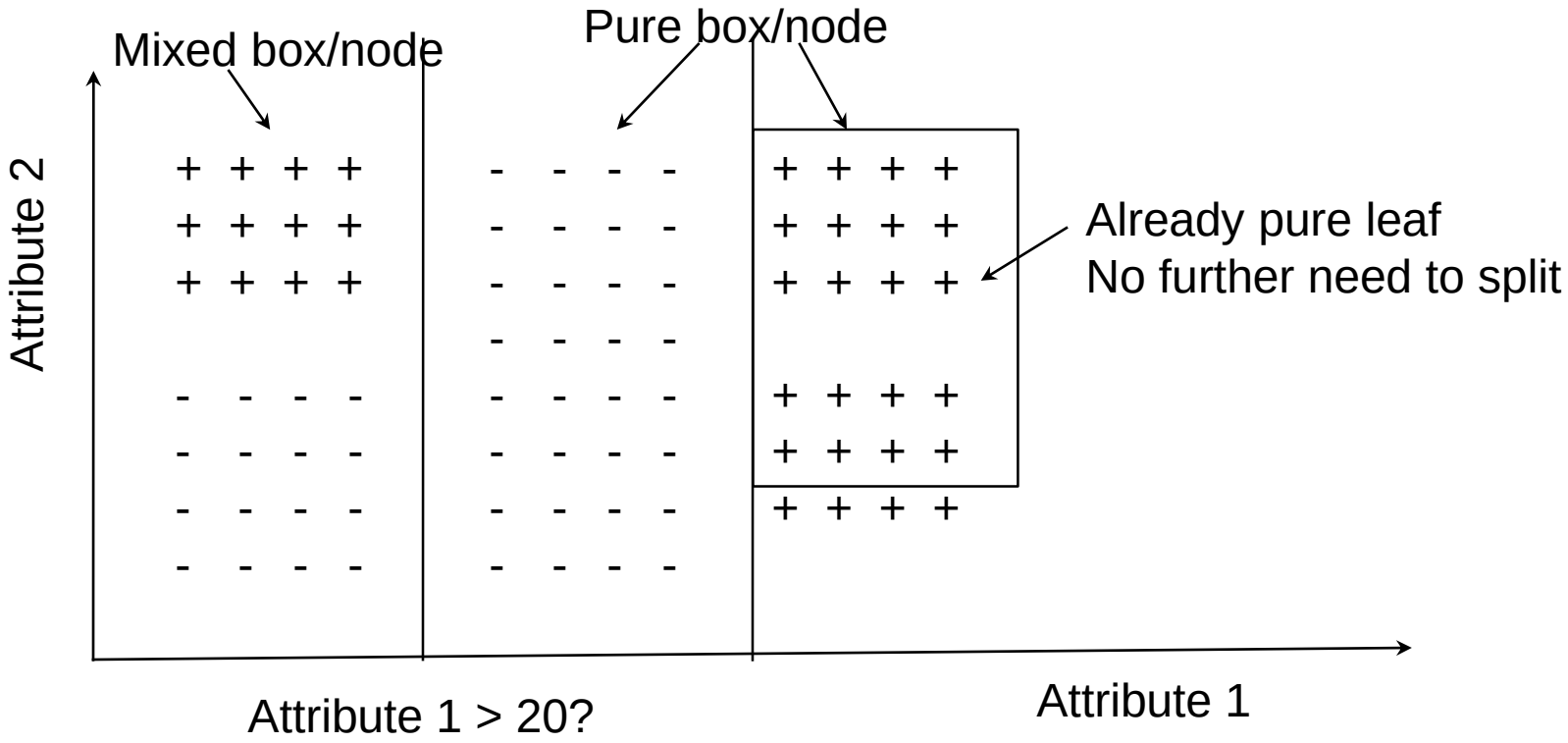
Best Attribute to Split?



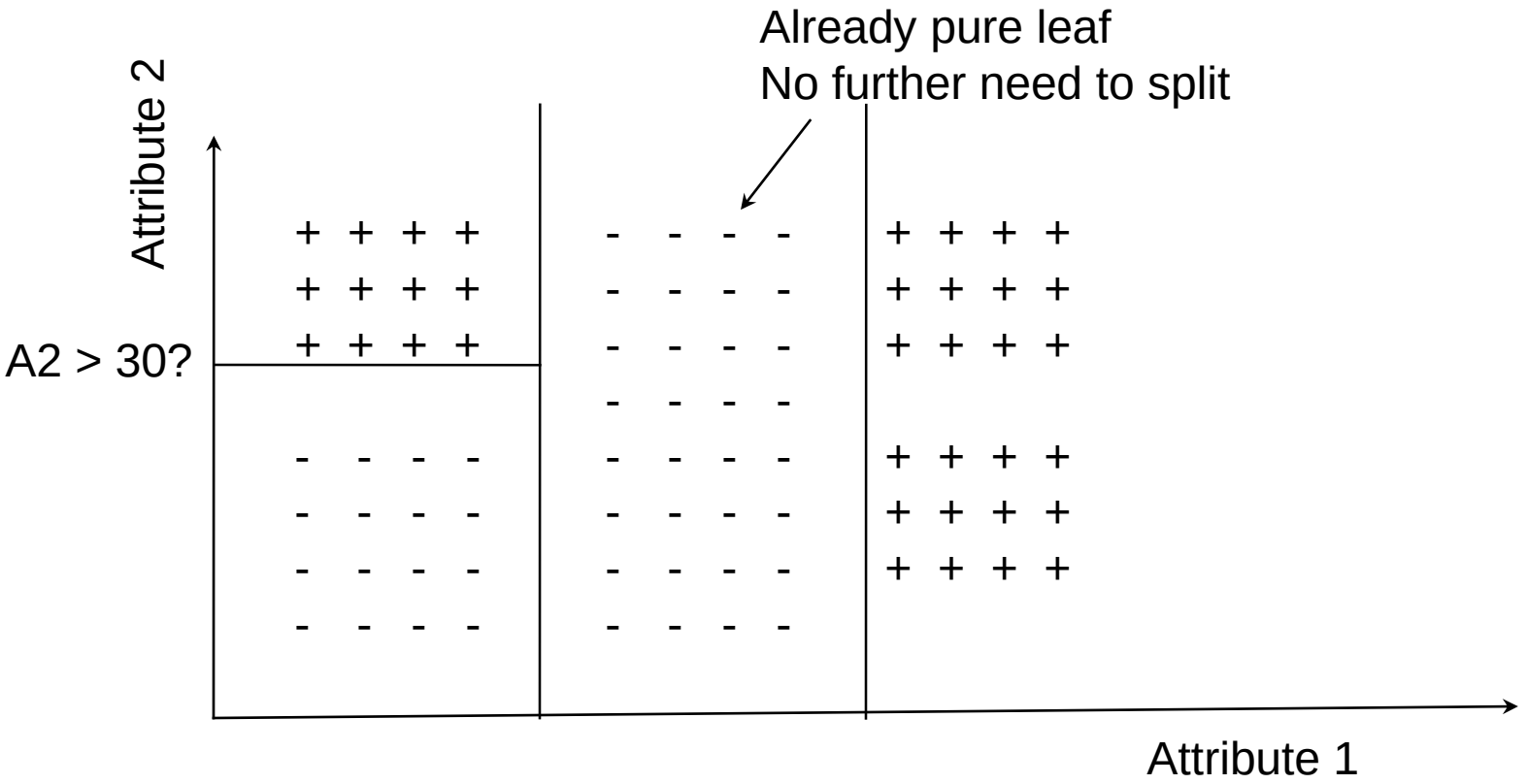
Best Attribute to Split?



Which Split to Make Next?

