LMP 1210H: Basic Principles of Machine Learning in Biomedical Research

Lecture 2 — Tree-based classifiers

Sana Tonekaboni

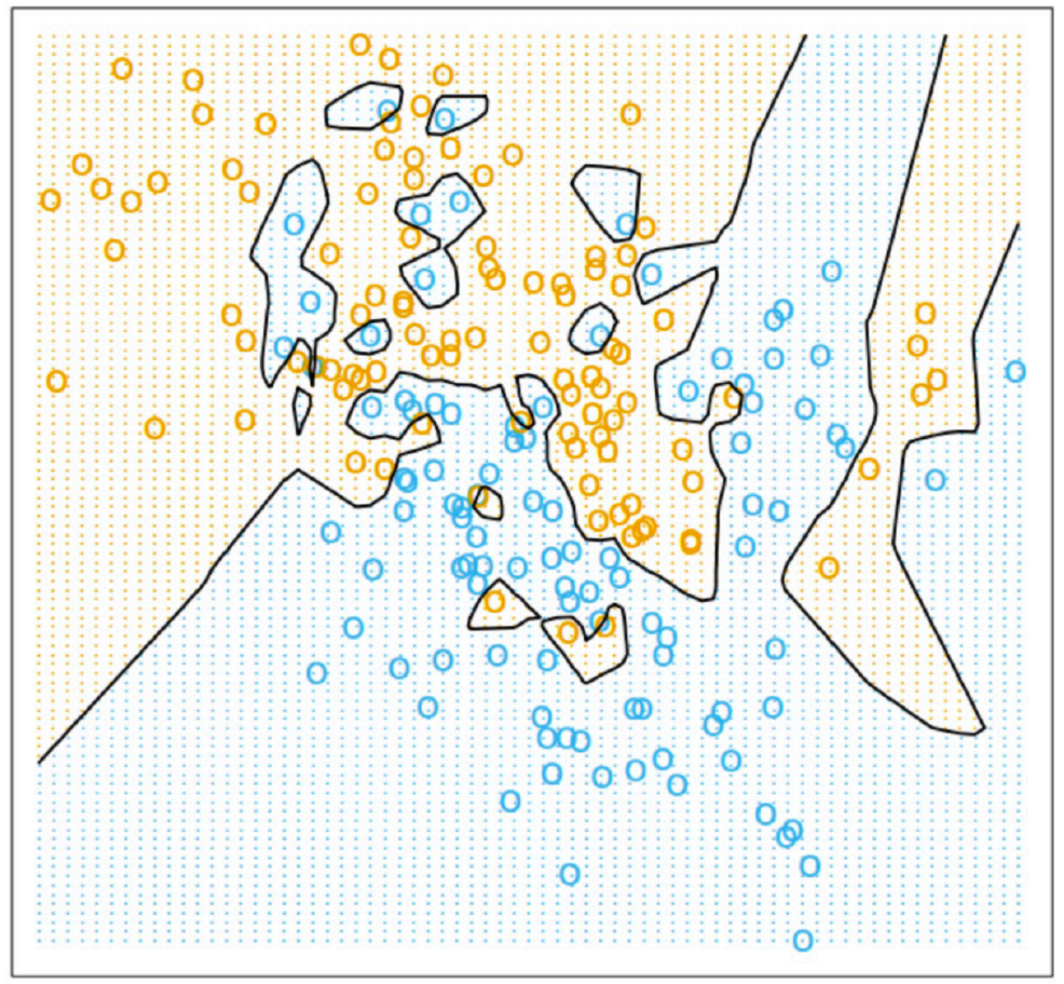
January 18, 2024

Quick check in…

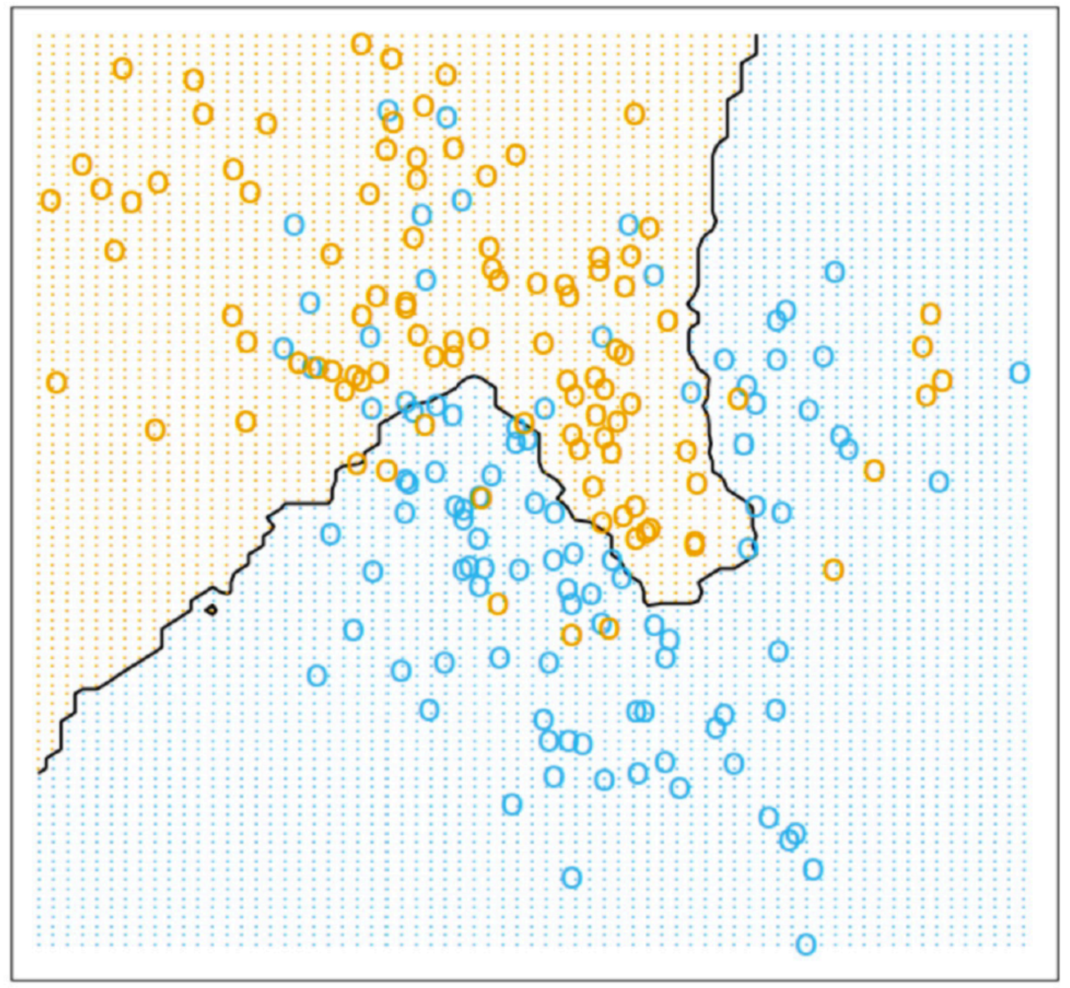
1. Math diagnostic assignment due today!
2. Assignment 1:
    1. Released today!
    2. Due February 1st, EOD
3. Make sure to sign up on Piazza and go to office hours!

Recap

- Typical ML workflow
- Supervised learning and vector representations
- KNN for classification (we will talk about regression later)
- Decision boundary
- What is a hyper-parameter and how to tune it
- KNN challenges like curse of dimensionality and some solutions



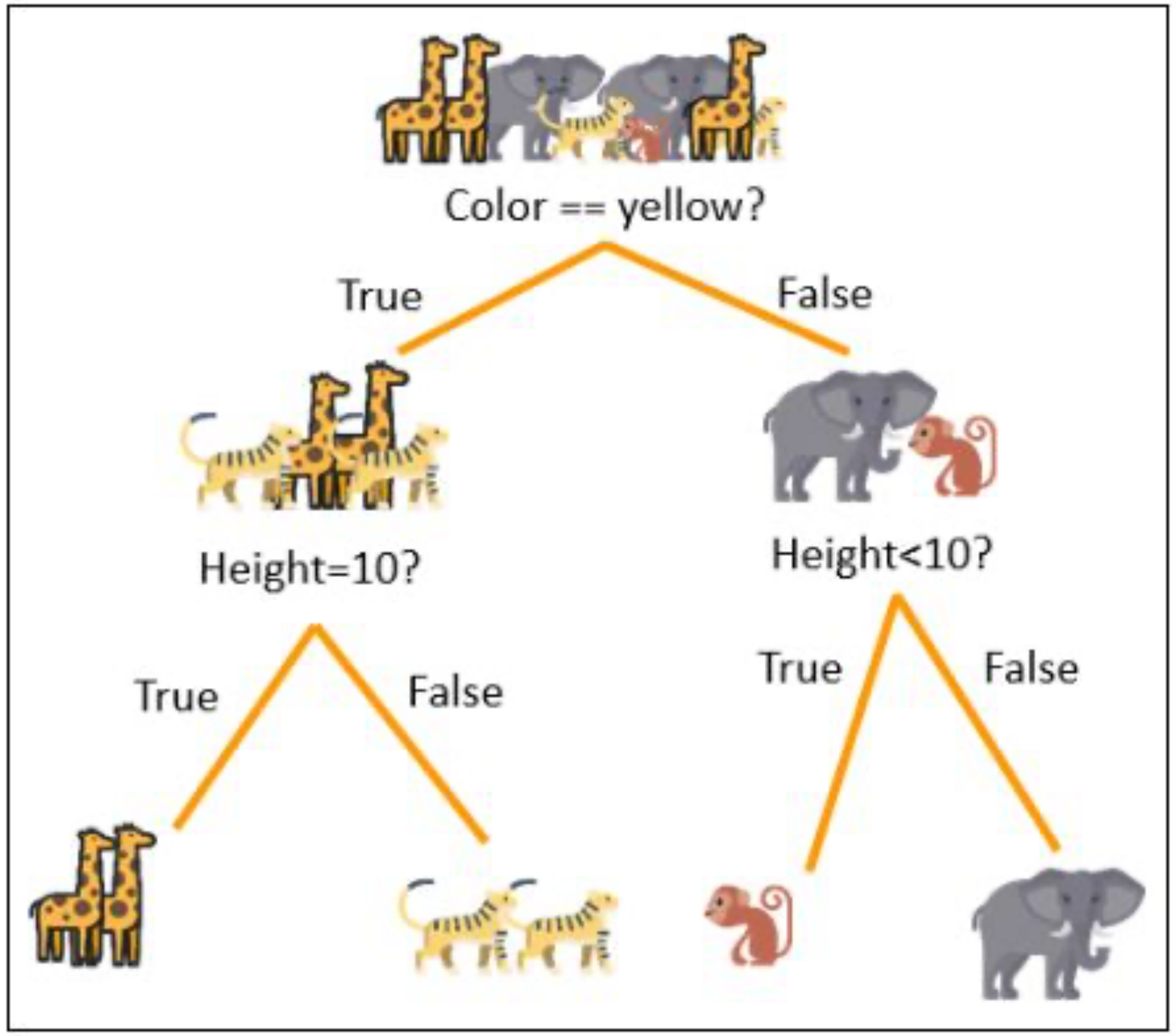
K = 1



K = 15

Decision trees

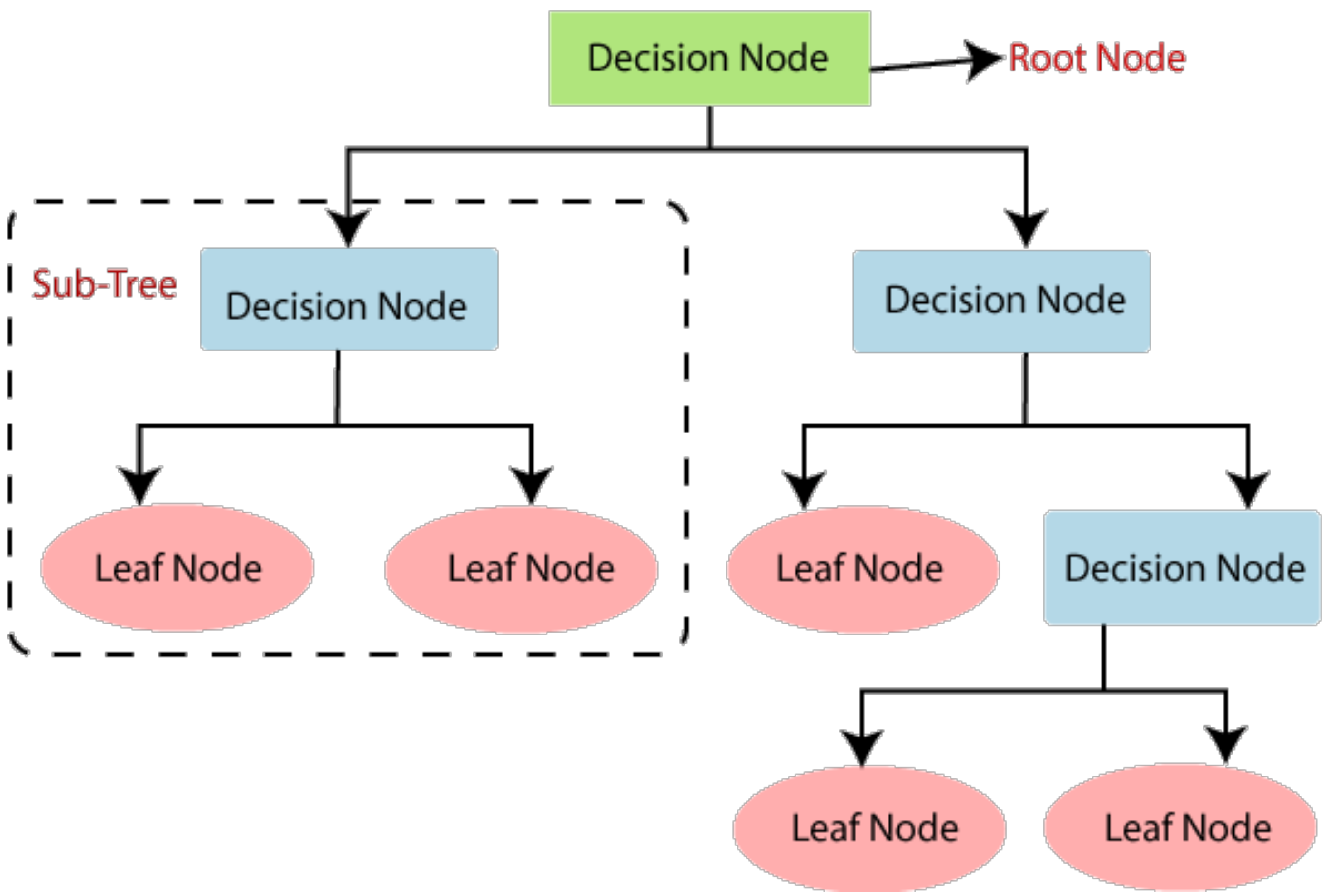
- Decision trees are simple but powerful learning algorithms
- Decision trees make predictions by recursively splitting on different attributes according to a tree structure.



