Machine Learning

Chapter 3: Decision Trees

Fall 2023

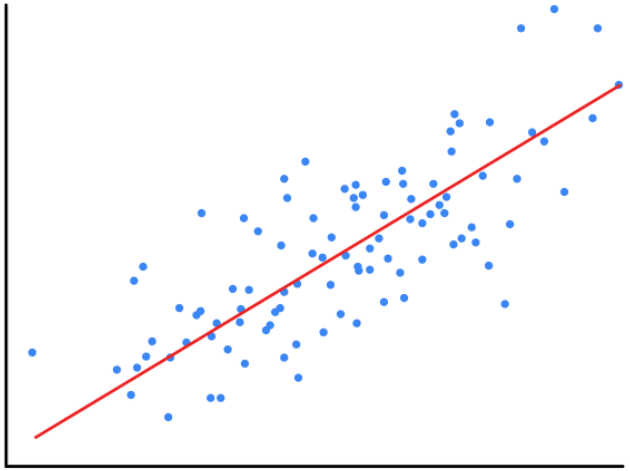
Instructor: Xiaodong Gu

What we have learned so far?

Linear model for regression

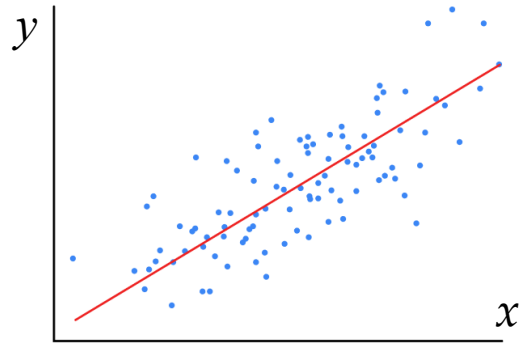
$$y = f(x) = \boldsymbol{w}^T x + w_0$$

characterize the relationship between one or more independent variables and a target variable



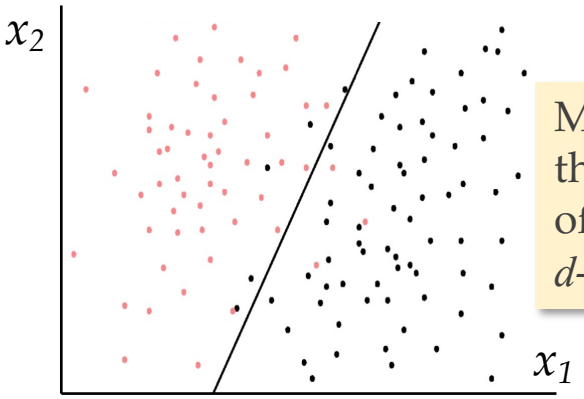
Regression vs. Classification

- Machine Learning
  - **Supervised Learning**
    - Regression (✓)
    - Classification
    - …
  - Unsupervised Learning
  - Reinforcement Learning



Model the data points spread in (d+1)-dim space.

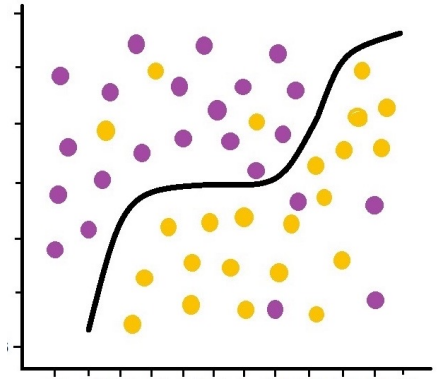
Regression: (predicts real-valued labels)

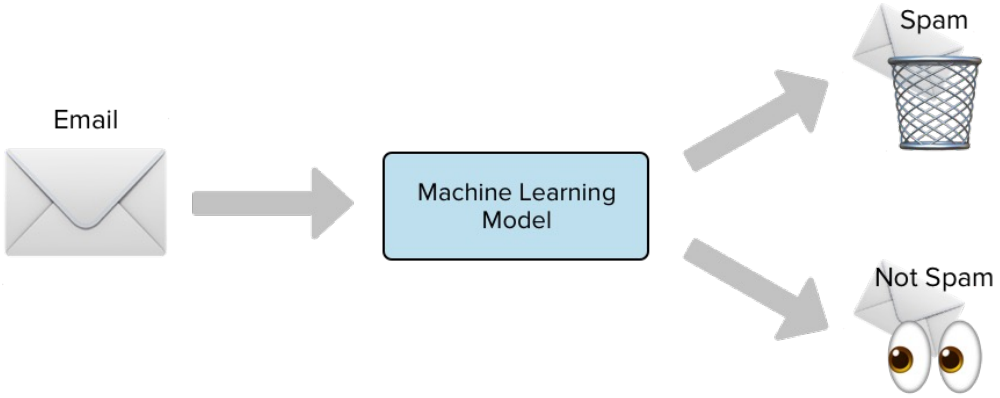


Model the boundaries that separate the data of different labels in *d*-dim space

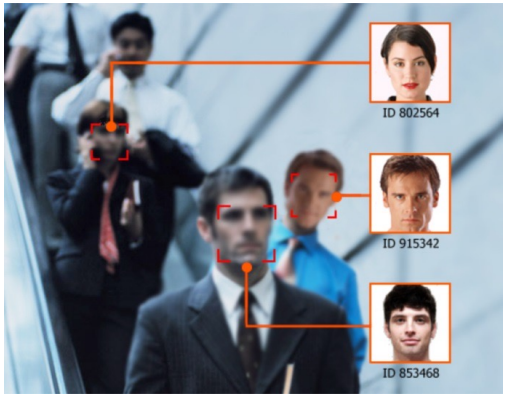
Classification: (predicts categorical labels)

