香港浸會大學
HONG KONG BAPTIST UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE
計算機科學系

COMP7015 Artificial Intelligence (S1, 2024-25)

Lecture 6: Multiclass Classification and Decision Trees

Instructor: Dr. Kejing Yin (cskjyin@hkbu.edu.hk)

Department of Computer Science
Hong Kong Baptist University

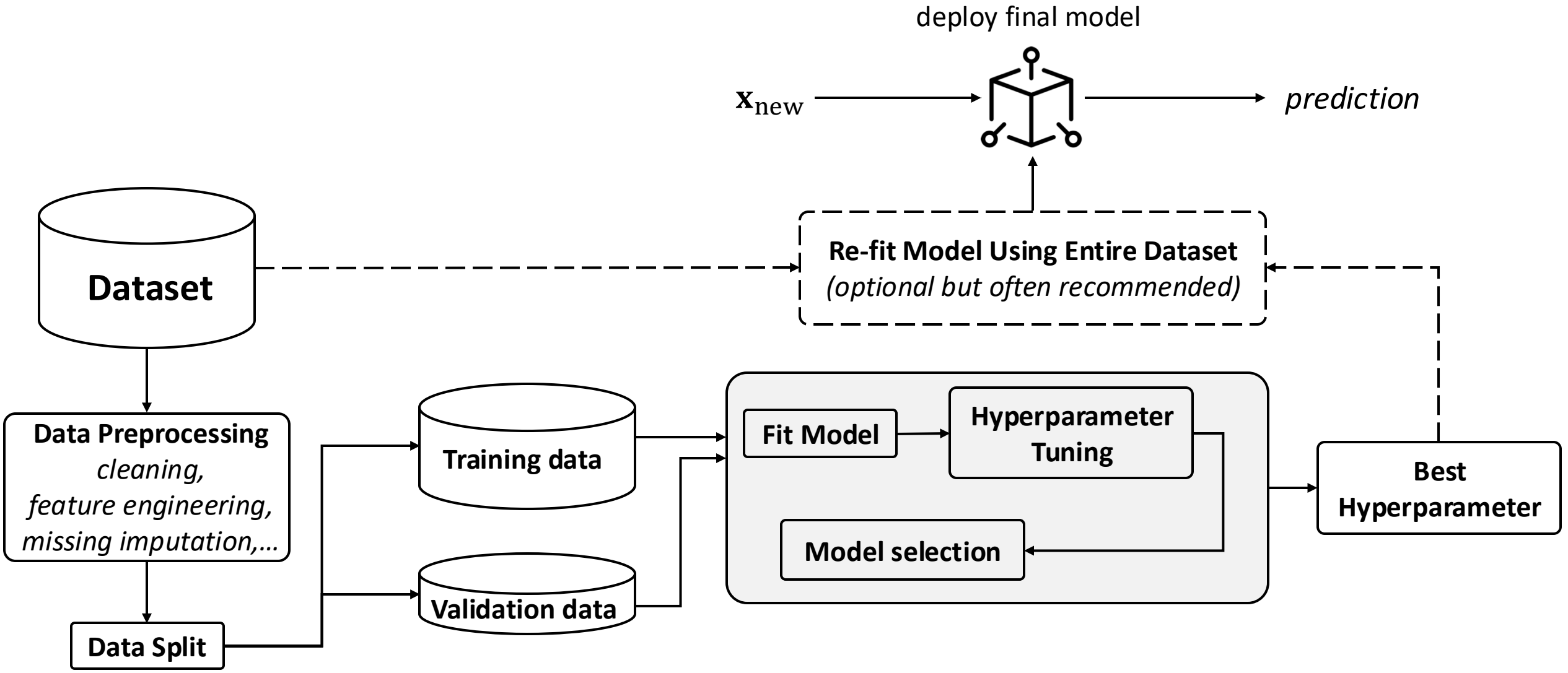
October 25, 2024

Announcements

- Lab 2: Machine learning with scikit-learn (Oct. 26)

- Course group registration deadline: Oct. 25 (tonight)!

Recap: The Typical Machine Learning Workflow

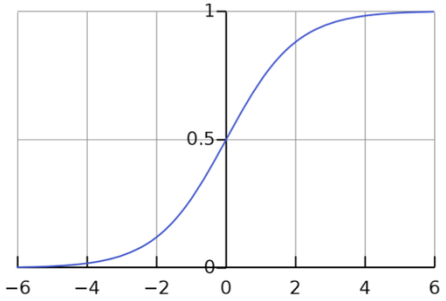


Multiclass Classification: Methods and Evaluation

Recap: Logistic Regression for Binary Classification

Consider this transfromation retruns a "probability":

- $p(y = +1|x) = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x})}$
- $p(y = -1|x) = 1 - \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{\exp(-\mathbf{w}^\top \mathbf{x})}{1+\exp(-\mathbf{w}^\top \mathbf{x})} = \frac{1}{1+\exp(+\mathbf{w}^\top \mathbf{x})}$



- The *log likelihood* (numerically more stable):

$$\ell(\mathbf{w}) = \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \mathbb{1}[y_i = +1] \log p(y_i = +1|\mathbf{x}) + \mathbb{1}[y_i = -1] \log p(y_i = -1|\mathbf{x})$$

- We can maximize the likelihood, or equivalently, minimize the negative likelihood:

$$\text{Loss}(\mathbf{x}, y, \mathbf{w}) = \log(1 + \exp(-(\mathbf{w}^\top \mathbf{x})y))$$

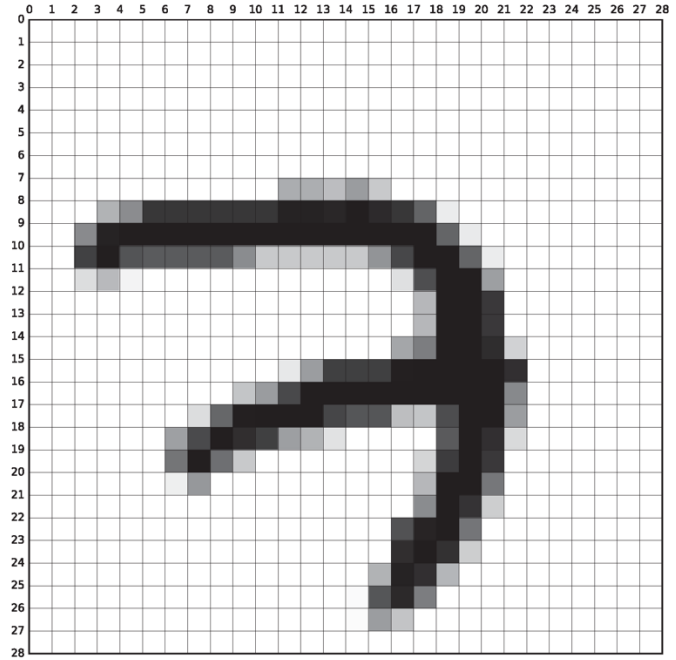
Features			Label
surface	texture	density	ripe?
hard	clear	0.77	yes
hard	clear	0.56	yes
hard	slightly blurry	0.64	no
hard	clear	0.61	yes
hard	clear	0.63	yes
hard	slightly blurry	0.66	no
hard	slightly blurry	0.67	no
hard	clear	0.44	yes
soft	clear	0.24	no
soft	clear	0.40	yes
hard	blurry	0.25	no
soft	blurry	0.34	no
soft	slightly blurry	0.48	yes

Binary class labels: Yes(+)/No(-)

Multiclass Classification Applications

- Hand-written digit recognition

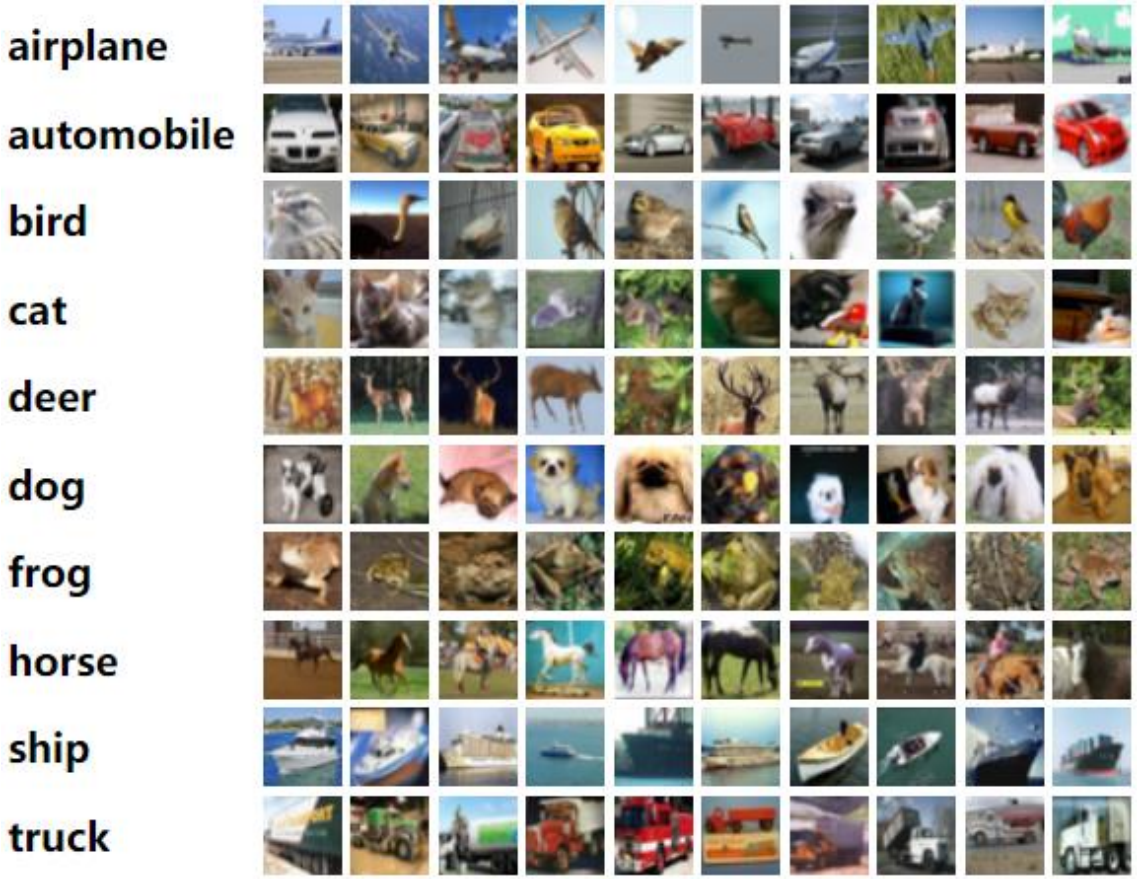
10 classes

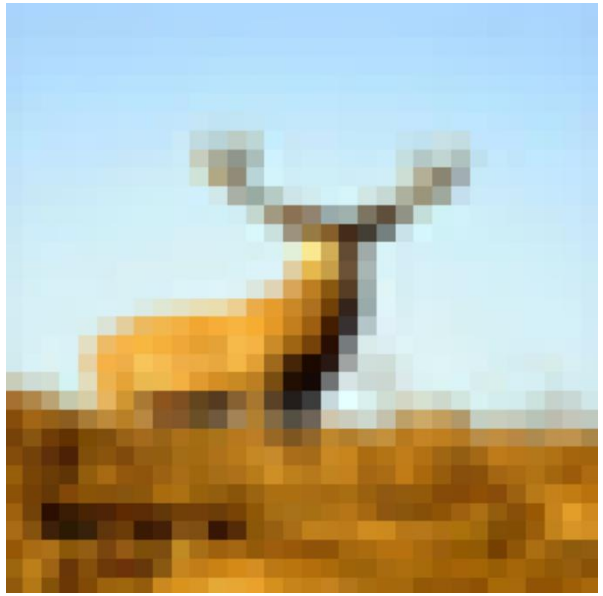


A sample in test set

Multiclass Classification Applications

- Image classification





A sample in test set

Multiclass Classification

- Some algorithms can be directly applied to multiclass classification
  - Example: Decision Tree

- More often, we extend binary classification methods to multiclass classification.

- Decomposition: dividing the multiclass classification problem into several binary classification problems.

```
ID3(D,X) =
  Let T be a new tree
  If all instances in D have same class c
    Label(T) = c; Return T
  If X = ∅ or no attribute has positive information gain
    Label(T) = most common class in D; return T
  X ← attribute with highest information gain
  Label(T) = X
  For each value x of X
    D_x ← instances in D with X = x
    If D_x is empty
      Let T_x be a new tree
      Label(T_x) = most common class in D
    Else
      T_x = ID3(D_x, X − {X})
    Add a branch from T to T_x labeled by x
  Return T
```

Decomposition-based Multiclass Classification

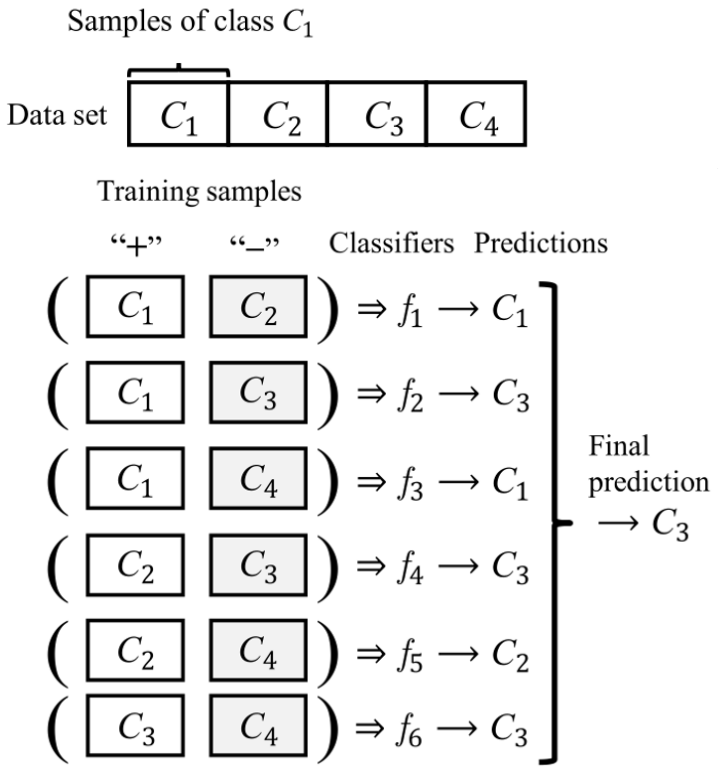
- Consider N classes: C_1, C_2, …, C_N.
- Decomposition: dividing the multiclass classification problem into several binary classification problems.
- In testing phase, we ensemble the outputs collected from all binary classifiers into the final multiclass predictions.
- **Key question: How to divide and how to ensemble.**
- We focus on two classical dividing strategies:
  One vs One (OvO) & **One vs Rest (OvR)**

Decomposition-based Multiclass Classification

- Given a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, where $y_i \in \{C_1, C_2, \ldots, C_N\}$
- One vs One (OvO) puts the N classes into pairs: in total $N(N-1)/2$ binary classification problems.

Example: OvO trains a classifier to distinguish class C_i and C_j , where it regards C_i as positive and C_j as negative.

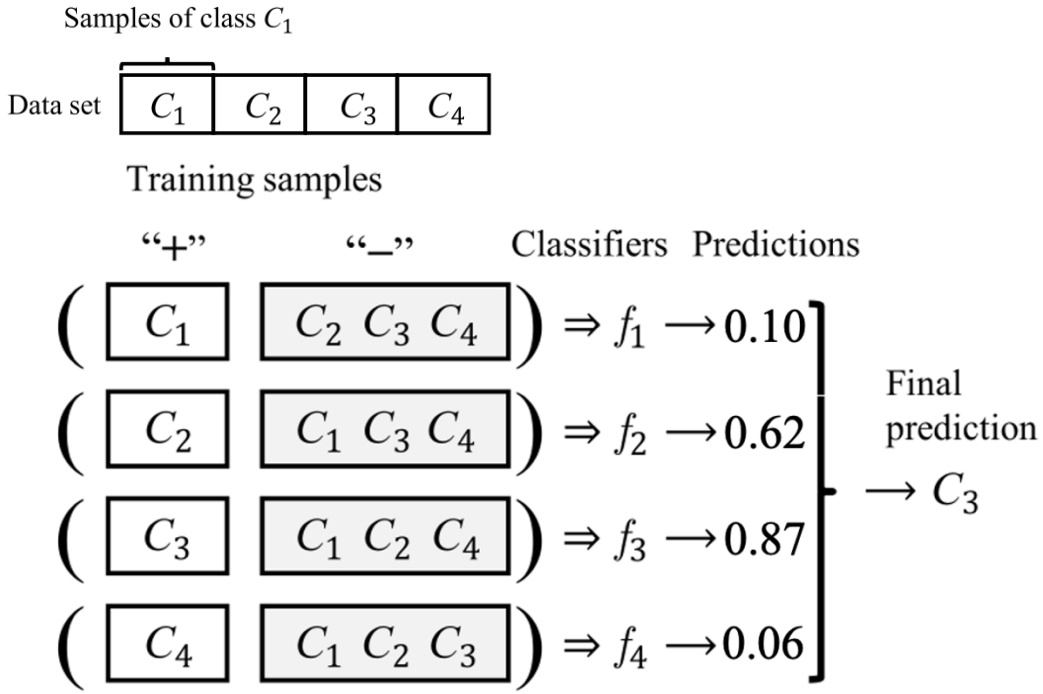
OvO strategy



Decomposition-based Multiclass Classification

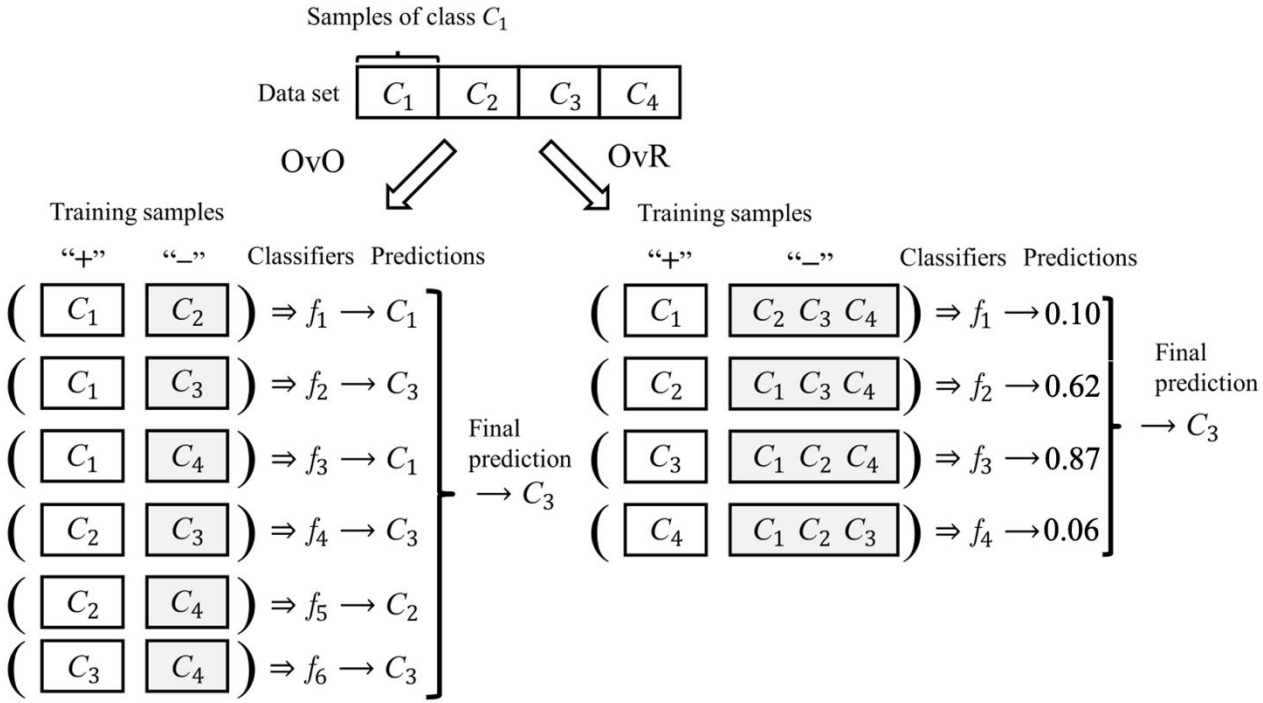
- Given a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, where $y_i \in \{C_1, C_2, \ldots, C_N\}$
- OvR trains N classifiers by considering each class as positive in turn, and the rest classes are considered as negative.

