Lecture 06

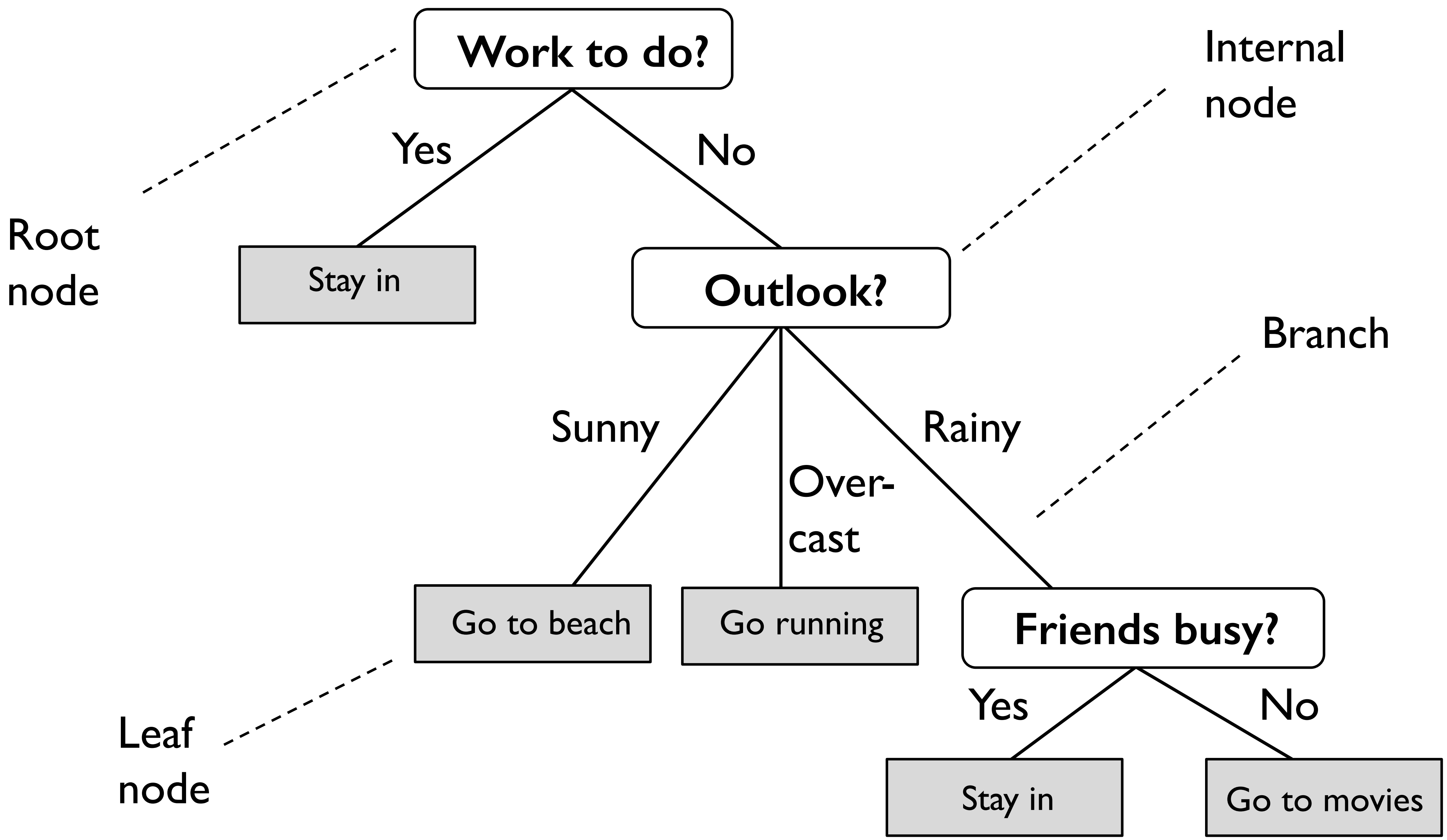
Decision Trees

STAT 479: Machine Learning, Fall 2018

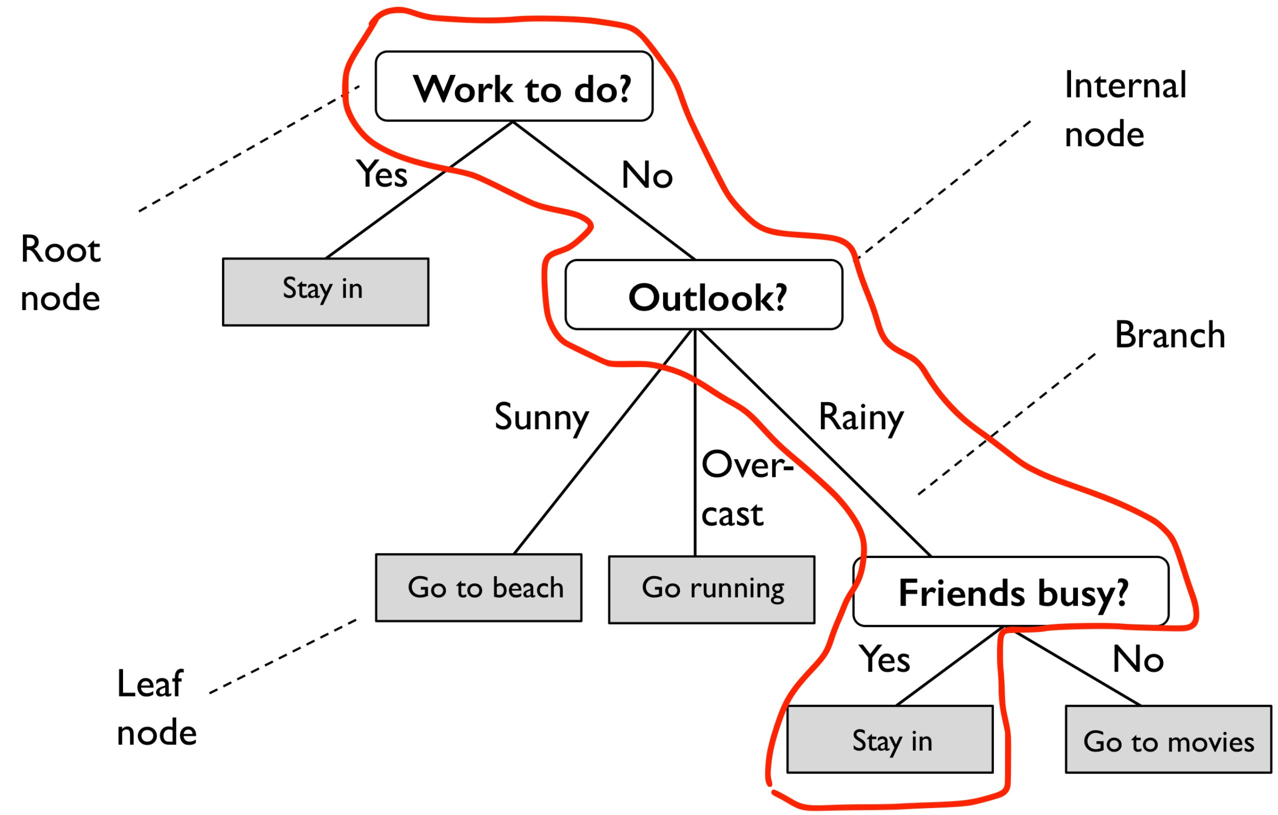
Sebastian Raschka

http://stat.wisc.edu/~sraschka/teaching/stat479-fs2018/

Decision Tree Terminology



Decision Trees as Rulesets



IF ______________________________ ___ __________________________

___ __________________________

THEN _________ ___ _____________

Decision Trees and ML Categories

- Supervised vs. unsupervised learning algorithm
- classification vs. regression
- Optimization method: ______
- Eager vs. lazy learning algorithm
- Batch vs. online learning algorithm
- Parametric vs. nonparametric model
- Deterministic vs. stochastic

