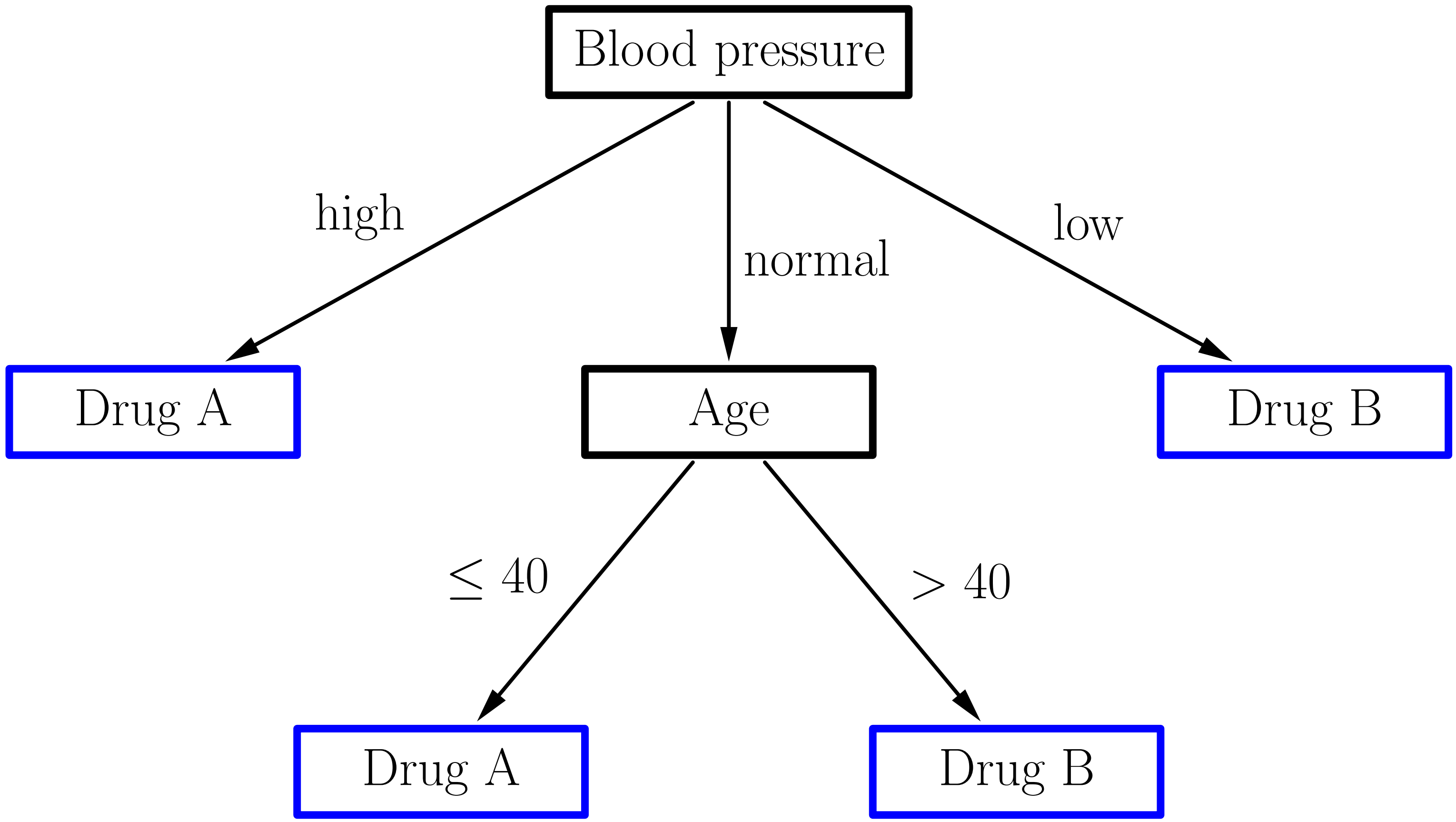
Excursus: Learning Decision Trees

Assignment of a drug to a patient:



Classification with a Decision Tree

Recursive Descent:

Start at the root node.

If the current node is an **leaf node**:

