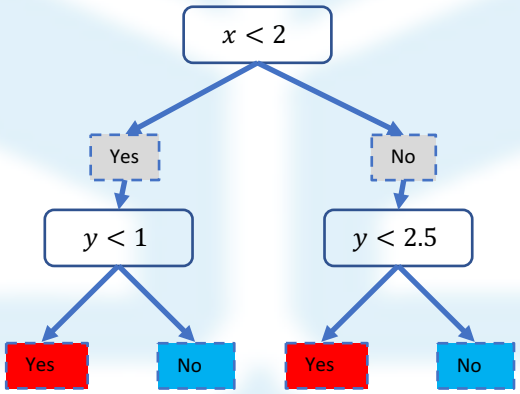
به نام او، به یاد او، برای او

Machine Learning

Decision trees



Dr. *Ali Valinejad*

valinejad.ir
valinejad@umz.ac.ir

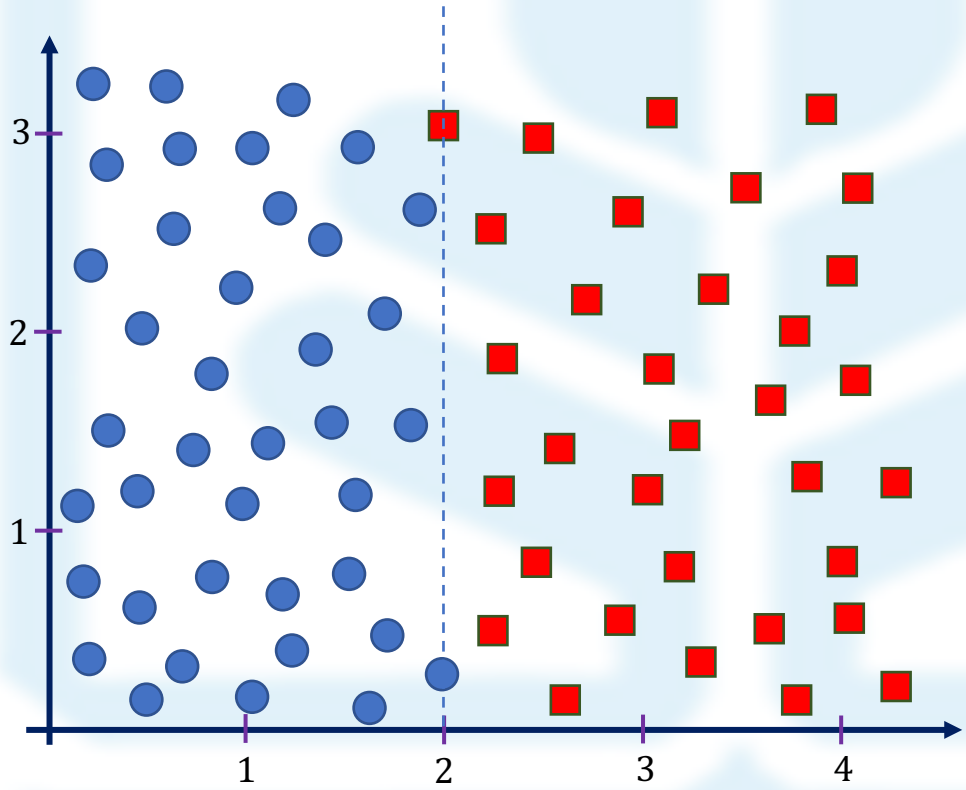
What is DecisionTrees?

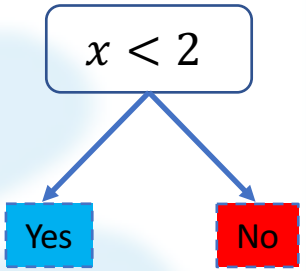
- Decision tree builds ***classification*** or ***regression*** models in the form of a ***tree structure***.
- It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.
- The final result is a tree with **decision nodes** and **leaf nodes**.
- A decision node has two or more branches.
- Leaf node represents a classification or decision.
- The topmost decision node in a tree which corresponds to the best predictor called **root node**.
- Decision trees can handle both ***categorical*** and ***numerical*** data.

A decision tree has two kinds of *nodes*:

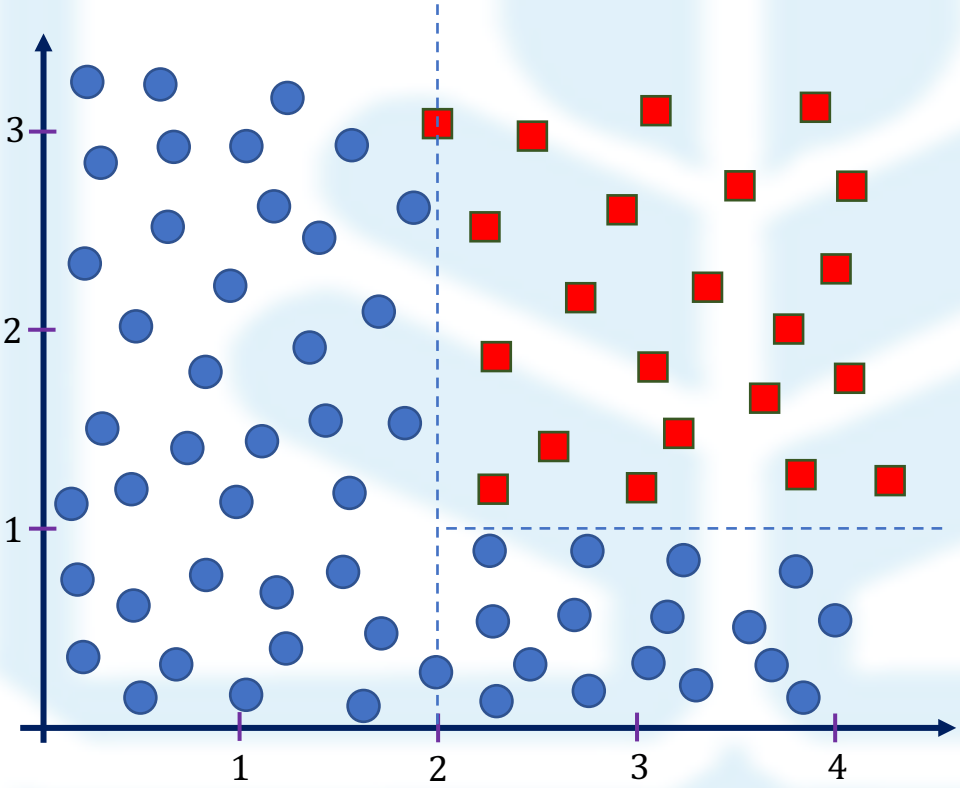
1. Each ***leaf node*** has a *class label*, determined by majority vote of training examples reaching that leaf.
2. Each ***internal node*** is a *question on features*. It branches out according to the answers.

