Universität Stuttgart
KI – Institute for Artificial Intelligence

Analytic Computing

Machine Learning
8 Decision Trees

Prof. Dr. Steffen Staab

Nadeen Fatallah
Daniel Frank
Akram Sadat Hosseini
Jiaxin Pan
Osama Mohamed
Arvindh Arunbabu
Tim Schneider
Yi Wang

https://www.ki.uni-stuttgart.de/

- based on slides by
  - Thomas Gottron, U. Koblenz-Landau

https://west.uni-koblenz.de/de/studying/courses/ws1718/machine-learning-and-data-mining-1

Learning Objectives

- Entropy, Cross-entropy, Kullback-Leibler divergence
- What is a decision tree?
- How to construct a decision tree from training data?
- What is information gain?
- Why to use gain ratio?
- How to prune a decision tree?
- How to split attribute values?
- What is overfitting?
- What is binning and how does it work?

Measuring Distributions

Entropy

- Entropy (Shannon 1948) is a measure of
  - information content (in information theory)
  - uncertainty
- Given random variable X, entropy $H(X)$ is defined as

$$H(X) = -\sum_{x} P(X = x) \cdot \log_b P(X = x)$$

- Alternative, equivalent formulations are

$$H(X) = E\big(I(X)\big) = E[-\log_b P(X)]$$

where $I(x)$ denotes the information content of character x

Typically, we assume $b = 2$

Entropy of coin tosses

- Entropy of a fair coin toss
  - $X = \{h, t\}, P(X = h) = P(X = t) = 0.5$
  - How uncertain am I about the coint toss?
    Or: how much information have I received when I have been told the outcome?
    $$-(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = -\big(0.5 \cdot (-1) + 0.5 \cdot (-1)\big) = 1$$
  - 1 bit
- Entropy of an unfair coin toss
  - $X = \{h, t\}, P(X = h) = 0, P(X = t) = 1$
    $$-(0 \log_2 0 + 1 \log_2 1) = 0$$

Cross Entropy

For discrete probability distributions P, Q defined on the same probability space X the **cross entropy** is defined as:

$$H(P, Q) = E_P[-\log Q] = -\sum_{x \in X} P(X = x) \log Q(X = x)$$

“How far is Q away from P?“

Drawback: $H(P, P) = H(P)$, which usually is not 0

Indeed: $H(P, Q) = H(P) + D(P||Q)$, see next slide

Relative Entropy (Kullback-Leibler Divergence)

For discrete probability distributions P, Q defined on the same probability space X the **relative entropy** is defined as:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right) = -\sum_{x \in X} P(x) \log\left(\frac{Q(x)}{P(x)}\right)$$

This can be generalized for continuous distributions with probability density functions p, q:

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} \, dx$$

Interpretation of relative entropy

„How far is the distribution Q from the distribution P?“

$D(P||Q)$ measures the expected number of extra bits required to encode samples from P using a code optimized for Q.

$D(P||Q)$ is not symmetric, but $D(P||P) = 0$

$$D_{KL}(P||Q) = -\sum_{x \in X} P(x) \log\left(\frac{Q(x)}{P(x)}\right)$$

Relating cross entropy to maximum likelihood

Given data $\{(x_i, y_i)\}_{i=1}^{N}$
we maximize likelihood for our model over parameter configurations β:

$$\prod_{i=1}^{N} P(y_i|x_i) = \prod_{j} q_j^{N_{y_j}}$$

Each (x_i, y_i) in the training set has the same probability, therefore

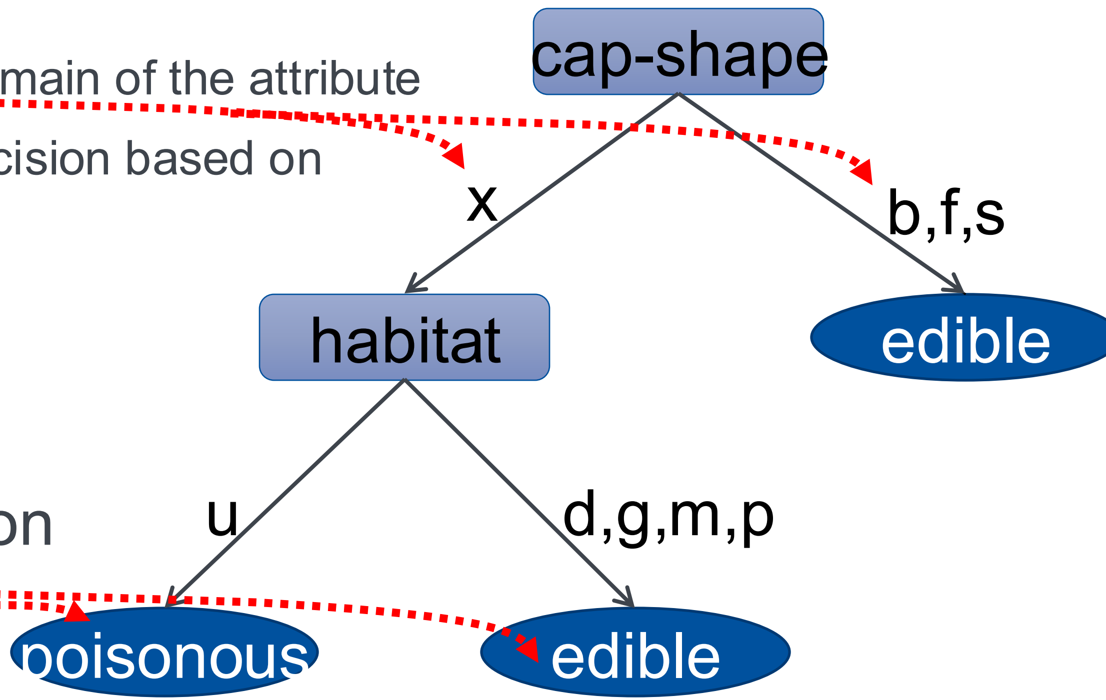
$$\frac{1}{N}\log\prod_{j} q_j^{N_{y_j}} = \sum_{j} \frac{N_{y_j}}{N}\log q_j = -H(P(Y), P(Y|X))$$