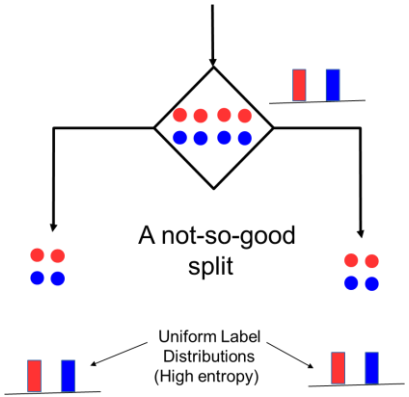


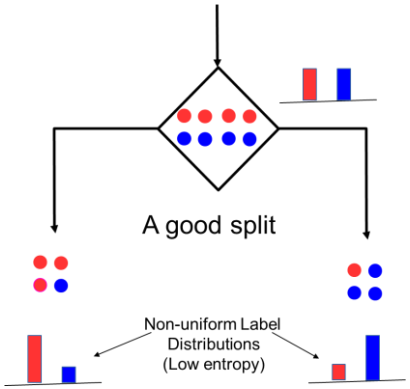
Which Split to Make Next?



How to Split at Internal Nodes?

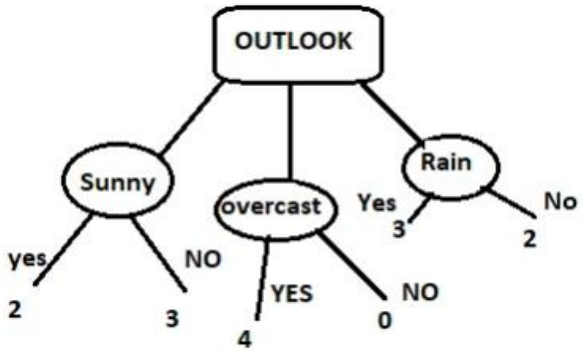
- Recall that each internal node receives a subset of all the training inputs
- Regardless of the criterion, the split should result in as "pure" groups as possible
  - A pure group means that the majority of the inputs have the same label/output





Concept of “pure node”

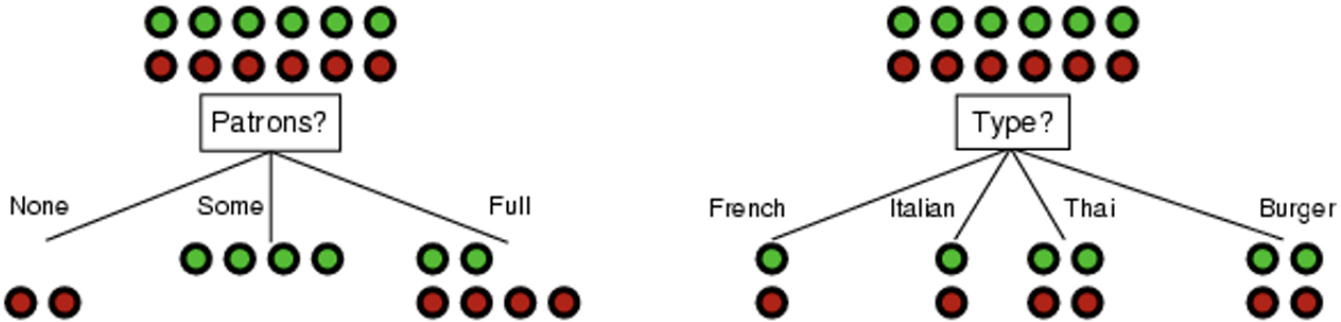
Pure node is a node **wherein all (or majority) of the data points belong to the same class** and thus it is very easy to make the prediction at such node.



Here, “Overcast” is a pure node

Try:

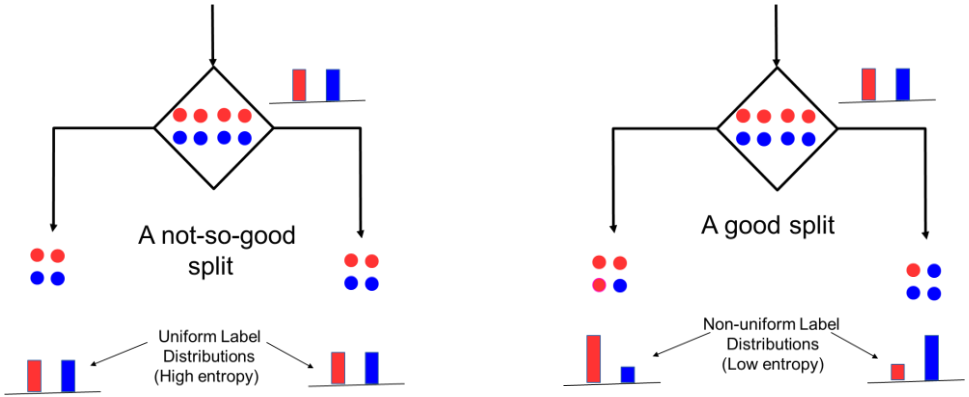
Idea: a good attribute splits the examples into subsets that are (ideally) “all positive” or “all negative”



Which split is more informative: *Patrons?* or *Type?*

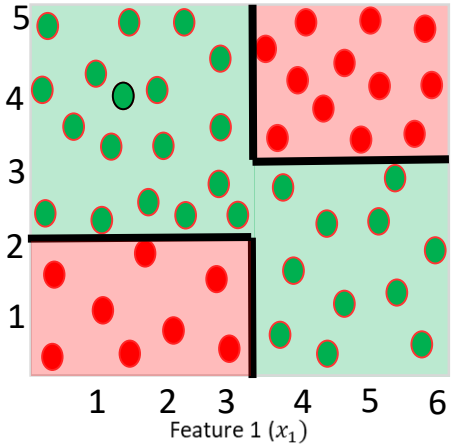
Back to: How to Split at Internal Nodes?

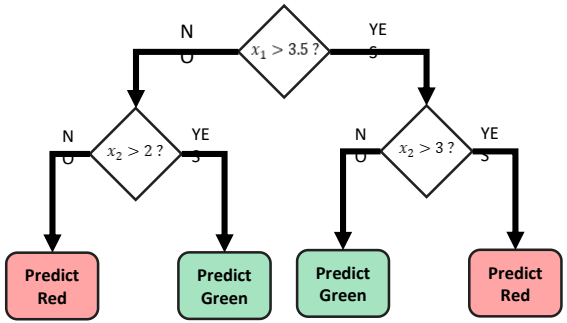
- Regardless of the criterion, the split should result in as "pure" groups as possible
  - A pure group means that the majority of the inputs have the same label/output



- For classification problems **(discrete outputs)**, entropy is a measure of purity.
  - Low entropy ⇒ high purity (less uniform label distribution → labels are mostly the same)
  - High entropy ⇒ Low purity (uniform label distribution → labels are mix)
  - Splits that give the largest reduction (before split vs after split) in entropy are preferred (this reduction is also known as "information gain")

Constructing Decision Trees





- In general, constructing DT is an intractable problem (NP-hard)
- Often we can use some “greedy” heuristics to construct a “good” DT
  - To do so, we use the training data to figure out which rules should be tested at each node
  - The same rules will be applied on the test inputs to route them along the tree until they reach some leaf node where the prediction is made

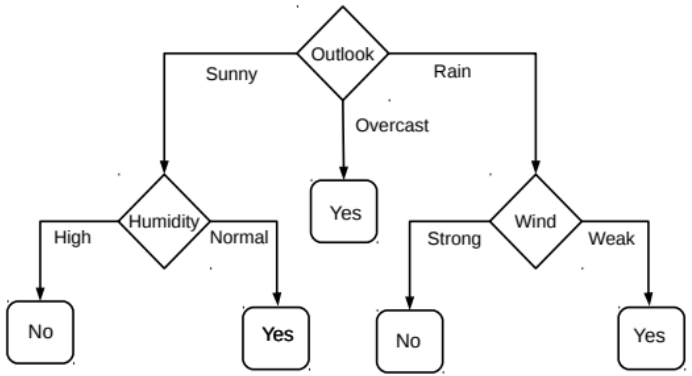
Principle of Decision Tree Construction

- We want to form pure leaves
  - Correct classification
- Greedy approach to reach correct classification
  1. Initially treat the entire data set as a single box
  2. For each box choose the spilt that reduces its impurity (in terms of class labels) by the maximum amount
  3. Split the box having highest reduction in impurity
  4. Continue to Step 2
  5. Stop when all boxes are pure

Decision Tree Construction: An Example

- Let's consider the playing Tennis example
- Assume each internal node will test the value of one of the features