Recursion / Recursive Algorithms

```
def some_func(x):
    if x == []:
        return 0
    else:
        return 1 + some_func(x[1:])
```

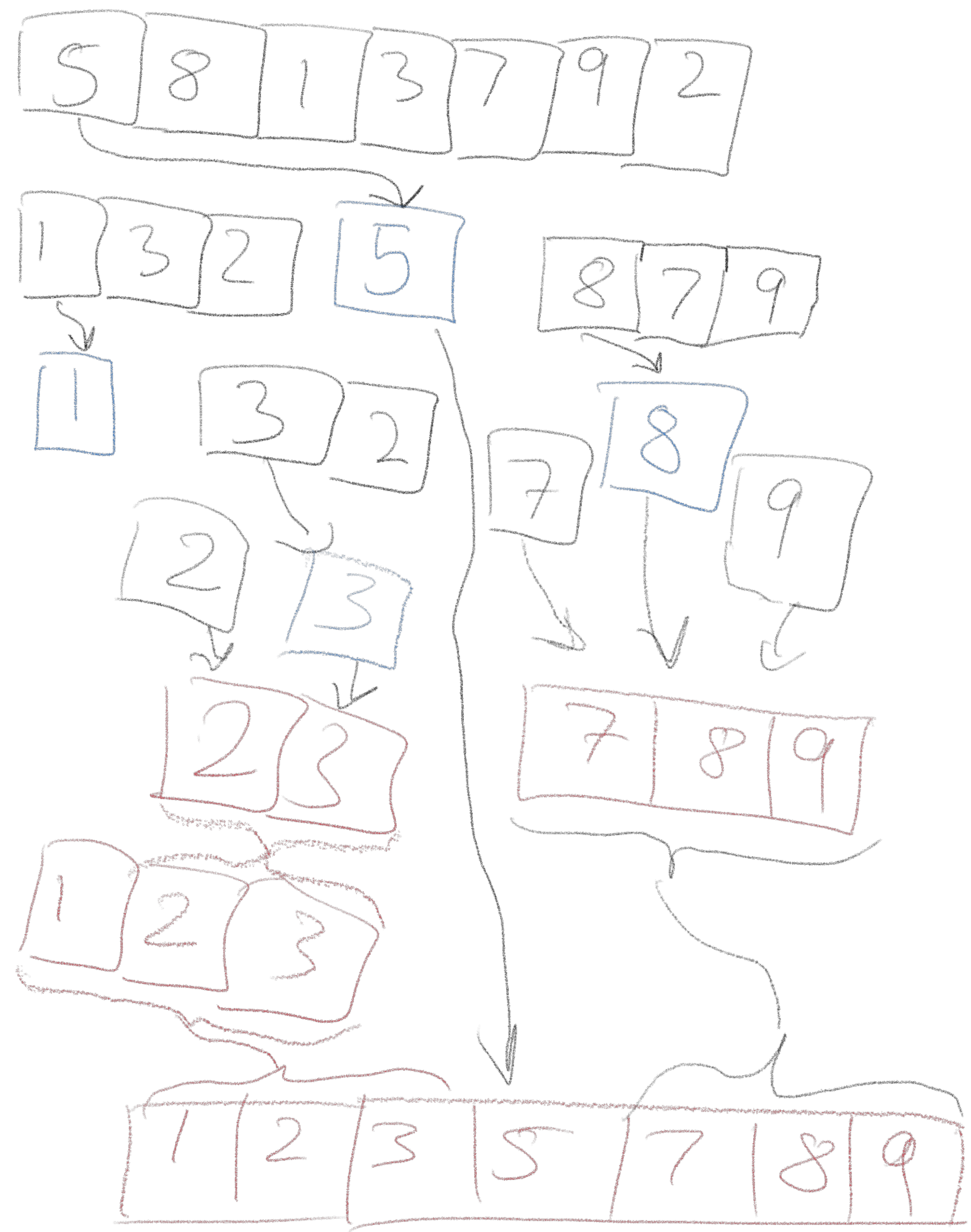
What does this function do?

Divide & Conquer Algorithms

```python
def quicksort(array):
    if len(array) < 2:
        return array
    else:
        pivot = array[0]
        smaller, bigger = [], []
        for ele in array[1:]:
            if ele <= pivot:
                smaller.append(ele)
            else:
                bigger.append(ele)
        return quicksort(smaller) + [pivot] + quicksort(bigger)
```

Divide & Conquer Algorithms

```python
def quicksort(array):
    if len(array) < 2:
        return array
    else:
        pivot = array[0]
        smaller, bigger = [], []
        for ele in array[1:]:
            if ele <= pivot:
                smaller.append(ele)
            else:
                bigger.append(ele)
        return quicksort(smaller) + [pivot] + quicksort(bigger)
```



Time complexity of quicksort:

______________ ("on average")

```
def quicksort(array):
    if len(array) < 2:
        return array
    else:
        pivot = array[0]
        smaller, bigger = [], []
        for ele in array[1:]:
            if ele <= pivot:
                smaller.append(ele)
            else:
                bigger.append(ele)
        return quicksort(smaller) + [pivot] + quicksort(bigger)
```

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2) *	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

http://www.bigocheatsheet.com

* "worst" worst (inversely-sorted array)

Time Complexity ("Big-O")

Growing the tree:

Tip: It can be shown that optimal split is on boundary between adjacent examples (similar feature value) with different class labels.

Fayyad, Usama Mohammad. "On the induction of decision trees for multiple concept learning." (1992).

