Machine Learning

4. Decision Trees

Lars Schmidt-Thieme

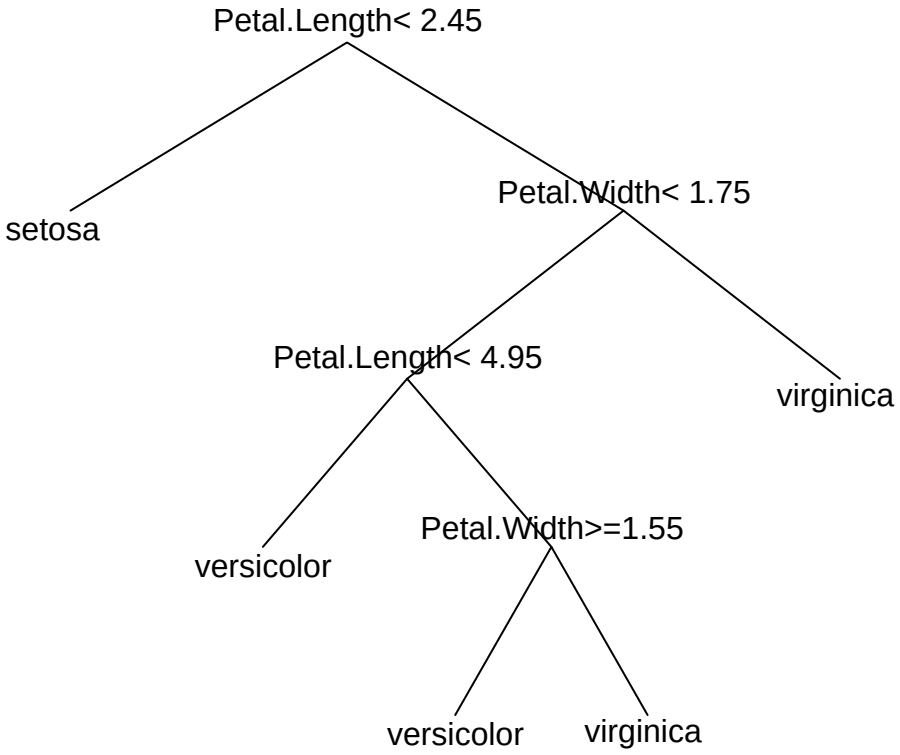
Information Systems and Machine Learning Lab (ISMLL)
Institute for Business Economics and Information Systems
& Institute for Computer Science
University of Hildesheim
http://www.ismll.uni-hildesheim.de

Decision Tree

A **decision tree** is a tree that

1. at each **inner node** has a **decision rule** that assigns instances uniquely to child nodes of the actual node, and

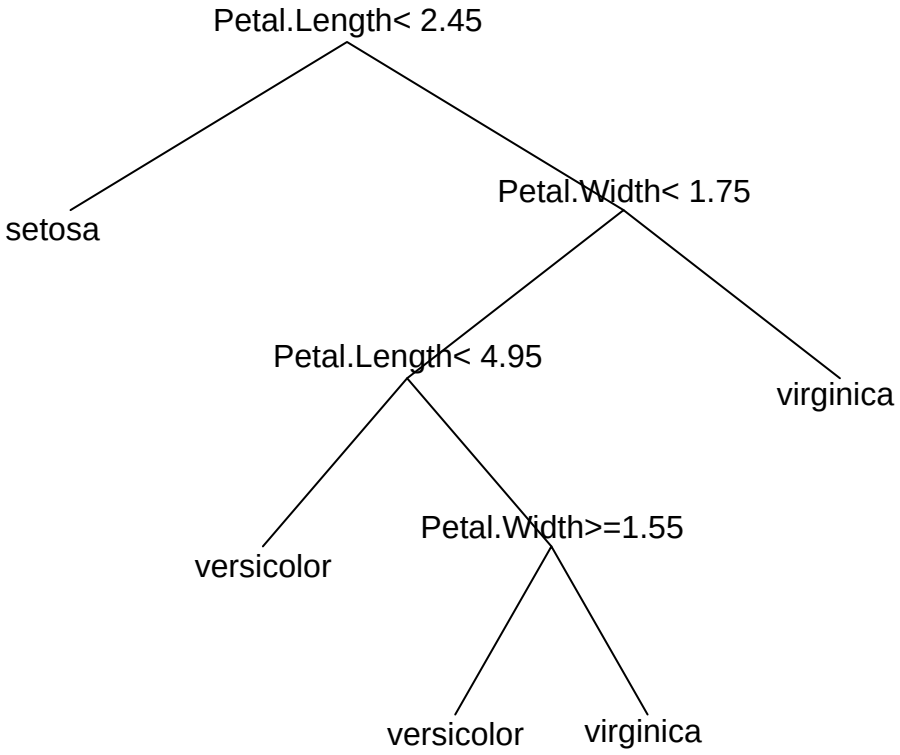
2. at each **leaf node** has a class label.



Using a Decision Tree

The class of a given case $x \in X$ is predicted by

1. starting at the root node,
2. at each interior node
   - evaluate the decision rule for x and
   - branch to the child node picked by the decision rule,
     (default: left = “true”, right = “false”)
3. once a leaf node is reached,
   - predict the class assigned to that node as class of the case x.