OvR strategy



Decomposition-based Multiclass Classification

- Given a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, where $y_i \in \{C_1, C_2, \ldots, C_N\}$



- OvR trains only N classifier but OvO trains $N(N-1)/2$ classifier.
- OvO is usually more expansive in storage and test time.
- In training, OvO uses data of two classes, but OvR uses all data. Usually, OvO can be trained faster.
- Performance are usually similar.

Evaluation Methods for Multiclass Classification

- Confusion matrix for multiclass classification

No	Actual	Predicted
1	Airplane	Airplane
2	Car	Boat
3	Car	Car
4	Car	Car
5	Car	Boat
6	Airplane	Boat
7	Boat	Boat
8	Car	Airplane
9	Airplane	Airplane
10	Car	Car

Three labels: Airplane, Car, Boat

		Predicted		
		Airplane	Boat	Car
Actual	Airplane	2	1	0
	Boat	0	1	0
	Car	1	2	3

Number of images of cars being predicted as boat

Confusion matrix: rows are actual labels (ground truth), columns are predictions

Example and figures from https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

Evaluation Methods for Multiclass Classification

- We can compute the metrics for each class

		Predicted		
		Airplane	Boat	Car
Actual	Airplane	2	1	0
	Boat	0	1	0
	Car	1	2	3

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	F1 Score
Airplane	2	1	1	0.67	0.67	2 * (0.67 * 0.67) / (0.67 + 0.67) = **0.67**
Boat	1	3	0	0.25	1.00	2*(0.25 * 1.00) / (0.25 + 1.00) = **0.40**
Car	3	0	3	1.00	0.50	2 * (1.00 * 0.50) / (1.00 + 0.50) = **0.67**

Number of images of other classes being predicted as boat

How to evaluate the overall performance for this multiclass classification model as a whole?

Taking averages!

- Macro average
- Weighted average
- Micro average

Example and figures from https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

Evaluation Methods for Multiclass Classification

- Macro Average

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	F1 Score
Airplane	2	1	1	0.67	0.67	2 * (0.67 * 0.67) / (0.67 + 0.67) = **0.67**
Boat	1	3	0	0.25	1.00	2*(0.25 * 1.00) / (0.25 + 1.00) = **0.40**
Car	3	0	3	1.00	0.50	2 * (1.00 * 0.50) / (1.00 + 0.50) = **0.67**

Label	Per-Class F1 Score	Macro-Averaged F1 Score
Airplane	0.67	$\frac{0.67 + 0.40 + 0.67}{3}$ = **0.58**
Boat	0.40	
Car	0.67	

Taking the average of per-class metrics (Precision, Recall, and F1).

$$\text{Macro-Precision} = \frac{\text{Precision}_1 + \text{Precision}_2 + \cdots + \text{Precision}_N}{N}$$

$$\text{Macro-Recall} = \frac{\text{Recall}_1 + \text{Recall}_2 + \cdots + \text{Recall}_N}{N}$$

$$\text{Macro-F1} = \frac{\text{F1}_1 + \text{F1}_2 + \cdots + \text{F1}_N}{N}$$

Example and figures from https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

Evaluation Methods for Multiclass Classification

- Weighted Average

  Taking the weighted average of per-class metrics (Precision, Recall, and F1).

		Predicted		
		Airplane	Boat	Car
Actual	Airplane	2	1	0
	Boat	0	1	0
	Car	1	2	3

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	F1 Score
Airplane	2	1	1	0.67	0.67	2 * (0.67 * 0.67) / (0.67 + 0.67) = **0.67**
Boat	1	3	0	0.25	1.00	2*(0.25 * 1.00) / (0.25 + 1.00) = **0.40**
Car	3	0	3	1.00	0.50	2 * (1.00 * 0.50) / (1.00 + 0.50) = **0.67**

Label	Per-Class F1 Score	Support	Support Proportion	Weighted Average F1 Score
Airplane	0.67	3	0.3	$(0.67 * 0.3) + (0.40 * 0.1) + (0.67 * 0.6)$ = **0.64**
Boat	0.40	1	0.1	
Car	0.67	6	0.6	
Total	-	10	1.0	

frequency of actual occurrence of each class

Weighted-Precision = $p_1 \text{Precision}_1 + \cdots + p_N \text{Precision}_N$

Weighted-Recall = $p_1 \text{Recall}_1 + \cdots + p_N \text{Recall}_N$

Weighted-F1 = $p_1 \text{F1}_1 + \cdots + p_N \text{F1}_N$

Example and figures from https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

Evaluation Methods for Multiclass Classification

- Micro Average

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	F1 Score
Airplane	2	1	1	0.67	0.67	2 * (0.67 * 0.67) / (0.67 + 0.67) = **0.67**
Boat	1	3	0	0.25	1.00	2*(0.25 * 1.00) / (0.25 + 1.00) = **0.40**
Car	3	0	3	1.00	0.50	2 * (1.00 * 0.50) / (1.00 + 0.50) = **0.67**

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Micro-Averaged Values
Airplane	2	1	1	Precision = $\frac{6}{6+4}$ = **0.60**
Boat	1	3	0	Recall = $\frac{6}{6+4}$ = **0.60**
Car	3	0	3	F1 Score = $\frac{6}{6+\frac{1}{2}(4+4)}$ = **0.60**
TOTAL	**6**	**4**	**4**	

Add up the confusion matrix for each binary classification and compute the metrics

$$\text{Micro-Precision} = \frac{TP_1+\cdots+TP_N}{(TP_1+FP_1)+\cdots+(TP_N+FP_N)}$$

$$\text{Micro-Recall} = \frac{TP_1+\cdots+TP_N}{(TP_1+FN_1)+\cdots+(TP_N+FN_N)}$$

$$\text{Micro-F1} = \frac{2 \times \text{Micro-Precision} \times \text{Micro-Recall}}{\text{Micro-Precision} + \text{Micro-Recall}}$$

Example and figures from https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

Evaluation Methods for Multiclass Classification

- Which one to use?
  - Macro averaged: all classes are equally important.
  - Weighted averaged: assigns greater weighting to classes with more samples in the dataset.
  - Micro averaged: overall performance regardless of the class.

Feature Engineering

Feature Engineering

- Recall from the summary of the linear models:

$$\underbrace{\mathbf{w} \cdot \boxed{\phi(x)}}_{\text{score}}$$

Two components: $\phi(x)$ and $\mathbf{w}$

	Regression	Classification
Prediction $f_{\mathbf{w}}(x)$	score	sign(score)
Relate to target y	residual (score $- y$)	margin (score y)
Loss functions	squared absolute deviation	zero-one hinge logistic
Algorithm	gradient descent	gradient descent

So far, we just used $\phi(x) = x$ and focused on learning of $\mathbf{w}$.

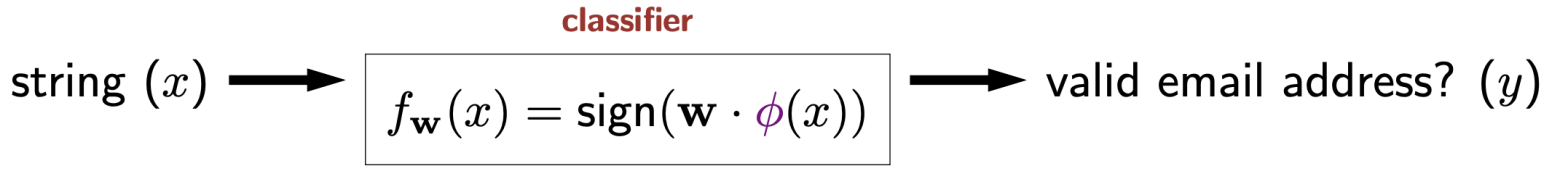
Another important part of ML pipeline:

Feature extraction (feature engineering) specifies $\phi(x)$ based on some domain knowledge.

Often the bottleneck for real applications

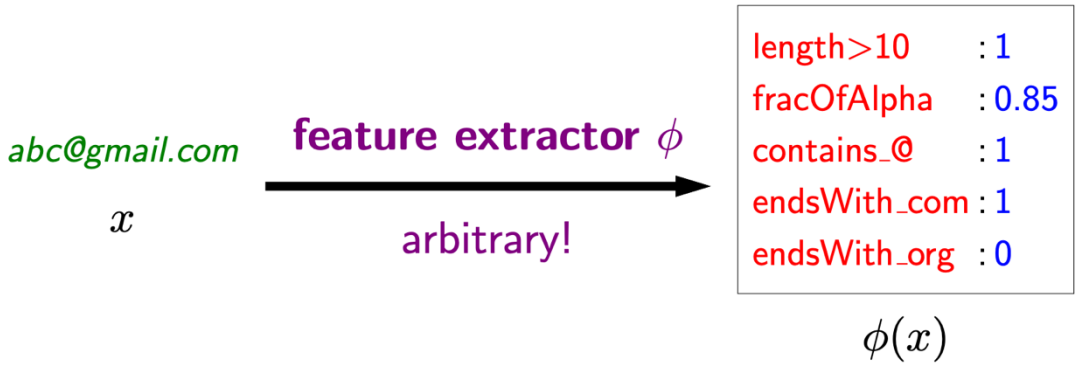
Feature Engineering

- An example task: predict whether a string is an email address



Questions: what properties of x might be relevant for predicting y?

Feature extractor ϕ produces (feature name, feature value) pairs



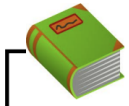
What features to include?
Need some organizational principles.

Feature engineering is sort of an "art".

Example and figures from https://stanford-cs221.github.io/spring2022-extra/modules/machine-learning/nonlinear-and-neural.pdf

Feature Engineering

- Feature templates



Definition: feature template

A **feature template** is a group of features all computed in a similar way.

abc@gmail.com → last three characters equals ___

endsWith_aaa : 0
endsWith_aab : 0
endsWith_aac : 0
...
endsWith_com : 1
...
endsWith_zzz : 0

Define types of pattern to look for, not particular patterns

Example and figures from https://stanford-cs221.github.io/spring2022-extra/modules/machine-learning/nonlinear-and-neural.pdf

Feature Engineering

- Feature templates Example

Latitude: 37.4068176

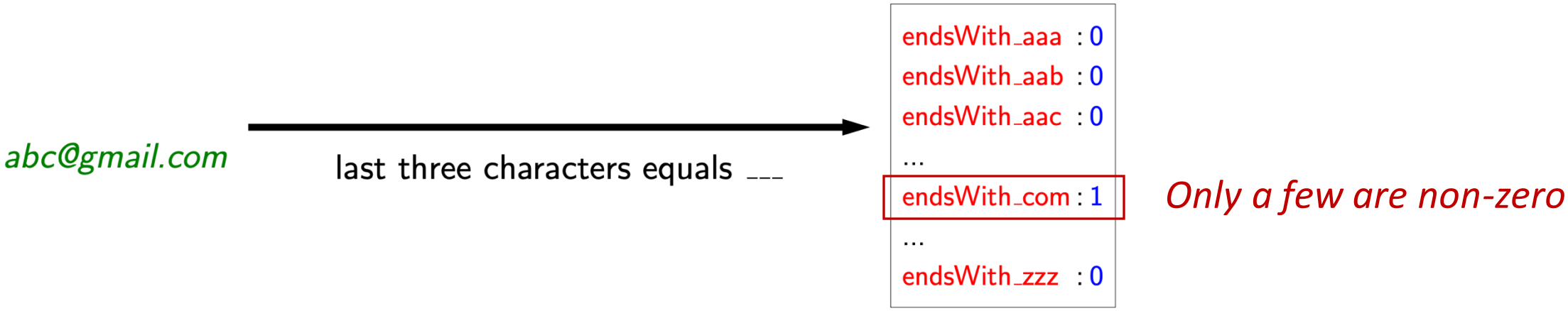
Longitude: -122.1715122

Feature template	Example feature name
Pixel intensity of image at row ___ and column ___ (___ channel)	Pixel intensity of image at row *10* and column *93* (*red* channel) : 0.8
Latitude is in [___, ___] and longitude is in [___, ___]	Latitude is in [*37.4*, *37.5*] and longitude is in [*-122.2*, *-122.1*] : 1

Example and figures from https://stanford-cs221.github.io/spring2022-extra/modules/machine-learning/nonlinear-and-neural.pdf

Sparse Features

- Dictionary is more efficient to represent sparse features (few non-zeros)



A more compact way to represent such sparse features:

(dictionary) {"endsWith_com": 1}

Instead of an array: [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]

Example and figures from https://stanford-cs221.github.io/spring2022-extra/modules/machine-learning/nonlinear-and-neural.pdf

Hypothesis Space

- Which feature extractor ϕ to use?

  - Recall that a hypothesis space (*a.k.a.* hypothesis class) is the set of possible predictors with a fixed $\phi(x)$ and varying w:

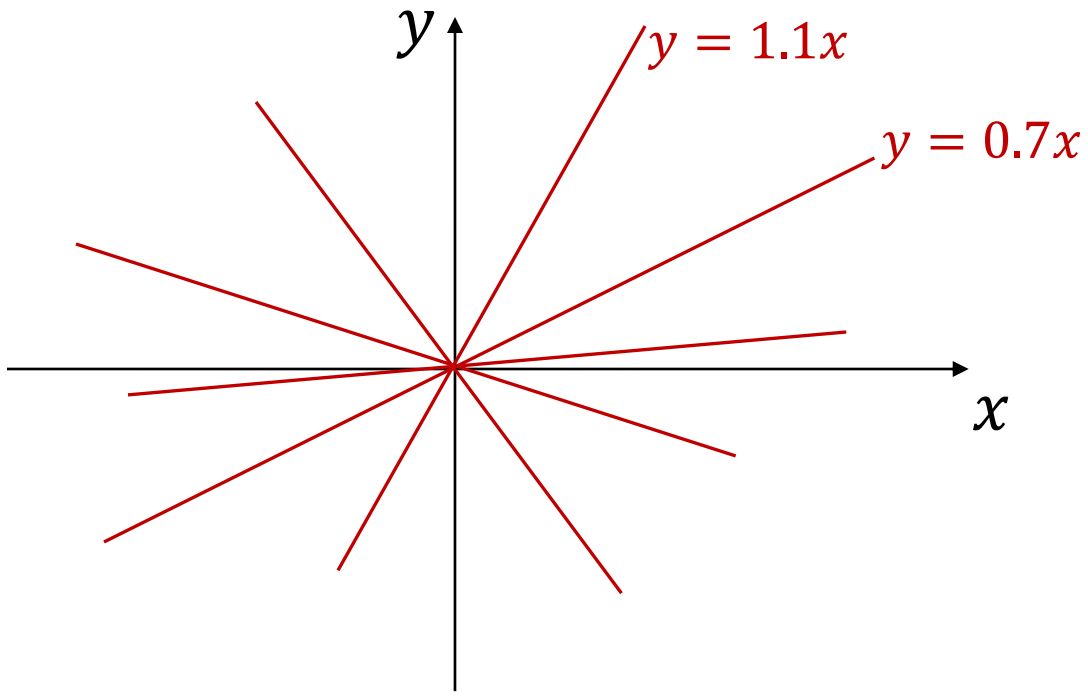
$$\mathcal{H} = \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^d\}$$

  where $f_{\mathbf{w}}(x) = \mathbf{w}^{\mathrm{T}}\phi(x)$ or $\mathrm{sign}(\mathbf{w}^{\mathrm{T}}\phi(x))$ is the predictor.

Hypothesis Space

- Example of hypothesis space:
  - Linear regression: $x \in \mathbb{R}$ and $y \in \mathbb{R}$
  - Linear function: $\phi(x) = x$, the hypothesis class is $\mathcal{H}_1 = \{w_1 x : w_1 \in \mathbb{R}\}$

All lines go through the origin



Constructing a feature vector:

Considering the hypothesis class defined by this feature map.

Hypothesis Space

- Example of hypothesis space:
  - Linear regression: $x \in \mathbb{R}$ and $y \in \mathbb{R}$
  - Linear function: $\phi(x) = x$, the hypothesis class is $\mathcal{H}_1 = \{w_1 x : w_1 \in \mathbb{R}\}$

All lines go through the origin

  - Quadratic function: $\phi(x) = [x, x^2]$, the hypothesis class is

$$\mathcal{H}_2 = \{w_1 x + w_2 x^2 : w_1 \in \mathbb{R}, w_2 \in \mathbb{R}\}$$

All quadratic functions that go through the origin
Also includes all linear functions

- $\mathcal{H}_2$ is a larger set than $\mathcal{H}_1$: more expressive (can represent more things).
- The feature map ϕ defnes the hypothesis class.