Time Complexity ("Big-O")

Querying the tree:

GenerateTree($\mathcal{D}$):

- if $y = 1 \; \forall \; \langle \mathbf{x}, \mathbf{y} \rangle \in \mathcal{D}$ or $y = 0 \; \forall \; \langle \mathbf{x}, y \rangle \in \mathcal{D}$:
    - return Tree
- else:
    - Pick best feature x_j:
        - $\mathcal{D}_0$ at Child$_0$: $x_j = 0 \; \forall \; \langle \mathbf{x}, y \rangle \in \mathcal{D}$
        - $\mathcal{D}_1$ at Child$_1$: $x_j = 1 \; \forall \; \langle \mathbf{x}, y \rangle \in \mathcal{D}$

    return Node(x_j, GenerateTree($\mathcal{D}_0$), GenerateTree($\mathcal{D}_1$))

Generic Tree Growing Algorithm

1) Pick the feature that, when parent node is split, results in the largest information gain

2) Stop if child nodes are pure or information gain <= 0

3) Go back to step 1 for each of the two child nodes

Generic Tree Growing Algorithm

1) Pick the feature that, when parent node is split, results in the largest information gain

2) Stop if child nodes are pure or information gain <= 0

3) Go back to step 1 for each of the two child nodes

- How make predictions of features in dataset not sufficient to make child nodes pure?

Design choices

- How to split
  - what measurement/criterion as measure of goodness
  - binary vs multi-category split
- When to stop
  - if leaf nodes contain only examples of the same class
  - feature values are all the same for all examples
  - statistical significance test

ID3 -- Iterative Dichotomizer 3

- one of the earlier/earliest decision tree algorithms
- Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106.
- cannot handle numeric features
- no pruning, prone to overfitting
- short and wide trees (compared to CART)
- maximizing information gain/minimizing entropy
- discrete features, binary and multi-category features

C4.5

- continuous and discrete features
- Ross Quinlan 1993, Quinlan, J. R. (1993). C4. 5: Programming for machine learning. *Morgan Kauffmann*, *38*, 48.
- continuous is very expensive, because must consider all possible ranges
- handles missing attributes (ignores them in gain compute)
- post-pruning (bottom-up pruning)
- Gain Ratio

CART

- Breiman, L. (1984). *Classification and regression trees*. Belmont, Calif: Wadsworth International Group.
- continuous and discrete features
- strictly binary splits (taller trees than ID3, C4.5)
- binary splits can generate better trees than C4.5, but tend to be larger and harder to interpret; k-attributes has a ways to create a binary partitioning
- variance reduction in regression trees
- Gini impurity, twoing criteria in classification trees
- cost complexity pruning

Others

- CHAID (CHi-squared Automatic Interaction Detector); Kass, G. V. (1980). "An exploratory technique for investigating large quantities of categorical data". *Applied Statistics*. 29 (2): 119–127.

- MARS (Multivariate adaptive regression splines); Friedman, J. H. (1991). "Multivariate Adaptive Regression Splines". *The Annals of Statistics*. 19: 1

- C5.0 (patented)

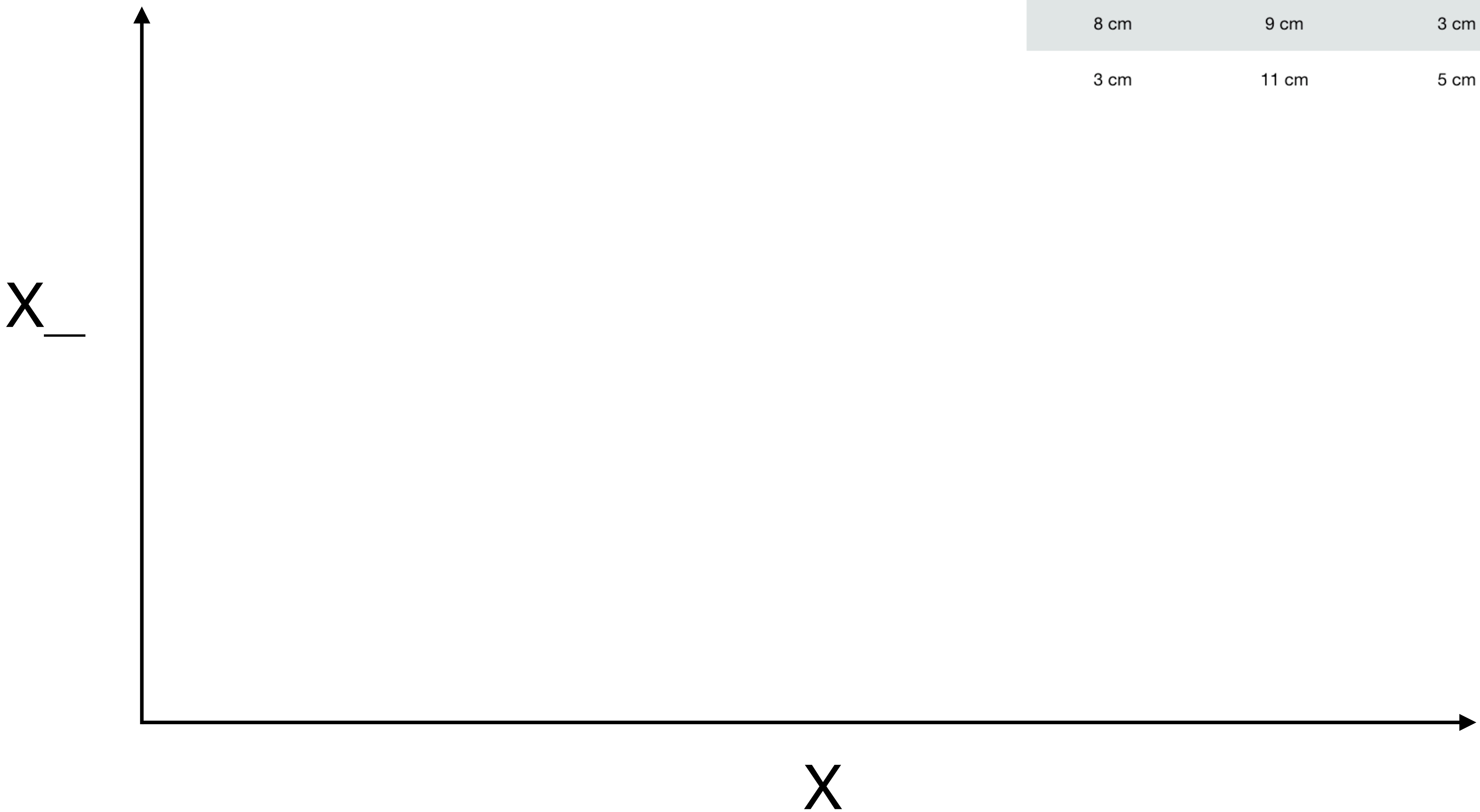
- ...

