
Decision Tree

Decision Tree

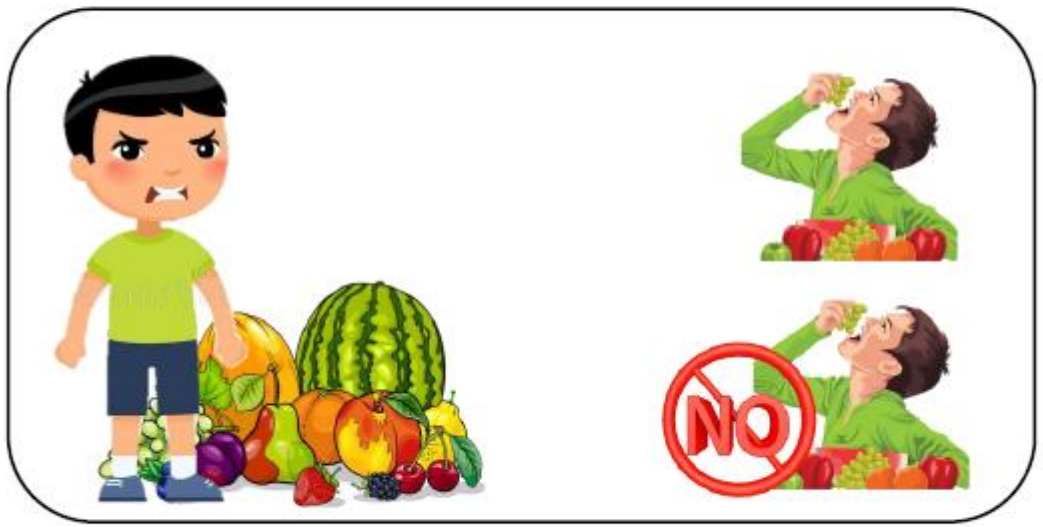
- Decision Tree algorithm belongs to the family of supervised learning algorithms.

- Decision trees are hierarchical structures comprising nodes, branches, and leaf nodes. Nodes represent attributes or features, branches represent decisions, and leaf nodes represent outcomes or classes.

- Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems as well.

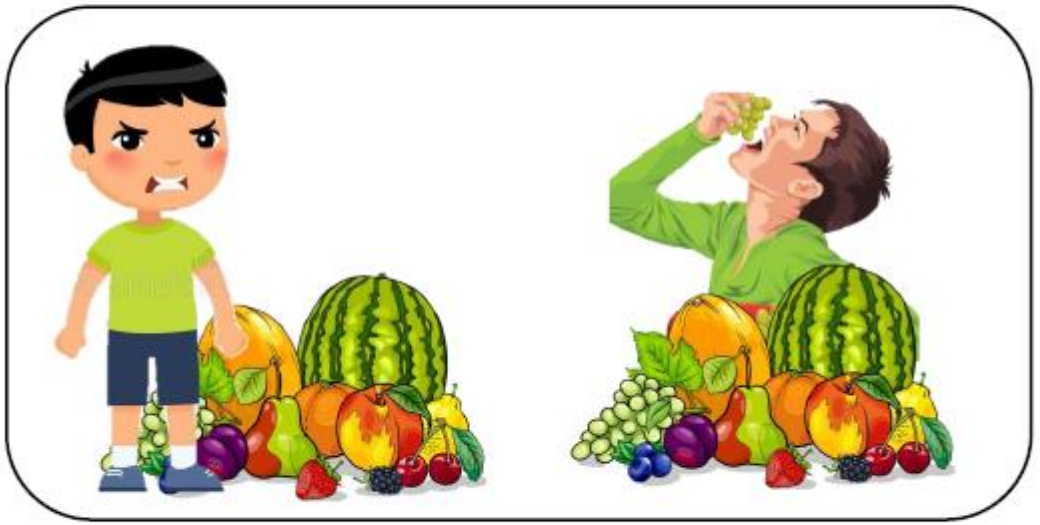
CART - CLASSIFICATION AND REGRESSION TREES

CLASSIFICATION :



INPUT: A KID WITH FRUITS
OUTPUT: WILL HE EAT OR NOT

REGRESSION:



INPUT: A KID WITH FRUITS
OUTPUT: HOW MANY FRUITS HE EATS

Image source: J.Sai Vardhan

Important Terminology related to Decision Trees

Point	Description
1. Root Node	Represents the entire population or sample, further divided into two or more sets.
2. Splitting	Process of dividing a node into two or more sub-nodes.
3. Decision Node	Occurs when a sub-node splits into further sub-nodes.
4. Leaf/ Terminal Node	A node that does not split, also known as leaf or sub-tree.
5. Pruning	Involves cutting down some nodes to prevent overfitting.
6. Parent and Child Node	A node divided into sub-nodes is the parent node, while the sub-nodes are its children.

Types of Decision Trees

Decision trees can be categorized in various ways based on different criteria

Category	Types
Based on Target Variable	Categorical Variable Decision Tree
	Continuous Variable Decision Tree
Based on Tree Structure	Binary Decision Tree
	Multiway Decision Tree
Based on Usage	Classification Tree
	Regression Tree
Based on Algorithm	ID3 (Iterative Dichotomiser 3)
	C4.5
	CART (Classification and Regression Trees)
	CHAID (Chi-squared Automatic Interaction Detection)
	Random Forest
	Gradient Boosting Decision Trees

The general steps to build a Decision tree are as follows

- **Data Preparation**

Collect and preprocess the data to be used in building the decision tree.