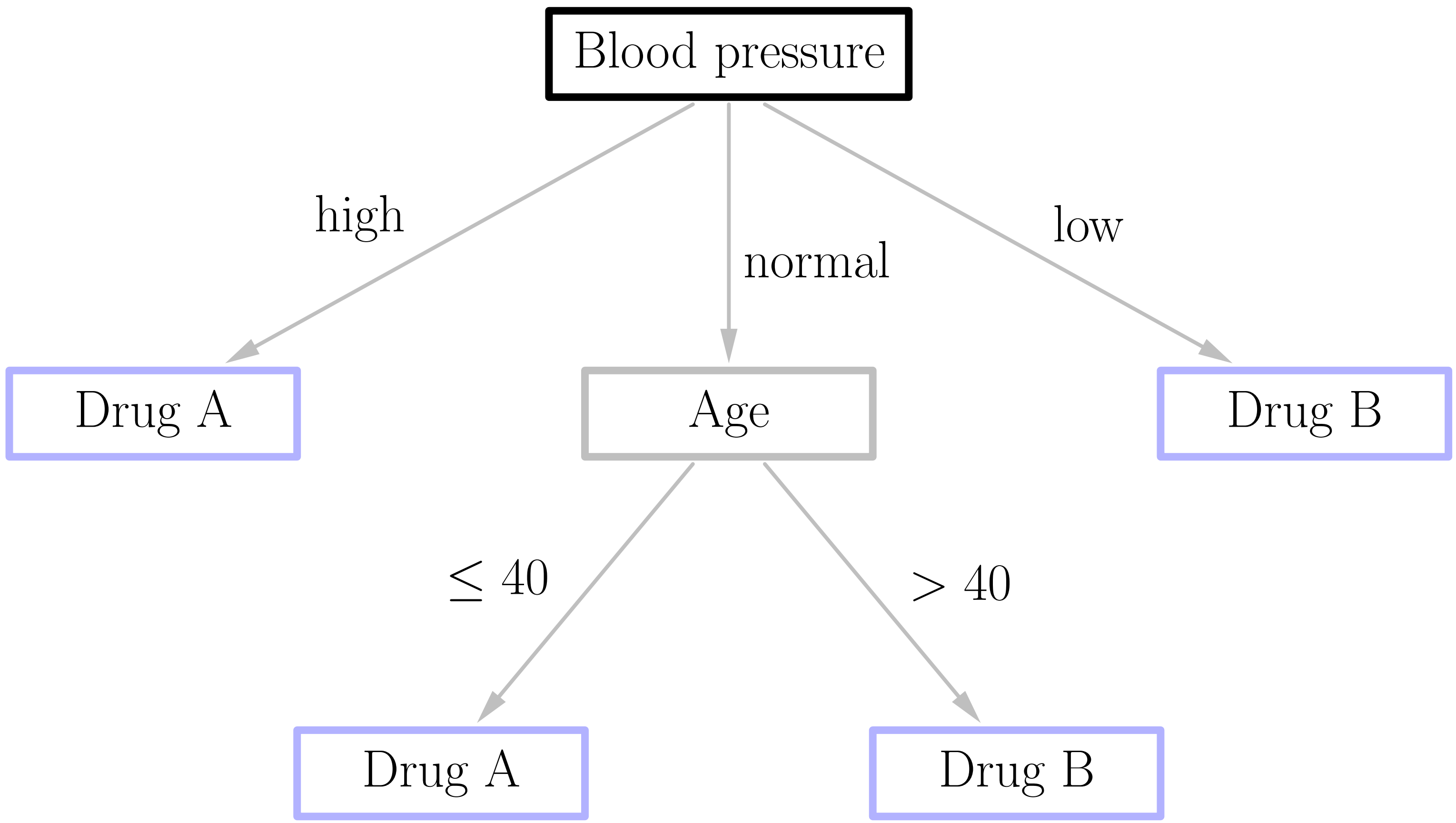
- Return the class assigned to the node.

If the current node is an **inner node**:

- Test the attribute associated with the node.
- Follow the branch labeled with the outcome of the test.
- Apply the algorithm recursively.

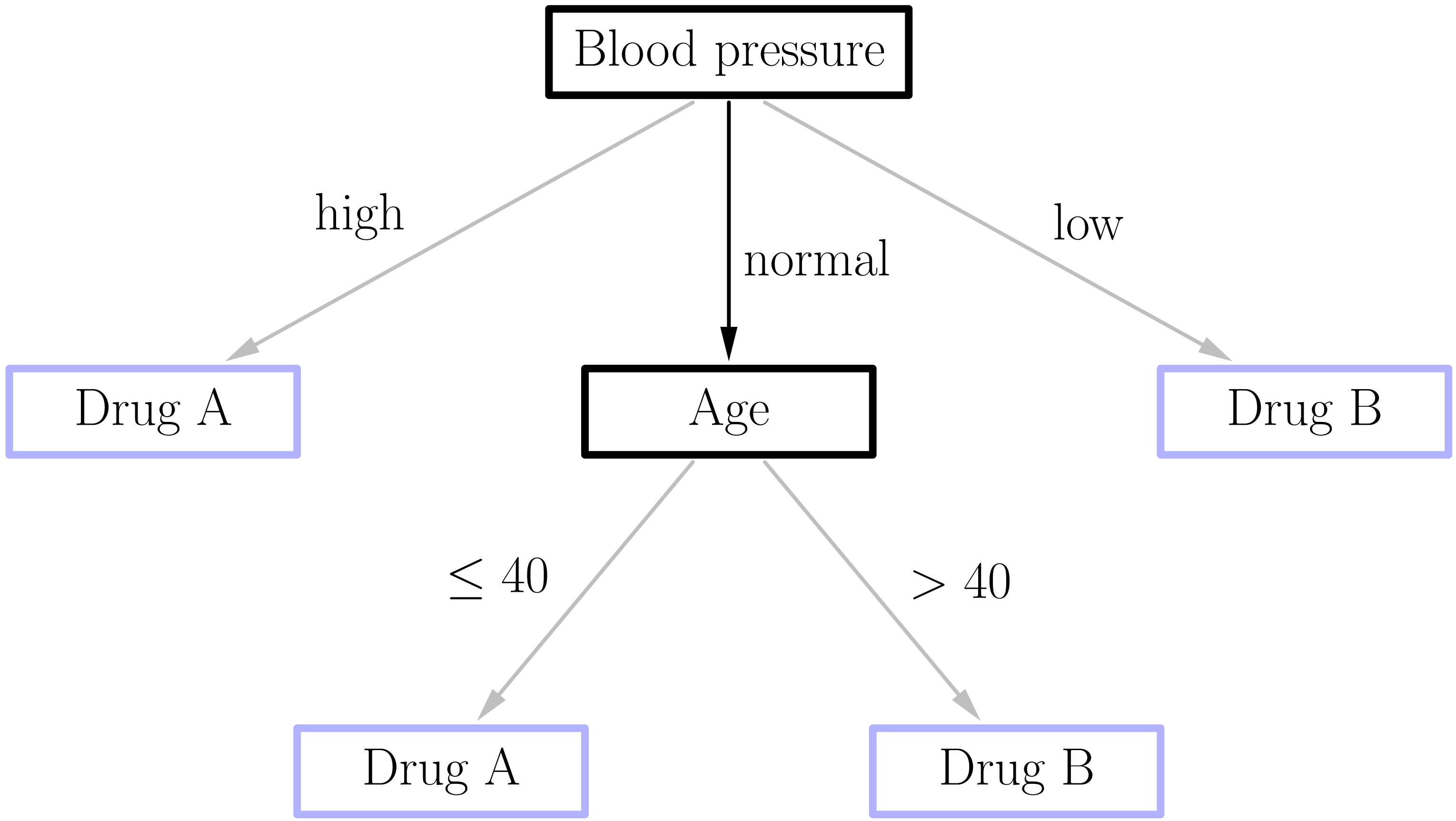
Intuitively: Follow the path corresponding to the case to be classified.