Finding a Decision Rule

x_1	x_2	x_3	y
6 cm	8 cm	9 cm	1
4 cm	11 cm	2 cm	0
6 cm	12 cm	4 cm	0
10 cm	9 cm	3 cm	1
5 cm	7 cm	8 cm	0
8 cm	9 cm	3 cm	1
3 cm	11 cm	5 cm	0

Drawing a Decision Boundary

x_1	x_2	x_3	y
6 cm	8 cm	9 cm	1
4 cm	11 cm	2 cm	0
6 cm	12 cm	4 cm	0
10 cm	9 cm	3 cm	1
5 cm	7 cm	8 cm	0
8 cm	9 cm	3 cm	1
3 cm	11 cm	5 cm	0



Information Gain

$$GAIN(\mathscr{D}, x_j) = H(\mathscr{D}) - \sum_{v \in Values(x_j)} \frac{|\mathscr{D}_v|}{|\mathscr{D}|} H(\mathscr{D}_v)$$

Shannon Entropy

Refer to lecture notes

Entropy

$$H = -\sum_{i} p(i \mid x_j)\log_2(p(i \mid x_j))$$

Gini Impurity

$$Gini = 1 - \sum_{i} \left(p(i \mid x_j)^2 \right)$$

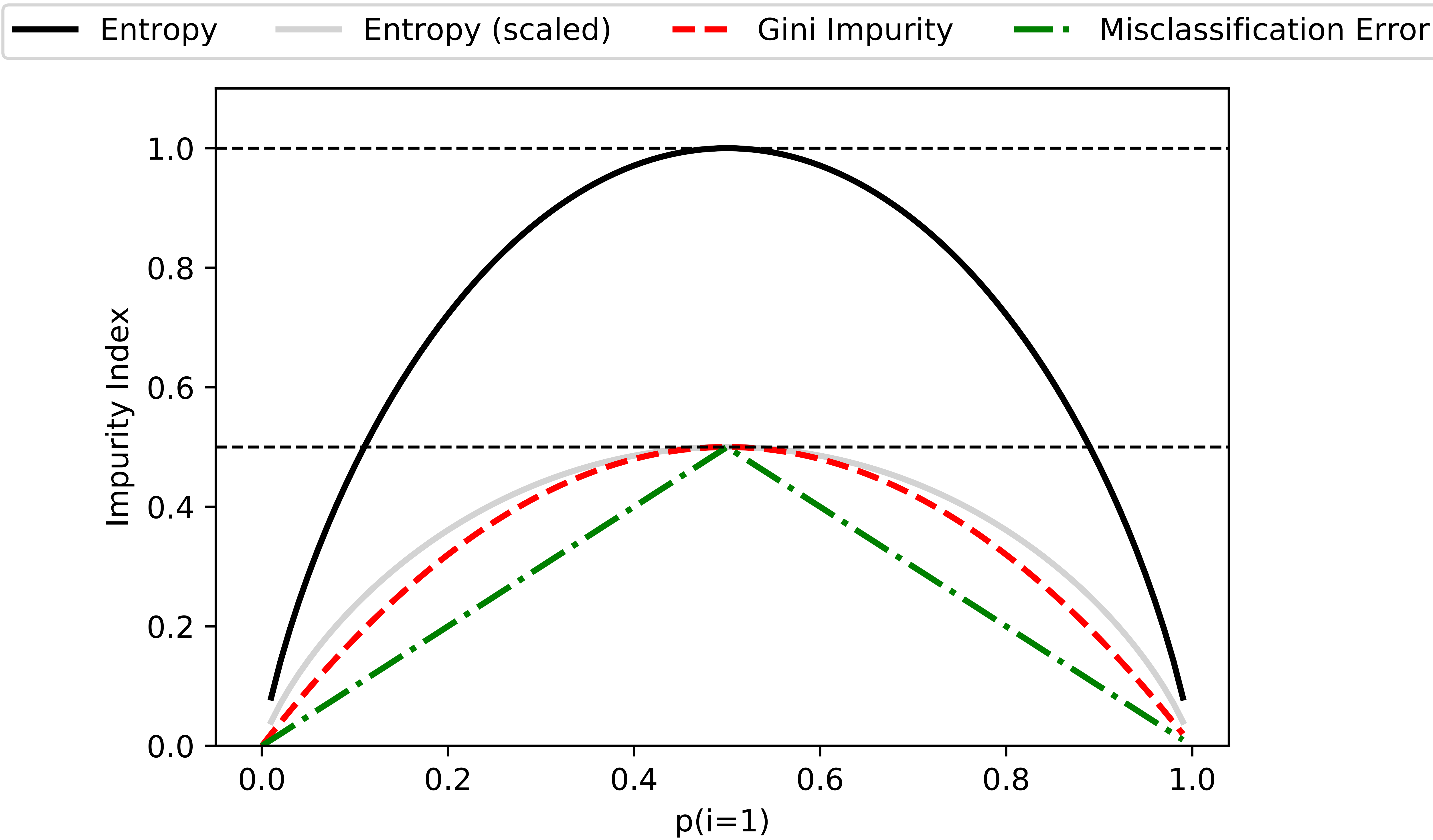
Misclassification Error

$$ERR = \frac{1}{n} \sum_{i=1}^{n} L(\hat{y}^{[i]}, y^{[i]}),$$

$$L(\hat{y}, y) = \begin{cases} 0 \textbf{ if } \hat{y} = y, \\ 1 \textbf{ otherwise.} \end{cases}$$