https://www.simplilearn.com/tutorials/machine-learning-tutorial/decision-tree-in-python

Decision trees

Terminology

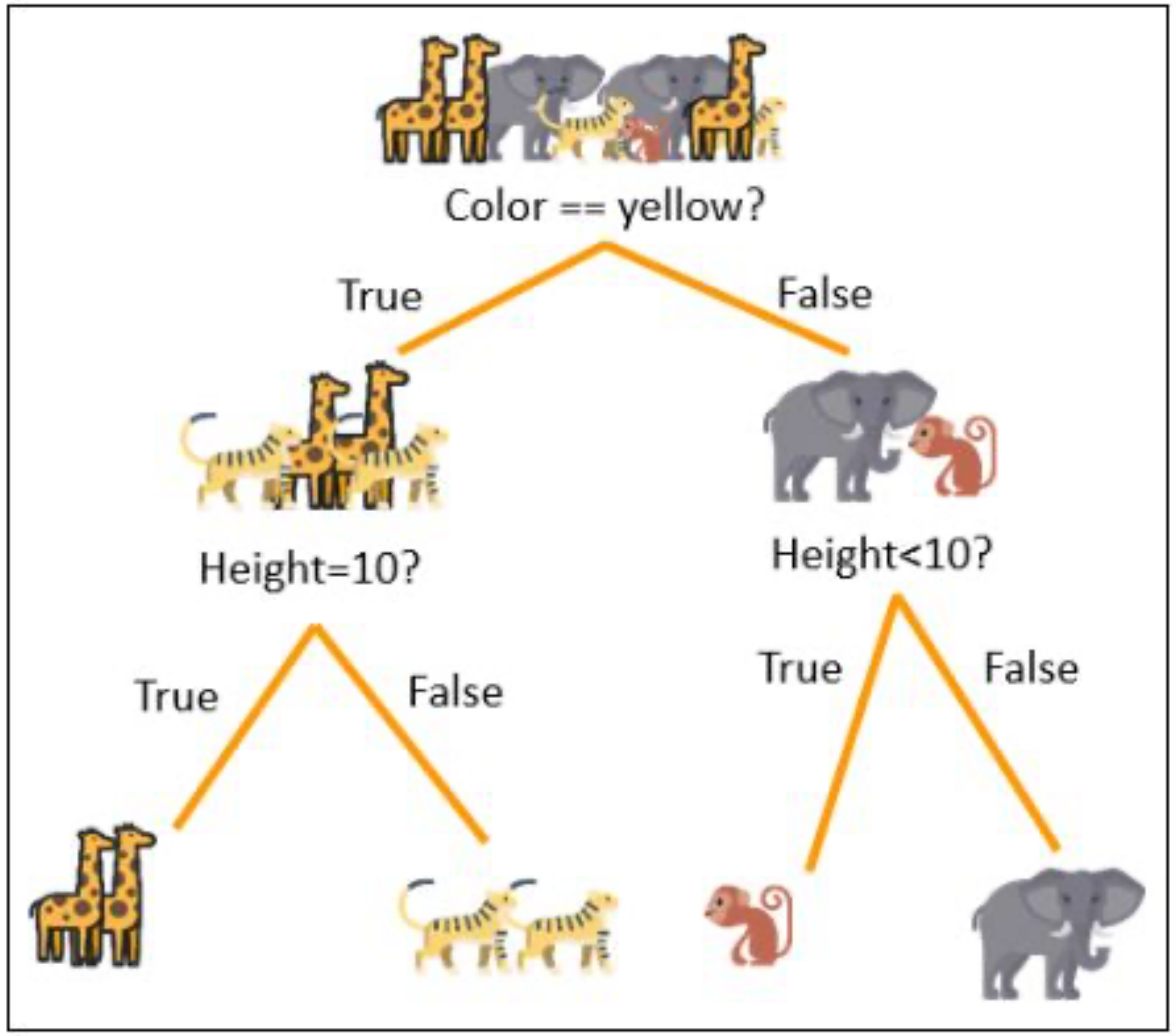


- **Root Node**: The top node on the tree that represents entire population being analyzed. The feature attribute in this node is selected based on Attribute Selection Techniques for dividing the data into two or more sets.
- **Branch or Sub-Tree:** Part of the entire decision tree is called a branch or sub-tree.
- **Splitting**: Dividing a node into two or more sub-nodes based on a condition.
- **Decision Node**: A node that splits a sub-nodes into further sub-nodes is called the decision node
- **Leaf or Terminal Node**: This is the end of the decision tree where it cannot be split into further sub-nodes.

Decision trees

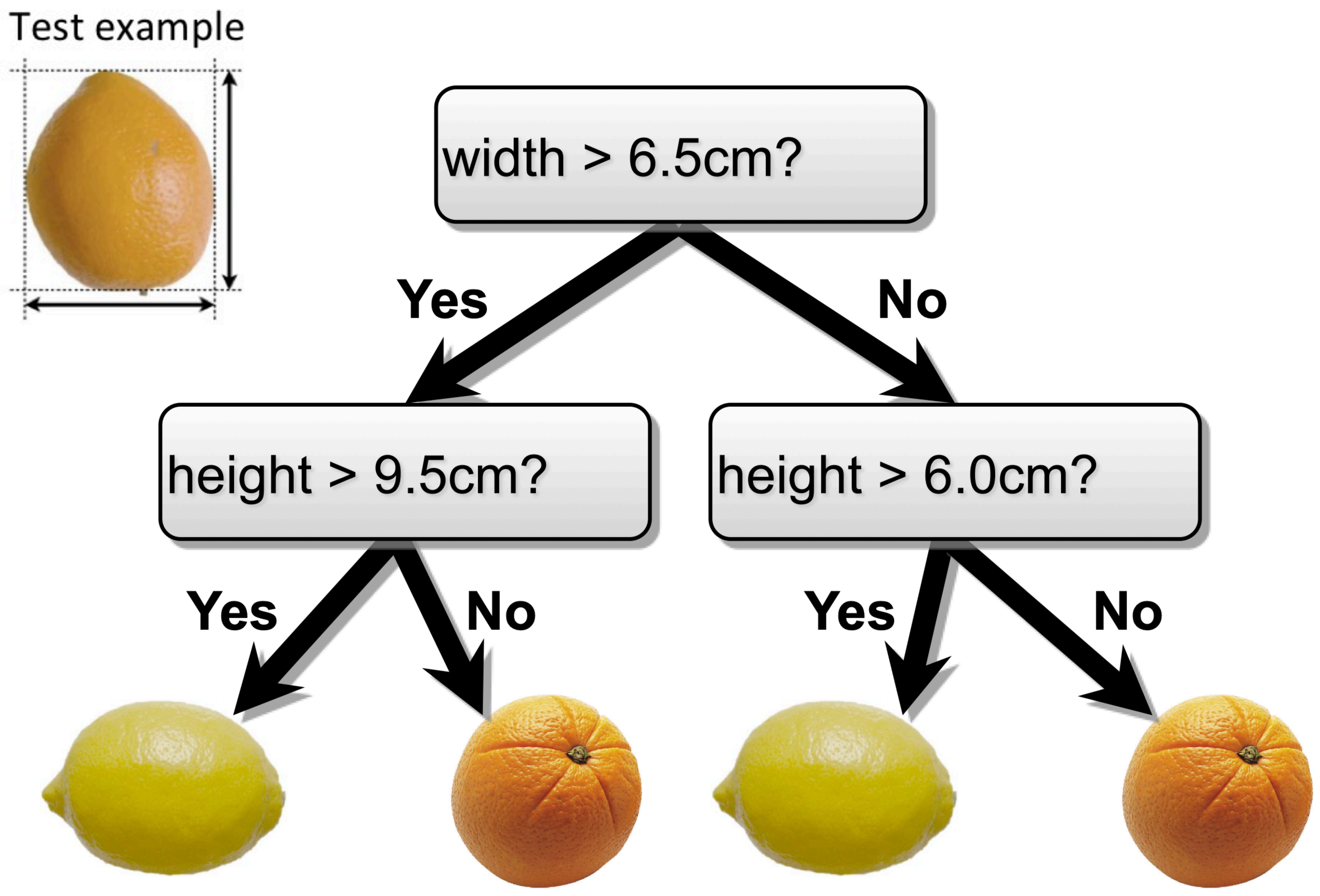
1. Each internal node tests an attribute
2. Create a branch for each possible attribute value
3. Each leaf assigns a class y
4. To classify input x: traverse the tree from root to leaf, output the labeled y

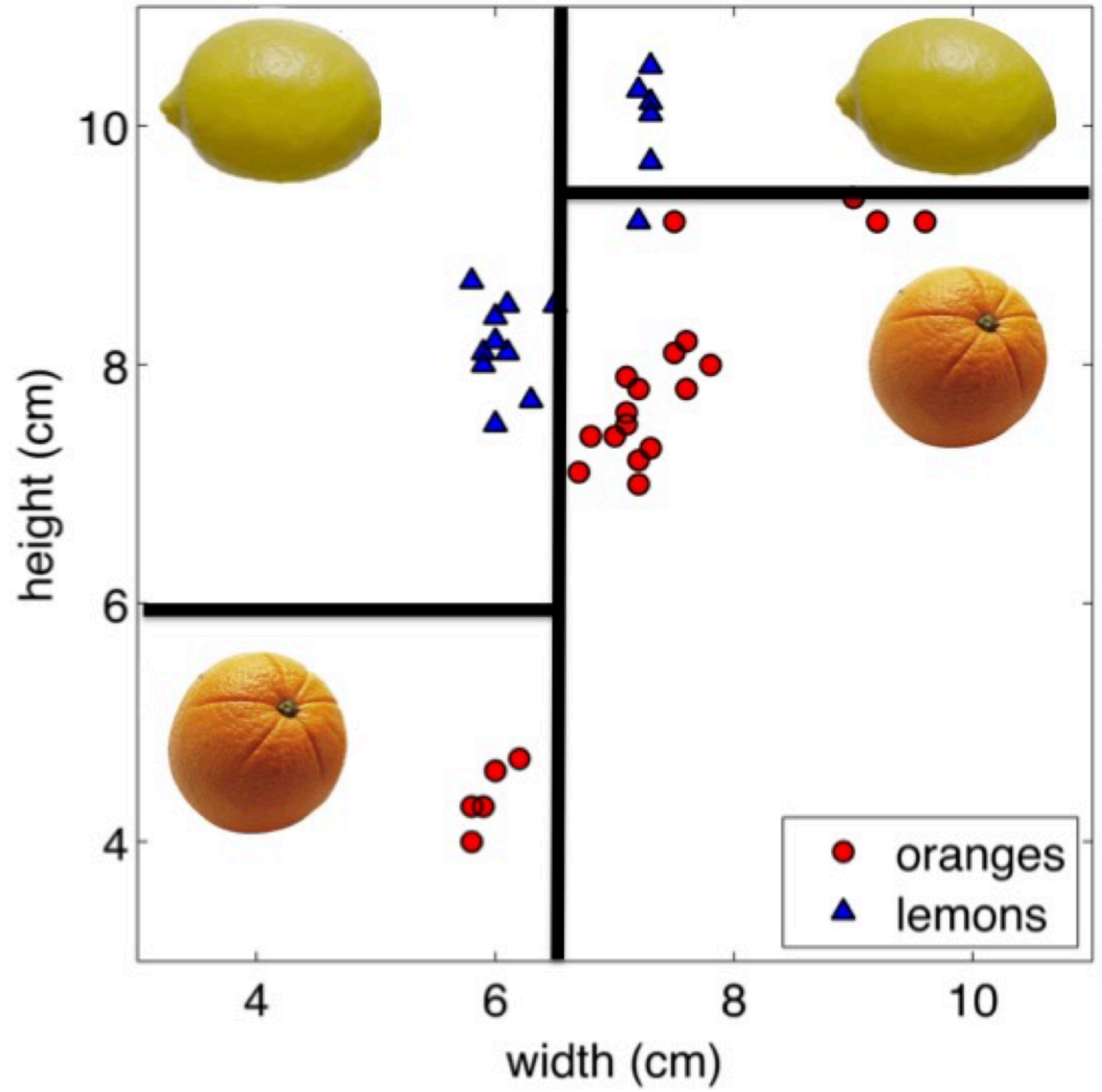
test sample



Decision trees

- For continuous variables, split based on less than or greater than some threshold
- Thus input space is divided into regions with boundaries parallel to axes.