Assignment of a drug to a patient:

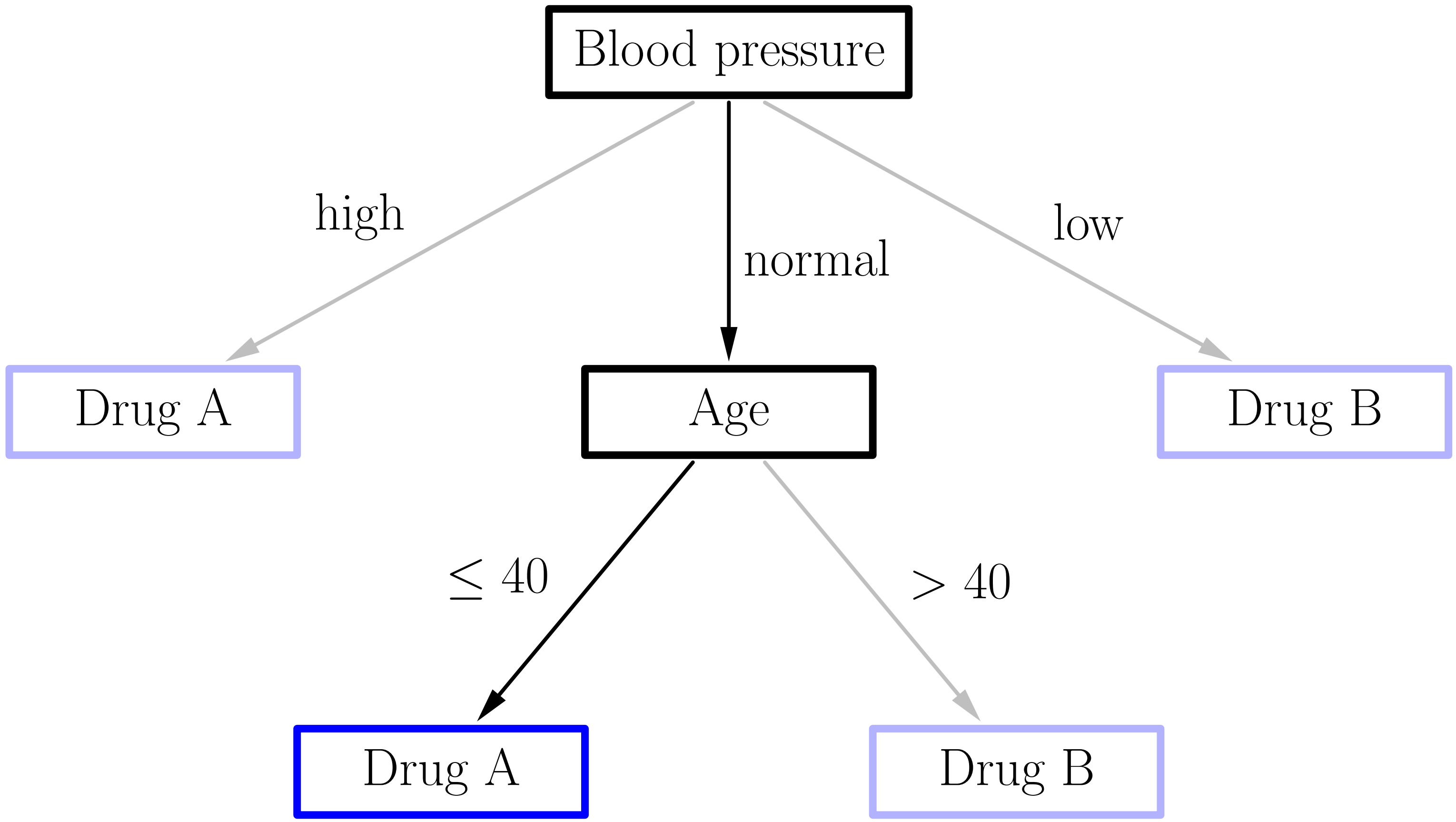


Classification in the Example

Assignment of a drug to a patient:



Assignment of a drug to a patient:



Top-down approach

- Build the decision tree from top to bottom (from the root to the leaves).

Greedy Selection of a Test Attribute

- Compute an evaluation measure for all attributes.
- Select the attribute with the best evaluation.

Divide and Conquer / Recursive Descent

- Divide the example cases according to the values of the test attribute.
- Apply the procedure recursively to the subsets.
- Terminate the recursion if
  - all cases belong to the same class
  - no more test attributes are available

Induction of a Decision Tree: Example

Patient database

12 example cases

3 descriptive attributes

1 class attribute

No	Sex	Age	Blood pr.	Drug
1	male	20	normal	A
2	female	73	normal	B
3	female	37	high	A
4	male	33	low	B
5	female	48	high	A
6	male	29	normal	A
7	female	52	normal	B
8	male	42	low	B
9	male	61	normal	B
10	female	30	normal	A
11	female	26	low	B
12	male	54	high	A

Assignment of drug

(without patient attributes)

always drug A or always drug B:

50% correct (in 6 of 12 cases)

Sex of the patient

Division w.r.t. male/female.

No	Sex	Drug
1	male	A
6	male	A
12	male	A
4	male	B
8	male	B
9	male	B
3	female	A
5	female	A
10	female	A
2	female	B
7	female	B
11	female	B

Assignment of drug

male:	50% correct	(in 3 of 6 cases)
female:	50% correct	(in 3 of 6 cases)
total:	**50% correct**	(in 6 of 12 cases)

Induction of a Decision Tree: Example

Age of the patient

Sort according to age.