Hypothesis Space

- Prediction is driven by score: $\mathbf{w}^T \phi(x)$
  - Is score linear in $\boldsymbol{w}$? *Yes!*
  - Is score linear in $\phi(x)$? *Yes!*
  - Is score linear in x? *Not necessarily!* E.g., $\phi(x) = [x, x^2]$

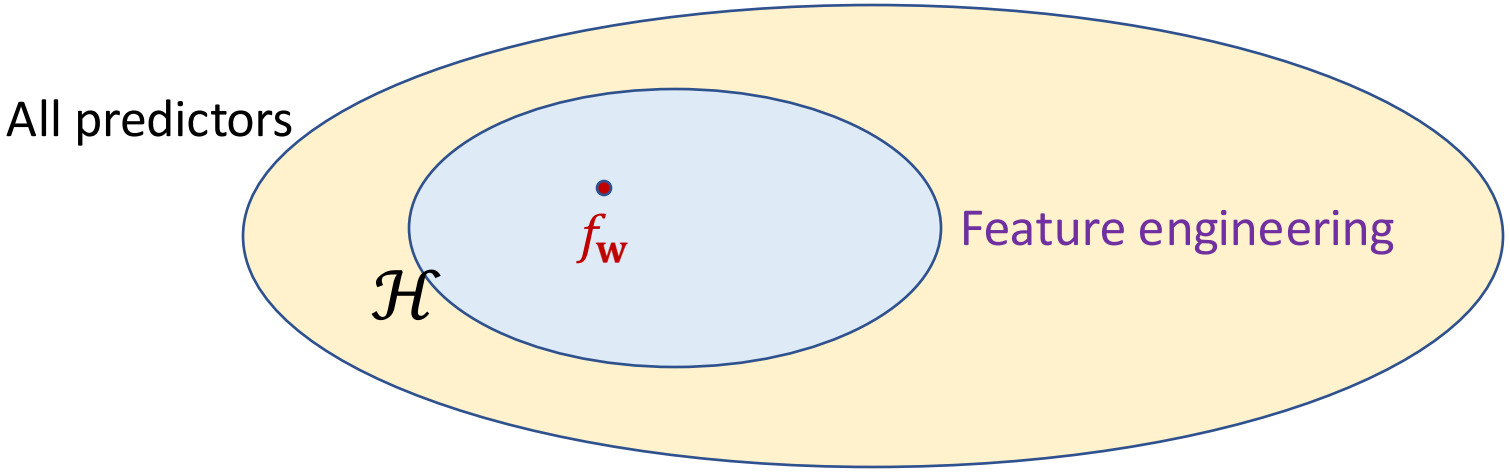
- Predictor $f_{\mathbf{w}}(x)$ can be expressive non-linear functions and decision boundaries of x.

  online demos: https://playground.tensorflow.org/
  https://www.youtube.com/watch?v=3liCbRZPrZA

- Score $\mathbf{w}^T \phi(x)$ is linear function of $\mathbf{w}$, which allows efficient learning.

Feature Engineering + Learning

$$\mathcal{H} = \{f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w}^{\mathrm{T}}\phi(x)) : \mathbf{w} \in \mathbb{R}^d\}$$

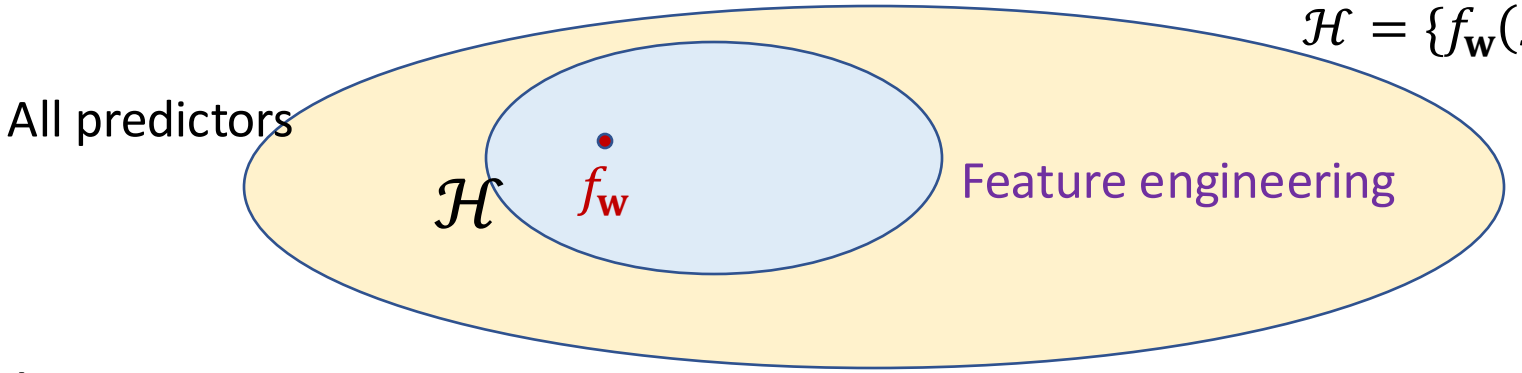


- Feature engineering/extraction: choose $\mathcal{H}$ based on domain knowledge.
- Learning: choose $f_{\mathbf{w}} \in \mathcal{H}$ based on data.

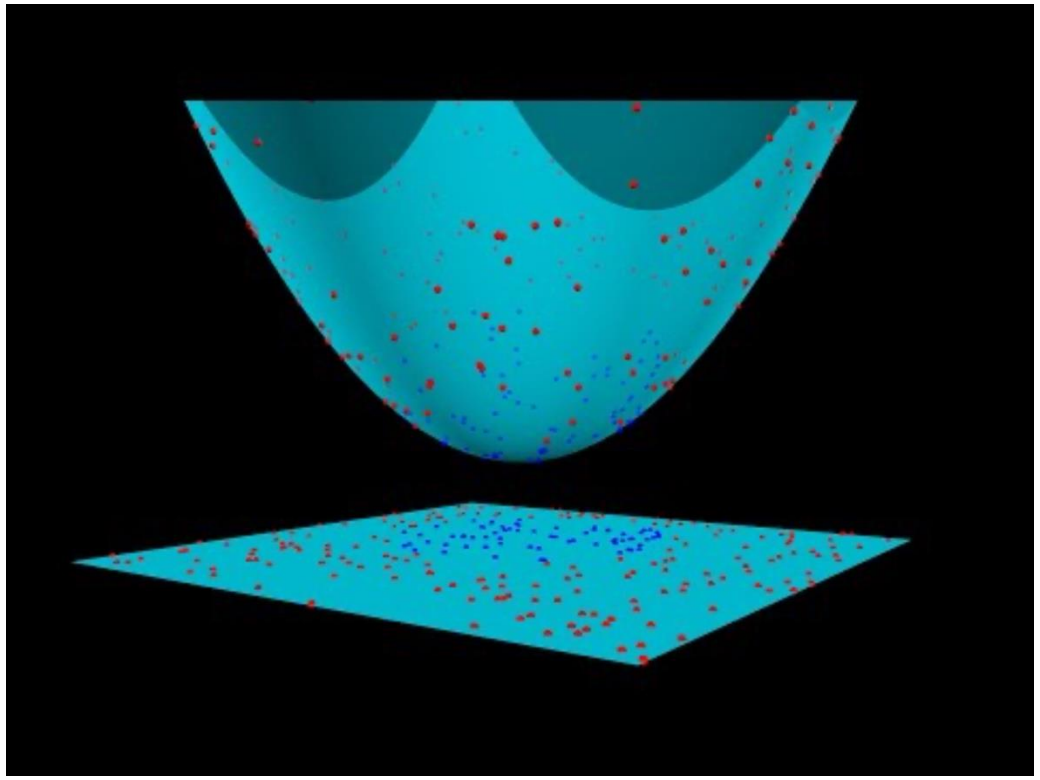
We want $\mathcal{H}$ to contain good predictors but not be too big

Feature Engineering + Learning

$$\mathcal{H} = \{f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w}^{\mathrm{T}}\phi(x)) : \mathbf{w} \in \mathbb{R}^d\}$$



- Example:



Demo: polynomial kernel in SVM

https://www.youtube.com/embed/3liCbRZPrZA

Decision Tree Algorithm

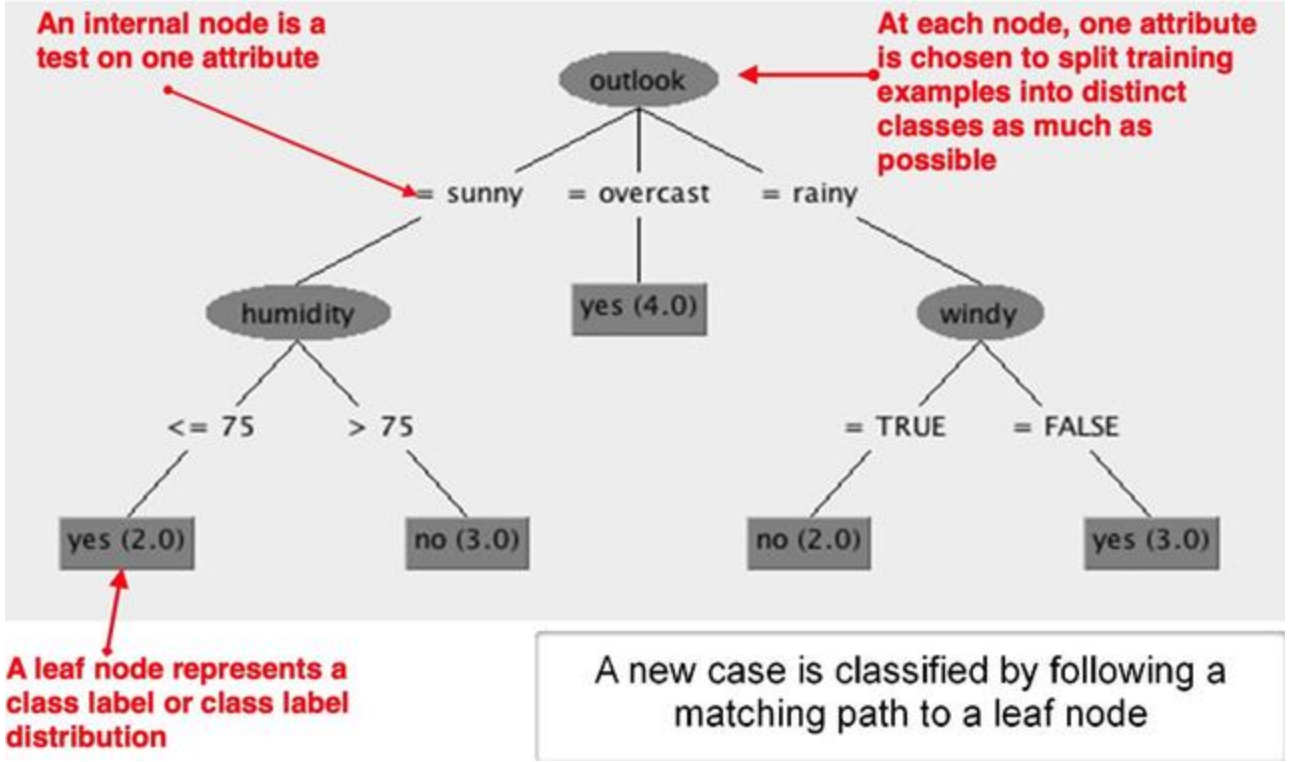
- Information Entropy
- Information Gain and ID3 Algorithm
- Information Gain Ratio

Play Tennis or Not?

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

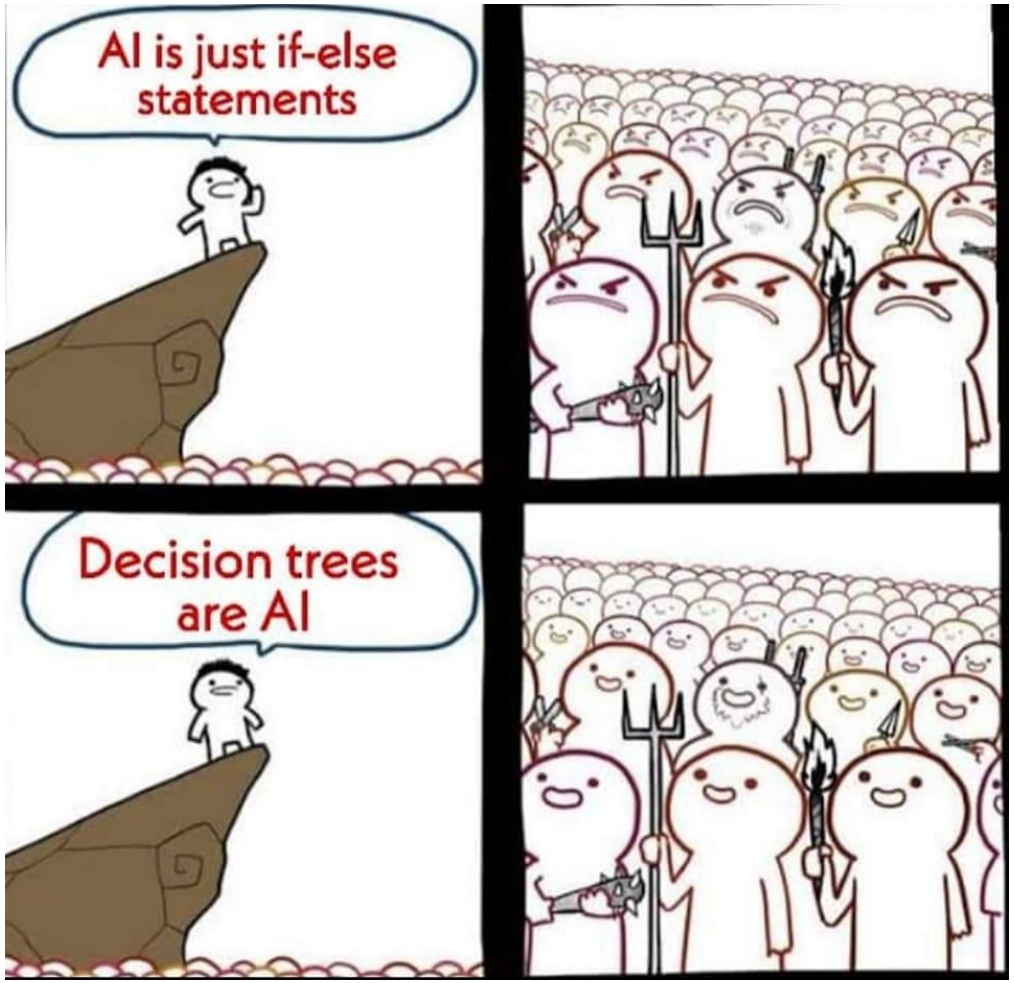
Given past data:

build a system to predict **Play/Not Play** ?



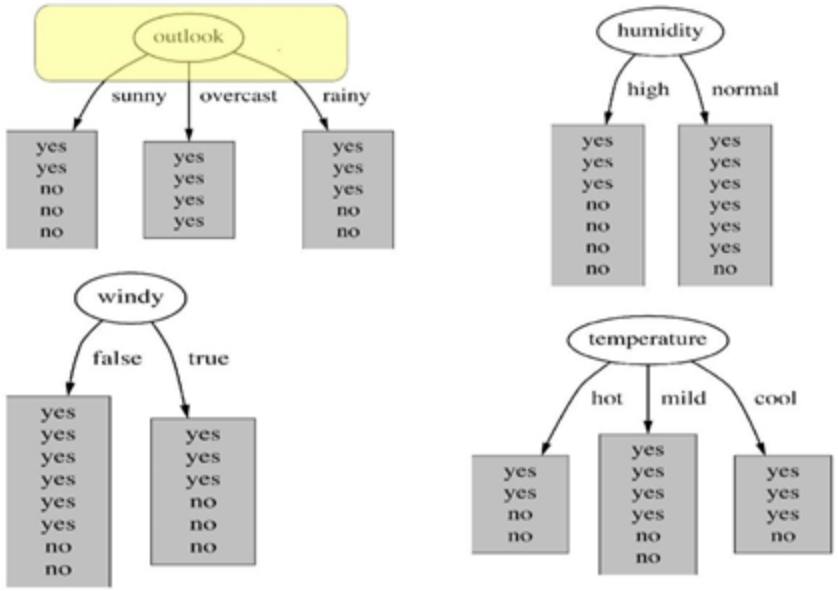
- Decision Tree is a method for approximating classification functions by means of a **tree-based representation**.
- A learned Decision Tree can be represented as a set of **if-then** rules

Idea: Building a Classification Tree



Idea: Building a Classification Tree

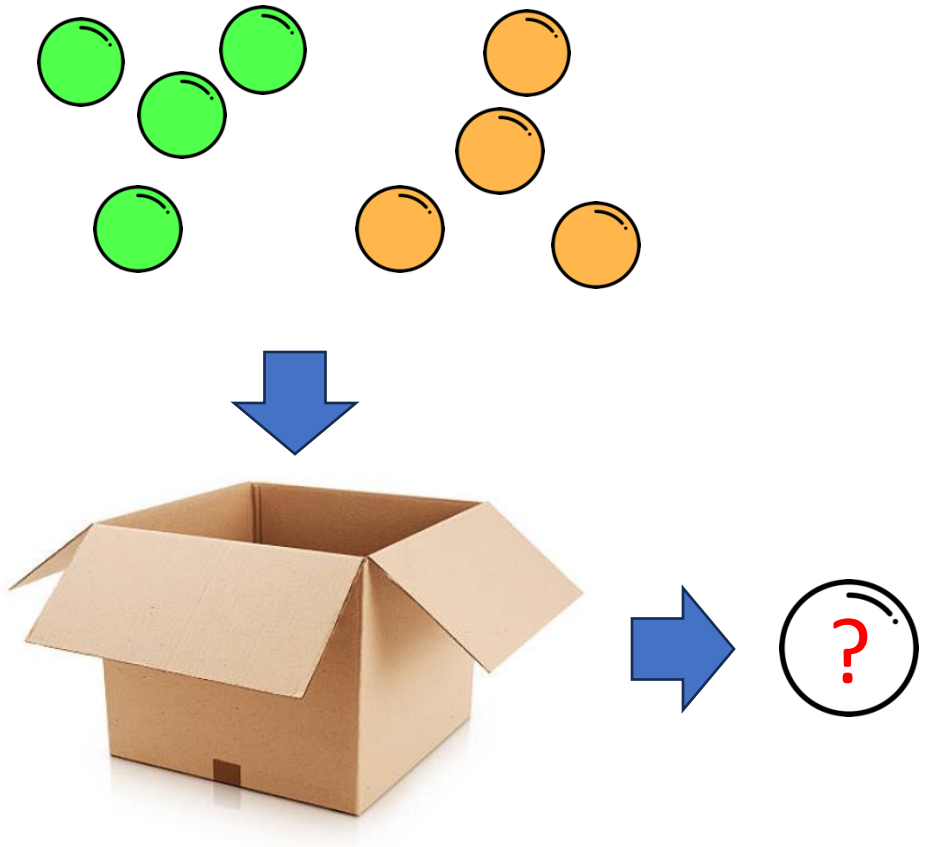
- **Top-down tree construction:**
  - At start, all training samples are at the root.
  - Split the samples recursively by choosing one **attribute** each time



Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

  - Which attribute should we choose at each node?
  - Divide-and-conquer: Split the into smaller subset.
  - Criteria: choose the attribute that **"best" separates** the classes on the training samples
  - How do we measure the information contained in each attribute? information measure

Information Entropy: Let's Play a Mini Game



We have N balls in the box with colors:
Green (50%) and Orange (50%)

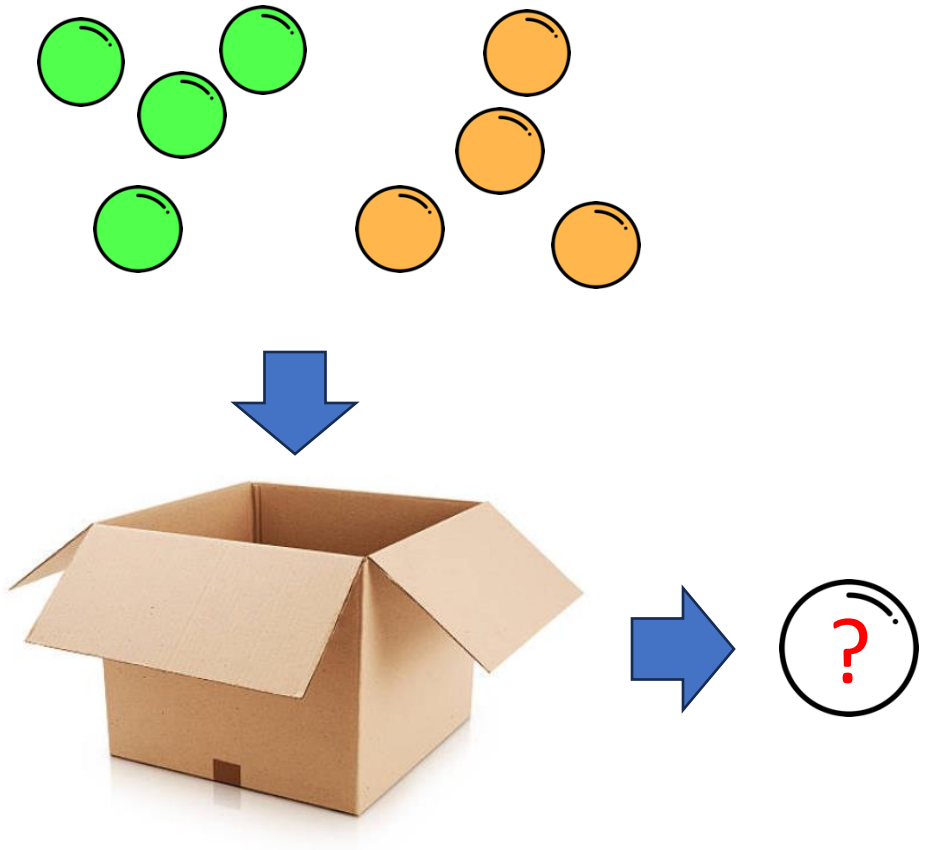
Take one out and guess its color.
Ask me questions and I can only answer
Yes or No.

