Artificial Intelligence

- Learning with Decision Trees -

HOCHSCHULE
LANDSHUT

Prof. Dr. Eduard Kromer
Winter Semester 2023/2024
University of Applied Sciences Landshut

Content

- Introduction to Learning
- Decision Trees as Data Structures
- Decision Trees for Classification
- Maximizing Information Gain
- Decision Tree Learning Algorithms
- Decision Trees for Regression
- Computational Complexity
- Advantages and Disadvantages

Learning

- What is learning?

  An agent learns when it improves its performance w.r.t. a specific task with experience.

- There is no intelligence without learning!

Agents that learn - Supervised Learning

- we want to build agents that learn from observations about the world and are able to improve their performance on future tasks
- in this lecture, we will focus only on supervised learning:
  - we have a training set of N example-label pairs

    $$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

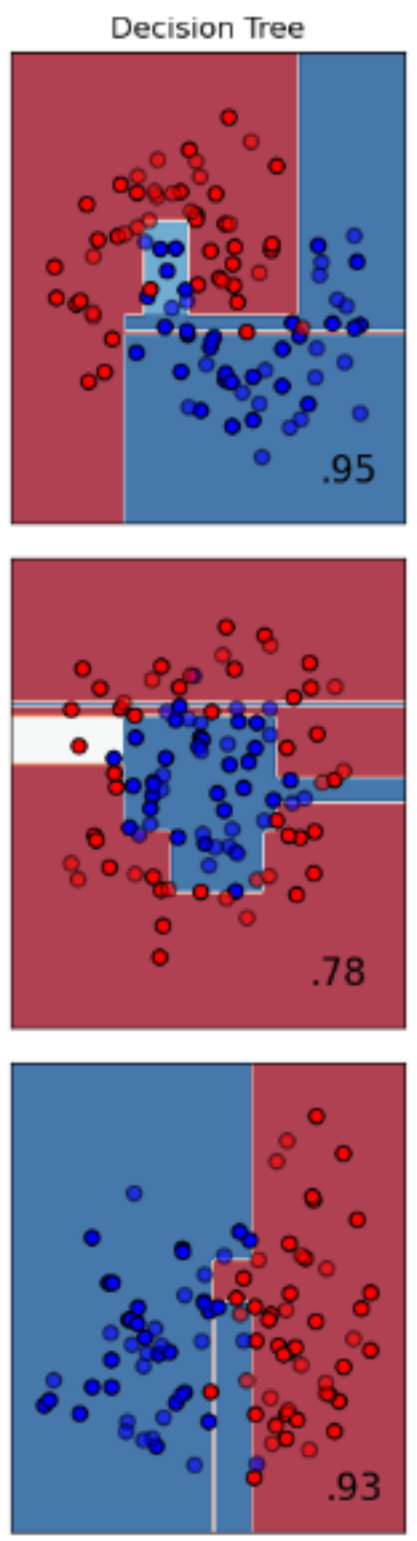
    that were generated by an unknown function f $[f(x) = y]$
  - our goal is to discover a function $\hat{f}$ that approximates f well
  - supervised learning involves learning a function from examples of its inputs and outputs
  - discrete output → classification problem; continuous output → regression problem

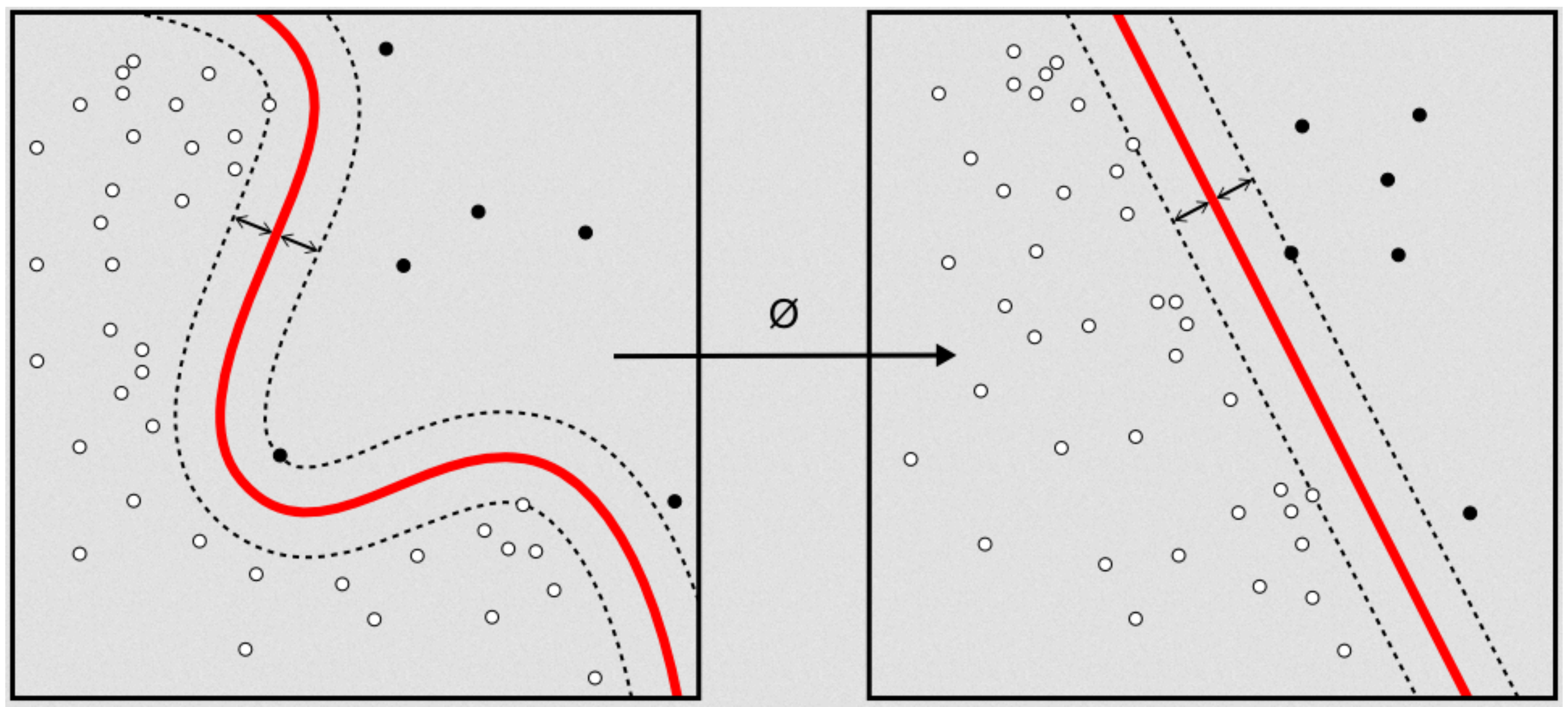
	Outlook	Temperature	Humidity	Wind	Play Tennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

X y

Learning Algorithm

- a learning algorithm receives as input a learning sample and returns a function $\hat{f}$:
- It is defined by
    - a hypothesis space H, which is a set of candidate models
    - a quality measure for a model
    - an optimization strategy