Find best age split.
here: ca. 40 years

No	Age	Drug
1	20	A
11	26	B
6	29	A
10	30	A
4	33	B
3	37	A
8	42	B
5	48	A
7	52	B
12	54	A
9	61	B
2	73	B

Assignment of drug

≤ 40:	A	67% correct	(in 4 of 6 cases)
> 40:	B	67% correct	(in 4 of 6 cases)
total:		**67% correct**	(in 8 of 12 cases)

Induction of a Decision Tree: Example

Blood pressure of the patient

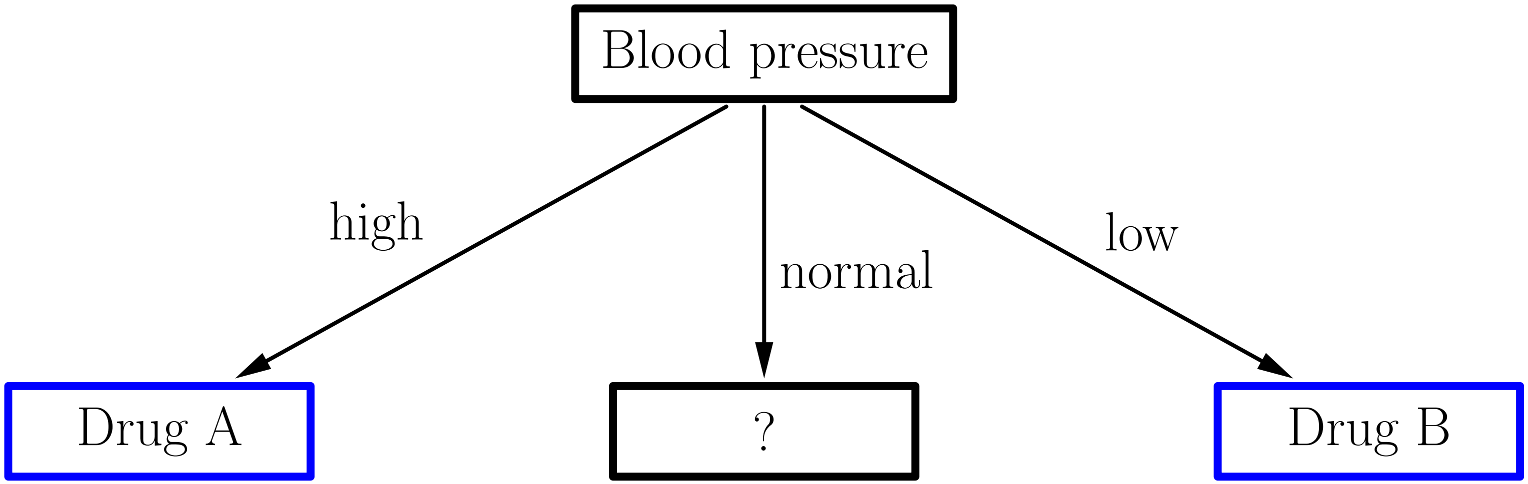
Division w.r.t. high/normal/low.

Assignment of drug

high:	A	100% correct	(in 3 of 3 cases)
normal:		50% correct	(in 3 of 6 cases)
low:	B	100% correct	(in 3 of 3 cases)
total:		**75% correct**	(in 9 of 12 cases)

No	Blood pr.	Drug
3	high	A
5	high	A
12	high	A
1	normal	A
6	normal	A
10	normal	A
2	normal	B
7	normal	B
9	normal	B
4	low	B
8	low	B
11	low	B

Current Decision Tree:



Induction of a Decision Tree: Example

Blood pressure and sex

Only patients
with normal blood pressure.

Division w.r.t. male/female.

Assignment of drug

male:	A	67% correct	(2 of 3)
female:	B	67% correct	(2 of 3)
total:		**67% correct**	(4 of 6)

No	Blood pr.	Sex	Drug
3	high		A
5	high		A
12	high		A
1	normal	male	A
6	normal	male	A
9	normal	male	B
2	normal	female	B
7	normal	female	B
10	normal	female	A
4	low		B
8	low		B
11	low		B

Blood pressure and age

Only patients
with normal blood pressure.

Sort according to age.