- **Attribute Selection**

Identify the most important attributes or features in the data that will be used to build the decision tree. There are several techniques for attribute selection, such as information gain, gain ratio, and Gini Index.

- **Tree Building**

Use an algorithm such as ID3, C4.5, or CART to construct the decision tree. This involves recursively splitting the data into subsets based on the selected attributes until the tree is complete.

- **Tree Pruning**

Remove unnecessary branches or nodes from the decision tree to improve its accuracy and reduce overfitting. There are several pruning techniques such as reduced error pruning and cost complexity pruning.

Attributes Selection

- If the dataset consists of N attributes, deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step.
- Randomly selecting any node to be the root can't solve the issue; following a random approach may result in poor accuracy.

- To address this attribute selection problem, researchers have devised some solutions. They suggested using criteria such as:

  - Entropy
  - Information Gain
  - Gini Index

Attributes Selection

Criterion	Explanation	When to Use
Entropy	- Measures impurity or randomness of dataset - Quantifies uncertainty of dataset's labels - Goal is to minimize entropy by selecting attributes resulting in homogeneous subsets at each split	Use when aiming for a balanced split of classes in the dataset, especially when classes are evenly distributed.
Information Gain	- Measures usefulness of an attribute in classifying data - Quantifies reduction in entropy achieved by splitting data on attribute - Higher information gain implies attribute is more effective in classifying data, making it a good candidate for splitting nodes in decision tree	Use when aiming to maximize the amount of information gained about the class labels after splitting the dataset on a particular attribute.
Gini Index	- Measures degree of impurity in dataset - Calculates probability of misclassifying randomly chosen element - Goal is to minimize Gini Index by selecting attributes resulting in more homogeneous subsets, similar to minimizing entropy	Use when seeking to minimize misclassification error by selecting attributes that result in the most homogeneous subsets in terms of class labels.

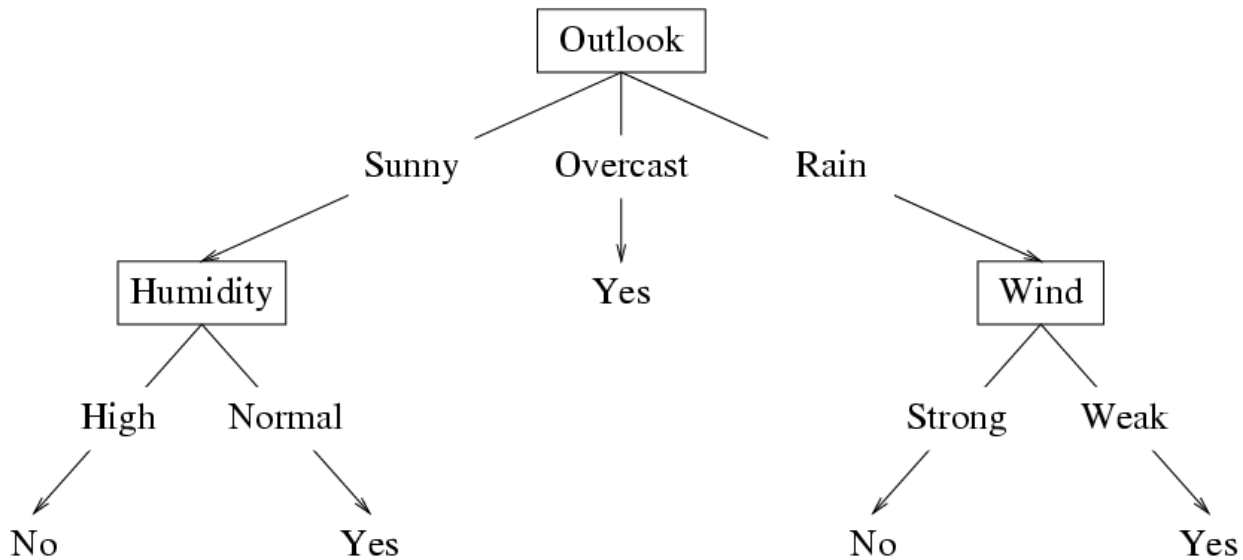
Training Data Example Goal is to Predict When This Player Will Play Tennis?