Misclassification Error

$$ERR = 1 - \max_i(p(i \mid x_j))$$

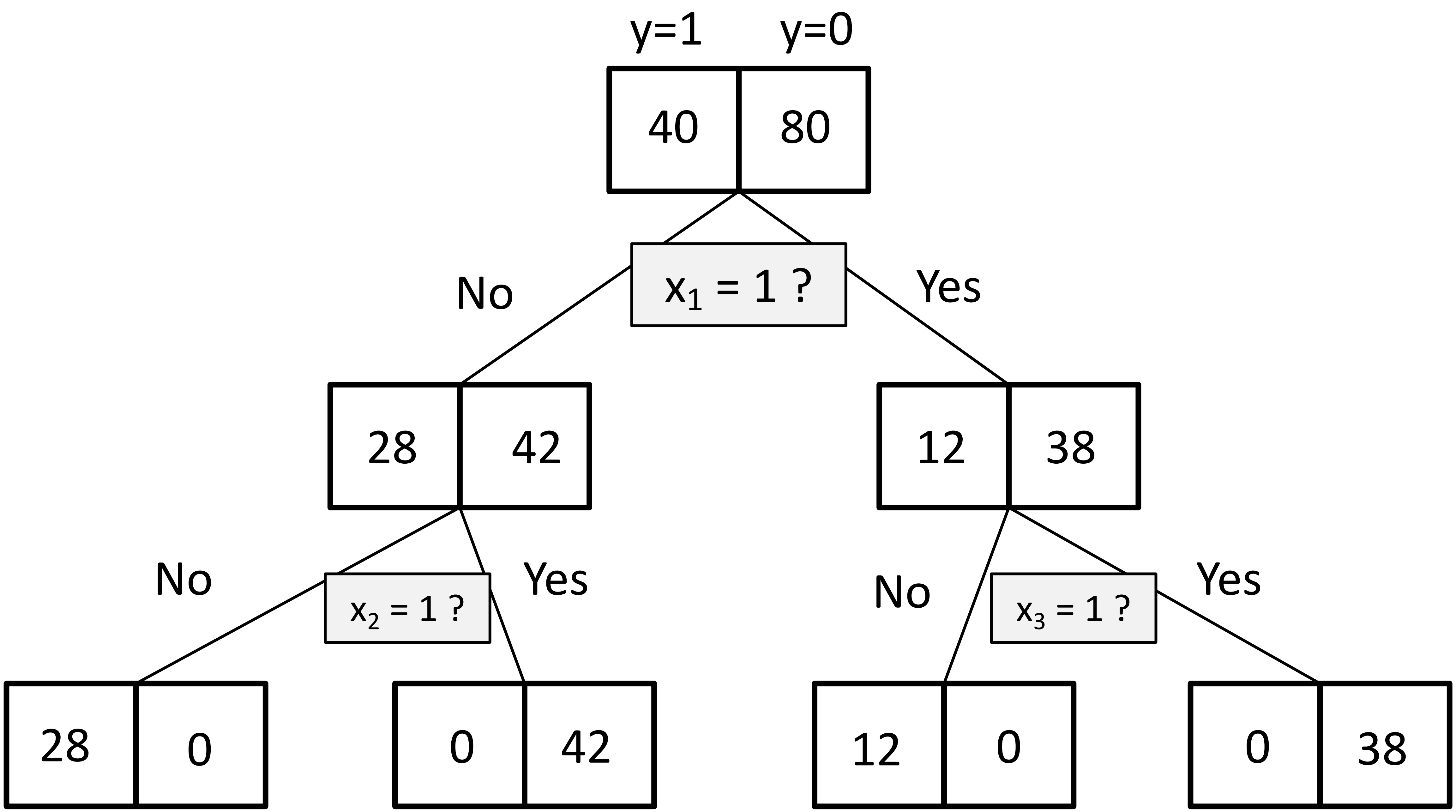


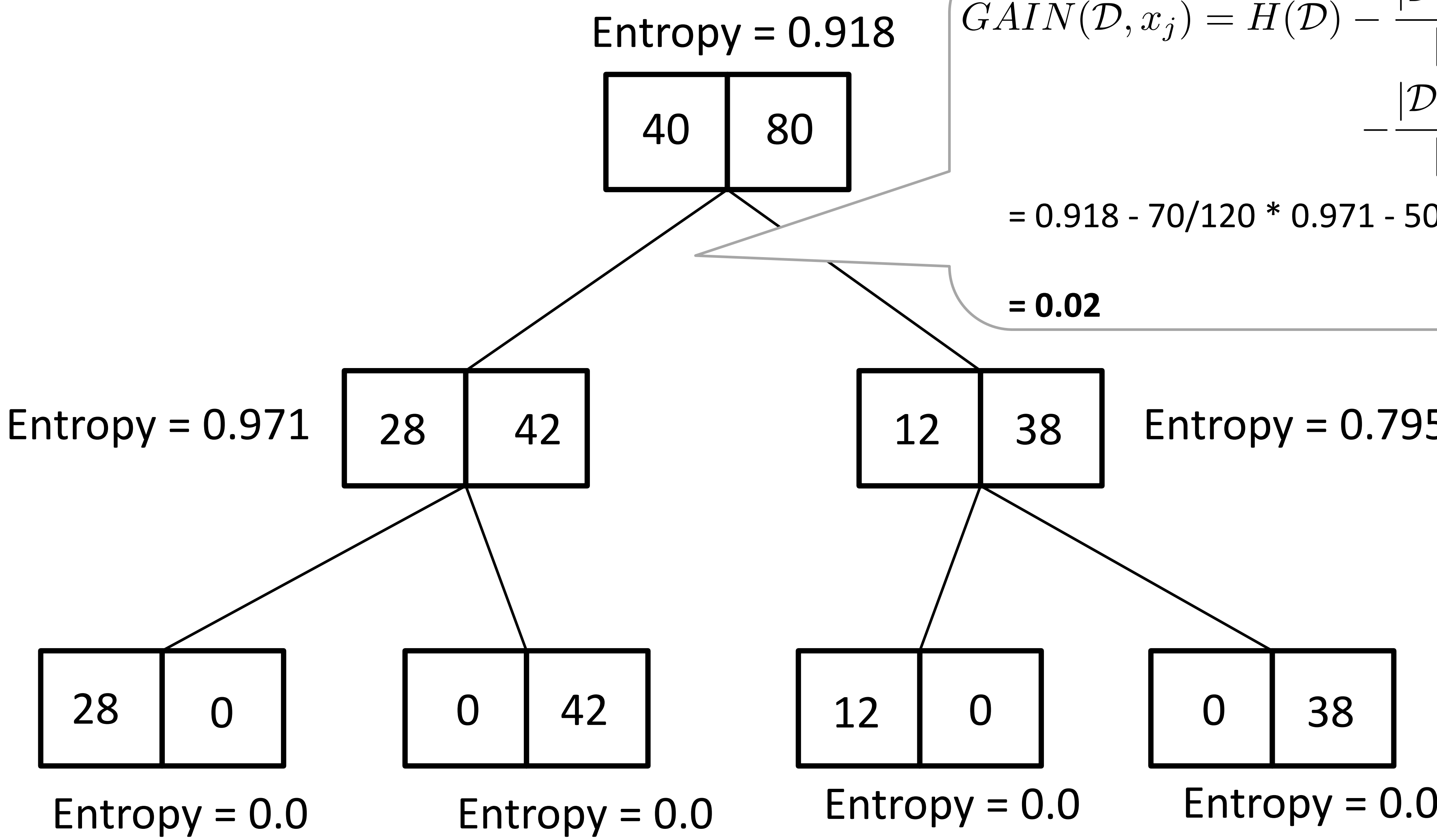
Entropy
Entropy (scaled)
Gini Impurity
Misclassification Error
Impurity Index
p(i=1)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Why Growing Decision Trees via Entropy instead of Misclassification Error?