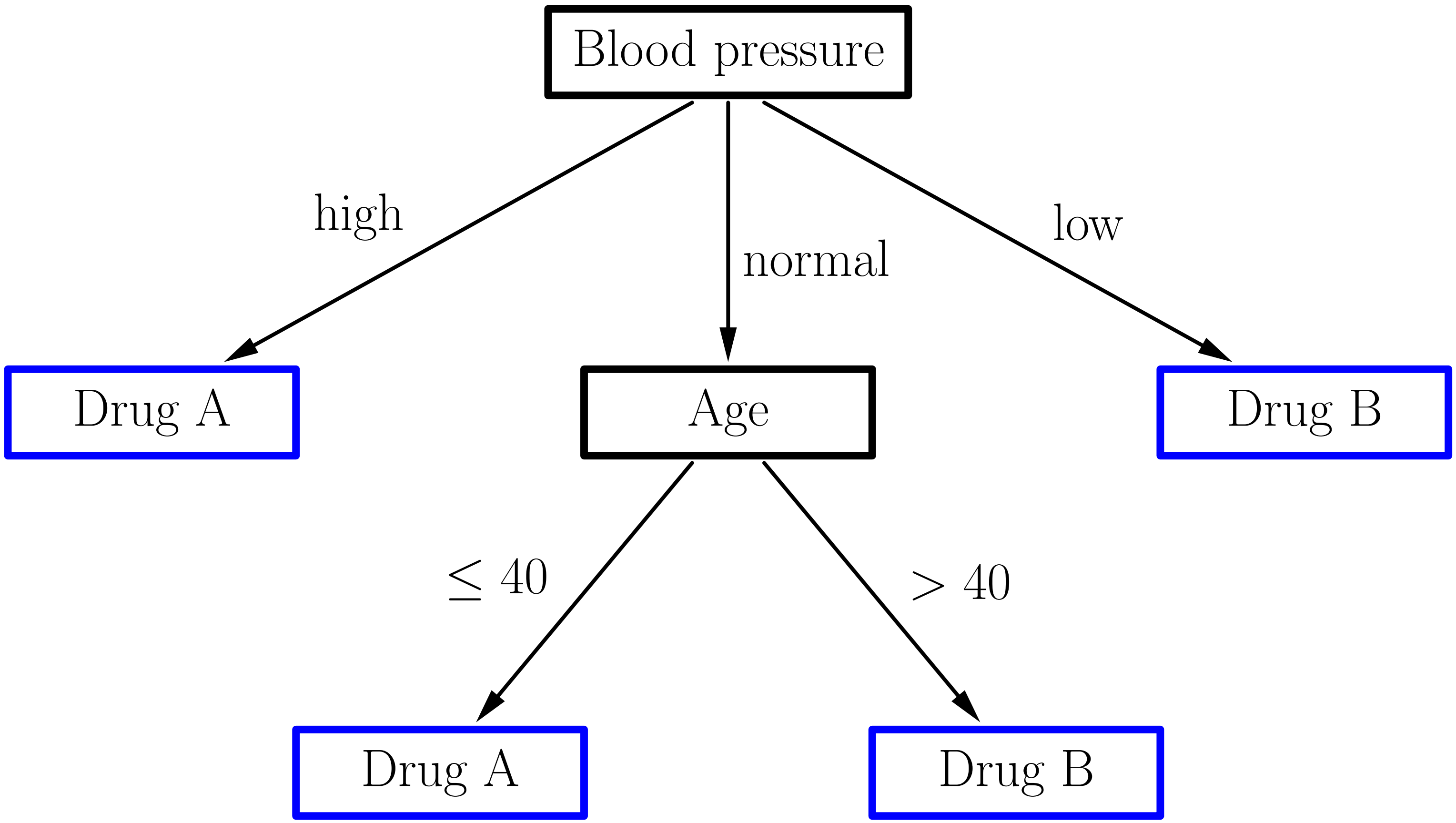
Find best age split.
here: ca. 40 years

Assignment of drug

≤ 40:	A	100% correct	(3 of 3)
> 40:	B	100% correct	(3 of 3)
total:		**100% correct**	(6 of 6)

No	Blood pr.	Age	Drug
3	high		A
5	high		A
12	high		A
1	normal	20	A
6	normal	29	A
10	normal	30	A
7	normal	52	B
9	normal	61	B
2	normal	73	B
11	low		B
4	low		B
8	low		B

Assignment of a drug to a patient:



Decision Tree Induction: Notation

S	a set of case or object descriptions
C	the class attribute
$A^{(1)}, \ldots, A^{(m)}$	other attributes (index dropped in the following)
$\mathrm{dom}(C)$	$= \{c_1, \ldots, c_{n_C}\}$, n_C: number of classes
$\mathrm{dom}(A)$	$= \{a_1, \ldots, a_{n_A}\}$, n_A: number of attribute values
$N_{..}$	total number of case or object descriptions i.e. $N_{..} = \vert S \vert$
$N_{i.}$	absolute frequency of the class c_i
$N_{.j}$	absolute frequency of the attribute value a_j
N_{ij}	absolute frequency of the combination of the class c_i and the attribute value a_j. It is $N_{i.} = \sum_{j=1}^{n_A} N_{ij}$ and $N_{.j} = \sum_{i=1}^{n_C} N_{ij}$.
$p_{i.}$	relative frequency of the class c_i, $p_{i.} = \frac{N_{i.}}{N_{..}}$
$p_{.j}$	relative frequency of the attribute value a_j, $p_{.j} = \frac{N_{.j}}{N_{..}}$
p_{ij}	relative frequency of the combination of class c_i and attribute value a_j, $p_{ij} = \frac{N_{ij}}{N_{..}}$
$p_{i\vert j}$	relative frequency of the class c_i in cases having attribute value a_j, $p_{i\vert j} = \frac{N_{ij}}{N_{.j}} = \frac{p_{ij}}{p_{.j}}$

Decision Tree Induction: General Algorithm

function grow_tree (S : set of cases) : node;
begin
  $best\_v$:= WORTHLESS;
  for all untested attributes A **do**
    compute frequencies N_{ij}, $N_{i.}$, $N_{.j}$ for $1 \leq i \leq n_C$ and $1 \leq j \leq n_A$;
    compute value v of an evaluation measure using N_{ij}, $N_{i.}$, $N_{.j}$;
    if $v > best\_v$ **then** $best\_v := v$; $best\_A := A$; **end**;
  end
  if $best\_v$ = WORTHLESS
  then create leaf node x;
    assign majority class of S to x;
  else create test node x;
    assign test on attribute $best\_A$ to x;
    for all $a \in \mathrm{dom}(best\_A)$ **do** x.child$[a]$:= grow_tree$(S|_{best_A=a})$; **end**;
  end;
  return x;
end;

Evaluation Measures

Evaluation measure used in the above example:
rate of correctly classified example cases.

- Advantage: simple to compute, easy to understand.
- Disadvantage: works well only for two classes.

If there are more than two classes, the rate of misclassified example cases **neglects a lot of the available information**.

- Only the majority class—that is, the class occurring most often in (a subset of) the example cases—is really considered.
- The distribution of the other classes has no influence. However, a good choice here can be important for deeper levels of the decision tree.

Therefore: Study also other evaluation measures. Here:

- **Information gain** and its various normalizations.
- χ^2 **measure** (well-known in statistics).

Excursus: Shannon Entropy

Let X be a random variable with domain $\mathrm{dom}(X) = \{x_1, \ldots, x_n\}$. Then,

$$H^{(\text{Shannon})}(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

is called the **Shannon entropy** of (the probability distribution of) X, where $0 \cdot \log_2 0 = 0$ is assumed.