PlayTennis: 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

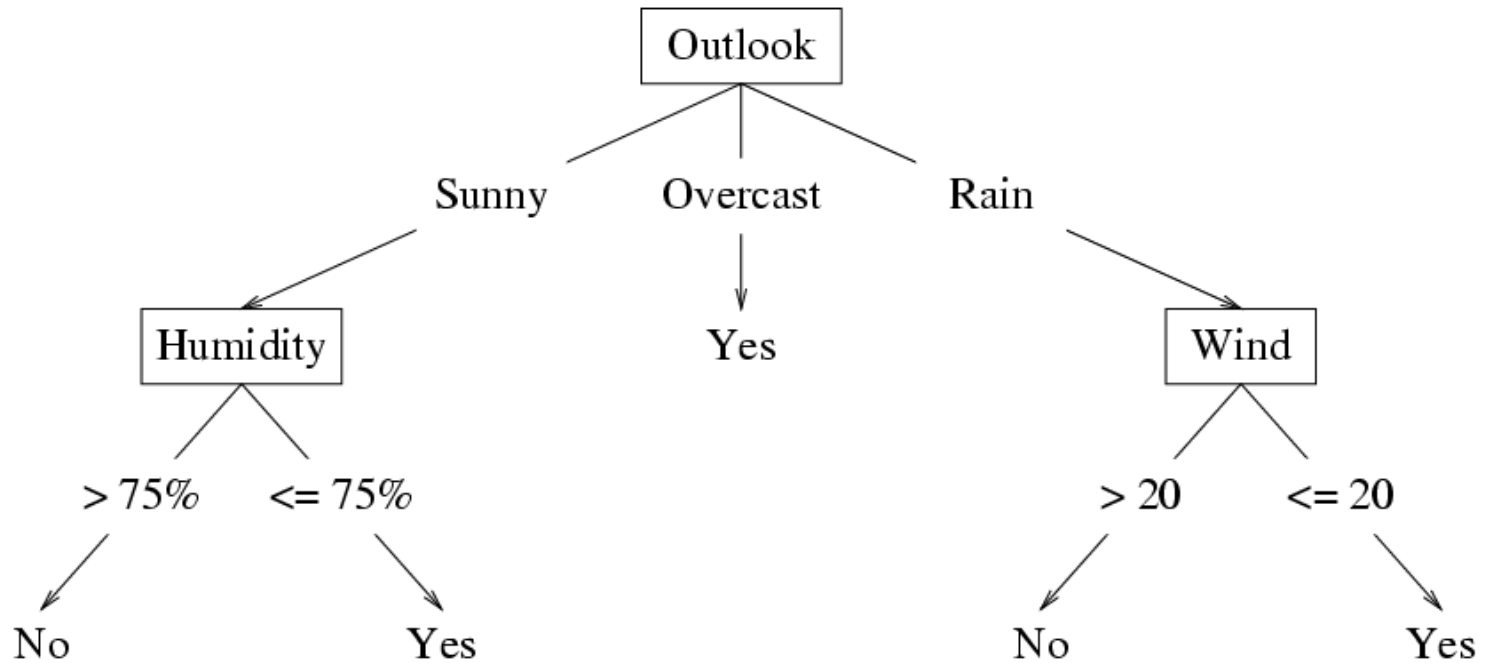
Decision Tree Hypothesis Space

- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.
- **Leaf nodes** specify the class $h(\mathbf{x})$.



Suppose the features are **Outlook** (x_1), **Temperature** (x_2), **Humidity** (x_3), and **Wind** (x_4). Then the feature vector $\mathbf{x} = (Sunny, Hot, High, Strong)$ will be classified as **No**. The **Temperature** feature is irrelevant.

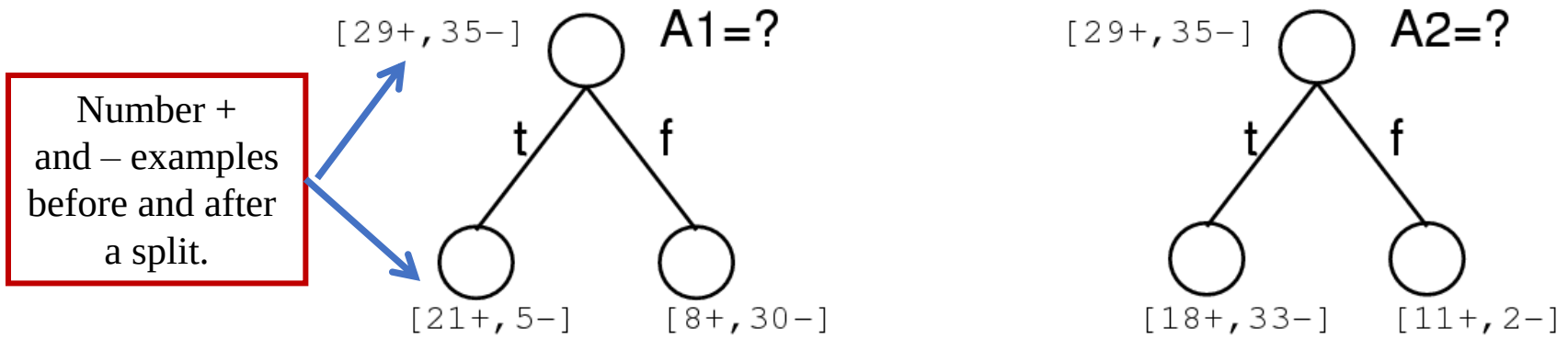
If the features are continuous, internal nodes may test the value of a feature against a threshold.



Choosing the Best Attribute

A1 and A2 are “attributes” (i.e. features or inputs).

Which attribute is best?



