Business Analytics & Machine Learning

Decision Tree Classifiers

Prof. Dr. Martin Bichler

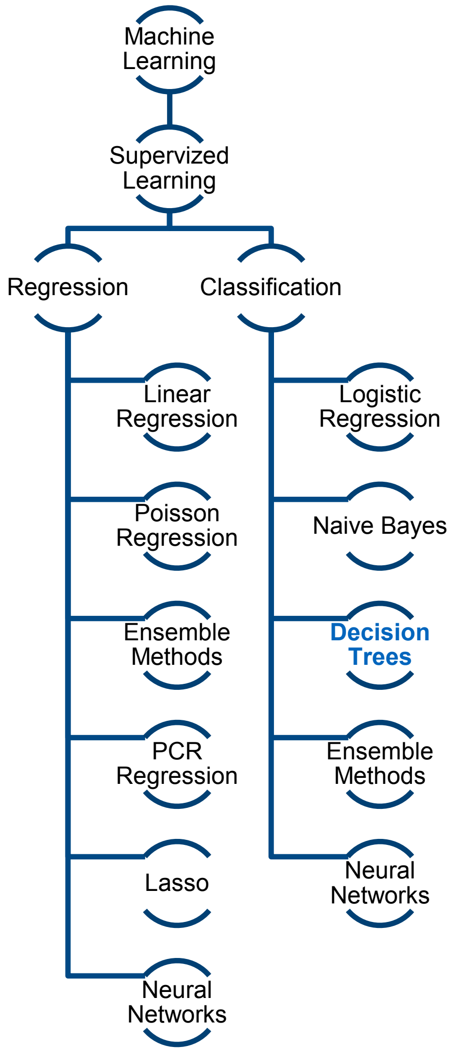
Department of Computer Science

School of Computation, Information, and Technology

Technical University of Munich

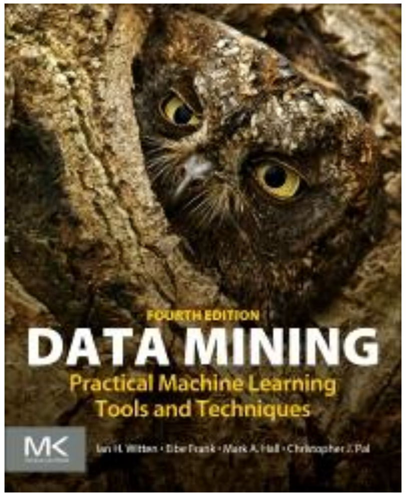
Course Content

- Introduction
- Regression Analysis
- Regression Diagnostics
- Logistic and Poisson Regression
- Naive Bayes and Bayesian Networks
- **Decision Tree Classifiers**
- Data Preparation and Causal Inference
- Model Selection and Learning Theory
- Ensemble Methods and Clustering
- Dimensionality Reduction
- Association Rules and Recommenders
- Convex Optimization
- Neural Networks
- Reinforcement Learning



Recommended Literature

- **Data Mining: Practical Machine Learning Tools and Techniques**
  - Ian H. Witten, Eibe Frank, Mark A. Hall, Christopher Pal
  - http://www.cs.waikato.ac.nz/ml/weka/book.html
  - Section: 4.3, 6.1


FOURTH EDITION
DATA MINING
Practical Machine Learning Tools and Techniques
MK
Ian H. Witten · Eibe Frank · Mark A. Hall · Christopher J. Pal

Alternative literature:

- **Machine Learning**
  - Tom M. Mitchell, 1997

- **Data Mining: Introductory and Advanced Topics**
  - Margaret H. Dunham, 2003

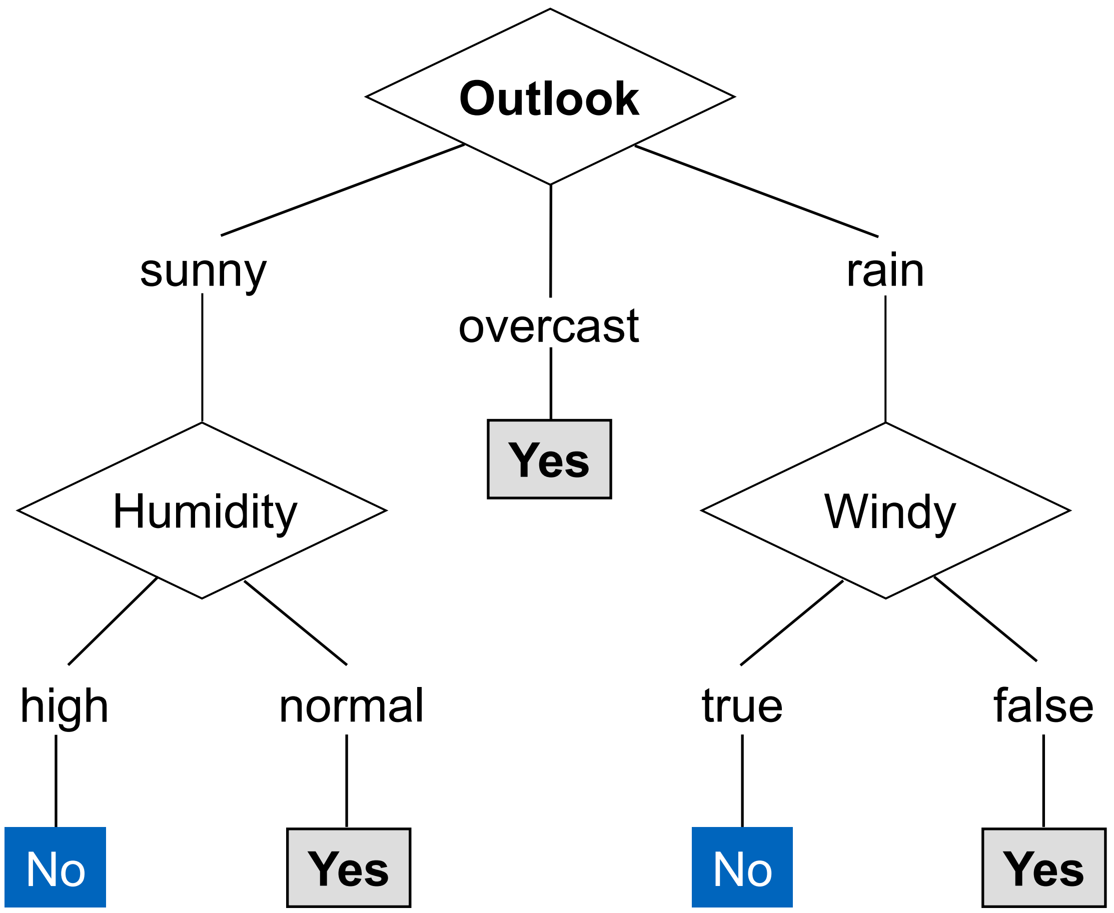
Agenda for Today

- **Choosing a splitting attribute in decision trees**
  - Information gain $\longrightarrow$ C4.5
  - Gain ratio
  - Gini index $\longrightarrow$ CART
- Numeric attributes
- Missing values
- Relational rules