Intuitively: **Expected number of yes/no questions that have to be asked in order to determine the obtaining value of X.**

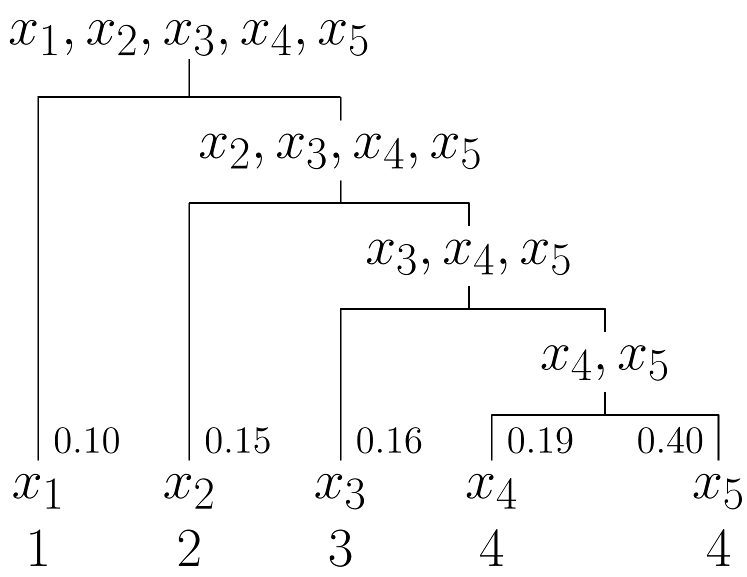
- Suppose there is an oracle, which knows the obtaining value, but responds only if the question can be answered with "yes" or "no".
- A better question scheme than asking for one alternative after the other can easily be found: Divide the set into two subsets of about equal size.
- Ask for containment in an arbitrarily chosen subset.
- Apply this scheme recursively → number of questions bounded by $\lceil \log_2 n \rceil$.

Question/Coding Schemes

$$P(x_1) = 0.10, \quad P(x_2) = 0.15, \quad P(x_3) = 0.16, \quad P(x_4) = 0.19, \quad P(x_5) = 0.40$$

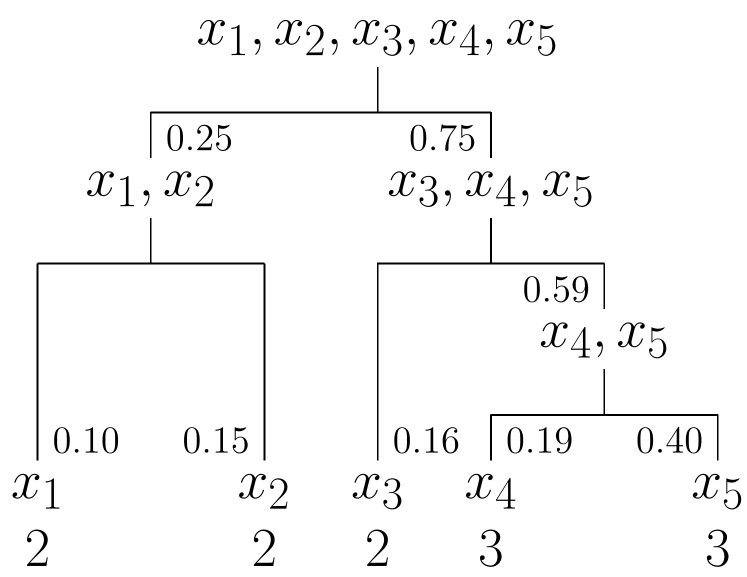
Shannon entropy: $\quad -\sum_i P(x_i) \log_2 P(x_i) = 2.15$ bit/symbol

Linear Traversal



Code length: 3.24 bit/symbol
Code efficiency: 0.664

Equal Size Subsets



Code length: 2.59 bit/symbol
Code efficiency: 0.830

Question/Coding Schemes

Splitting into subsets of about equal size can lead to a bad arrangement of the alternatives into subsets → high expected number of questions.

Good question schemes take the probability of the alternatives into account.

Shannon-Fano Coding (1948)

- Build the question/coding scheme top-down.
- Sort the alternatives w.r.t. their probabilities.
- Split the set so that the subsets have about equal *probability* (splits must respect the probability order of the alternatives).

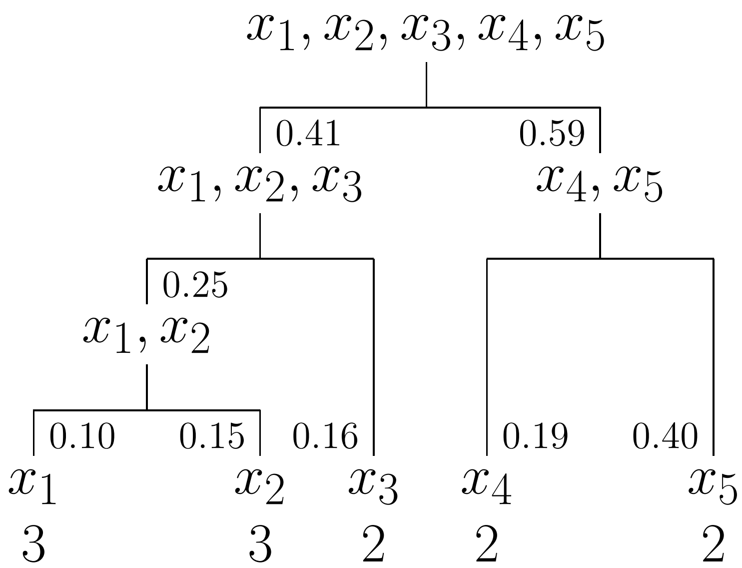
Huffman Coding (1952)

- Build the question/coding scheme bottom-up.
- Start with one element sets.
- Always combine those two sets that have the smallest probabilities.