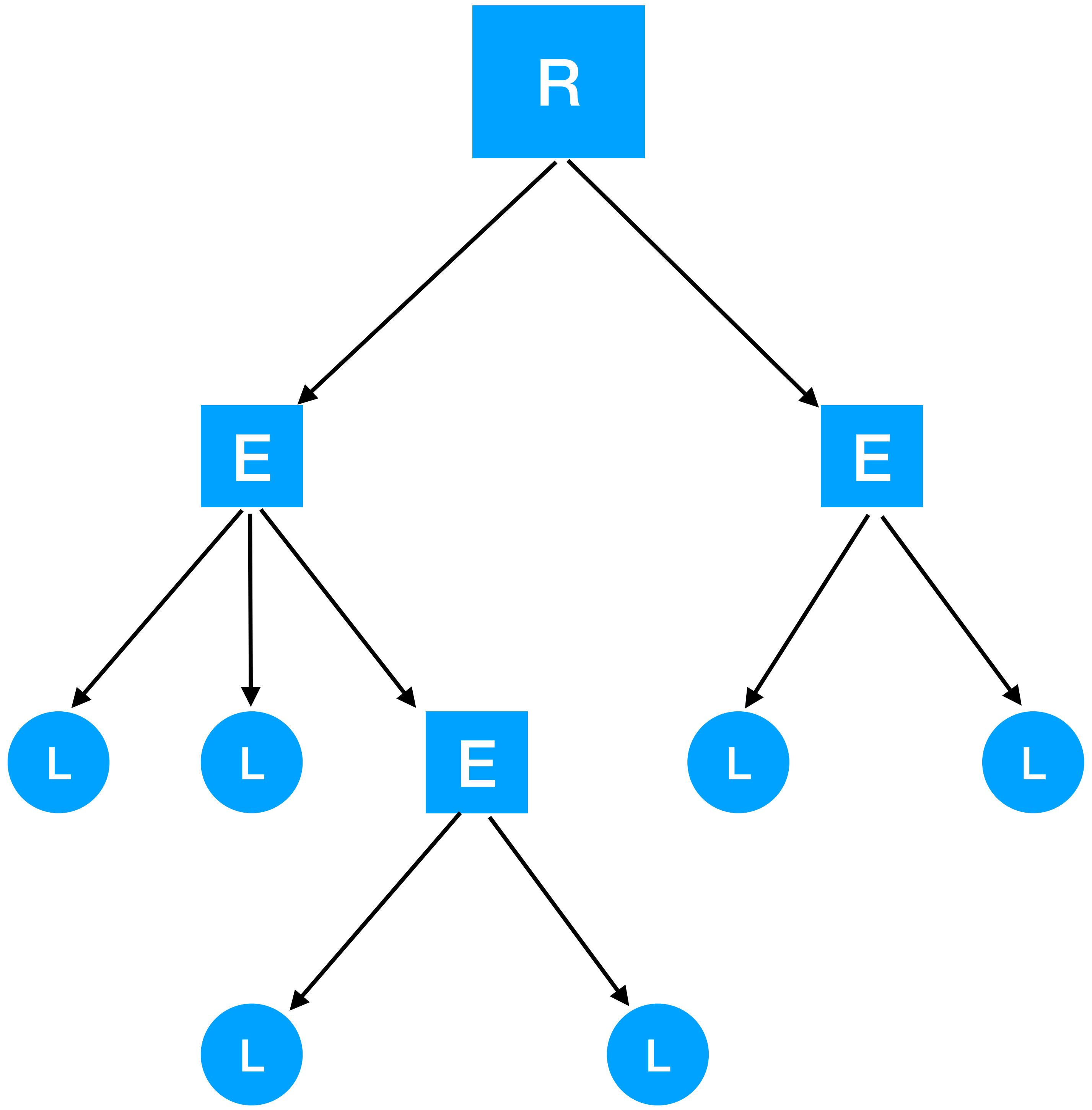




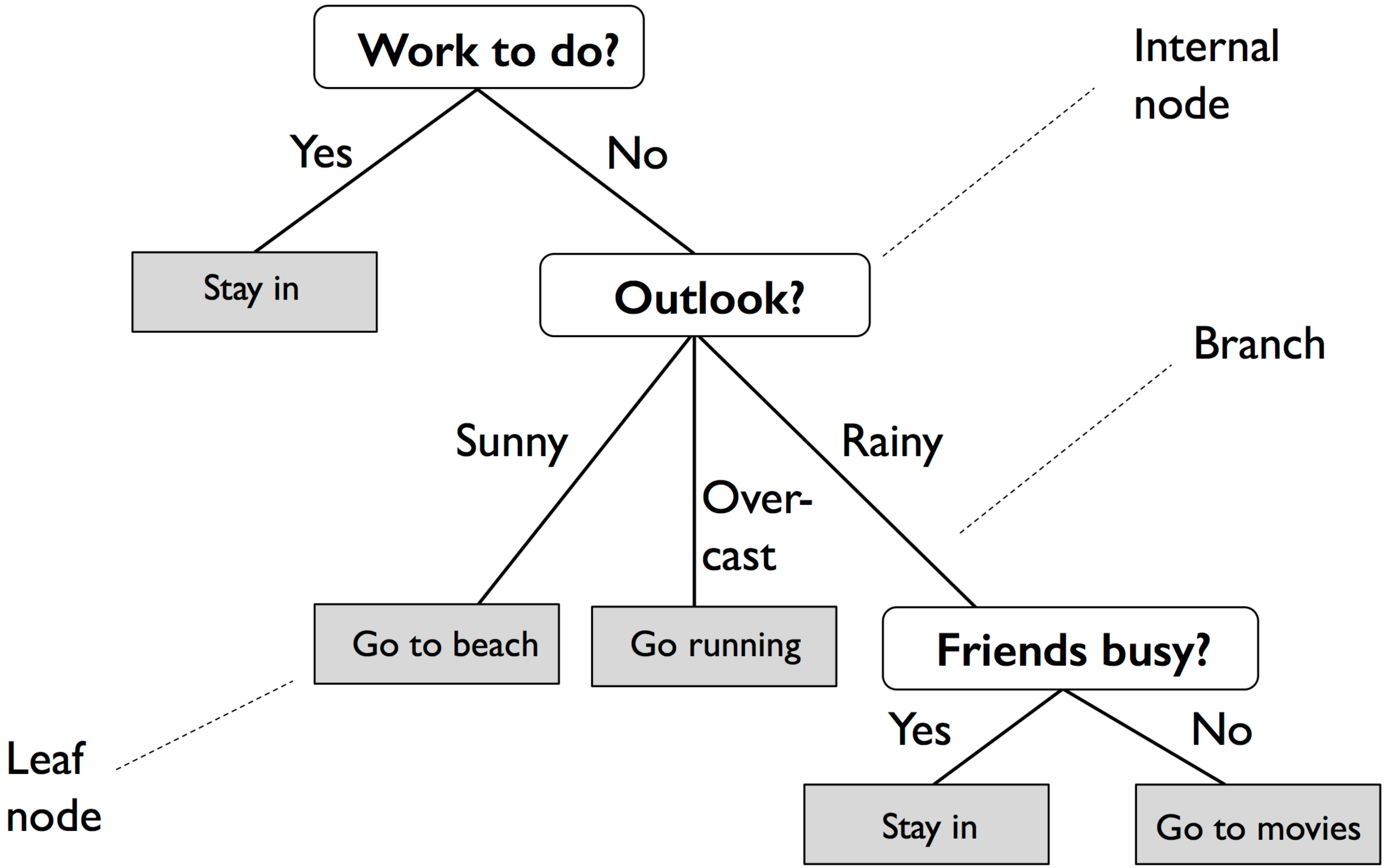
Source: Wikipedia

Decision Trees as Data Structures

- *Decision Trees* are based on the data structure of a tree
  - a tree consists of a root (R) at the top - this is where the decision process starts
  - leafs (L) represent possible outcomes of the decision process (classification or regression result)
  - at the root (R) and all the edges (E) a decision is made and a specific path is followed
  - the training data is used to build / learn the tree structure - this is the ML part