Rewards:

+10 if the guess is correct, -50 if wrong

-2 for each question you ask.

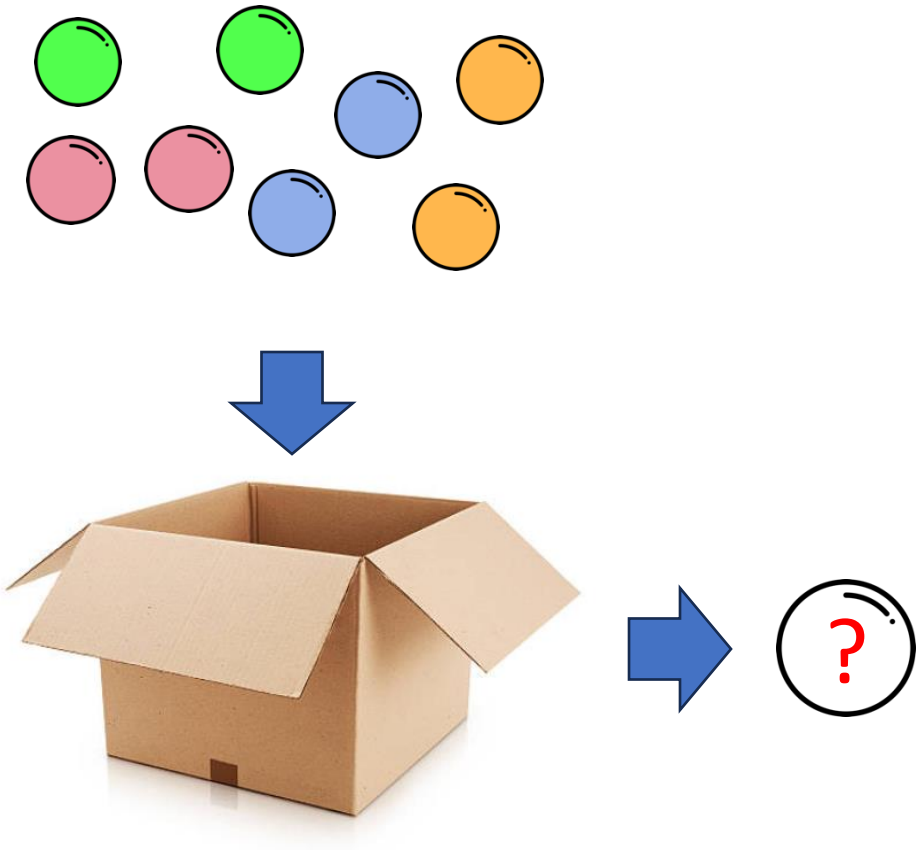
Information Entropy: Let's Play a Mini Game



We have N balls in the box with colors:
Green (50%) and Orange (50%)

What question would you ask?
How many questions do you need to ask?

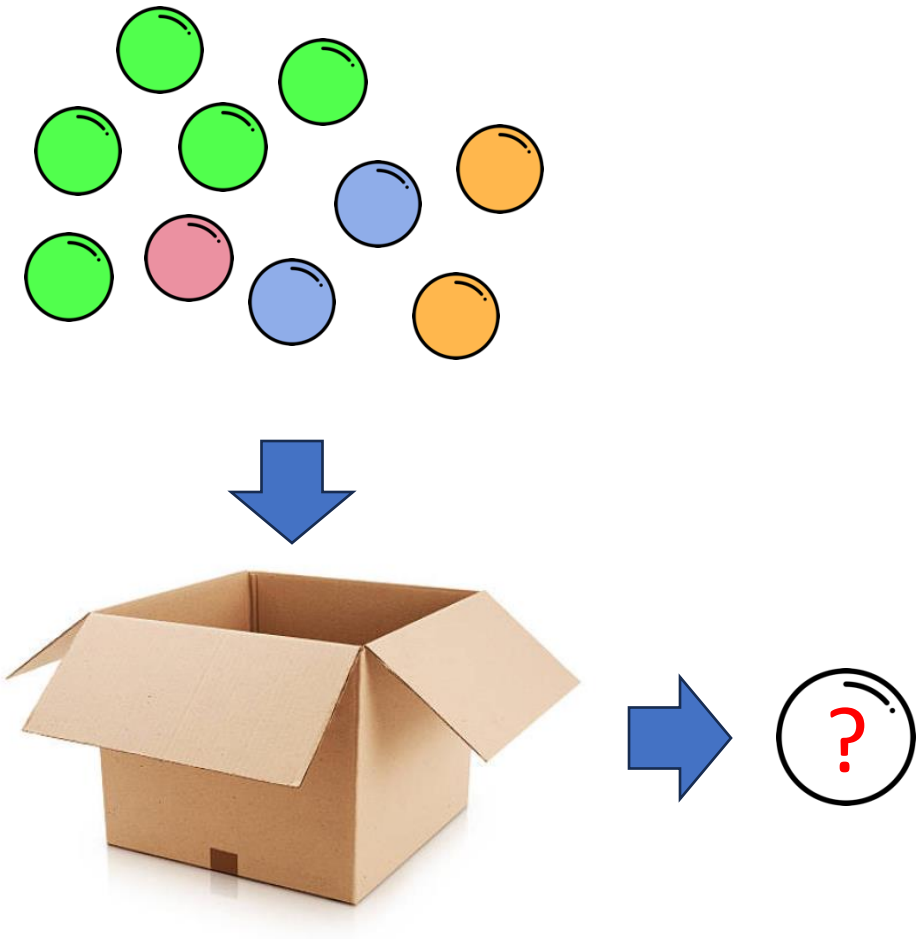
Information Entropy: Let's Play a Mini Game



We have N balls in the box with colors:
Green (25%), Orange (25%),
Pink (25%), and Blue (25%)

What question would you ask?
How many questions do you need to ask?

Information Entropy: Let's Play a Mini Game

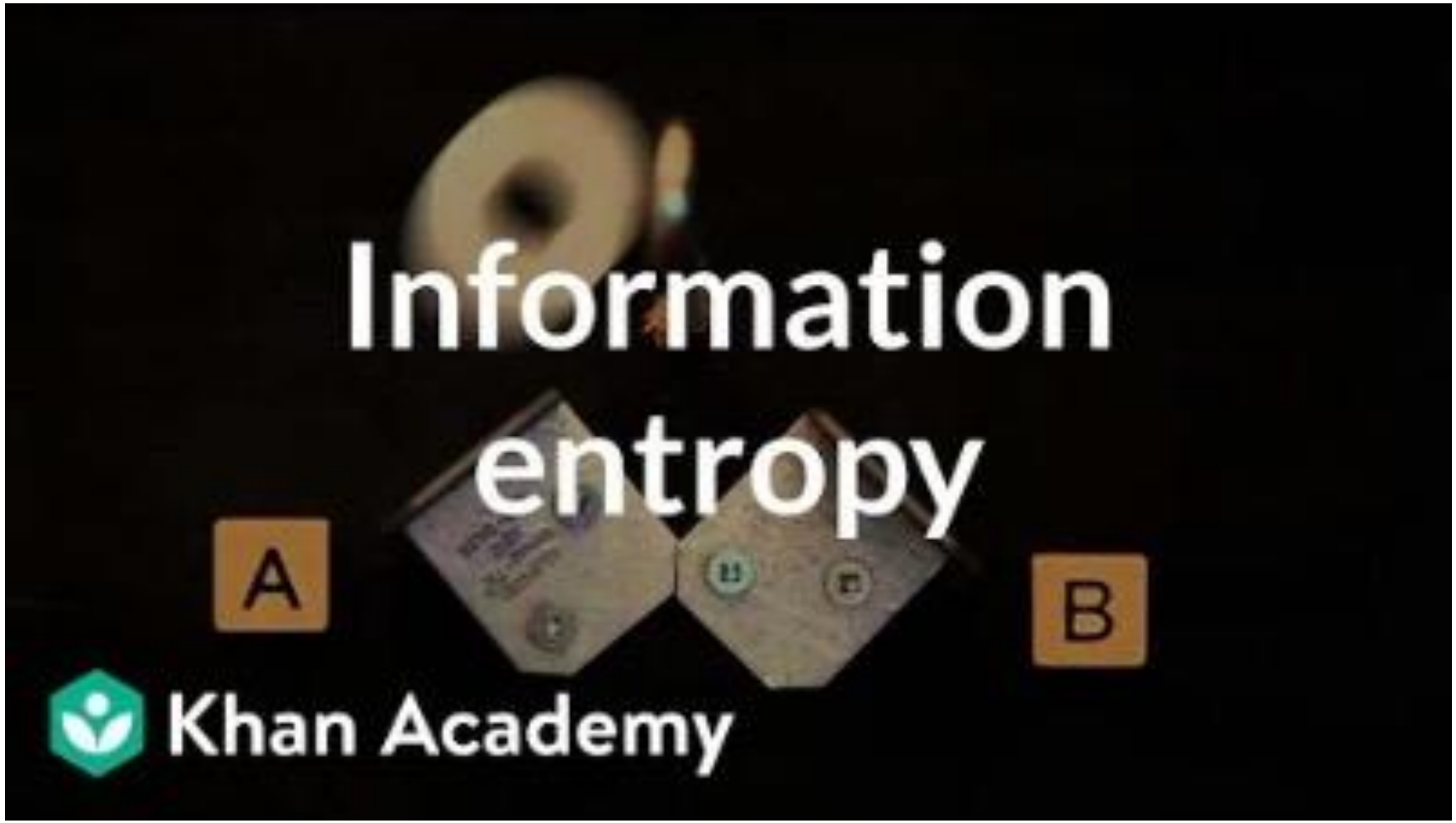


We have N balls in the box with colors:
Green (50%), Orange (20%),
Pink (10%), and Blue (20%)

What question would you ask?
How many questions do you need to ask?

Information measures: Entropy

- Entropy is a measure of the uncertainty of a random variable.



https://www.youtube.com/watch?v=2s3aJfRr9gE

Information measures: Entropy

- Entropy is a measure of the uncertainty of a random variable (or purity of a dataset).

- Given dataset D with K classes of samples,
  e.g., play/not play data: $K = 2$

- Entropy of dataset D:

$$\mathrm{Ent}(D) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

We define $0 \log_2 0 = 0$

p_k is the frequency of the k-th class

- The smaller $\mathrm{Ent}(D)$, the purer D.

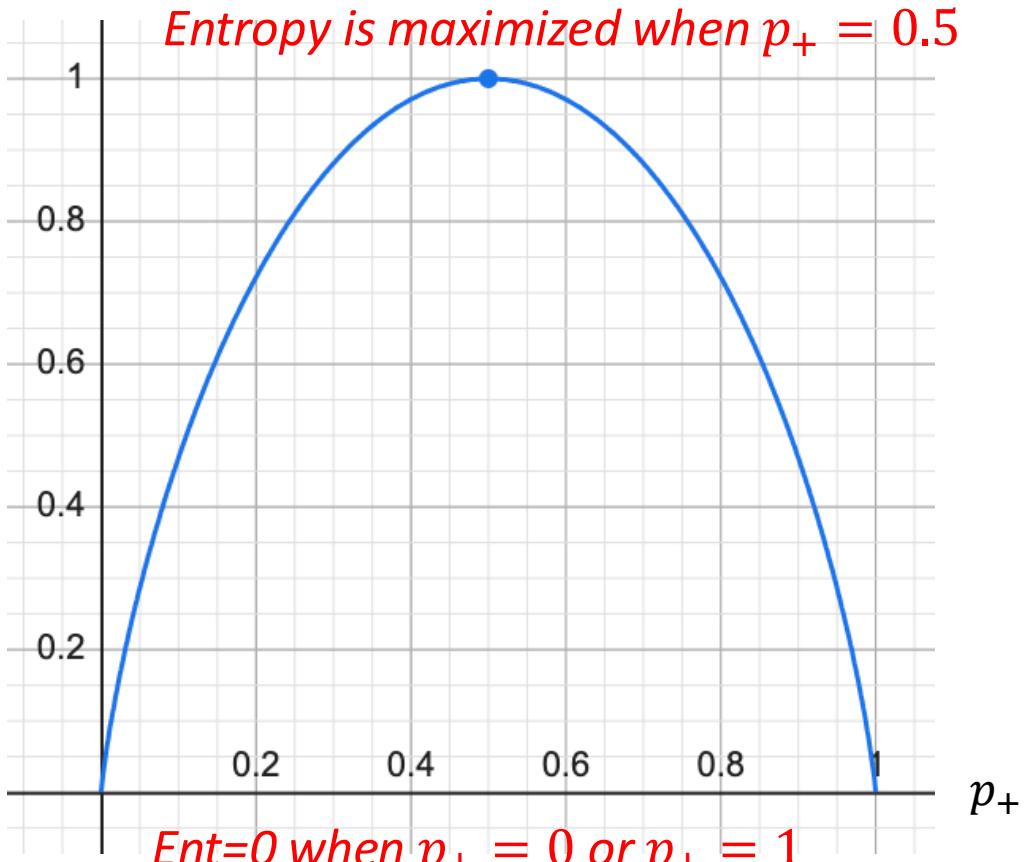
Information measures: Entropy

- Entropy is a measure of the uncertainty of a random variable (or purity of a dataset).

- Consider $K = 2$:

$$\mathrm{Ent}(D) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

p_+: frequency of the positive class
p_-: frequency of the negative class



Exercise: Compute $\mathrm{Ent}(D)$

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

$$\mathrm{Ent}(D) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

$$\log_2 9 \approx 3.17 \quad \log_2 5 \approx 2.32 \quad \log_2 14 \approx 3.81$$

- $K = 2$ (Yes/No)

- Samples with $k = 1$ (Yes): $p_k = 9/14$
- Samples with $k = 2$ (No): $p_k = 5/14$

$$\begin{aligned}
&\mathrm{Ent}(D) \\
&= -\left(\frac{9}{14} * \log_2 \frac{9}{14} + \frac{5}{14} * \log_2 \frac{5}{14}\right) \\
&= -\left(\frac{9}{14} * (\log_2 9 - \log_2 14) + \frac{5}{14} * (\log_2 5 - \log_2 14)\right) \\
&= 0.944
\end{aligned}$$

Exercise: Compute $\mathbf{Ent}(D)$ for the Following Datasets

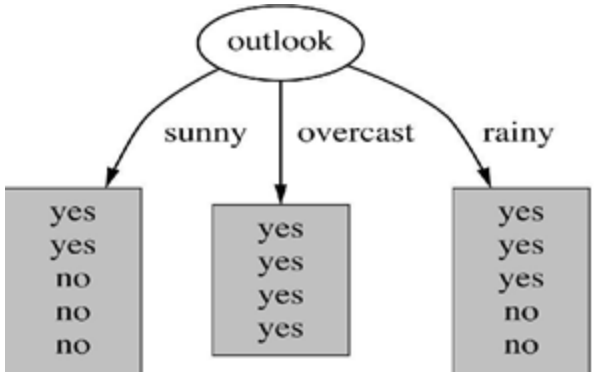
(only labels are shown)

- $D_1 = \{+1, +1, -1, -1, -1, +1, -1, +1\}$ Ans: 1
- $D_2 = \{A, B, B, A, C, D, B, A, C, D, D, C\}$ Ans: 2
- $D_3 = \{A, A, A, A, A, B, A, A, C, C, D, A\}$ Ans: 1.418

$\log_2 3 = 1.585$
$\log_2 12 = 3.585$

Information Gain

- For attribute a, it has V possible values.
  E.g., a="outlook", $V = 3$



- If we divide the data using a, the information gain is:

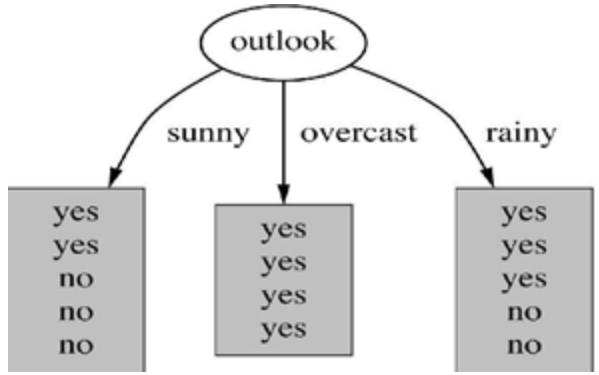
$$\text{Gain}(D, a) = \boxed{\text{Ent}(D)} - \boxed{\sum_{v=1}^{V} \frac{|D^v|}{|D|} \text{Ent}(D^v)}$$

purity before split *purity after split*

D^v: dataset that has value of v in D

Information Gain

- For attribute a, it has V possible values.
  E.g., a="outlook", $V = 3$



- If we divide the data using a, the information gain is:

$$\text{Gain}(D, a) = \text{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \text{Ent}(D^v)$$

D^v: dataset that has value of v in D

- What is the information gain $\text{Gain}(D, \text{“outlook”})$?