Example:

x: Petal.Length = 6, Petal.Width = 1.6

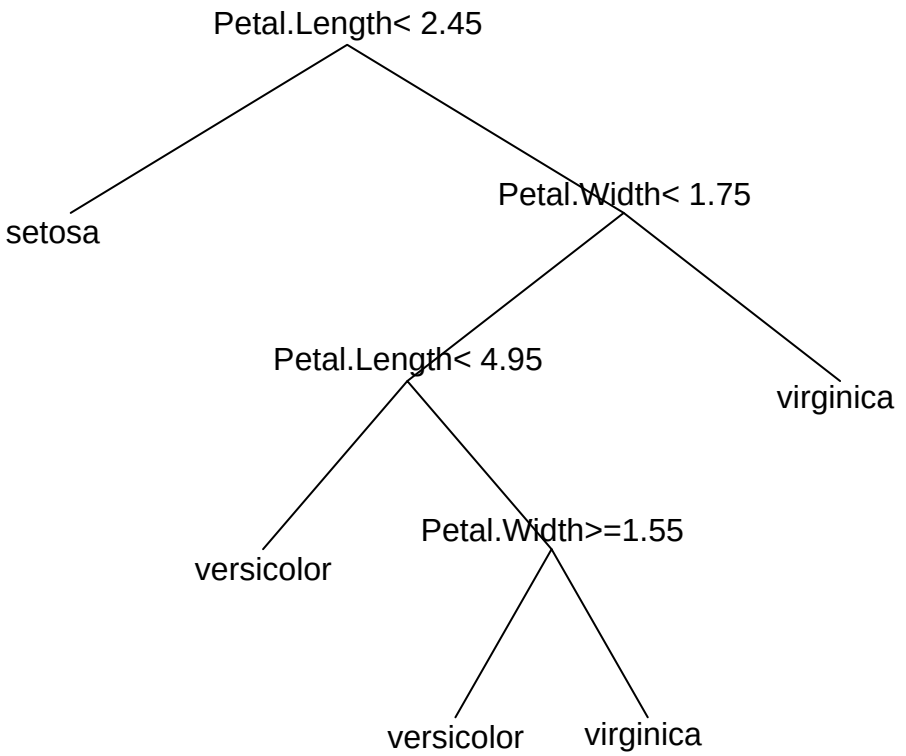
Decision Tree as Set of Rules

Each branch of a decision tree can be formulated as a single conjunctive rule

> if condition$_1(x)$ and condition$_2(x)$ and ... and condition$_k(x)$, then $y =$ class label at the leaf of the branch.

A decision tree is equivalent to a set of such rules, one for each branch.

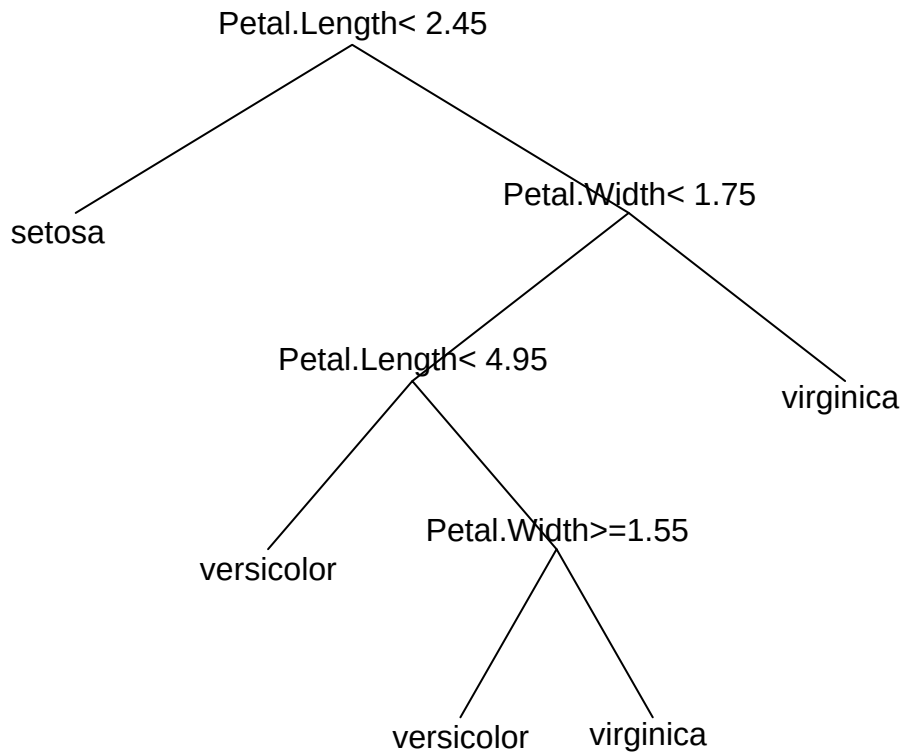
Decision Tree as Set of Rules



set of rules:

Petal.Length $<$ 2.45 $\rightarrow$ class=setosa
Petal.Length $\geq$ 2.45 and Petal.Width $<$ 1.75 and Petal.Length $<$ 4.95 $\rightarrow$ class=versicolor
Petal.Length $\geq$ 2.45 and Petal.Width $<$ 1.75 and Petal.Length $\geq$ 4.95 and Petal.Width $\geq$ 1.55 $\rightarrow$ class=versicolor
Petal.Length $\geq$ 2.45 and Petal.Width $<$ 1.75 and Petal.Length $\geq$ 4.95 and Petal.Width $<$ 1.55 $\rightarrow$ class=virginica
Petal.Length $\geq$ 2.45 and Petal.Width $\geq$ 1.75 $\rightarrow$ class=virginica