Decision Trees as Data Structures



Source: Raschka and Mirjalili. *Python Machine Learning*

Decision Trees for Classification

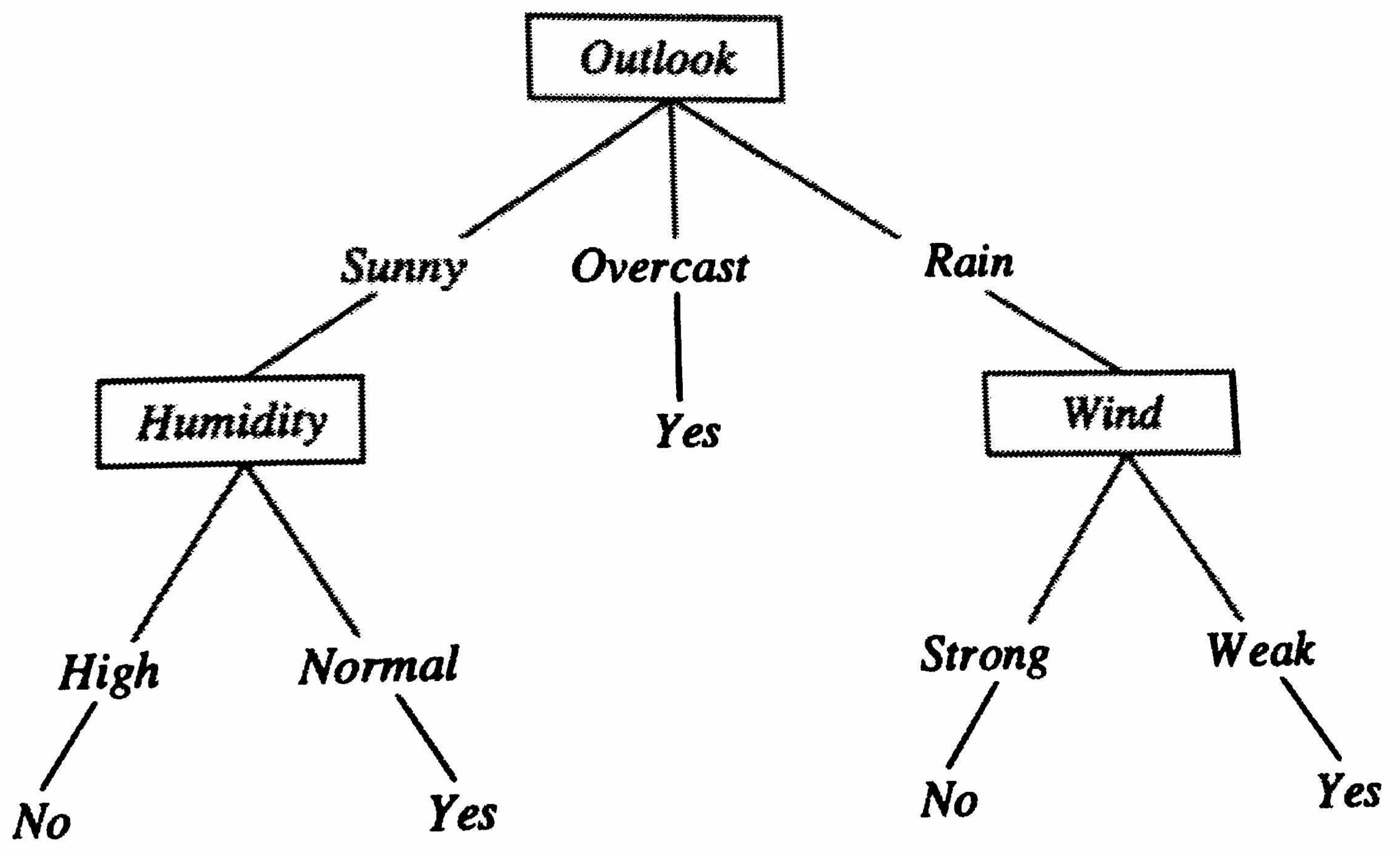
- DTs classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance
- each node in the tree specifies a test of some attribute of the instance
  - each branch descending from that node corresponds to one of the possible values for this attribute
- in general, DTs represent a disjunction of conjunctions of constraints on the attribute values of instances

Disjunction of Conjunctions

$$(\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal})$$

$$\vee \quad (\text{Outlook} = \text{Overcast})$$

$$\vee \quad (\text{Outlook} = \text{Rain} \wedge \text{Wind} = \text{Weak})$$



Source: Mitchell - Machine Learning

Training a Decision Tree for Classification

- we could simply construct a tree with one path to a leaf for each example
  - we test all attributes along the path and attach the classification of the example to the leaf
  - this tree will classify all given examples correctly — it just memorizes the observations and does not generalize
- instead we want to find the smallest decision tree that is consistent with the training set (finding the smallest tree is computationally intractable)
- Goal: learn decision trees that are small and generalize well

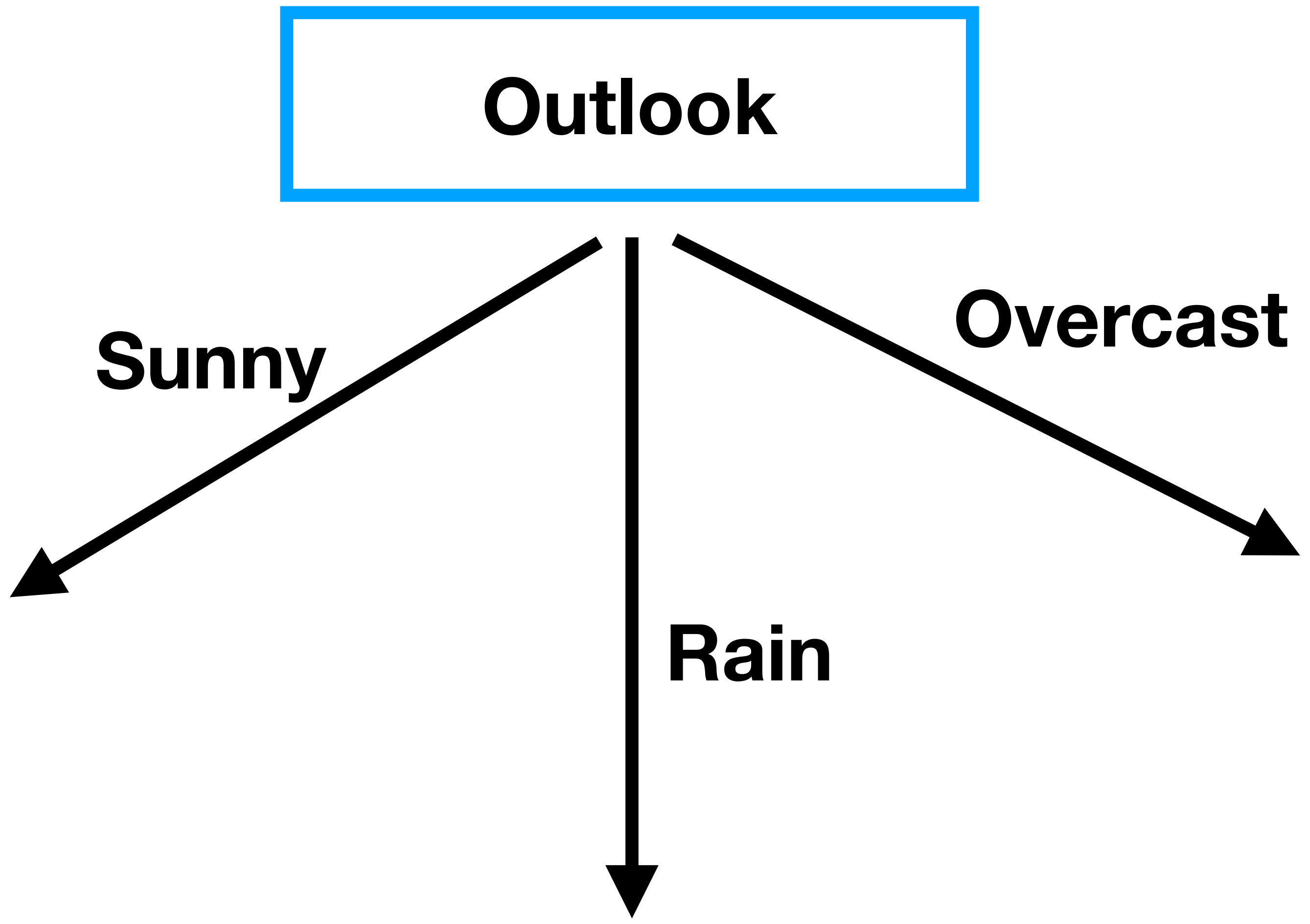
Training a Decision Tree for Classification