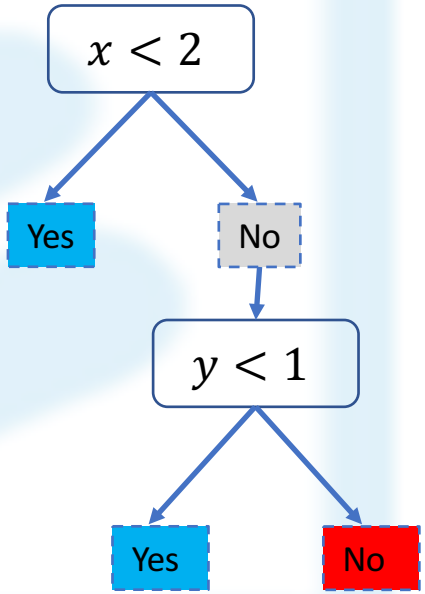
Decision Trees



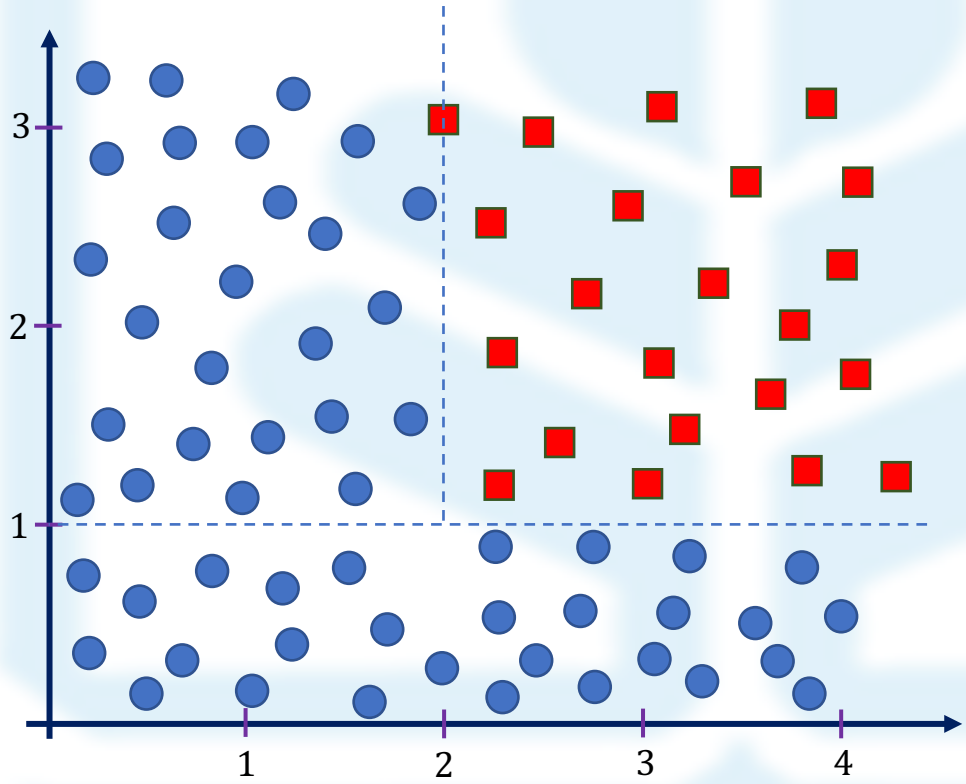


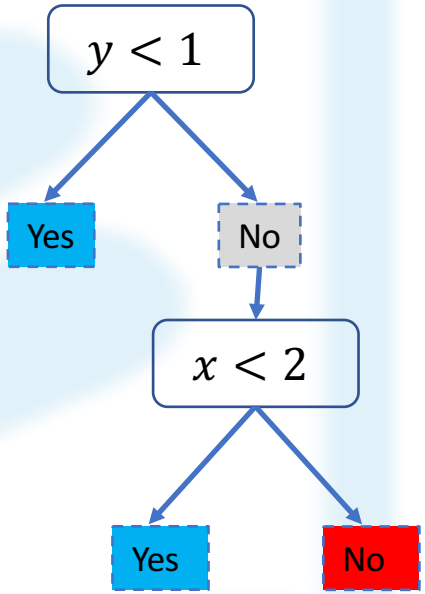
Decision Trees



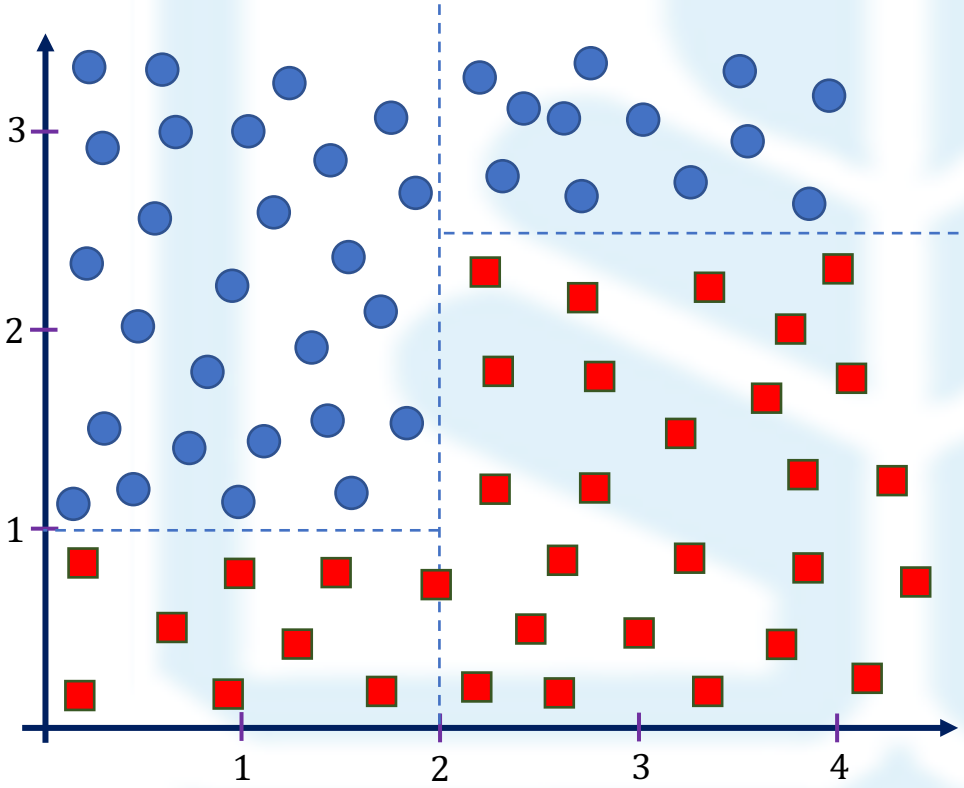


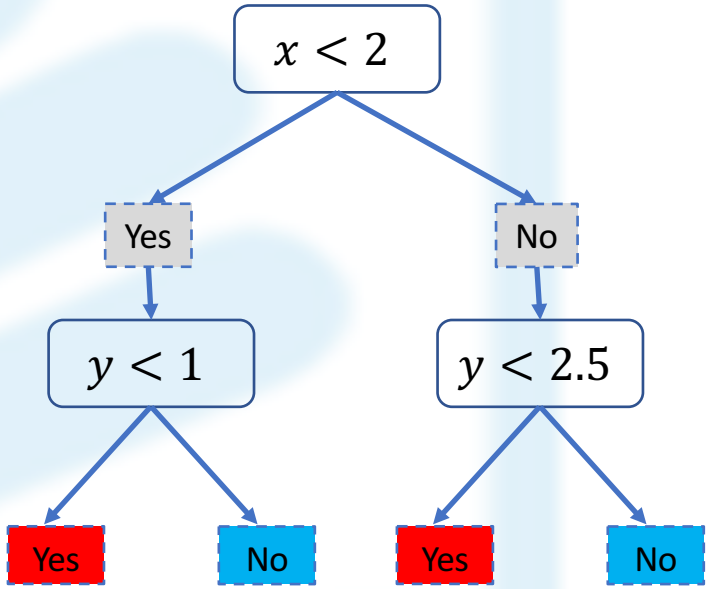
DecisionTrees



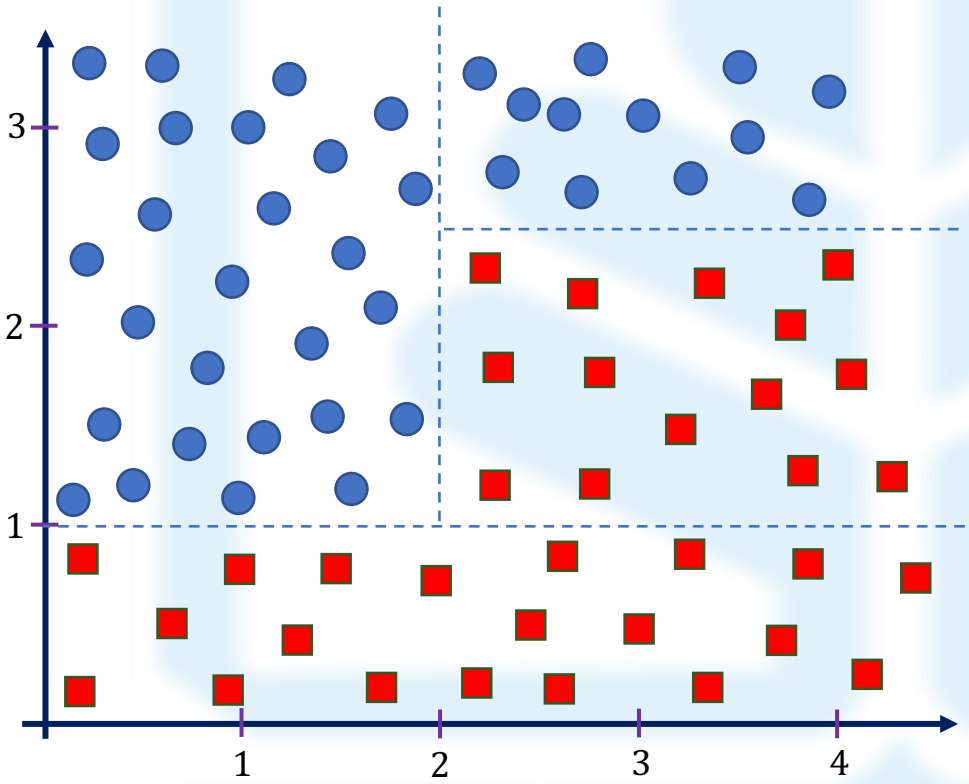


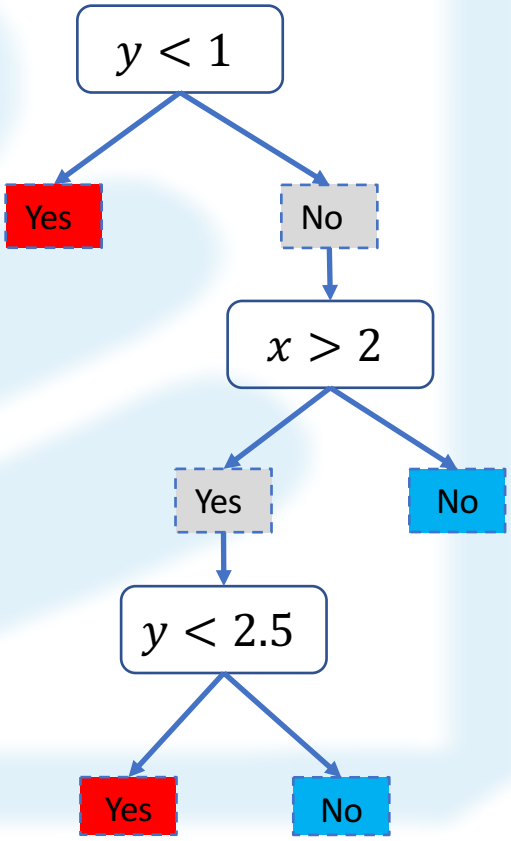
Decision Trees





Decision Trees





Decision tree algorithm

def buildtree (examples, questions, default):
/* ***examples***: *a list of training examples*
questions: *a set of candidate questions, e.g., "what's the value of feature x_i?"*
default: *default label prediction, e.g., over-all majority vote* */

if empty(examples):
return(default)

if (examples have same label y):
return(y)

if empty(questions)
return(majority vote in examples)

q = best_question(examples, questions)
Let there be *n* answers to q
– Create and return an internal node with *n* children
– The *i*th child is built by calling
buildtree({example|q=*i*th answer}, questions\{q}, default)

Decision tree algorithms:

The best question:

What do we want?

pure leaf nodes, i.e. all examples having (almost) the same y.

A good question → a split that results in pure child nodes

How do we measure the degree of purity induced by a question?

Decision tree algorithms:

- Iterative Dichotomiser 3 (ID3) algorithm
- C4.5
- Classification and Regression Tree (CART)

Decision tree algorithms:
ID3 algorithm

ID3 by J. R. Quinlan, employs a ***top-down***, ***greedy search*** through the space of possible branches with no backtracking.

How dose ID construct a decision tree?

How dose ID3 measure the degree of purity of a node induced by a question?

ID3 uses ***Entropy*** and ***Information Gain*** (a..k.a mutual information) to construct a decision tree.

In the context of training Decision Trees, ***Entropy*** can be roughly thought of as **how much variance the data has**.

https://victorzhou.com/blog/information-gain/

Decision tree algorithms:
ID3 algorithm

ID3 uses ***Entropy*** and ***Information Gain*** to construct a decision tree.

What is Entropy?

- A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous).
- ID3 algorithm uses **entropy** to calculate the homogeneity of a sample.
  If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.
- The entropy is a measure of the uncertainty(impurity) associated with a random variable.
- As uncertainty and or randomness increases for a result set so does the entropy.
- Values range from 0-1 to represent the entropy of information.

Decision tree algorithms:
ID3 algorithm

How can compute Entropy?