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

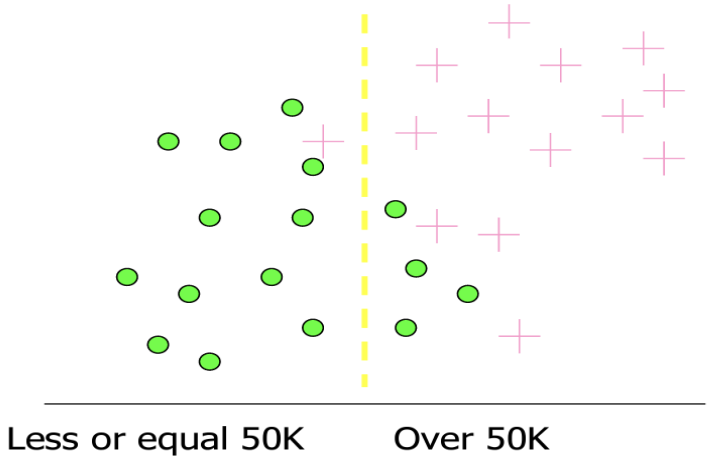


- Question: Why does it make more sense to test the feature "outlook" first?
- Answer: Of all the 4 features, it's the most informative
  - It has the highest information gain as the root node

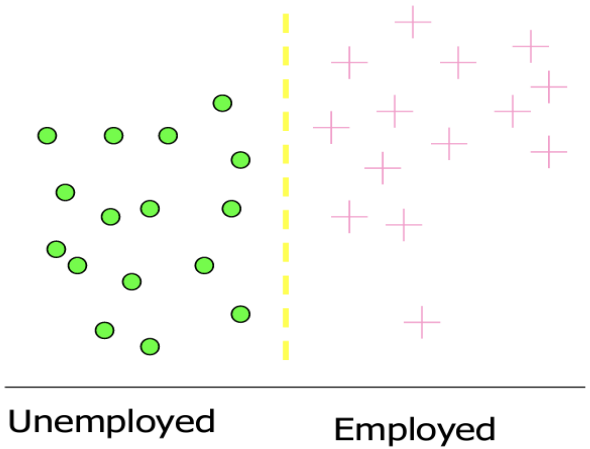
Think about

Which test is more informative?

Split over whether Balance exceeds 50K



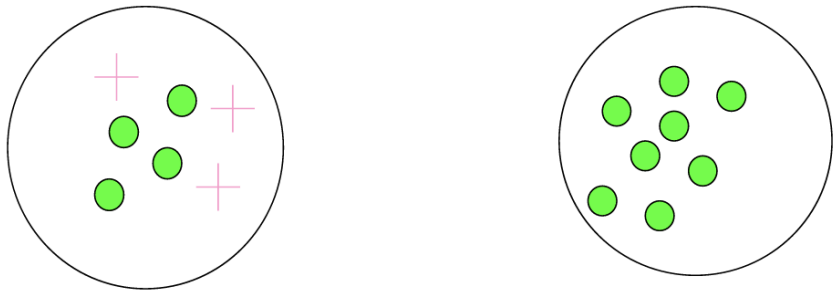
Split over whether applicant is employed



Impurity/ Entropy

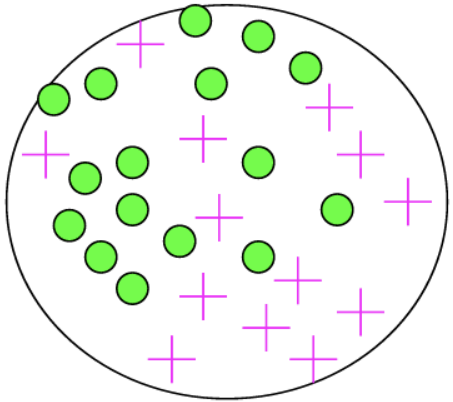
Impurity/Entropy (informal)

– Measures the level of **impurity** in a group of examples

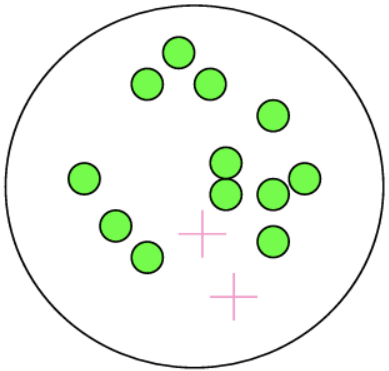


Impurity/ Entropy

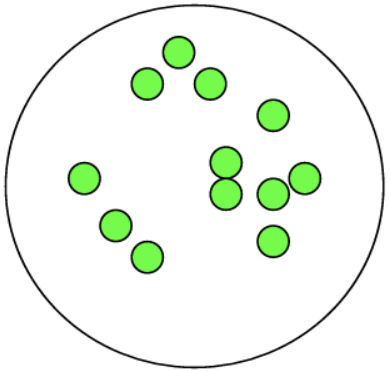
Very impure group



Less impure



Minimum impurity



How to determine the Best Split

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, **High degree of impurity**	**Homogeneous,** **Low degree of impurity**

“Entropy”

A common way to measure impurity (or uncertainty)

Measuring uncertainty

- Good split if we are more certain about classification after split
  - Deterministic good (all true or all false)
  - Uniform distribution bad
  - What about distributions in between?

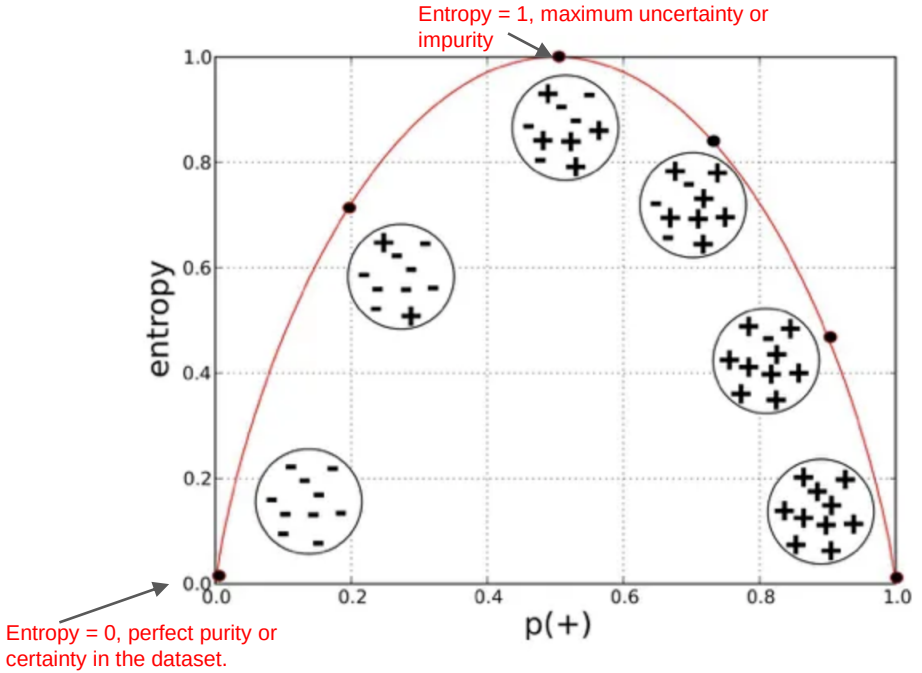
P(Y=A) = 1/2	P(Y=B) = 1/4	P(Y=C) = 1/8	P(Y=D) = 1/8

P(Y=A) = 1/4	P(Y=B) = 1/4	P(Y=C) = 1/4	P(Y=D) = 1/4

Sample Entropy

- 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

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



Provost, Foster; Fawcett, Tom. Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking

The x-axis measures the proportion of data points belonging to the positive class in each bubble and the y-axis axis measures their respective entropies.

Entropy Formula

- Assume a set of labelled inputs $\boldsymbol{S}$ from C classes, p_c as fraction of class c inputs
- Entropy of the set $\boldsymbol{S}$ is defined as $\mathrm{H}(\boldsymbol{S}) = -\sum_{c \in C} p_c \log p_c$

Entropy Computing Example:

Entropy = $$H(S) = -\sum_{c \in C} p_c \log p_c$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Information Gain

In the context of Decision Trees (DT), "information gain" is a metric used to determine the relevance of a feature in classifying the data.