Classification: The Core Task of Machine Learning





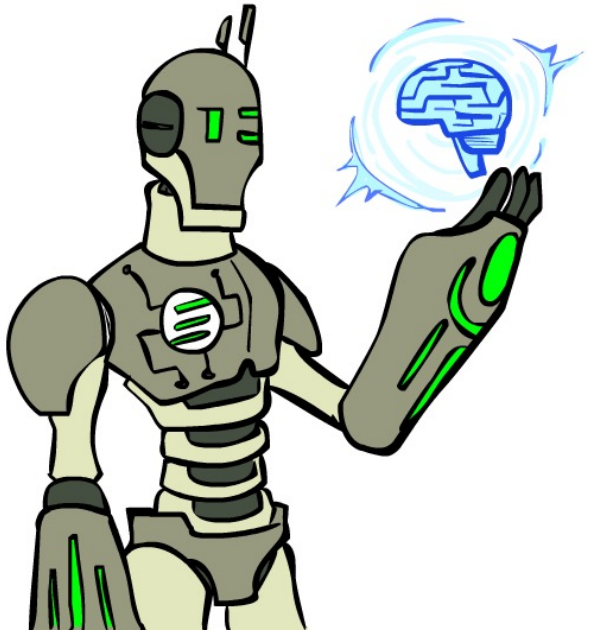
Sentiment analysis



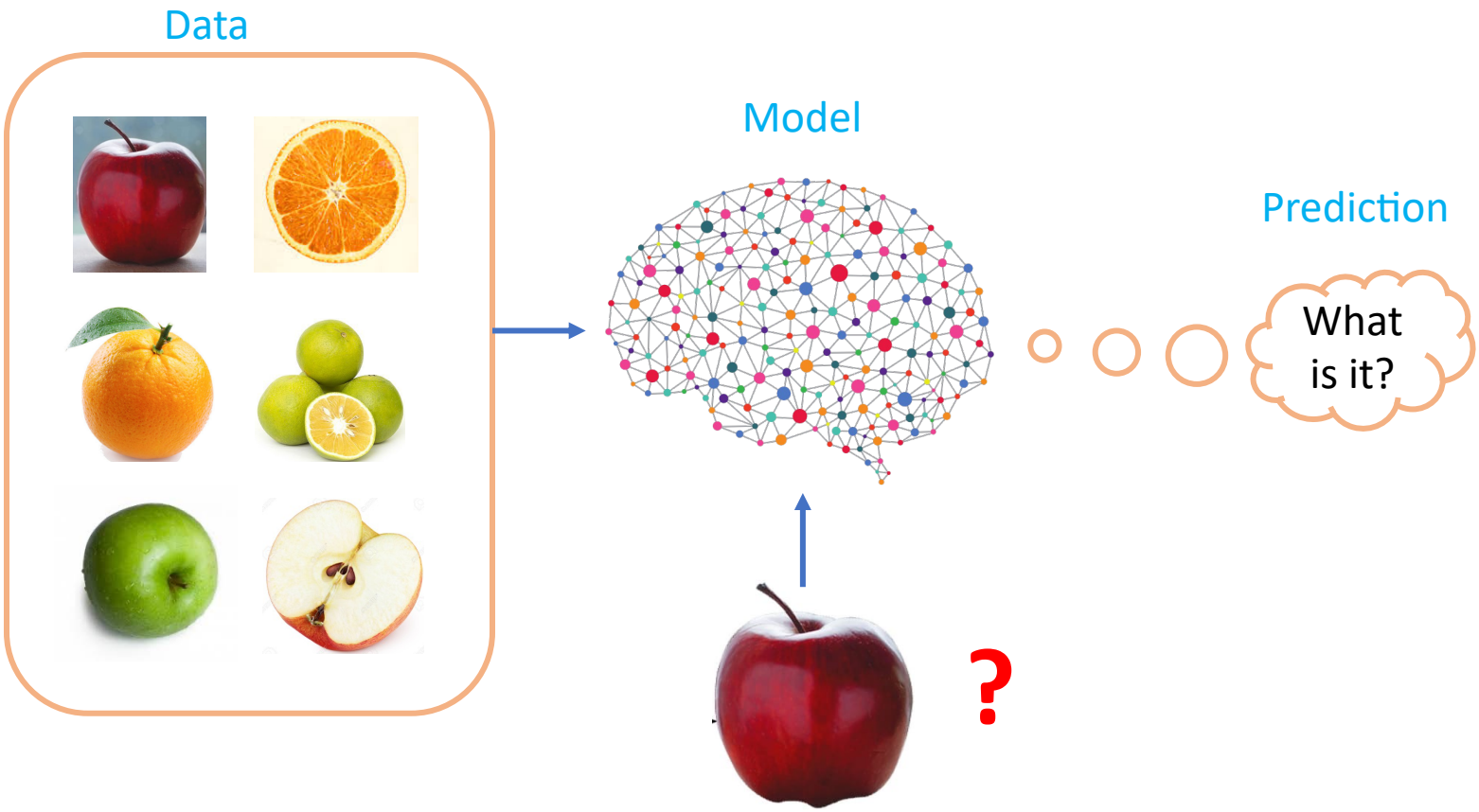
Today

Let's start from a simple family of classification approaches

- Decision Trees

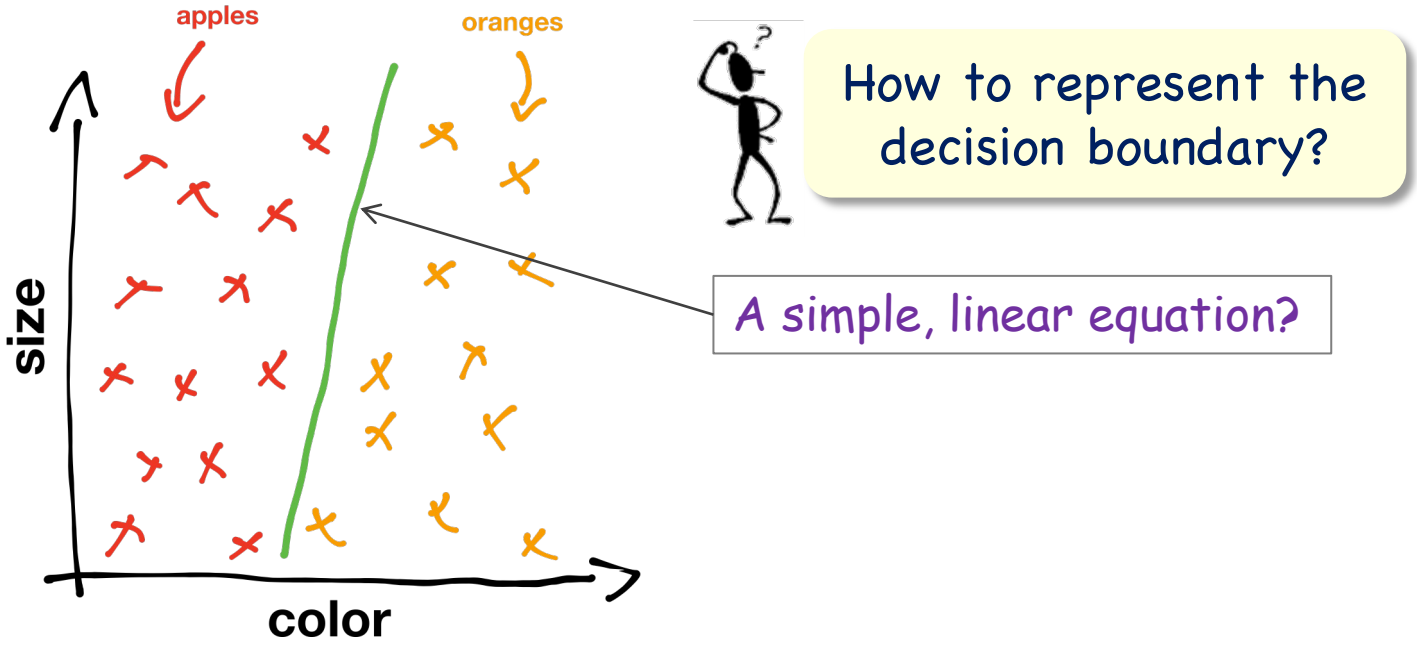


Review: Automatic Fruit Classification

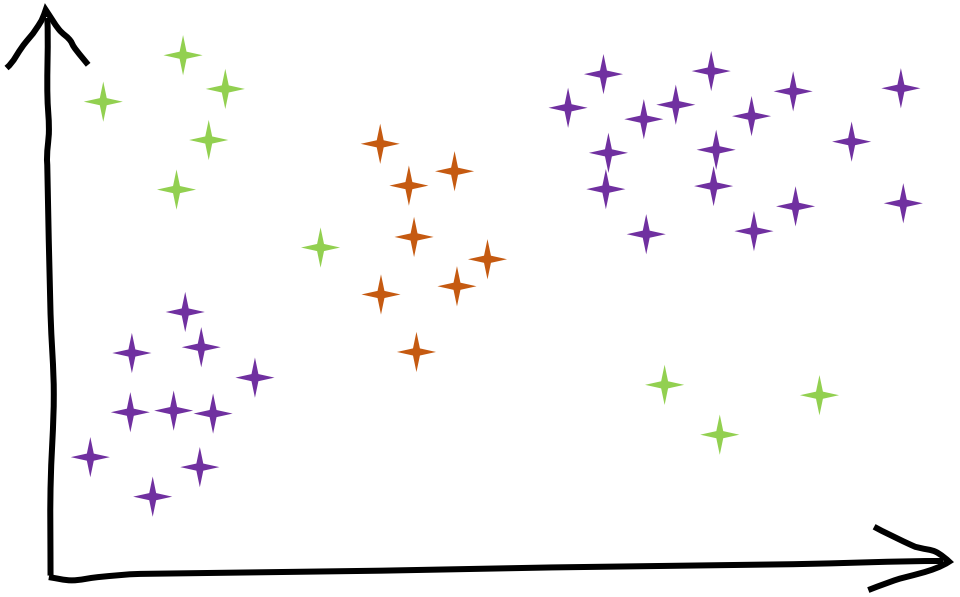


Review: Automatic Fruit Classification

Classification = **Partition** the feature space into 2 regions (one for each type of fruit) by a **decision boundary**.