Maximizing likelihood is equivalent to minimizing cross entropy

Decision Trees

Structure of a Decision Tree

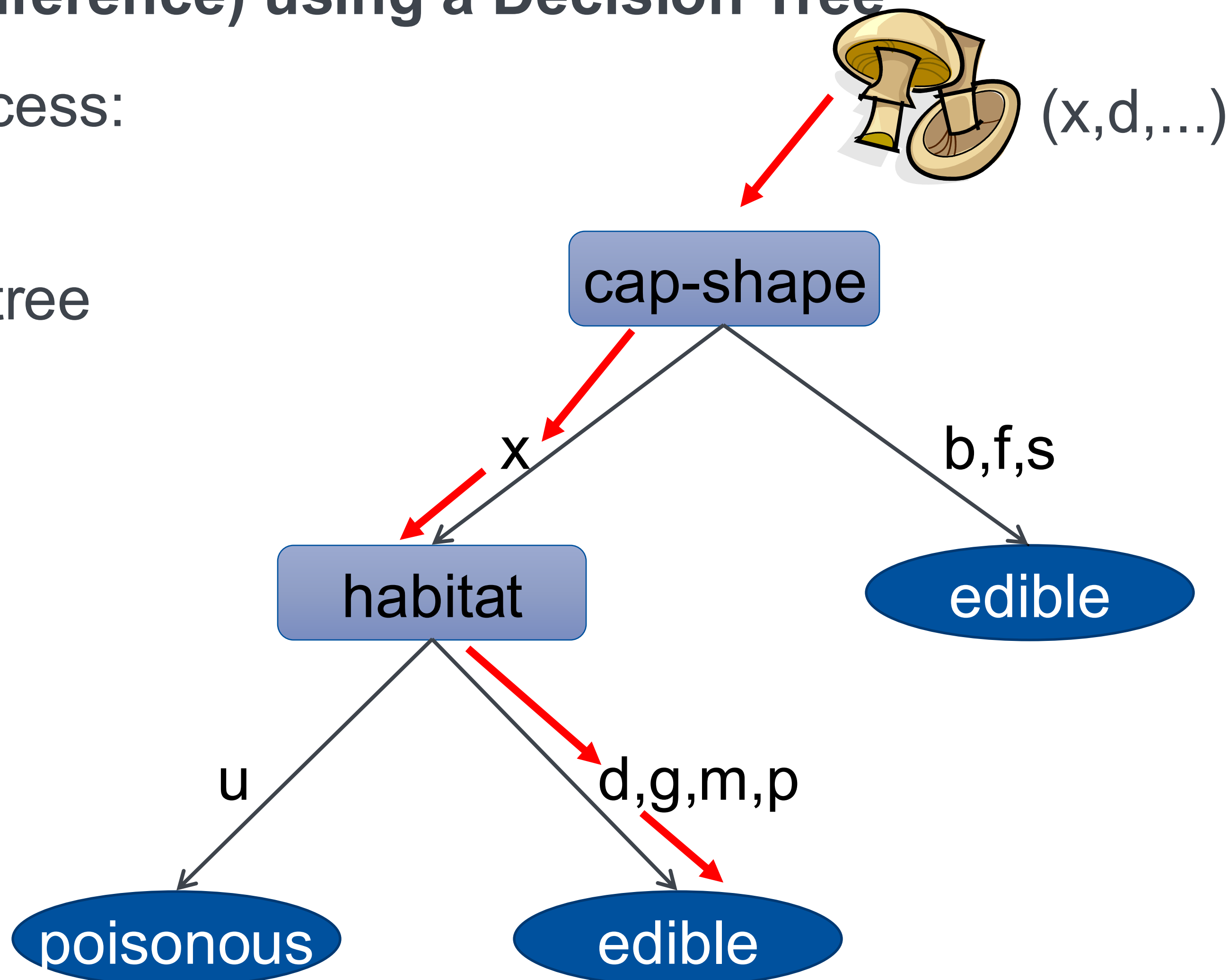
- Each inner node
  - Labeled with one attribute name (univariate)
  - Outgoing edges
    - labeled with values from domain of the attribute
    - represent n-ary discrete decision based on single named attribute
- Child nodes
    - segment the data
- Leaf nodes: classification
    - Assign one class label



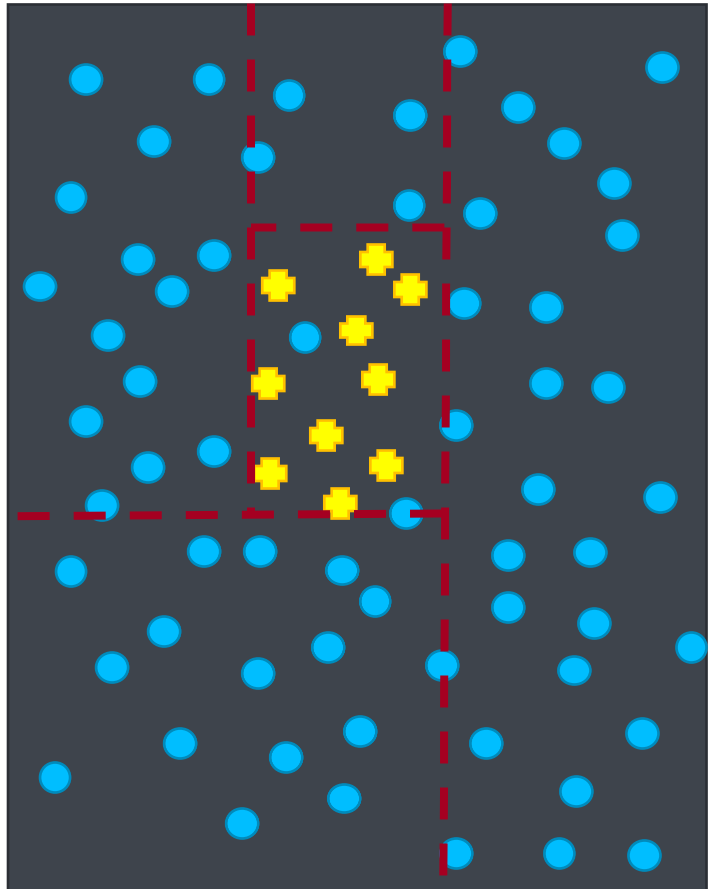
Classification (inference) using a Decision Tree

Classification process:

- Object "follows" the paths in the tree until it reaches a class label



Decision Tree: Divide and conquering the object space



Decision Trees

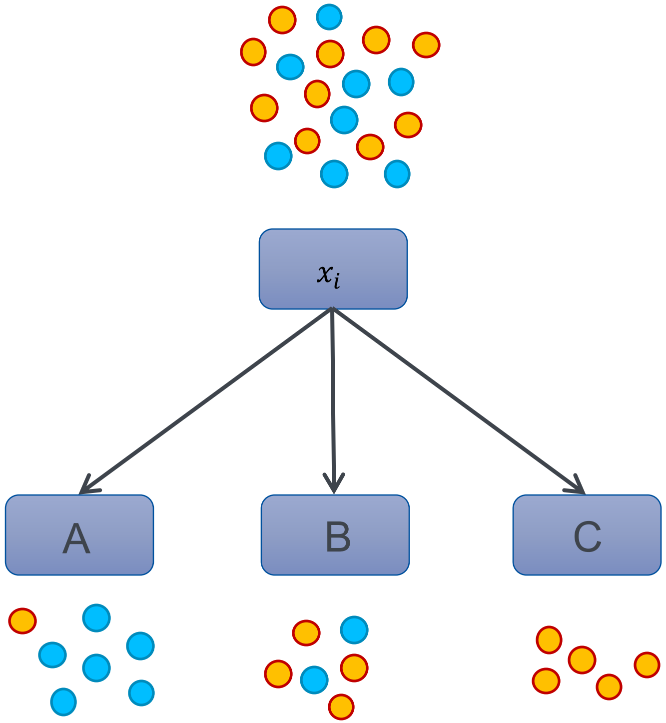
- Model can be interpreted by humans quite well
  - unless the tree is too deep and/or too broad