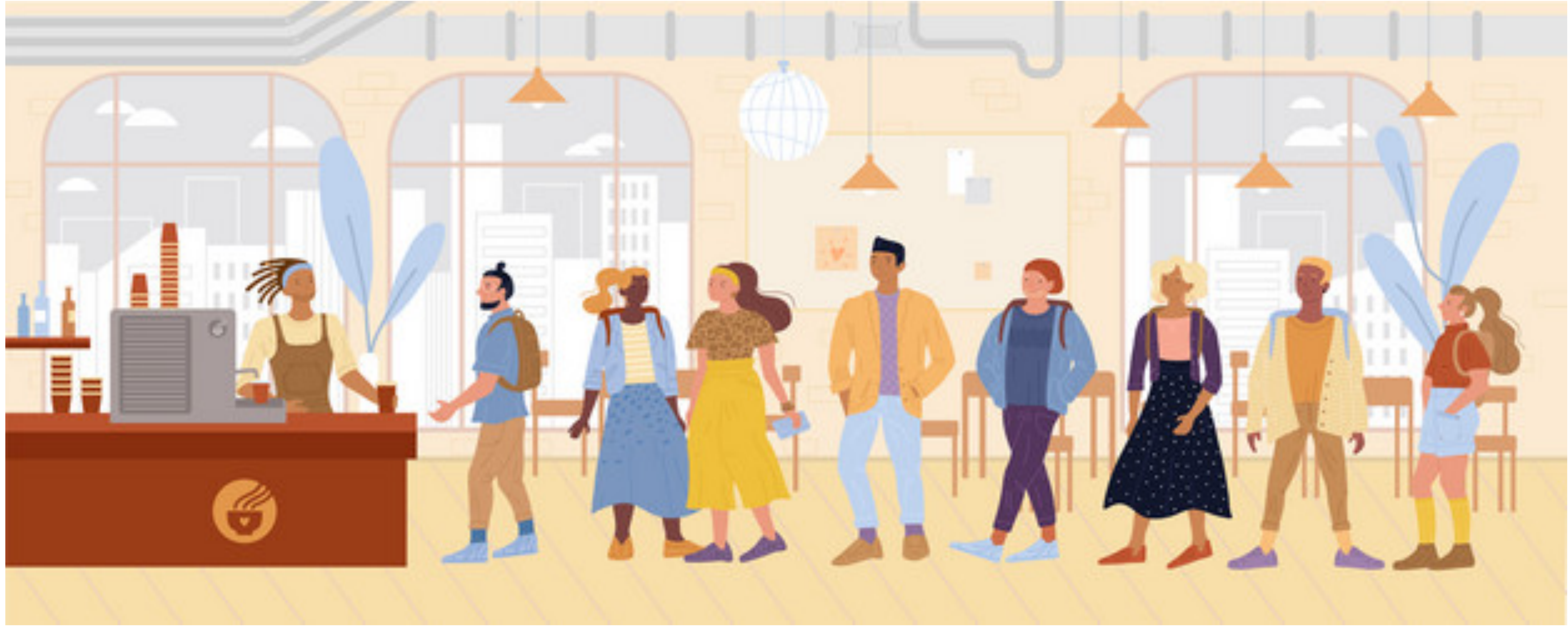




Decision trees

Let's look at an example….

To wait, or not to wait, that is the question….



Decision trees

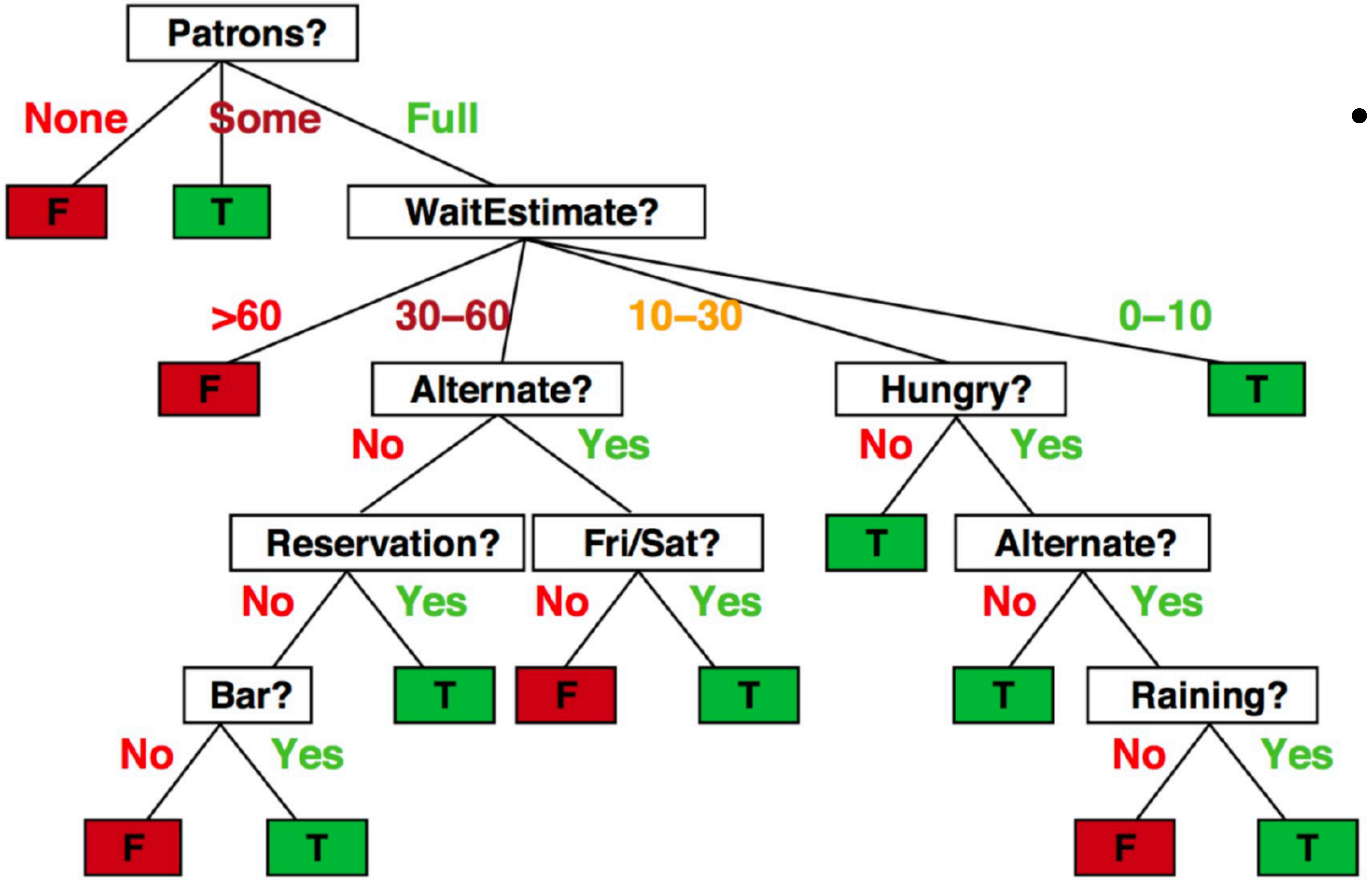
Discrete attributes

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	$	No	No	Burger	0–10	$y_3 =$ Yes
$\mathbf{x}_4$	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10–30	$y_4 =$ Yes
$\mathbf{x}_5$	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	$y_5 =$ No
$\mathbf{x}_6$	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0–10	$y_6 =$ Yes
$\mathbf{x}_7$	No	Yes	No	No	None	$	Yes	No	Burger	0–10	$y_7 =$ No
$\mathbf{x}_8$	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0–10	$y_8 =$ Yes
$\mathbf{x}_9$	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	$y_9 =$ No
$\mathbf{x}_{10}$	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10–30	$y_{10} =$ No
$\mathbf{x}_{11}$	No	No	No	No	None	$	No	No	Thai	0–10	$y_{11} =$ No
$\mathbf{x}_{12}$	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30–60	$y_{12} =$ Yes

Decision trees

Discrete attributes

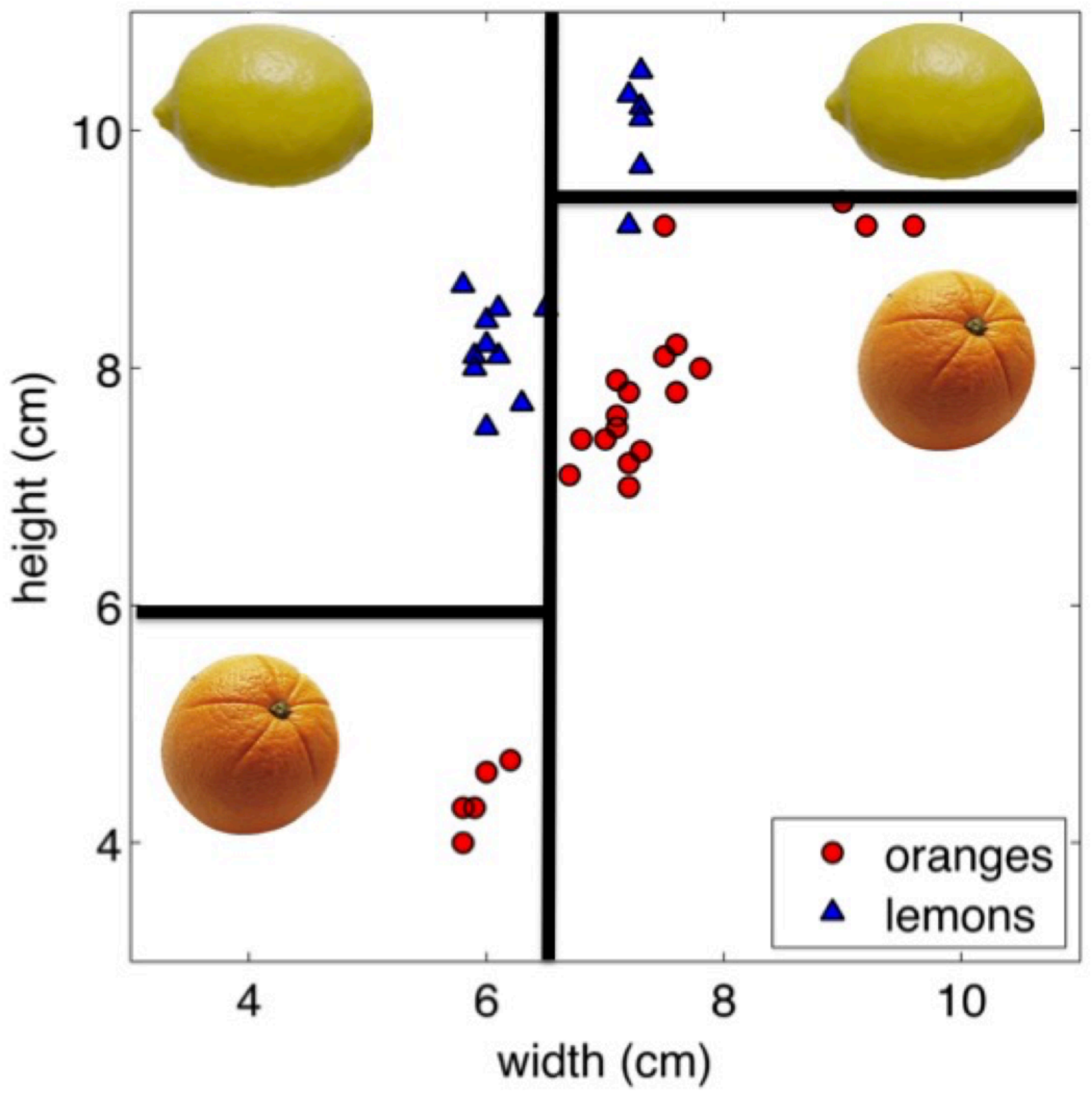
A possible tree to decide whether to wait or not



- **Internal nodes** test **attributes**
- **Branching** is determined by attribute values
- **Leaf nodes** are **outputs** (predictions)

Decision trees

Classification and regression



- Each path from root to a leaf defines a region of input space R_m
- Let $\{(x^{(m_1)}, t^{(m_1)}), (x^{(m_2)}, t^{(m_2)}), \cdots, (x^{(m_k)}, t^{(m_k)})\}$ be the training examples that fall into R_m
- **Classification tree:**
  - discrete output
  - leaf value y^m typically set to the most common value in $\{t^{(m_1)}, t^{(m_2)}, \cdots, t^{(m_k)}\}$
- **Regression tree:**
  - continuous output
  - leaf value y^m typically set to the mean value in $\{t^{(m_1)}, t^{(m_2)}, \cdots, t^{(m_k)}\}$
- Note: We will focus on classification