$$P(x_1) = 0.10, \quad P(x_2) = 0.15, \quad P(x_3) = 0.16, \quad P(x_4) = 0.19, \quad P(x_5) = 0.40$$

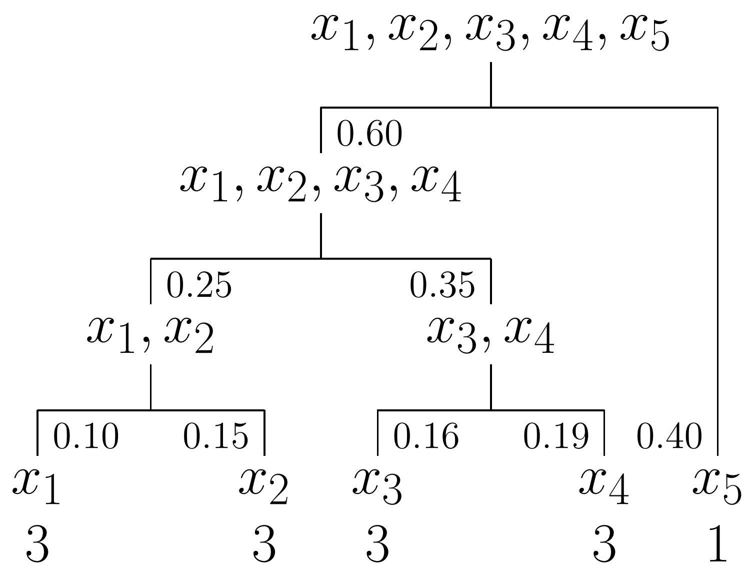
Shannon entropy: $\quad -\sum_i P(x_i) \log_2 P(x_i) = 2.15$ bit/symbol

Shannon–Fano Coding (1948)



Code length: 2.25 bit/symbol
Code efficiency: 0.955

Huffman Coding (1952)



Code length: 2.20 bit/symbol
Code efficiency: 0.977

Question/Coding Schemes

It can be shown that Huffman coding is optimal if we have to determine the obtaining alternative in a single instance.
(No question/coding scheme has a smaller expected number of questions.)

Only if the obtaining alternative has to be determined in a sequence of (independent) situations, this scheme can be improved upon.

Idea: Process the sequence not instance by instance, but combine two, three or more consecutive instances and ask directly for the obtaining combination of alternatives.

Although this enlarges the question/coding scheme, the expected number of questions per identification is reduced (because each interrogation identifies the obtaining alternative for several situations).

However, the expected number of questions per identification cannot be made arbitrarily small. Shannon showed that there is a lower bound, namely the Shannon entropy.

Interpretation of Shannon Entropy

$$P(x_1) = \tfrac{1}{2}, \quad P(x_2) = \tfrac{1}{4}, \quad P(x_3) = \tfrac{1}{8}, \quad P(x_4) = \tfrac{1}{16}, \quad P(x_5) = \tfrac{1}{16}$$

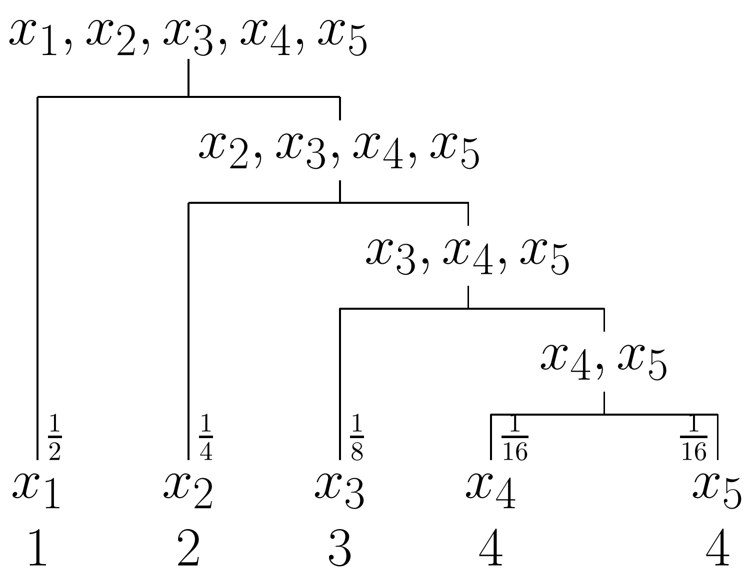
Shannon entropy: $\quad -\sum_i P(x_i) \log_2 P(x_i) = 1.875$ bit/symbol

If the probability distribution allows for a perfect Huffman code (code efficiency 1), the Shannon entropy can easily be interpreted as follows:

$$-\sum_i P(x_i) \log_2 P(x_i)$$
$$= \sum_i \underbrace{P(x_i)}_{\substack{\text{occurrence}\\ \text{probability}}} \cdot \underbrace{\log_2 \frac{1}{P(x_i)}}_{\substack{\text{path length}\\ \text{in tree}}} .$$

In other words, it is the expected number of needed yes/no questions.

Perfect Question Scheme



Code length: 1.875 bit/symbol
Code efficiency: 1

Information Content

The information content of an event $F \in \mathcal{E}$ that occurs with probability $P(F)$ is defined as

$$\mathrm{Inf}_P(F) = -\log_2 P(F).$$

Intention:

Neglect all subjective references to F and let the information content be determined by $P(F)$ only.

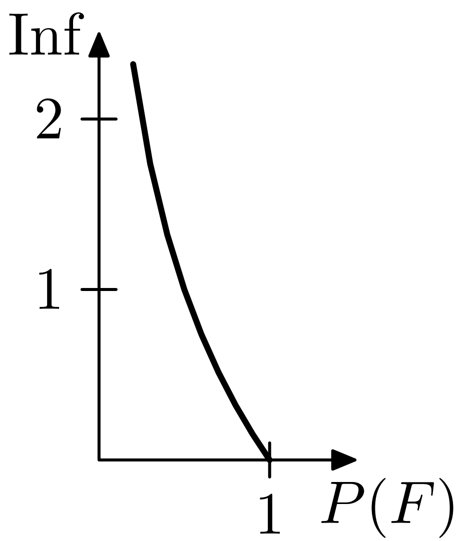
The information of a certain message ($P(\Omega) = 1$) is zero.

