CSC 480/580 Principles of Machine Learning

01 Supervised learning; Decision Trees

Chicheng Zhang

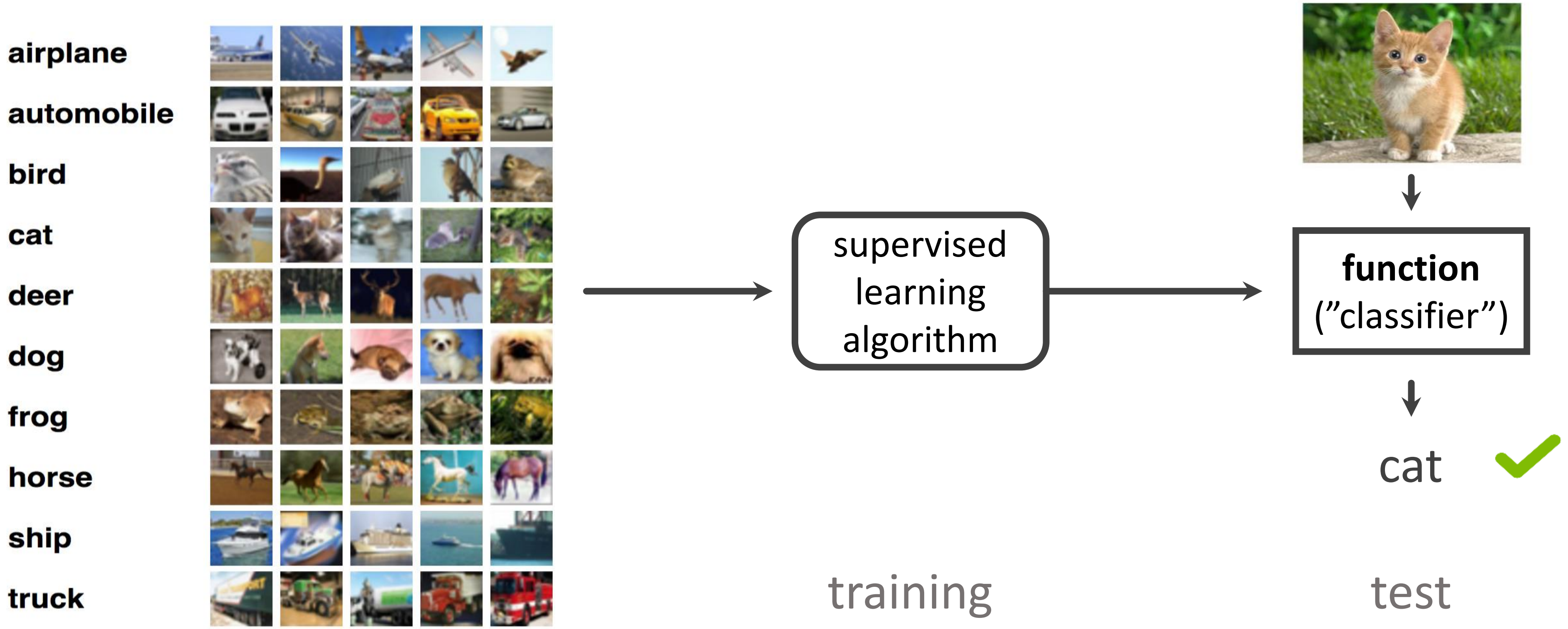
Department of Computer Science

The University of Arizona

The supervised learning problem

Recap: Supervised learning

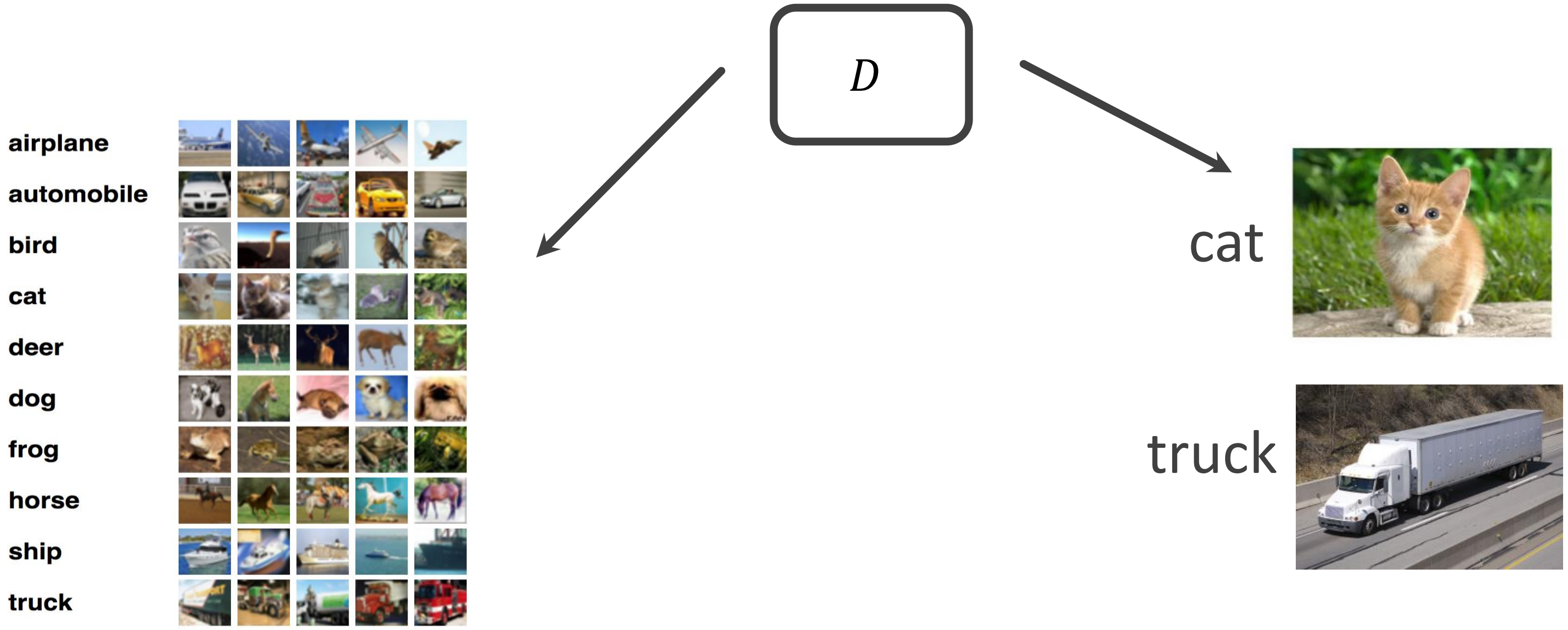
- Training / test data: datasets comprised of *labeled examples*: pairs of (feature, label)



- Question: what makes a test procedure "reasonable"?
  - Test data: should it come from some other population? Should it overlap with the training data?
  - Compare predicted labels with true labels: how?