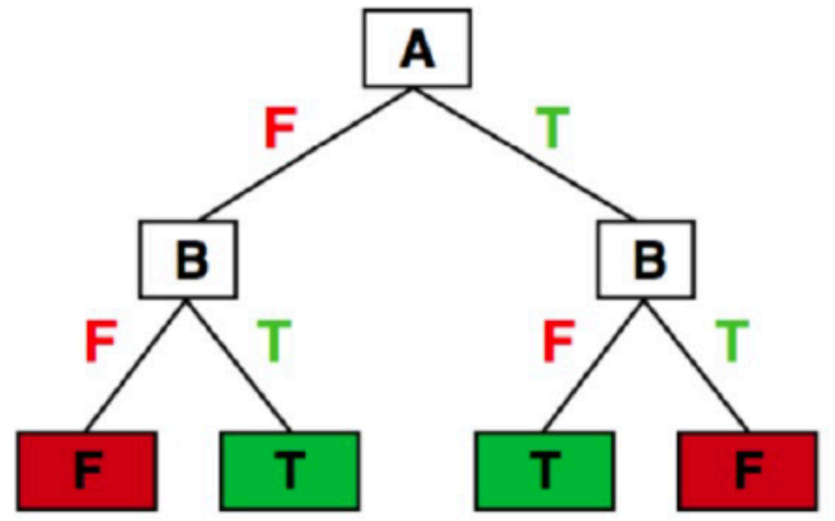
Decision trees

Expressiveness

Discrete-input, discrete-output case:

- Decision trees can express any function of the input attributes
- Example: Boolean functions, truth table row → path to leaf

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Continuous-input, Continuous-output case:

- Decision trees can approximate any function arbitrarily close

Note: Trivially, there is a decision tree for any training set with one path to leaf for each sample. But it probably won't generalize to new examples!

Decision trees

How do we learn a decision tree?

How do we construct a useful decision tree?

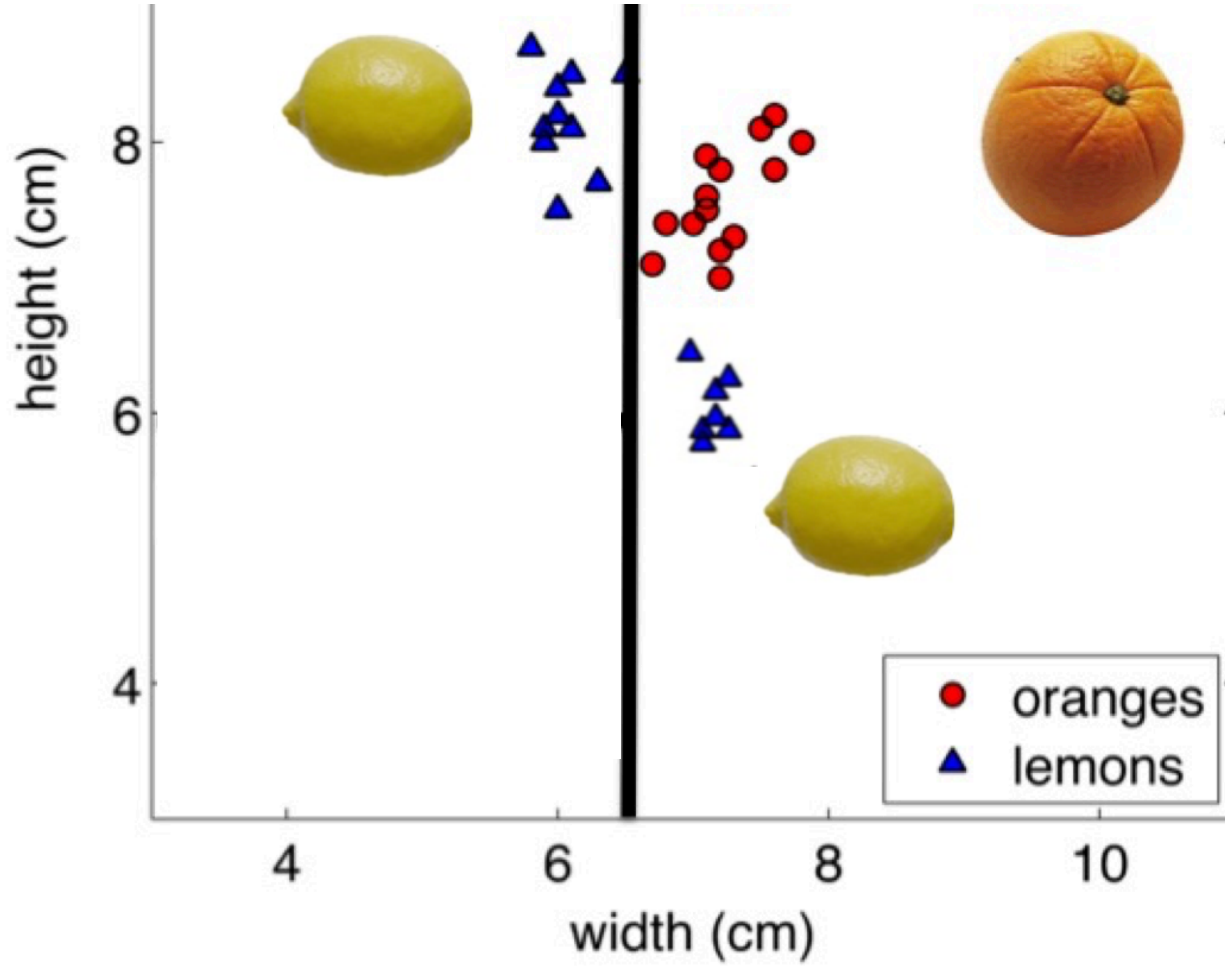
- Learning the simplest (smallest) decision tree is an NP complete problem [if you are interested, check: Hyafil & Rivest'76]
- Resort to a **greedy heuristic!** (start with an empty tree and a training set)**:**
  - Split on the "best" attribute
  - Recurse on sub partitions
- When should we stop?
- How to choose the "best" attribute or where to do a split if there is a continuous value?

Decision trees

How to choose the “best” attribute?

- Is accuracy a good measure?
  - Loss: Misclassification error
  - Let's say the region R is split into R_1 and R_2 based on $L(R)$
  - Then the accuracy gain will be:

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

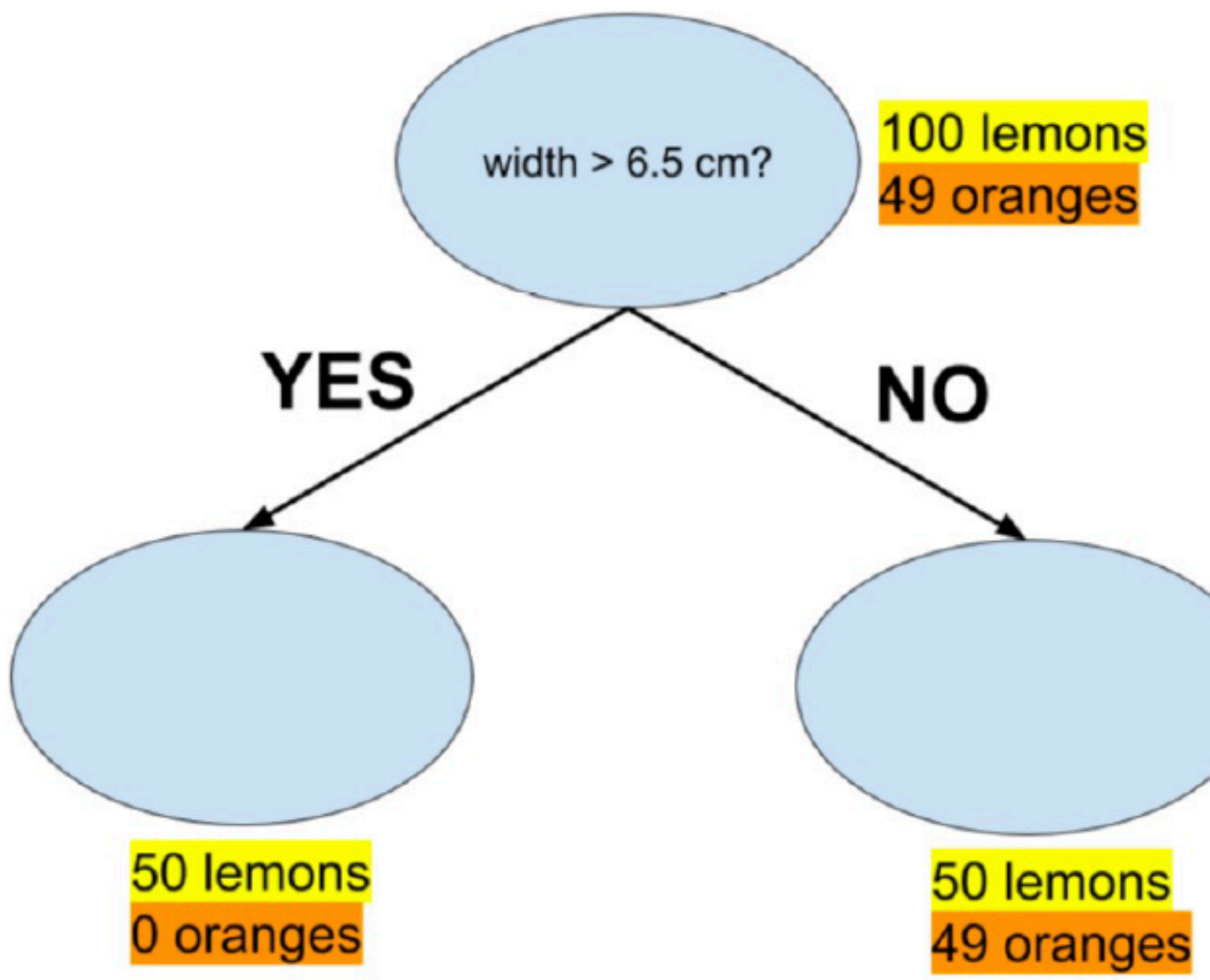


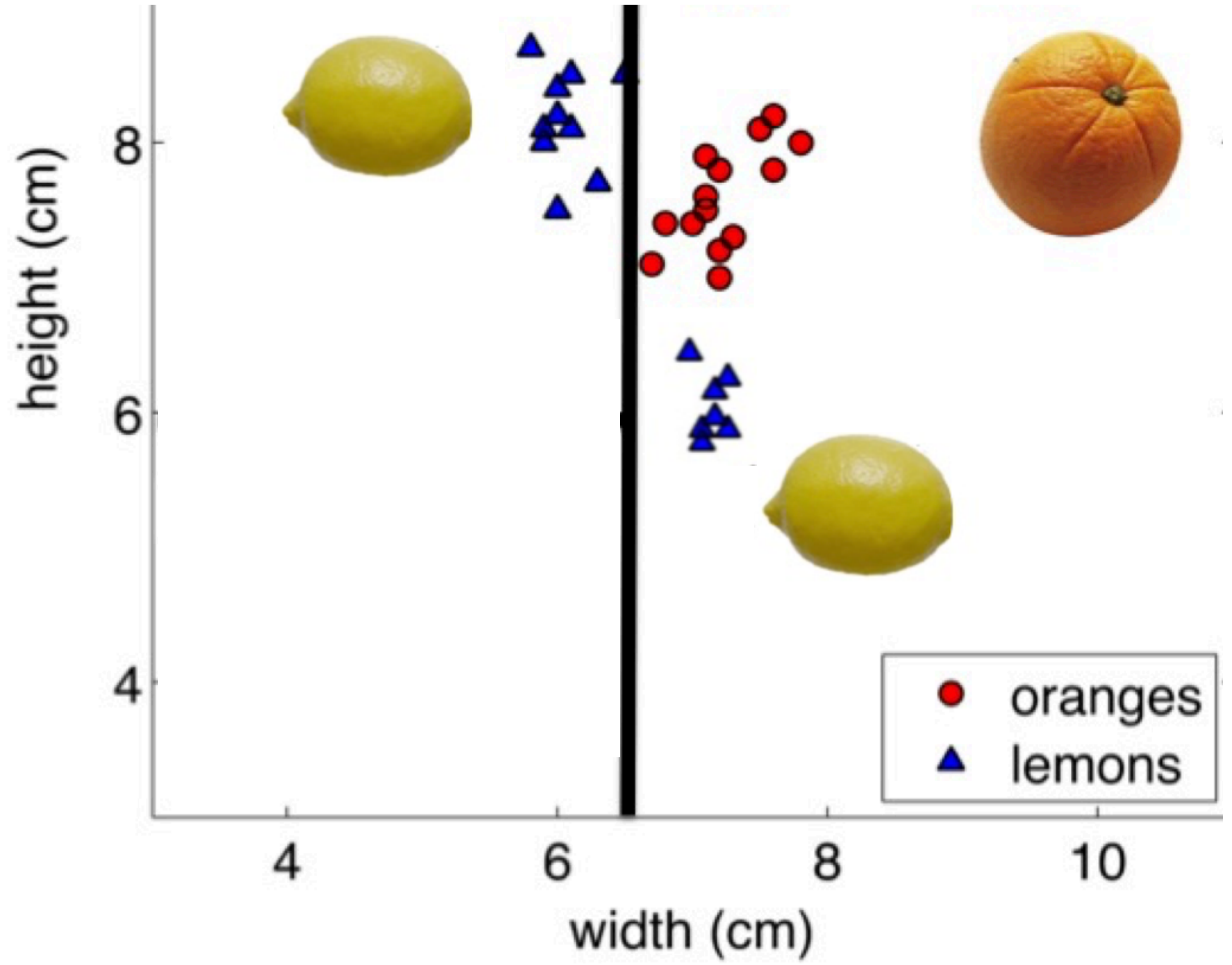
Training data:
100 Lemons
49 Oranges

Decision trees

How do we learn a decision tree?