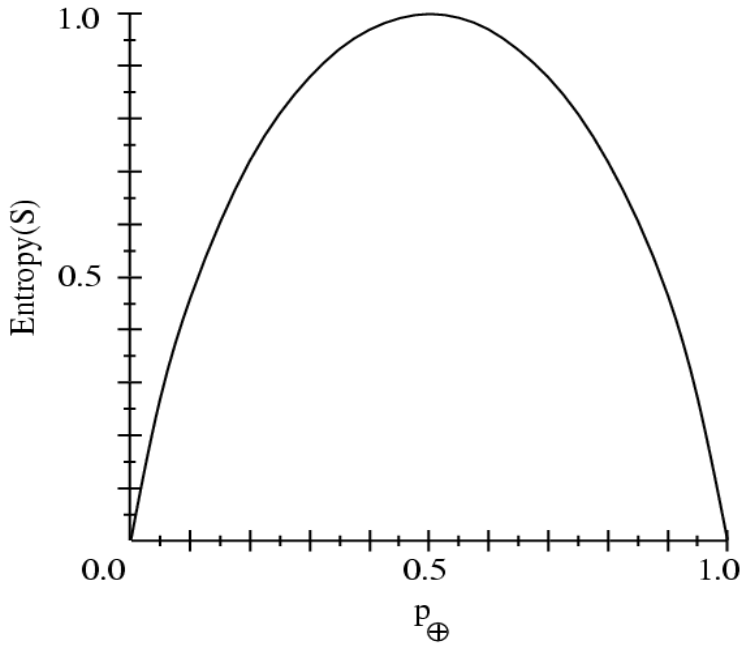
- Many different frameworks for choosing BEST have been proposed!
- We will look at Entropy Gain.

Entropy

- $p_\oplus$ is the proportion of positive examples in S
- $p_\ominus$ is the proportion of negative examples in S
- Entropy measures the impurity of S

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

Entropy



- S is a sample of training examples

Entropy is like a measure of impurity…

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:

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

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

Entropy

- Entropy measures the randomness/uncertainty in the data
- Let's consider a set S of examples with C many classes. Entropy of this set:

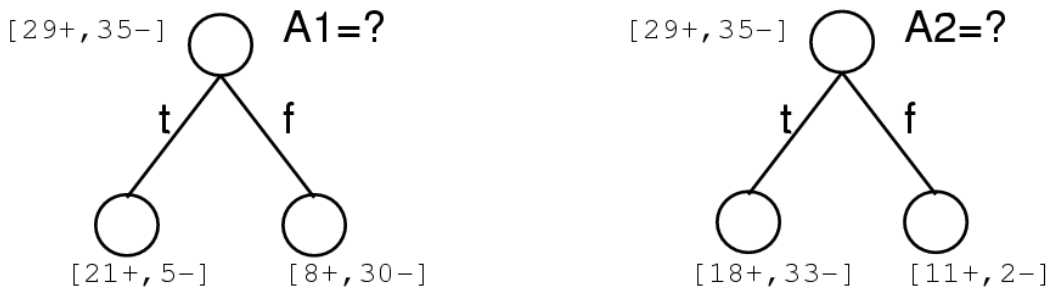
$$H(S) = -\sum_{c \in C} p_c \log_2 p_c$$

- p_c is the probability that an element of S belongs to class c
  - .. basically, the fraction of elements of S belonging to class c
- Intuition: Entropy is a measure of the "degree of surprise"
  - Some dominant classes $\Longrightarrow$ small entropy (less uncertainty)
  - Equiprobable classes $\Longrightarrow$ high entropy (more uncertainty)
- Entropy denotes the average number of bits needed to encode S

Information Gain

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

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



Information Gain

- Let's assume each element of S consists of a set of features
- Information Gain (IG) on a feature F

$$IG(S, F) = H(S) - \sum_{f \in F} \frac{|S_f|}{|S|} H(S_f)$$

- S_f number of elements of S with feature F having *value* f
- $IG(S, F)$ measures the increase in our certainty about S once we know the value of F
- $IG(S, F)$ denotes the number of bits saved while encoding S once we know the value of the feature F

Computing Information Gain

- Let's begin with the root node of the DT and compute IG of each feature
- Consider feature "wind" $\in$ {weak,strong} and its IG w.r.t. the root node

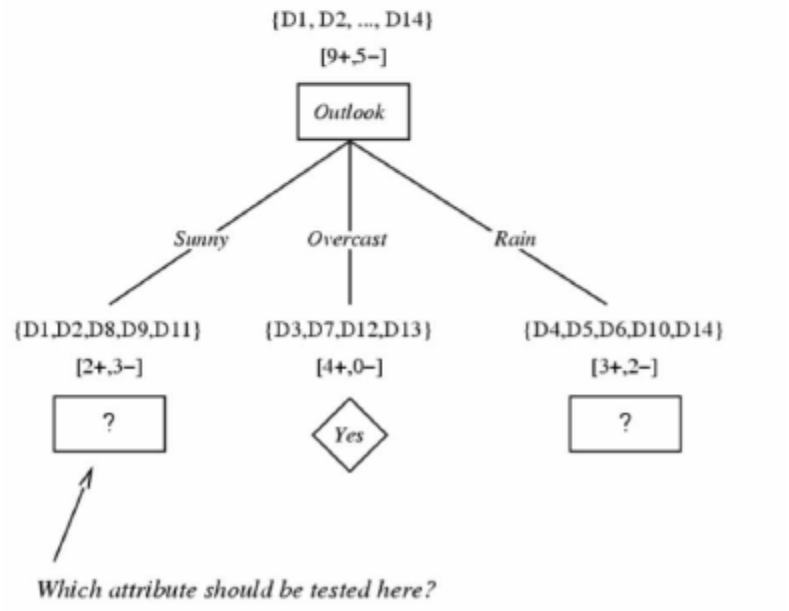
day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain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