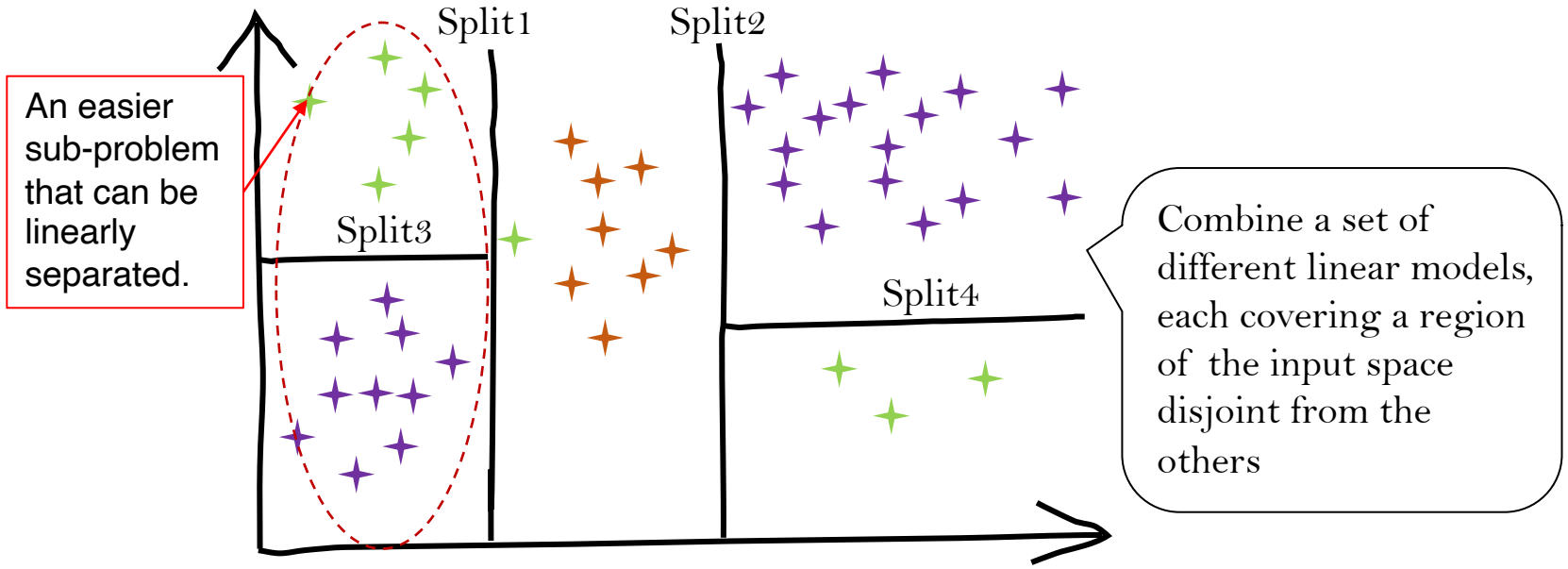
What if the data is distributed like this?



- nonlinear (can not be linearly separated)
- can be nominal (e.g., “male”, “female”)

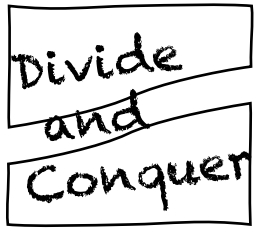
From Linear to Piecewise Linear

What if the data is distributed like this?

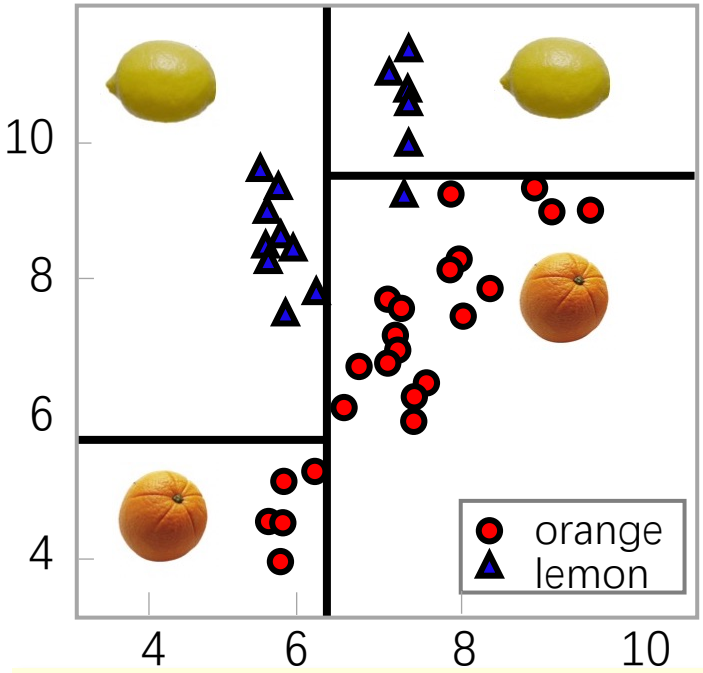


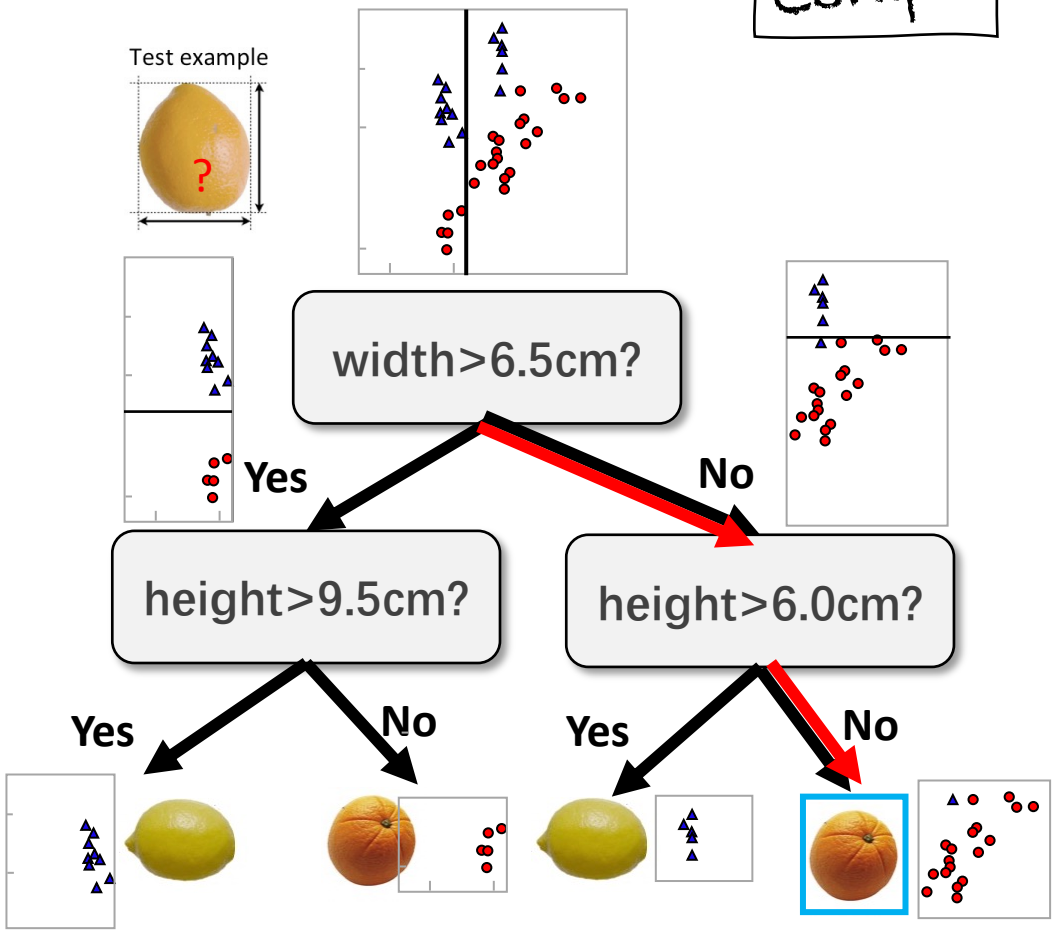
- nonlinear (can not be linearly separated)
- can be nominal (e.g., "male", "female")
- **but have** local-piecewise linearity **along axes**

Decision Tree: The Key Idea



- Divide and Conquer
  - split data attributes.
  - create if-then rules



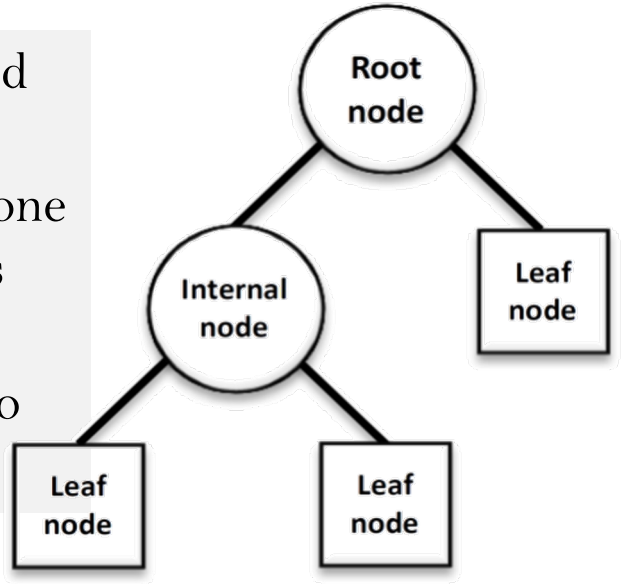


Decision Tree: classify data through a sequence of decision rules.