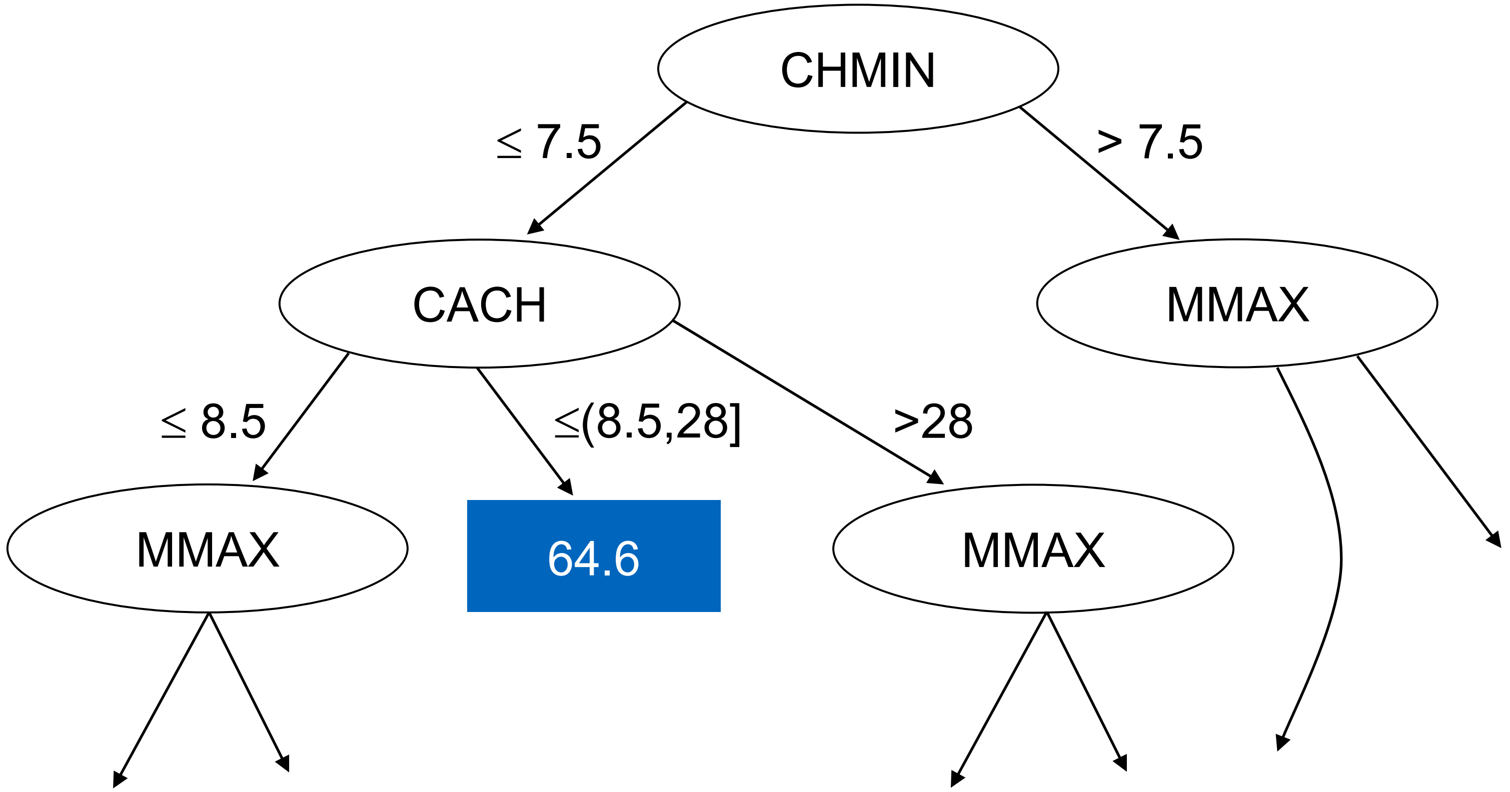
The Weather Data

x_0	x_1	x_2	x_3	y
Outlook	**Temp**	**Humidity**	**Windy**	**Play**
Sunny	Hot	High	False	**No**
Sunny	Hot	High	True	**No**
Overcast	Hot	High	False	**Yes**
Rainy	Mild	High	False	**Yes**
Rainy	Cool	Normal	False	**Yes**
Rainy	Cool	Normal	True	**No**
Overcast	Cool	Normal	True	**Yes**
Sunny	Mild	High	False	**No**
Sunny	Cool	Normal	False	**Yes**
Rainy	Mild	Normal	False	**Yes**
Sunny	Mild	Normal	True	**Yes**
Overcast	Mild	High	True	**Yes**
Overcast	Hot	Normal	False	**Yes**
Rainy	Mild	High	True	**No**

Example Tree for “Play?”

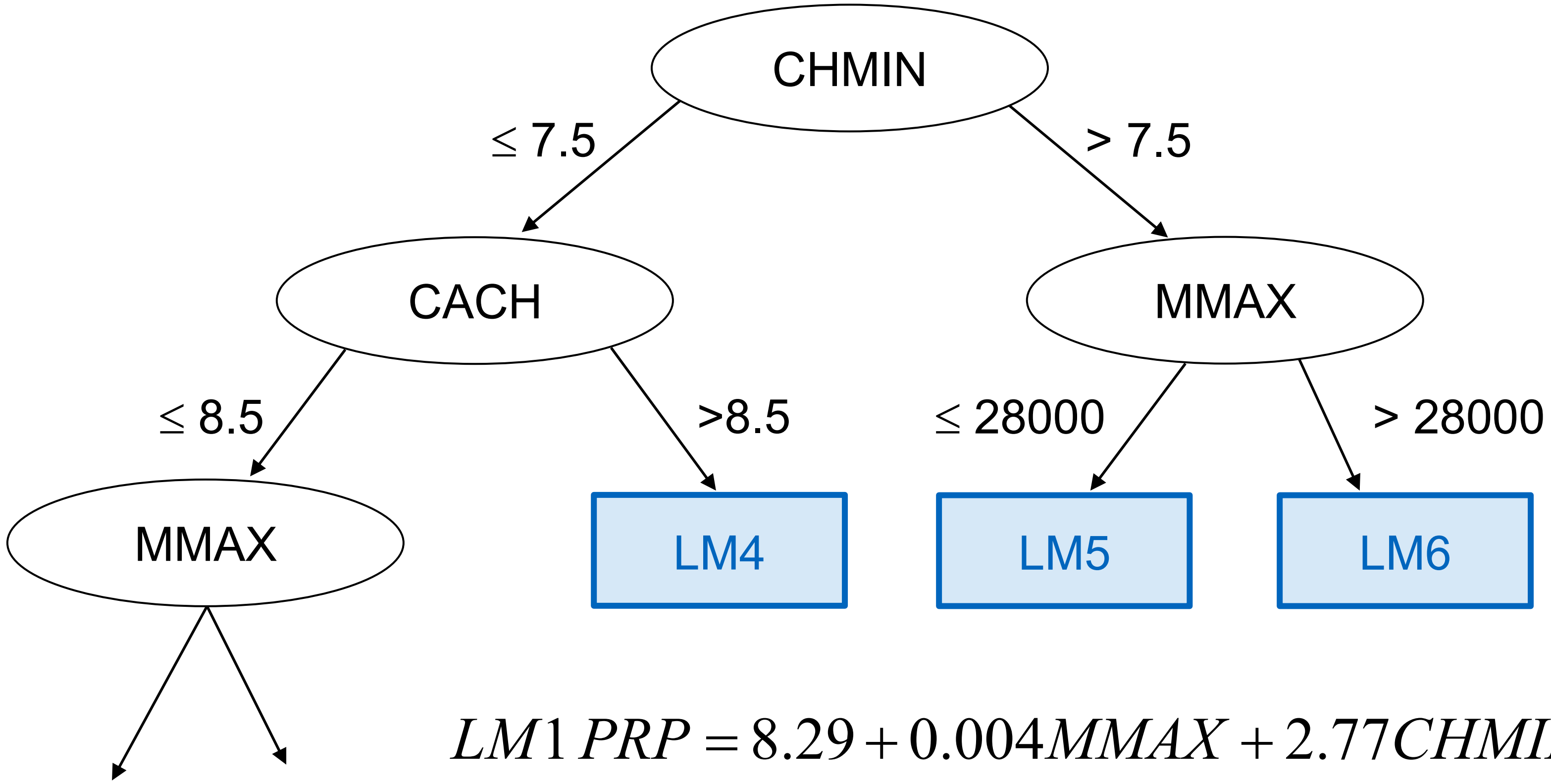


Regression Tree



- High Accuracy
- Large and possibly awkward

Model Trees



$$LM1\ PRP = 8.29 + 0.004 MMAX + 2.77 CHMIN$$

$$LM2\ PRP = \cdots$$

$$\vdots$$

Decision Trees

An internal node is a test on an attribute.