- we start at the root of the tree and split the data on the feature that results in the largest **Information Gain (IG)**
- we repeat this process at each child node until the leaves are **pure** or $IG \leq 0$
    - **pure:** samples at each node belong to the same class
- in practice this procedure can result in a very deep tree which might lead to overfitting
    - typically, we prune the tree by setting a limit for the maximal depth of the tree

HOCHSCHULE LANDSHUT

Top-Down Induction of Decision Trees

- choose the best attribute, split the learning sample accordingly and proceed recursively until each object is correctly classified



Outlook: Sunny

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes

Outlook: Overcast

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
2	Overcast	Hot	High	Weak	Yes
6	Overcast	Cool	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes

Outlook: Rain

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
9	Rain	Mild	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

Decision Tree Pruning

- **Pre-Training**:
  - set a depth cut-off a priori (maximal depth of the tree)
  - set a minimum number of data points for each node
  - …
- **Post-Training**:
  - grow full tree first, then remove nodes
  - remove nodes using a validation set
  - …

HOCHSCHULE LANDSHUT

Pre-Training

- stop splitting a node if either:
  - the local sample size is below some threshold N_{min}
  - the local sample information value is below some threshold I_{min}
  - the information gain of the best test is not large enough (statistical hypothesis test at some level α)

Post-Training

- split the learning sample into two sets, a growing sample G and a validation sample V
  - compute a sequence of trees $\{T_1, T_2, \ldots\}$ where:
    - T_1 is the complete tree
    - T_i is obtained by removing some test nodes from T_{i-1}
  - select the tree T_{i*} from the sequence the minimizes the validation error

Maximizing Information Gain

Which objective function do we want to optimize with our tree learning algorithm?

$$S = \{(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_n, y_n)\}, y_i \in \{1, \ldots, c\}$$

our labeled data with c classes

$$S_k = \{(\mathbf{x}, y) \in S \,|\, y = k\}$$

all inputs with labels k

$$p_k = \frac{|S_k|}{|S|}$$

fraction of inputs in S with labels k

$$I_H(S) = -\sum_{k=1}^{c} p_k \log_2(p_k)$$

Entropy

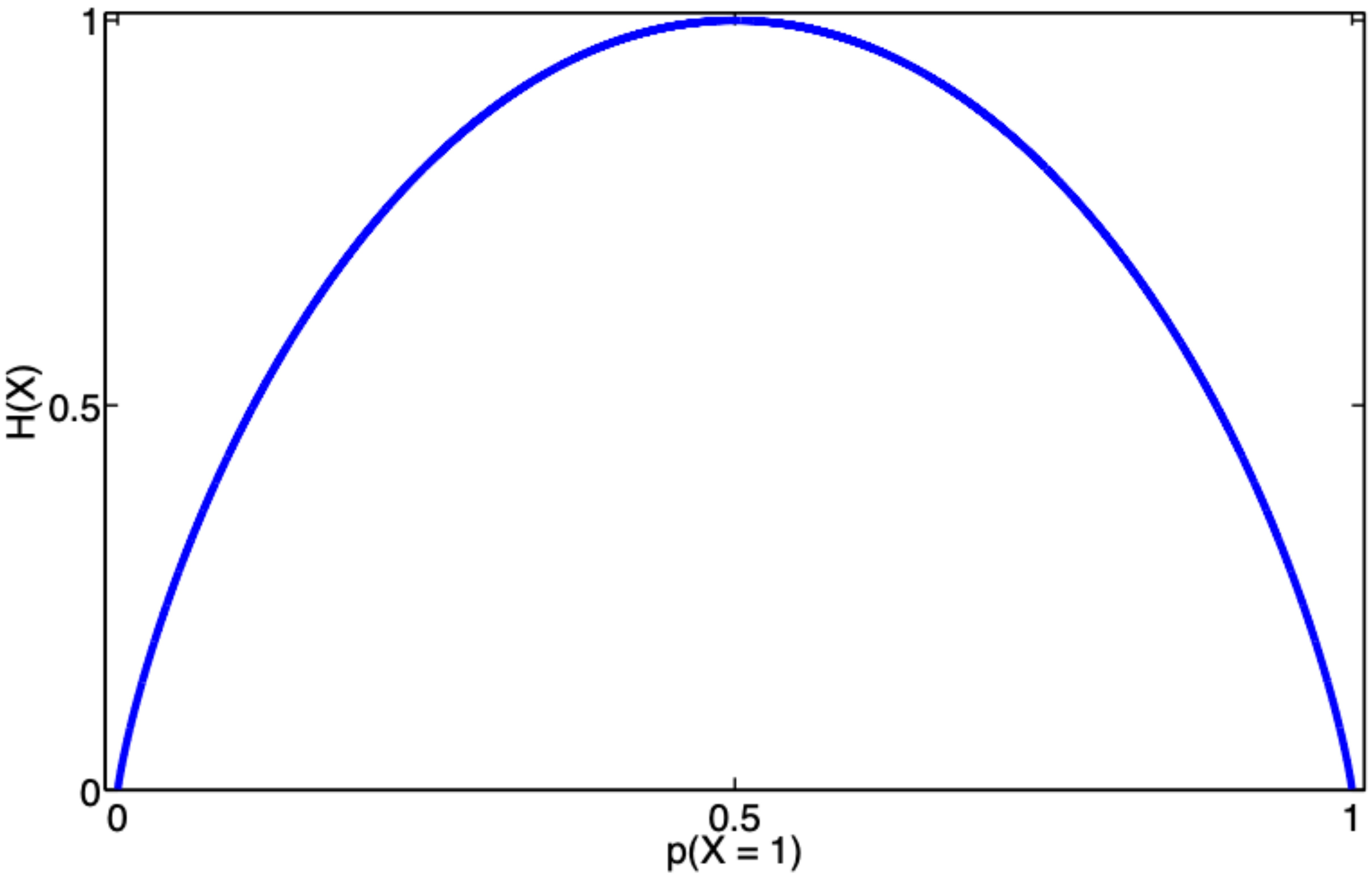
$$I_G(S) = \sum_{k=1}^{c} p_k(1 - p_k)$$