Supervised learning: formal setup

- Training and test data are drawn independently from the same *data generating distribution* D
  - IID: independent and identically distributed
- Training and test data are independent



Supervised learning: formal setup (cont'd)

- Scenario 1: classification

, cat

function
("classifier")

cat

- Scenario 2: regression

2000 sqft , \$907K

function
("regressor")

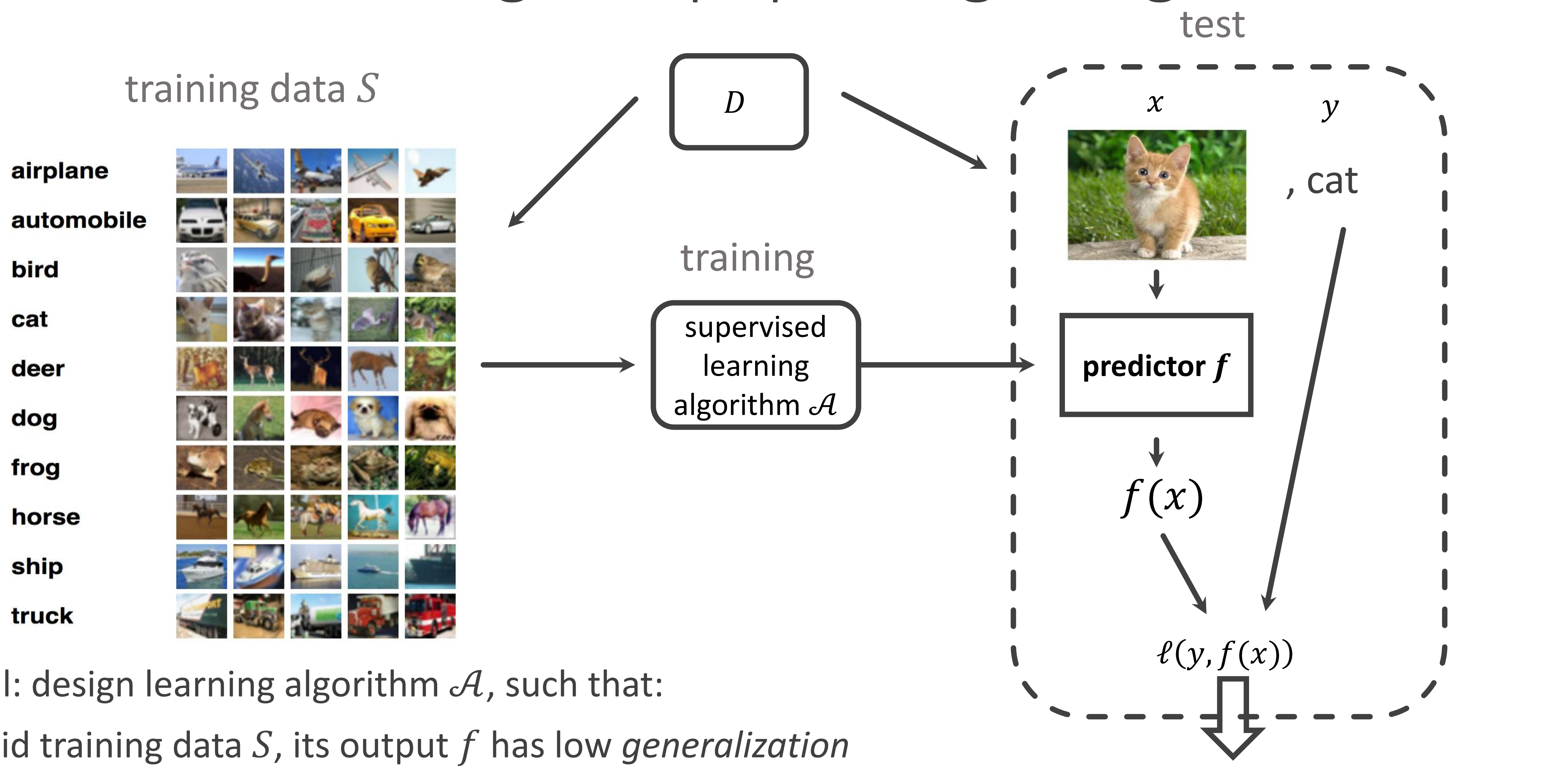
\$840K

How to evaluate?

- Loss function ℓ: measuring the prediction quality with respect to ground truth label

- Examples:
  - Zero-one loss
    $\ell(y, \hat{y}) = I(y \neq \hat{y})$ - classification
  - Square loss
    $\ell(y, \hat{y}) = (y - \hat{y})^2$ - regression
  - Absolute loss:
    $\ell(y, \hat{y}) = |y - \hat{y}|$ - regression

Supervised learning setup: putting it together



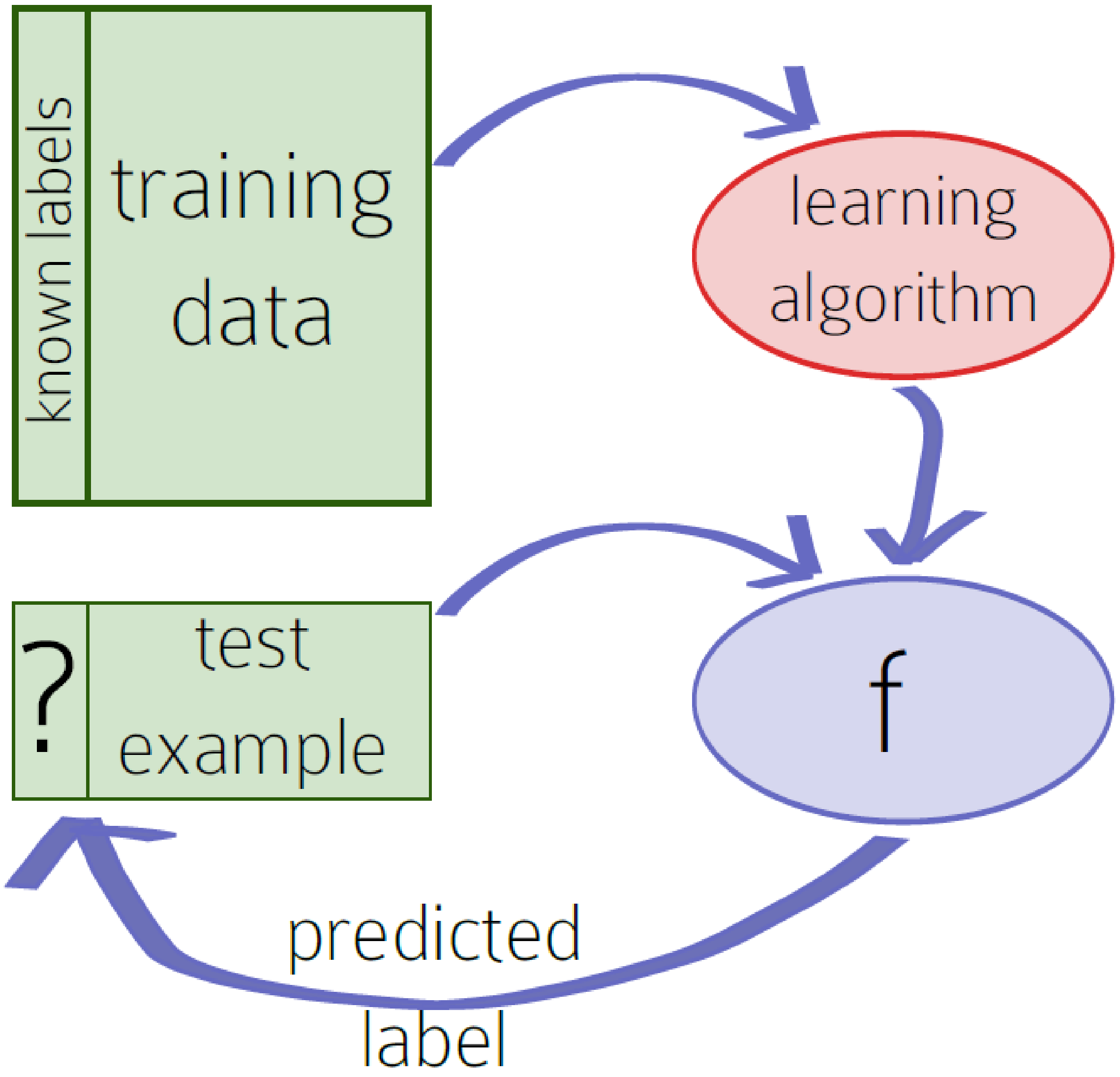
- Goal: design learning algorithm $\mathcal{A}$, such that:

  on iid training data S, its output f has low *generalization error* $L_D(f)$

Generalization error: $L_D(f) = \mathrm{E}_{(x,y)\sim D}\, \ell(y, f(x))$

Supervised learning algorithm: decision trees

Supervised Learning



Goal Learn function f from training data that makes predictions on unseen test data

Question Why is it important that the learning algorithm doesn't see test examples during training?