Let's measure the accuracy gain: $L(R) - \dfrac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|} = \dfrac{49}{149} - \dfrac{50 \times 0 + 99 \times \frac{49}{99}}{149} = 0$





There is no accuracy gain, but this is actually a good split because we have **reduced our uncertainty** about whether a fruit is a lemon!

Decision trees

How do we learn a decision tree?

- A split is a good split if we are more certain about classification after the split. Meaning:
  - All samples in a leaf have the same class (deterministic) —> good (low uncertainty)
  - Each class has the same number of examples in a leaf (uniform distribution) —> bad (high uncertainty)
    What about distributions in between?

Idea: Use counts at leaves to define probability distribution and use **information theory** to measure uncertainty!

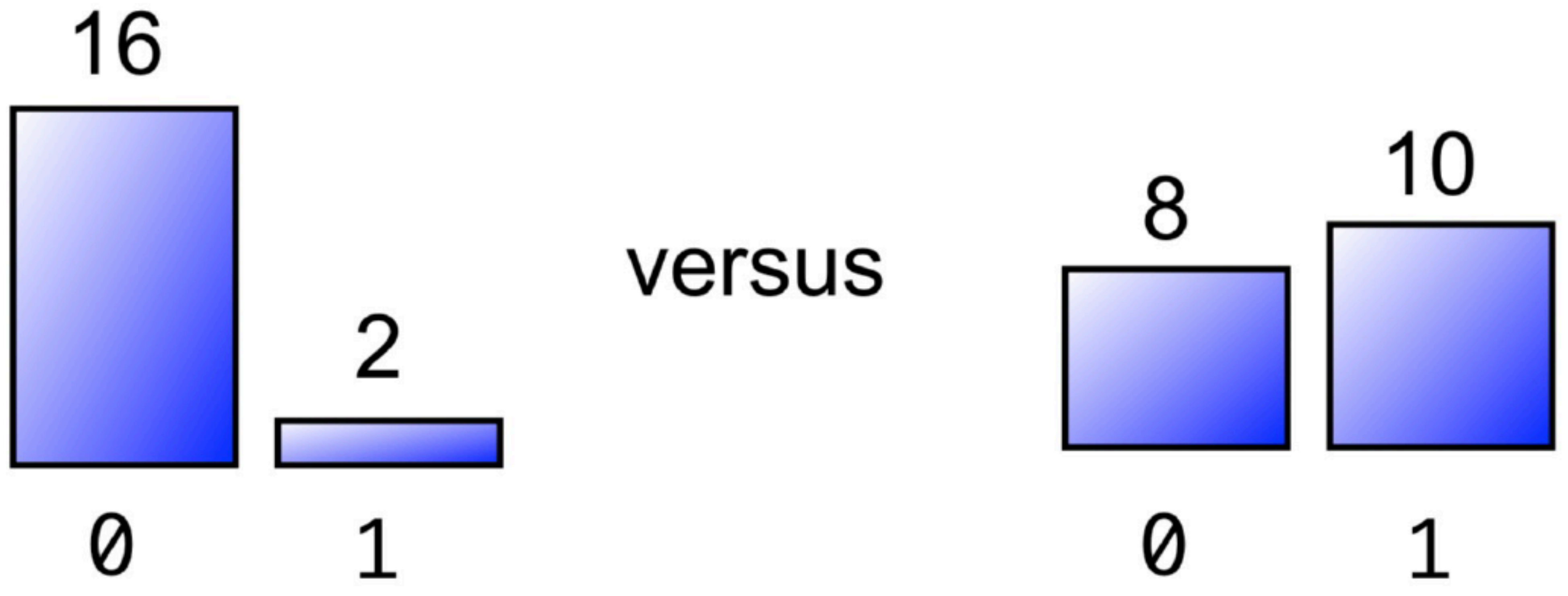
Decision trees

Entropy: A way to measure uncertainty

Which coin is more certain?

Sequence 1:
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ... ?

Sequence 2:
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?



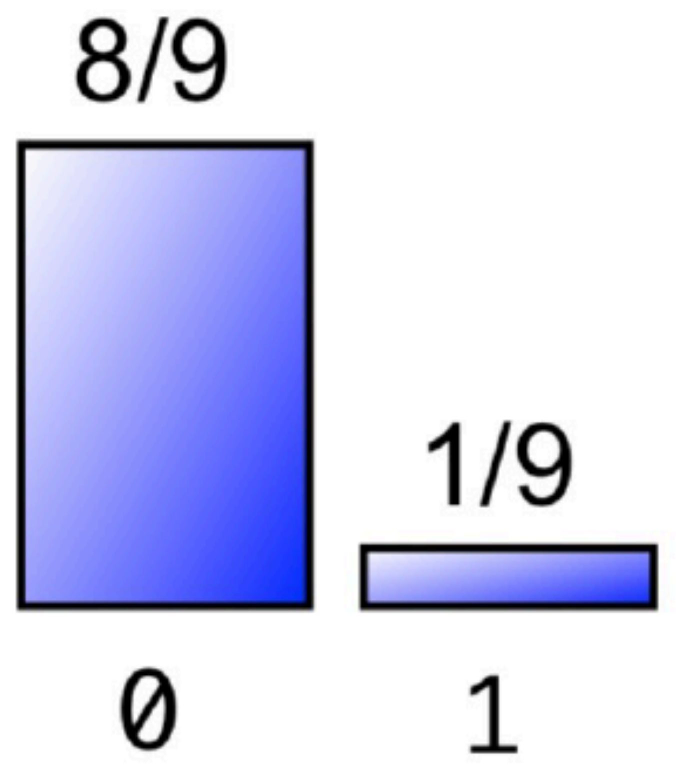
Decision trees

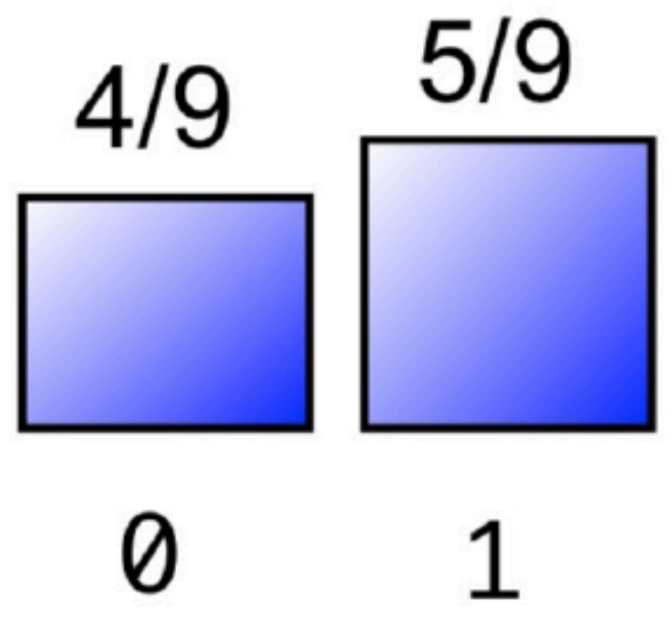
Entropy: A way to measure uncertainty

Entropy is a measure of expected surprise: How uncertain are we about the value of a draw from a distribution?

$$H(X) = -\mathbb{E}_{X\sim p}[\log_2 p(X)] = -\sum_{x\in X} p(x)\log_2 p(x)$$

Unit = bits





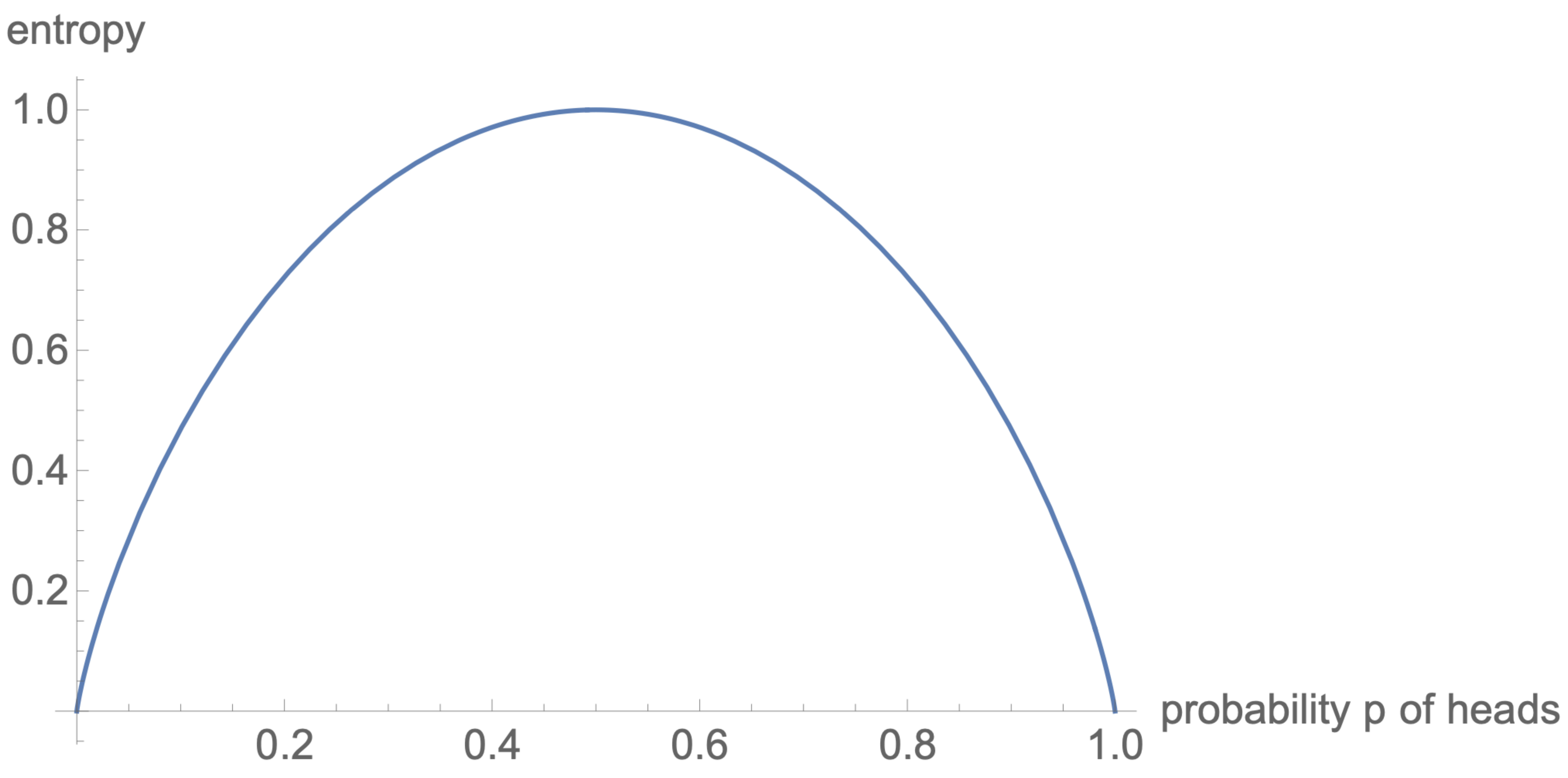
$$-\frac{8}{9}\log_2\frac{8}{9} - \frac{1}{9}\log_2\frac{1}{9} \approx \frac{1}{2}$$

$$-\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9} \approx 0.99$$

Decision trees

Entropy: A way to measure uncertainty

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$



Decision trees

Entropy: A way to measure uncertainty

- **High Entropy:**
  - Variable has a uniform like distribution
  - Flat histogram
  - Values sampled from it are less predictable
- **Low Entropy:**
  - Distribution of variable has many peaks and valleys
  - Histogram has many lows and highs
  - Values sampled from it are more predictable

Decision trees

Entropy: A way to measure uncertainty

Example: $X = \{\text{Raining, Not raining}\}$, $Y = \{\text{Cloudy, Not cloudy}\}$

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

$$
\begin{aligned}
H(X,Y) &= -\sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 p(x,y) \\
&= -\frac{24}{100} \log_2 \frac{24}{100} - \frac{1}{100} \log_2 \frac{1}{100} - \frac{25}{100} \log_2 \frac{25}{100} - \frac{50}{100} \log_2 \frac{50}{100} \\
&\approx 1.56\text{bits}
\end{aligned}
$$

Decision trees

Entropy: A way to measure uncertainty

Some useful properties:

- H is always non-negative
- Chainrule: $H(X, Y) = H(X|Y) + H(Y) = H(Y|X) + H(X)$
- If X and Y independent, then X doesn't tell us anything about Y: $H(Y|X) = H(Y)$
- But Y tells us everything about Y : $H(Y|Y) = 0$
- By knowing X , we can only decrease uncertainty about Y: $H(Y|X) \leq H(Y)$

Decision trees

Expected conditional entropy

What is the entropy of cloudiness, given the knowledge of whether or not it is raining?