- It measures how much a particular feature contributes to decreasing the uncertainty about the classification of the data points.
- It is calculated as the difference between the entropy of the dataset before and after splitting on feature A.

At each node of the decision tree, we choose the **attribute that maximizes information gain as the splitting criterion**. This helps us decide how to partition the dataset into smaller, more homogeneous subsets based on the values of the chosen attribute.

From Entropy to Information Gain Formula

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

$$Gain(S, A) = GAIN_{split} = H(S) - \sum \left(\frac{|S_i|}{|S|} \cdot H(S_i) \right)$$

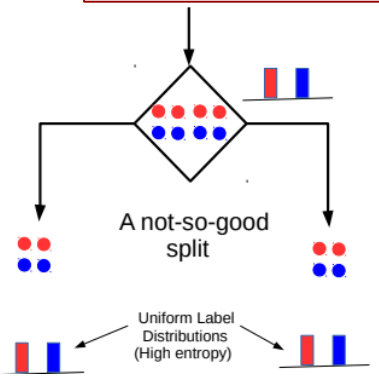
- G represents the information gain.
- $H(S)$ is the entropy of the original dataset S.
- k is the number of subsets created by the split.
- $|S_i|$ is the number of instances in the i-th subset S_i after the split.
- $|S|$ is the total number of instances in the original dataset S.
- $H(S_i)$ is the entropy of the i-th subset S_i.

Information Gain with Formula:

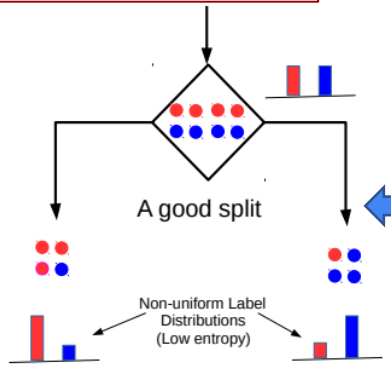
- Suppose a rule splits $\boldsymbol{S}$ into two smaller disjoint sets $\boldsymbol{S_1}$ and $\boldsymbol{S_2}$
- Reduction in entropy after the split is called information gain

$$IG = H(S) - \frac{|S_1|}{|S|} H(S_1) - \frac{|S_2|}{|S|} H(S_2)$$

This split has a low IG
(in fact zero IG)



vs



This split has higher IG

Remember: goal is to

Important: Find the attribute or feature for splitting the dataset that **minimizes entropy or maximizes information gain**, thereby reducing the uncertainty and making the dataset more homogeneous (in terms of the target variable or class labels) after the split; in order to achieve better classification performance in decision tree algorithms.

Example: Entropy and Information Gain

Consider, Training Examples

Day	Outlook	Temp	Humidity	Wind	Tennis?
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*

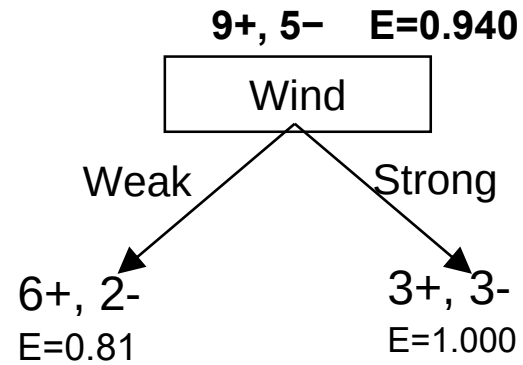
Determine the Root Attribute:

Let's use IG based criterion to construct a DT for the Tennis example

- At root node, let's compute IG of each of the 4 features
- Consider feature "wind". Root contains all examples ***S*** = [9+,5-]

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

$$H(S) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.94$$



$$S_{\text{weak}} = [6+, 2-] \Rightarrow \text{H}(S_{\text{weak}}) = 0.811$$
$$S_{\text{strong}} = [3+, 3-] \Rightarrow \text{H}(S_{\text{strong}}) = 1$$

$$IG(S, wind) = H(S) - \frac{|S_{\text{weak}}|}{|S|} H(S_{\text{weak}}) - \frac{|S_{\text{strong}}|}{|S|} H(S_{\text{strong}}) = 0.94 - 8/14 * 0.811 - 6/14 * 1 = 0.048$$

Determine the Root Attribute:

Let's use IG based criterion to construct a DT for the Tennis example

- At root node, let's compute IG of each of the 4 features
- Consider feature "wind". Root contains all examples ***S*** = [9+,5-]

$$H(S) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.94$$

$$S_{\text{weak}} = [6+, 2-] \Rightarrow \text{H}(S_{\text{weak}}) = 0.811$$

$$S_{\text{strong}} = [3+, 3-] \Rightarrow \text{H}(S_{\text{strong}}) = 1$$

$$IG(S, wind) = H(S) - \frac{|S_{\text{weak}}|}{|S|} H(S_{\text{weak}}) - \frac{|S_{\text{strong}}|}{|S|} H(S_{\text{strong}}) = 0.94 - 8/14 * 0.811 - 6/14 * 1 = 0.048$$

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

- Likewise, at root: IG(S, outlook) = **0.246**, IG(S, humidity) = 0.151, IG(S,temp) = 0.029
- Thus w**e choose "outlook" feature** to be tested at the root node

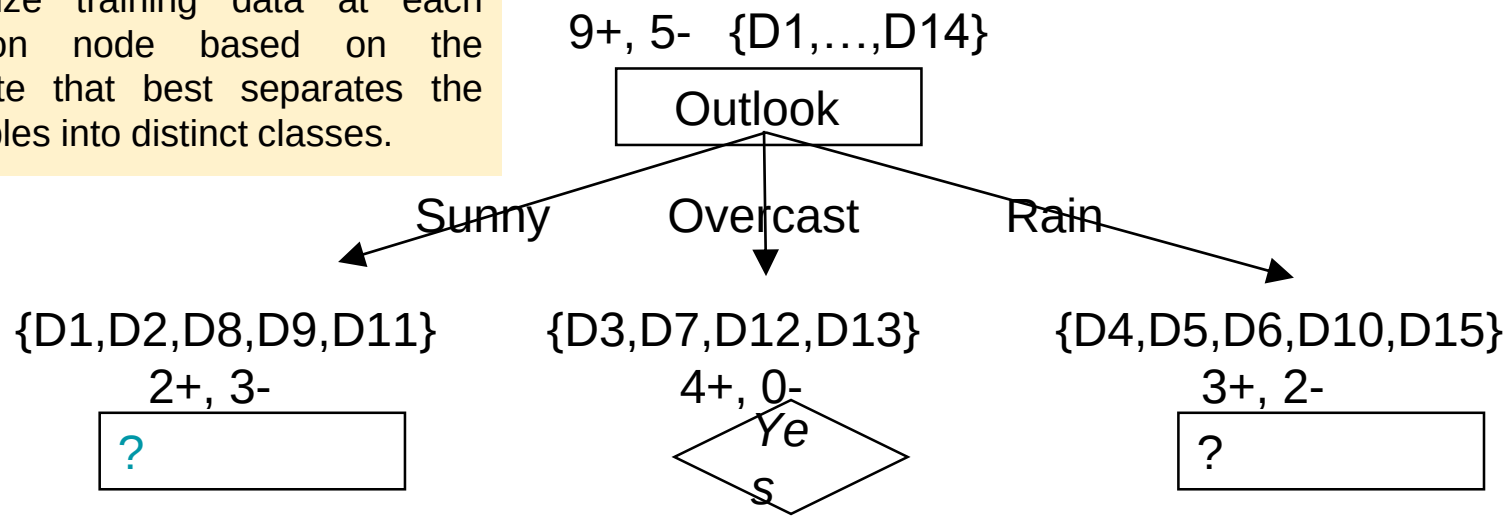
Now

- Now how to grow the DT, i.e., what to do at the next level? Which feature to test next?
- Rule: Iterate - for each child node, **select the feature with the highest IG**

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

Sort the Training Example and Growing the tree

Organize training data at each decision node based on the attribute that best separates the examples into distinct classes.