Prototypical supervised learning problems:

- Regression
- Classification (binary / multiclass)
- Ranking
- …

This lecture will focus on binary classification…

Example: course recommendation

Task *Given a student, recommend a set of courses that s/he would like*

We are allowed to ask a sequence of questions…

You: Is the course under consideration in Systems?
Me: Yes
You: Has this student taken any other Systems courses?
Me: Yes
You: Has this student liked most previous Systems courses?
Me: No
You: *I predict this student will not like this course.*

The Machine Learning approach:

course description
student info. → function → rating $\in \{+, -\}$

Model: Decision Tree

Use our questions to build a binary tree:

You: Is the course under consideration in Systems?
Me: Yes
You: Has this student taken any other Systems courses?
Me: Yes
You: Has this student liked most previous Systems courses?
Me: No
You: *I predict this student will not like this course.*

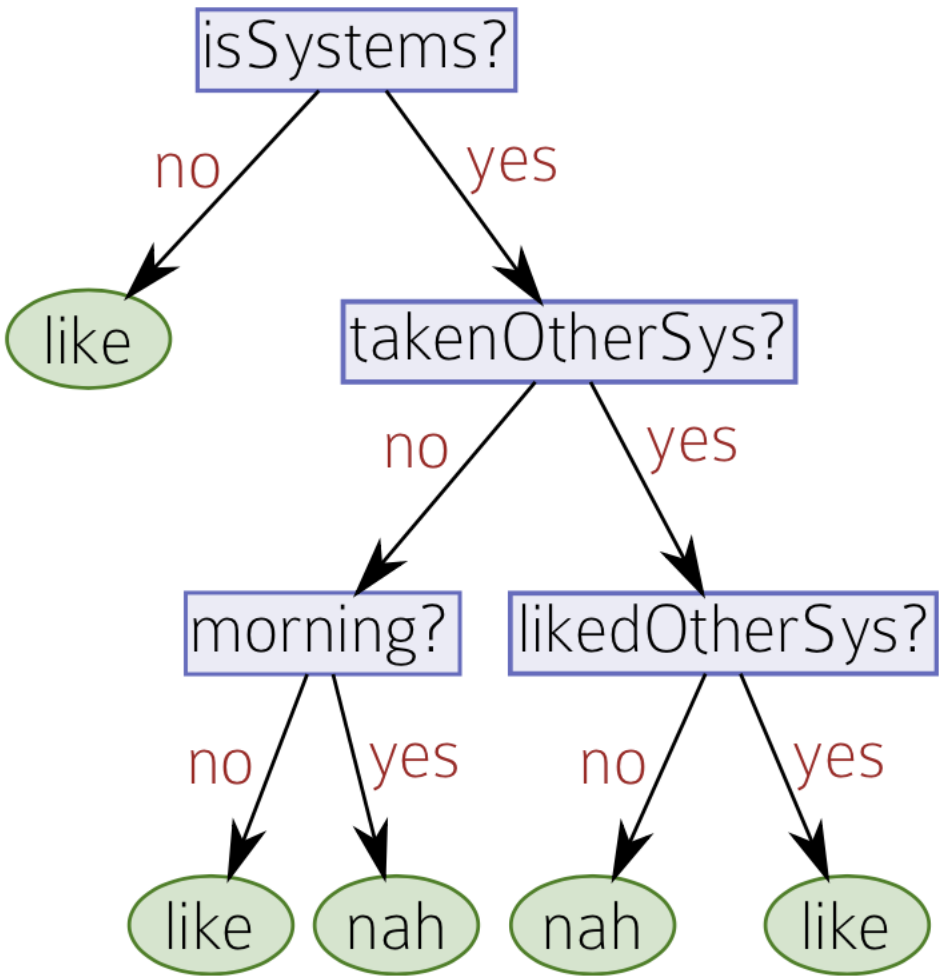


Figure 1.2: A decision tree for a course recommender system, from which the in-text "dialog" is drawn.

Terminology:

- Question & Answer → Feature
- Set of Question & Answers → Training Data
- "Like" / "Nah" → Labels

Training Dataset

Define the labeled training dataset $S = \{(x_i, y_i)\}_{i=1}^{m}$

To make this a <u>binary</u> classification we set
"Like" = {+2,+1,0}
"Nah" = {-1,-2}

Features

Feature Values

Labels

Data Point

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

Decision trees: basic terminology

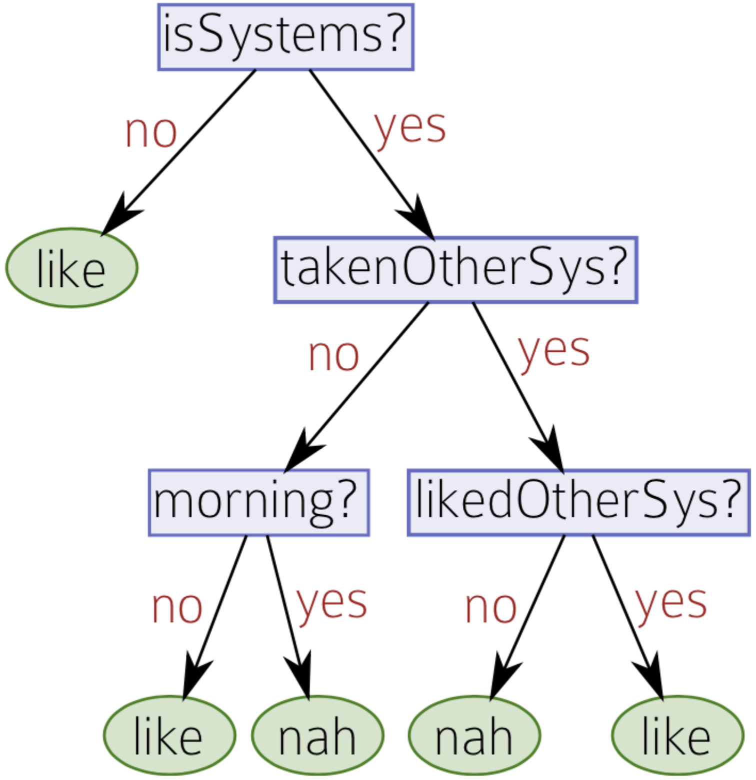


Figure 1.2: A decision tree for a course recommender system, from which the in-text "dialog" is drawn.

node
root node
leaf node
internal node

parent
children
subtree
depth

- Key advantage of using decision trees for decision making: *intepretability*
- Useful in consequential settings, e.g. medical treatment, loan approval, etc.

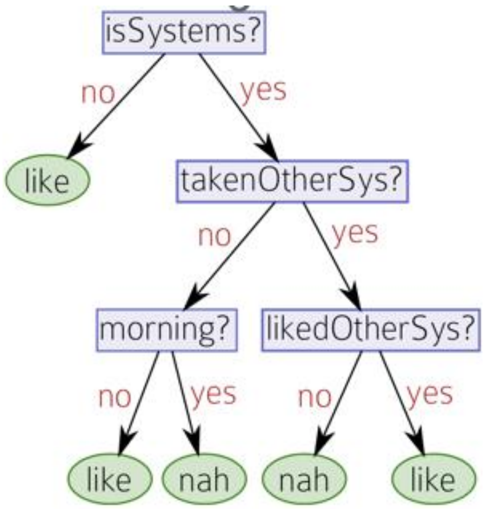
nodes organized in a tree-based structure, leading to a prediction (Fig. 1). The interpretability of decision trees allows physicians to understand why a prediction or stratification is being made, providing an account of the reasons behind the decision to subsequently accept or override the model's output. This interaction between humans and algorithms can provide

Prediction using decision trees

- Test: predict using a decision tree:

Algorithm 2 DecisionTreeTest(*tree*, *test point*)

if *tree* is of the form Leaf(*guess*) **then**
    return *guess*
else if *tree* is of the form Node(f, *left*, *right*) **then**
    if $f = no$ in *test point* **then**
        return DecisionTreeTest(*left*, *test point*)
    else
        return DecisionTreeTest(*right*, *test point*)
    end if
end if



guess=prediction

left=no
right=yes

- Training: how to design a learning algorithm $\mathcal{A}$ that can build trees f from training data?