Why Growing Decision Trees via Entropy instead of Misclassification Error?

$$GAIN(\mathscr{D}, x_j) = I(\mathscr{D}) - \sum_{v \in Values(x_j)} \frac{|\mathscr{D}_v|}{|\mathscr{D}|} I(\mathscr{D}_v)$$

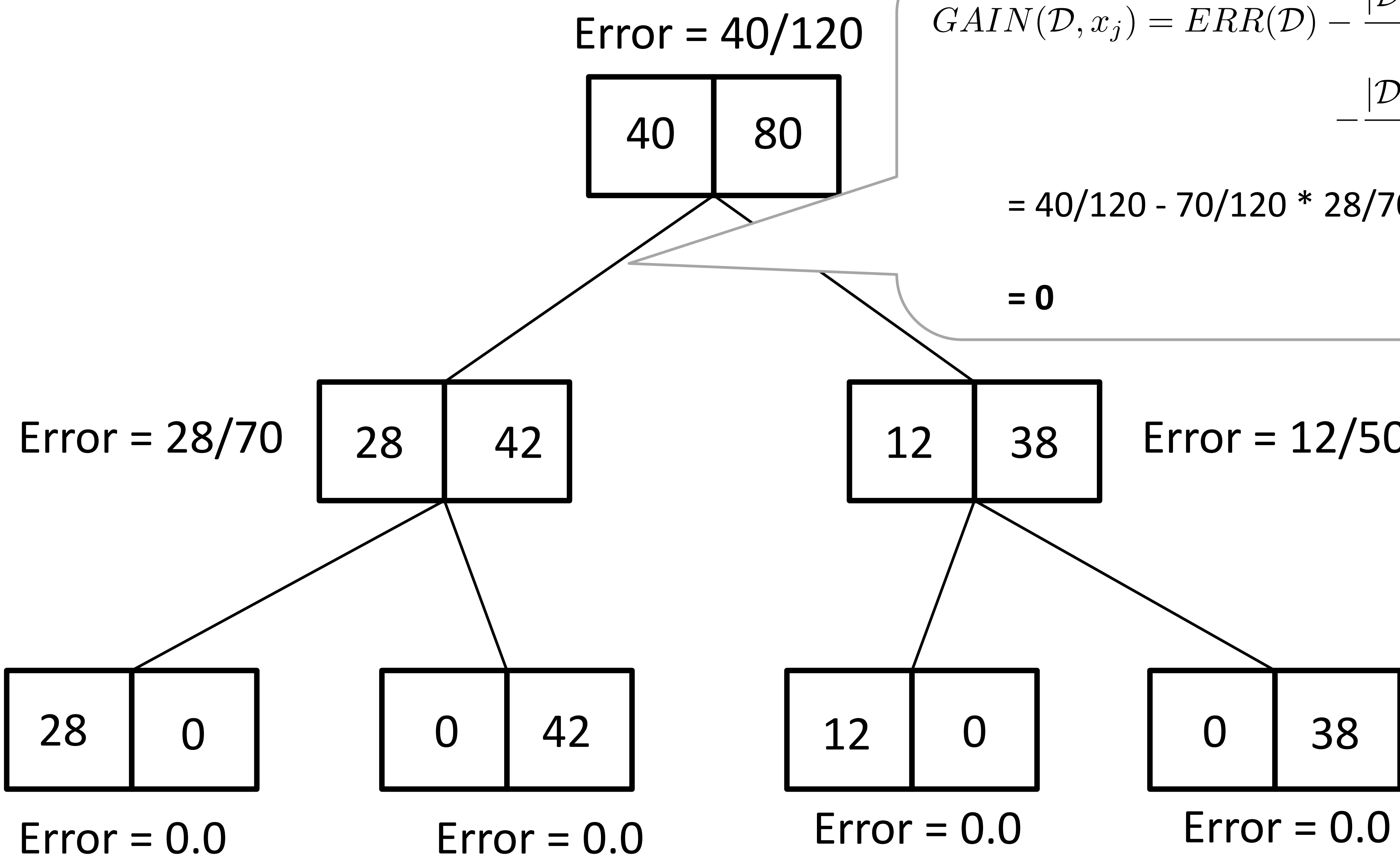

y=1
y=0
40
80
x_1 = 1 ?
No
Yes
28
42
12
38
x_2 = 1 ?
No
Yes
x_3 = 1 ?
No
Yes
28
0
0
42
12
0
0
38



$$GAIN(\mathcal{D}, x_j) = H(\mathcal{D}) - \frac{|\mathcal{D}_{x_j=1}|}{|\mathcal{D}|} H(\mathcal{D}_{x_j=1}) - \frac{|\mathcal{D}_{x_j=0}|}{|\mathcal{D}|} H(\mathcal{D}_{x_j=0})$$