- Which tree is a good tree?
  - Loss function
    - though it was/is not called ”loss function” wrt decision trees
- Efficient construction needed
  - too many trees to try and compare all of them…

Inductive Construction of Decision Trees

General Idea: Greedy tree construction

- Tree grows from root
  - root is assigned no *intension* (i.e. no attributes)
  - root is assigned total *extension* (i.e. all training objects)
- Inductive principle:
  Each node grows children which are more „pure“ in their prediction quality over the training data
  - Children have one more attribute restriction (i.e. larger intension)
  - Children partition the extension
- When stopping criteria is met nodes are labeled with classes



Central metrics: Entropy and Information Gain

- Entropy of a distribution P over events X

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

  - Expected number of bits needed to encode an event using optimal compression

P_1	26%	20%	37%	17%	$H(P_1) = 1.935$
P_2	1%	97%	1%	1%	$H(P_2) = 0.242$
P_3	25%	25%	25%	25%	$H(P_3) = 2.000$

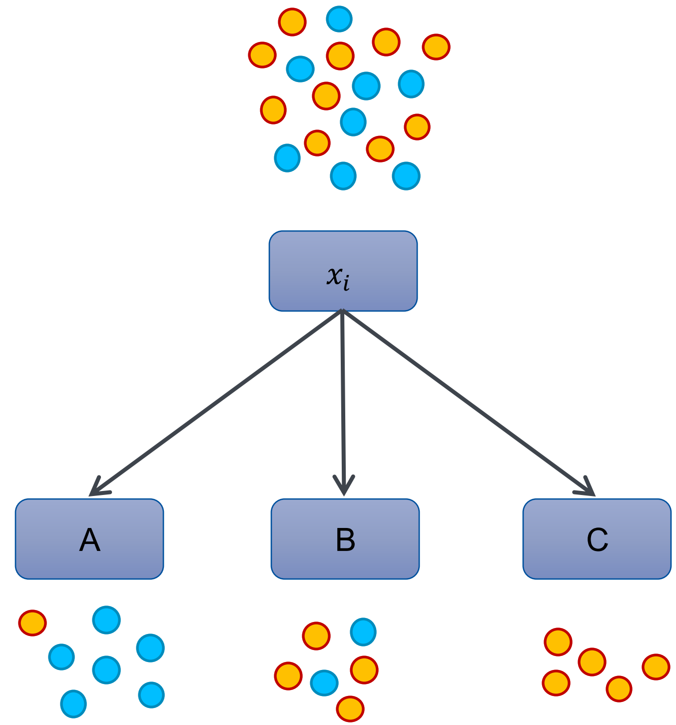
Low Entropy means skewed distribution

Skewed distributions provide more insights

- Here: we apply entropy on the observed class labels
- Aim: reduce entropy in the child nodes

Central metrics: Entropy and Information Gain

- Example
  - Parent node:
    - 10 orange, 8 blue objects
    - Entropy H = 0.997
  - Child Node A:
    - 1 orange, 6 blue objects
    - Entropy H = 0.592
  - Child Node B:
    - 4 orange, 2 blue objects
    - Entropy H = 0.918
  - Child Node C:
    - 5 orange, 0 blue objects
    - Entropy H = 0



Central metrics: Entropy and Information Gain

- Compute the expected entropy in the child nodes of node t
  - Probability of landing in a node
  - Entropy observed in this node

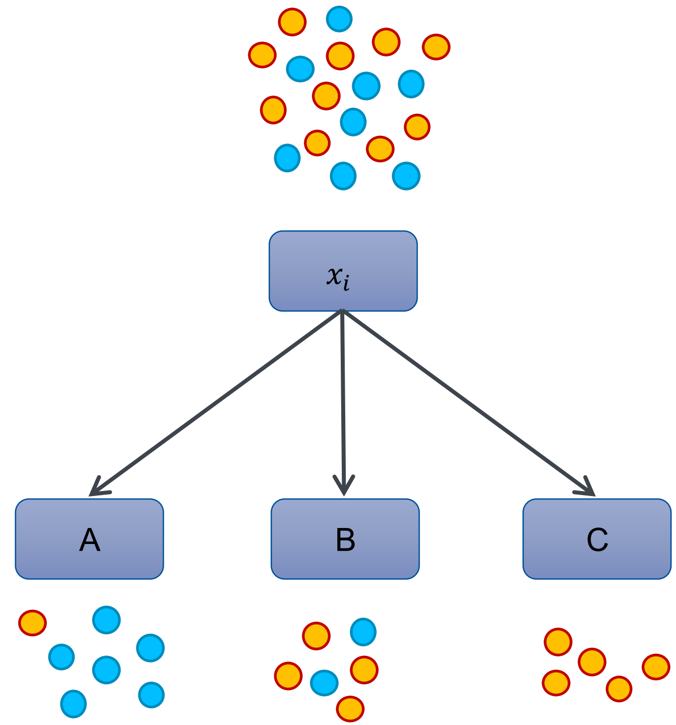
$$\sum_{n \in children(t)} p(n) \cdot H(P_n)$$

- Information gain:
  - Difference between entropy in a node and the expected entropy in its children nodes

$$IG = H(P_t) - \sum_{n \in children(t)} p(n) \cdot H(P_n)$$

Central metrics: Entropy and Information Gain

- Example
  - Parent node:
    - Entropy H = 0.997
  - Child Node A:
    - 1 orange, 6 blue objects
    - Entropy H = 0.592
  - Child Node B:
    - 4 orange, 2 blue objects
    - Entropy H = 0.918
  - Child Node C:
    - 5 orange, 0 blue objects
    - Entropy H = 0



$$IG = 0.977 - \left(\frac{7}{18} \cdot 0.592 + \frac{6}{18} \cdot 0.918 + \frac{5}{18} \cdot 0\right) = 0.441$$

Decision Tree Induction

- Top down approach
  - Start from “empty” root node.
  - Apply *GrowTree* to root node
- *GrowTree* (empty node):
  - If stop criterion does match node
    - Render node into a label
    - Use most frequent class
  - Else
    - Render node into decision node
    - Chose attribute to maximize information gain
    - Introduce empty child nodes
    - Apply GrowTree to child nodes
- Stop criteria: zero entropy, too few instances or all attributes have equal values

Example

???