Model Structure

- A decision tree consists of three types of nodes:

1. A **root node** that has no incoming edges and zero or more outgoing edges
2. **Internal nodes**, each of which has exactly one incoming edge and multiple outgoing edges
3. **Leaf nodes** (or terminal nodes), each of which has exactly one incoming edge and no outgoing edges



▻ each leaf node is assigned a class label

▻ non-terminal nodes contain attribute test conditions to separate records that have different characteristics.

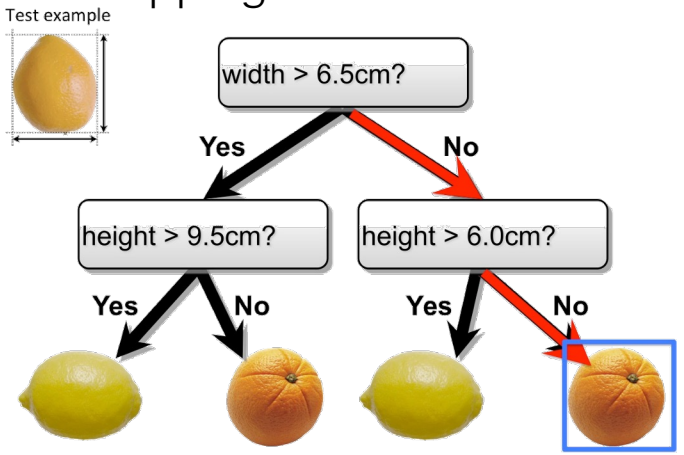
How to build (train) a decision tree?

Training – (Build a Decision Tree)

A top-down divide-and-conquer learning procedure

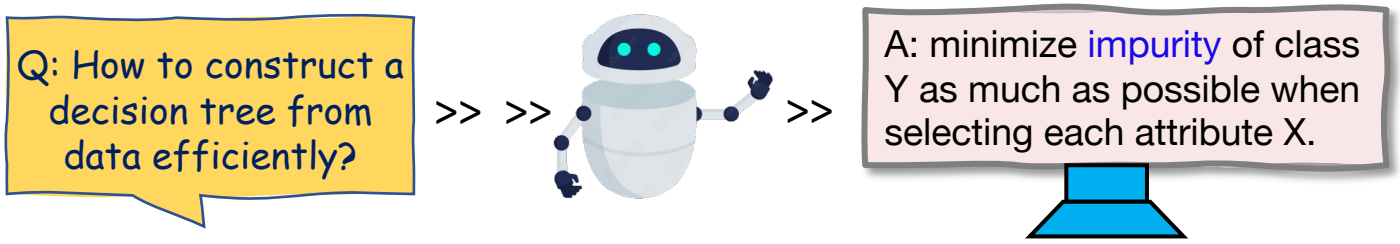
1. Construct a root node which contains the whole data set.
2. Selecting an attribute that benefits the task most according to some criterion.
3. Split the examples of the current node into subsets based on values of the selected attributes.
4. Create a child node for each subset and passes the examples in the subset to the node.
5. Recursively repeat step 2~4 until some stopping criterion is met.

ID	width	height	Type
1	5.2	8.0	lemons
2	6.7	9.8	lemons
3	7.2	7.5	orang
4	6.1	5.3	orang
5	4.1	6.5	lemons

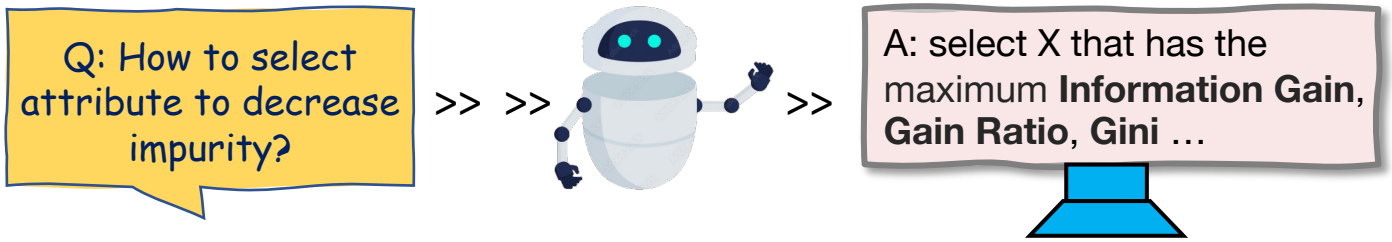


Key Problems

- There could be more than one tree that fits the same data!
- **Exponentially** many decision trees can be built.



Any decision tree will successively split the data into smaller and smaller subsets. It would be ideal if all the samples associated with a leaf node were from the same class. Such a subset, or node, is considered **pure** in this case.



Each of these metrics corresponds to a training algorithm.

Training Algorithms

ID3 (Information Gain)

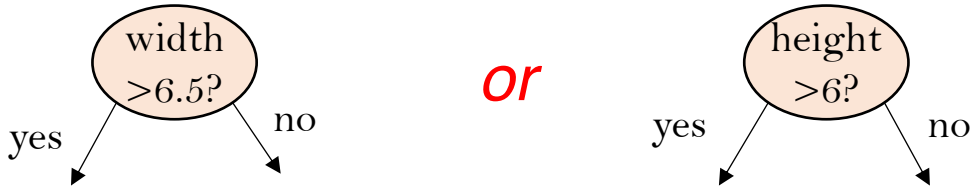
CART (Gini Index)

C4.5 (Gain Ratio)

…

ID3 –

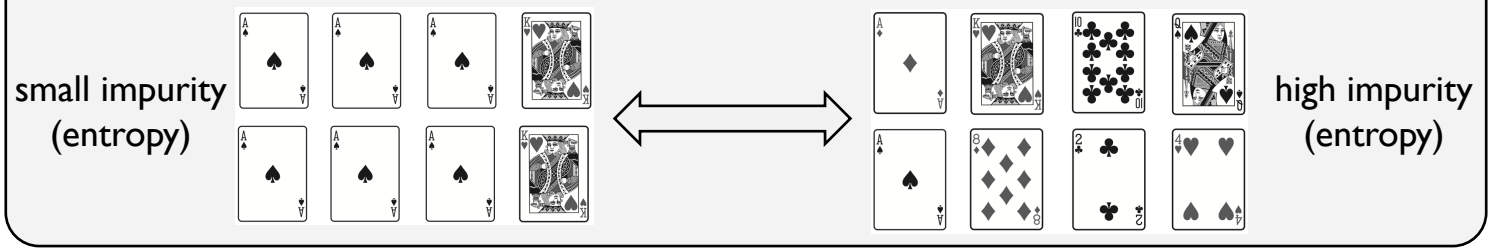
- Split attributes that have the maximum **Information Gain.**
  - Try to split at each attribute. See which split works the "**best**" .



  - Measure the **reduction** in the overall **entropy** of a set of instances

Recall: Entropy – measures the impurity of the elements of a set.

$$H(D) = -\sum_{k=1}^{K} p_k \log p_k$$



ID3

$$H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} \log \frac{|C_k|}{|D|}$$

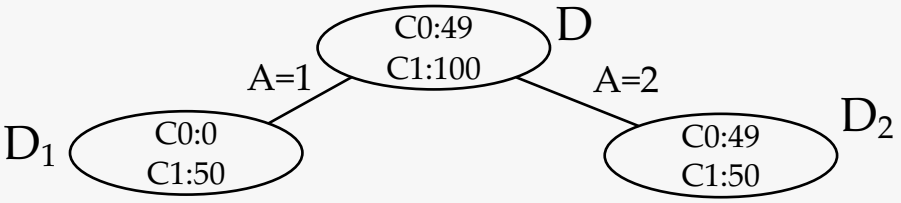
($|C_k|$: # samples of class C_k in dataset D)

$$H(D|A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \left(\sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} \log \frac{|D_{ik}|}{|D_i|} \right)$$

($|D_i|$: # samples whose attribute A is set to the i-th value in D; $|D_{ik}|$: #samples of class C_k in D_i)

Gain $(D,A) = H(D) - H(D|A)$

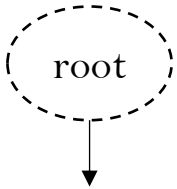
Example: What is the information gain of this split?