Decision Tree as Set of Rules

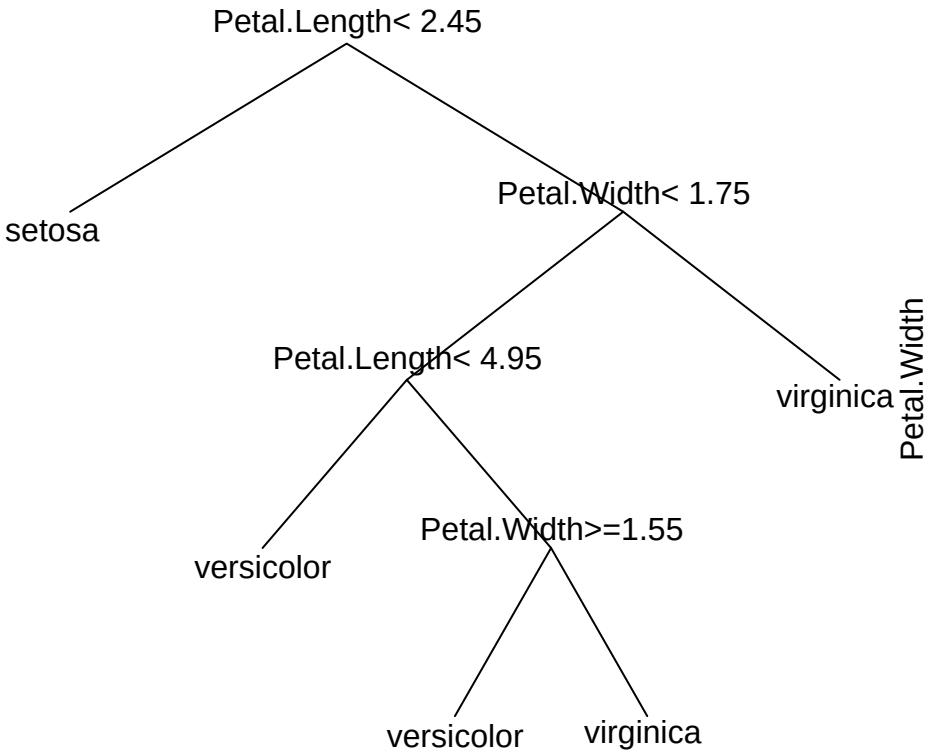


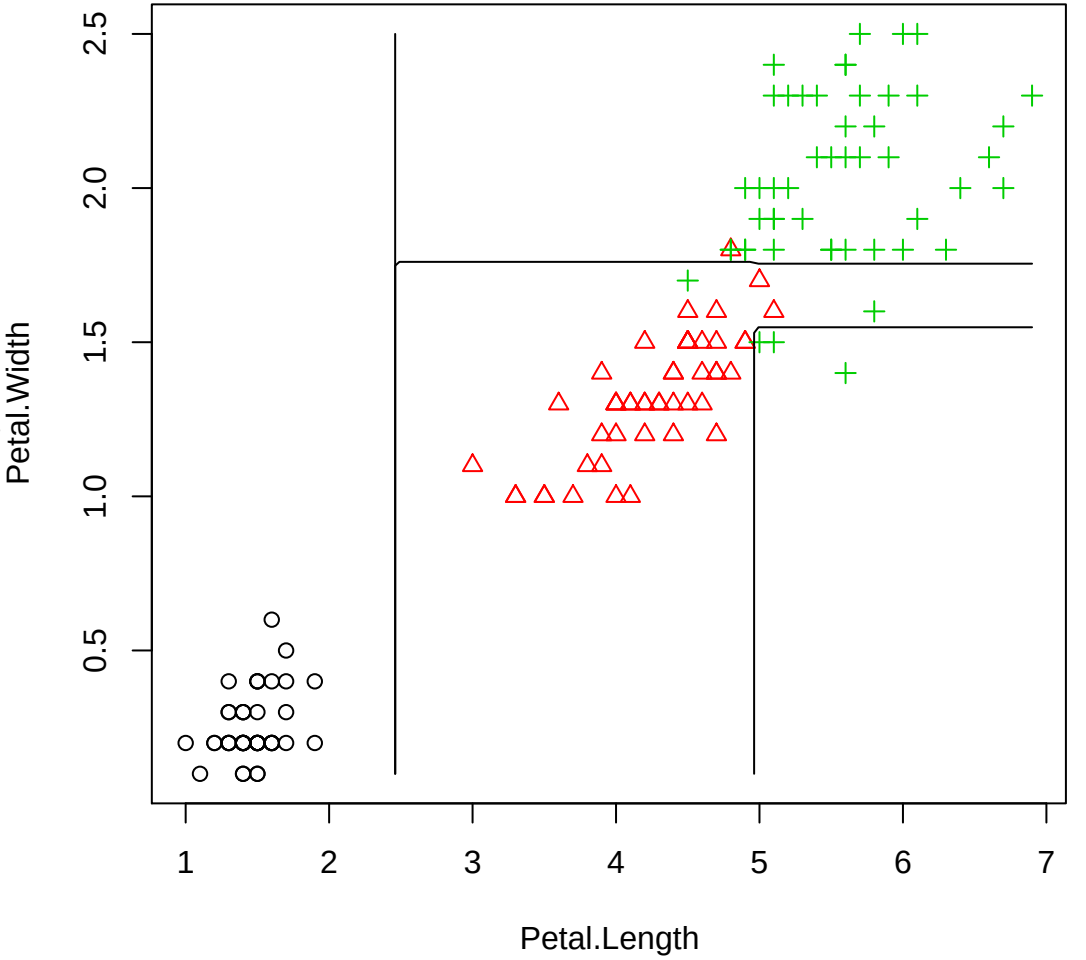
set of rules:

Petal.Length $<$ 2.45		$\rightarrow$ class=setosa
Petal.Length $\in$ [2.45, 4.95[	and Petal.Width $<$ 1.75	$\rightarrow$ class=versicolor
Petal.Length $\geq$ 4.95	and Petal.Width $\in$ [1.55, 1.75[	$\rightarrow$ class=versicolor
Petal.Length $\geq$ 4.95	and Petal.Width $<$ 1.55	$\rightarrow$ class=virginica
Petal.Length $\geq$ 2.45	and Petal.Width $\geq$ 1.75	$\rightarrow$ class=virginica

Decision Boundaries

Decision boundaries are rectangular.





Regression Tree

A **regression tree** is a tree that

1. at each **inner node** has a **decision rule** that assigns instances uniquely to child nodes of the actual node, and

2. at each **leaf node** has a target value.

Probability Trees

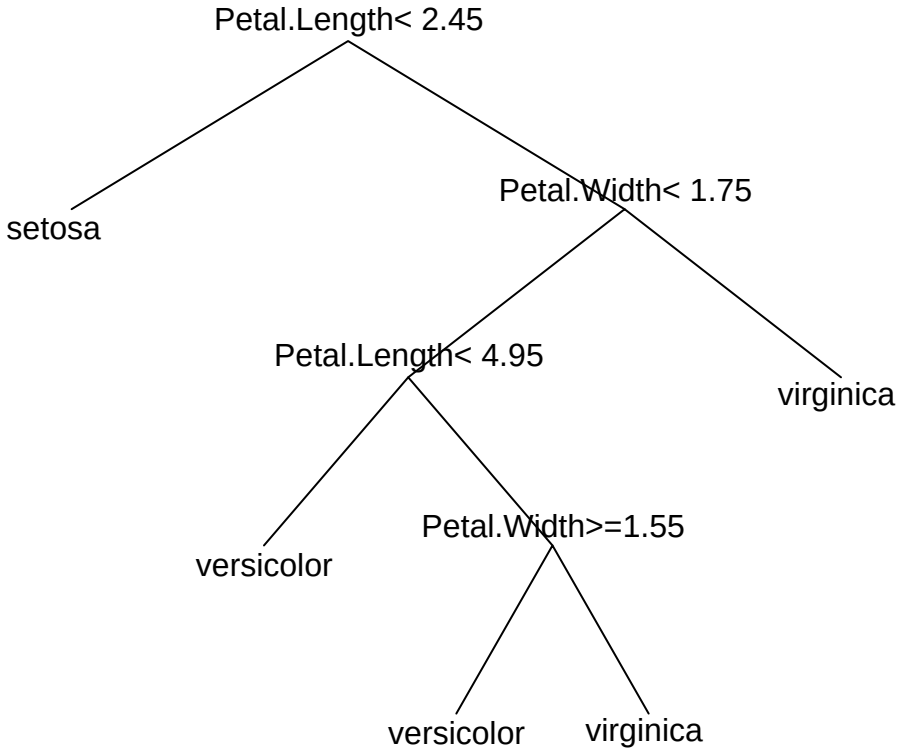
A **probability tree** is a tree that

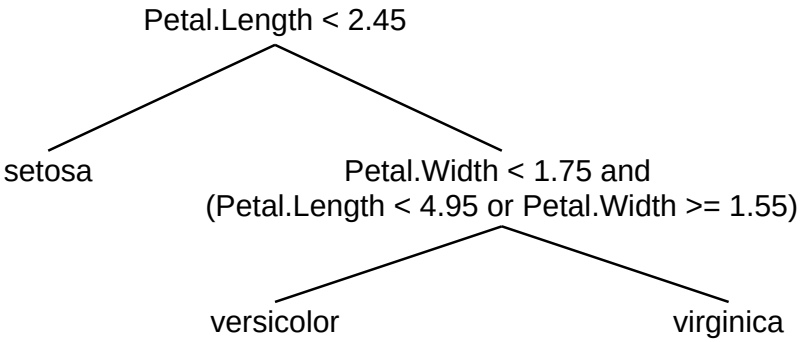
1. at each **inner node** has a **decision rule** that assigns instances uniquely to child nodes of the actual node, and

2. at each **leaf node** has a class probability distribution.