- **At the current node, there are $n = n_1 + n_2 + \cdots + n_k$ examples:**
  - n_1 examples have label y_1
  - n_2 examples have label y_2
  - ...
  - n_k examples have label y_k

- **What's the impurity of the node?**

- **Turn it into a game:**
  If I put these examples in a bag, and grab one at random, what is the probability the example has label y_k?

Decision tree algorithms:
ID3 algorithm

How can compute Entropy?

- **Probability estimated from samples in the current node:**
  - with probability $p_1 = n_1/n$ the example has label y_1
  - with probability $p_2 = n_2/n$ the example has label y_2
  - …
  - with probability $p_k = n_k/n$ the example has label y_k
- $p_1 + p_2 + \cdots + p_k = 1$
- **The "*outcome*" of the draw is a random variable y with probability $(p_1, p_2, \ldots, p_k)$**
- **What's the impurity of the node? → what's the uncertainty of y in a random drawing?**

S = Set of the examples in the current node.
Entropy measures the impurity of *S*

Decision tree algorithms:
ID3 algorithm

How can compute Entropy?

$$H(y) = \sum_{i=1}^{k} -p(y = y_i) \log_2 p(y = y_i) = \sum_{i=1}^{k} -p_i \log_2 p_i$$

Interpretation:
The number of **yes**/**no** questions (bits) needed on average to pin down the value of y in a random drawing(using an optimal, shortest-length code).

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

$H(y) = ?$

$p(y = Red) = \frac{3}{6}$

$p(y = Blue) = \frac{2}{6}$

$p(y = green) = \frac{1}{6}$

$H = -\frac{3}{6}\log_2\frac{3}{6} - \frac{2}{6}\log_2\frac{3}{6} - \frac{1}{6}\log_2\frac{1}{6} = 1.46$

$H(y) = ?$

$p(y = Red) = \frac{2}{3}$

$p(y = Blue) = \frac{1}{3}$

$H = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} \approx 0.92$

$H(y) = ?$

$p(y = Red) = \frac{1}{3}$

$p(y = Blue) = \frac{1}{3}$

$p(y = Yellow) = \frac{1}{3}$

$H = -\frac{1}{3}\log_2\frac{1}{3} - \frac{1}{3}\log_2\frac{1}{3} - \frac{1}{3}\log_2\frac{1}{3} \approx 1.58$

$H(y) = ?$

$p(y = Red) = 0.5$

$p(y = Blue) = 0.5$

$H = -0.5\log_2 0.5 - 0.5\log_2 0.5 = 1$

Decision tree algorithms:
ID3 algorithm

Conditional entropy:

y: label,
***x*:** a question (e.g., a feature),
***v*:** an answer to the question

$$H(y|x=v) = \sum_{i=1}^{k} p(y=y_i|x=v)\log_2 p(y=y_i|x=v)$$

$$H(y|x) = \sum_{v:\ values\ of\ x} p(x=v)\log_2 p(y|x=v)$$

Decision tree algorithms:

ID3 algorithm

To build a decision tree, we need to calculate two types of ***entropy*** using ***frequency tables*** as follows:

a) Entropy using the frequency table of one feature.

b) Entropy using the frequency table of two features.

Building a Decision Tree-Classification
ID3 algorithm

To build a decision tree, we need to calculate two types of ***entropy*** using ***frequency tables*** as follows:

a) Entropy using the frequency table of **one** feature.

frequency tables

Color	
#Red	#Blue
2	1

$$H(\text{color}) = H(2,1)$$
$$= H\left(\frac{2}{3},\frac{1}{3}\right)$$
$$= -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} \approx 0.92$$

Building a Decision Tree-Classification
ID3 algorithm

To build a decision tree, we need to calculate two types of ***entropy*** using ***frequency tables*** as follows:

b) Entropy using the frequency table of **two** features.

frequency tables

		color		
		Red	Blue	
shape	Circle	2	1	3
	Square	1	2	3
				6

$$H(\text{color}, \text{shape}) = p(\text{Circle})H(2,1) + p(\text{Square})H(1,2)$$

$$= -\frac{3}{6}H(2,1) - \frac{3}{6}H(1,2)$$

Decision tree algorithms:
ID3 algorithm

Information gain:

- Information gain, or mutual information
  $I(y; x) = H(y) - H(y|x)$
- Choose question (feature) x which maximizes $I(y; x)$

- The information gain is based on the decrease in entropy after a dataset is split on an feature.
- Constructing a decision tree is all about finding feature that returns the highest information gain (i.e., the most homogeneous branches).

Decision tree algorithms:
ID3 algorithm

Step 1:
- Calculate entropy of the target.

Step 2:
- The dataset is then split on the different features.
- The entropy for each branch is calculated.
- Then it is added proportionally, to get total entropy for the split.
- The resulting entropy is subtracted from the entropy before the split.
- The result is the ***Information Gain***, or ***decrease in entropy***.

Step 3:
- Choose attribute with the largest information gain as the decision node,
- Divide the dataset by its branches and repeat the same process on every branch.

Step 4:
- A branch with entropy of 0 is a leaf node.
- A branch with entropy more than 0 needs further splitting.