$X = \{\text{Raining, Not raining}\},\ Y = \{\text{Cloudy, Not cloudy}\}$

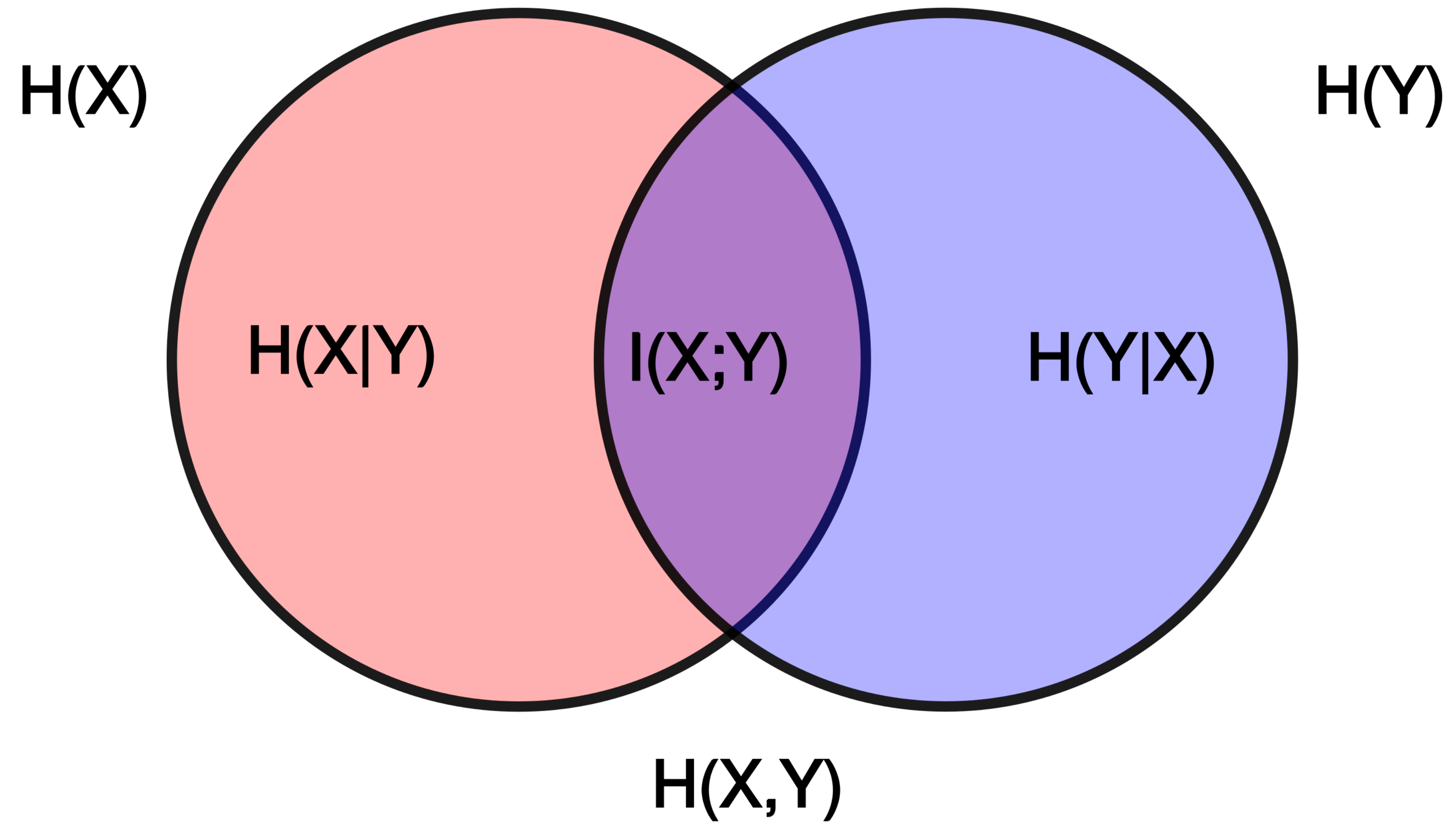
	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

$$
\begin{aligned}
H(Y|X) &= \mathbb{E}_{X\sim p(x)}[H(Y|X)] \\
&= \sum_{x \in X} p(x) H(Y|X=x) \\
&= -\sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 p(y|x) \\
&= -\mathbb{E}_{(X,Y)\sim p(x,y)}[\log_2 p(Y|X)] \\
&= \frac{1}{4} H(\text{cloudy}|\text{raining}) + \frac{3}{4} H(\text{cloudy}|\text{not raining}) \\
&\approx 0.75 \text{ bits}
\end{aligned}
$$

Decision trees

Information Gain (IG)

How much information about cloudiness do we get by discovering whether it is raining?



$$
\begin{aligned}
IG(Y|X) &= H(Y) - H(Y|X) \\
&\approx 0.25 \text{ bits}
\end{aligned}
$$

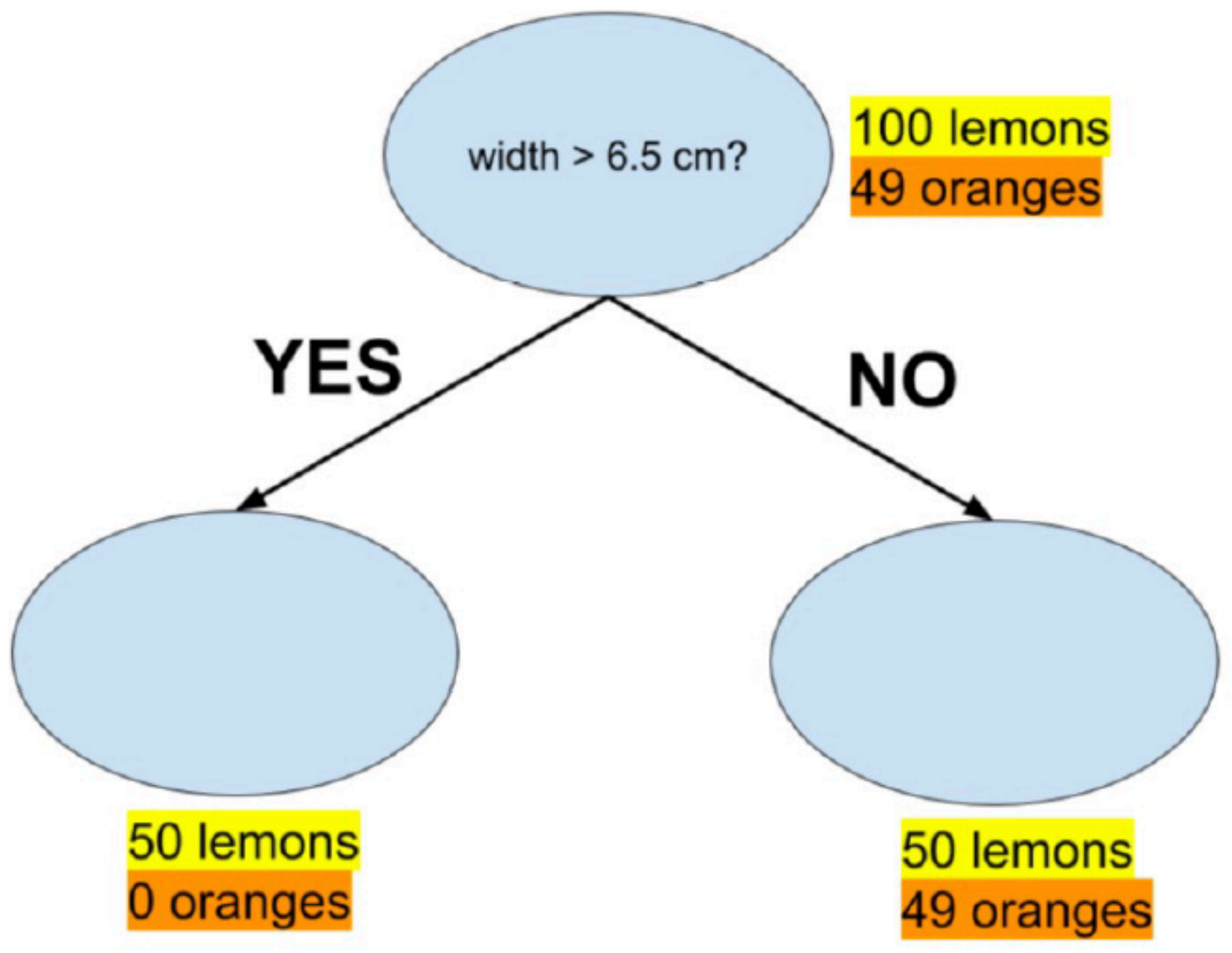
This is called the **information gain** in Y due to X, or the **mutual information** of Y and X

- If X is completely uninformative about Y : IG (Y |X) = 0
- If X is completely informative about Y: IG(Y|X) = H(Y)

Decision trees

Learn a tree with entropy

- Information gain measures the informativeness of a variable, which is exactly what we desire in a decision tree attribute!
- What is the information gain of this split?



Root entropy: $H(Y) = -\frac{49}{149}\log_2(\frac{49}{149}) - \frac{100}{149}\log_2(\frac{100}{149}) \approx 0.91$

Leafs entropy: $H(Y|left) = 0,\ H(Y|right) \approx 1.$

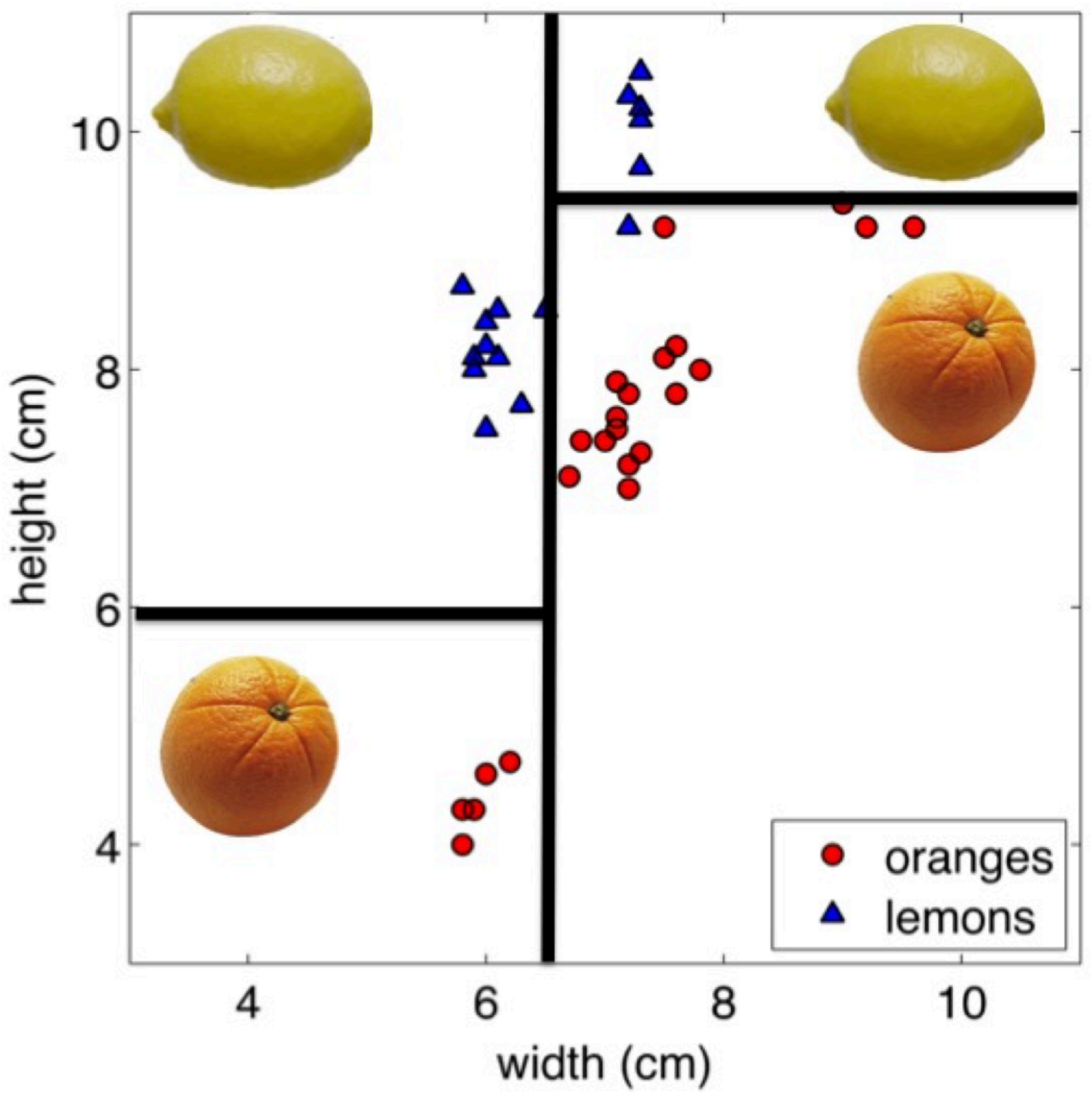
$IG(split) \approx 0.91 - (\frac{1}{3}\cdot 0 + \frac{2}{3}\cdot 1) \approx 0.24 > 0$