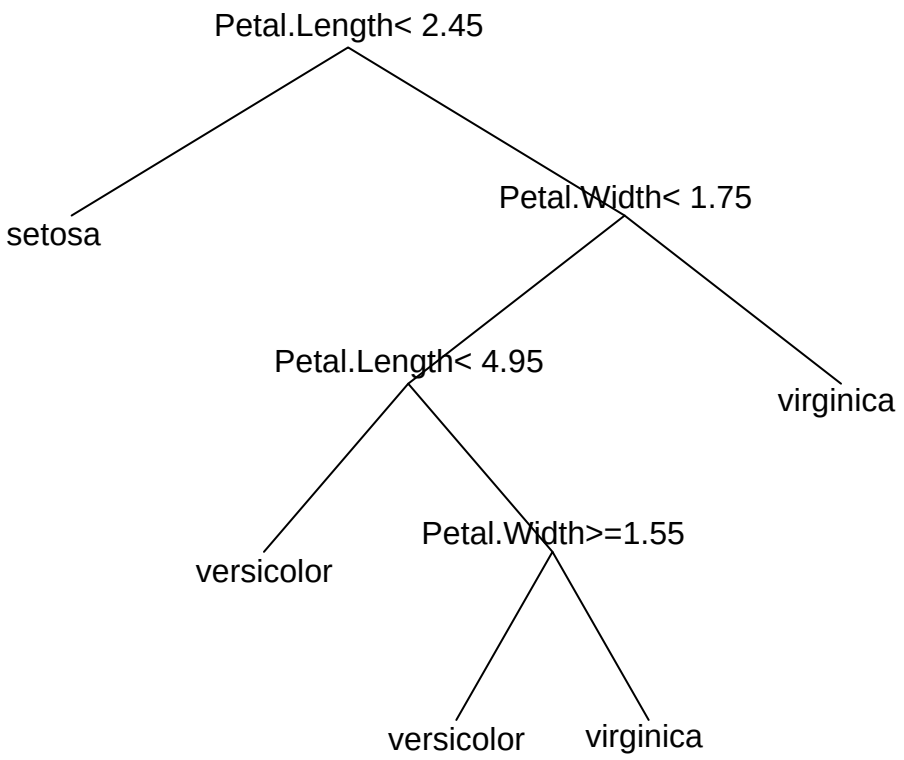
1. What is a Decision Tree?
2. Splits
3. Regularization
4. Learning Decision Trees
5. Digression: Incomplete Data
6. Properties of Decision Trees
7. Pruning Decision Trees

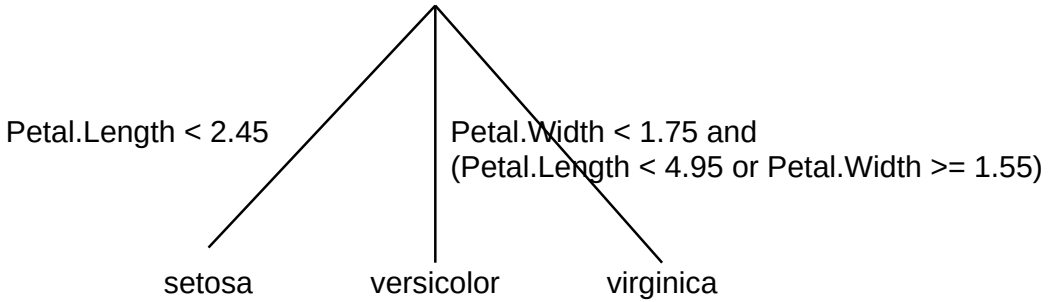
An alternative Decision Tree?





An alternative Decision Tree?





Simple Splits

To allow all kinds of decision rules at the interior nodes (also called **splits**) does not make much sense. The very idea of decision trees is that

- the splits at each node are rather simple and
- more complex structures are captured by chaining several simple decisions in a tree structure.

Therefore, the set of possible splits is kept small by opposing several types of restrictions on possible splits:

- by restricting the number of variables used per split (univariate vs. multivariate decision tree),
- by restricting the number of children per node (binary vs. n-ary decision tree),
- by allowing only some special types of splits (e.g., complete splits, interval splits, etc.).

Types of Splits: Univarite vs. Multivariate

A split is called **univariate** if it uses only a single variable, otherwise **multivariate**.

Example:
“Petal.Width < 1.75” is univariate,
“Petal.Width < 1.75 and Petal.Length < 4.95” is bivariate.

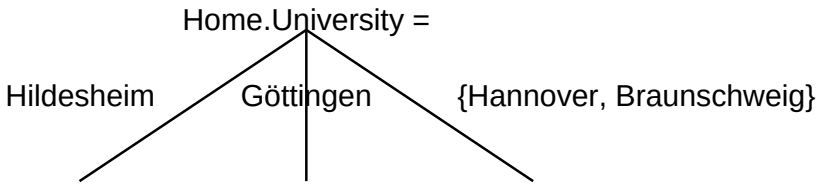
Multivariate splits that are mere conjunctions of univariate splits better would be represented in the tree structure.

But there are also multivariate splits than cannot be represented by a conjunction of univariate splits, e.g.,
“Petal.Width / Petal.Length < 1”

Types of Splits: n-ary

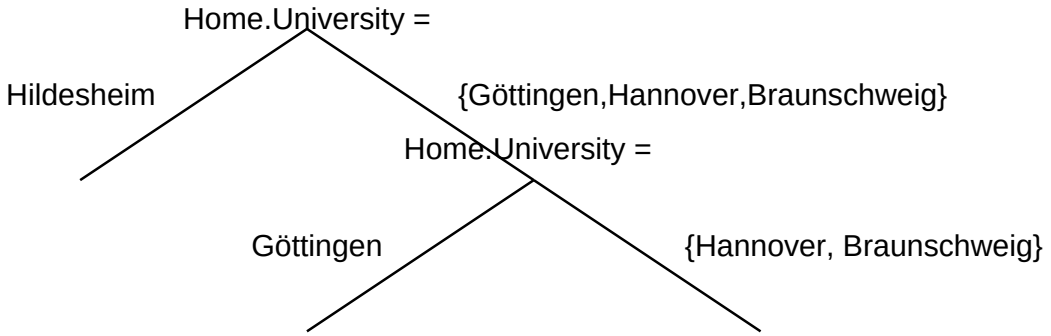
A split is called **n-ary** if it has n children.
(**Binary** is used for 2-ary, **ternary** for 3-ary.)