- root entropy: $H(D) = -{}^{49}/_{149} \log({}^{49}/_{149}) - {}^{100}/_{149} \log({}^{100}/_{149}) \approx 0.91$
- leaves entropy: $H(D|A=1) = 0,\ H(D|A=2) \approx 1$
- IG(D|A) $\approx 0.91 - (\frac{1}{3} \cdot 0 + \frac{2}{3} \cdot 1) \approx 0.24 > 0$

ID3 – Example

ID	width	height	Type
1	5.2	8.0	lemons
2	6.7	9.8	lemons
3	7.2	7.5	orang
4	6.1	5.3	orang
5	4.1	6.5	lemons



$\{1,2,3,4,5\}$

Step1 – create a root node

Step2 – calculate the entropy of the entire (sub) dataset.

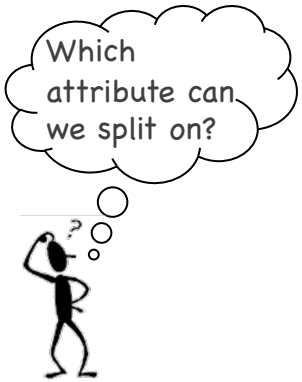
lemon	orange
3	2

$$H(D) = -\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5} = 0.97$$

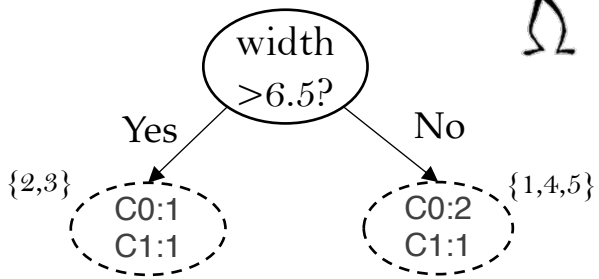
ID3 – Example

Step 3 – compute IG for each attribute and select the attribute with the maximum IG.

ID	width	Type
1	5.2	lemons
2	6.7	lemons
3	7.2	orang
4	6.1	orang
5	4.1	lemons

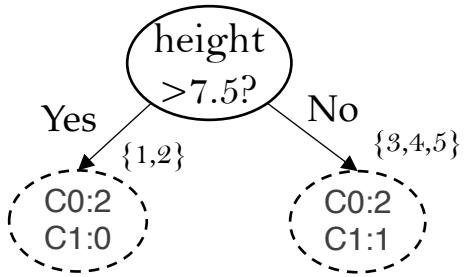


ID	height	Type
1	8.0	lemons
2	9.8	lemons
3	7.5	orang
4	5.3	orang
5	6.5	lemons



H=-½log½-½log½ = 1

H=-⅔log⅔-⅓log⅓ =0.92



H=-1log1-0log0 = 0

H=-⅔log⅔-⅓log⅓ =0.92

H(D | width) = ⅖ H(D | width>6.5) + ⅗H(D | width<6.5)
= ⅖ x 1 + ⅗ x 0.92 = 0.95
Gain(D | width) = H(D) – H(D | width) = 0.97-0.95=0.02

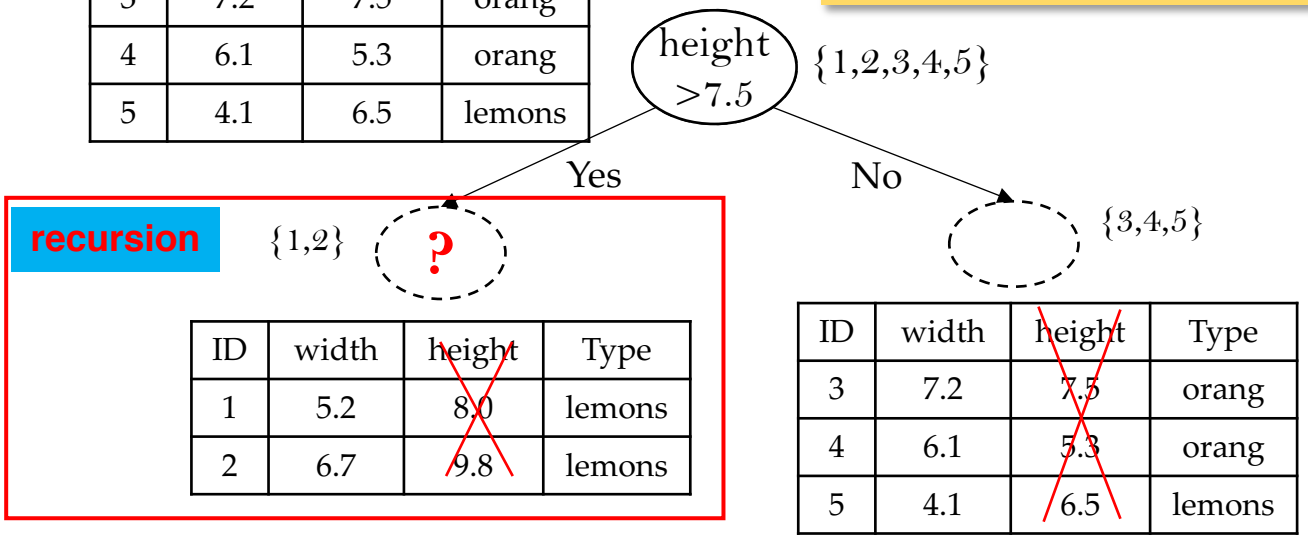
H(D | height) = ⅖ H(D | height>7.5) + ⅗ H(D |height<7.5)
= ⅖ x 0 + ⅗ x 0.92 = 0.55
Gain(D | height) = H(D) – H(D | height) = 0.97-0.55=0.42

ID3 – Example

Step 4 - Assign the current (root) node with the attribute that has the maximum IG. For each attribute value, expand an outgoing branch to a newly-created node.

ID	width	height	Type
1	5.2	8.0	lemons
2	6.7	9.8	lemons
3	7.2	7.5	orang
4	6.1	5.3	orang
5	4.1	6.5	lemons

Step 5 – Split the dataset along the values of the maxIG attribute and remove this feature from the dataset.



ID	width	~~height~~	Type
1	5.2	~~8.0~~	lemons
2	6.7	~~9.8~~	lemons

ID	width	~~height~~	Type
3	7.2	~~7.5~~	orang
4	6.1	~~5.3~~	orang
5	4.1	~~6.5~~	lemons

Step 6 – For each subset, repeat steps 2-5 until a stopping criteria is satisfied → here the recursion kicks in.

CART

- Find the best split using **Gini Index**.

$$\textbf{Gini}\ (\mathrm{D}) = \sum_{k=1}^{K} \frac{|C_k|}{|D|}\left(1 - \frac{|C_k|}{|D|}\right) = 1 - \sum_{k=1}^{K}\left(\frac{|C_k|}{|D|}\right)^2$$

$$\textbf{Gini}\ (\mathrm{D}\,|\,\mathrm{A}) = \sum_{i=1}^{n} \frac{|D_i|}{|D|}\mathrm{Gini}(\mathrm{D_i})$$

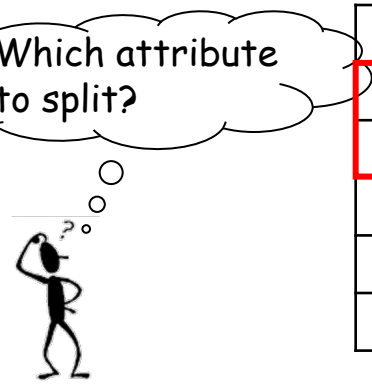
Gini represents the probability that two randomly selected samples belong to different classes.

Gini is **cheaper in computation** than Entropy which needs to compute *log* functions.

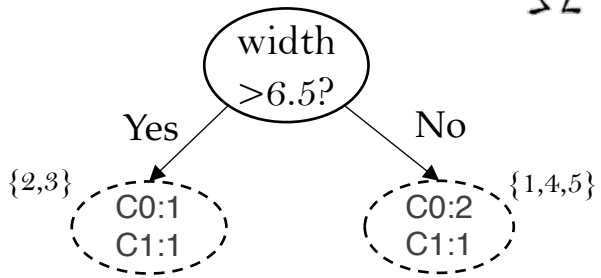
CART – Example

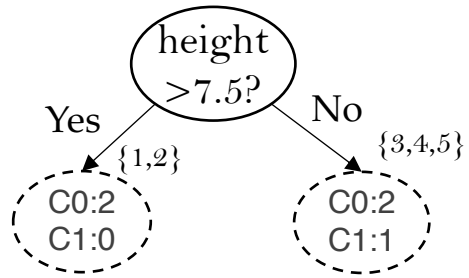
Step3 - calculate the **Gini Gain** of each attribute and pick the attribute with the max GiniGain.