Gini Impurity

$$I_E(S) = 1 - \max_k(p_k)$$

Classification Error

Entropy for Two-Class Classification



Maximizing Information Gain

Which objective function do we want to optimize with our tree learning algorithm?

$$IG(S_{parent}, A) = I(S_{parent}) - \sum_{v \in Values(A)} \frac{|S_v|}{|S_{parent}|} I(S_v)$$

the general case: multiple splits are allowed

$$\boxed{IG(S_{parent}) = I(S_{parent}) - \frac{|S_{left}|}{|S_{parent}|} I(S_{left}) - \frac{|S_{right}|}{|S_{parent}|} I(S_{right})}$$

for binary decision trees

The ID3 Training Algorithm

ID3 (Iterative Dichotomiser 3; Ross Quinlan 1986)

- cannot handle numeric features
- no pruning
- uses Entropy minimization

ID3 - Example

	Outlook	Temperature	Humidity	Wind	Play Tennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

Source: Mitchell - Machine Learning

- Which attribute should be first tested in the tree?
- ID3 determines the information gain for each candidate in the tree (Outlook, Temperature, Humidity, Wind), then selects one with highest information gain

$$Values(Wind) = \{Weak, Strong\}$$

$$S = [9+, 5-]$$

$$S_{Strong} \leftarrow [3+, 3-]$$

$$S_{Weak} \leftarrow [6+, 2-]$$

ID3 - Example

	Outlook	Temperature	Humidity	Wind	Play Tennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

Source: Mitchell - Machine Learning

$$Values(Wind) = \{Weak, Strong\}$$

$$S = [9+, 5-] \qquad S_{Strong} \leftarrow [3+, 3-]$$

$$S_{Weak} \leftarrow [6+, 2-]$$

$$\text{Entropy}(S_{Strong}) = -\frac{3}{6}\log_2(3/6) - \frac{3}{6}\log_2(3/6) = 1$$

$$\text{Entropy}(S_{Weak}) = -\frac{6}{8}\log_2(6/8) - \frac{2}{8}\log_2(2/8) \approx 0.811$$

ID3 - Example

	Outlook	Temperature	Humidity	Wind	Play Tennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

Source: Mitchell - Machine Learning

$$IG(S, Wind) = I(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} I(S_v)$$

$$= Entropy(S) - \frac{8}{14} Entropy(S_{Weak}) - \frac{6}{14} Entropy(S_{Strong})$$

$$= 0.940 - \frac{8}{14} 0.811 - \frac{6}{14} 1.00$$

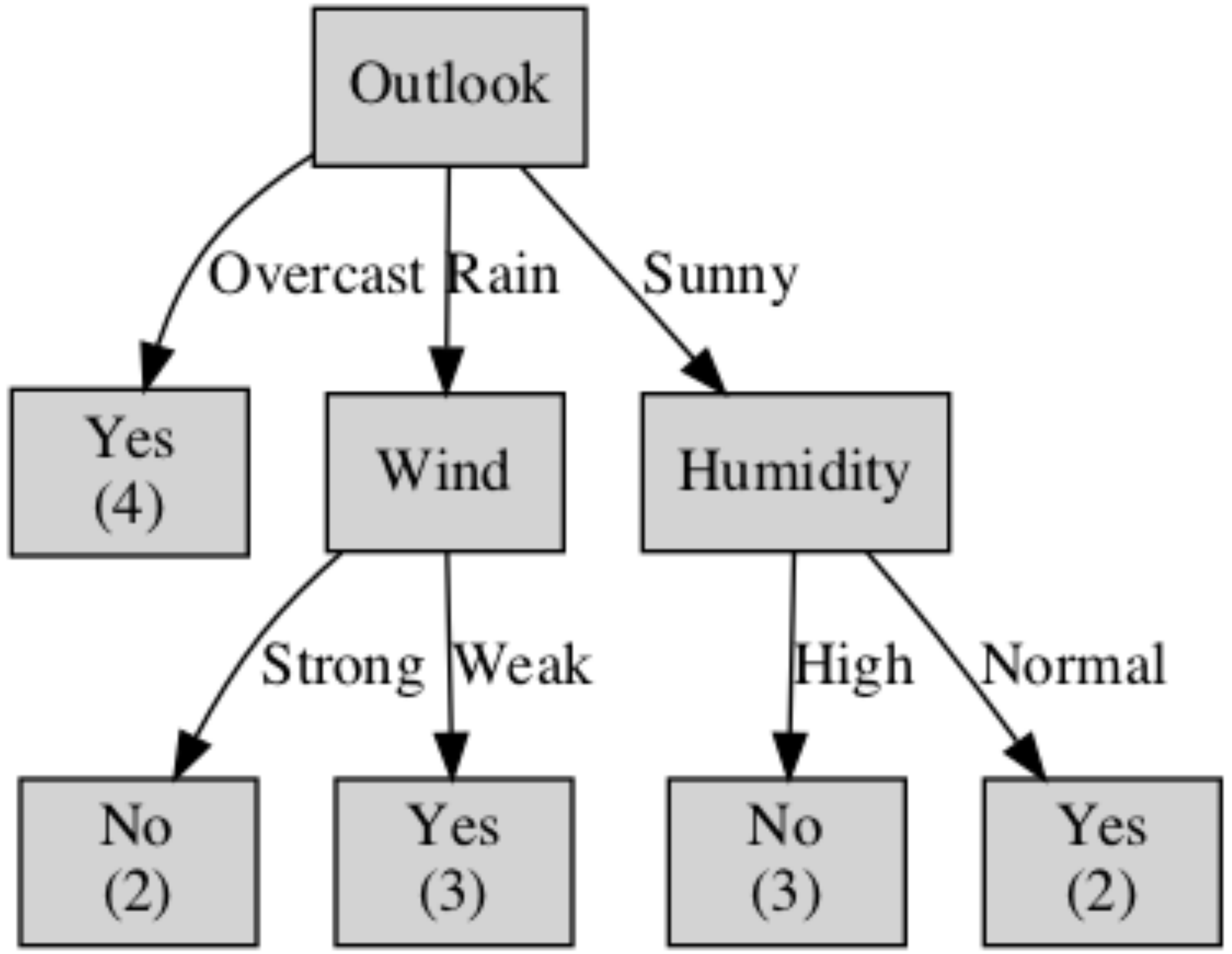
$$\boxed{IG(S, Outlook) = 0.246}$$

$$IG(S, Humidity) = 0.151$$

$$IG(S, Temperature) = 0.029$$

ID3 on Tennis Dataset

	Outlook	Temperature	Humidity	Wind	Play Tennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No




```
Outlook Overcast: Yes (4)
Outlook Rain
|   Wind Strong: No (2)
|   Wind Weak: Yes (3)
Outlook Sunny
|   Humidity High: No (3)
|   Humidity Normal: Yes (2)
```

Missing Values?