Example:
"Petal.Length < 1.75" is binary,



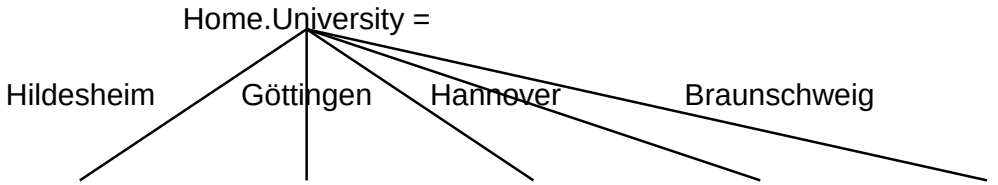
is ternary.

All n-ary splits can be also represented as a tree of binary splits, e.g.,



Types of Splits: Complete Splits

A univariate split on a nominal variable is called **complete** if each value is mapped to a child of its own, i.e., the mapping between values and children is bijective.



A complete split is n-ary (where n is the number of different values for the nominal variable).

Types of Splits: Interval Splits

A univariate split on an at least ordinal variable is called **interval split** if for each child all the values assigned to that child are an interval.

Example:
“Petal.Width < 1.75” is an interval split,
“Petal.Width < 1.75 and Petal.Width >= 1.45” also is an interval split.

“Petal.Width < 1.75 or Petal.Width >= 2.4” is not an interval split.

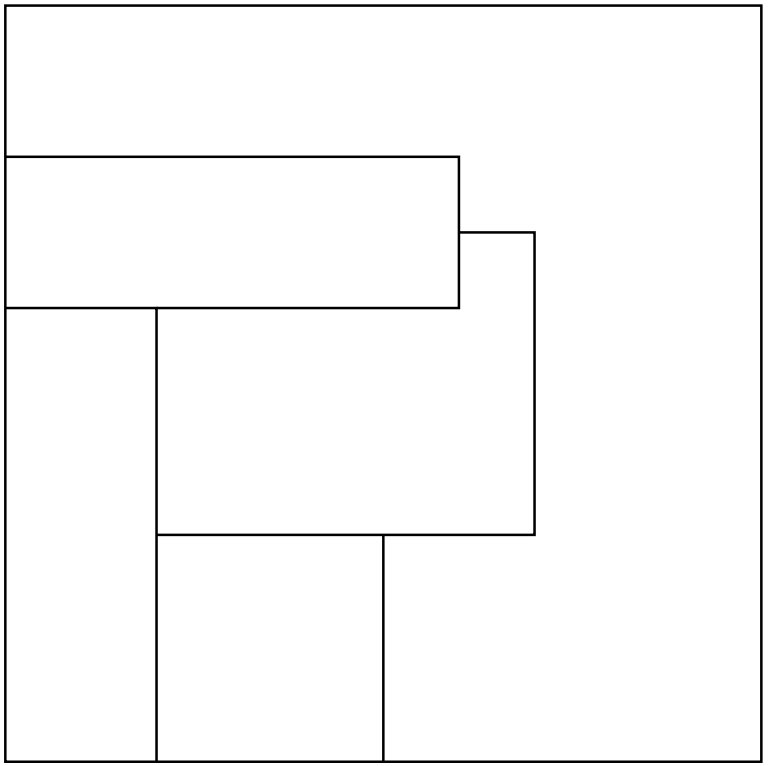
Types of Decision Trees

A decision tree is called
- **univariate**,
- n**-ary**,
- **with complete splits** or
- **with interval splits**,

if all its splits have the corresponding property.