- Root node: $S = [9+, 5-]$ (all training data: 9 play, 5 no-play)
- Entropy: $H(S) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.94$
- $S_{weak} = [6+, 2-] \Longrightarrow H(S_{weak}) = 0.811$
- $S_{strong} = [3+, 3-] \Longrightarrow H(S_{strong}) = 1$

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

Choosing the most informative feature

- At the root node, the information gains are:
  - $IG(S, \text{wind}) = 0.048$ (we already saw)
  - $IG(S, \text{outlook}) = 0.246$
  - $IG(S, \text{humidity}) = 0.151$
  - $IG(S, \text{temperature}) = 0.029$
- "outlook" has the maximum $IG \Longrightarrow$ chosen as the root node
- Growing the tree:
  - Iteratively select the feature with the highest information gain for each child of the previous node



Metric	Entropy	Information Gain
Definition	Measures the impurity of a node based on class labels	Measures the reduction in impurity achieved by splitting a node based on a certain feature
Calculation	$H(S) = -\Sigma\ p(x)\log_2 p(x)$	$IG(S, A) = H(S) - \Sigma\ p(x)H(x)$
	where S is the current node, A is a feature,	where S is the current node, A is a feature, p(x) is the proportion of instances of class x, and H is the entropy of a node or subset of instances
Use	Chooses the feature that minimizes entropy to split node	Chooses the feature that maximizes information gain to split node
Advantages	Works well when the class distribution is balanced	Works well when the class distribution is imbalanced
	Computationally efficient	Can handle continuous features and missing values
Disadvantages	Biased towards attributes with many categories	Prone to overfitting with noisy data
	Inefficient to handle continuous features and missing values	Biased towards features with many values

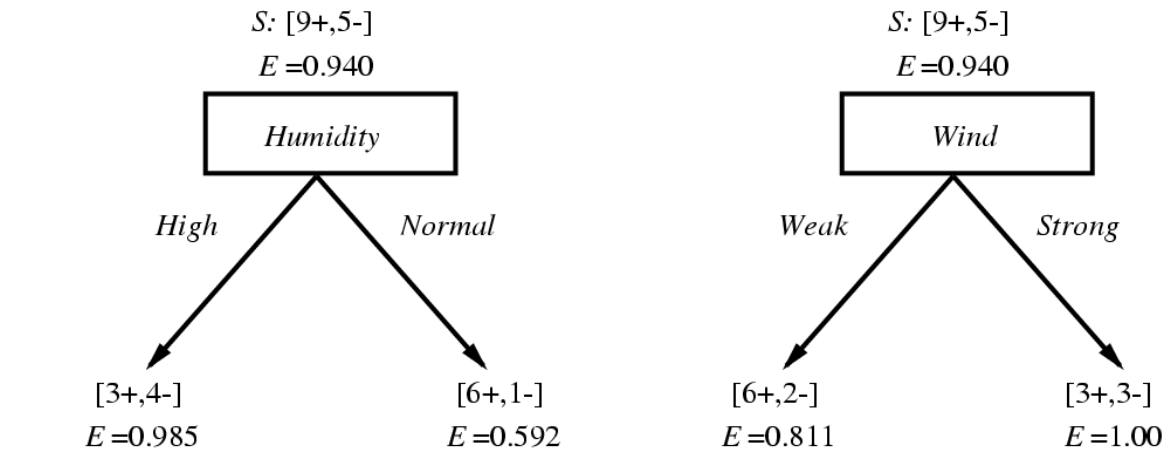
Training Example

PlayTennis: 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

Selecting the Next Attribute

Which attribute is the best classifier?