The less frequent a message occurs (i. e., the less probable it is), the more interesting is the fact of its occurrence:

$$P(F_1) < P(F_2) \quad \Rightarrow \quad \mathrm{Inf}_P(F_1) > \mathrm{Inf}_P(F_2)$$

We only use one bit to encode the occurrence of a message with probability $\frac{1}{2}$.

The function Inf fulfills all these requirements:



The expected value (w. r. t. to a probability distribution P_1) of Inf_{P_2} can be written as follows:

$$E_{P_1}(\mathrm{Inf}_{P_2}) = -\sum_{F \in \mathcal{E}} P_1(F) \cdot \log_2 P_2(F)$$

$H^{(\mathrm{Shannon})}(P)$ is the expected value (in bits) of the information content that is related to the occurrence of the events $F \in \mathcal{E}$:

$$H(P) = E_P(\mathrm{Inf}_P)$$

$$H^{(\mathrm{Shannon})}(P) = \sum_{F \in \mathcal{E}} \underbrace{P(F)}_{\text{Probability of } F} \cdot \underbrace{(-\log_2 P(F))}_{\text{Information content of } F}$$

Excursus: Approximation Measure

Let P^* be a hypothetical probability distribution and P a (given or known) probability distribution that acts as a reference.

We can compare both P^* and P by computing the **difference of the expected information contents**:

$$\begin{aligned}
E_P(\mathrm{Inf}_{P^*}) - E_P(\mathrm{Inf}_P) &= -\sum_{F \in \mathcal{E}} P(F) \log_2 P^*(F) + \sum_{F \in \mathcal{E}} P(F) \log_2 P(F) \\
&= \sum_{F \in \mathcal{E}} \Big(P(F) \log_2 P(F) - P(F) \log_2 P^*(F) \Big) \\
&= \sum_{F \in \mathcal{E}} P(F) \Big(\log_2 P(F) - \log_2 P^*(F) \Big) \\
I_{\mathrm{KLdiv}}(P, P^*) &= \sum_{F \in \mathcal{E}} P(F) \log_2 \frac{P(F)}{P^*(F)}
\end{aligned}$$

Information Gain (Kullback and Leibler 1951, Quinlan 1986)

Based on Shannon Entropy $H = -\sum_{i=1}^{n} p_i \log_2 p_i$ (Shannon 1948)

$$\begin{aligned} I_{\text{gain}}(C, A) &= \quad H(C) \quad - \quad H(C|A) \\ &= \overbrace{-\sum_{i=1}^{n_C} p_{i.} \log_2 p_{i.}} - \overbrace{\sum_{j=1}^{n_A} p_{.j} \left(-\sum_{i=1}^{n_C} p_{i|j} \log_2 p_{i|j} \right)} \end{aligned}$$

$H(C)$	Entropy of the class distribution (C: class attribute)
$H(C\|A)$	*Expected entropy* of the class distribution if the value of the attribute A becomes known
$H(C) - H(C\|A)$	Expected entropy reduction or *information gain*

Inducing the Decision Tree with Information Gain

Information gain for drug and sex:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Sex}) = \frac{1}{2}\underbrace{\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)}_{H(\text{Drug}|\text{Sex=male})} + \frac{1}{2}\underbrace{\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)}_{H(\text{Drug}|\text{Sex=female})} = 1$$

$$I_{\text{gain}}(\text{Drug}, \text{Sex}) = 1 - 1 = 0$$

No gain at all since the initial the uniform distribution of drug is splitted into two (still) uniform distributions.

Inducing the Decision Tree with Information Gain

Information gain for drug and age:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Age}) = \frac{1}{2}\underbrace{\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right)}_{H(\text{Drug}|\text{Age}\leq 40)} + \frac{1}{2}\underbrace{\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right)}_{H(\text{Drug}|\text{Age}>40)} \approx 0.9183$$

$$I_{\text{gain}}(\text{Drug}, \text{Age}) = 1 - 0.9183 = 0.0817$$

Splitting w. r. t. age can reduce the overall entropy.

Inducing the Decision Tree with Information Gain

Information gain for drug and blood pressure:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Blood\_pr}) = \frac{1}{4}\cdot 0 + \frac{1}{2}\Bigg(\underbrace{-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}}_{H(\text{Drug}|\text{Blood\_pr=normal})}\Bigg) + \frac{1}{4}\cdot 0 = 0.5$$

$$I_{\text{gain}}(\text{Drug}, \text{Blood\_pr}) = 1 - 0.5 = 0.5$$

Largest information gain, so we first split w. r. t. blood pressure (as in the example with misclassification rate).

Inducing the Decision Tree with Information Gain

Next level: Subtree blood pressure is normal.

Information gain for drug and sex:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Sex}) = \frac{1}{2}\underbrace{\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right)}_{H(\text{Drug}|\text{Sex=male})} + \frac{1}{2}\underbrace{\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right)}_{H(\text{Drug}|\text{Sex=female})} = 0.9183$$

$$I_{\text{gain}}(\text{Drug}, \text{Sex}) = 0.0817$$

Entropy can be decreased.

Inducing the Decision Tree with Information Gain

Next level: Subtree blood pressure is normal.

Information gain for drug and age:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Age}) = \frac{1}{2}\cdot 0 + \frac{1}{2}\cdot 0 = 0$$

$$I_{\text{gain}}(\text{Drug}, \text{Age}) = 1$$

Maximal information gain, that is we result in a perfect classification (again, as in the case of using misclassification rate).