Learning Decision Trees

Example Guess a number between 1 and 100. Which set of questions are better? Why?

Set 1	**Set 2**
1) Greater than 20? **Y**	1) Greater than 50? **Y**
2) Has a 7 in it? **N**	2) Greater than 75? **N**
3) Odd? **N**	3) Greater than 63? **N**

How many questions should this problem require?

Key Intuition: The decision tree should try to ask informative questions

Training accuracy / error

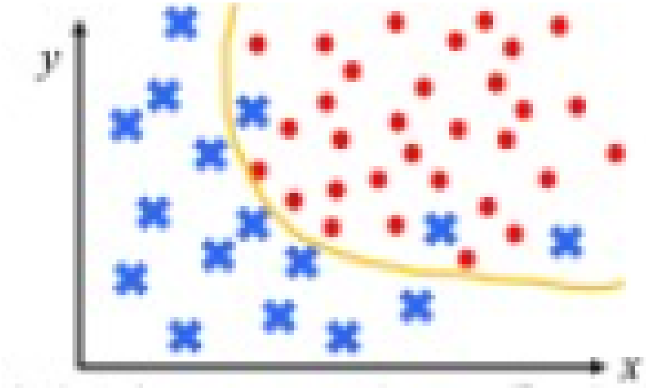
- The training data $S = \{(x_i, y_i)\}_{i=1}^m$

- Predictor f's training accuracy = fraction of examples in S that are **correctly classified** by f
- In formula,

$$A_S(f) = \frac{1}{m}\sum_{i=1}^{m} I(f(x_i) = y_i)$$

- Training error $L_S(f) = 1 - A_S(f) = \frac{1}{m}\sum_{i=1}^{m} I(f(x_i) \neq y_i)$

- The "Empirical risk minimization" (ERM) approach:
  - Idea: f has low $L_S(f) \Rightarrow f$ has low $L_D(f)$
  - Approach: Find f with lowest $L_S(f) \Leftrightarrow f$ with highest $A_S(f)$
  - Problem with this approach?



Decision tree training: single level case

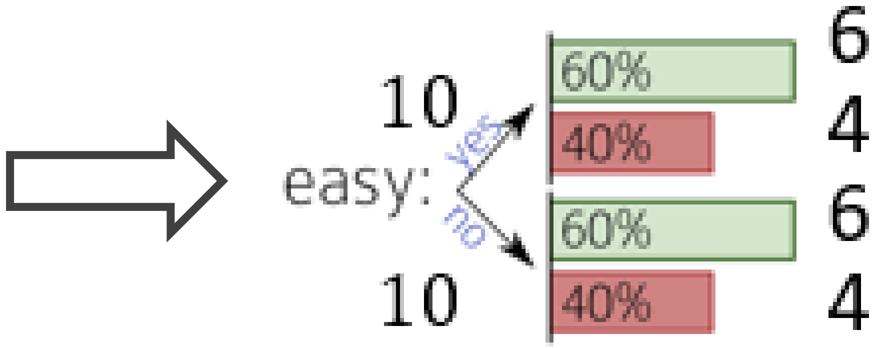
- Q: if I could only ask one question (design a depth-1 tree), what question would I ask?

- Intuition: look at the histograms of labels for each feature

- Example: feature "easy"

Histogram calculation

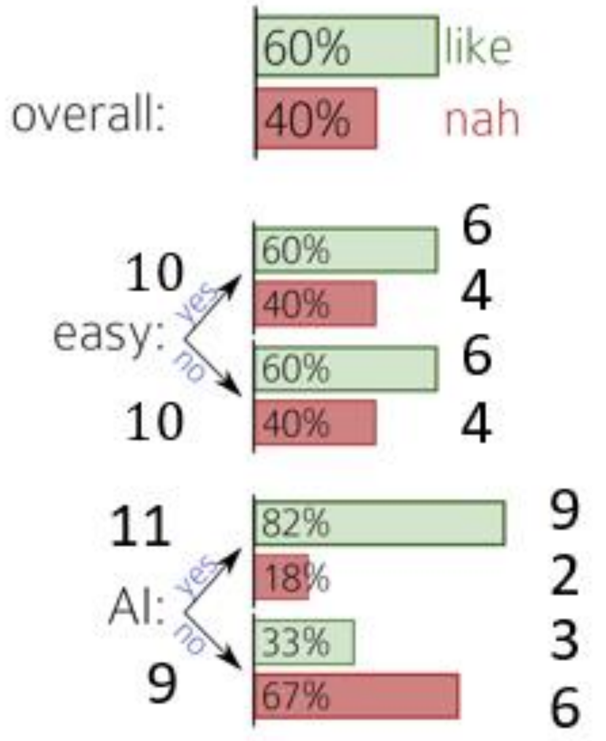
Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y



Decision tree training: single level case

- Q: if I could only ask one question (design a depth-1 tree), what question would I ask?

- Intuition: look at the histograms of labels for each feature
- Which feature (question) is better, 'easy' or 'AI'?



- Best training accuracy using 'easy':

(max(6,4) + max(6,4)) / 20 = 0.6

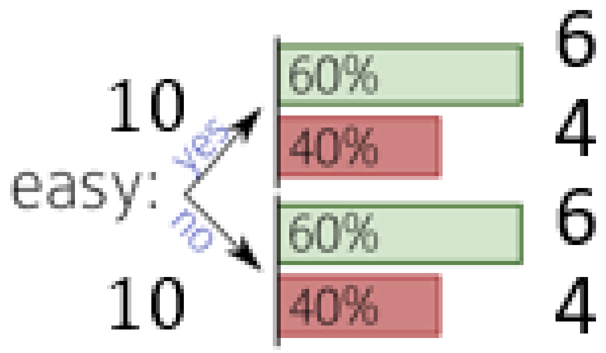
- Best training accuracy using 'AI':

(max(9,2) + max(3,6)) / 20 = 0.75

- In other words, 'AI' is better (more informative in telling apart like / nah)

Decision tree training: single level case

- In formula:
- $\mathrm{Score}(f, S) \coloneqq \max\left(P_S(y=+, x_f = yes), P_S(y=-, x_f = yes)\right)$
  $+\max\left(P_S(y=+, x_f = no), P_S(y=-, x_f = no)\right)$

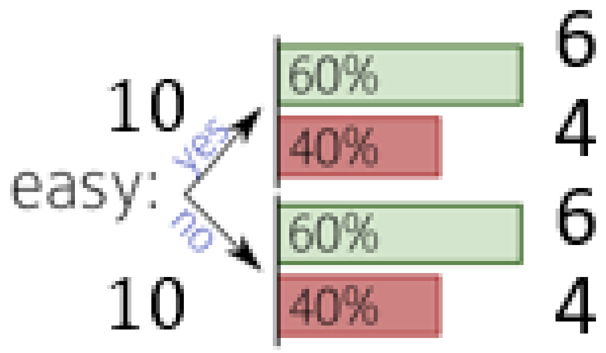


- Here, $P_S(E)$ denotes the probability of event E if an example (x, y) is chosen uniformly from S
- In other words: the frequency of E in sample S
- E.g.

$$\mathrm{Score}('\mathrm{easy}', S) = \max\left(\frac{6}{20}, \frac{4}{20}\right) + \max\left(\frac{6}{20}, \frac{4}{20}\right) = 0.6$$

Decision tree training: single level case

- In formula:



- $\text{Score}(f, S) \;\coloneqq \max\big(P_S(y=+, x_f=yes), P_S(y=-, x_f=yes)\big)$
$$+\max\big(P_S(y=+, x_f=no), P_S(y=-, x_f=no)\big)$$