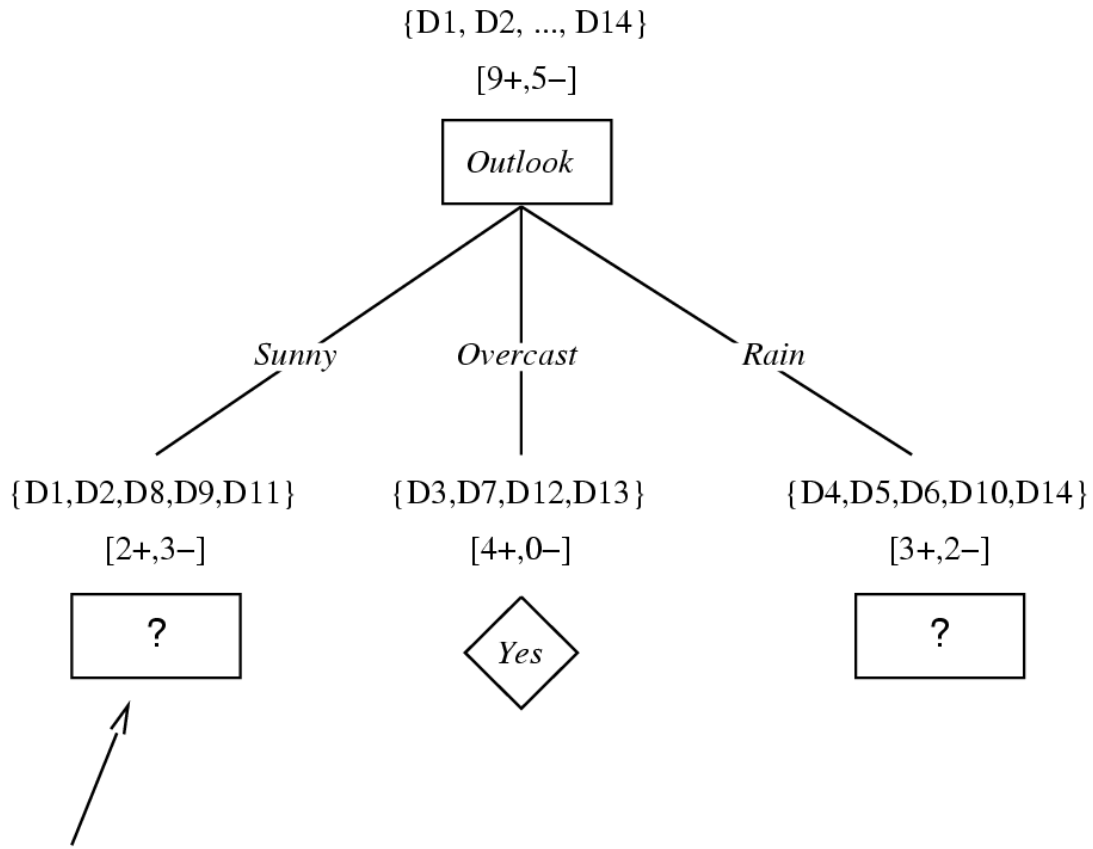


Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048



Which attribute should be tested here?

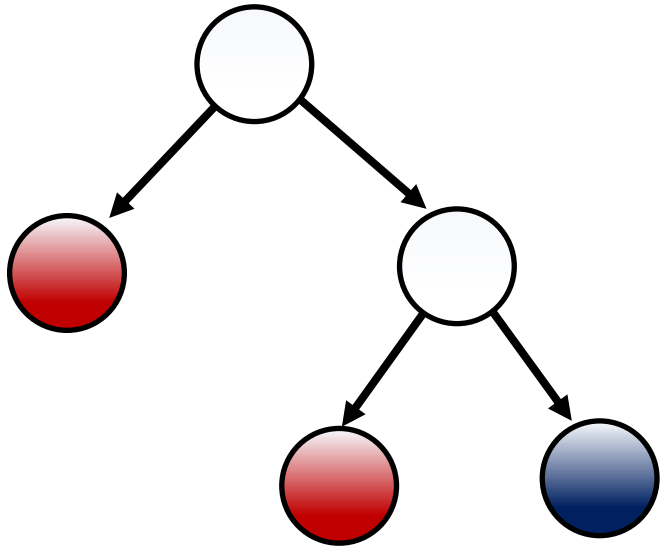
S_{sunny} = {D1,D2,D8,D9,D11}

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

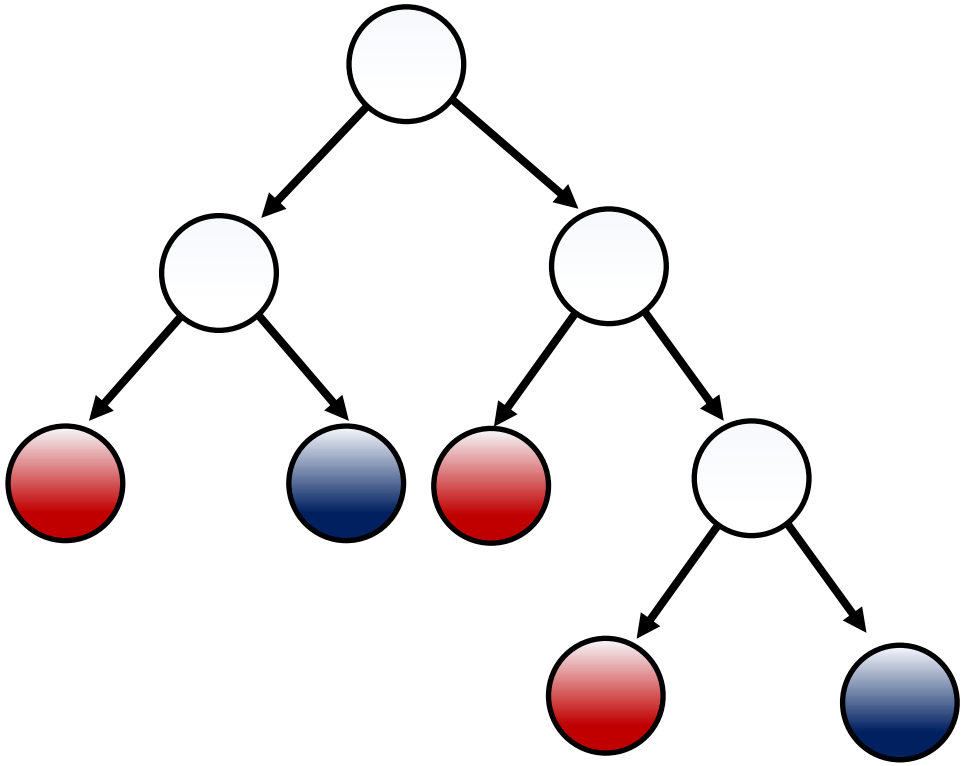
Gain (S_{sunny} , *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