- Entropy root node
  - 100 instances
  - 21 poisonous
  - 79 edible
  - Entropy: H(P)=0.741
- Attribute: Cap-shape

value	p	e	total	entropy
b	0	29	29	0
f	1	13	14	0.371
s	0	3	3	0
x	20	34	54	0.951

-

Expected Entropy: 0.565

$$IG = 0.176$$

Example

- Attribute: Cap-surface

value	p	e	total	entropy
f	0	14	14	0
s	8	29	37	0.753
y	13	36	49	0.835

$$IG = 0.053$$

- Attribute: Cap-color

$$IG = 0.236$$

- ...
- Attribute: Habitat

$$IG = 0.279$$

Example

???

value	p	e	total	entropy
d	0	8	8	0
g	8	28	36	0.764
m	0	28	28	0
p	0	8	8	0
u	13	7	20	0.934

- Attribute: Cap-surface

value	p	e	total	entropy
f	0	14	14	0
s	8	29	37	0.753
y	13	36	49	0.835

$$IG = 0.053$$

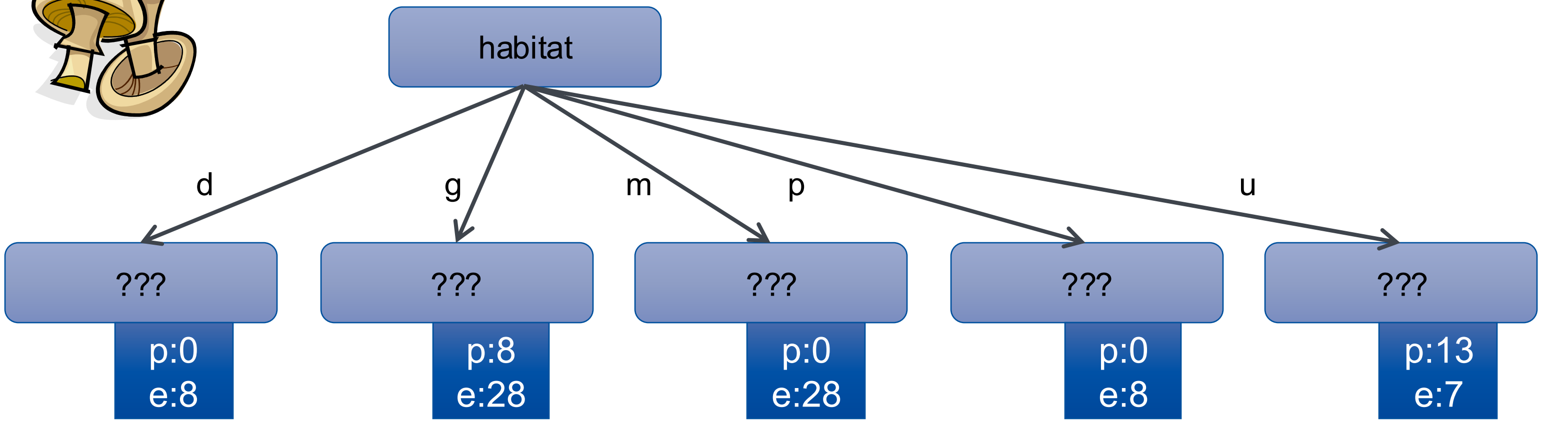
- Attribute: Cap-color

$$IG = 0.236$$

- …
- Attribute: Habitat

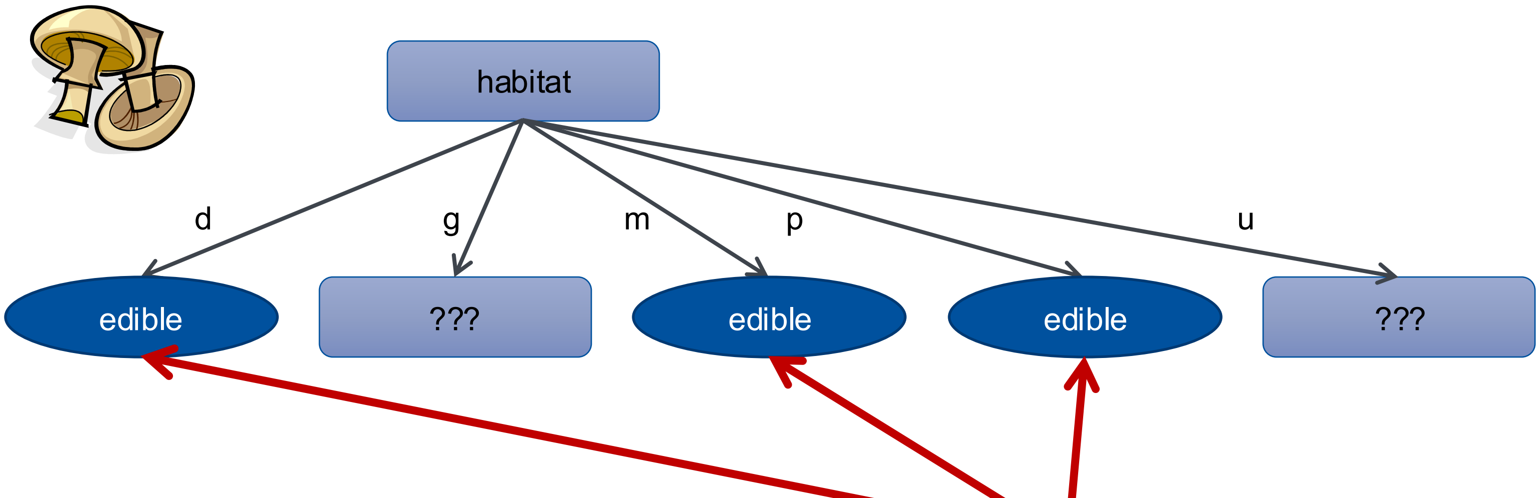
$$IG = 0.279$$

Example



value	p	e	total	entropy
d	0	8	8	0
g	8	28	36	0.764
m	0	28	28	0
p	0	8	8	0
u	13	7	20	0.934

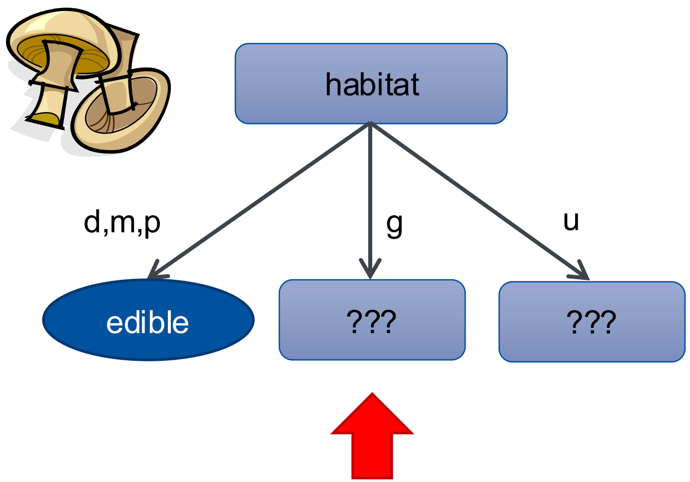
Example



value	p	e	total	entropy
d	0	8	8	0
g	8	28	36	0.764
m	0	28	28	0
p	0	8	8	0
u	13	7	20	0.934

Merge nodes (typically: post processing)

Example



value	p	e	total	entropy
d	0	8	8	0
g	8	28	36	0.764
m	0	28	28	0
p	0	8	8	0
u	13	7	20	0.934