A branch represents an outcome of the test, e.g., $Color = red$.

A leaf node represents a class label or class label distribution.

At each node, one attribute is chosen to split training examples into distinct classes as much as possible.

A new case is classified by following a matching path to a leaf node.

Building Decision Tree

Top-down tree construction

- At start, all training examples are at the root
- Partition the examples recursively by choosing one attribute each time

Bottom-up tree pruning

- Remove subtrees or branches, in a bottom-up manner, to improve the estimated accuracy on new cases

Choosing the Splitting Attribute

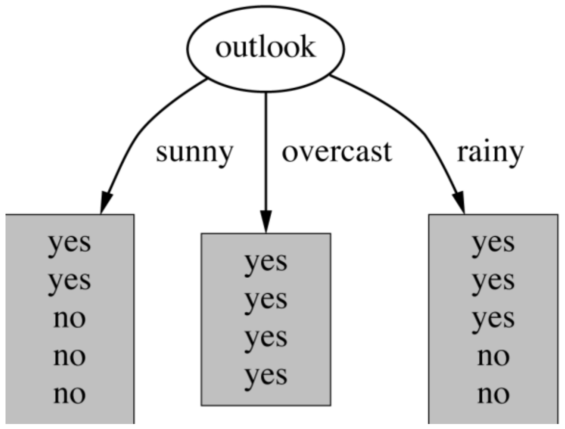
At each node, available attributes are evaluated on the basis of separating the classes of the training examples.

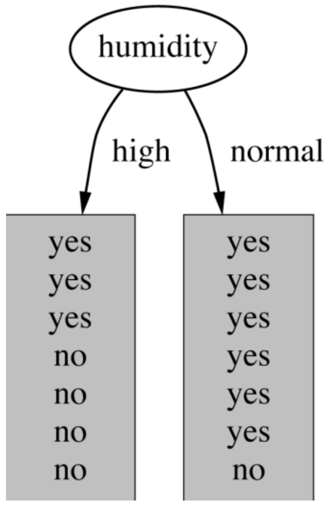
A "goodness" function is used for this purpose.

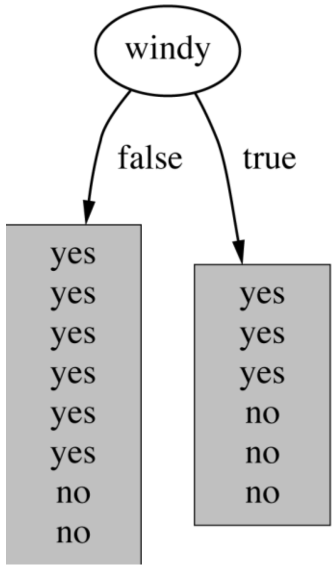
Typical goodness functions:

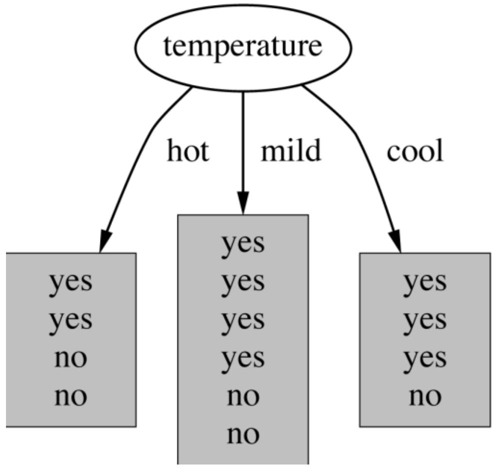
- information gain (ID3/C4.5)
- information gain ratio
- gini index (CART)

Which Attribute to Select?









A Criterion for Attribute Selection

Which is the best attribute?

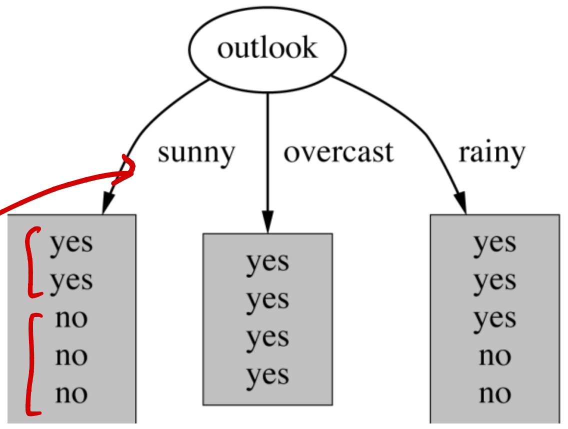
- The one which will result in the smallest tree.
- Heuristic: choose the attribute that produces the "purest" nodes.

Popular impurity criterion: information gain.

- Information gain increases with the average purity of the subsets that an attribute produces.

Strategy: choose attribute that results in greatest information gain.

Example: attribute "Outlook"



"Outlook" = "Sunny":

$\text{info}([2,3]) = \text{entropy}(2/5, 3/5) = -\,2/5\,log_2(2/5\,) - 3/5\,log_2(3/5) = 0.971 \text{ bits}$

"Outlook" = "Overcast":

$\text{info}([4,0]) = \text{entropy}(1,0) = -1log_2(1) - 0\,log_2(0) = 0 \text{ bits}$

*Note: log(0) is not defined, but we evaluate 0*log(0) as zero*

"Outlook" = "Rainy":

$\text{info}([3,2]) = \text{entropy}(3/5, 2/5) = -3/5\,log_2(3/5) - 2/5\,log_2(2/5) = 0.971 \text{ bits}$

Expected information for attribute:

$$\begin{aligned}\text{info}([2,3],[4,0],[3,2]) &= (5/14) \times 0.971 + (4/14) \times 0 + (5/14) \text{ x } 0.971 \\ &= 0.693 \text{ bits}\end{aligned}$$

Computing the Information Gain

Information gain: (information before split) – (information after split)

$\text{gain}(\text{Outlook}) = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2])$

$= 0.940 - 0.693 = 0.247 \text{ bits}$

Information gain for attributes from weather data:

$\text{gain}(\text{Outlook}) = 0.247 \text{ bits}$ $\leftarrow$

$\text{gain}(\text{Temperature}) = 0.029 \text{ bits}$

$\text{gain}(\text{Humidity}) = 0.152 \text{ bits}$

$\text{gain}(\text{Windy}) = 0.048 \text{ bits}$

- higher IG means that after the split, the decisions are more deterministic
- purest subnodes will result in a higher IG.