ID	width	Type
1	5.2	lemons
2	6.7	lemons
3	7.2	orang
4	6.1	orang
5	4.1	lemons



ID	height	Type
1	8.0	lemons
2	9.8	lemons
3	7.5	orang
4	5.3	orang
5	6.5	lemons





Gini = 1-(½)²–(½)² = 0.5

Gini = 1-(⅔)²-(⅓)² =0.44

Gini=1-(⅔)²-(⅓)² =0.44

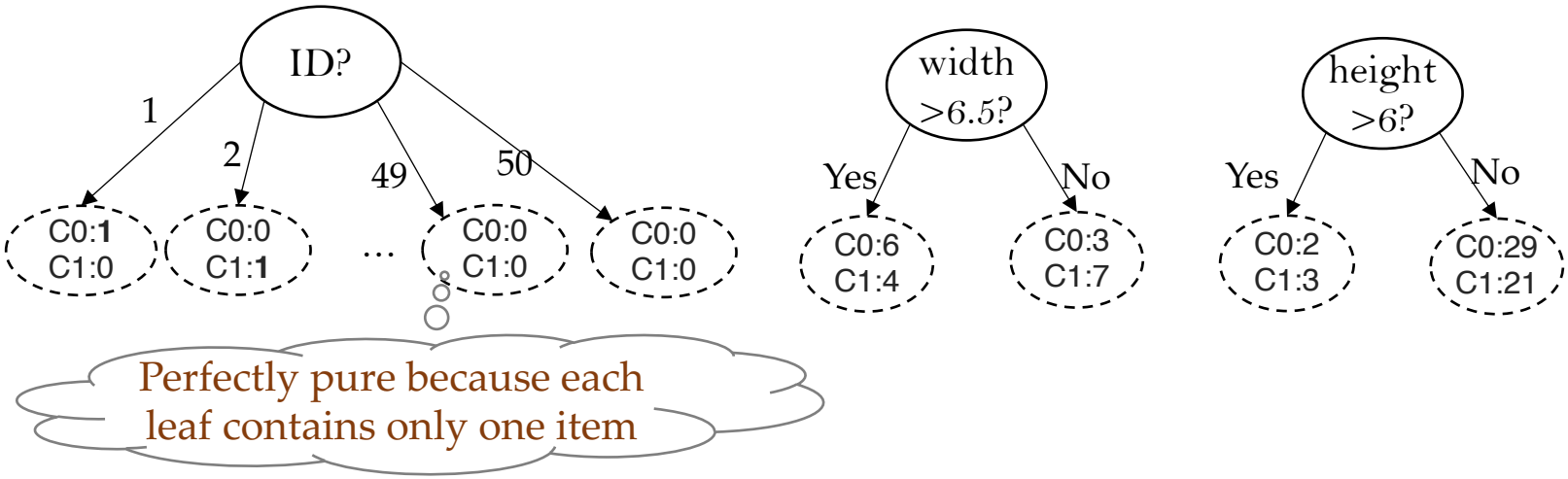
Gini(D|width) = ⅖Gini(D|width>6.5) + ⅗Gini(D|width<6.5)
= ⅖ x 0.5 + ⅗ x 0.44 =
Gain (D | width) = Gini(D) – Gini(D | width) =

Gini(D |height) = ⅖Gini(D |height>7.5)+⅗Gini(D |height<7.5)
= ⅖ x 0 + ⅗x 0.44 =
Gain (D | height) = Gini(D) – Gini(D | height) =

Shortcoming of ID3 and CART

- Entropy and Gini are bias to attributes with many distinct values.

Possible nodes to split at:



- ID will result in perfectly pure children.
- Will have the greatest information gain.
- Should have been removed as a predictor variable.

C4.5

C4.5 uses gain ratio instead of information gain.

- takes into account the number of outcomes produced by attribute split condition.
- Adjusts information gain by the entropy of the partitioning.

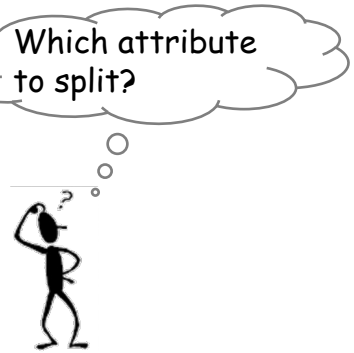
$$\textbf{Gain Ratio}\ (\mathrm{D}, \mathrm{A}) = \frac{\mathrm{Gain(D,A)}}{\mathrm{H_A(D)}}$$

$$\mathrm{H_A(D)} = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \log \frac{|D_i|}{|D|}$$

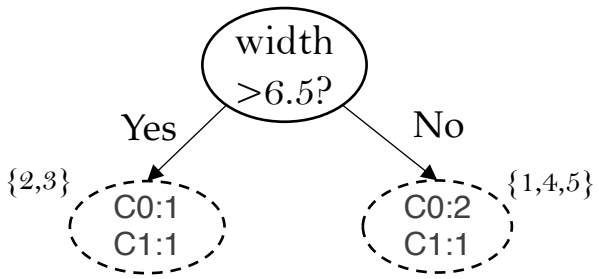
C4.5 – Example

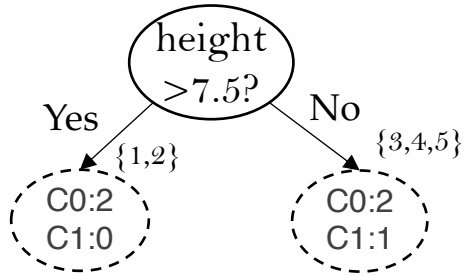
Step 3 - calculate the **GainRatio** for each attribute and select the attribute with the max GR.

ID	width	Type
1	5.2	pear
2	6.7	pear
3	7.2	orang
4	6.1	orang
5	4.1	pear



ID	height	Type
1	8.0	pear
2	9.8	pear
3	7.5	orang
4	5.3	orang
5	6.5	pear





H=-½log½-½log½ = 1

H=-⅔log⅔-⅓log⅓=0.92

H=-1log1-0log0 = 0

H=-⅔log⅔-⅓log⅓=0.92

H(D | width) = ⅖ H(D | width>6.5) + ⅗H(D | width<6.5)
= ⅖ x 1 + ⅗x 0.92 = 0.95
Gain (D | width) = H(D) – H(D | width) = 0.97-0.95=0.02
$H_{width}(D)$ = - ⅖ log⅖ – ⅗ log⅗ = 0.97
GainRatio (D | width) = 0.02 /0.97= 0.02

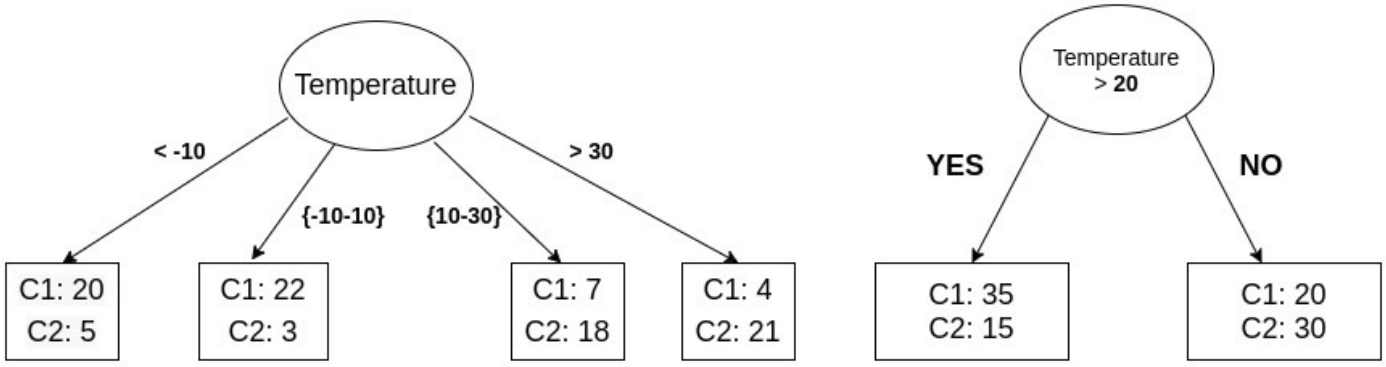
H(D | height) = ⅖ H(D | height>7.5) + ⅗H(D | height<7.5)
= ⅖ x 0 + ⅗ x 0.92 = 0.55
Gain(D | height) = H(D)–H(D | height) = 0.97-0.55 = 0.42
$H_{height}(D)$ = -⅖ log⅖ – ⅗log⅗ = 0.97
GainRatio (D | height) = 0.43 /0.97= 0.5