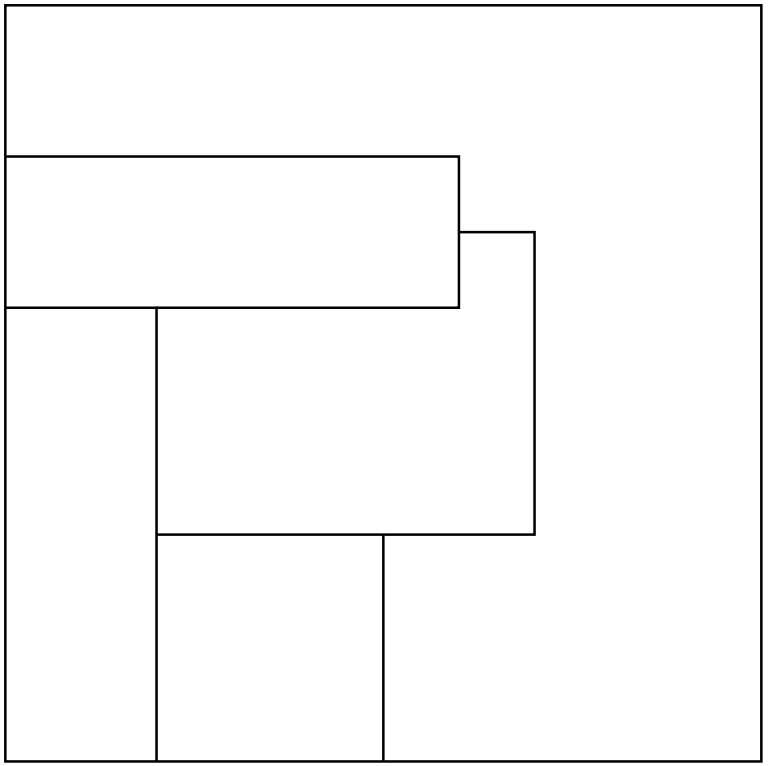
Binary Univariate Interval Splits

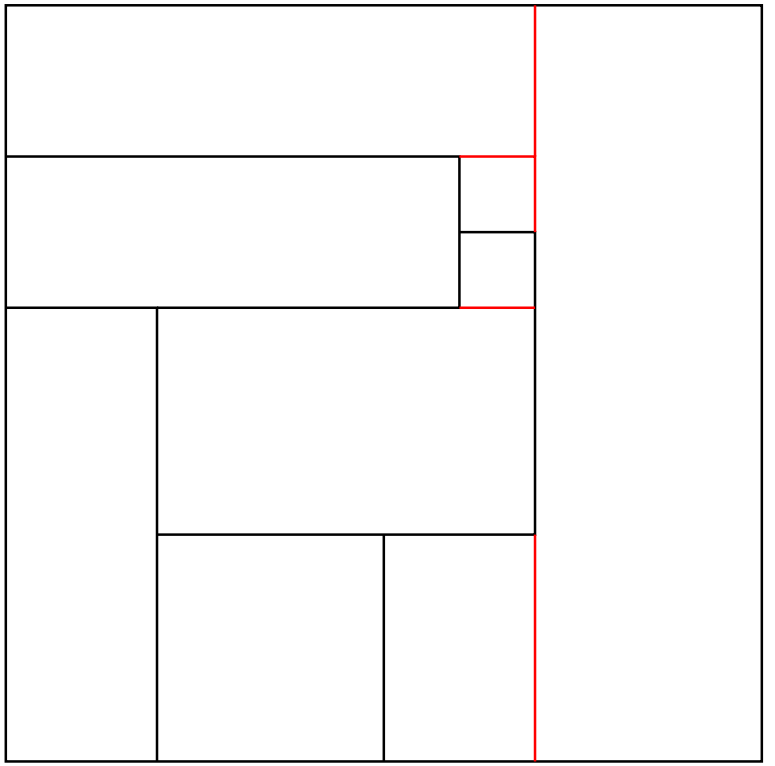
There are partitions (sets of rules)
that cannot be created by binary univariate splits.



Binary Univariate Interval Splits

There are partitions (sets of rules)
that cannot be created by binary univariate splits.





But all partitions can be refined
s.t. they can be created by binary univariate splits.

1. What is a Decision Tree?
2. Splits
3. Regularization
4. Learning Decision Trees
5. Digression: Incomplete Data
6. Properties of Decision Trees
7. Pruning Decision Trees

Learning Regression Trees (1/2)

Imagine, the tree structure is already given,
thus the partition

$$R_j, \quad j = 1, \ldots, k$$

of the predictor space is already given.

Then the remaining problem is to assign a predicted value

$$\hat{y}_j, \quad j = 1, \ldots k$$

to each cell.

Learning Regression Trees (2/2)

Fit criteria such as the smallest residual sum of squares can be decomposed in partial criteria for cases falling in each cell:

$$\sum_{i=1}^{n}(y_i - \hat{y}(x_i))^2 = \sum_{j=1}^{k}\sum_{i=1, x_i \in R_j}^{n}(y_i - \hat{y}_j)^2$$

and this sum is minimal if the partial sum for each cell is minimal.

This is the same as fitting a constant model to the points in each cell and thus the $\hat{y}_j$ with smallest RSS are just the means:

$$\hat{y}_j := \text{average}\{y_i \,|\, i = 1, \ldots, n; x_i \in R_j\}$$

Learning Decision Trees

The same argument shows that
for a probability tree with given structure
the class probabilities with maximum likelihood are just
the relative frequencies of the classes of the points in that region:

$$\hat{p}(Y = y \,|\, x \in R_j) = \frac{|\{i \,|\, i = 1, \ldots, n; x_i \in R_j, y_i = y\}|}{|\{i \,|\, i = 1, \ldots, n; x_i \in R_j\}|}$$

And for a decision tree with given structure, that
the class label with smallest misclassification rate is just
the majority class label of the points in that region:

$$\hat{y}(x \in R_j) = \text{argmax}_y \, |\{i \,|\, i = 1, \ldots, n; x_i \in R_j, y_i = y\}|$$

Possible Tree Structures

Even when possible splits are restricted,
e.g., only binary univariate interval splits are allowed,
then tree structures can be build that separate all cases in tiny cells that contain just a single point
(if there are no points with same predictors).

For such a very fine-grained partition,
the fit criteria would be optimal
(RSS=0, misclassification rate=0, likelihood maximal).

Thus, decision trees need some sort of regularization to make sense.

Regularization Methods

There are several simple regularization methods:

minimum number of points per cell:
require that each cell (i.e., each leaf node) covers a given minimum number of training points.

maximum number of cells:
limit the maximum number of cells of the partition (i.e., leaf nodes).

maximum depth:
limit the maximum depth of the tree.

The number of points per cell, the number of cells, etc. can be seen as a hyperparameter of the decision tree learning method.

1. What is a Decision Tree?
2. Splits
3. Regularization
4. Learning Decision Trees
5. Digression: Incomplete Data
6. Properties of Decision Trees
7. Pruning Decision Trees

Decision Tree Learning Problem

The decision tree learning problem could be described as follows:
Given a dataset

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

find a decision tree $\hat{y} : X \to Y$ that

- is binary, univariate, and with interval splits,
- contains at each leaf a given minimum number m of examples,
- and has minimal misclassification rate

$$\frac{1}{n} \sum_{i=1}^{n} I(y_i \neq \hat{y}(x_i))$$

among all those trees.