Computing Information

Information is measured in **bits**

- given a probability distribution, the info required to encode/predict an event.
- **entropy** gives the information required in bits (this can involve fractions of bits!)

Formula for computing the information entropy:

$$\boxed{\text{entropy}(p_1, p_2, \ldots, p_n) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 \ldots - p_n \log_2 p_n}$$

Note: In the exercises, we will use the **natural logarithm** for convenience (easier with calculators in the exam), which turns „bits“ into „nits“

Expected Information Gain

$$\text{gain}(S,a) = \text{entropy}(S) - \sum_{v \in Values(a)} \frac{|S_v|}{|S|}\,\text{entropy}(S_v)$$

$$S_v = \{s \in S : a(s) = v\}$$

All possible values for attribute a

$\text{gain}(S,a)$ is the information gained adding a sub-tree

(Reduction in number of bits needed to classify an instance)

Problems?

Wish List for a Purity Measure

Properties we require from a purity measure:

only one output in the data

50/50

- When node is pure, measure should be zero (=0)
- When impurity is maximal (i.e. all classes equally likely), measure should be maximal (e.g., 1 for boolean values)
- Multistage property: info[2,3,4]=info[2,7]+7/9 info[3,4]

Entropy is a function that satisfies these properties

Number of classes

$$\boxed{entropy(S) = \sum_{i=1}^{n} -p_i \log_2 p_i}$$

(scales from 0 to max $\log_2 n$)

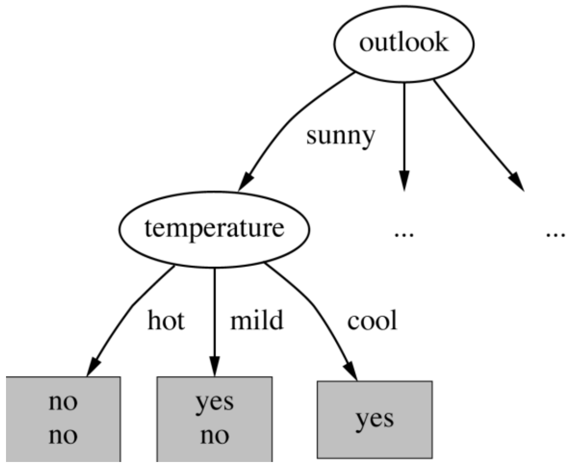
Training data (instances)

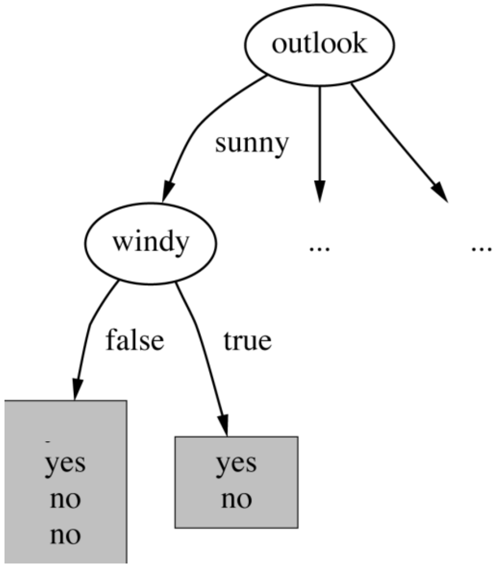
Probability of *S being* classified to *i*

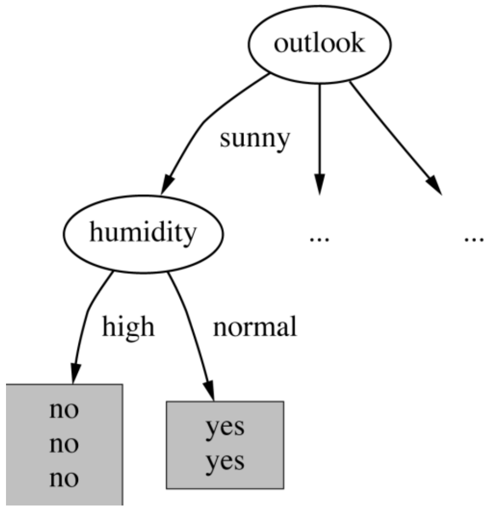
The Weather Data

Outlook	**Temp**	**Humidity**	**Windy**	**Play**
Sunny	Hot	High	False	**No**
Sunny	Hot	High	True	**No**
Overcast	Hot	High	False	**Yes**
Rainy	Mild	High	False	**Yes**
Rainy	Cool	Normal	False	**Yes**
Rainy	Cool	Normal	True	**No**
Overcast	Cool	Normal	True	**Yes**
Sunny	Mild	High	False	**No**
Sunny	Cool	Normal	False	**Yes**
Rainy	Mild	Normal	False	**Yes**
Sunny	Mild	Normal	True	**Yes**
Overcast	Mild	High	True	**Yes**
Overcast	Hot	Normal	False	**Yes**
Rainy	Mild	High	True	**No**

Continuing to Split





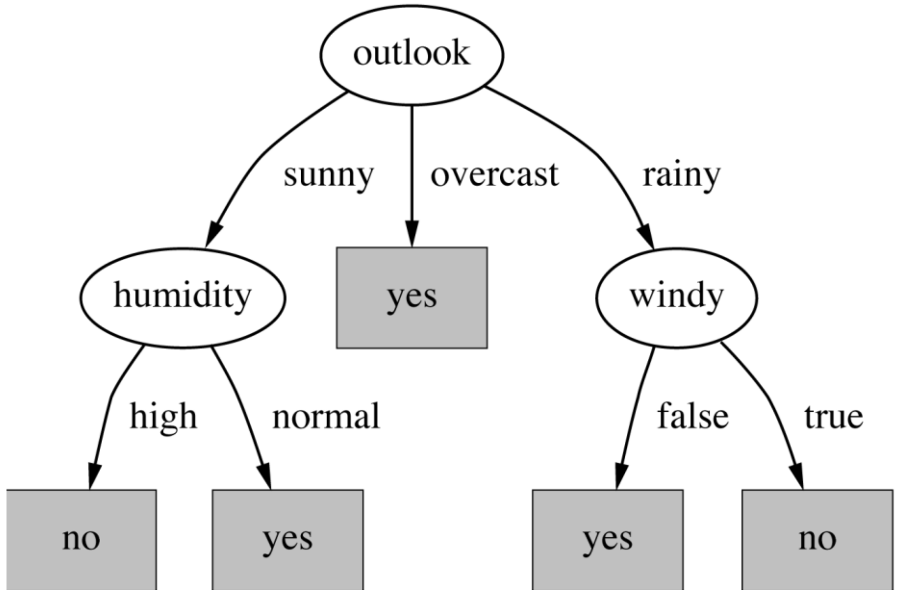


$\text{gain}(\text{Temperatur}) = \underline{0.571}\ \text{bits}$

$\text{gain}(\text{Humidity}) = \underline{0.971}\ \text{bits}$

$\text{gain}(\text{Windy}) = \underline{0.020}\ \text{bits}$

The Final Decision Tree



Note: not all leaves need to be pure;

→ Splitting stops when data can't be split any further

Claude Shannon

Born: 30 April 1916, Died: 23 February 2001

Shannon is famous for having founded **information theory** with one landmark paper published in 1948 (*A Mathematical Theory of Communication*).