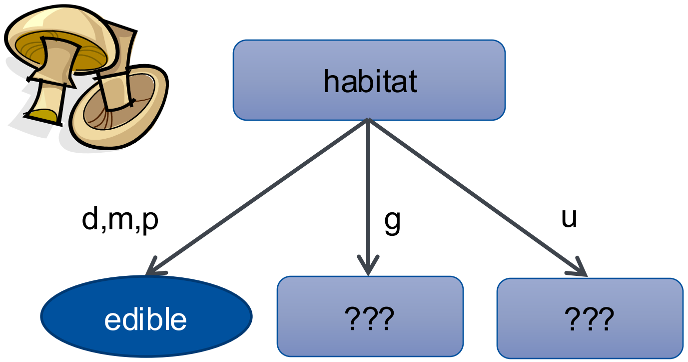
- Next node
  - 36 instances
  - 8 poisonous
  - 28 edible
  - Entropy: $H(P)=0.764$
- Attribute: Cap-shape

value	p	e	total	entropy
b	0	8	8	0
f	1	6	7	0.592
s	0	0	0	0
x	7	14	21	0.918

- Expected Entropy: 0.651

$$IG = 0.113$$

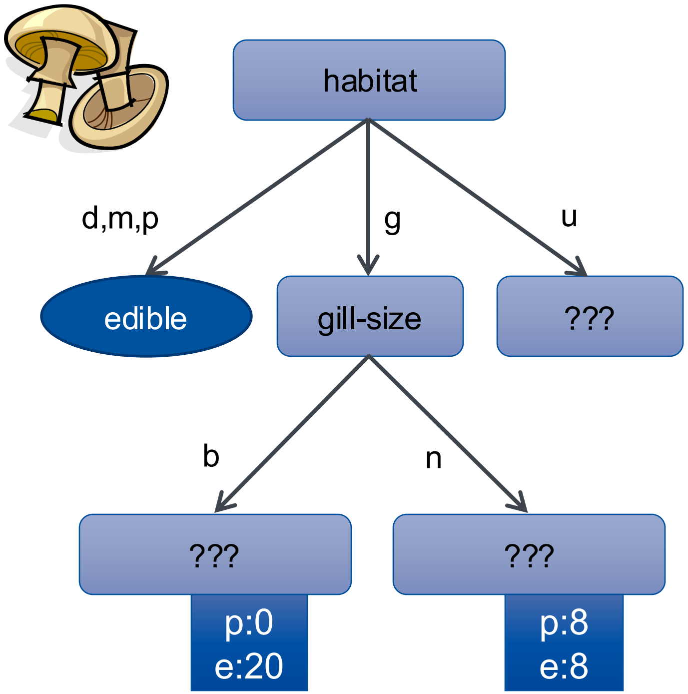
Example



- Attribute: Cap-surface
  $$IG = 0.113$$
- ...
- Attribute: Gill-size
  $$IG = 0.32$$

value	p	e	total	entropy
b	0	20	20	0
n	8	8	16	1

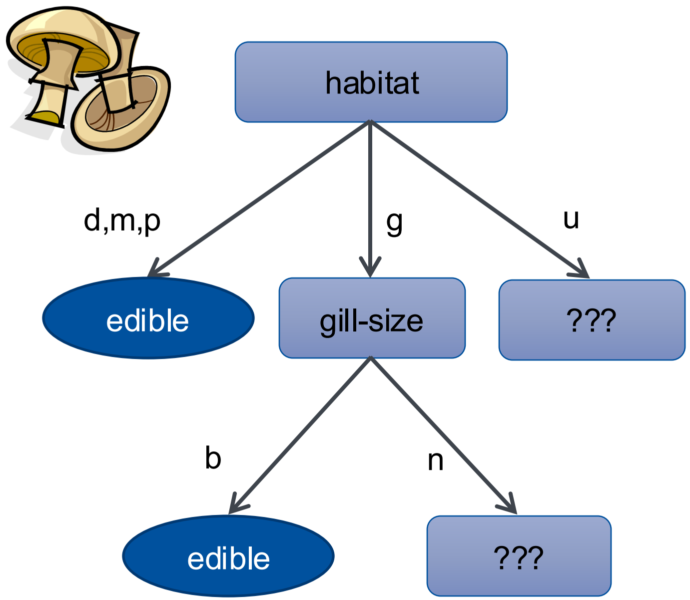
Example



- Attribute: Cap-surface
  $$IG = 0.113$$
- ...
- Attribute: Gill-size
  $$IG = 0.32$$

value	p	e	total	entropy
b	0	20	20	0
n	8	8	16	1

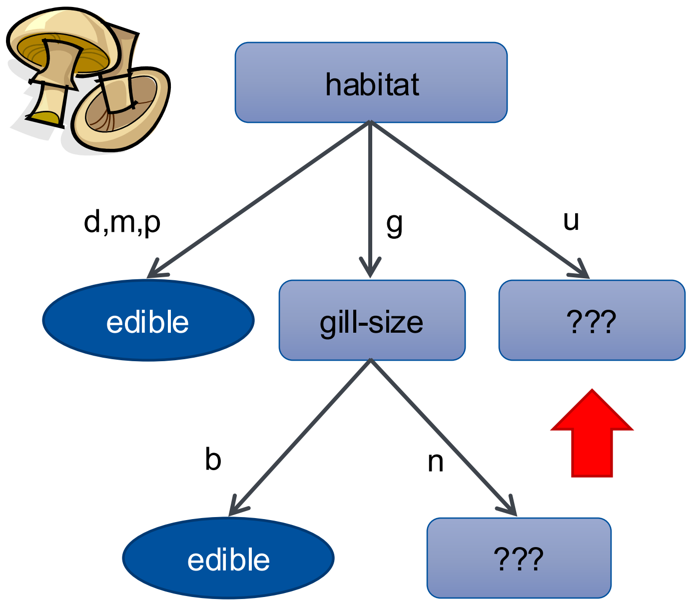
Example



- Attribute: Cap-surface
$$IG = 0.113$$
- ...
- Attribute: Gill-size
$$IG = 0.32$$

value	p	e	total	entropy
b	0	20	20	0
n	8	8	16	1

Example



- Next node
  - 20 instances
  - 13 poisonous
  - 7 edible
  - Entropy: H(P)=0.934
- …

Impact of a Perfect Predictor Attribute

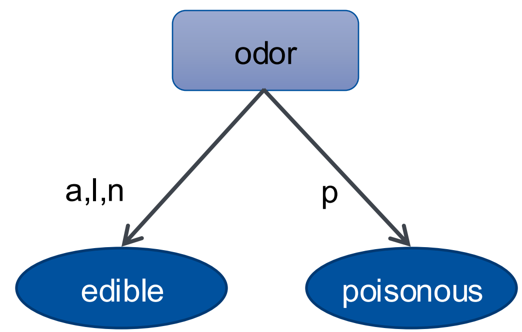
- Assume an attribute which is perfectly indicative for the class label.

new

cap-shape	cap-surface	cap-color	bruises	gill-spacing	gill-size	gill-color	habitat	odor
bell=b convex=x flat=f sunken=s	fibrous=f scaly=y smooth=s	brown=n gray=g white=w yellow=y	bruises=t no=f	close=c crowded=w	broad=b narrow=n	black=k brown=n gray=g pink=p white=w	grasses=g meadows=m paths=p urban=u woods=d	almond=a, anise=l none=n pungent=p

value	p	e	total	entropy
a	0	31	31	0
l	0	35	35	0
n	0	13	13	0
p	21	0	21	0

$$IG = 0.741$$



Avoiding overfitting

Attributes with many values

- The introduced approach for constructing decision trees favours attributes which have many values
  - Extreme example:
    - Object ID as attribute – one value per object
    - Clearly identifies training objects and, thereby, their labels

- Normalize w.r.t. the distribution of attribute values, i.e. its entropy

$$H(x_i) = -\sum_{v} p(x_i = v) \cdot \log_2(p(x_i = v))$$

→ *Gain ratio* (or *Normalized Impurity Decrease*)

$$IGR = IG(x_i)/H(x_i)$$

Use gain ratio instead of information gain for deciding on which attribute to use.