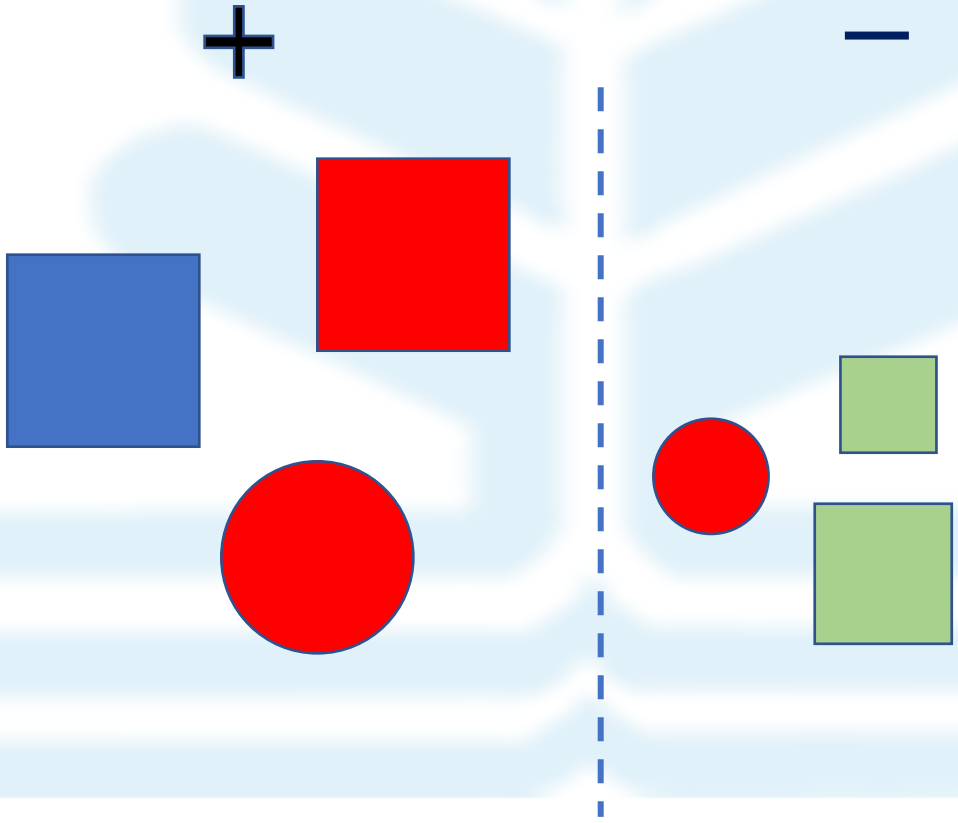
Step 5:
- The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

ID3 Algorithm

Example

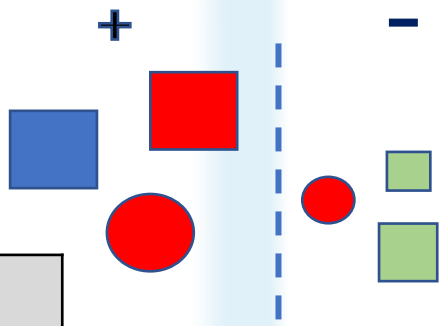
Features: color, shape, size

What's the best question at root?



ID3 Algorithm

Example



The training set

Example	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-

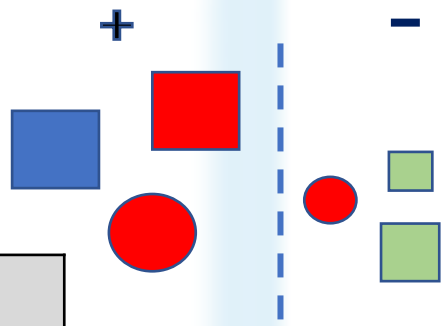
$H(\text{class}) = ?$

$H(\text{class}|\text{Color}) = ?$

ID3 Algorithm

Example

The training set



Example	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-

$$H(\text{class}) = H\left(\frac{3}{6}, \frac{3}{6}\right) = 1$$

$$H(\text{class}|\text{Color}) = \frac{3}{6} * H\left(\frac{2}{3}, \frac{1}{3}\right) + \frac{1}{6} * H\left(\frac{1}{1}, \frac{0}{1}\right) + \frac{2}{6} * H\left(\frac{0}{2}, \frac{2}{2}\right)$$

1 out of 6 is Blue

1 of the Blue are +

0 of the Blue is -

0 of the Green is +

3 out of 6 are Red

2 of the Red are +

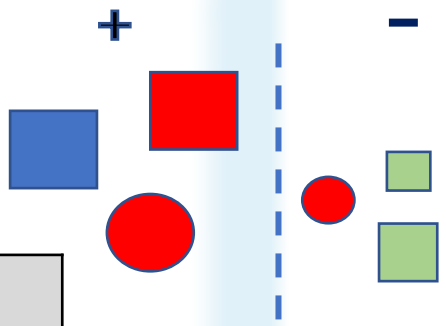
1 of the Red are -

2 out of 6 are Green

2 of the Green are -

ID3 Algorithm

Example



The training set

Example	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-

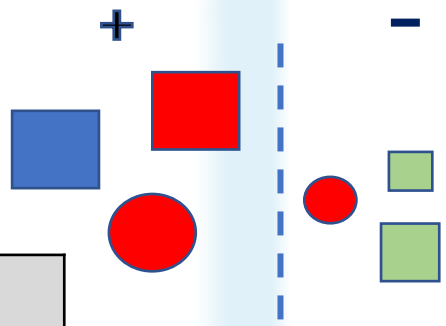
$$H(\text{class}) = H\left(\frac{3}{6}, \frac{3}{6}\right) = 1$$

$$H(\text{class}|\text{Color}) = \frac{3}{6} * H\left(\frac{2}{3}, \frac{1}{3}\right) + \frac{1}{6} * H(1, 0) + \frac{2}{6} * H(0, 1)$$

$$I(\text{class};\text{Color}) = H(\text{class}) - H(\text{class}|\text{Color}) = 0.54 \text{ bit}$$