- not all attribute values are known for every object during learning or testing

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
X	**Sunny**	**Hot**	**High**	**?**	**No**

- there are three strategies:
  - use most common value in the learning sample
  - use most common value in the tree
  - assign a probability to each possible value

Information Theory — Measuring Information with Entropy

- information on a computer is represented by binary strings

- imagine that you want to encode a character from the Latin alphabet (including a few other characters like punctuation marks)

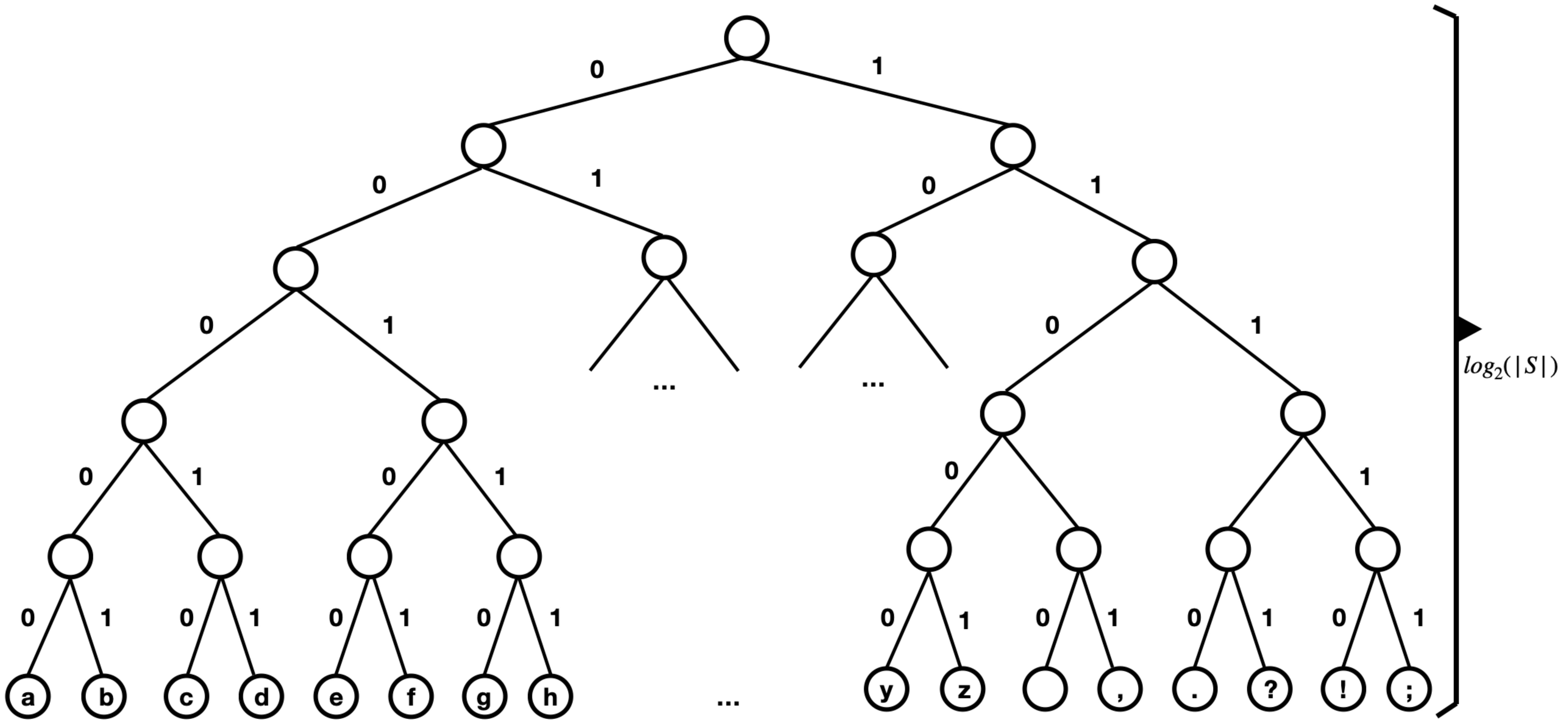
$$S = \{a, b, c, \ldots, z, \text{space}, \text{','}, \text{'.'}, \text{'?'}, \text{'!'}, \text{';'}\}, \quad |S| = 32$$

- with sequences of $log_2(32) = 5$ bits you could encode all 32 characters as $a = 00000$, $b = 00001$, $c = 00010$, …

- Are 5 bits enough?

HOCHSCHULE LANDSHUT

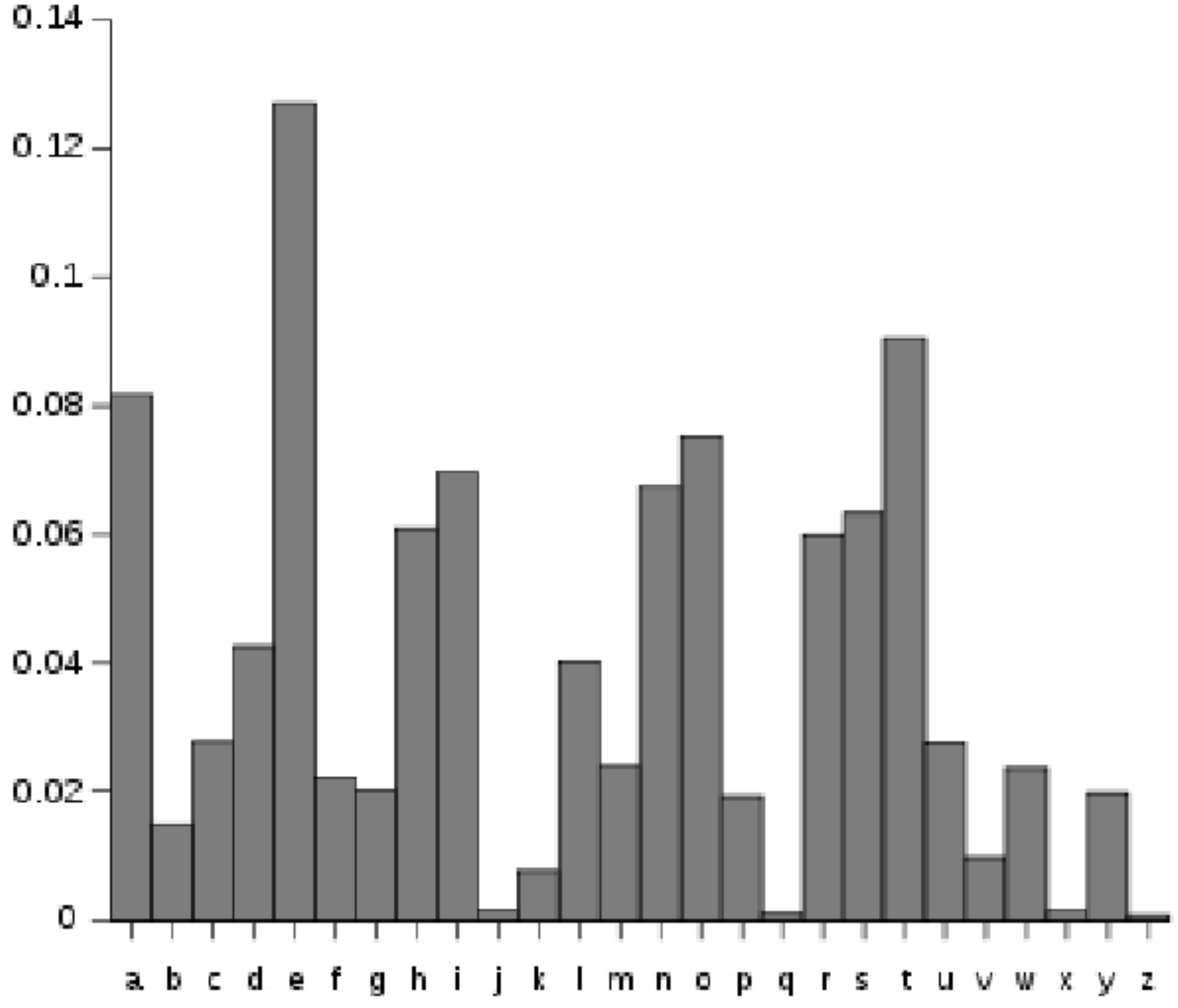
Information Theory — Measuring Information with Entropy