Strengths of Decision Trees



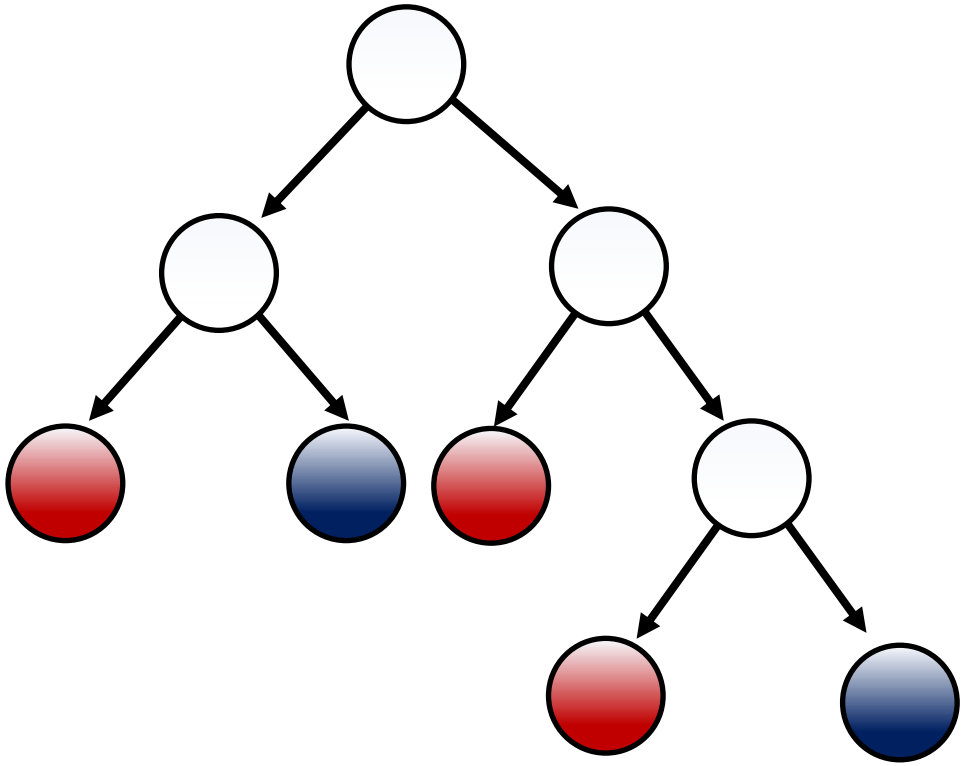
- Easy to interpret and implement—"if … then … else" logic
- Handle any data category—binary, ordinal, continuous
- No preprocessing or scaling required

Decision Trees are High Variance



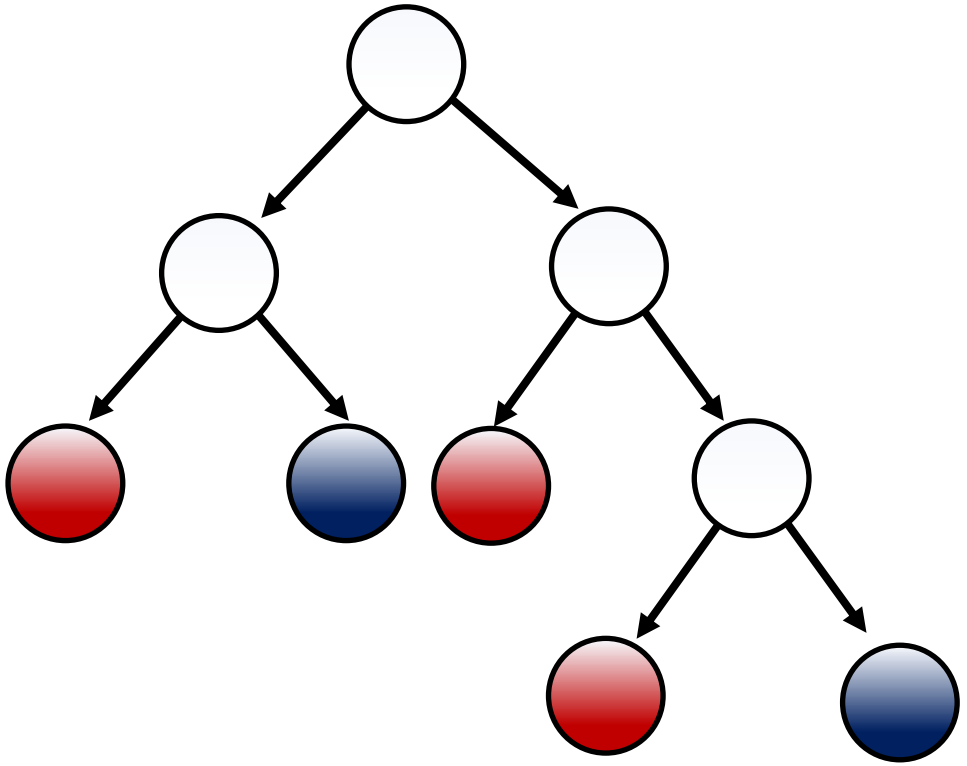
- Problem decision trees tend to overfit