- Written in conditional probability form: label “purity” measure

$P(AB) = P(A)P(B \mid A)$

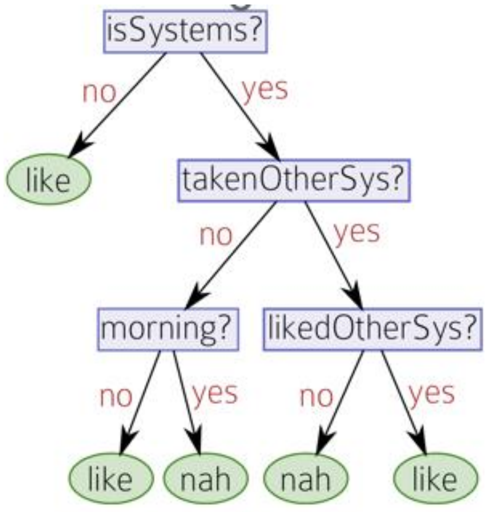
$$= \boxed{\max\big(P_S(y=+ \mid x_f=yes), P_S(y=- \mid x_f=yes)\big)} \cdot P_S(x_f=yes)$$
$$+\max\big(P_S(y=+ \mid x_f=no), P_S(y=- \mid x_f=no)\big) \cdot P_S(x_f=no)$$

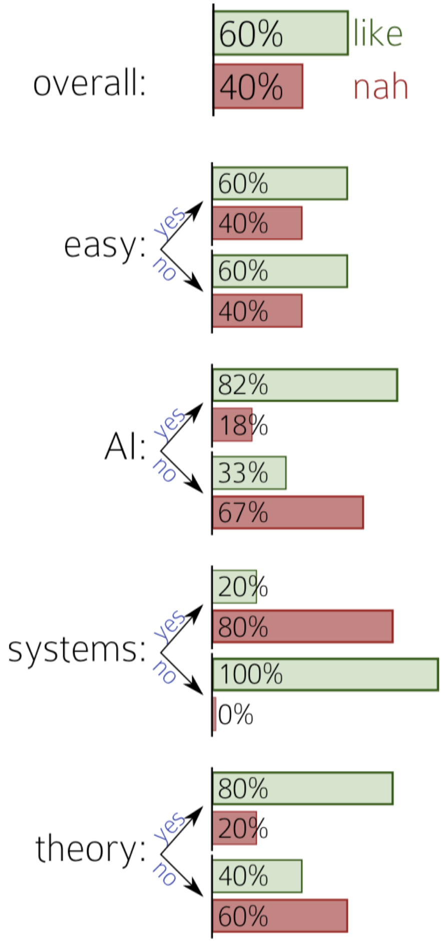
- e.g. $\text{Score}('\text{easy}', S) = \max(0.6, 0.4) \times 0.5 + \max(0.6, 0.4) \times 0.5 = 0.6$
- Interpretation: $\text{Score}(f, S)$ measures the ability of feature f in predicting label y – **informativeness** of feature f

Decision tree training: general level case

- High-level idea: greedy + divide & conquer

- Build the root of the tree greedily
- Build the left and left subtrees *recursively*
- When to stop the recursion?



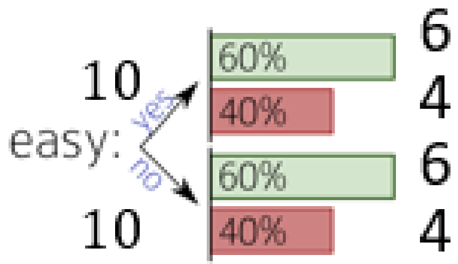


Algorithm 1 DecisionTreeTrain(*data*, *remaining features*)

```
guess ← most frequent answer in data                    // default answer for this data
if the labels in data are unambiguous then
    return Leaf(guess)                                  // base case: no need to split further
else if remaining features is empty then
    return Leaf(guess)                                  // base case: cannot split further
else                                                    // we need to query more features
    for all f ∈ remaining features do
        NO ← the subset of data on which f=no
        YES ← the subset of data on which f=yes
        score[f] ← # of majority vote answers in NO
                      + # of majority vote answers in YES
                              // the accuracy we would get if we only queried on f
    end for
    f ← the feature with maximal score(f)
    NO ← the subset of data on which f=no
    YES ← the subset of data on which f=yes
    left ← DecisionTreeTrain(NO, remaining features \ {f})
    right ← DecisionTreeTrain(YES, remaining features \ {f})
    return Node(f, left, right)
end if
```

answer=label

unambiguous=achieves 100% acc.



Score(f, S) = "informativeness of f (in predicting y) for dataset S"

Q: is this algorithm guaranteed to terminate?

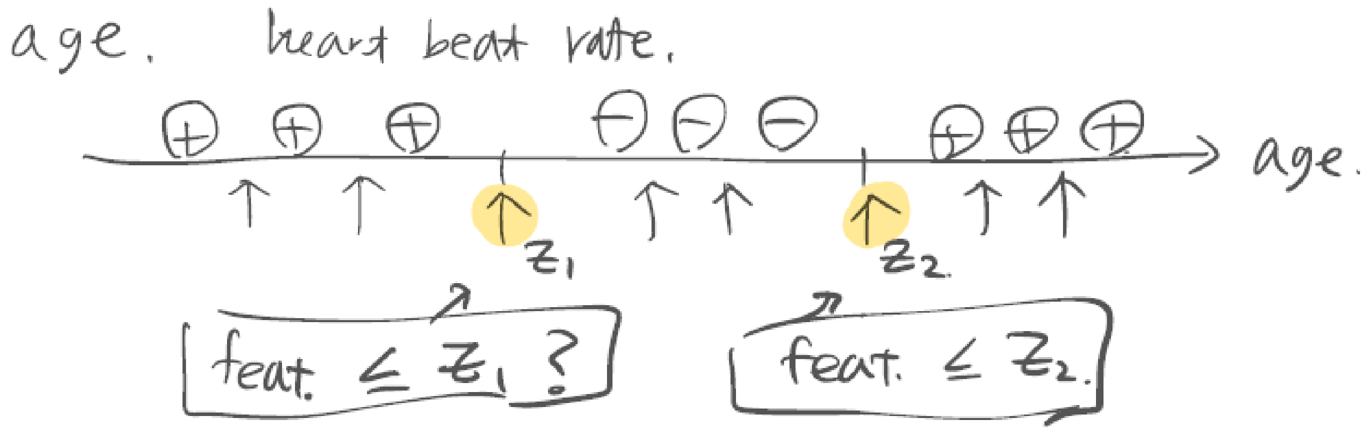
Dealing with various types of features

- Binary: $x_f \in \{0,1\}$
  - Node: $x_f = 0?$

Easy?	AI?	Sys?	Thy?	Morning?
y	y	n	y	n

- Categorical: $x_f \in \{1,2,\ldots,C\}$
  - Node: $x_f \in \{i_1,\ldots,i_l\}?$