Normalized Impurity Decrease (alternative notation)

- Use *Normalized Impurity Decrease* (Gain ratio) instead of *Impurity Decrease* (Information Gain). The *Normalized Impurity Decrease* at a node t is then:
  - $\hat{\Delta} I(t) = \frac{\Delta I(t)}{E_d(t)}$, where $E_d(t)$ is the *entropy of distribution* (Intrinsic Information).
  - $E_d(t) = -\sum_j P\left(t_j \middle| t\right) \log_2 P\left(t_j \middle| t\right)$,
  - $P\left(t_j \middle| t\right) = \frac{N_{t_j}}{N_t}$.

Example

- Root note
- Attribute: Cap-shape

value	p	e	total	entropy
b	0	29	29	0
f	1	13	14	0.371
s	0	3	3	0
x	20	34	54	0.951

- Expected Entropy: 0.565

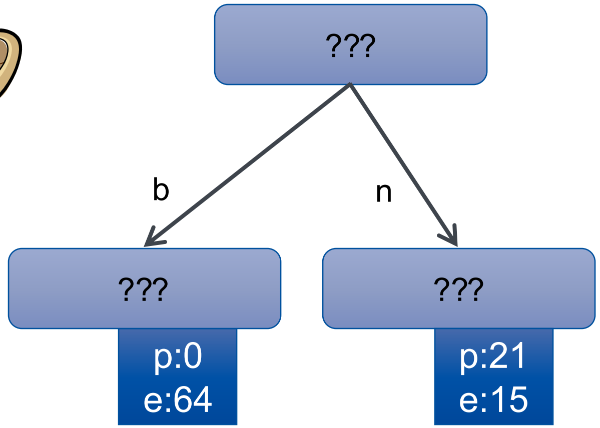
$$IG = 0.176$$

- Entropy of value distribution:1.547

$$IGR = 0.114$$

- IGR Habitat: 0.134

Example

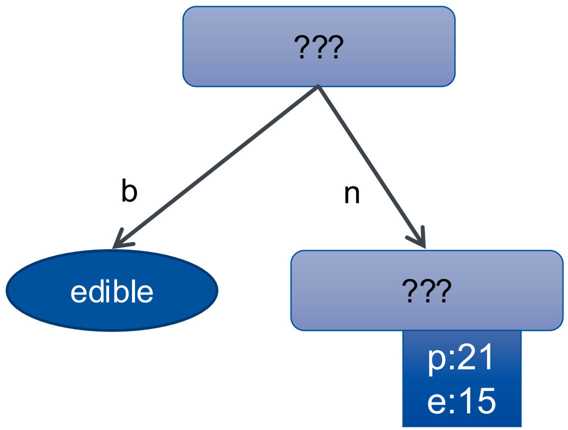


- Attribute: Gill-size

value	p	e	total	entropy
b	0	64	64	0
n	21	15	36	1

- Entropy of value distribution:0.943
- IGR gill-size: 0.412

Example



- Attribute: Gill-size

value	p	e	total	entropy
b	0	64	64	0
n	21	15	36	1

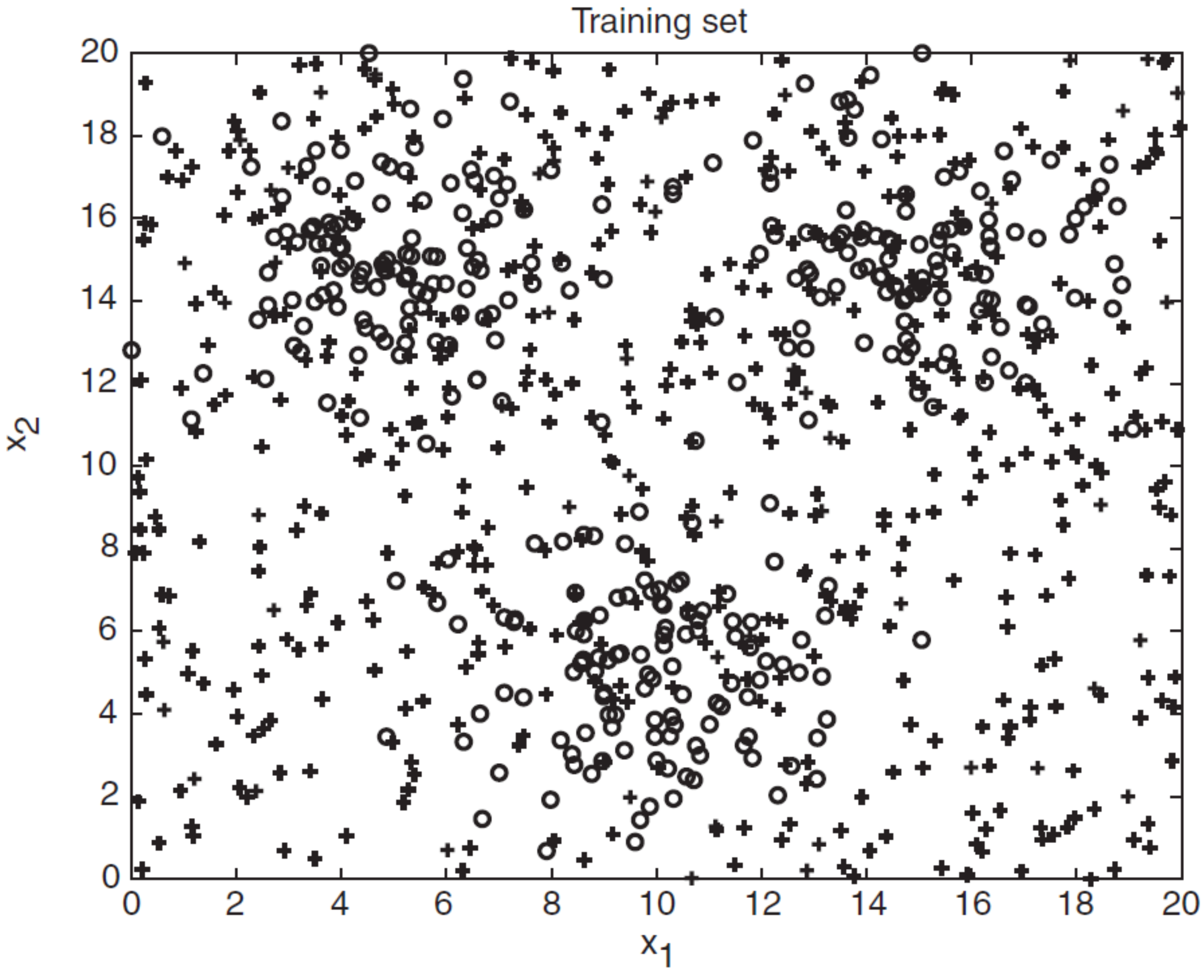
- Entropy of value distribution:0.943
- IGR gill-size: 0.412
- Next node ...