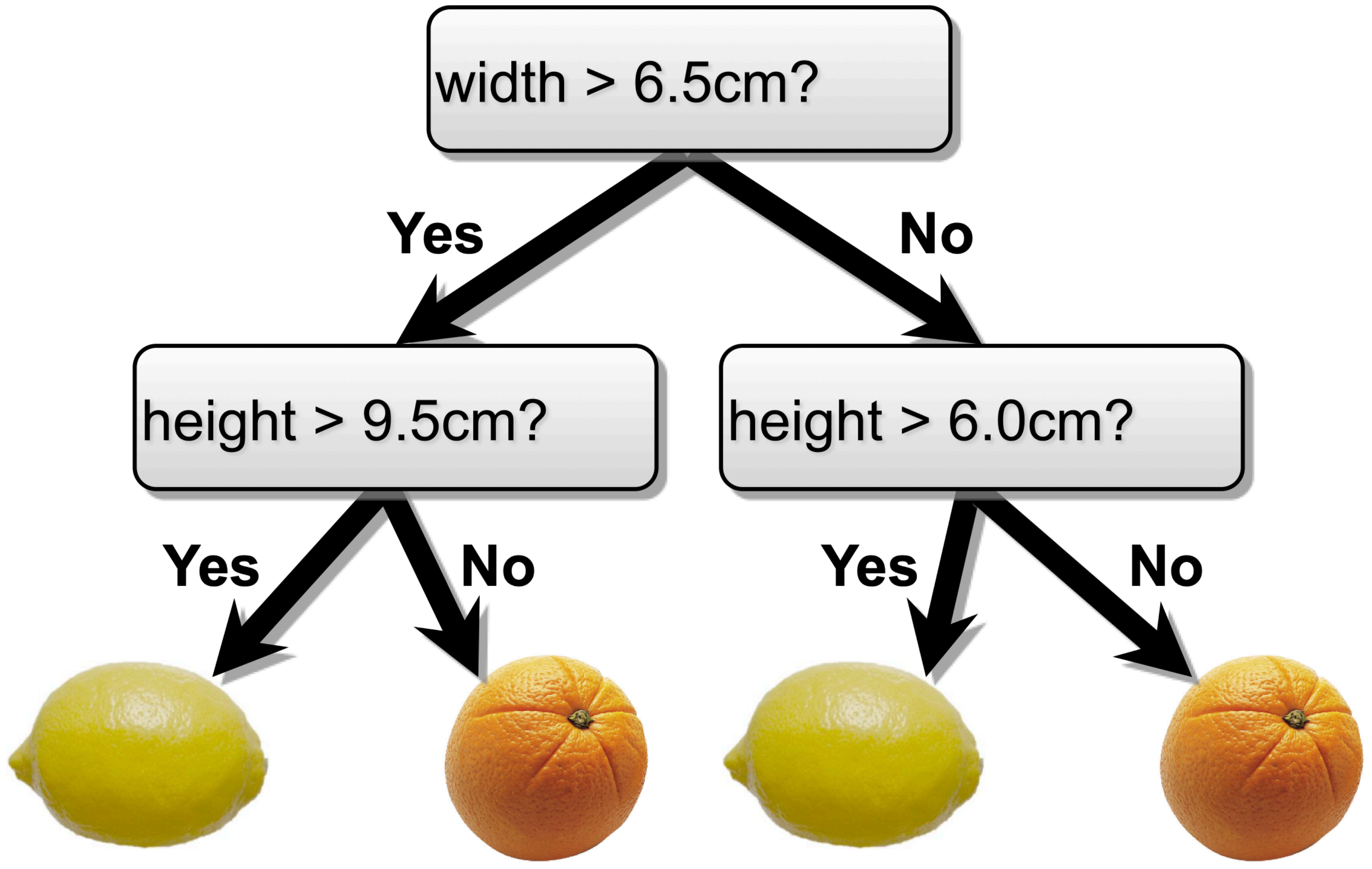
Decision trees

Learn a tree with entropy

At each level, one must choose:

1. Which variable to split.
2. Possibly where to split it.





Choose them based on how much information we would gain from the decision!
(choose attribute that gives the highest gain)

Decision trees

Learn a tree with entropy

- Simple, greedy, recursive approach, builds up tree node-by-node
- Starting with an empty tree and a training dataset:
  - Split on the most informative attribute, partitioning the dataset
  - Recurse on sub partitions
- Possible termination condition: End if all examples in current sub partition share the same class

Decision trees

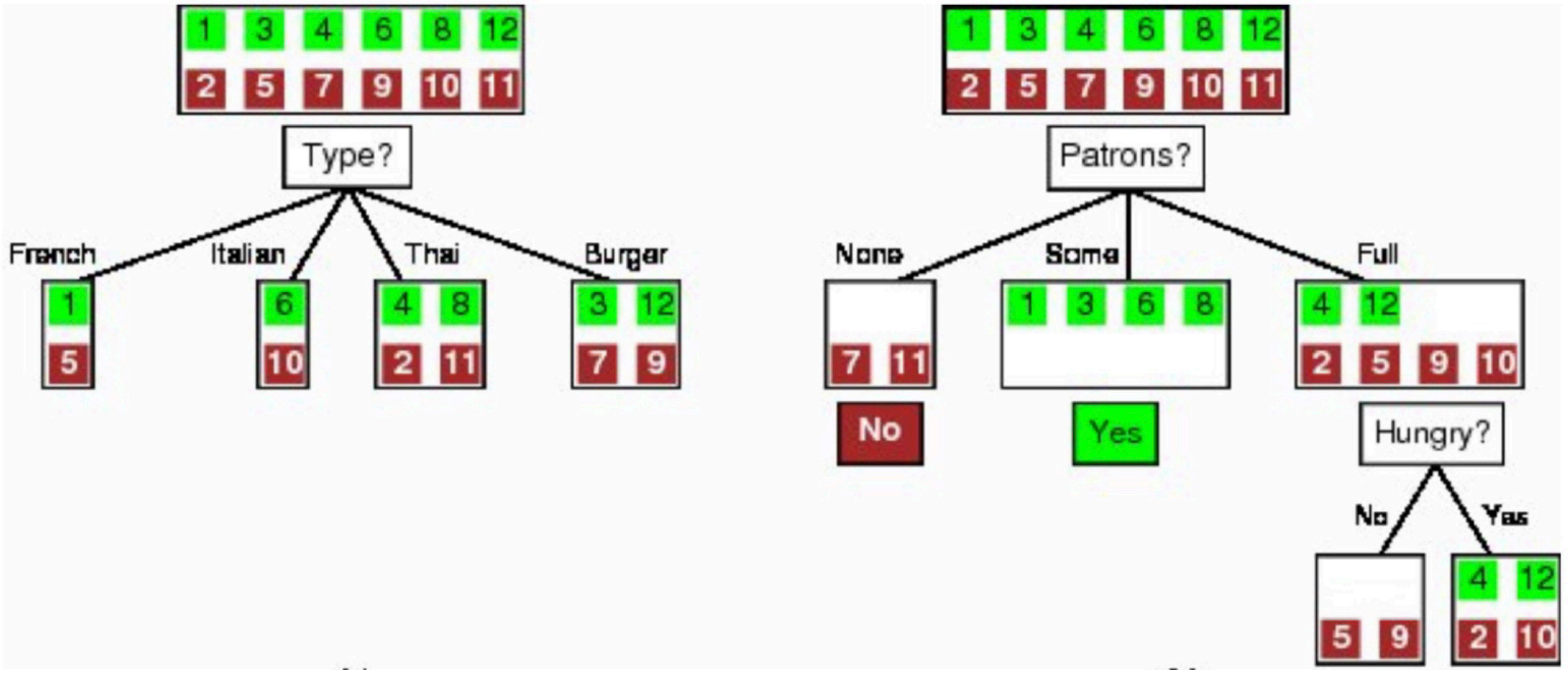
Let's go back to the restaurant example

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	$	No	No	Burger	0–10	$y_3 =$ Yes
$\mathbf{x}_4$	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10–30	$y_4 =$ Yes
$\mathbf{x}_5$	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	$y_5 =$ No
$\mathbf{x}_6$	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0–10	$y_6 =$ Yes
$\mathbf{x}_7$	No	Yes	No	No	None	$	Yes	No	Burger	0–10	$y_7 =$ No
$\mathbf{x}_8$	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0–10	$y_8 =$ Yes
$\mathbf{x}_9$	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	$y_9 =$ No
$\mathbf{x}_{10}$	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10–30	$y_{10} =$ No
$\mathbf{x}_{11}$	No	No	No	No	None	$	No	No	Thai	0–10	$y_{11} =$ No
$\mathbf{x}_{12}$	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30–60	$y_{12} =$ Yes

Decision trees

Let's go back to the restaurant example

Which attribute to choose?