= 0.918 - 70/120 * 0.971 - 50/120 * 0.795

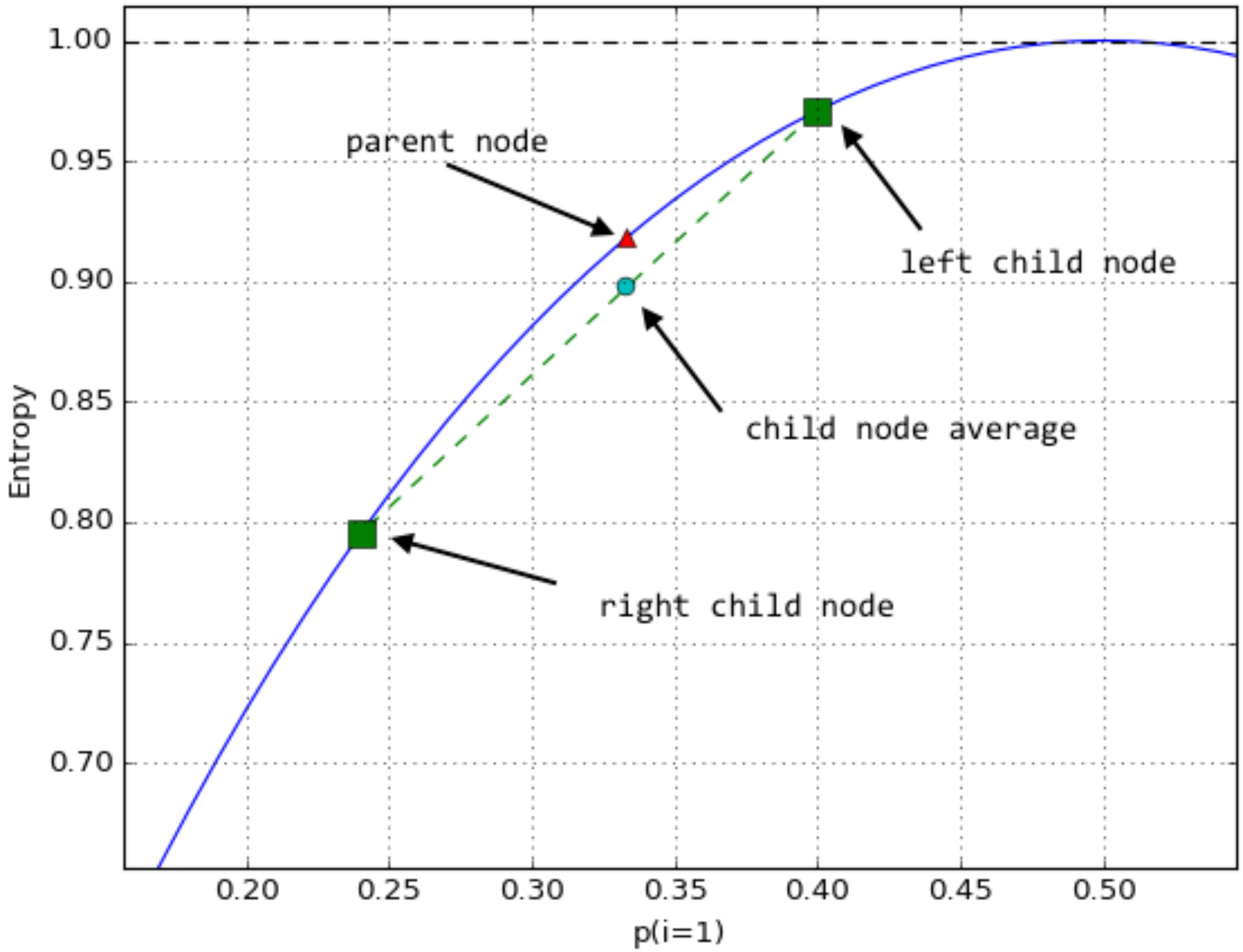
= 0.02



$$GAIN(\mathcal{D}, x_j) = ERR(\mathcal{D}) - \frac{|\mathcal{D}_{x_j=1}|}{|\mathcal{D}|} ERR(\mathcal{D}_{x_j=1}) - \frac{|\mathcal{D}_{x_j=0}|}{|\mathcal{D}|} ERR(\mathcal{D}_{x_j=0})$$

= 40/120 - 70/120 * 28/70 - 50/120 * 12/50

= 0


parent node
left child node
child node average
right child node
Entropy
p(i=1)
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.20
0.25
0.30
0.35
0.40
0.45
0.50

Gain Ratio

$$GainRatio(\mathscr{D}, x_j) = \frac{Gain(\mathscr{D}, x_j)}{SplitInfo(\mathscr{D}, x_j)}$$

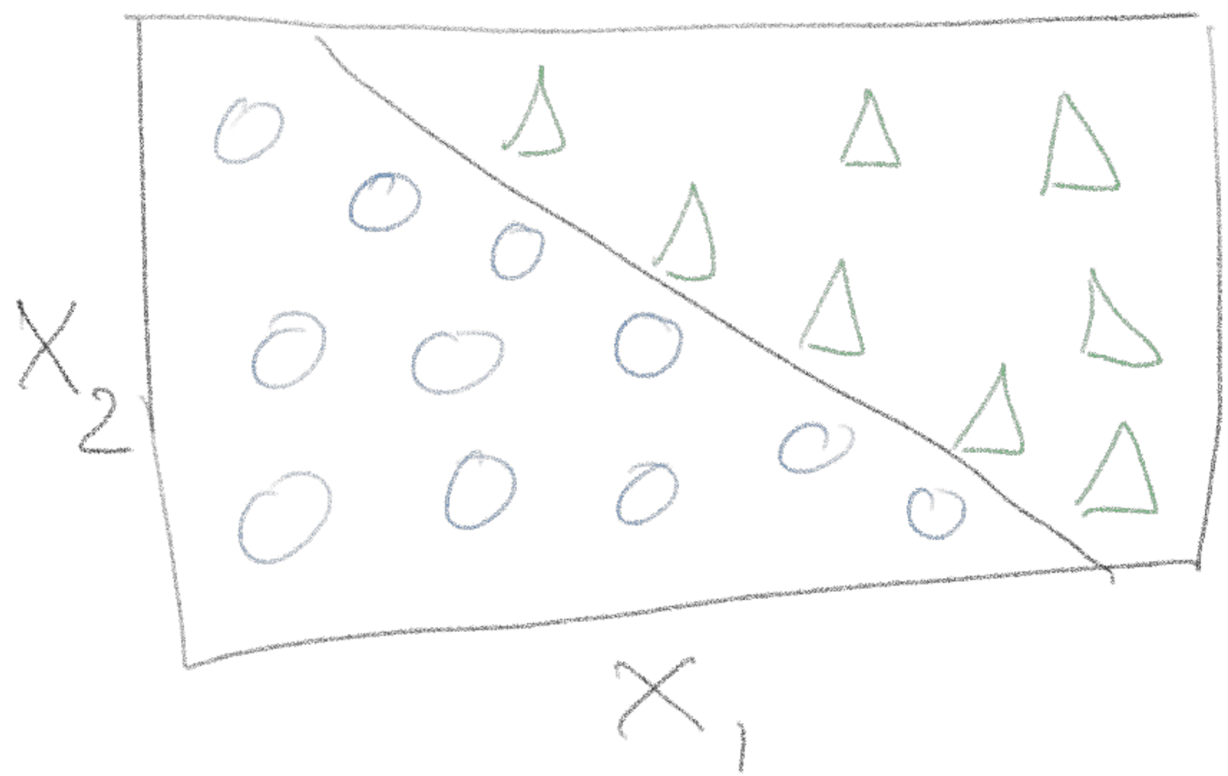
Quinlan 1986

where the split information measures the entropy of the feature:

$$SplitInfo(\mathscr{D}, x_j) = -\sum_{v \in x_j} \frac{|\mathscr{D}_v|}{|\mathscr{D}|} \log_2 \frac{|\mathscr{D}_v|}{|\mathscr{D}|}$$