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

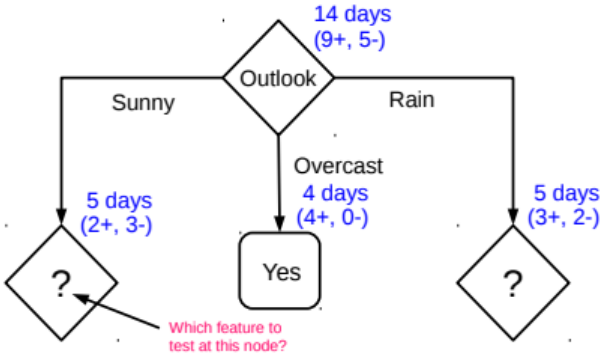
Gain (S_{sunny}, Humidity) = .970

Gain (S_{sunny}, Temp) = .570

Gain (S_{sunny}, Wind) = .019

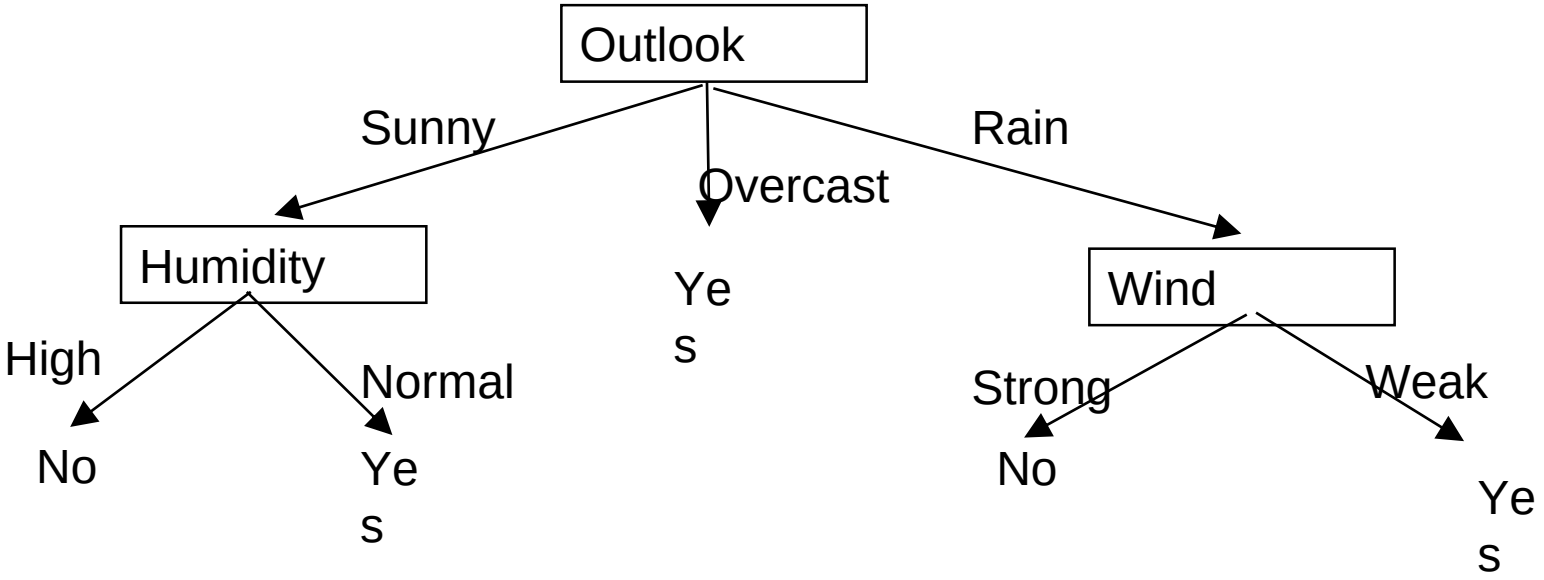
Growing the tree

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



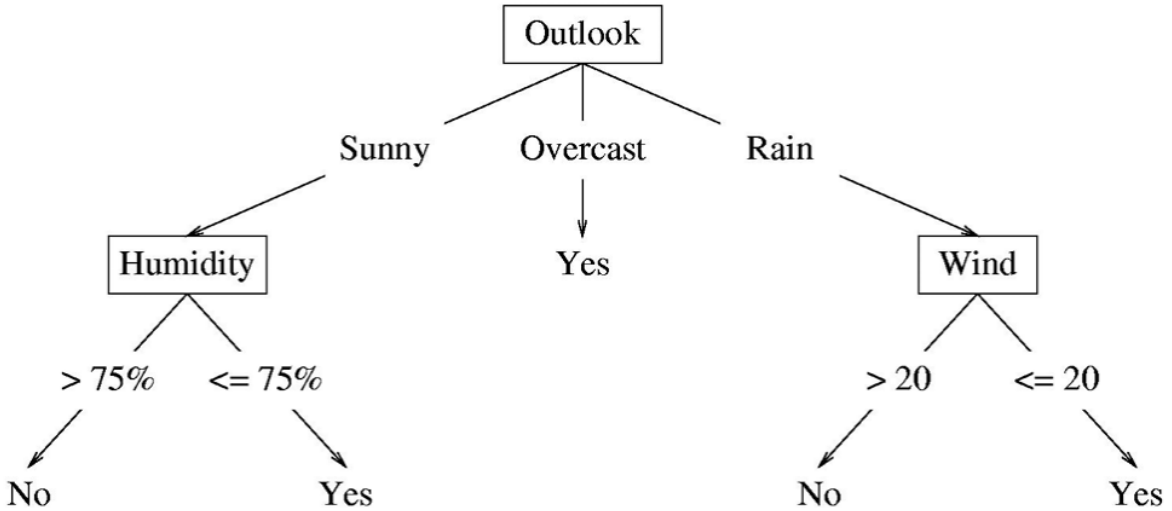
- So, proceeding as before, for level 2, left node, we can verify that
  - IG(S,temp) = 0.570, **IG(S, humidity) = 0.970**, IG(S, wind) = 0.019
- **Thus humidity chosen as the feature to be tested at level 2, left node**
  - No need to expand the middle node (already “pure” - all “yes” training examples)
- Can also verify that **wind has the largest IG for the right node.**
- **Note:** If a feature has already been tested along a path earlier, we don’t consider it again

Final Decision Tree



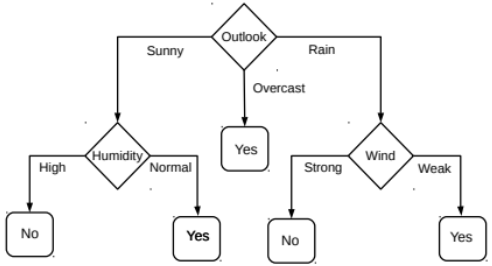
What about? If features are continuous:

Internal nodes can test the value of a feature against a threshold.

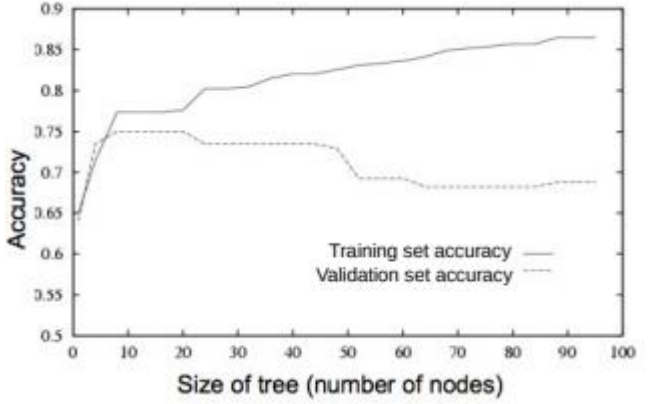


When to stop growing the tree?

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

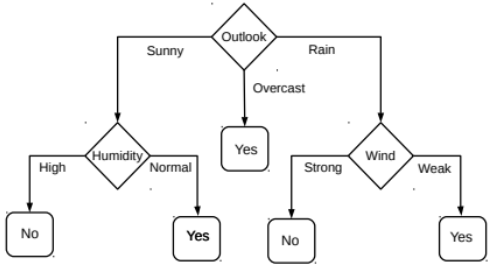


- Stop expanding a node further (i.e., make it a leaf node) when
  - It consist of all training examples having the same label (the node becomes “pure”)
  - We run out of features to test along the path to that node
  - The DT starts to overfit (can be checked by monitoring the validation set accuracy)



When to stop growing the tree?