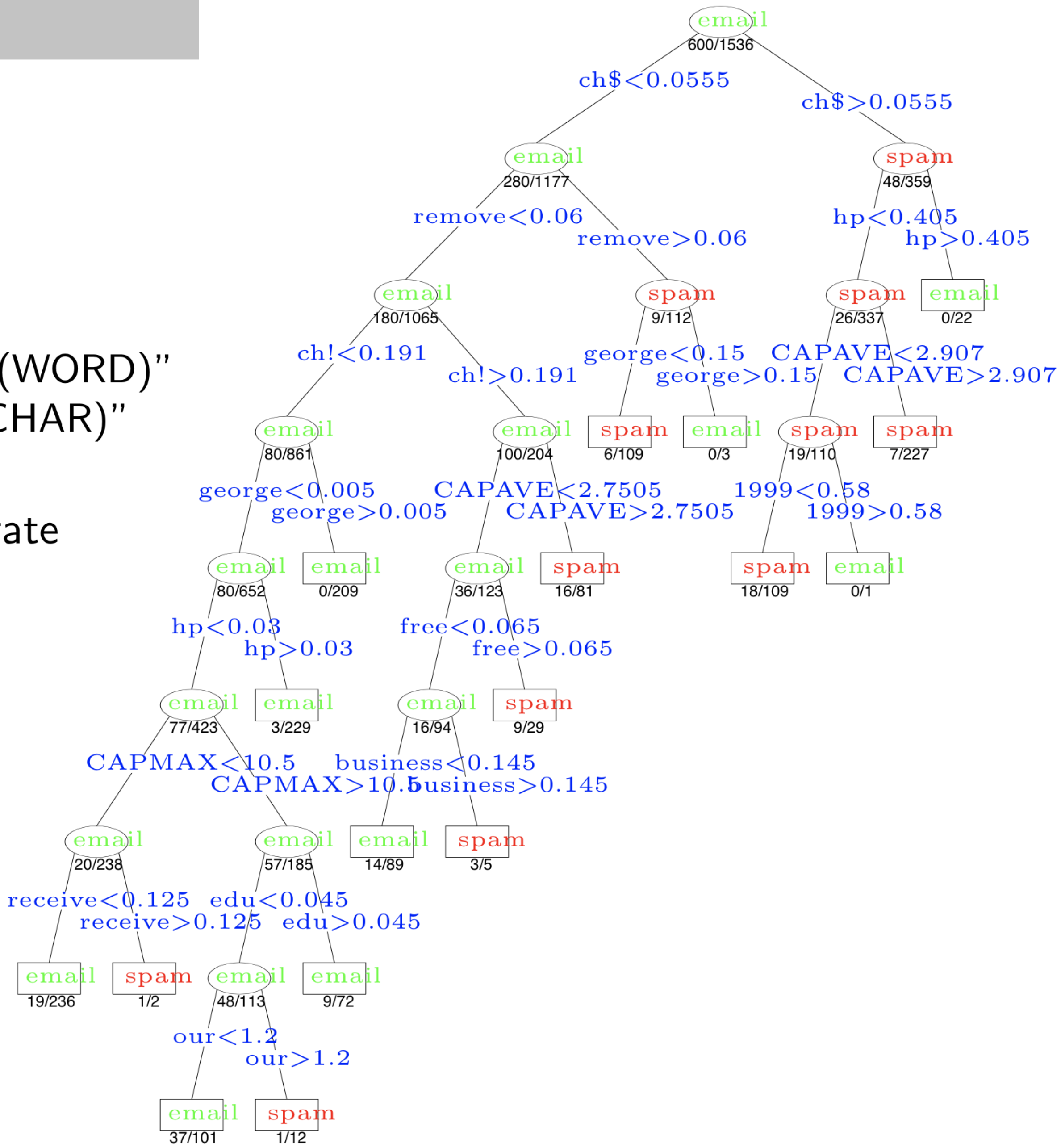
- real value: $x_f \in \mathbb{R}$
  - Node: $x_f \le z?$



https://medium.com/@data.science.enthusiast/ways-to-handle-categorical-data-in-python-e89d25c40338

Example: spam filtering I

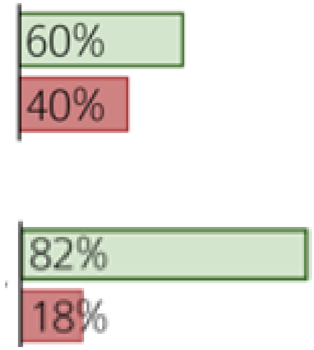
- Spam dataset
- 4601 email messages, about 39% are spam
- Classify message by spam and not-spam
- 57 features
  - 48 are of the form "percentage of email words that is (WORD)"
  - 6 are of the form "percentage of email characters is (CHAR)"
  - 3 other features (e.g., "longest sequence of all-caps")
- Final tree after pruning has 17 leaves, 9.3% test error rate



Q: what is the best depth-0 decision tree, and what is its accuracy?

Decision tree training: generalized informativeness scores

- $\text{Score}(f)$ = a measure of **informativeness** of f
- Alternative view -- **uncertainty reduction**: how much uncertainty about y can be reduced if we know f
- Uncertainty measures of population:



Notions of uncertainty: binary case ($\mathcal{Y} = \{0, 1\}$)

Suppose in a set of examples $S \subseteq \mathcal{X} \times \{0, 1\}$, a $\boxed{p}$ fraction are labeled as 1

1. **Classification error**:

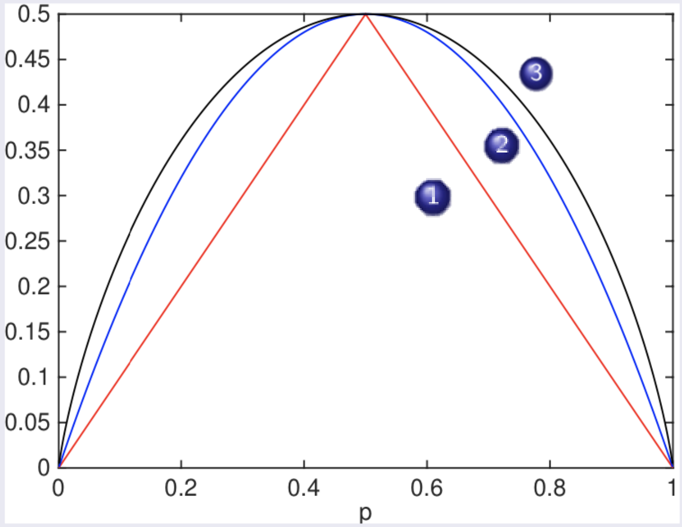
$$u(S) := \min\{p,\ 1 - p\}$$

2. **Gini index**:

$$u(S) := 2p(1 - p)$$

3. **Entropy**:

$$u(S) := p \log \frac{1}{p} + (1-p) \log \frac{1}{1 - p}$$

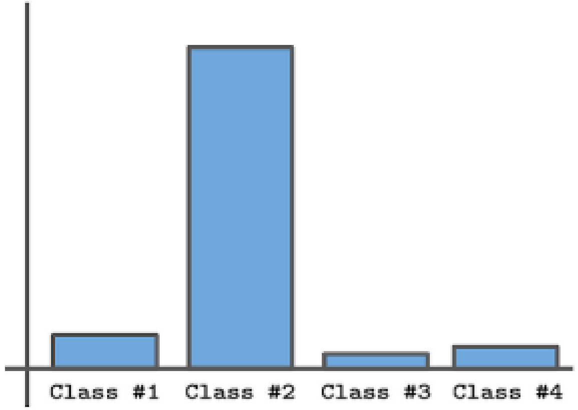


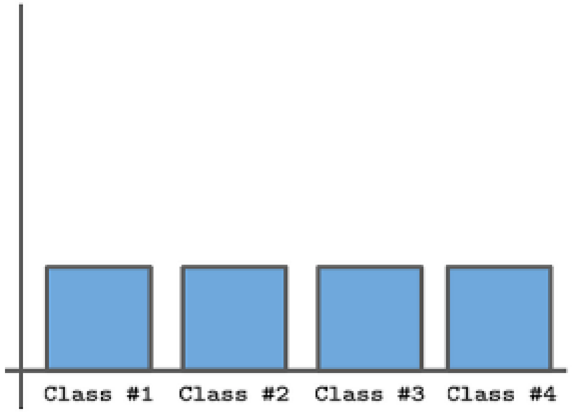
(3) is divided by 2 so the plot looks comparable

log here is base-2

Decision tree training: generalized informativeness scores

- Multiclass classification setting: $\mathcal{Y} = \{1, \dots, K\}$





Notions of uncertainty: general case

Suppose in $S \subseteq \mathcal{X} \times \mathcal{Y}$, a $\boxed{p_k}$ fraction are labeled as k (for each $k \in \mathcal{Y}$).