ID3 Algorithm

Example



The training set

Example	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-

$$H(\text{class}) = H\left(\frac{3}{6}, \frac{3}{6}\right) = 1$$

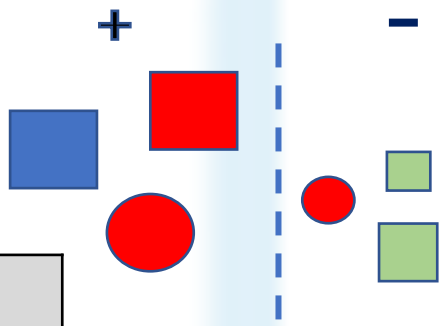
$$H(\text{class}|\text{Shape}) = \frac{4}{6} * H\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{2}{6} * H\left(\frac{1}{2}, \frac{1}{2}\right)$$

Shape tells us nothing about the class!

$$I(\text{class};\text{Shape}) = H(\text{class}) - H(\text{class}|\text{Shape}) = 0 \text{ bit}$$

ID3 Algorithm

Example



The training set

Example	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-

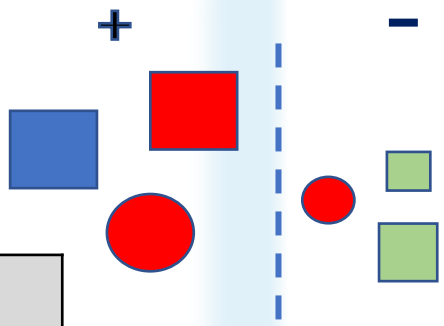
$$H(\text{class}) = H\left(\frac{3}{6}, \frac{3}{6}\right) = 1$$

$$H(\text{class}|\text{Size}) = \frac{4}{6} * H\left(\frac{3}{4}, \frac{1}{4}\right) + \frac{2}{6} * H(0, 1)$$

$$I(\text{class};\text{Size}) = H(\text{class}) - H(\text{class}|\text{Size}) = 0.46 \text{ bit}$$

ID3 Algorithm

Example



The training set

Example	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-

$I(\text{class};\text{Color}) = H(\text{class}) - H(\text{class}|\text{Color}) = 0.54$ bit

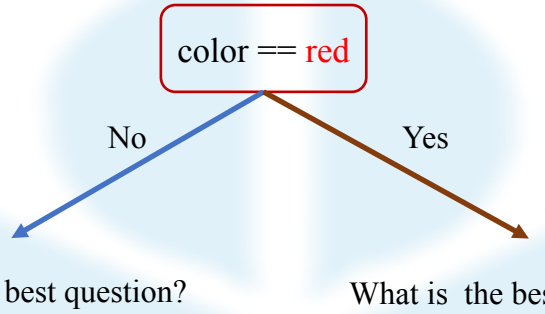
$I(\text{class};\text{Shape}) = H(\text{class}) - H(\text{class}|\text{Shape}) = 0$ bit

$I(\text{class};\text{Size}) = H(\text{class}) - H(\text{class}|\text{Size}) = 0.46$ bit

We select ***color*** as the question at root (e.g color == red)

ID3 Algorithm

Example



Example	Color	Shape	Size	Class
2	Blue	Square	Big	+
5	Green	Square	Small	-
6	Green	Square	Big	-

$I(\text{class;Color}) = ?$ bit

~~$I(\text{class;Shape}) = ?$ bit~~

$I(\text{class;Size}) = ?$bit

Color == Blue

Example	Color	Shape	Size	Class
1	Red	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-