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



- **Important:** No need to obsess too much for purity

(To help prevent the tree from growing too much!)

- It is okay to have a leaf node that is not fully pure, e.g., this
- **Predictions at Impure Leaf Nodes:** For test inputs that reach an impure leaf, we can predict probability of belonging to each class (in above example, p(red) = 3/8, p(green) = 5/8), or simply predict the majority labell, which in this case would be green.

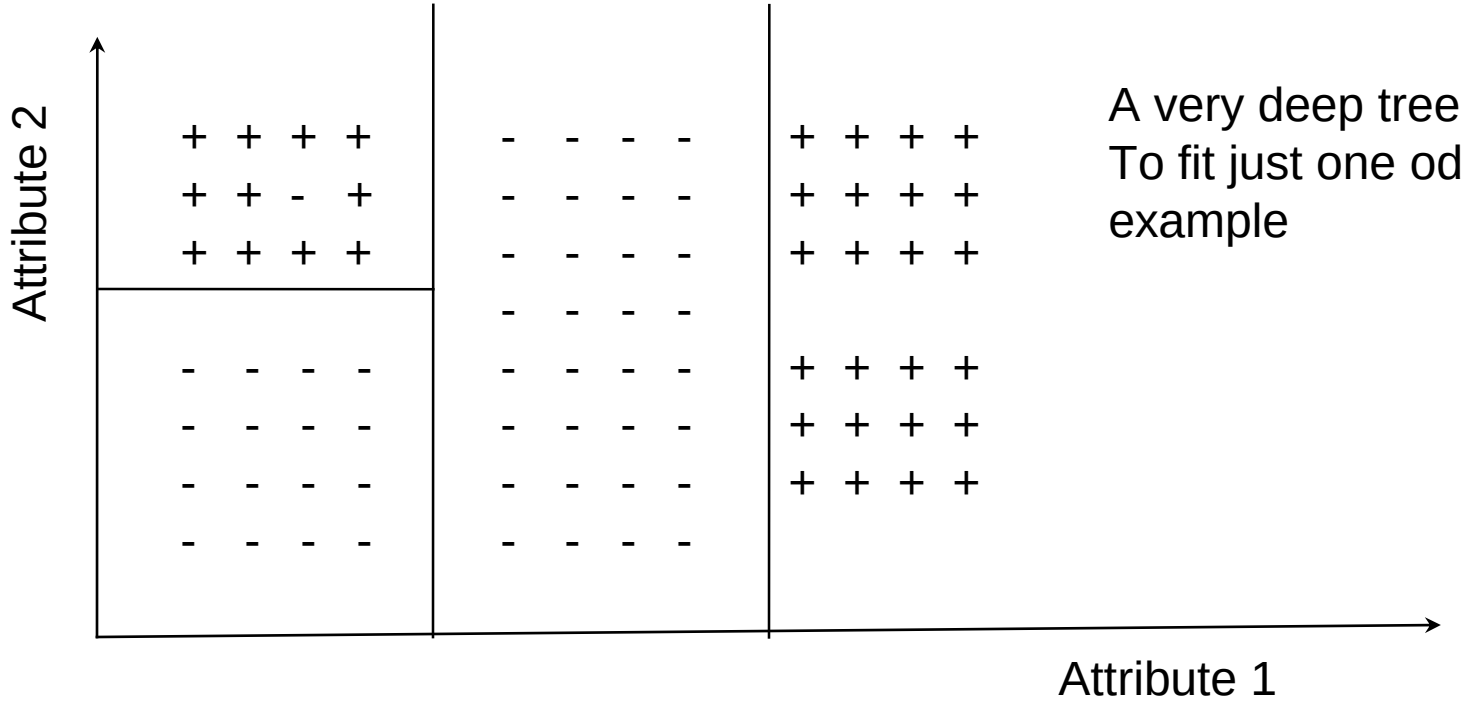
Summary: Determine when to stop splitting

Stopping Criteria:

The recursive splitting process continues until a stopping criterion is met. Common stopping criteria include

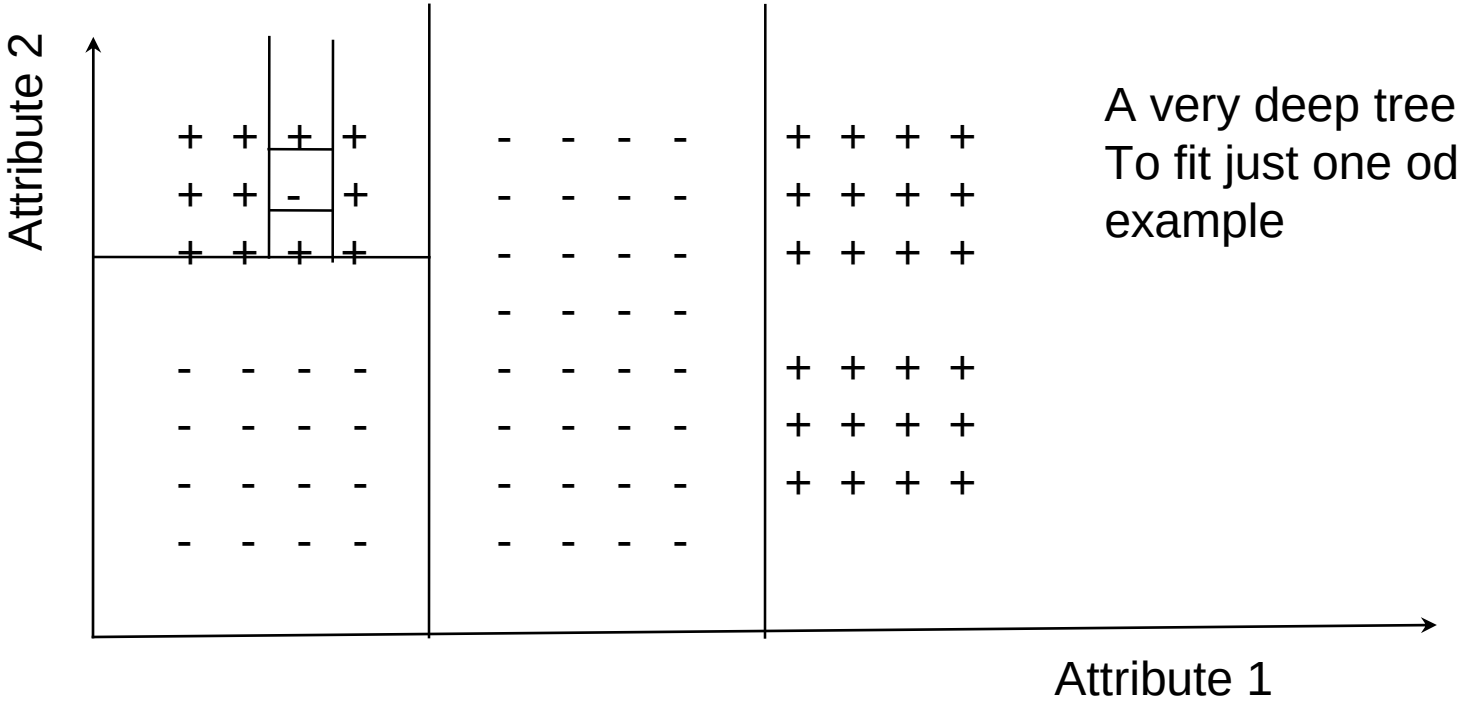
- ❏ Reaching a maximum depth (preventing it from becoming overly complex).
- ❏ Having a minimum number of samples per leaf (Splitting stops if a leaf node would contain fewer samples than a specified threshold).
- ❏ Encountering pure nodes (nodes containing only one class).

Overfitting



A very deep tree required
To fit just one odd training example

When to stop Splitting Further?