1. **Classification error**:

$$u(S) := 1 - \max_{k \in \mathcal{Y}} p_k$$

2. **Gini index**:

$$u(S) := 1 - \sum_{k \in \mathcal{Y}} p_k^2$$

3. **Entropy**:

$$u(S) := \sum_{k \in \mathcal{Y}} p_k \log \frac{1}{p_k}$$

Each is *maximized* when $p_k = 1/|\mathcal{Y}|$ for all $k \in \mathcal{Y}$ (i.e., equal numbers of each label in S)

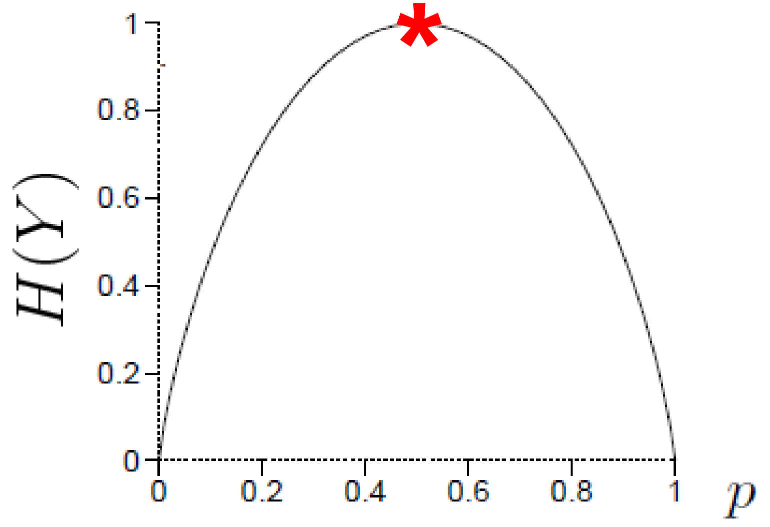
Each is *minimized* when $p_k = 1$ for a single label $k \in \mathcal{Y}$ (so S is **pure** in label)

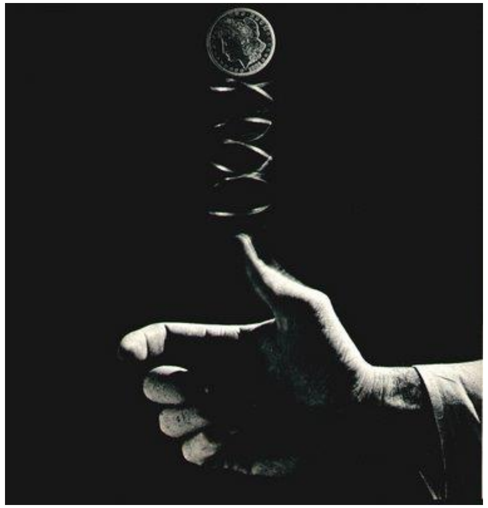
https://towardsdatascience.com/confidence-calibration-for-deep-networks-why-and-how-e2cd4fe4a086

Entropy Uncertainty

Entropy of random variable Y: $H(Y) = \sum_y P(Y = y) \ln \frac{1}{P(Y=y)}$

Coin Flip Example: $Y \sim \mathrm{Bernoulli}(p)$





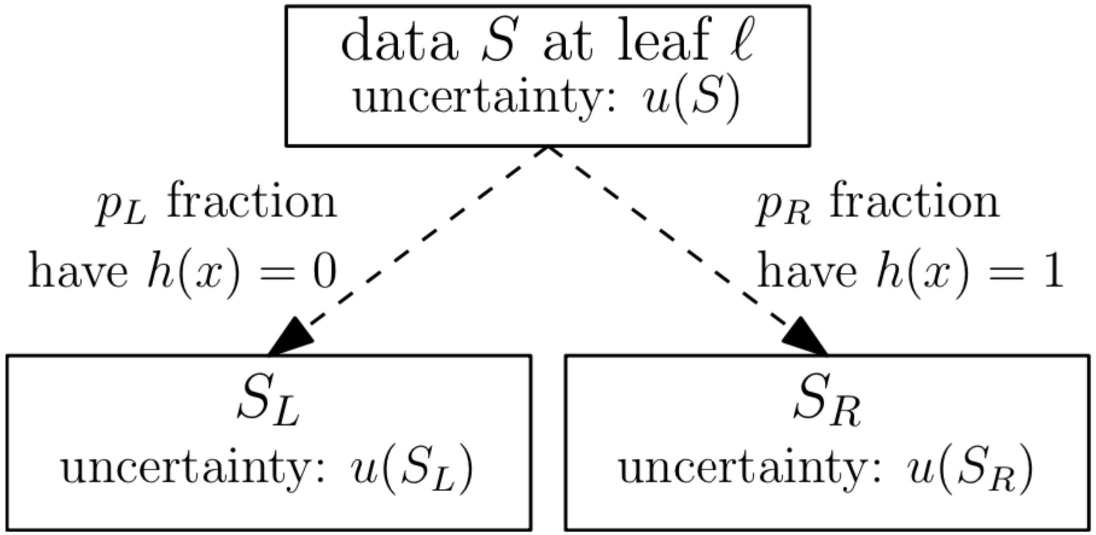
- Key object studied in **<u>information & coding theory</u>**

- Interpretations:
  - the minimum number of bits needed to reliably encode Y
  - the uncertainty about outcome of Y

Maximum uncertainty when coin is <u>fair</u>.

Decision tree training: generalized informativeness scores

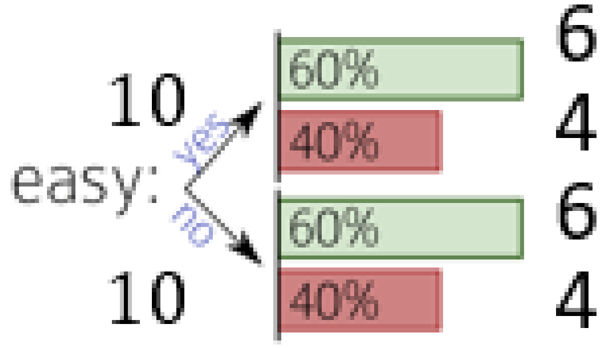
Suppose the data S at a leaf ℓ is split by a rule h into S_L and S_R, where $p_L := |S_L|/|S|$ and $p_R := |S_R|/|S|$



The **reduction in uncertainty** from using rule h at leaf ℓ is

$$u(S) - \Big(p_L \cdot u(S_L) + p_R \cdot u(S_R)\Big) \quad \boxed{=: \mathrm{Score}(h, S) \ \text{(Generalized)}}$$

Generalized informativeness score in action



- Example: consider **entropy uncertainty**, on dataset S

  $u(S) = \sum_{y_0 \in \{+,-\}} P_S(y = y_0) \log \frac{1}{P_S(y=y_0)}$

- $\text{Score}(S, f) = u(S) - \left(p_L\, u(S_L) + p_R\, u(S_R)\right)$

$$\boxed{P_S\left(x_f = no\right)}$$

$$\boxed{\sum_{y_0 \in \{+,-\}} P_S\left(y = y_0 \mid x_f = no\right) \log \frac{1}{P_S\left(y = y_0 \mid x_f = no\right)}}$$

- In the above example:
  - $p_L = 0.5, p_R = 0.5$
  - $u(S) = 0.6 \log \frac{1}{0.6} + 0.4 \log \frac{1}{0.4}$,
  - $u(S_L) = u(S_R) = 0.6 \log \frac{1}{0.6} + 0.4 \log \frac{1}{0.4}$

  $\Rightarrow \text{Score}('\text{ easy}', S) = 0$

- In this case, $\text{Score}(S, f)$ is also known as the **mutual information** between x_f and y (under P_S)