Information theory was developed to find fundamental limits on compressing and reliably communicating data. Communications over a channel was the primary motivation. Channels (such as a phone line) often fail to produce exact reconstruction of a signal; noise, periods of silence, and other forms of signal corruption often degrade quality. How much information can one hope to communicate over a noisy (or otherwise imperfect) channel?

An important application of information theory is coding theory:

- Data compression (by removing redundancy in data)
- Error-correcting codes add just the right kind of redundancy (i.e. error correction) needed to transmit the data efficiently and faithfully across a noisy channel.

Background on Entropy and Information Theory

- Suppose there are n possible states or messages.
- If the messages are equally likely, then the probability of each message is $p = 1/n$ or $n = 1/p$
- Information (number of bits) of a message is described as

$$\log_2(n) = \log_2(1/p) = -\log_2(p)$$

- Example: with 16 possible messages, the information is $\log(16) = 4$ and we require 4 bits for each message.
- If the following probability distribution is given:

$$P = (p_1, p_2, .., p_n)$$

then the information (or entropy of P) conveyed by the distribution can be computed as follows:

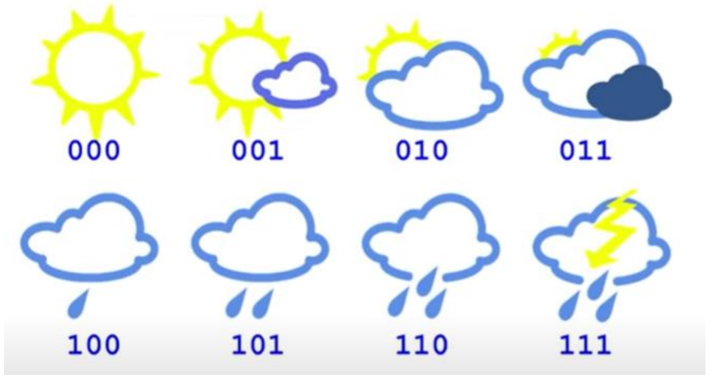
info(...) =

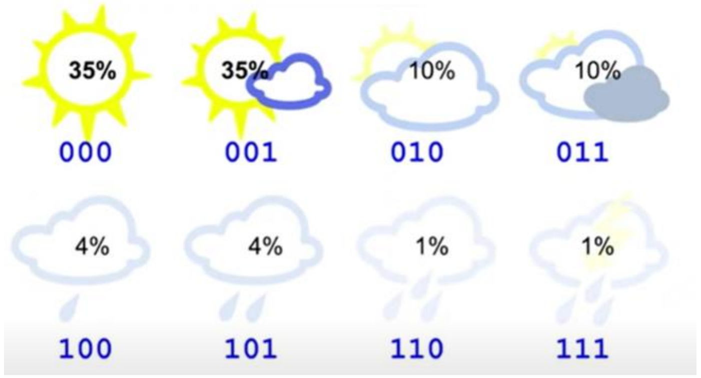
$$\boxed{\begin{aligned} I(P) &= -(p_1 * \log(p_1) + p_2 * \log(p_2) + .. + p_n * \log(p_n)) \\ &= -\textstyle\sum_i p_i * \log(p_i) = \sum_i p_i * \log(1/p_i) \end{aligned}}$$

Entropy = info (P)

- Example: P = 75% sun and 25% rain
  $\boxed{I(P) = -(p_1 * \log(p_1) + p_2 * \log(p_2))} = -(0{,}75 * \log(0{,}75) - 0{,}25 * \log(0{,}25)) =$ 0,81 bits

- Example: 8 equally likely states $I(P) = -\sum_i p_i * \log(p_i) = -8 * (\frac{1}{8} * -3) =$ 3 bits





less bits because they are more "organized"

Example: 8 states that are not equally likely
$I(P) = -\sum_i q_i * \log(q_i) = -(0.35 * \log(0.35) + \ldots + 0.01 * \log(0.01)) =$ 2.23 bits

Detour: Cross-Entropy

Suppose, you have a classifier that produces a discrete probability distribution, and you have the true underlying distribution. For example, the probabilities for high, medium, and low risk of a customer might be:

Outcome distribution q	Ground truth p
0.7	1
0.2	0
0.1	0

The **cross-entropy** between the two distributions p and q is used to quantify the difference between these two distributions.

$$\boxed{H(p,q) = -\sum_i p_i * \log(q_i)}$$

Example: $H(p,q) = -1 * \log(0.7) = 0{,}5146$

The cross-entropy is not symmetric!

The lower the cross-entropy, the higher the probability that an event will happen.

Detour: KL-Divergence (or Relative Entropy)

The **Kullback-Leibler-Divergence** between two distributions p and q is a measure of how one probability distribution is different from another. If KL-divergence is 0, there is no difference.

cross-entropy = entropy + KL-divergence

$$H(p,q) = -\sum_i p_i * \log(q_i)$$

KL-divergence is defined as the difference between the cross entropy and entropy:

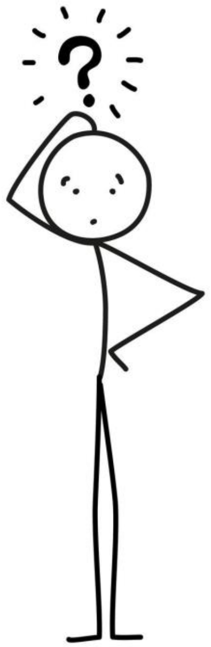
$$\boxed{\text{KL-divergence = cross-entropy – entropy}}$$

$$D_{KL}(p,q) = H(p,q) - H(p)$$

$$= -\sum_i p_i * (\log(q_i) - \log(p_i)) = -\sum_i p_i * (\log(q_i/p_i))) = \sum_i p_i * (\log(p_i/q_i))$$

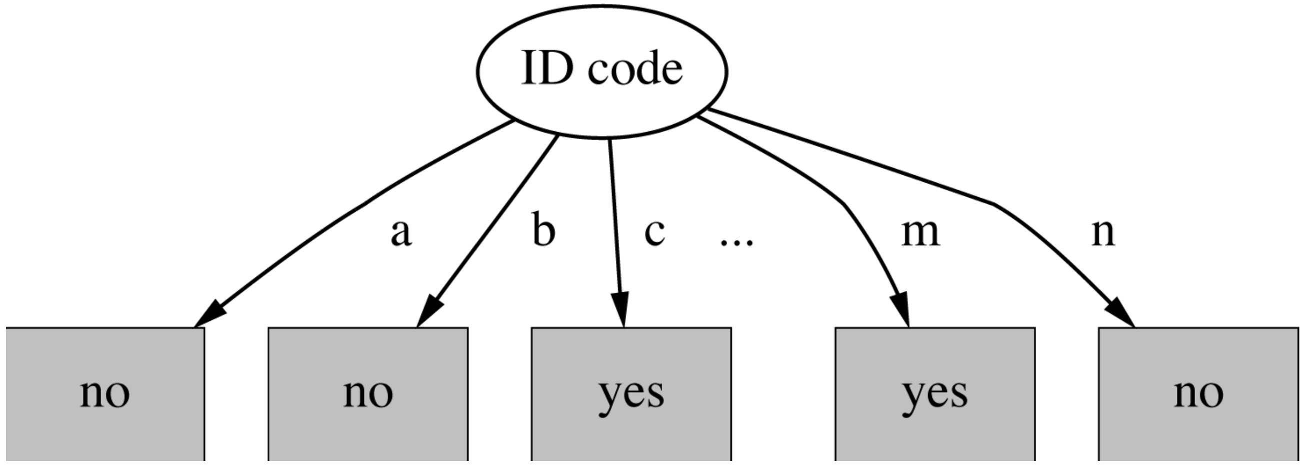
Both, cross-entropy and KL-divergence are used as loss function in machine learning.