$|S| = 32$ **characters**

$$\log_2(|S|) = \log_2(1/p_i) = -\log_2(p_i), \quad p_i = \frac{1}{|S|}$$

Information Theory — Measuring Information with Entropy



Relative Frequencies of Letters in the English Language (Wikipedia)

- if we are interested in the "expected" number of bits necessary to display a message, we should take into account the relative frequencies of letters; p_i for letter with index i
- thus, for $|S|$ characters we compute

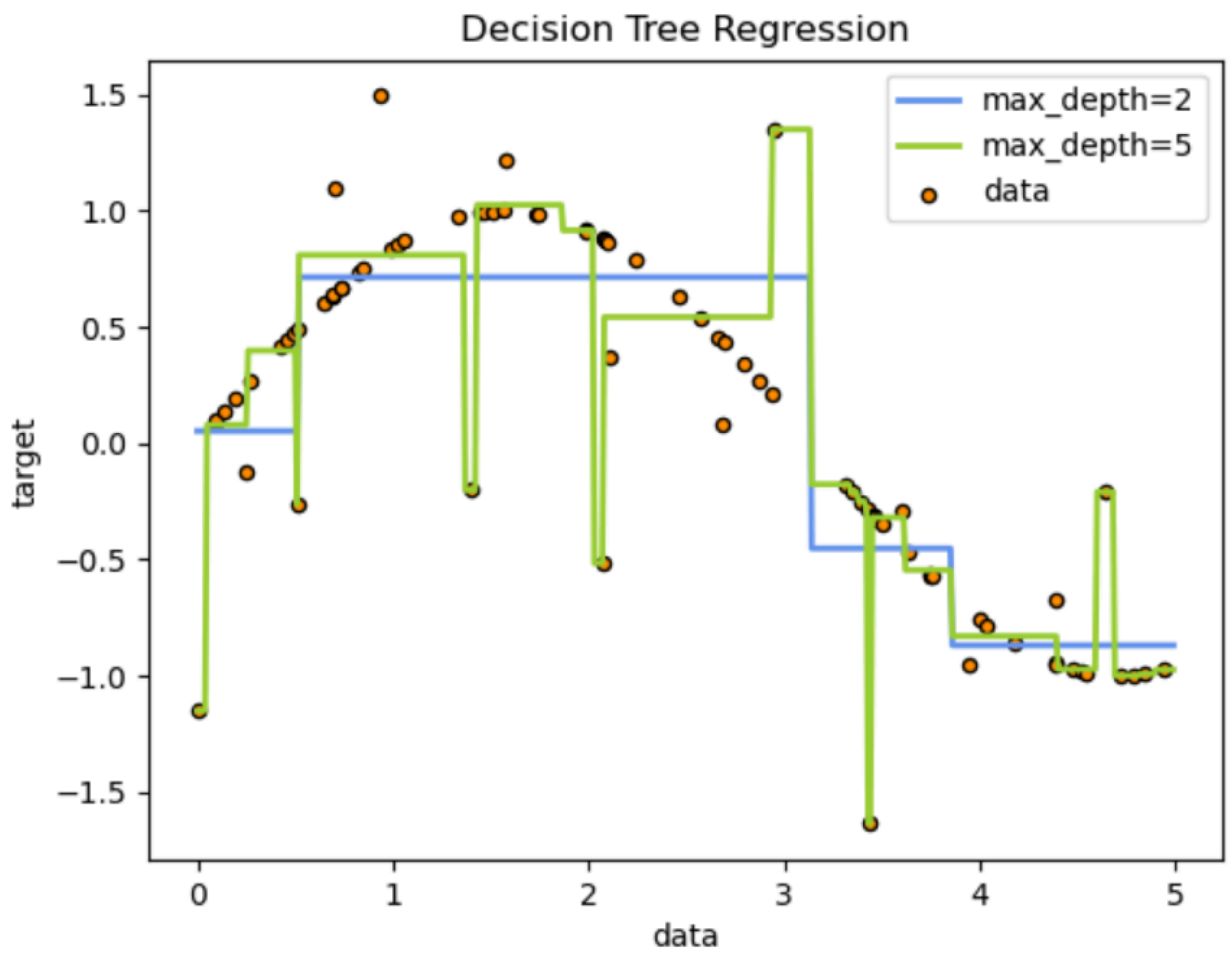
$$-\sum_{i=1}^{|S|} p_i \log_2(p_i)$$

Decision Trees for Regression

- labels are continuous $y_i \in \mathbb{R}$
- Impurity:

$$I(S) = MSE(S) = \frac{1}{|S|} \sum_{(x,y) \in S} (y - \bar{y}_S)^2$$

$$\bar{y}_S = \frac{1}{|S|} \sum_{(x,y) \in S} y$$



https://scikit-learn.org/stable/modules/tree.html#tree

Computational Complexity

Prediction complexity

- making predictions requires traversing the Decision Tree from root to leaf
- traversing the tree requires going through roughly $O(log_2(|S|))$ nodes
- since each node requires checking the value of one feature only, the overall prediction complexity is $O(log_2(|S|))$

Training complexity

- the training algorithm compares all features on all samples at each node
- training complexity: $O(m \cdot |S| log_2(|S|))$

Decision Trees for Regression and Classification

- *Decision Trees (DTs)* are powerful algorithms capable of performing regression and classification tasks, and even multi-output tasks

- they are attractive models if we care about interpretability

- *DTs* can handle categorical features as well as features represented by real numbers

- *DTs* are very fast during test time, as the inputs just need to traverse down the tree to the leaf

- *DTs* require no metric because splits are based on feature thresholds and not distances

Advantages / Disadvantages

+ model interpretability (white-box model)

+ require little data preparation

+ low prediction complexity

+ training data may contain errors (decision tree learning algorithms are robust to errors; errors in classification and errors in the attribute values that describe these examples)

+ training data may contain missing attribute values

- easy to overfit (do not generalize well)

- unstable (small variations in data might result in completely different tree)