Overfitting

- Algorithm for decision tree induction is optimizing the <u>training error</u>
  - Adding more nodes to the tree will reduce the training error
  - Model becomes more complex!
  - Risk of overfitting

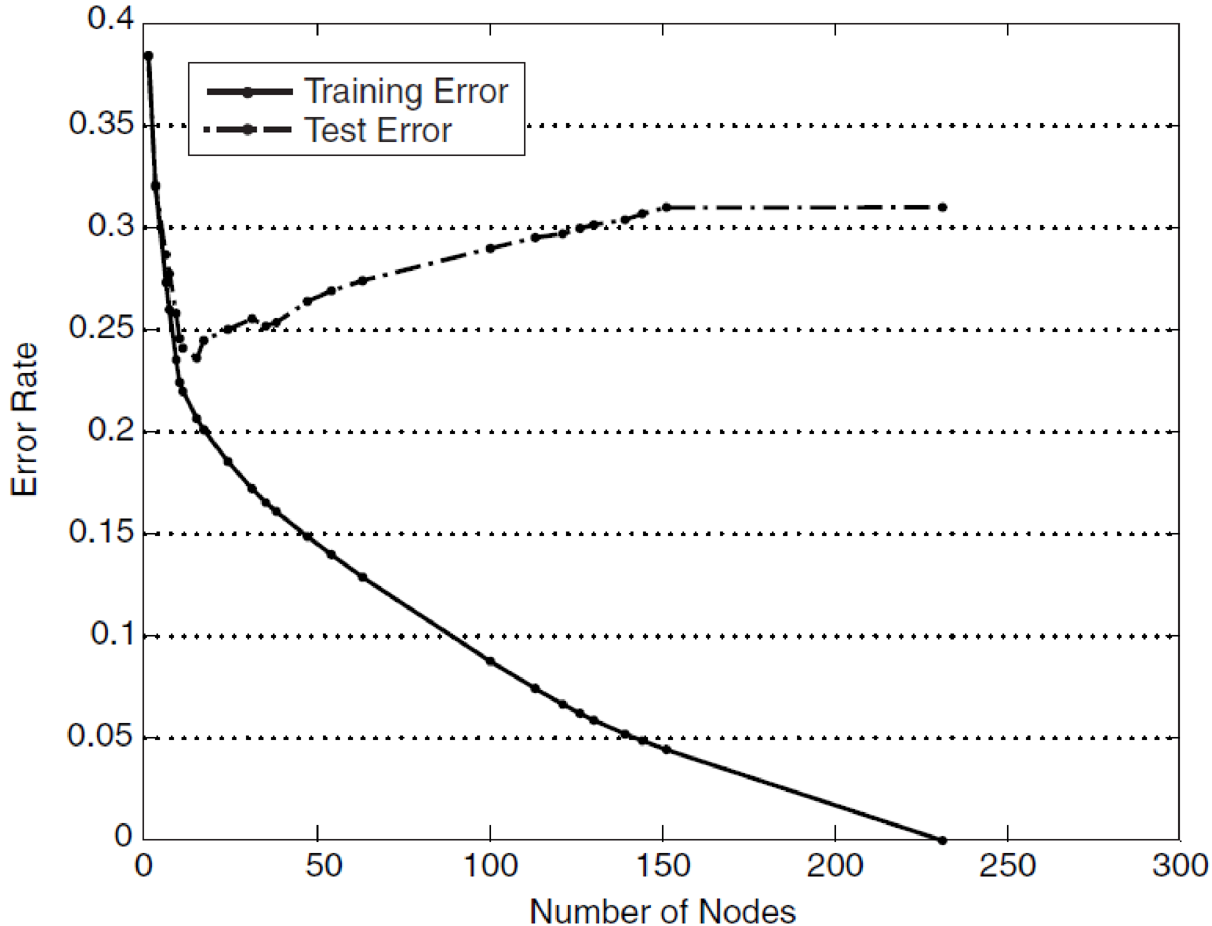
- Wanted: idea of the error rate on new data
  - <u>Generalization error</u>
  - Training error is not a good estimator for the generalization error

Overfitting



1200 o
1800 +

Overfitting



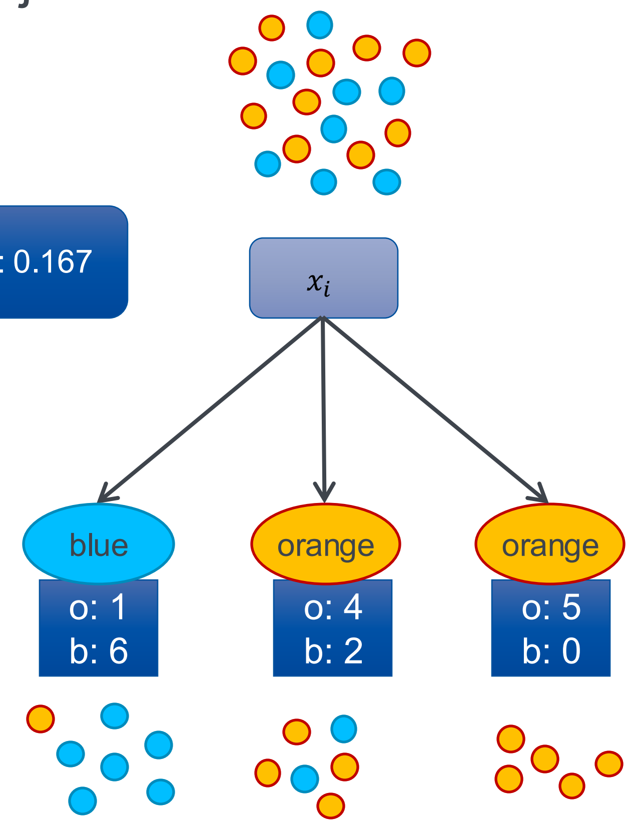
Data split: 30% training, 70% evaluation