Which splitting attribute would be selected in the following example?



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**

Split for ID Code Attribute



Entropy of split = 0 (since each leaf node is "pure"), having only one case.

Information gain is maximal for ID code.

but is this good

Highly-Branching Attributes

Problematic: attributes with a large number of values
(extreme case: ID code)

Subsets are more likely to be pure if there is a large number of values.

- Information gain is biased towards choosing attributes with a large number of values.
- This may result in overfitting (selection of an attribute that is non-optimal for prediction).

Gain Ratio

Gain ratio: a modification of the information gain that reduces its bias on high-branch attributes.

Gain ratio takes number and size of branches into account when choosing an attribute.

It corrects the information gain by taking the intrinsic information of a split into account (i.e. how much info do we need to tell which branch an instance belongs to).

Computing the Gain Ratio

Example: intrinsic information for ID code

$$\text{intrinsic\_info}(1,1,\ldots,1) = 14 * \left(-\frac{1}{14} * \log_2\left(\frac{1}{14}\right)\right) = 3.807 \text{ bits}$$

↳ same as info([1,1,…])

Importance of attribute decreases as intrinsic information grows.

Example of gain ratio: $\boxed{\text{gainRatio}(S,a) = \frac{\text{gain(S,a)}}{\text{intrinsic\_info(S,a)}}}$

Example: $\text{gainRatio}(\text{ID\_Code}) = \frac{0.940\text{bits}}{3.807\text{bits}} = 0.246$ ← penalty for many branches

Gain Ratios for Weather Data

Outlook		**Temperature**	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	$\underline{0.247}$	Gain: 0.940-0.911	$\underline{0.029}$
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.557
Gain ratio: 0.247/1.577	$\underline{0.156}$	Gain ratio: 0.029/1.557	$\underline{0.019}$

Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	$\underline{0.152}$	Gain: 0.940-0.892	$\underline{0.048}$
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	$\underline{0.152}$	Gain ratio: 0.048/0.985	$\underline{0.049}$

More on the Gain Ratio

„Outlook” still comes out top.

However:

- “ID code” has still greater gain ratio (0.246).
- Standard fix: ad hoc test to prevent splitting on that type of attribute.

Problem with gain ratio: it may overcompensate.

- May choose an attribute just because its intrinsic information is very low.
- Standard fix:
  - First, only consider attributes with greater than average information gain.
  - Then, compare them on gain ratio.

The Splitting Criterion in CART

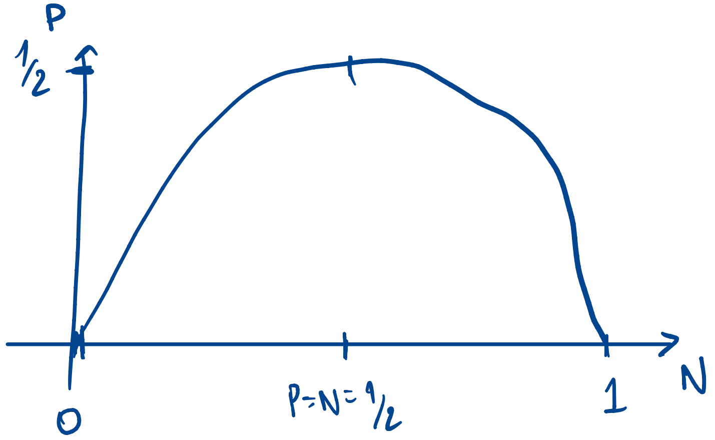
- Classification and Regression Tree (CART)
- developed 1974-1984 by 4 statistics professors
  - Leo Breiman (Berkeley), Jerry Friedman (Stanford), Charles Stone (Berkeley), Richard Olshen (Stanford)
- Gini Index is used as a splitting criterion
- both C4.5 and CART are robust tools
- no method is always superior – experiment!

Gini Index for 2 Attribute Values

For example, two classes, **Pos** and **Neg**, and dataset S with p **Pos**-elements and n **Neg**-elements. The frequency of positives and negative classes is:

$$P = p / (p+n)$$
$$N = n / (p+n)$$

$$\boxed{Gini(S) = 1 - P^2 - N^2 \quad \in [0,0.5]}$$



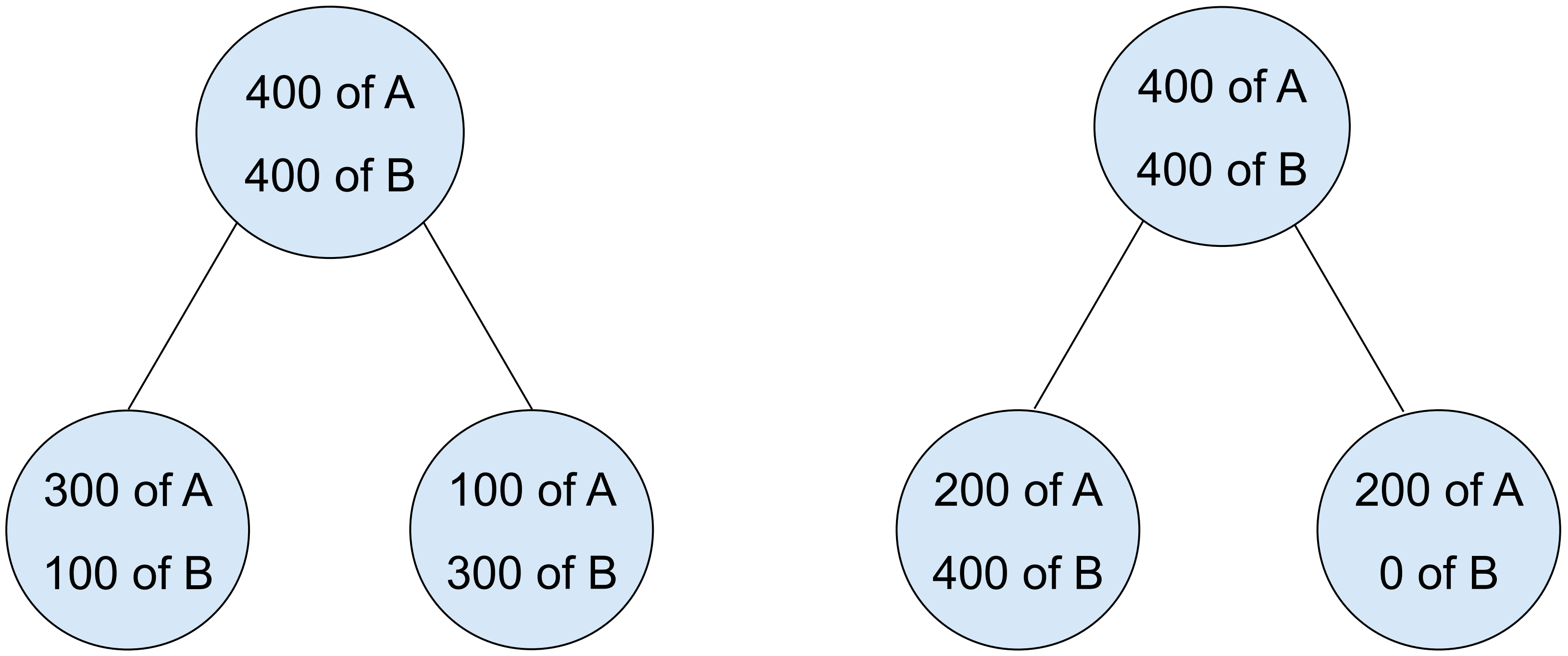
If dataset S is split into S_1, S_2 then

$$\boxed{Gini_{split}(S_1,S_2) = (p_1+n_1)/(p+n) \cdot Gini(S_1) + (p_2+n_2)/(p+n) \cdot Gini(S_2)}$$

↳ measures the impurity of a node → <u>lower</u> Gini = <u>better</u>

Example

Split by attribute I or II?