Stop Criteria

- Purity
  - The leaves contain the training examples from the same class
- Minimum number of points
  - The number of training examples contained in the leaves are less than a threshold
- No more attribute for split (ID3)

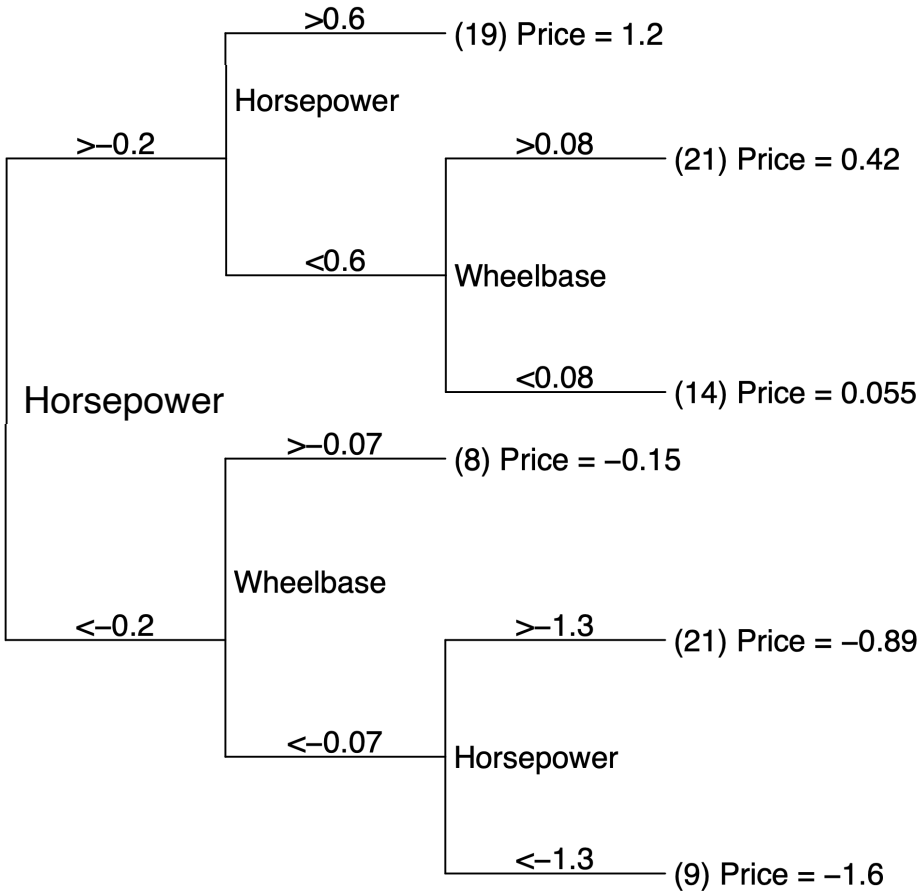
Splitting Continuous-Valued Attributes

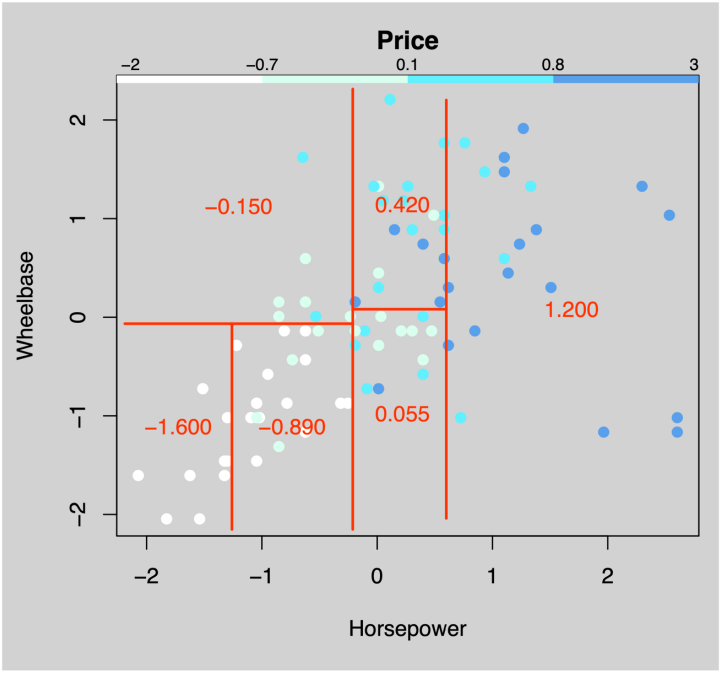
- Convert continuous-valued attribute into a categorical one by splitting it into *n* range buckets with equal width.
  - the number of categories (buckets) is a hyperparameter.
  - more sophisticated methods involve the use of unsupervised clustering algorithms to define the optimal categories.
- Binary split using a comparison operator (≥, ≤)
  - need to determine the threshold which is computationally expensive.
  - can sort the values of the continuous attribute and take the midpoint.



Regression Trees (扩展内容)

Regression: $y = w_0 + w_1 x_1 + \ldots + w_d x_d$





Decision Tree vs. Regression Tree (扩展内容)

Classification Trees

- Gain of a split: "reduction of impurity.
- Larger gain => better split

$$\text{gain} = \text{H(parent)} - \Sigma_i \, p_i \, \text{H}(\text{ch}_i)$$

Regression Trees

- Impurity (variance) at a node:

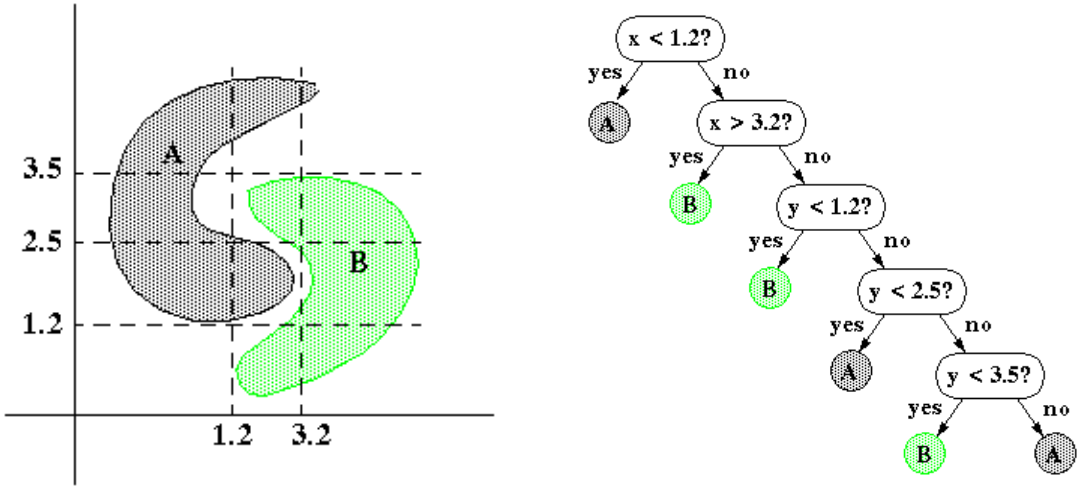
$$\text{var}(t, D) = \frac{\sum_{i=1}^{n} (t_i - \bar{t})^2}{n - 1}$$

- Select feature to split on that minimizes the weighted variance across all resulting partitions

Pros and Cons

Advantages:

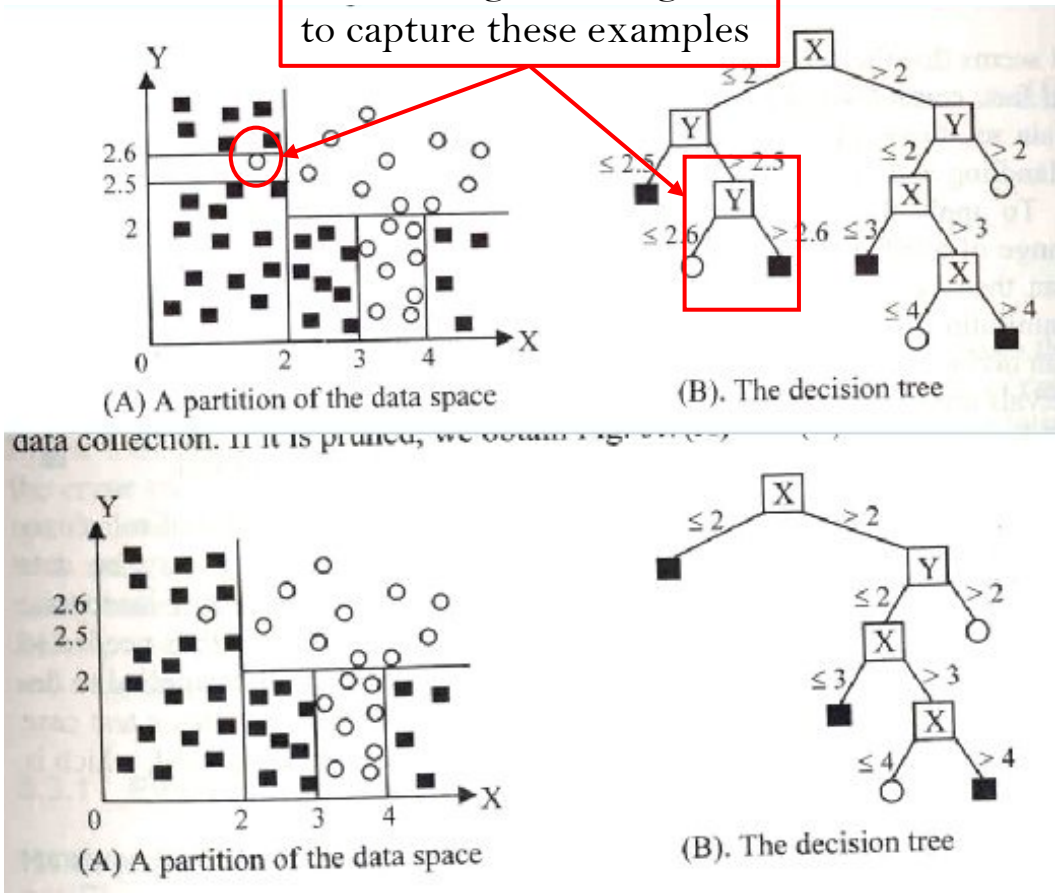
- simple to understand, explain, and visualization,
- little effort for data preprocessing,
- can produce nonlinear decision surfaces,
- data-driven and can give arbitrarily high levels of precision on the training data.



Pros and Cons

Disadvantages:

- overfitting



Avoid Overfitting – Pruning

- Why pruning?
  - To reduce the chance of overfitting

- Pruning strategy
  - Pre-pruning

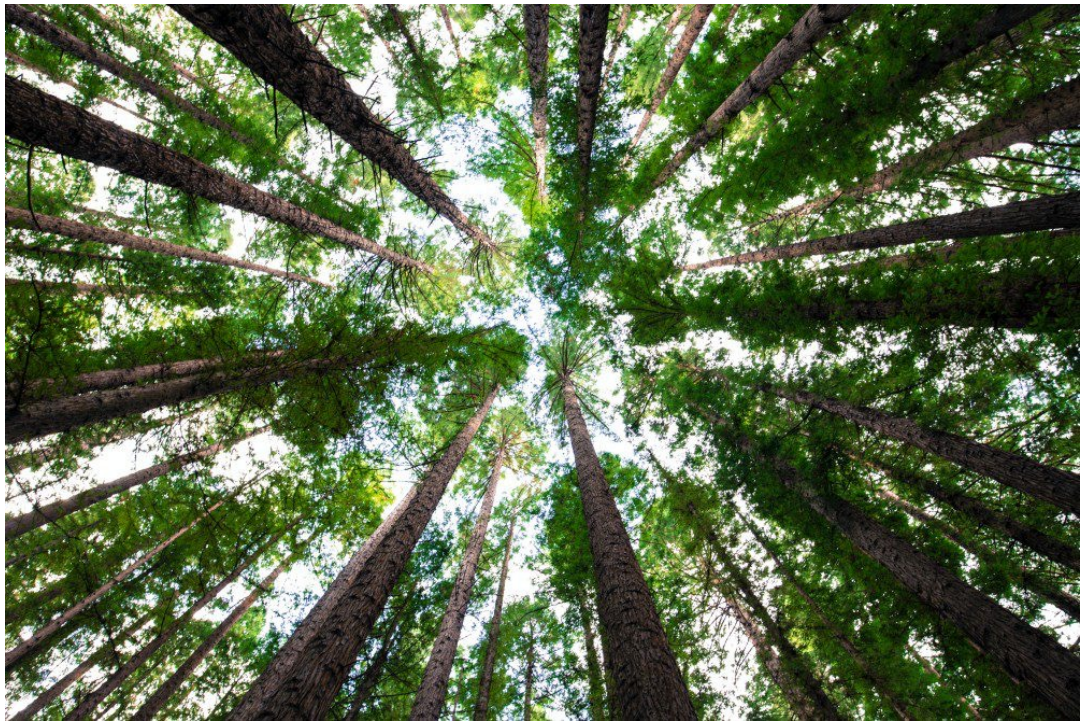
    Stop growing tree early if the goodness measure is less than a threshold
  - Post-pruning

    Remove branches after a tree has been fully grown.

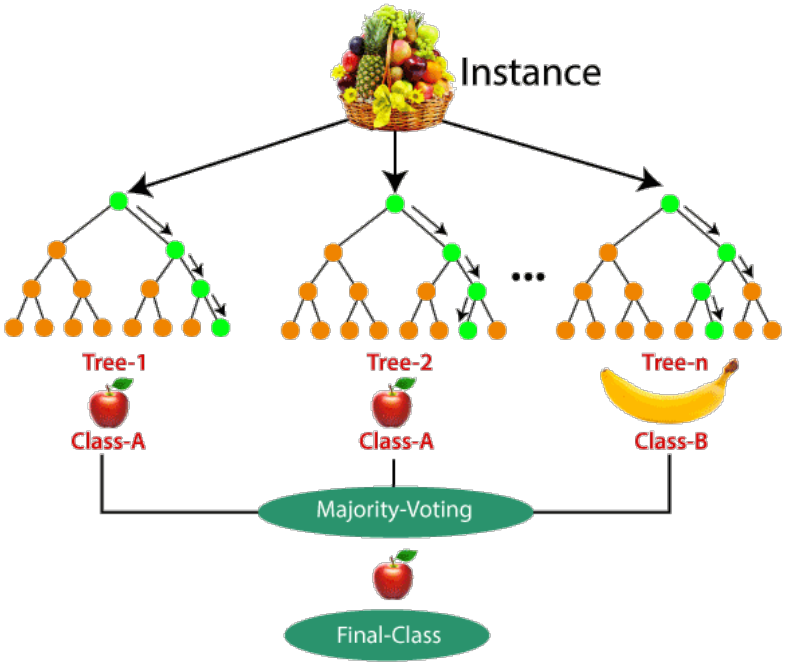
Post-pruning usually outperforms pre-pruning, but needs heavier computational cost.

Ensemble of Multiple Decision Trees

- Build multiple decision trees on random subsets of data and attributes, and combine their results. **(三个臭皮匠，顶一个诸葛亮)**



Random Forest



1. Sampling a set of N samples from the original dataset, **putting them back after sampling.**
2. Train a decision tree using the random dataset. For each node:
   a. Randomly sample d attributes
   b. Split nodes based on the selected attributes (e.g., information gain)
3. Repeat 1~2 for k times
4. Aggregate predictions by all decision trees, produce the final results by majority voting.

Random Forest

Advantages:

- More accurate than decision tree.
- Fast, and easy for parallel (independence between trees).
- Can deal with high-dimensional data (i.e., with many attributes) without feature selection (because the subset of attributes for training are randomly selected).
- Can deal with missing attributes.
- Can explain the results (i.e., tell which attributes are more important after training).
- Can identify the correlations between attributes during training.
- Balance errors on unbalanced datasets.
- ...

TIME for Coding

Tutorial: Decision Trees from Scratch with Python

Manually setup using the Jupyter lab online [no sklearn]:
../tutorials/ch3_decision_tree.ipynb
../tutorials/data/zoo.csv

A runnable online tutorial using sklearn:
https://www.kaggle.com/code/prashant111/decision-tree-classifier-tutorial

What's Next?

Bayes Classifier

- A **new perspective** of machine learning and classification.
- Classify data by estimating posterior probability.
  - Bayes Decision
  - Classification using Bayes Net
  - Naïve Bayes