Mutual Information

$$I(X;Y) = H(X) - H(X \mid Y)$$

Recall: $H(X) = \sum_x P(X=x) \ln \frac{1}{P(X=x)}$

- Measures entropy reduction after observing Y

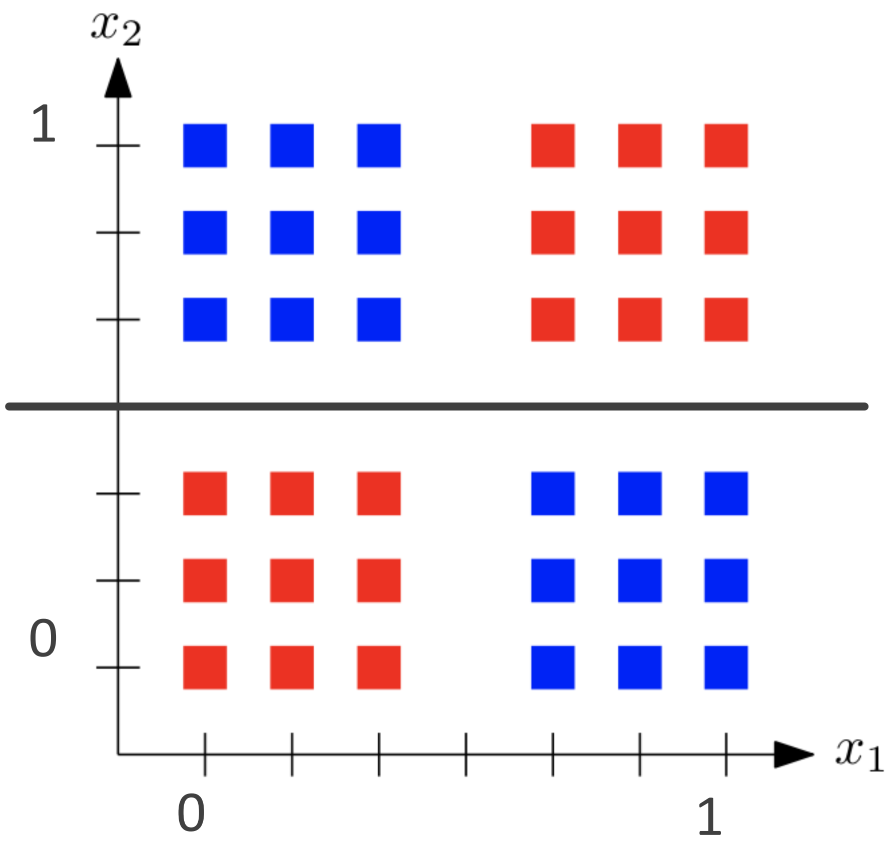
- How much information does Y carry about X?

Stop splitting when there is no reduction in uncertainty? This is a bad idea!

The ‘XOR’ data:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

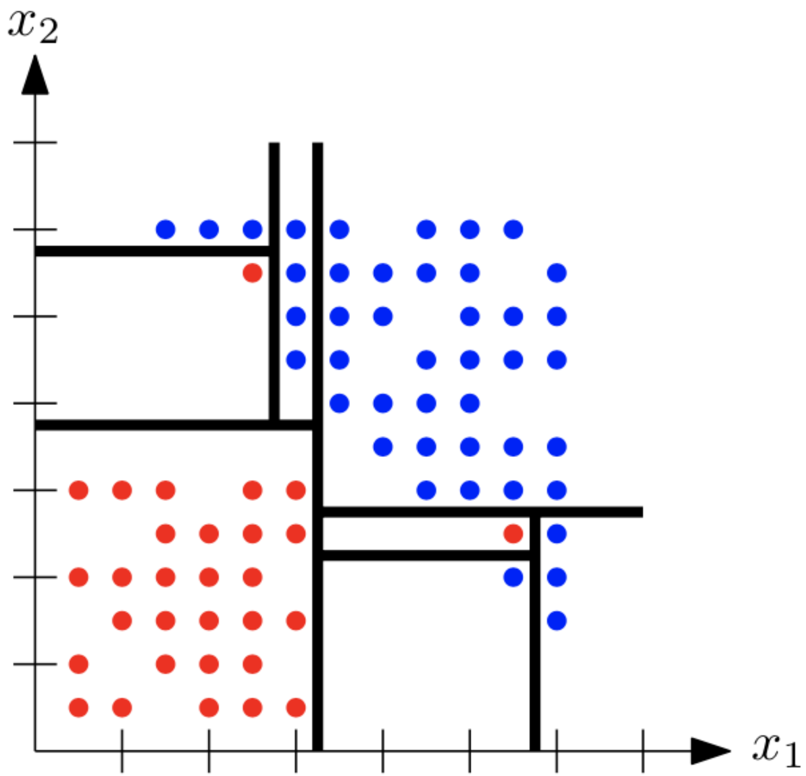
Suppose $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{Y} = \{\text{red}, \text{blue}\}$, and the data is as follows:



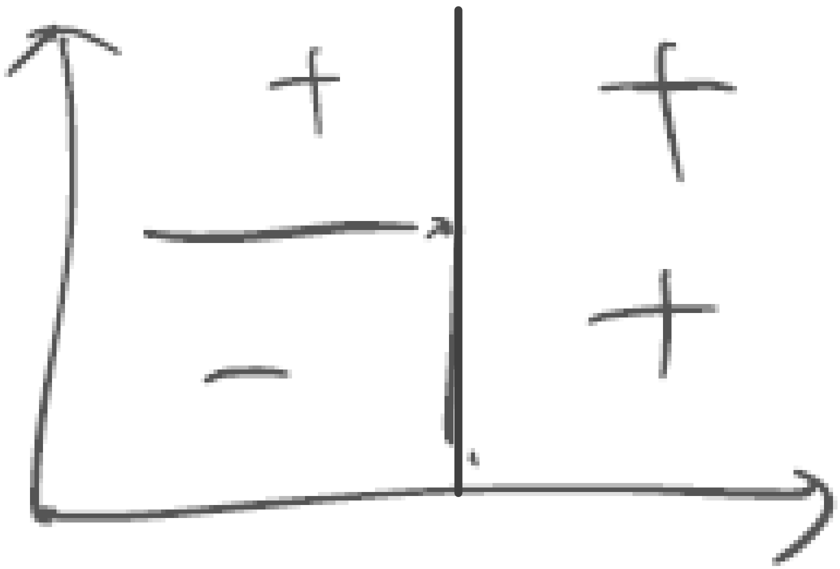
- Any axis-aligned split has no reduction on uncertainty on S
- However, a depth-2 decision tree (with axis-aligned splits) has zero training error

Overfitting can happen

- "Spurious" patterns can be learned.



A better alternative:



- A fix to the training algorithm: stop recursion & return leaf once we reach depth k (say $k = 2$)
- Alternatively: do **pruning** on trained decision tree – an ongoing research topic

Next lecture (1/18)

- Supervised learning: what to do if the data distribution is *known*?

- Models, parameters, hyperparameters

- Practical considerations

- Assigned reading: CIML Chap. 2 (Limits of learning)
- HW0 due

Generalization error vs. training error

- The training data $S = \{(x_i, y_i)\}_{i=1}^{m}$

- Given a predictor f, its training error $L_S(f) = \mathrm{E}_{(x,y)\sim S}\, \ell(y, f(x)) = \frac{1}{m}\sum_{i=1}^{m} \ell(y_i, f(x_i))$
- Consider zero-one loss, $\ell(y, \hat{y}) = I(y \neq \hat{y}) \Rightarrow L_S(f) = \mathrm{E}_{(x,y)\sim S}\, I(y \neq f(x)) = P_{(x,y)\sim S}\,(y \neq f(x))$

- Heuristic: f with low $L_S(f) \Rightarrow f$ with low $L_D(f)$
  - Also known as the "Empirical risk minimization" (ERM) approach
  - Issues with ERM?

- How easy is it to compute a decision tree f that minimize $L_S(f)$?
  - k-node decision tree, d-dimensional data $\Rightarrow$ at least $O(d^k)$ time complexity
  - Can we design efficient algorithms?

Training Dataset

Define the labeled training dataset $S = \{(x_i, y_i)\}_{i=1}^{m}$

Features

Feature Values

Labels

Data Point

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

To make this a binary classification we set
“Like” = {+2,+1,0}
“Nah” = {-1,-2}