- problems with unbalanced datasets

- sophisticated pruning required

Broadening the Applicability of Decision Trees

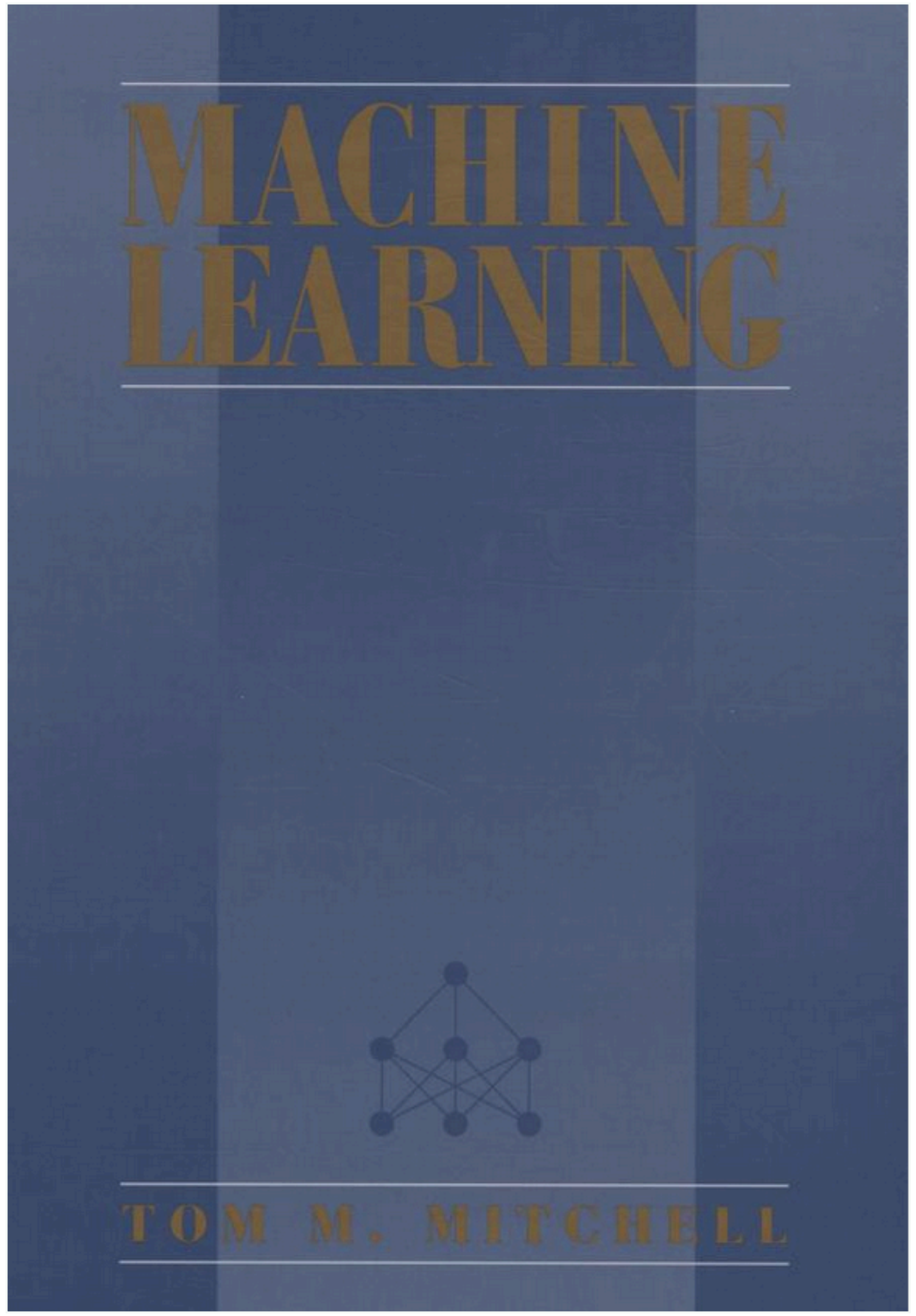
- decision trees can be made more widely useful by handling the following complications
    - missing data
    - continuous and multivalued input attributes
    - continuous-valued output attribute

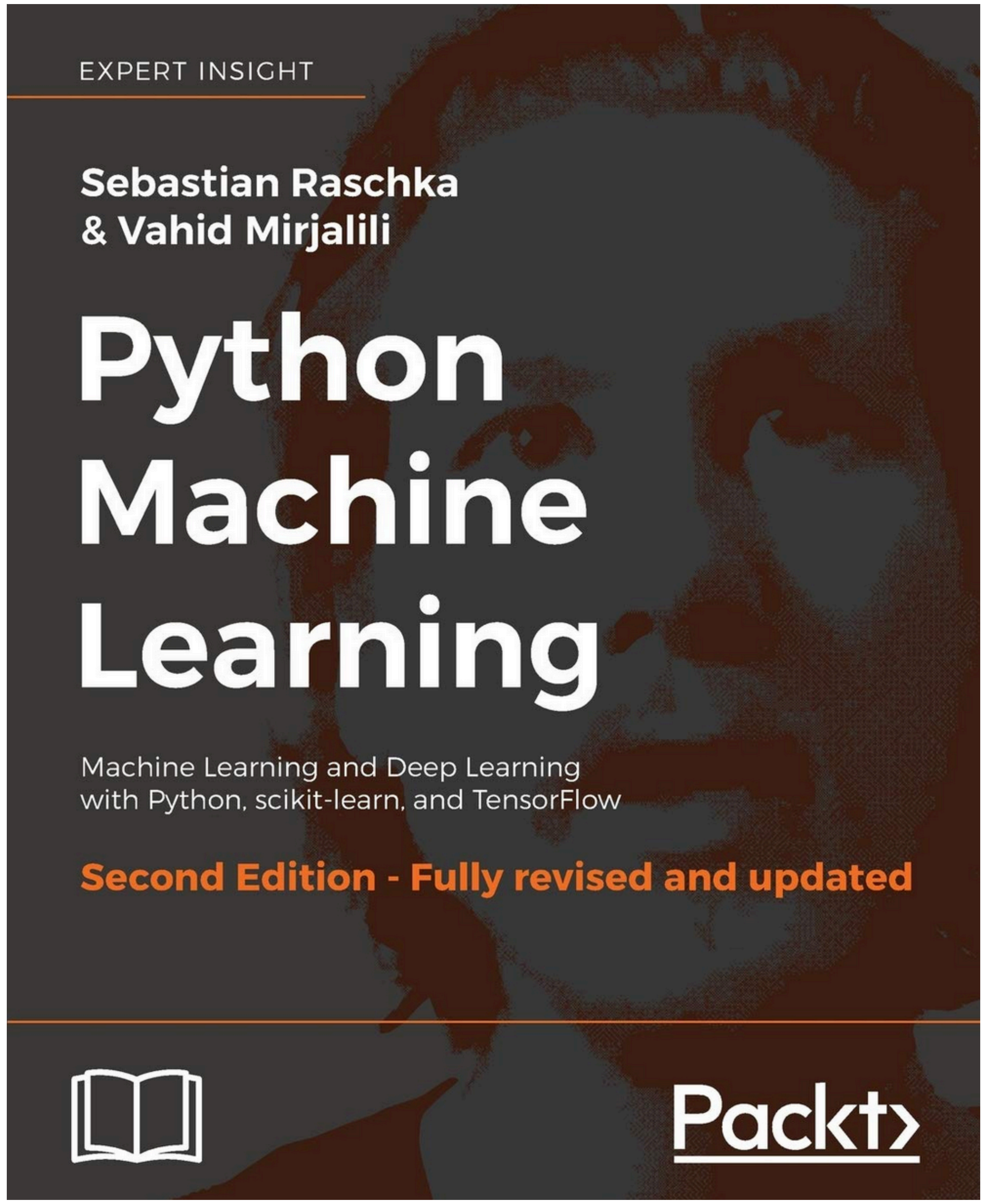
Decision Tree Learning

function LEARN-DECISION-TREE(*examples*, *attributes*, *parent_examples*) **returns** a tree

if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
else if all *examples* have the same classification **then return** the classification
else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
else
    $A \leftarrow \mathrm{argmax}_{a \in attributes}$ IMPORTANCE(a, *examples*)
    tree ← a new decision tree with root test A
    for each value v of A **do**
        $exs \leftarrow \{e : e \in examples$ **and** $e.A = v\}$
        subtree ← LEARN-DECISION-TREE(*exs*, *attributes* − A, *examples*)
        add a branch to *tree* with label ($A = v$) and subtree *subtree*
    return *tree*

Literature





- *L. Breiman et al.: Classification and regression trees*
- *J.R. Quinlan, C4.5: programs for machine learning*
- *Hastie et al., The Elements of Statistical Learning: Data Mining, Inference, and Prediction; Chapter 9*