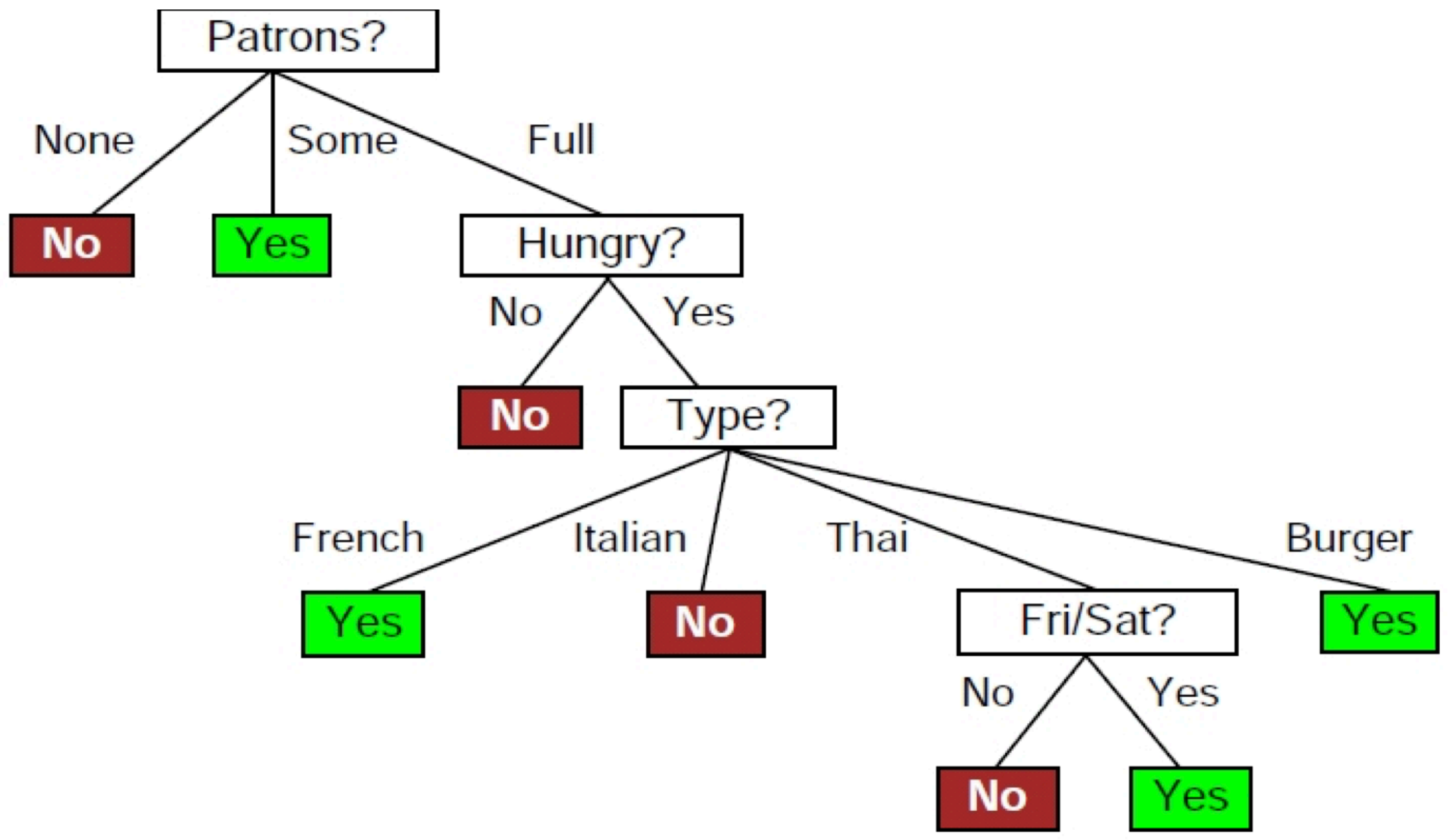
$$IG(Y) = H(Y) - H(Y|X)$$

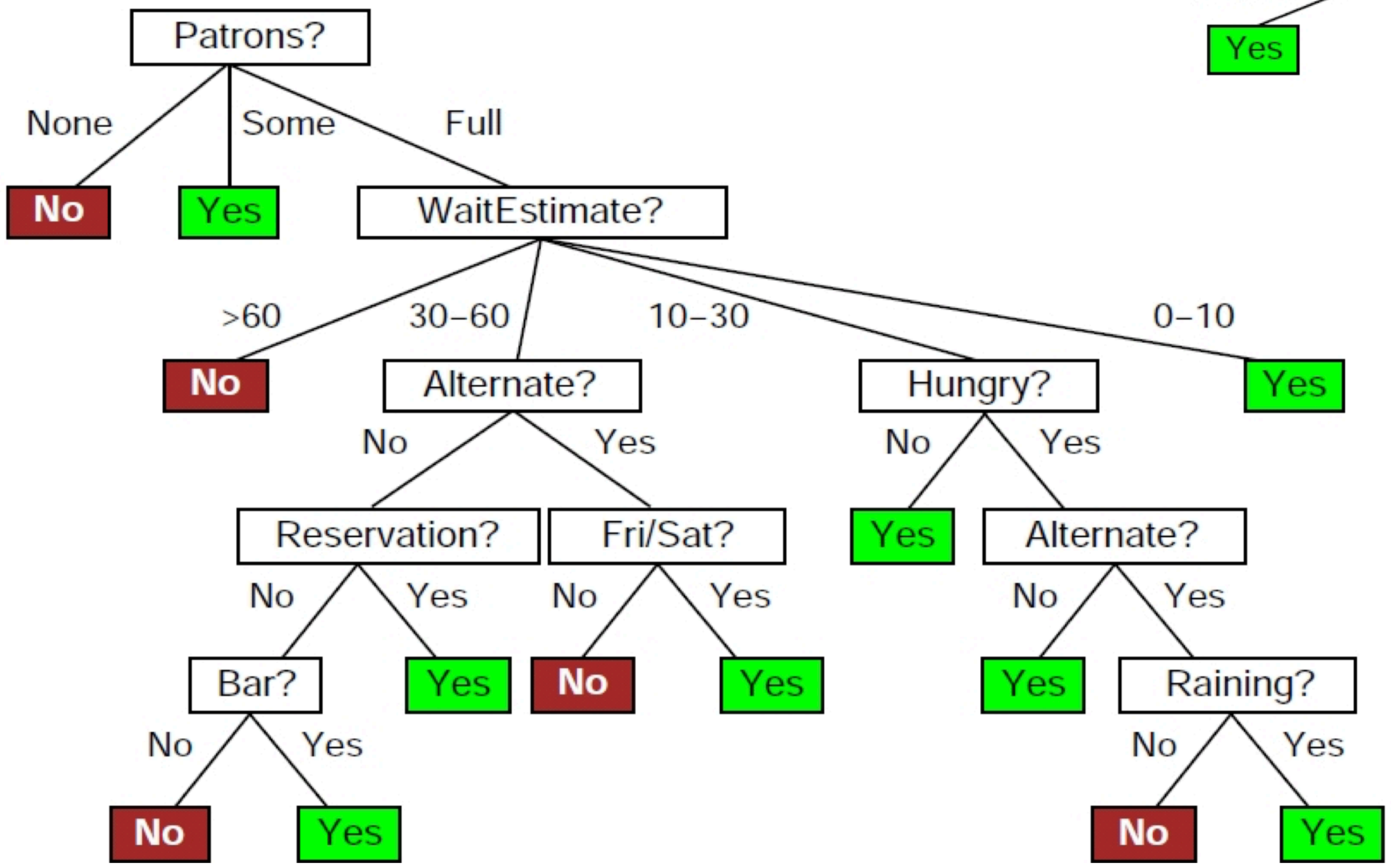
$$IG(type) = 1 - \left[\frac{2}{12}H(Y|\text{Fr.}) + \frac{2}{12}H(Y|\text{It.}) + \frac{4}{12}H(Y|\text{Thai}) + \frac{4}{12}H(Y|\text{Bur.})\right] = 0$$

$$IG(Patrons) = 1 - \left[\frac{2}{12}H(0,1) + \frac{4}{12}H(1,0) + \frac{6}{12}H(\frac{2}{6},\frac{4}{6})\right] \approx 0.541$$

Decision trees

Model selection: which tree is better?



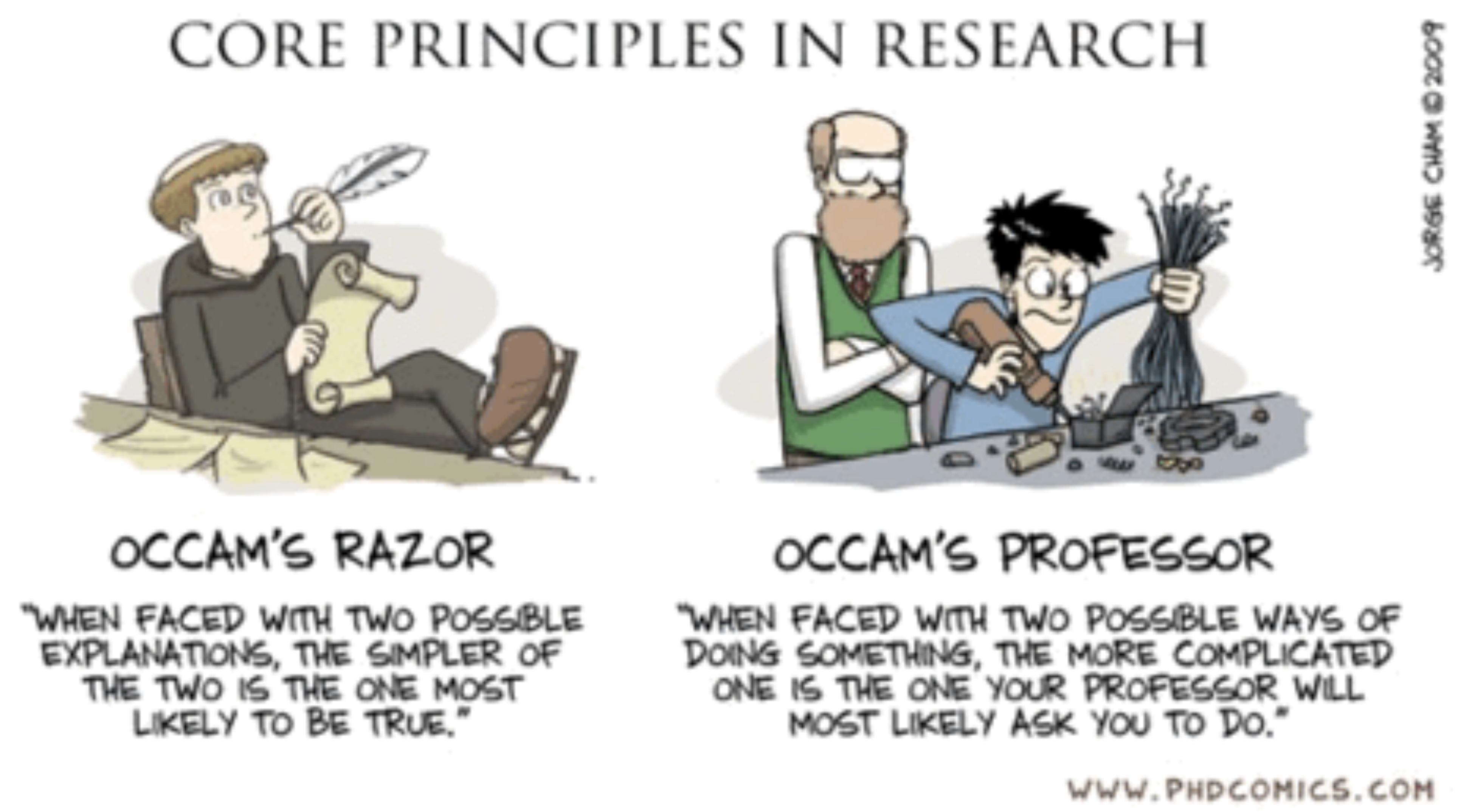


Decision trees

Model selection: which tree is better?

- Not too small: need to handle important but possibly subtle distinctions in data
- Not too big:
  - Computational efficiency (avoid redundant, spurious attributes)
  - Avoid over-fitting training examples
  - Human interpretability

- "**Occam's Razor**": find the simplest hypothesis that fits the observations
- We desire small trees with informative nodes near the root



Decision trees

Miscellany

- Handling continuous attributes: Split based on a threshold, chosen to maximize information gain.
- There are other criteria used to measure the quality of a split. E.g. Gini index
- Trees can be pruned to be made less complex.
- Decision trees can also be used for regression on real-valued outputs. Choose splits to minimize squared error, rather than maximize information gain.

Decision trees

Challenges

- You have exponentially less data at lower levels
- Too big of a tree can overfit the data
- Greedy algorithms don't necessarily yield the global optimum
- mistake at top level propagates down the tree

Do I need to do all this math when I build a decision tree for my data?

scikit learn

```
>>> from sklearn import tree
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, Y)
```

Decision trees Vs. KNN

Advantages of decision tree

- Good when there are lots of attributes, but only a few are important Good with discrete attributes
- Easily deals with missing values (just treat as another value)
- Robust to scale of inputs
- Fast at test time
- More interpretable

Fun fact: Decision tree is the most widely adopted method in healthcare

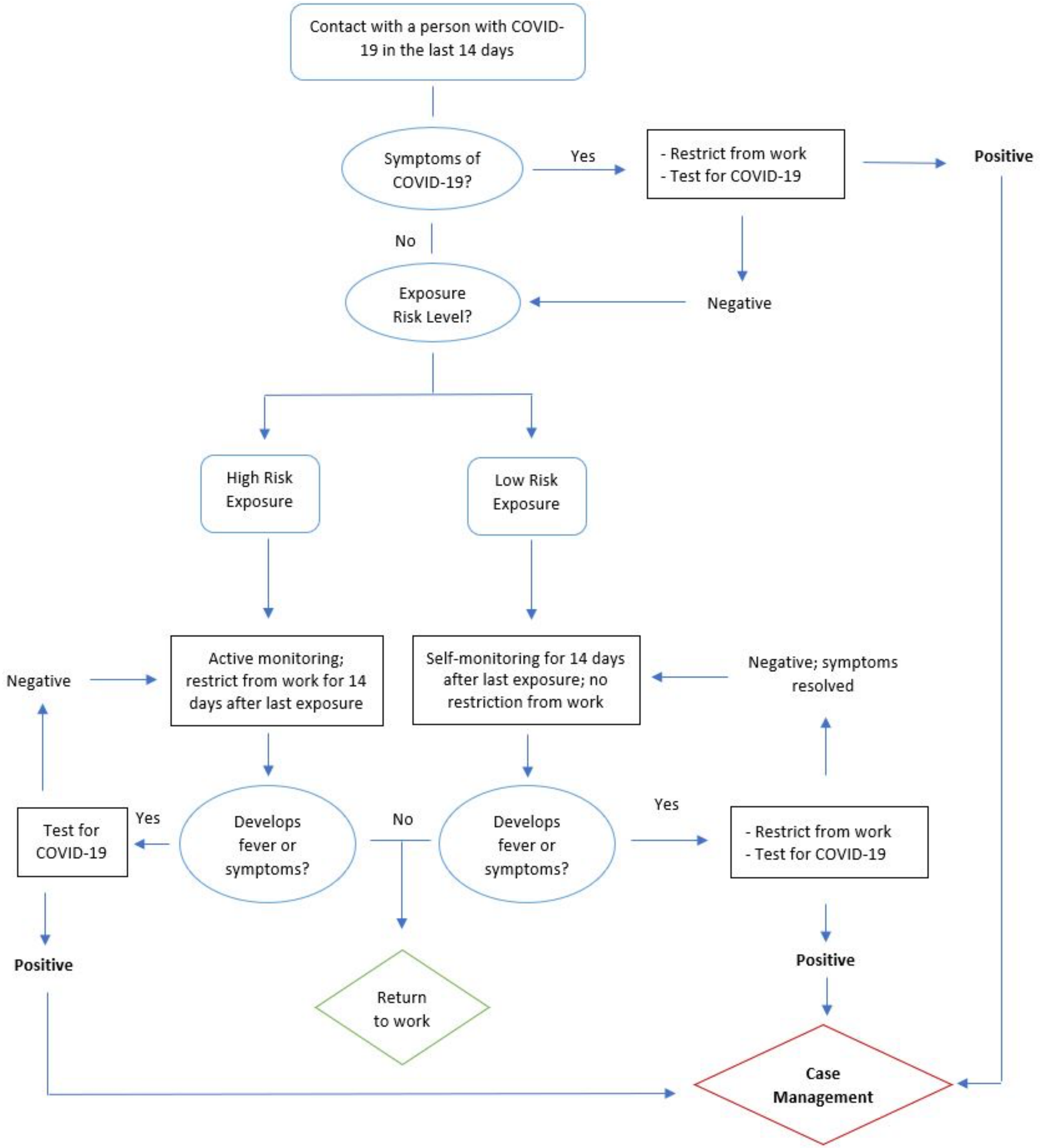
Decision trees Vs. KNN

Advantages of KNN

- Few hyper-parameters
- Able to handle attributes/features that interact in complex ways (e.g. pixels)
- Can incorporate interesting distance measures (e.g. shape contexts)

Can you think of an application of decision tree in healthcare?

Decision trees in covid management



Source: https://www.cdc.gov/

Decision trees

Summary

- Decision trees are simple and interpretable models
- Decision trees use information gain to learn to split the trees
- Decision trees can learn almost any function, but they tend to overfit.
- Next lecture:
    - Linear models

Break 

10 minutes

introduction to

Python