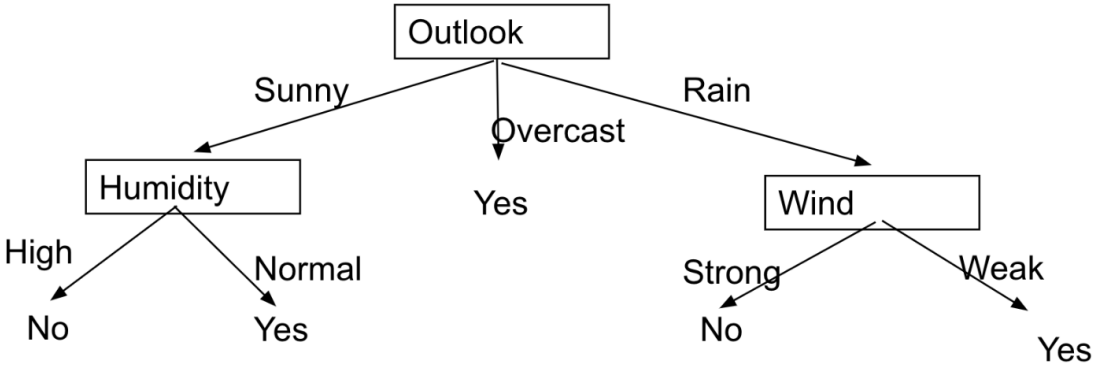
A very deep tree required
To fit just one odd training example

Overfitting in Decision Tree

- Consider adding *noisy* training example (should be +):

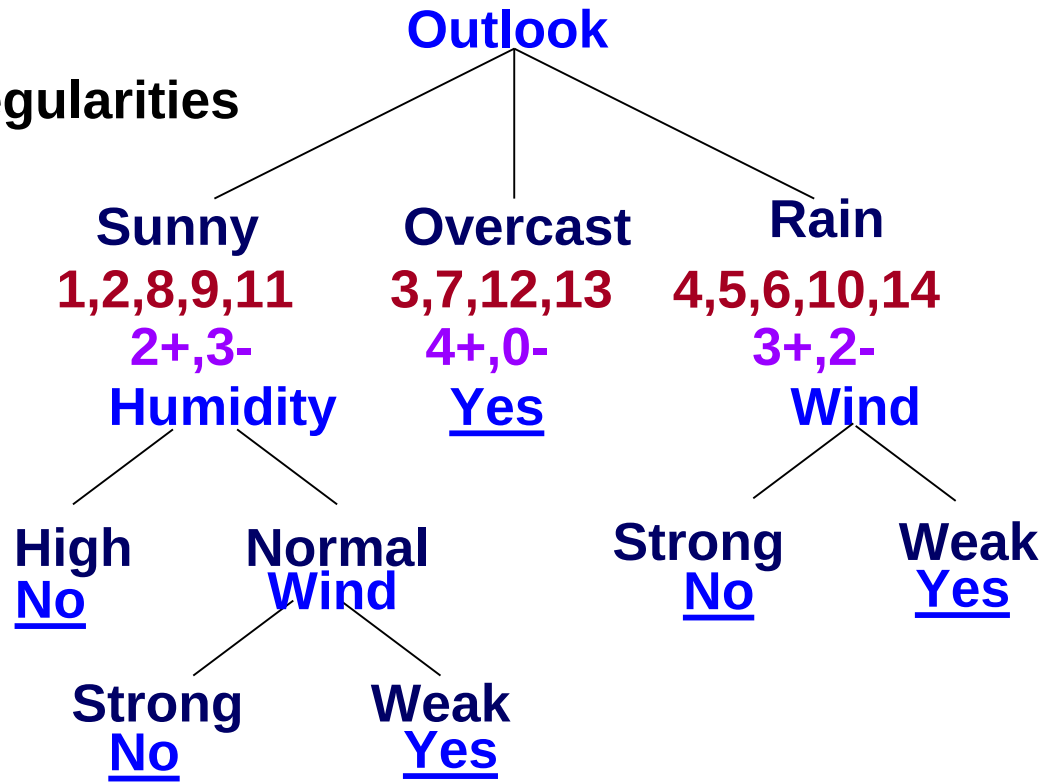
Day	Outlook	Temp	Humidity	Wind	Tennis?
D15	Sunny	Hot	Normal	Strong	*No*

- What effect on earlier tree?



Overfitting in Decision Tree - Example

Noise or other coincidental regularities



Avoiding Overfitting in DTs

- **Desired:** a DT that is not too big in size, yet fits the training data reasonably

- **Mainly two approaches to prune a complex DT**
  - Prune while building the tree (stopping early)
  - Prune after building the tree (post-pruning)

Avoiding Overfitting in DTs

- **Criteria for judging which nodes could potentially be pruned**
  - Use a validation set (separate from the training set)
    - Prune each possible node that doesn't hurt the accuracy on the validation set
    - Greedily remove the node that improves the validation accuracy the most
    - Stop when the validation set accuracy starts worsening

Prepruning

Two basic approaches

1. Prepruning: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.

Advantages:

- Prevents overfitting early on.
- Faster computation as the tree is smaller.

Disadvantages:

- May miss important patterns if stopped too early.
- Decision to stop may be subjective.

Postpruning

Two basic approaches

2. Postpruning: Grow the full tree (Allow tree to grow until best fit (allow overfitting)) and then remove nodes that seem not to have sufficient evidence or or do not significantly contribute to the overall accuracy of the tree. (more popular)

Advantages:

- Captures all possible patterns.
- Reduces overfitting by removing unnecessary nodes.

Disadvantages:

- More computationally expensive initially.
- Risk of overfitting if not properly validated.

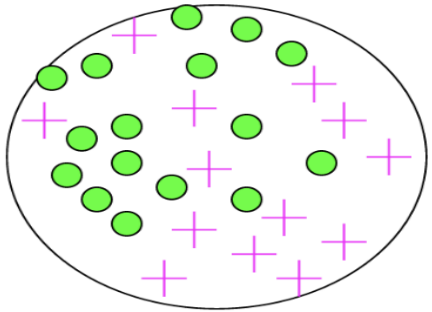
One more example of Calculating Entropy and Information Gain

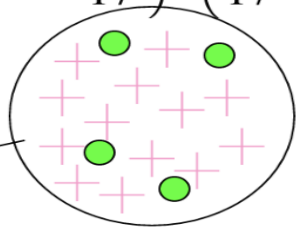
Calculating Information Gain

Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$

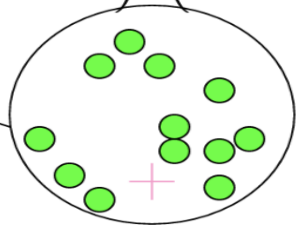
Entire population (30 instances)





17 instances

child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$



13 instances

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain= 0.996 - 0.615 = 0.38

Entropy Summary

Entropy of 1: Maximum Uncertainty: signifies that all outcomes are equally likely, and there's maximum uncertainty or disorder. It takes a full bit of information to describe each outcome.

Entropy Between 1 and 0: Intermediate Uncertainty: as entropy decreases from 1 to a fraction between 1 and 0, uncertainty decreases. This indicates that some outcomes are more likely than others.

Entropy of 0: Minimum Uncertainty: there is no uncertainty or disorder. One outcome is certain, and all others have no probability. It takes no information to describe the outcome.

Utilizing Decision Trees: From Training to Classification

To use a decision tree:

- We first go through the **training phase** where the tree decides on all its possible splits
- We then use it for **classification**, where we show it data with no label and it picks the label it thinks is best

Insight: Smaller is Better

- Fewer rules: More generalization
- Many rules: More specificity

We want to use as few rules as possible to classify our data.

Decision Trees: A Summary

Some key strengths:

- Simple and easy to interpret
- Nice example of “divide and conquer” paradigm in machine learning (thus helping us learn complex rule as a combination of several simpler rules)
- Easily handle different types of features (real, categorical, etc.)
- Very fast at test time
- Multiple DTs can be combined via ensemble methods: more powerful (e.g., Decision Forests; will see later)
- Used in several real-world ML applications, e.g., recommender systems, gaming, etc.