Estimating the Generalization Error

- Determine for each node T with children $t \in T$ the number of classification errors on training data $(e(T), e(t))$ and the number of objects assigned to this node $(n(T), n(t))$
- Overall error rate:

$$\frac{1}{n(T)} \sum_{t \in T} e(t)$$

Example: 0.167



- Add a penalty for each node (pessimistic error estimate):

$$\frac{1}{n(T)} \sum_{t \in T} e(t) + \lambda$$

Example (λ=0.5): 0.25

Regularization

Tree Pruning

- During construction
  - Extend stop criterion:
    - Require minimum information gain
    - Require mimimum number of instances to be covered by a node
- Post processing
  - Remove leaf nodes
    - Optimize other criteria
      (e.g. estimates for the generalization error)
    - Prune excessively deep sub-trees
- Selection
  - Construct different trees (e.g chosing second best attributes, using subsets of training data)
  - Select best and smallest tree

Attribute Splitting

Attribute Splitting

- So far:
  - One child node per value
  - Problem:
    - High fan-out of tree
    - Not applicable to all attribute types

See data preproessing

- Alternative for nominal values:
  - Binary splits of value set
  - Advantage: binary tree
  - Disadvantage: Many possible splits
    - k values → $2^k - 2$ possible splits

Ordinal values

See data preprocessing

- We can use the order of values to create „meaningful“ splits

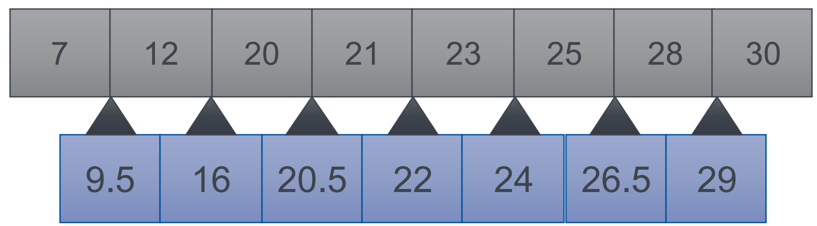
- Example:
  - Ratings of shares
  - Generate splits, such that neighbour categories are merged

	Moody's	S&P	Fitch	Meaning
	Aaa	AAA	AAA	Prime
	Aa1	AA+	AA+	
	Aa2	AA	AA	High Grade
Investment Grade	Aa3	AA-	AA-	
	A1	A+	A+	
	A2	A	A	Upper Medium Grade
	A3	A-	A-	
	Baa1	BBB+	BBB+	
	Baa2	BBB	BBB	Lower Medium Grade
	Baa3	BBB-	BBB-	
	Ba1	BB+	BB+	
	Ba2	BB	BB	Non Investment Grade Speculative
	Ba3	BB-	BB-	
	B1	B+	B+	
	B2	B	B	Highly Speculative
Junk	B3	B-	B-	
	Caa1	CCC+	CCC+	Substantial Risks
	Caa2	CCC	CCC	Extremely Speculative
	Caa3	CCC-	CCC-	
	Ca	CC	CC+	In Default w/ Little Prospect for Recovery
		C	CC	
			CC-	In Default
	D	D	DDD	

Dealing with numerical data

- Attributes with numerical data (e.g. height and diameter of a mushroom)
  - Sort observed values
  - Use inbetween values for the split

See data preproessing

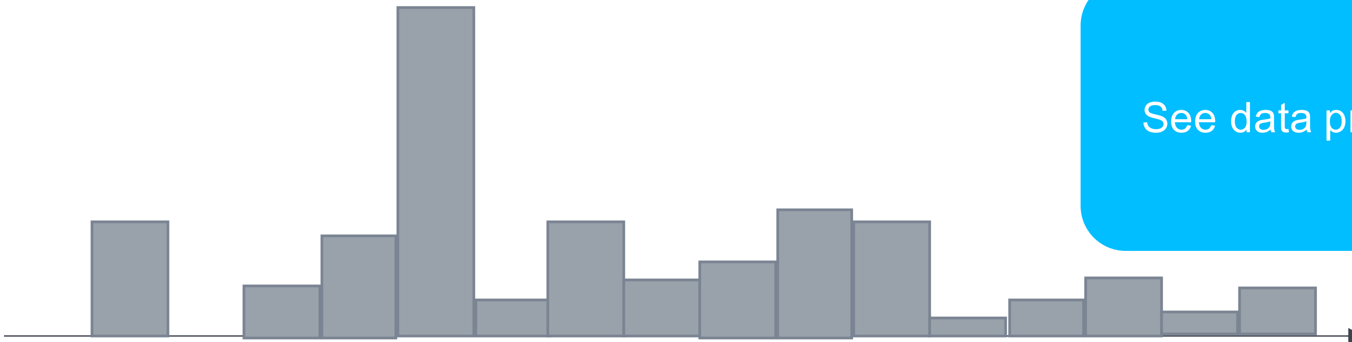


  - Using the middle between two observed values, provides better separation boundaries

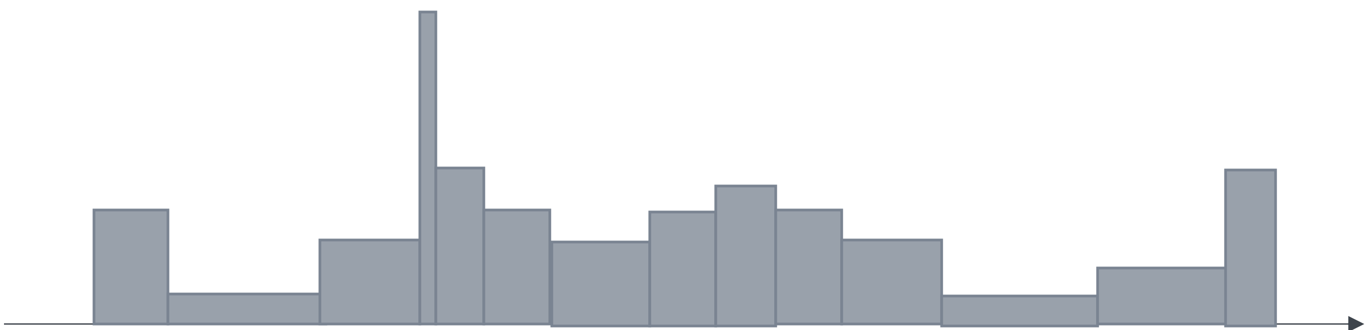
- Alternative: Binning – turns numerical data into ordinal data

Equi-width vs. Equi-depth Bins

See data preproessing



Fixed width for the bins



Fixed number of elements per bin (a.k.a. frequency binning)

Pros and Cons of Decision Trees

- Advantages
  - Interpretable.
  - Non-parametric.
  - Can handle missing data.
  - Low complexity (prediction) $O(l)$.
  - Invariant to feature scaling.
    - Does not require data normalization.
  - Can handle heterogeneous data.
    - Attributes of different types.
- Disadvantages
  - Splits are aligned w.r.t axes.
    - It might cause overfitting because the tree becomes far more complex than needed.

Universität Stuttgart
KI

Thank you!

Steffen Staab

E-Mail Steffen.staab@ki.uni-stuttgart.de

Telefon +49 (0) 711 685-88100

www.ki.uni-stuttgart.de/

Universität Stuttgart
Analytic Computing, KI
Universitätsstraße 32, 50569 Stuttgart