Sunny: $\frac{|D^v|}{D} \text{Ent}(D^v) = \frac{5}{14}\left(-\frac{2}{5} * \log_2 \frac{2}{5} - \frac{3}{5} * \log_2 \frac{3}{5}\right) = 0.347$

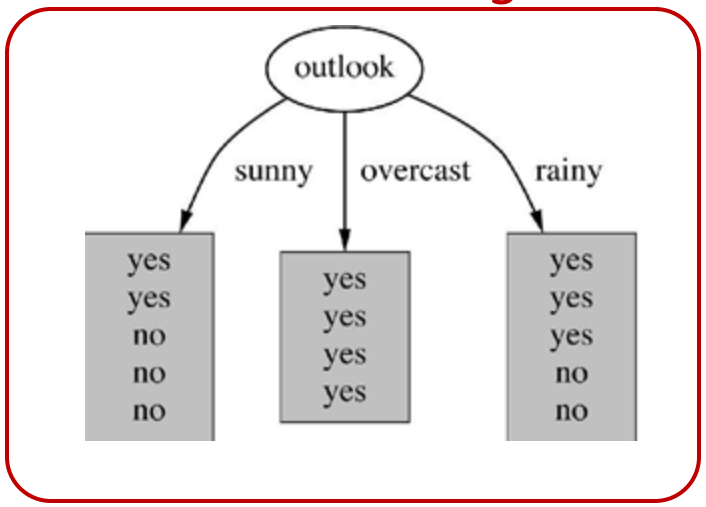
Overcast: $\frac{|D^v|}{D} \text{Ent}(D^v) = \frac{4}{14}\left(-\frac{4}{4} * \log_2 \frac{4}{4} - \frac{0}{4} * \log_2 \frac{0}{4}\right) = 0$ *We define* $0 \log_2 0 = 0$

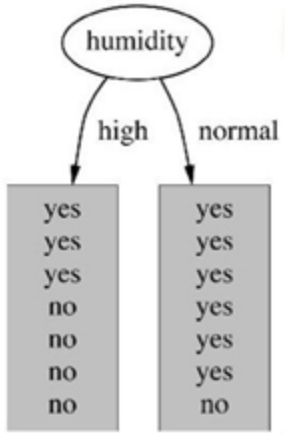
Rainy: $\frac{|D^v|}{D} \text{Ent}(D^v) = \frac{5}{14}\left(-\frac{3}{5} * \log_2 \frac{3}{5} - \frac{2}{5} * \log_2 \frac{2}{5}\right) = 0.347$

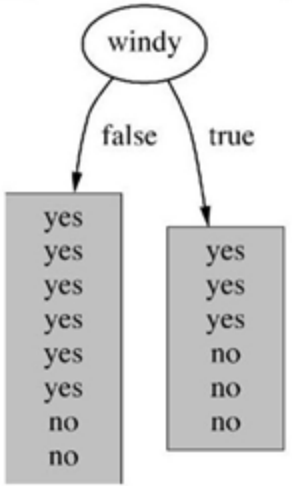
Information Gain: $\text{Gain}(D, \text{“outlook”}) = 0.94 - 0.347 - 0 - 0.347 = 0.246$

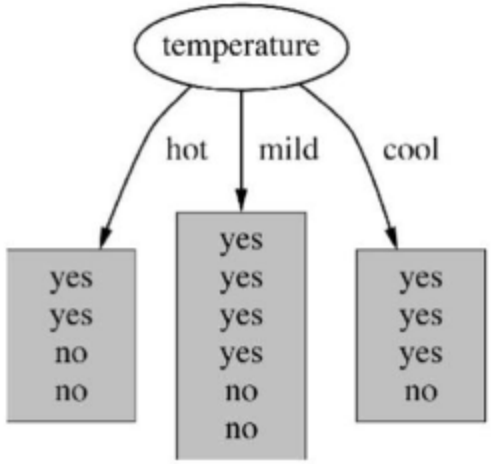
Compute the Information Gain for Other Attributes

highest









$\text{Gain}(D, \text{“outlook”}) = 0.246$

$\text{Gain}(D, \text{“windy”}) = 0.048$

$\text{Gain}(D, \text{“humidity”}) = 0.152$

$\text{Gain}(D, \text{“temperature”}) = 0.029$

We can split data using the attribute that has the highest information gain

Split by Information Gain: ID3 Algorithm

```
ID3(D,X) =
    Let T be a new tree
    If all instances in D have same class c
        Label(T) = c; Return T
    If X = ∅ or no attribute has positive information gain
        Label(T) = most common class in D; return T
    X ← attribute with highest information gain
    Label(T) = X
    For each value x of X
        D_x ← instances in D with X = x
        If D_x is empty
            Let T_x be a new tree
            Label(T_x) = most common class in D
        Else
            T_x = ID3(D_x, X − {X})
        Add a branch from T to T_x labeled by x
    Return T
```

(1) No further splitting is needed.

(2) No splitting brings improvement.

Select the feature to split

Select the samples according to the feature value

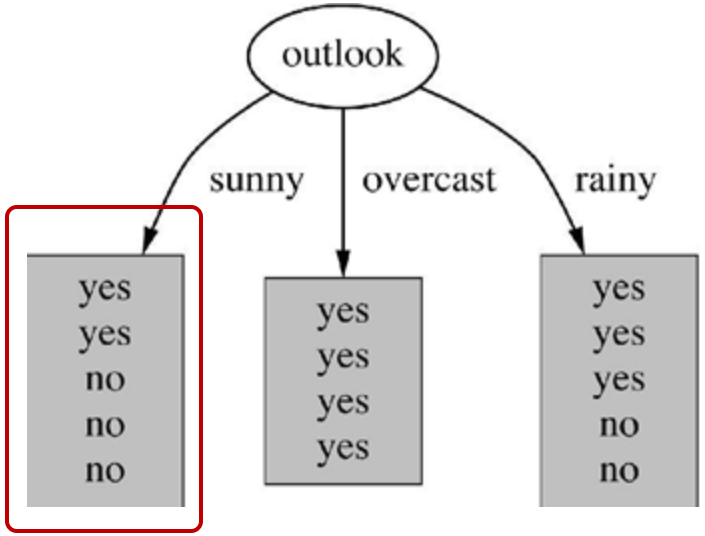
(3) No samples in the new tree

Recursively call ID3 with the new dataset D_x and feature set $\mathbf{X}\backslash\{X\}$ (exclude X from the feature set)

Split by Information Gain: ID3 Algorithm

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Step 2

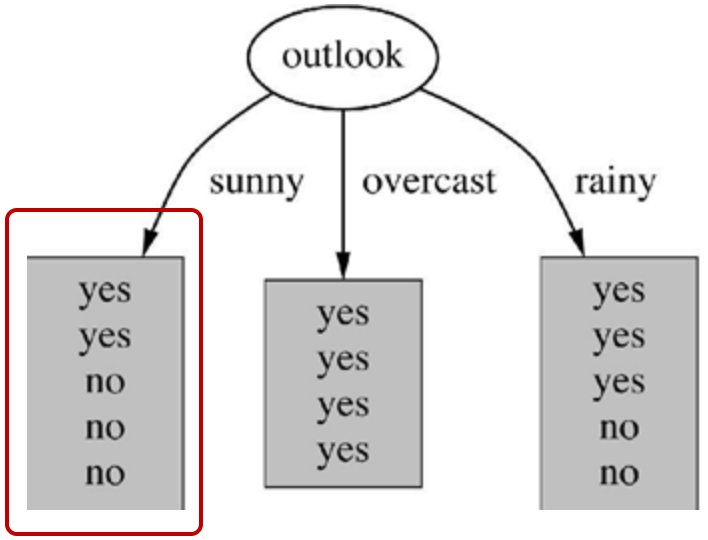


- When outlook=sunny

Split by Information Gain: ID3 Algorithm

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
sunny	mild	normal	true	Yes

Step 2



- As if we are given this dataset and continue selecting the attribute to split this "new" dataset.

$$\text{Ent}(D) = -(\frac{3}{5} * \log_2 \frac{3}{5} + \frac{2}{5} * \log_2 \frac{2}{5}) = 0.971$$

- Information Gain of the remaining attributes (temperature, humidity, windy)?

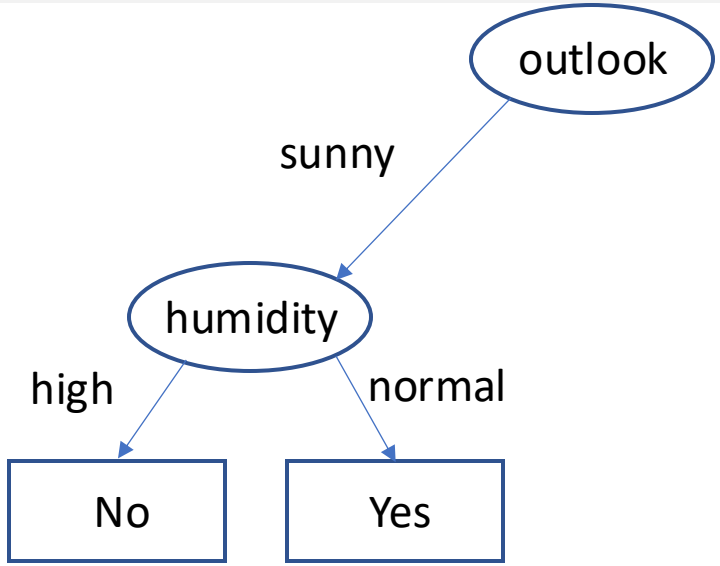
$$\text{Gain}(D, \text{"temperature"}) = 0.971 - \frac{2}{5}\text{Ent}(D^{\text{t=hot}}) - \frac{2}{5}\text{Ent}(D^{\text{t=mild}}) - \frac{1}{5}\text{Ent}(D^{\text{t=cool}}) = 0.971 - 0 - \frac{2}{5} - 0 = 0.571$$

$$\text{Gain}(D, \text{"humidity"}) = 0.971 - \frac{3}{5}\text{Ent}(D^{\text{h=high}}) - \frac{2}{5}\text{Ent}(D^{\text{h=normal}}) = 0.971 - 0 - 0 = 0.971$$

$$\text{Gain}(D, \text{"windy"}) = 0.971 - \frac{3}{5}\text{Ent}(D^{\text{w=false}}) - \frac{2}{5}\text{Ent}(D^{\text{w=true}}) = 0.971 - 0.5510 - 0.4 = 0.02$$

Split by Information Gain: ID3 Algorithm

ID3(**D**,**X**) =
- Let T be a new tree
- If all instances in **D** have same class c
  - Label(T) = c; Return T
- If **X** = ∅ or no attribute has positive information gain
  - Label(T) = most common class in **D**; return T
- $X \leftarrow$ attribute with highest information gain
- Label(T) = X
- For each value x of X
  - $\mathbf{D}_x \leftarrow$ instances in **D** with $X = x$
  - If $\mathbf{D}_x$ is empty
    - Let T_x be a new tree
    - Label(T_x) = most common class in **D**
  - Else
    - T_x = ID3($\mathbf{D}_x$, **X** − { X })
  - Add a branch from T to T_x labeled by x
- Return T

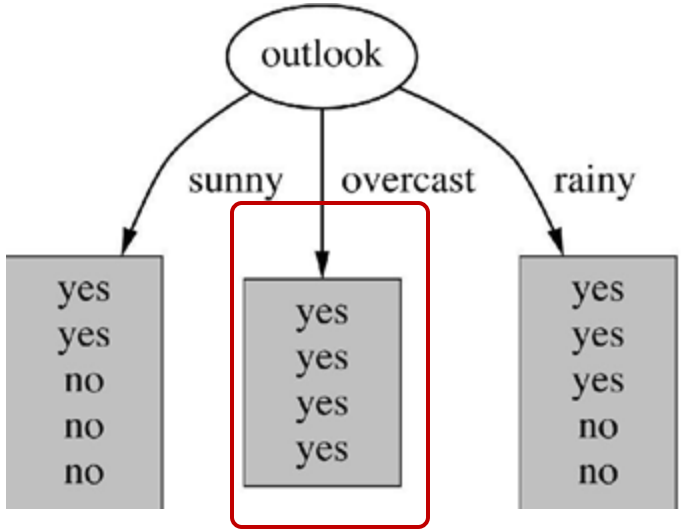


Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
sunny	mild	normal	true	Yes

Split by Information Gain: ID3 Algorithm

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

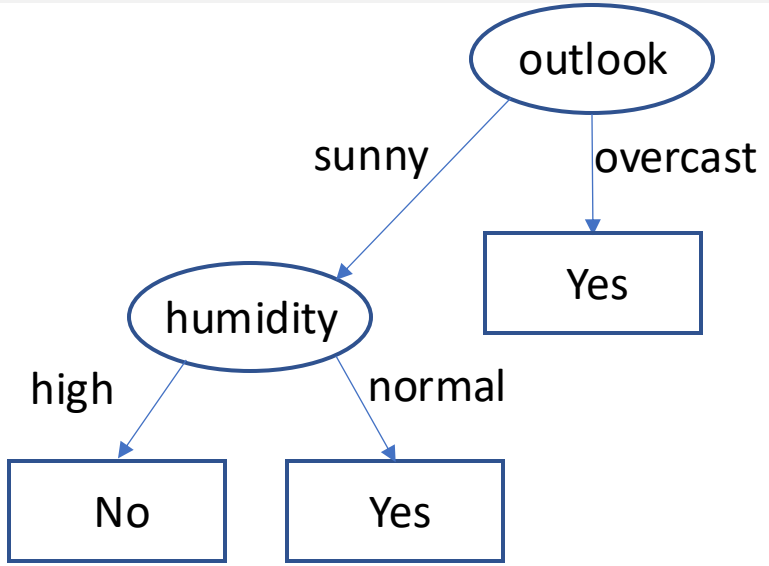
Step 3



- When outlook=overcast

Split by Information Gain: ID3 Algorithm

ID3(**D**,**X**) =
- Let T be a new tree
- If all instances in **D** have same class c
  - Label(T) = c; Return T
- If **X** = ∅ or no attribute has positive information gain
  - Label(T) = most common class in **D**; return T
- $X \leftarrow$ attribute with highest information gain
- Label(T) = X
- For each value x of X
  - $\mathbf{D}_x \leftarrow$ instances in **D** with $X = x$
  - If $\mathbf{D}_x$ is empty
    - Let T_x be a new tree
    - Label(T_x) = most common class in **D**
  - Else
    - T_x = ID3($\mathbf{D}_x$, **X** − { X })
  - Add a branch from T to T_x labeled by x
- Return T

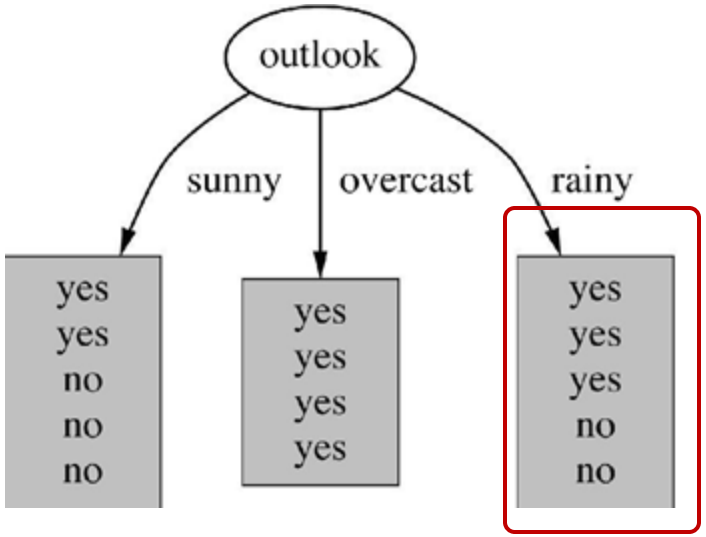


Outlook	Temperature	Humidity	Windy	Play?
overcast	hot	high	false	Yes
overcast	cool	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes

Split by Information Gain: ID3 Algorithm

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Step 4



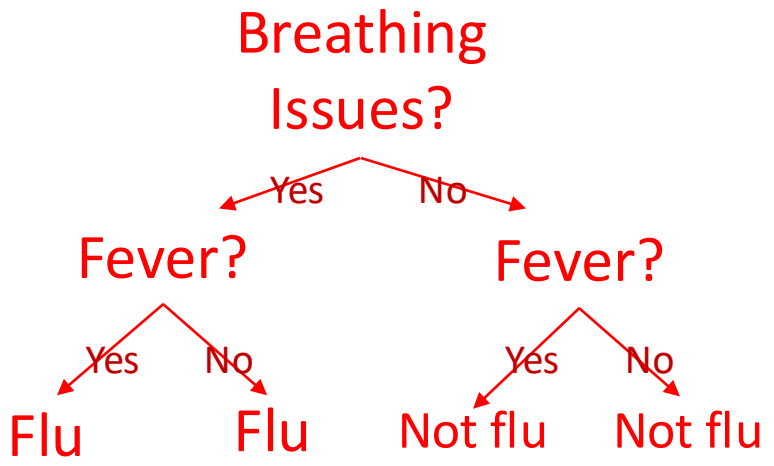
- When outlook=rainy
- Exercise: construct the remaining decision tree.

Exercise

Fever	Cough	Breathing Issues	Flu?
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	Yes	Yes
Yes	Yes	Yes	Yes
No	Yes	No	No
Yes	No	Yes	Yes
Yes	No	Yes	Yes
No	Yes	Yes	Yes
Yes	Yes	No	Yes
No	Yes	No	No
No	Yes	Yes	Yes
No	Yes	Yes	No
Yes	Yes	No	No

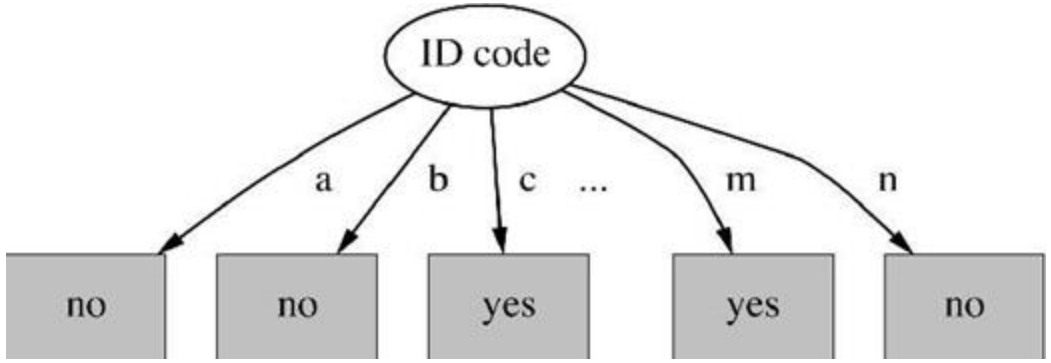
Construct a decision tree for this flu diagnosis dataset

Sample solution:



Suppose We Have An ID Column

$$\text{Gain}(D, \text{“ID”}) = 0.94 - 14 * \frac{1}{14}(-1 * \log_2 1) = 0.94 - 0 = 0.94$$



Useless: a new instance with ID=“p”?