Example

$$\mathrm{Gini}(p) = 1 - \sum_j p_j^2$$

Numbers of Cases		Proportion of Cases				Gini Index
A	B	A	B			
		p_A	p_B	p^2_A	p^2_B	$1- p^2_A - p^2_B$
400	400	0.5	0.5	0.25	0.25	0.5

Select the split that decreases the Gini Index most. This is done over all possible places for a split and all possible variables to split.

Gini Index Example

Number of Cases		Proportion of Cases				Gini Index	Info required	
A	B	A	B					
		p_A	p_B	p^2_A	p^2_B	$1- p^2_A - p^2_B$		
300	100	0.75	0.25	0.5625	0.0625	0.375	0.1875	0.5*Gini(i)
100	300	0.25	0.75	0.0625	0.5625	0.375	0.1875	
						Total	0.375	
200	400	0.33	0.67	0.1111	0.4444	0.4444	0.3333	0.75*Gini(i)
200	0	1	0	1	0	0	0	
						Total	0.3333	

Generate_DT(samples, attribute_list)

- Create a new node N.
- If samples are all of class C then label N with C and exit.
- If attribute_list is empty then label N with majority_class(samples) and exit.
- Select best_split from attribute_list.
- For each value v of attribute best_split:
  - Let S_v = set of samples with best_split=v.
  - Let N_v = **Generate_DT**(S_v, attribute_list \ best_split).
  - Create a branch from N to N_v labeled with the test best_split=v.

Time Complexity of Basic Algorithm

- Let m be the number of attributes.
- Let n be the number of instances.
- Assumption: Depth of tree is $\mathcal{O}(\log n)$.
  - usual assumption if the tree is not degenerate
- For each level of the tree all n instances are considered (best = $\mathrm{v_i}$).
  - $\mathcal{O}(n \log n)$ work for a single attribute over the entire tree
- Total cost is $\mathcal{O}(mn \log n)$ since all attributes are eventually considered.
  - without pruning (see next class)

Scalability of DT Algorithms

- need to design for large amounts of data

- two things to worry about:
  - large number of attributes
    - leads to a large tree
    - takes a long time

  - large amounts of data
    - Can the data be kept in memory?
    - some new algorithms do not require all the data to be memory resident

C4.5 History

The above procedure is the basis for Ross Quinlain's ID3 algorithm (so far works only for nominal attributes).

- ID3, CHAID – 1960s

The algorithm was improved and is now most widely used as C4.5 or C5.0 respectively, available in most DM software packages.

- Commercial successor: C5.0

Witten et al. write "a landmark decision tree program that is probably the machine learning workhorse most widely used in practice to date".

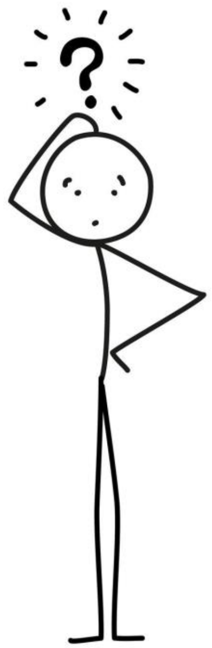
C4.5 An Industrial-Strength Algorithm

For an algorithm to be useful in a wide range of real-world applications it must:

- permit numeric attributes
- allow missing values
- be robust in the presence of noise (prunning) $\longrightarrow$ important when we want to optimize accuracy.

Basic algorithm needs to be extended to fulfill these requirements.

How would you deal with missing values in the training data?



Outline for today

- Choosing a splitting attribute in decision trees
  - Information gain
  - Gain ratio
  - Gini index
- **Numeric attributes**
- Missing values
- Relational rules

Numeric Attributes

Unlike nominal attributes, every attribute has many possible split points.

- Standard method: binary splits
- E.g. temp < 45

Solution is straightforward extension:

- Evaluate info gain (or other measure)
  for every possible split point of attribute
- Choose "best" split point
- Info gain for best split point is highest info gain for attribute

Numerical attributes can be used several times in a decision tree, nominal attributes only once.

Example

usually between an "yes" and a "no"

Split on temperature attribute:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

- e.g. temperature $<$ 71.5: yes/4, no/2
  temperature $\geq$ 71.5: yes/5, no/3
- Info([4,2],[5,3])
  = 6/14 info([4,2]) + 8/14 info([5,3])
  = 0.939 bits ↶ we want a small Info which will lead to higher IG.

Place split points halfway between values.

Binary Splits on Numeric Attributes

Splitting (multi-way) on a nominal attribute exhausts all information in that attribute.

- nominal attribute is tested (at most) once on any path in the tree

Not so for binary splits on numeric attributes!

- numeric attributes may be tested several times along a path in the tree

Disadvantage: tree is hard to read.

Remedy:

- pre-discretize numeric attributes, or
- allow for multi-way splits instead of binary ones using the Information Gain criterion

Outline for today

- Choosing a splitting attribute in decision trees
  - Information gain
  - Gain ratio
  - Gini index
- Numeric attributes
- **Missing values**
- Relational rules

Handling Missing Values / Training

→ Ignore instances with missing values.

- pretty harsh, and missing value might not be important

→ Ignore attributes with missing values.

- again, may not be feasible

→ Treat missing value as another nominal value.

- fine if missing a value has a significant meaning

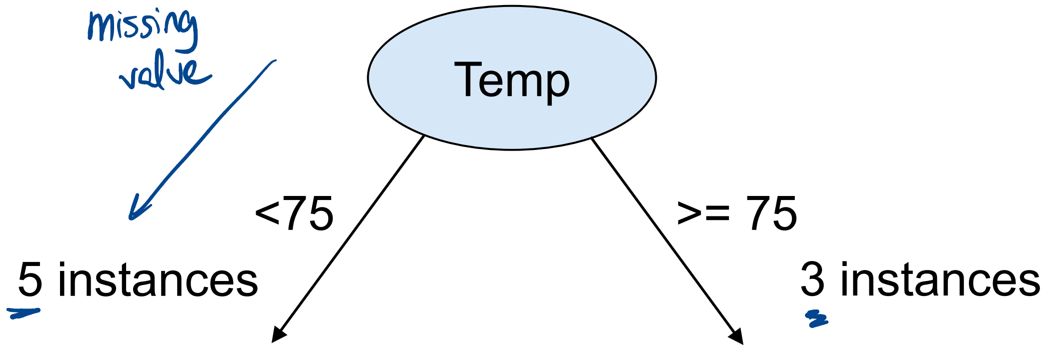
→ Estimate missing values.

- data imputation: regression, nearest neighbor, mean, mode, etc.

Handling Missing Values / Classification

Follow the leader.