Some key weaknesses:

- Learning optimal DT is (NP-hard) intractable. Existing algos mostly greedy heuristics
- Can sometimes become very complex unless some pruning is applied

Additional Reading Slides

Basic Algorithm for Top-Down Induction of Decision Trees

[ID3, C4.5 by Quinlan]

ID3 is a specific algorithm for building decision trees using a top-down induction approach

node = root of decision tree

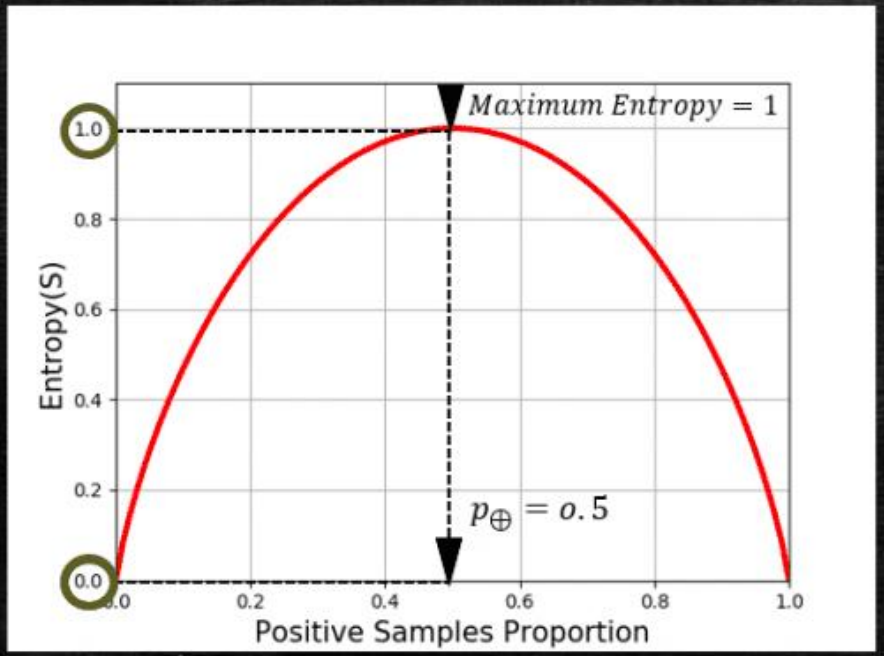
Main loop:

1. *A* ← the "best" decision attribute for the next node.
2. Assign *A* as decision attribute for *node*.
3. For each value of *A*, create a new descendant of *node*.
4. Sort training examples to leaf nodes.
5. If training examples are perfectly classified, stop. (Pure node)
   Else, recurse over new leaf nodes.

How do we choose which attribute is best?

https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/ml-decision-tree/tutorial/

The Behavior of the Entropy Function

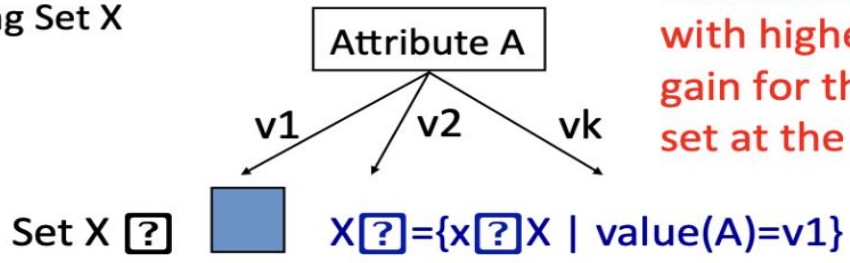


- When is $Entropy = 0$?
  - When the set is ***absolutely*** pure:
    - $p_{\oplus} = 0$ implies $p_{\ominus} = 1$
    - $p_{\oplus} = 1$ implies $p_{\ominus} = 0$
    - The state of absolute certainty!
    - Leaves have such states!

- But when is $Entropy = 1$?
  - When examples are half $\oplus$ and half $\ominus$, ***exactly***
  - $p_{\oplus} = o.5$ implies $p_{\ominus} = 0.50$
  - The state of absolute confusion!

- When is $0 < Entropy < 1$?
  - Only if $p_{\oplus} \neq p_{\ominus}$

Using Information Gain to Construct a Decision Tree



Choose the attribute A with highest information gain for the full training set at the root of the tree.

Construct child nodes for each value of A. Each has an associated subset of vectors in which A has a particular value.

Disadvantage of information gain:

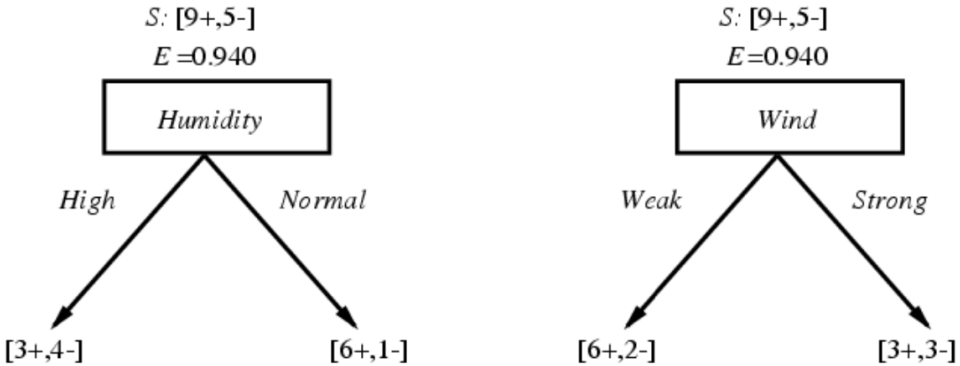
- It prefers attributes with large number of values that split the data into small, pure subsets
- Quinlan's gain ratio uses normalization to improve this

Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenı
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

Selecting the Next Attribute

Which attribute is the best classifier?

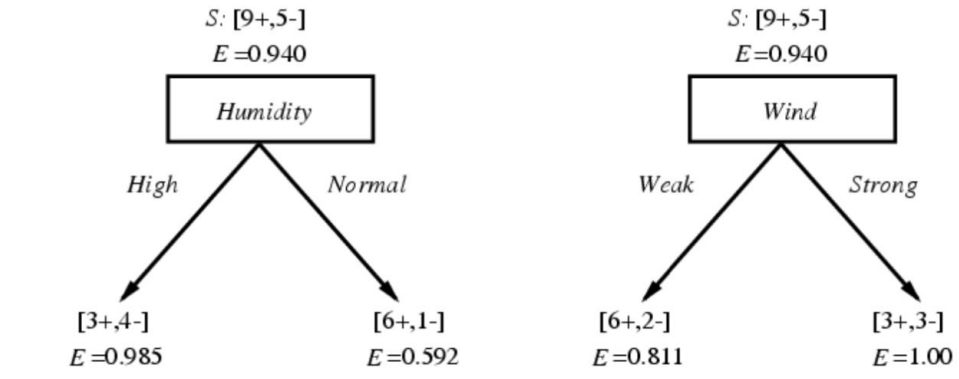


Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenn
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

Selecting the Next Attribute

Which attribute is the best classifier?



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

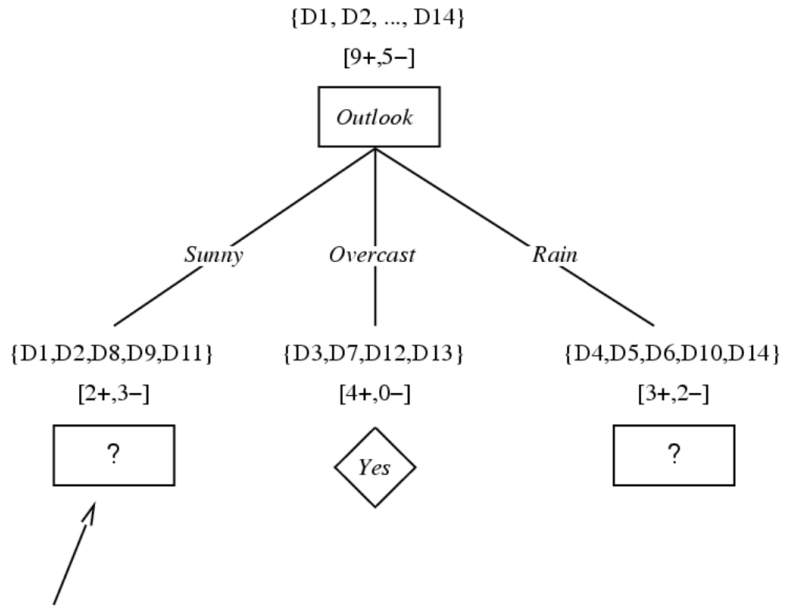
Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenn
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

Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenn
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 should be tested here?

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