Information gain favours attributes with a larger number of possible values (V)

ID	Outlook	Temperature	Humidity	Windy	Play?
a	sunny	hot	high	false	No
b	sunny	hot	high	true	No
c	overcast	hot	high	false	Yes
d	rain	mild	high	false	Yes
e	rain	cool	normal	false	Yes
f	rain	cool	normal	true	No
g	overcast	cool	normal	true	Yes
h	sunny	mild	high	false	No
i	sunny	cool	normal	false	Yes
j	rain	mild	normal	false	Yes
k	sunny	mild	normal	true	Yes
l	overcast	mild	high	true	Yes
m	overcast	hot	normal	false	Yes
n	rain	mild	high	true	No

An Alternative: Information Gain Ratio

- For attribute a, it has V possible values.
  E.g., a="outlook", $V = 3$

- If we divide the data using a, the information gain ratio is:

$$\text{Gain\_ratio}(D, a) = \frac{\text{Gain}(D, a)}{\text{IV}(a)}$$

$$\text{Gain}(D, a) = \text{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \text{Ent}(D^v)$$

D^v: dataset that has value of v in D

$$\text{IV}(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$$

$\text{IV}(a)$: intrinsic value of attribute a

An Alternative: Information Gain Ratio

$$\text{Gain\_ratio}(D, a) = \frac{\text{Gain}(D, a)}{\text{IV}(a)}$$

$$\text{Gain}(D, a) = \text{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \text{Ent}(D^v)$$

$$\text{IV}(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$$

Example: Compute $\text{Gain\_ratio}(D, \text{“ID”})$

$$\text{Gain}(D, \text{“ID”}) = 0.94 \qquad \text{IV}(\text{“ID”}) = 3.8073$$

$$\text{Gain\_ratio}(D, \text{“ID”}) = 0.247$$

Information gain ratio favours attributes with a smaller number of possible values (V)

ID	Outlook	Temperature	Humidity	Windy	Play?
a	sunny	hot	high	false	No
b	sunny	hot	high	true	No
c	overcast	hot	high	false	Yes
d	rain	mild	high	false	Yes
e	rain	cool	normal	false	Yes
f	rain	cool	normal	true	No
g	overcast	cool	normal	true	Yes
h	sunny	mild	high	false	No
i	sunny	cool	normal	false	Yes
j	rain	mild	normal	false	Yes
k	sunny	mild	normal	true	Yes
l	overcast	mild	high	true	Yes
m	overcast	hot	normal	false	Yes
n	rain	mild	high	true	No

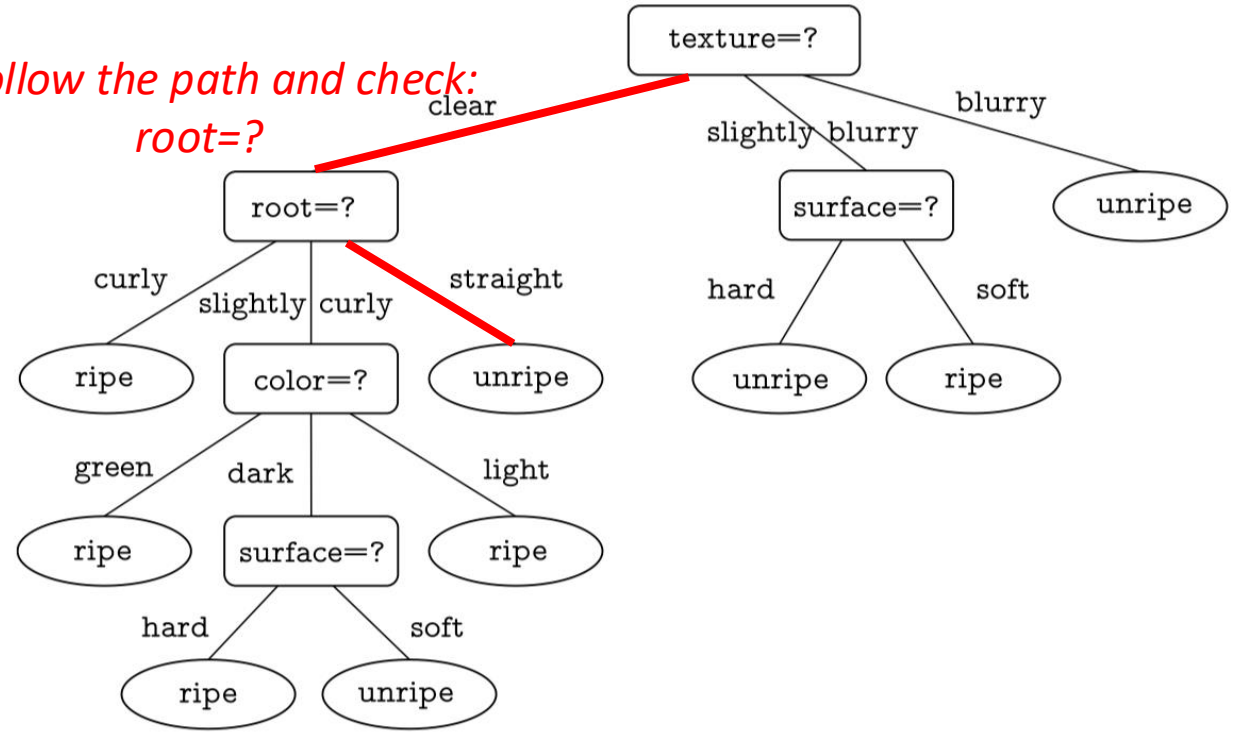
Making Predictions using Decision Trees

Given the decision tree, make predictions for the following

surface	root	texture	color	ripe?
hard	straight	clear	dark	unripe
soft	curly	blurry	light	
hard	slightly curly	clear	dark	
hard	straight	clear	dark	

check from top to bottom:
1. what is the value of feature "texture"?

2. follow the path and check: root=?



A decision tree for picking watermelons

Continuous Attributes in Decision Tree

Continuous Attributes in Decision Tree Algorithm

Outlook	Temperature	Humidity	Windy	Play?
sunny	34	high	false	No
sunny	33	high	true	No
overcast	31	high	false	Yes
rain	28	high	false	Yes
rain	24	normal	false	Yes
rain	23	normal	true	No
overcast	25	normal	true	Yes
sunny	27	high	false	No
sunny	25	normal	false	Yes
rain	27	normal	false	Yes
sunny	28	normal	true	Yes
overcast	28	high	true	Yes
overcast	29	normal	false	Yes
rain	27	high	true	No

Continuous attributes (features) is common in real datasets.

Number of possible values of a continuous attribute is infinite.

A simplest approach: discretization

E.g., divide into ranges:

"hot": $T \geq 29$

"mild": $29 > T \geq 25$

"cool": $T < 25$

Can algorithm automatically do this?

Bi-Partition for Continuous Attributes in Decision Tree

Idea: divide the dataset D into two parts by threshold t for the continuous attribute "a":

D_t^+: data samples which has the value of "a" greater than t;

D_t^-: data samples which has the value of "a" less than t.

The temperatures:

Temperature	34	33	31	28	24	23	25	27	25	27	28	28	29	27
Play?	No	No	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	No

1. Sort the unique values ascendingly: $\{a^1, a^2, \dots, a^n\}$

E.g., {23, 24, 25, 27, 28, 29, 31, 33, 34}

What are the possible thresholds? Thresholds between two values have the same effect

2. Consider the midpoint of the intervals:

$$T_a = \left\{ \frac{a^i + a^{i+1}}{2} \mid 1 \le i \le n-1 \right\}$$

E.g., $T_{\text{Temperature}} = \{23.5, 24.5, 26, 27.5, 28.5, 30, 32, 33.5\}$

Bi-Partition for Continuous Attributes in Decision Tree

The temperatures:

Temperature	34	33	31	28	24	23	25	27	25	27	28	28	29	27
Play?	No	No	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	No

for continuous attributes

3. Compute the information gain by:

$$\text{Gain}(D,a) = \max_{t \in \text{T}_a} \text{Gain}(D,a,t) = \max_{t \in \text{T}_a} \left[\text{Ent}(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_t^\lambda|}{|D|} \text{Ent}(D_t^\lambda) \right]$$

$T_{\text{Temperature}} = \{23.5, 24.5, 26, 27.5, 28.5, 30, 32, 33.5\}$ $\quad \text{Ent}(D) = 0.94$

1) t = 23.5: D_t^+ = {No, No, Yes, Yes, Yes, Yes, No, Yes, Yes, Yes, Yes, Yes, No} $\quad \text{Ent}(D_t^+) = 0.891$

D_t^- = {No} $\quad \text{Ent}(D_t^-) = 0$

$$\text{Gain}(D, \text{“Temperature”}, 23.5) = 0.94 - \frac{13}{14} * 0.89 - 0 \approx 0.113$$

Bi-Partition for Continuous Attributes in Decision Tree

The temperatures:

Temperature	34	33	31	28	24	23	25	27	25	27	28	28	29	27
Play?	No	No	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	No

for continuous attributes

3. Compute the information gain by:

$$\text{Gain}(D, a) = \max_{t \in \text{T}_a} \text{Gain}(D, a, t) = \max_{t \in \text{T}_a} \left[\text{Ent}(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_t^\lambda|}{|D|} \text{Ent}(D_t^\lambda) \right]$$

$T_{\text{Temperature}} = \{23.5, 24.5, 26, 27.5, 28.5, 30, 32, 33.5\}$ $\quad \text{Ent}(D) = 0.94$

2) t = 24.5: D_t^+ = {No, No, Yes, Yes, Yes, No, Yes, Yes, Yes, Yes, Yes, No} $\quad \text{Ent}(D_t^+) = 0.918$

D_t^- = {Yes, No} $\quad \text{Ent}(D_t^-) = 1$

$$\text{Gain}(D, \text{“Temperature”}, 23.5) = 0.94 - \frac{12}{14} * 0.92 - \frac{2}{14} * 1 \approx 0.01$$

Bi-Partition for Continuous Attributes in Decision Tree

The temperatures:

Temperature	34	33	31	28	24	23	25	27	25	27	28	28	29	27
Play?	No	No	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	No

for continuous attributes

3. Compute the information gain by:

$$\text{Gain}(D, a) = \max_{t \in \text{T}_a} \text{Gain}(D, a, t) = \max_{t \in \text{T}_a} \left[\text{Ent}(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_t^\lambda|}{|D|} \text{Ent}(D_t^\lambda) \right]$$

$$T_{\text{Temperature}} = \{23.5, 24.5, 26, 27.5, 28.5, 30, 32, 33.5\} \qquad \text{Ent}(D) = 0.94$$

Compute for all possible thresholds:

$$Gain(D, \text{“Temperature”}) = 0.245$$

$$\text{Gain}(D, \text{“Temperature”}, 23.5) \approx 0.113$$
$$\text{Gain}(D, \text{“Temperature”}, 24.5) \approx 0.01$$
$$\text{Gain}(D, \text{“Temperature”}, 26) \approx 0.015$$
$$\text{Gain}(D, \text{“Temperature”}, 27.5) \approx 0.016$$
$$\text{Gain}(D, \text{“Temperature”}, 28.5) \approx 0.025$$
$$\text{Gain}(D, \text{“Temperature”}, 30) \approx 0.079$$
$$\text{Gain}(D, \text{“Temperature”}, 32) \approx 0.245$$
$$\text{Gain}(D, \text{“Temperature”}, 33.5) \approx 0.113$$

threshold: $t = 32$

Pruning in Decision Trees

Overfitting in Decision Trees

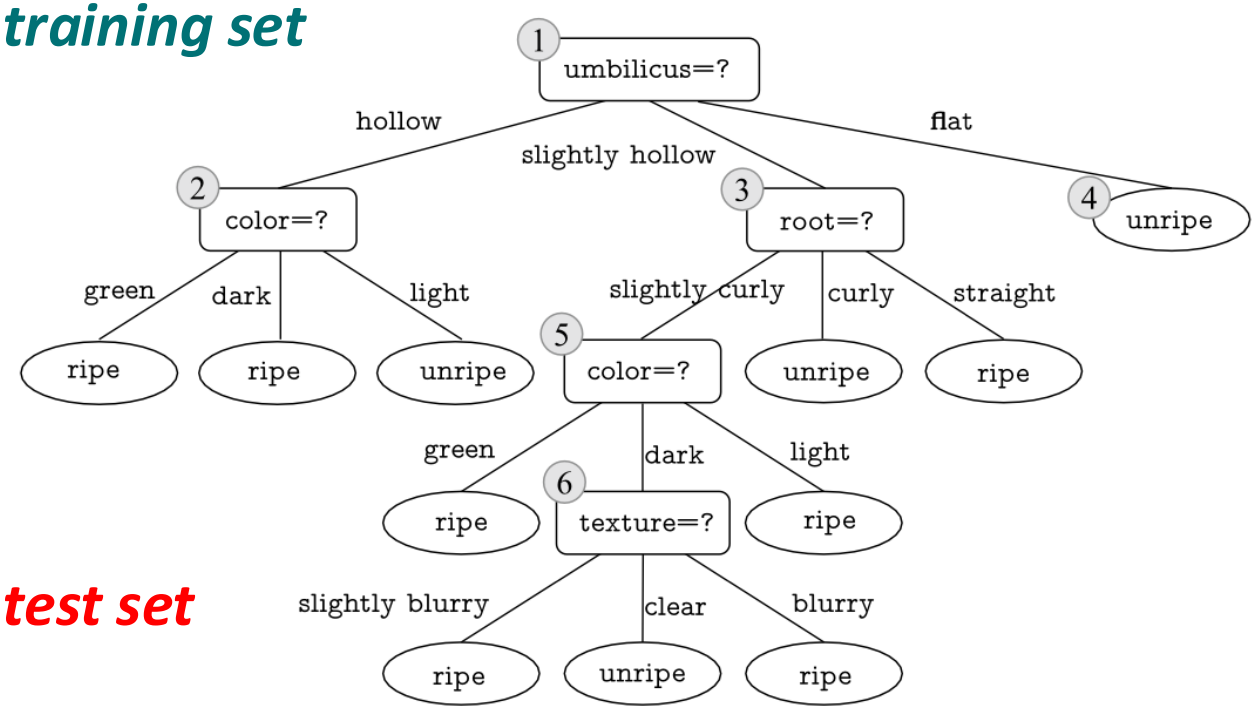
A watermelon dataset

ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
10	green	straight	crisp	clear	flat	soft	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

training set

ID	color	root	sound	texture	umbilicus	surface	ripe
4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false

test set



decision tree trained using information gain

Exercise: Compute the Accuracy in training and testing subsets. Is the tree overfitting?

Pruning: Preventing Overfitting in Decision Trees

- Pruning is the primary strategy of decision tree learning algorithms to deal with overfitting.
- General pruning strategies: *pre-pruning* & *post-pruning*.

- Pre-pruning: evaluates the improvement of the generalization ability of each split and cancels a split if the improvement is small.

- Post-pruning: re-examines the non-leaf nodes of a *fully grown* decision tree, and a node is replaced with a leaf node if the replacement leads to improved generalization ability.

- How do we know if the generalization ability improves?
- Cross validation! (e.g., hold-out method)

Pre-Pruning

A watermelon dataset

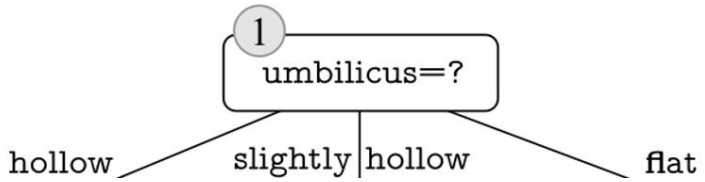
ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
10	green	straight	crisp	clear	flat	soft	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

↑ ***training set*** ↑ ↓ ***test set*** ↓

4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false

Step 1 (as before): compute information gain & choose the attribute to split.

Attribute with max. information gain: *umbilicus*



Step 2 (new): check improvement of generalization ability.

Accuracy of not splitting?

All samples classified as majority class (ripe)

Remark: When there is more than one class with the largest number of samples, we randomly select one of the classes. In this example, you can also classify all samples as unripe.

Pre-Pruning

A watermelon dataset

ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
10	green	straight	crisp	clear	flat	soft	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

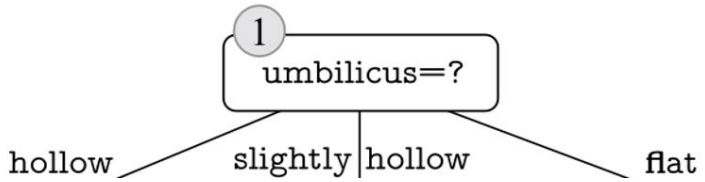
↑ ***training set*** ↑

↓ ***test set*** ↓

4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false

Step 1 (as before): compute information gain & choose the attribute to split.

Attribute with max. information gain: *umbilicus*



Step 2 (new): check improvement of generalization ability.

Accuracy of not splitting?

All samples classified as majority class (ripe)

Accuracy in test set = 3/7 ≈ 42.9%

Pre-Pruning

A watermelon dataset

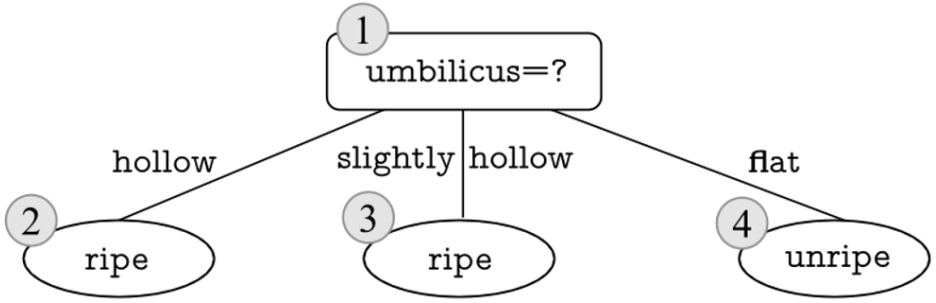
ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
10	green	straight	crisp	clear	flat	soft	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

↑ ***training set*** ↑ ↓ ***test set*** ↓

4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false

Step 1 (as before): compute information gain & choose the attribute to split.

Attribute with max. information gain: *umbilicus*



Step 2 (new): check improvement of generalization ability.

Accuracy of not splitting? Acc in test set ≈ 42.9%

Accuracy of splitting?

Samples placed into three child nodes.
Accuracy in test set = 5/7 ≈ 71.4%

Pre-Pruning

A watermelon dataset

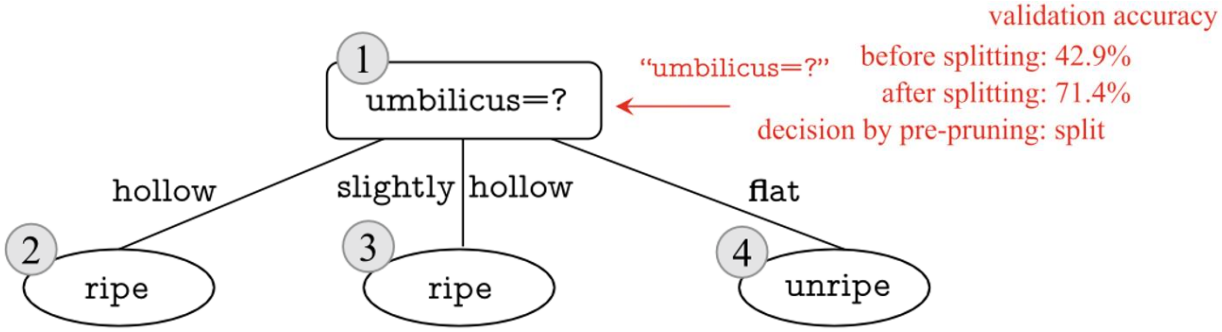
ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
10	green	straight	crisp	clear	flat	soft	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

↑ ***training set*** ↑ ↓ ***test set*** ↓

4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false

Step 1 (as before): compute information gain & choose the attribute to split.