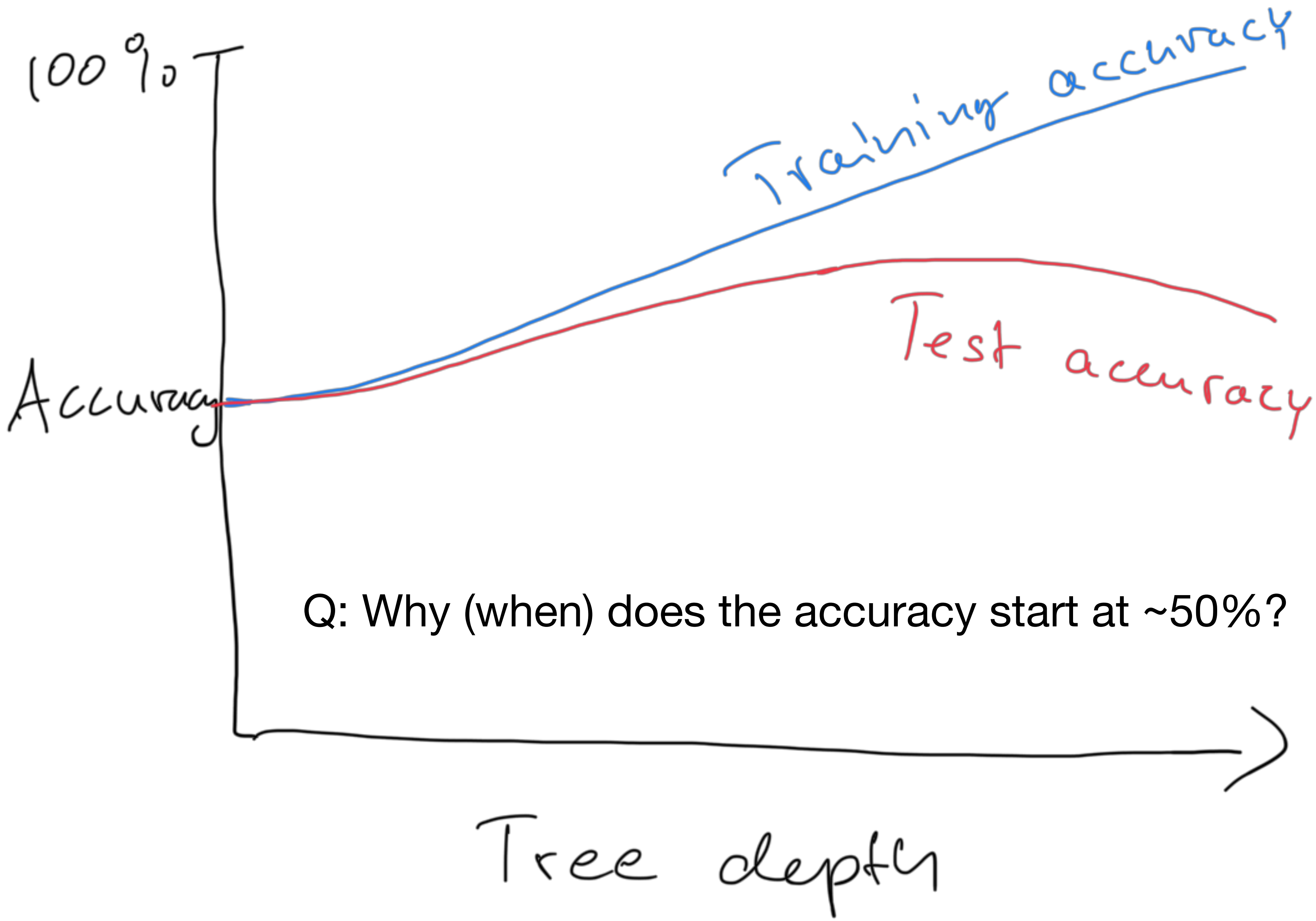
Penalizes splitting categorical attributes with many values (e.g., think date column, or really bad: row ID) via the split information

Shortcomings



How would the decision tree split look like?

Overfitting



Q: Why (when) does the accuracy start at ~50%?

Pre-Pruning

- Set a depth cut-off (maximum tree depth) *a priori*

- Cost-complexity pruning: , where is an impurity measure, is a tuning parameter, and is the total number of nodes.

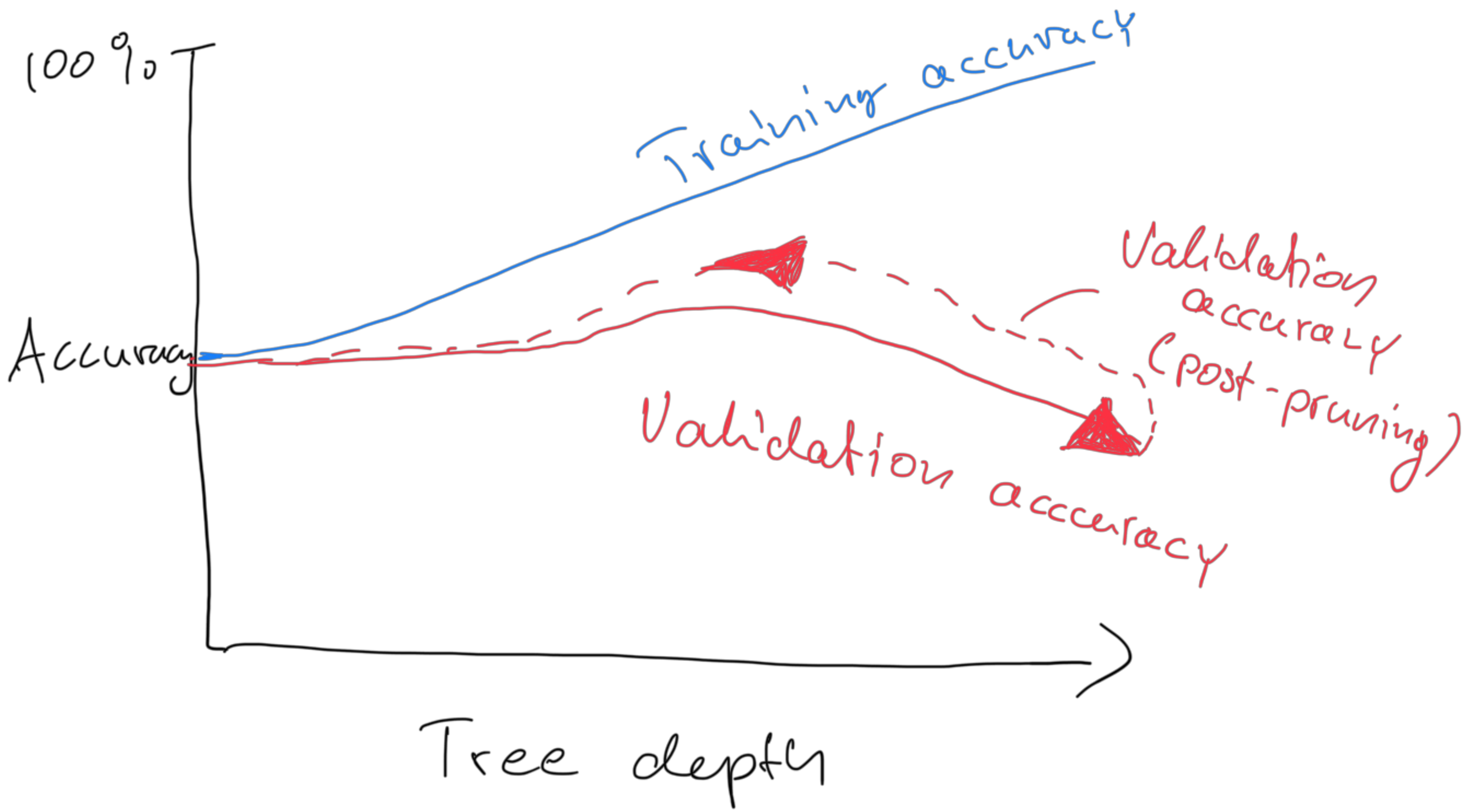
- Stop growing if split is not statistically significant (e.g., Chi^2 test)

- Set a minimum number of data points for each node

Post-Pruning

- Grow full tree first, then remove nodes, in C4.5
- Reduced-error pruning, remove nodes via validation set eval. (problematic for limited data)
- Can also convert trees to rules first and then prune the rules

Post-Pruning



Regression Trees

Summary: Pros and Cons

- (+) Easy to interpret and communicate
- (+) Can represent "complete" hypothesis space
- (-) Easy to overfit
- (-) Elaborate pruning required
- (-) Expensive to just fit a "diagonal line"
- (-) Output range is bounded (dep. on training examples) in regression trees

Reading Assignments

Python Machine Learning, 2nd Ed., Ch03 pg. 88-104

Demo

06_trees_demo.ipynb