~~$I(\text{class;Color}) = ?$ bit~~

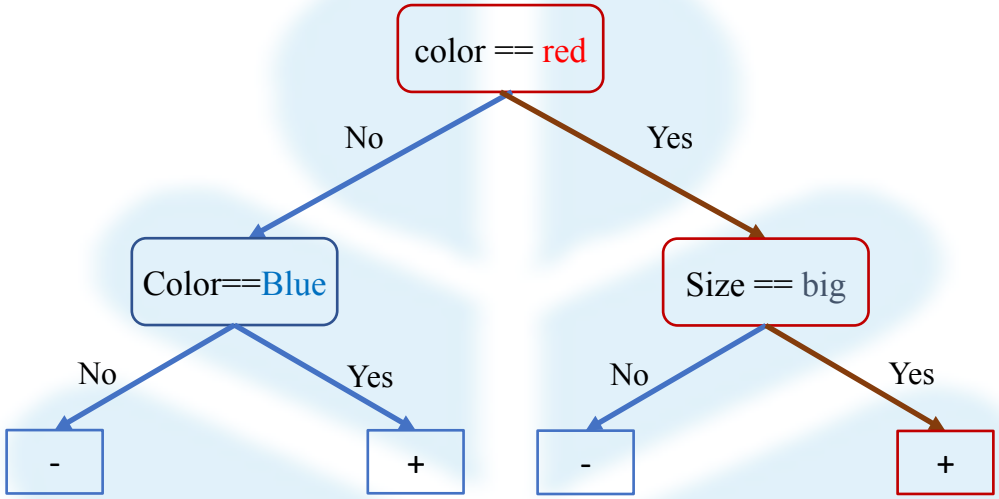
$I(\text{class;Shape}) = ?$ bit

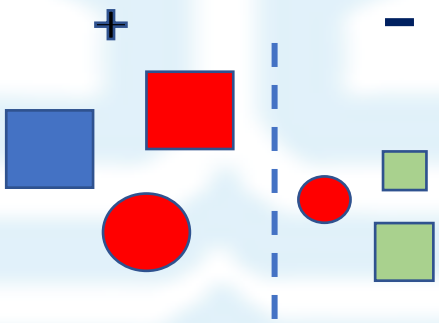
$I(\text{class;Size}) = ?$bit

Size == big

ID3 Algorithm

Example

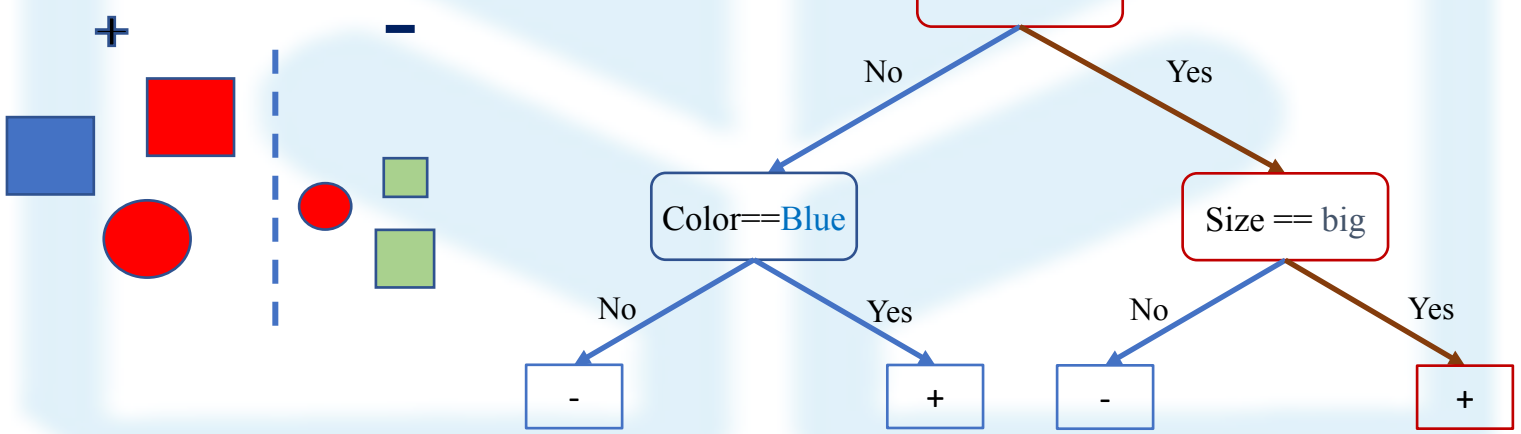




Tree ⟶ Rules

❖ Each ***path***, from the root to a leaf, corresponds to a ***rule*** where all of the decisions leading to the leaf define the antecedent to the rule, and the consequent is the classification at the leaf node

For example, from the tree in the color/shape/size example, we could generate the rule:
Rule1: **if** color == red & size == big **then** class = +
Rule2: **if** color == red & size != big **then** class = -
Rule3: **if** color != red & Color == Blue **then** class = +
Rule4: **if** color != red & Color != Blue **then** class = -



Shape tells us nothing about the class!

Avoid overfitting: pruning

(a greedy version)

Pruning with a tuning set

1. Randomly split data into TRAIN and TUNE, say 70% and 30%
2. Build a full tree using only TRAIN
3. Prune the tree down on the TUNE set.

def Prune(tree T, TUNE set):

1. Compute T's accuracy on TUNE, call it $A(T)$.
2. **for** every internal node ***node*** in T :
   - New tree T_{node} = copy of T, but prune (delete) the subtree under ***node***.
   - ***node*** becomes a leaf node in T_{node}. The label is the majority vote of TRAIN examples reaching ***node***.
   - A(T_{node}) = T_{node}'s accuracy on TUNE
3. Let T* be the tree (among the T_{node}'s and T) with the largest A(). Set $T \leftarrow T^*$ /* prune */
4. Repeat from step 1 until no more improvement available.

Return T.

Real-valued features

What if some (or all) of the features $x_1, x_2, \ldots, x_n$ are real-valued?
Example: x_1 =height (in inches)

Idea:

1. branch on each possible numerical value. (fragments the training data and prone to overfitting)
2. use questions in the form of ($x_i > t$?), where t is a threshold. There are fast ways to try all(?) t.

$$H(y|x_i > t\,?) = p(x_i > t)H(y|x_i > t) + p(x_i \leq t)H(y|x_i \leq t)$$

$$I(y|x_i > t?) = H(y) - H(y|x_i > t\,?)$$