Attribute with max. information gain: *umbilicus*



Step 2 (new): check improvement of generalization ability.

Accuracy of not splitting? Acc in test set ≈ 42.9%

Accuracy of splitting? Acc in test set ≈ 71.4%

Generalization ability improves → Split

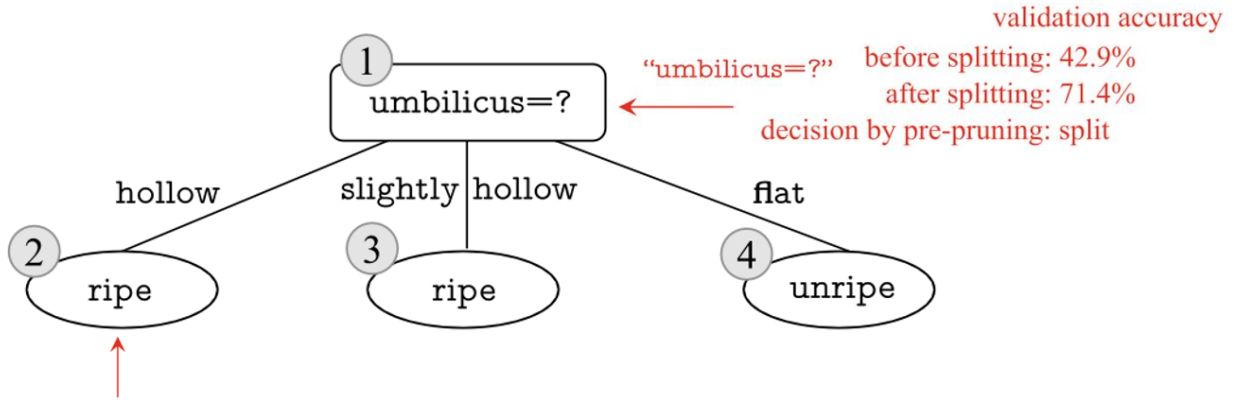
Pre-Pruning

A watermelon dataset

ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
~~6~~	~~green~~	~~slightly curly~~	~~muffled~~	~~clear~~	~~slightly hollow~~	~~soft~~	~~true~~
~~7~~	~~dark~~	~~slightly curly~~	~~muffled~~	~~slightly blurry~~	~~slightly hollow~~	~~soft~~	~~true~~
~~10~~	~~green~~	~~straight~~	~~crisp~~	~~clear~~	~~flat~~	~~soft~~	~~false~~
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
~~15~~	~~dark~~	~~slightly curly~~	~~muffled~~	~~clear~~	~~slightly hollow~~	~~soft~~	~~false~~
~~16~~	~~light~~	~~curly~~	~~muffled~~	~~blurry~~	~~flat~~	~~hard~~	~~false~~
~~17~~	~~green~~	~~curly~~	~~dull~~	~~slightly blurry~~	~~slightly hollow~~	~~hard~~	~~false~~

↑ ***training set*** ↑ ↓ ***test set*** ↓

4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false



Continue with the subset umbilicus=hollow

Step 1: choose the attribute.

Attribute with max. information gain: *color*

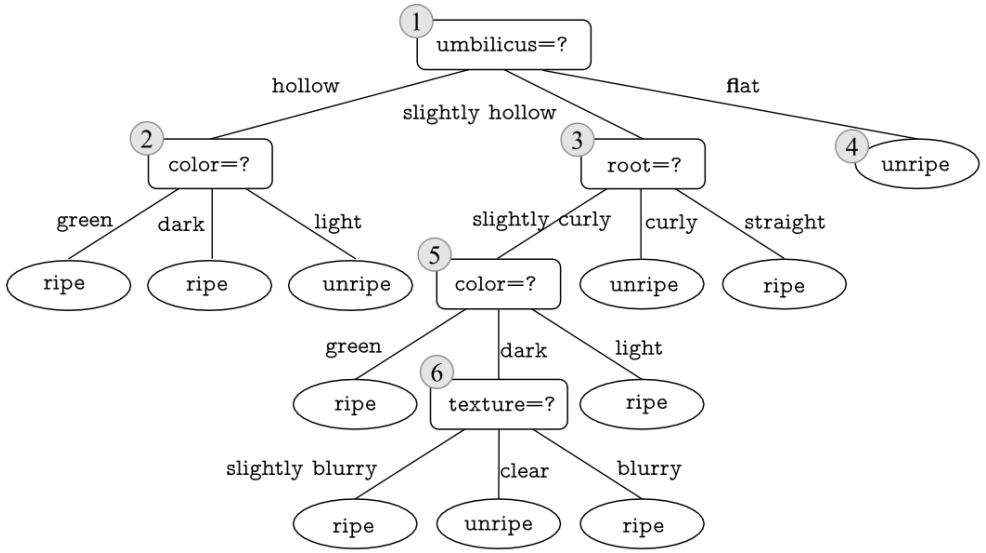
Step 2: check generalization

Accuracy of not splitting? Acc in test set ≈ 71.4%

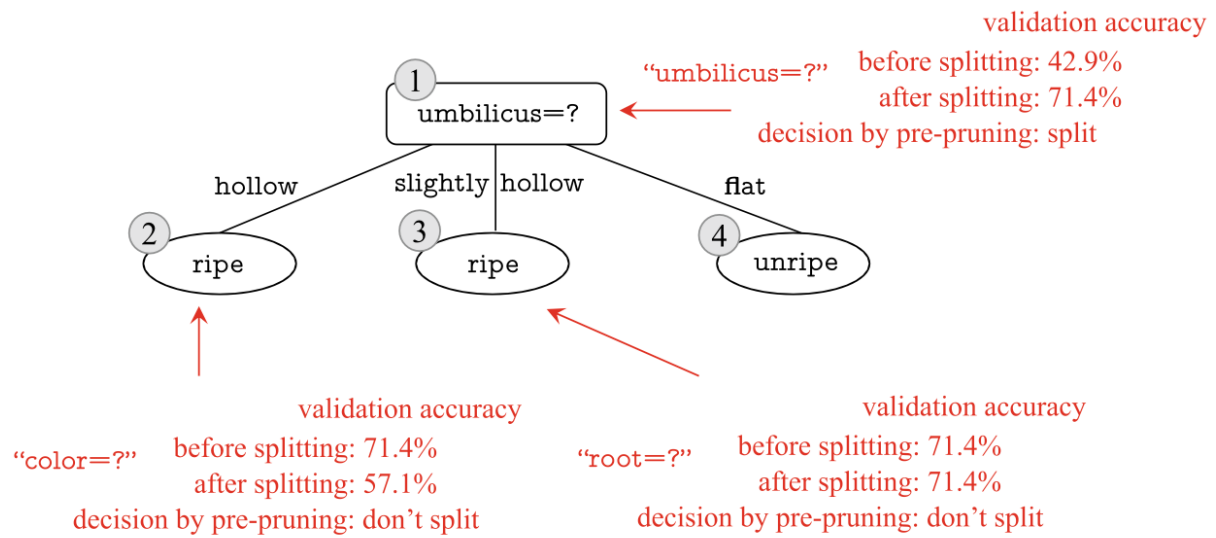
Accuracy of splitting? Acc in test set ≈ 57.1%

Generalization is worse → Don't Split

Pre-Pruning



Decision tree without pre-pruning



Decision tree with pre-pruning

- Pre-pruning: reduces the branches → reduce the risk of overfitting & computational cost.
- Some branches are prevented, but it is still possible that their subsequent splits lead to significant improvement. Branches are pruned due to the greedy nature of pre-pruning, and it may introduce the risk of underfitting.

Post-Pruning

A watermelon dataset

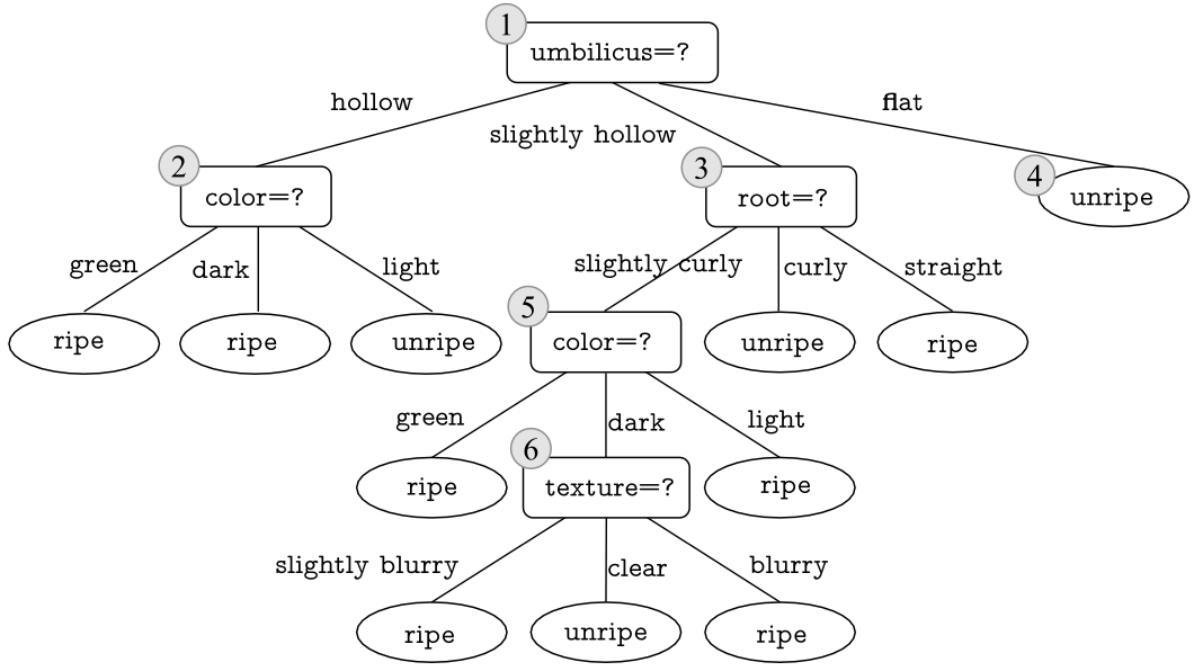
ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
10	green	straight	crisp	clear	flat	soft	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

↑ ***training set*** ↑

↓ ***test set*** ↓

4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false

Post-pruning allows the decision tree to grow completely:



Do pruning after constructing the tree:

- Check non-leaf nodes from bottom.
- If replacing it with a leaf node, does generalization improve?

Post-Pruning

A watermelon dataset

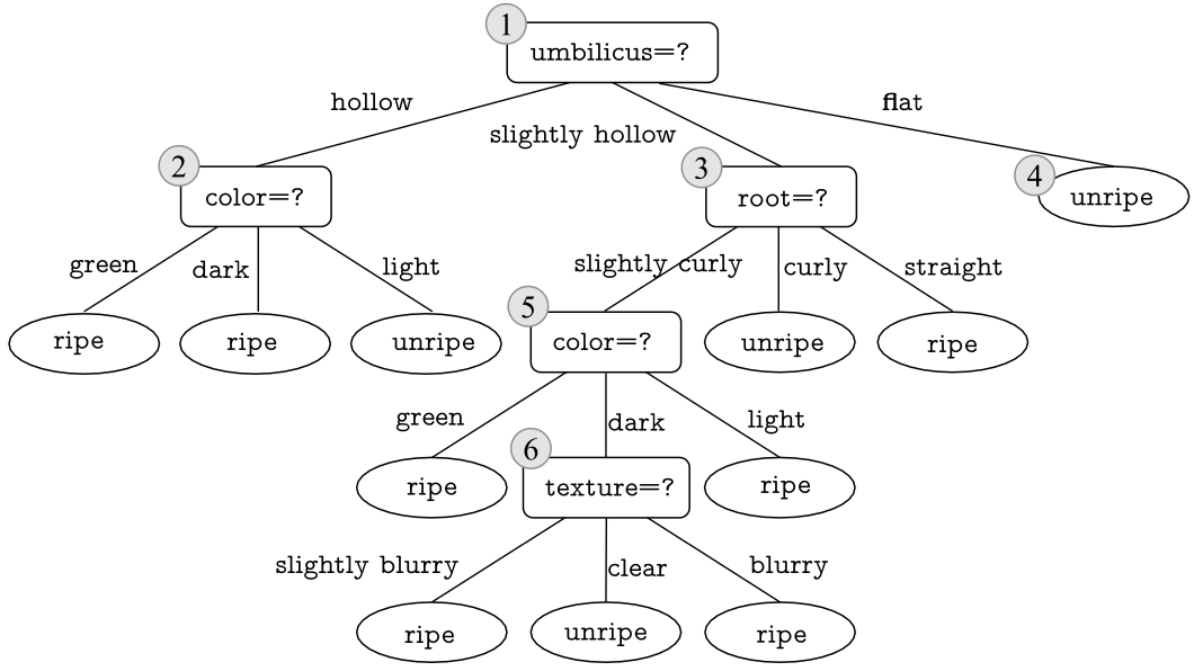
ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
10	green	straight	crisp	clear	flat	soft	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

↑ ***training set*** ↑

↓ ***test set*** ↓

4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false

Post-pruning allows the decision tree to grow completely:



Step 1: check ⑥

Accuracy of splitting? Acc in test set ≈ 42.9%

Post-Pruning

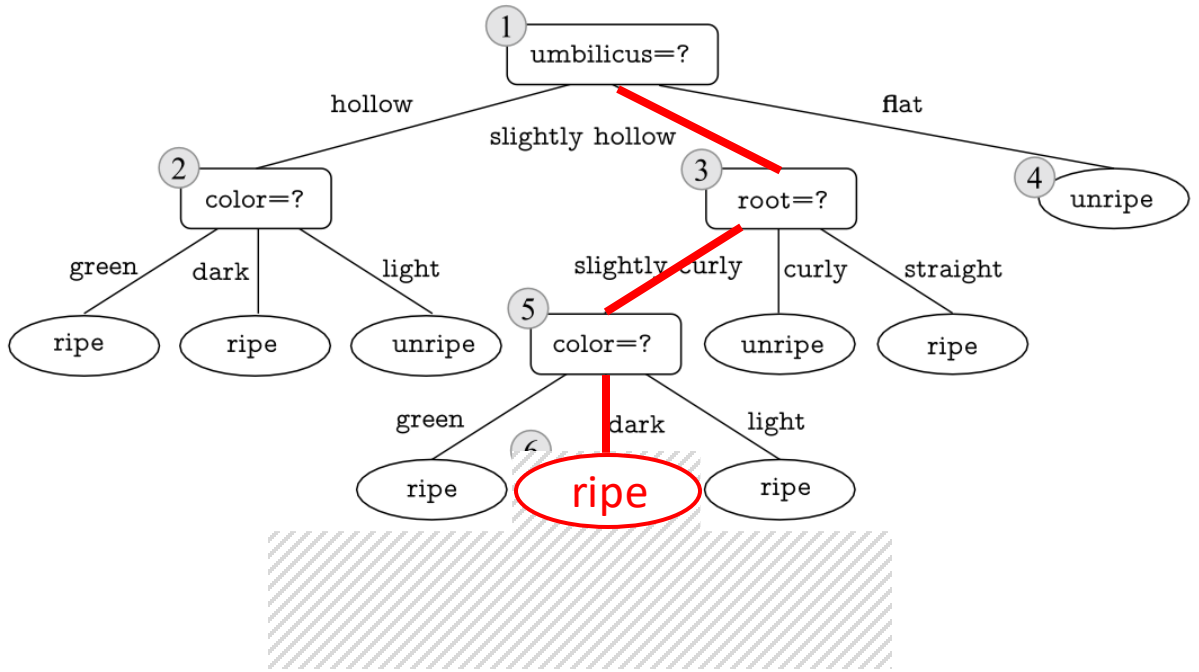
A watermelon dataset

ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
10	green	straight	crisp	clear	flat	soft	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

↑ ***training set*** ↑ ↓ ***test set*** ↓

4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false

Post-pruning allows the decision tree to grow completely:



Step 1: check ⑥

Accuracy of splitting? Acc in test set ≈ 42.9%

Accuracy of pruning? Acc in test set ≈ 57.1%

Generalization improves → Do pruning

Post-Pruning

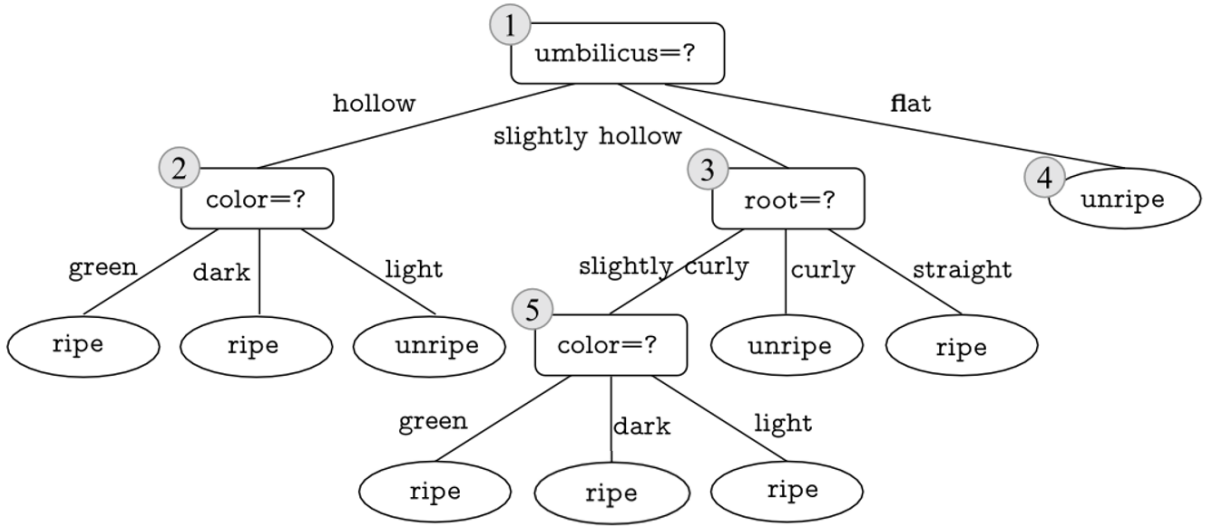
A watermelon dataset

ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
10	green	straight	crisp	clear	flat	soft	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

↑ ***training set*** ↑ ↓ ***test set*** ↓

4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false

Post-pruning allows the decision tree to grow completely:



Step 2: check ⑤

Accuracy of splitting? Acc in test set ≈ 57.1%

Accuracy of pruning? Acc in test set ≈ 57.1%

Generalization unchanged → Do pruning

(Occam's razor principle: prefer simpler models)