Decision Trees are High Variance



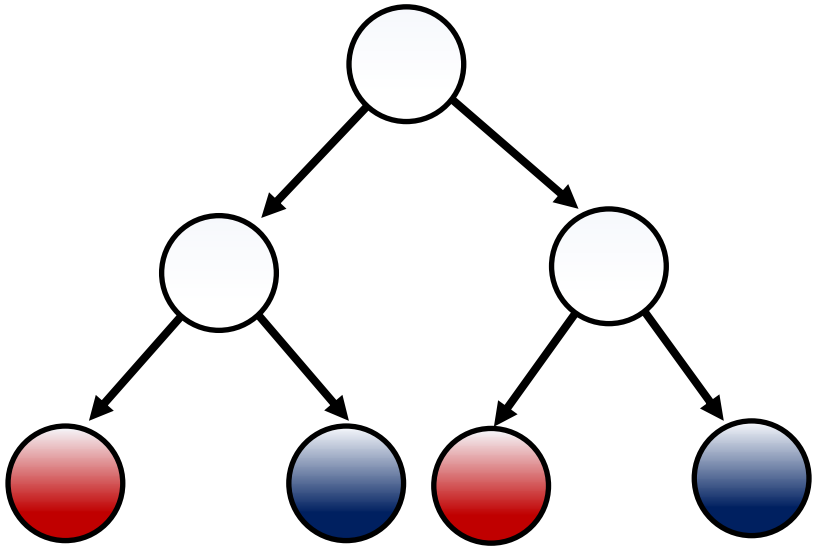
- Problem decision trees tend to overfit
- Small changes in data greatly affect prediction--high variance

Decision Trees are High Variance



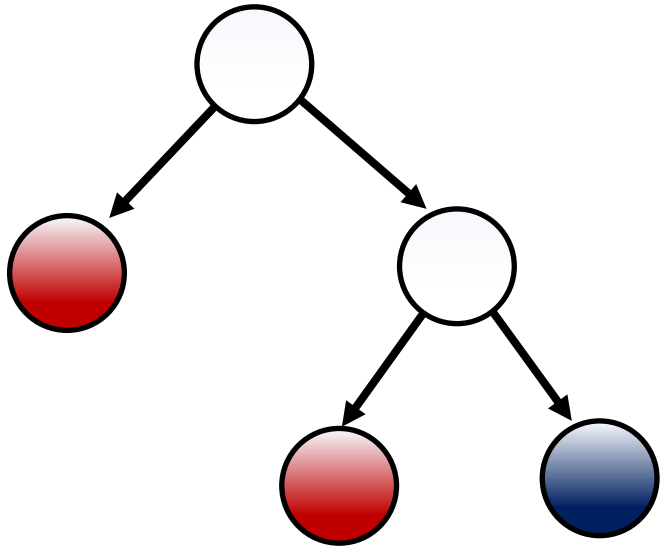
- Problem decision trees tend to overfit
- Small changes in data greatly affect prediction--high variance
- Solution Prune trees

Pruning Decision Trees



- Problem decision trees tend to overfit
- Small changes in data greatly affect prediction--high variance
- Solution Prune trees

Pruning Decision Trees



- Problem decision trees tend to overfit
- Small changes in data greatly affect prediction--high variance
- Solution Prune trees

Random forest algorithm

Random forest algorithm

- Random forest combines multiple decision trees to produce a single result, making it a popular machine learning algorithm.
- It can handle both classification and regression problems, and its ease of use and flexibility have contributed to its widespread adoption.

Random forest algorithm

- The random forest algorithm combines bagging and feature randomness to create an uncorrelated forest of decision trees, which reduces overfitting, bias, and overall variance in predictions.
- Feature randomness generates a random subset of features, ensuring low correlation among decision trees, which is a key difference between decision trees and random forests.
- In contrast to decision trees, which consider all possible feature splits, random forests only select a subset of features, making them more efficient and less prone to overfitting.
- The random forest algorithm can account for potential variability in data and produce more precise predictions by considering a wider range of questions than any single decision tree.
- By reducing correlation among decision trees, the random forest algorithm can provide more reliable predictions than other methods that rely on a single decision tree.

Ensemble methods

- Ensemble learning methods combine multiple classifiers to identify the most popular result.
- Bagging is a well-known ensemble method that involves selecting a random sample of data and training multiple independent models.
- The bagging method can be used for both regression and classification tasks, and the average or majority of predictions can yield a more accurate estimate.
- Boosting is another commonly used ensemble method.
- Ensemble methods such as bagging are used to reduce variance within a noisy dataset.

How it works

- Random forest algorithms have three main hyperparameters node size, the number of trees, and the number of features sampled.
- The random forest classifier can be used to solve for both regression and classification problems.
- Each tree in the random forest ensemble is made up of a data sample drawn from a training set with replacement, and feature bagging is used to reduce correlation among decision trees.
- The prediction is determined differently for regression and classification tasks, with averaging or majority vote used, respectively.
- The out-of-bag (oob) sample is set aside as test data and used for cross-validation to finalize the prediction.

Random Forests have several advantages as a machine learning algorithm

- They are highly accurate and have been shown to outperform many other algorithms in a variety of applications.
- They are robust to noise and overfitting, which means they can handle noisy or incomplete data without sacrificing accuracy.
- They can handle both classification and regression problems.
- They are easy to use and require minimal parameter tuning.
- They can handle high-dimensional data with many features.
- They provide measures of variable importance, which can be useful for feature selection and understanding the underlying data.

Source

- Greg Grudic - Thomas G. Dietterich and Tom Mitchell materials.
- Ansrew W.Moore, CMW,
- IntelAi Academy
- Random forest - IBM