- an instance with a missing value for a tested attribute (temp) is sent down the branch with the most instances

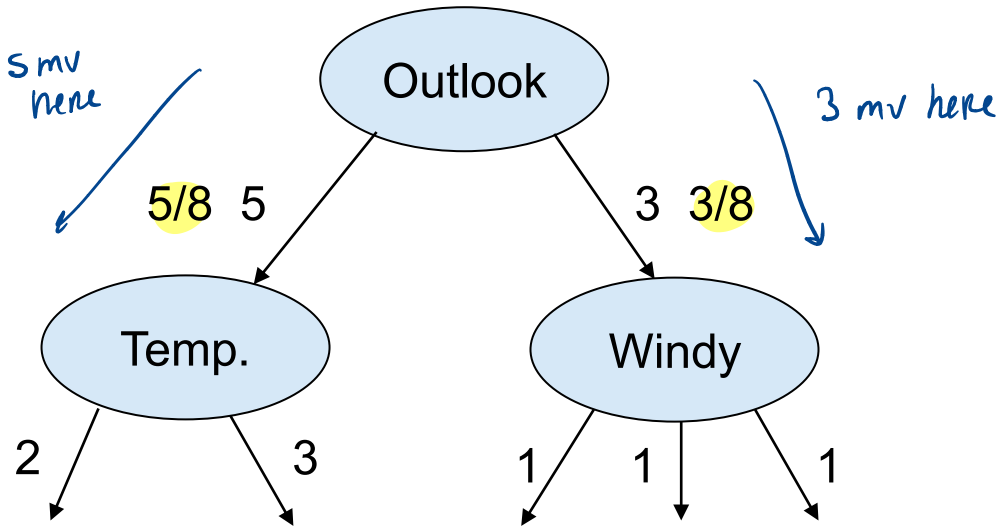


Instance included on the left branch

Handling Missing Values / Classification

"Partition" the instance.

- branches show # of instances
- Send down parts of the instance (e.g. 3/8 on Windy and 5/8 on Sunny) proportional to the number of training instances
- Resulting leaf nodes get weighted in the result



Overfitting

Two sources of abnormalities
- noise (randomness)
- outliers (measurement errors)

Chasing every abnormality causes overfitting
- decision tree gets too large and complex
- good accuracy on training set, poor accuracy on test set
- does not generalize to new data any more

Solution: prune the tree

↑ remove variables

Decision Trees - Summary

Decision trees are a classification technique.

The output of decision trees can be used for descriptive as well as predictive purposes.

They can represent any function in the form of propositional logic.

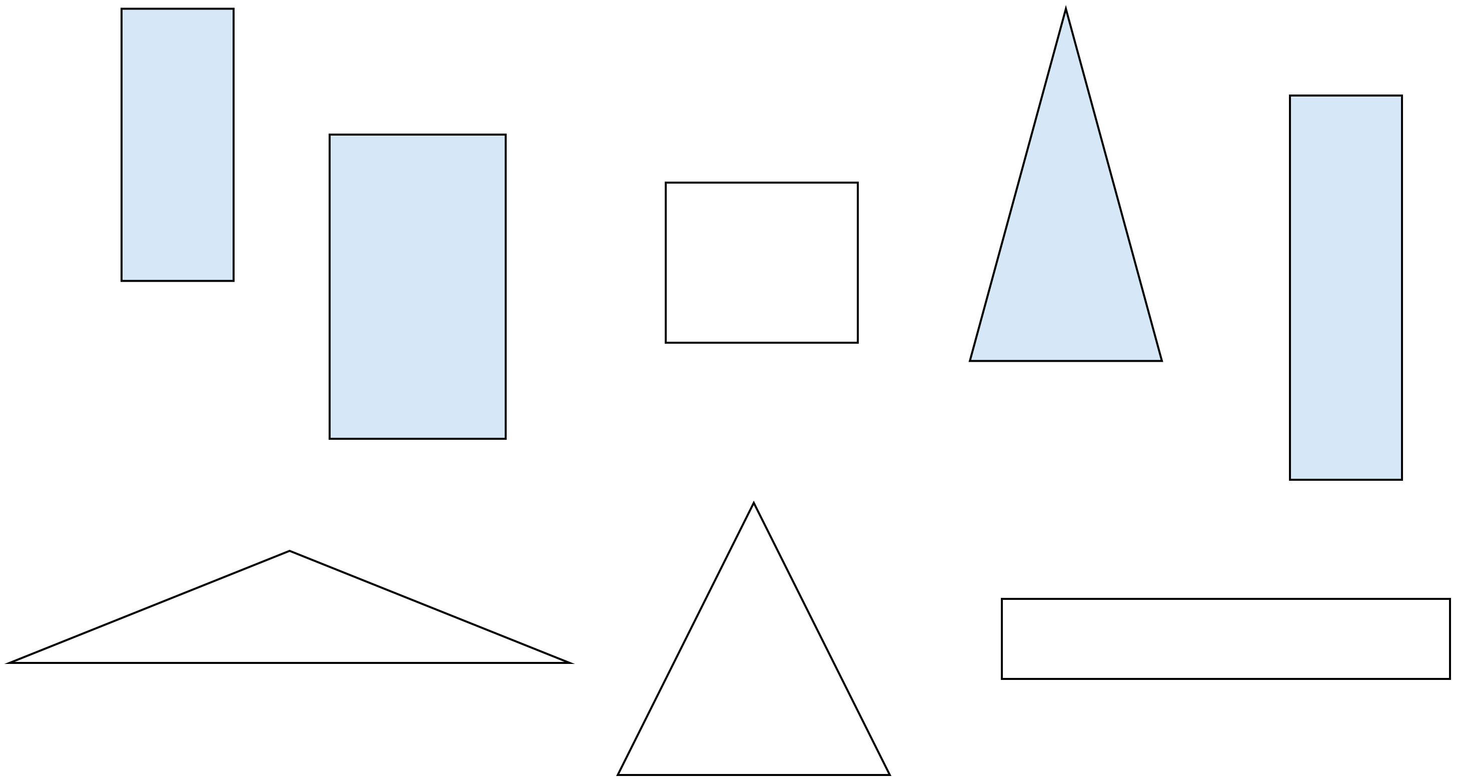
Heuristics such as information gain are used to select relevant attributes.

Pruning is used to avoid overfitting.

Outline for today

- Choosing a splitting attribute in decision trees
  - Information gain
  - Gain ratio
  - Gini index
- Numeric attributes
- Missing values
- **Relational rules**

The Shapes Problem



Shaded=standing
Unshaded=lying

Instances

Width	Height	Sides	Class
2	4	4	**Standing**
3	6	4	**Standing**
4	3	4	**Lying**
7	8	3	**Standing**
7	6	3	**Lying**
2	9	4	**Standing**
9	1	4	**Lying**
10	2	3	**Lying**

```
If width ≥ 3.5 and height < 7.0  then lying.
If height ≥ 3.5 then standing.
```

Classification Rules

```
If width ≥ 3.5 and height < 7.0  then lying.
If height ≥ 3.5 then standing.
```

Work well to classify these instances ... but not necessarily for new ones.

Problems?

Relational Rules

```
If width > height then lying
If height > width then standing
```

Rules comparing attributes to constants are called propositional rules (propositional data mining).

Relational rules are more expressive in some cases.
- define relations between attributes (relational data mining)
- most DM techniques do not consider relational rules

As a workaround for some cases, one can introduce additional attributes, describing if width > height.
- allows using conventional “propositional” learners

Propositional Logic

Essentially, decision trees can represent any function in propositional logic.

- A, B, C: propositional variables
- and, or, not, => (implies), <=> (equivalent): connectives

A proposition is a statement that is either true or false.

- The sky is blue: color of sky = blue

Decision trees are an example of a propositional learner.