Post-Pruning

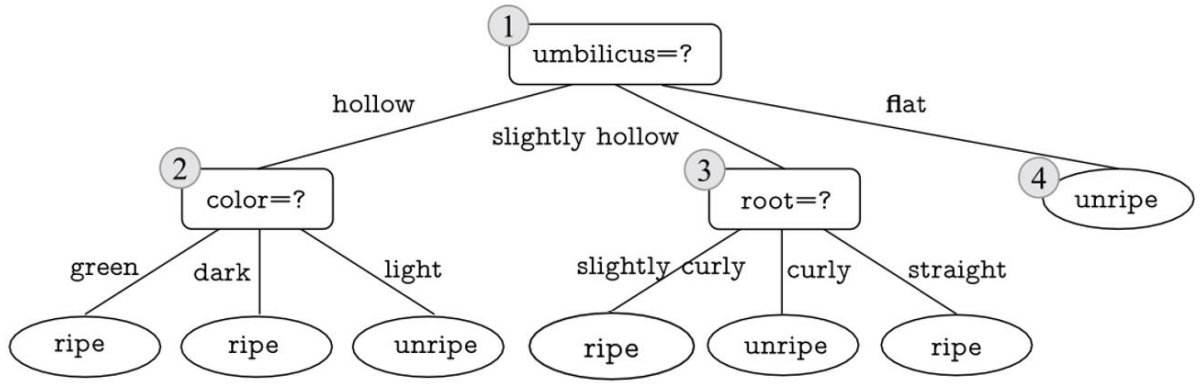
A watermelon dataset

ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
10	green	straight	crisp	clear	flat	soft	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

↑ ***training set*** ↑ ↓ ***test set*** ↓

4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false

Post-pruning allows the decision tree to grow completely:



Step 3: check ②

Accuracy of splitting? Acc in test set ≈ 57.1%

Accuracy of pruning? Acc in test set ≈ 71.4%

Generalization improves → Do pruning

Post-Pruning

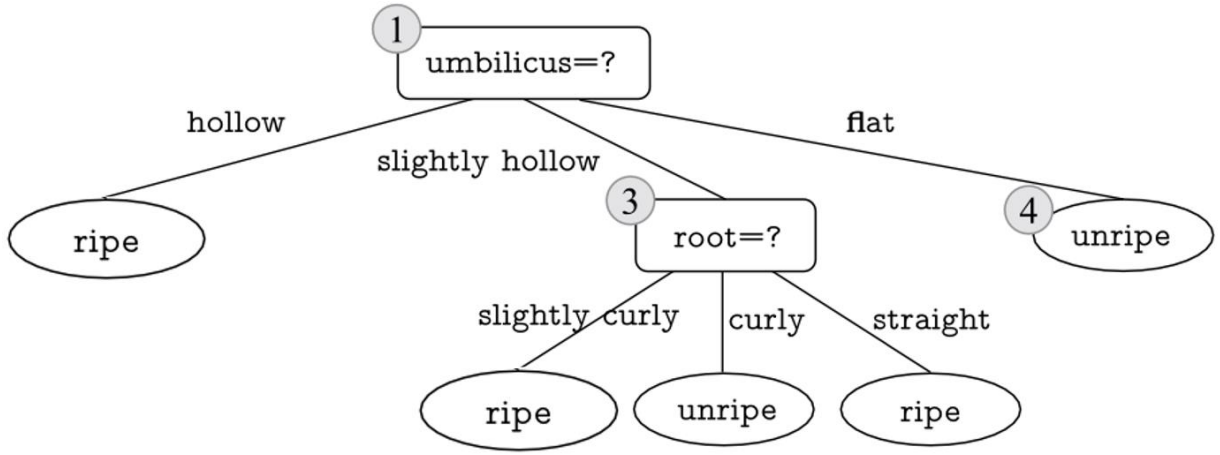
A watermelon dataset

ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
10	green	straight	crisp	clear	flat	soft	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

↑ ***training set*** ↑ ↓ ***test set*** ↓

4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false

Post-pruning allows the decision tree to grow completely:



Step 4: check ③

Accuracy of splitting? Acc in test set ≈ 71.4%

Accuracy of pruning? Acc in test set ≈ 71.4%

Generalization unchanged → Do pruning

Post-Pruning

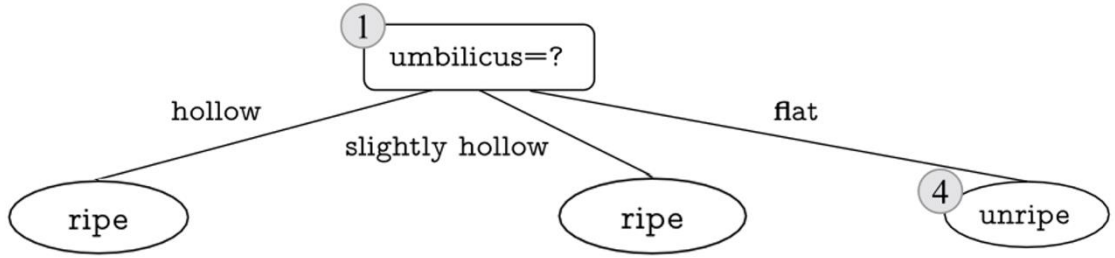
A watermelon dataset

ID	color	root	sound	texture	umbilicus	surface	ripe
1	green	curly	muffled	clear	hollow	hard	true
2	dark	curly	dull	clear	hollow	hard	true
3	dark	curly	muffled	clear	hollow	hard	true
6	green	slightly curly	muffled	clear	slightly hollow	soft	true
7	dark	slightly curly	muffled	slightly blurry	slightly hollow	soft	true
10	green	straight	crisp	clear	flat	soft	false
14	light	slightly curly	dull	slightly blurry	hollow	hard	false
15	dark	slightly curly	muffled	clear	slightly hollow	soft	false
16	light	curly	muffled	blurry	flat	hard	false
17	green	curly	dull	slightly blurry	slightly hollow	hard	false

↑ ***training set*** ↑ ↓ ***test set*** ↓

4	green	curly	dull	clear	hollow	hard	true
5	light	curly	muffled	clear	hollow	hard	true
8	dark	slightly curly	muffled	clear	slightly hollow	hard	true
9	dark	slightly curly	dull	slightly blurry	slightly hollow	hard	false
11	light	straight	crisp	blurry	flat	hard	false
12	light	curly	muffled	blurry	flat	soft	false
13	green	slightly curly	muffled	slightly blurry	hollow	hard	false

Post-pruning allows the decision tree to grow completely:



Step 5: check ①

Accuracy of splitting? Acc in test set ≈ 71.4%

Accuracy of pruning? Acc in test set ≈ 42.9%

Generalization is worse → No pruning

Post-Pruning

- In this particular example, we have the same tree as pre-pruning approach.
- In general, however, post-pruning is less prone to underfitting and leads to better generalization ability compared to pre-pruning.
- Downside: training time of post-pruning is much longer since it takes a bottom-up strategy to examine every non-leaf node in a completely grown decision tree.

Representative Decision Tree Algorithms

- **ID3** (Quinlan, 1979, 1986)
  - Uses Information gain to select attributes.
- **C4.5** (Quinlan, 1993)
  - Uses information gain ratio to select attributes.
  - First find attributes having information gain above average, then select the one with highest information gain ratio.
  - Handles continuous attributes.
  - Does pruning
- **CART** (Breiman et al., 1984)
  - Can do regression as well.

Quinlan, J.R. (1979) **Discovering Rules by Induction** from **Large Collections of Examples**. *Expert Systems in the Micro Electronic Age*.
Quinlan, J.R. (1986) **Induction of decision trees**. *Machine Learning*.
Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning.*
Breiman et al. (1984) *Classification and Regression Trees*.

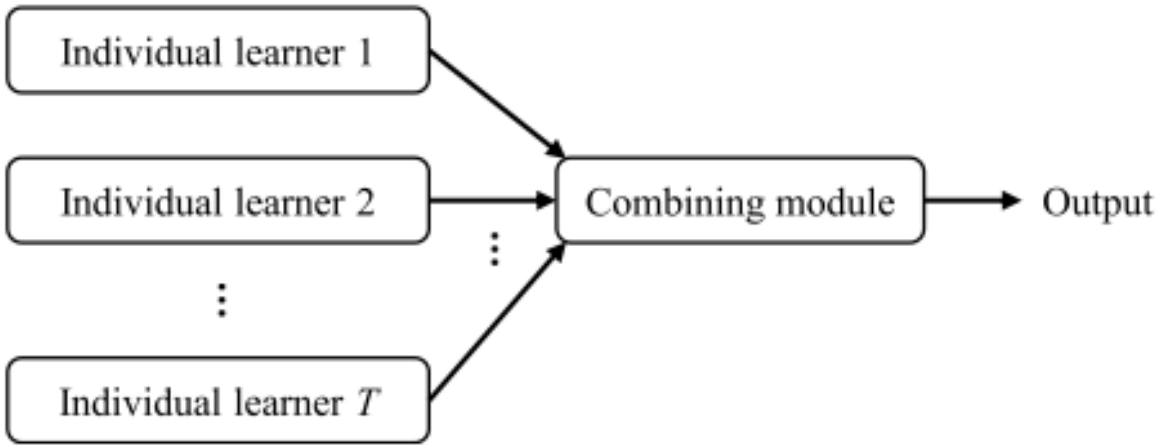
Ensemble Learning and Random Forest

Ensemble Learning

Predictive power of an individual model might be limited.

How about we train multiple models and let them vote?

Base model



Ensemble Learning

Why and when could ensemble learning help?

	Testing sample 1	Testing sample 2	Testing sample 3
h_1	✓	✓	✗
h_2	✗	✓	✓
h_3	✓	✗	✓
Ensemble	✓	✓	✓

(a) Ensemble helps.

	Testing sample 1	Testing sample 2	Testing sample 3
h_1	✓	✓	✗
h_2	✓	✓	✗
h_3	✓	✓	✗
Ensemble	✓	✓	✗

(b) Ensemble doesn't help.

	Testing sample 1	Testing sample 2	Testing sample 3
h_1	✓	✗	✗
h_2	✗	✓	✗
h_3	✗	✗	✓
Ensemble	✗	✗	✗

(c) Ensemble hurts.

In (a), every classifier only has an accuracy of 66.6%, but the ensemble achieves an accuracy of 100%

Base models should be better than a random one (>50%) and are diverse

Ensemble Learning

Why and when could ensemble learning help?

- Suppose the ground-truth classification function is f
- Error rate of each base learner is ϵ.
- For each base learner h_i, we have: $P(h_i(\boldsymbol{x}) \neq f(\boldsymbol{x})) = \epsilon$

- Suppose we ensemble T independent base learners by voting.
- The ensemble will make a correct classification if more than half of the base learners are correct:

$$F(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{T} h_i(\boldsymbol{x})\right)$$

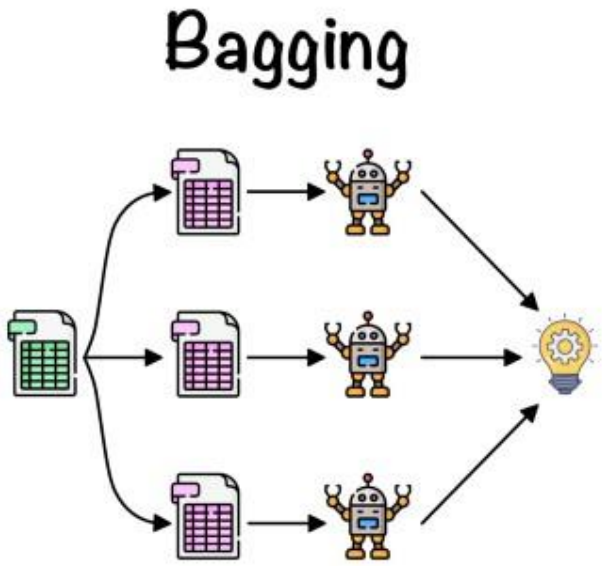
$$P(F(\boldsymbol{x}) \neq f(\boldsymbol{x})) = \sum_{k=0}^{\lfloor T/2 \rfloor} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k} \leqslant \exp\left(-\frac{1}{2} T(1-2\epsilon)^2\right)$$

$T \uparrow \rightarrow$ error decrease exponentially & eventually approach zero.

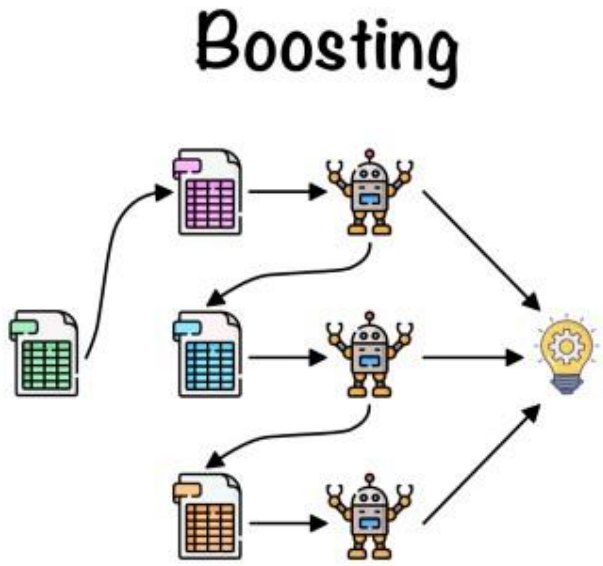
follow from Hoeffding's inequality

Ensemble Learning

Two categories of ensemble learning



Parallel



Sequential

Representative model: Random Forest AdaBoost

Bagging (Bootstrap AGGregatING)

Bootstrap sampling

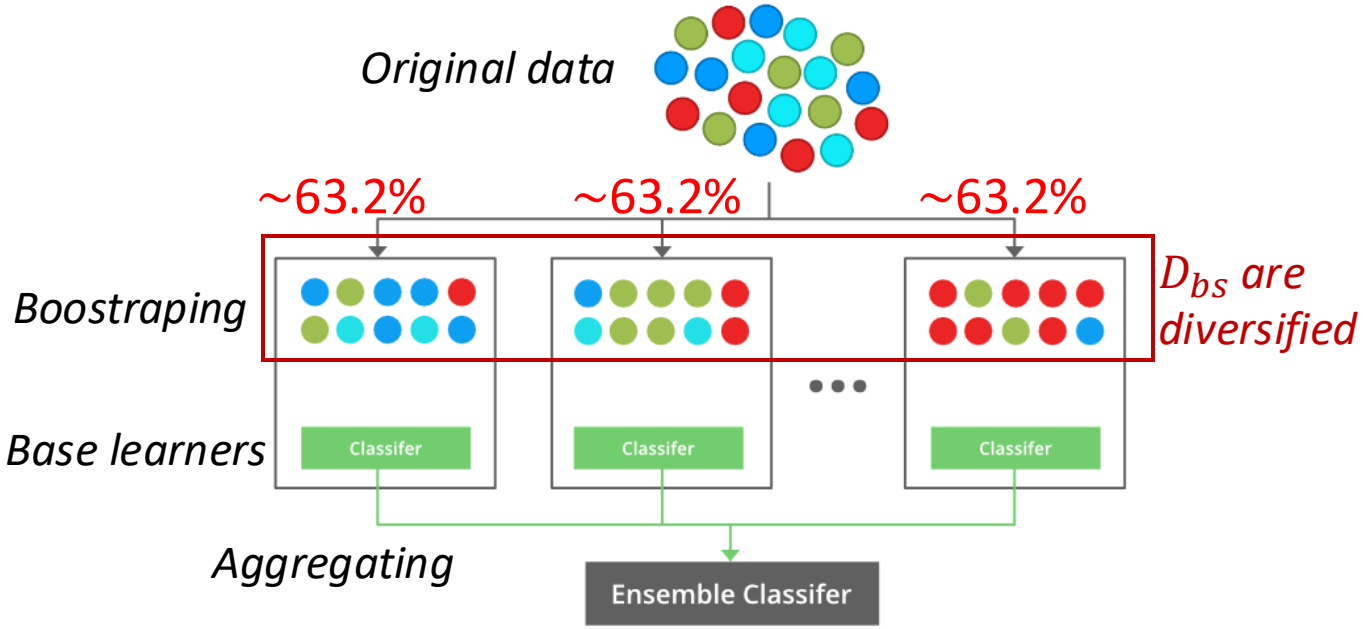
- Given the dataset D with m samples.
- Sample a dataset D' by randomly picking one sample from D, copying it to D', and then putting it back to D.
- Repeat m times: D' contains m samples.
- Also called sampling with replacement.

Chance of not being picked: $\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^m = \frac{1}{e} \approx 0.368$

→ Around 63.2% samples appear in D'

The name "bootstrap" comes from 1785 book *Baron Munchausen's Narrative*, in which Baron Munchausen pulls himself out of a swamp with his straps.

Bagging (Bootstrap AGGregatING)



Algorithm Bagging

Input: Training set: $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathcal{L}$;
Number of training rounds T.

Process:

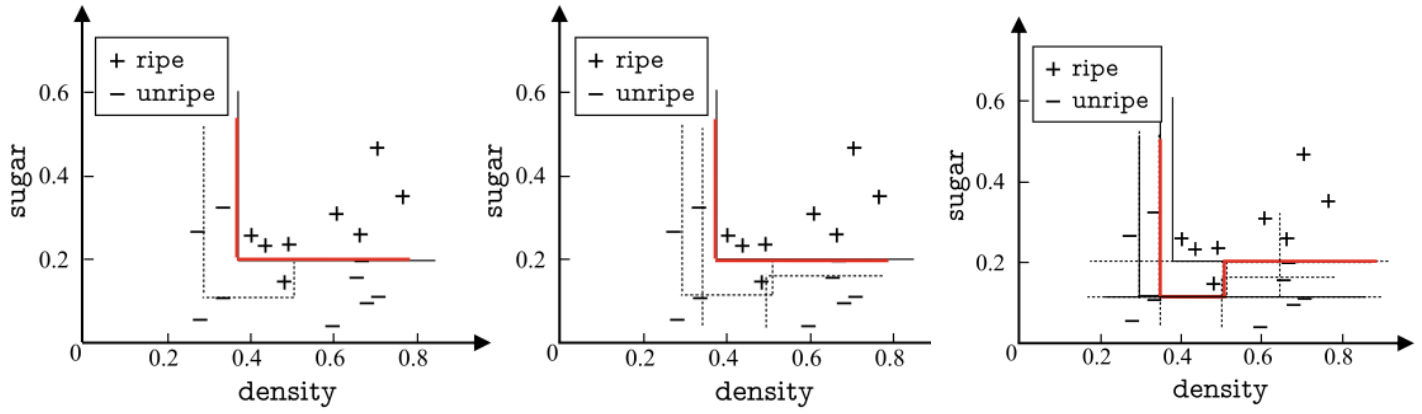
1: **for** $t = 1, 2, \ldots, T$ **do**
2: $h_t = \mathcal{L}(D, \mathcal{D}_{bs})$.
3: **end for**

Output: $H(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$.

Random Forest: Bagging with Decision Trees

Random: (1) Use bootstrap to randomly draw samples;
(2) RF selects from a subset of k features randomly generated from the feature set of the node (instead of all features).
a recommended value $k = \log_2 d$

Forest: make predictions with multiple trees



(a) 3 base learners. (b) 5 base learners. (c) 11 base learners.

Let your voice be heard!

Thank you for your feedback! 🙌