Unfortunately, this problem is not feasible as
there are too many tree structures / partitions to check
and no suitable optimization algorithms to sift efficiently through them.

Greedy Search

Therefore, a greedy search is conducted that

- builds the tree recursively starting from the root
- by selecting the locally optimal decision in each step.
  (or alternatively, even just some locally good decision).

Greedy Search / Possible Splits (1/2)

At each node one tries all possible splits.

For an univariate binary tree with interval splits
at the actual node let there still be the data

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

Then check for each predictor variable X with domain $\mathcal{X}$:

if X is a nominal variable:

all $2^{m-1} - 1$ possible splits in two subsets $X_1 \dot{\cup} X_2$.

E.g., for $\mathcal{X} = \{\text{Hi}, \text{Gö}, \text{H}\}$ the splits

$$\begin{array}{lll} \{\text{Hi}\} & \text{vs.} & \{\text{Gö}, \text{H}\} \\ \{\text{Hi}, \text{Gö}\} & \text{vs.} & \{\text{H}\} \\ \{\text{Hi}, \text{H}\} & \text{vs.} & \{\text{Gö}\} \end{array}$$

Greedy Search / Possible Splits (2/2)

if X is an ordinal or interval-scaled variable:

sort the x_i as

$$x'_1 < x'_2 < \ldots < x'_{n'}, \quad n' \leq n$$

and then test all $n' - 1$ possible splits at

$$\frac{x'_i + x'_{i+1}}{2}, \quad i = 1, \ldots, n' - 1$$

E.g.,

$$(x_1, x_2, \ldots, x_8) = (15, 10, 5, 15, 10, 10, 5, 5), \quad n = 8$$

are sorted as

$$x'_1 := 5 < x'_2 := 10 < x'_3 := 15, \quad n' = 3$$

and then split at 7.5 and 12.5.

Greedy Search / Original Fit Criterion

All possible splits – often called **candidate splits** – are assessed by a **quality criterion**.

For all kinds of trees the original fit criterion can be used, i.e.,

for regression trees:
the residual sum of squares.

for decision trees:
the misclassification rate.

for probability trees:
the likelihood.

The split that gives the best improvement is choosen.

Example

Artificial data about visitors of an online shop:

	referrer	num.visits	duration	buyer
1	search engine	several	15	yes
2	search engine	once	10	yes
3	other	several	5	yes
4	ad	once	15	yes
5	ad	once	10	no
6	other	once	10	no
7	other	once	5	no
8	ad	once	5	no

Build a decision tree that tries to predict if a visitor will buy.

Example / Root Split

Step 1 (root node): The root covers all 8 visitors.

There are the following splits:

		buyer		
variable	values	yes	no	errors
referrer	{s}	2	0	2
	{a, o}	2	4	
referrer	{s, a}	3	2	3
	{o}	1	2	
referrer	{s, o}	3	2	3
	{a}	1	2	
num.visits	once	2	4	2
	several	2	0	
duration	<7.5	1	2	3
	≥ 7.5	3	2	
duration	<12.5	2	4	2
	≥ 12.5	2	0	

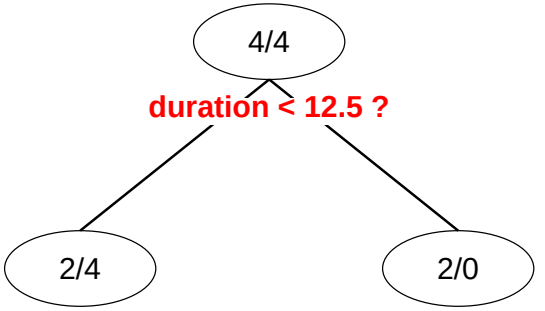
Example / Root Split

The splits

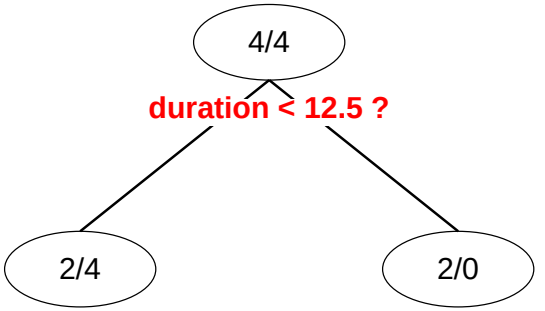
- referrer = search engine ?
- num.visits = once ?
- duration < 12.5 ?

are locally optimal at the root.

We choose “duration < 12.5”:



Example / Node 2 Split



The right node is pure and thus a leaf.

Step 2 (node 2): The left node (called ”node 2”) covers the following cases:

	referrer	num.visits	duration	buyer
2	search engine	once	10	yes
3	other	several	5	yes
5	ad	once	10	no
6	other	once	10	no
7	other	once	5	no
8	ad	once	5	no

Example / Node 2 Split

At node 2 are the following splits:

variable	values	buyer yes	buyer no	errors
referrer	{s}	1	0	1
	{a, o}	1	4	
referrer	{s, a}	1	2	2
	{o}	1	2	
referrer	{s, o}	2	2	2
	{a}	0	2	
num.visits	once	1	4	1
	several	1	0	
duration	<7.5	1	2	2
	≥ 7.5	1	2	

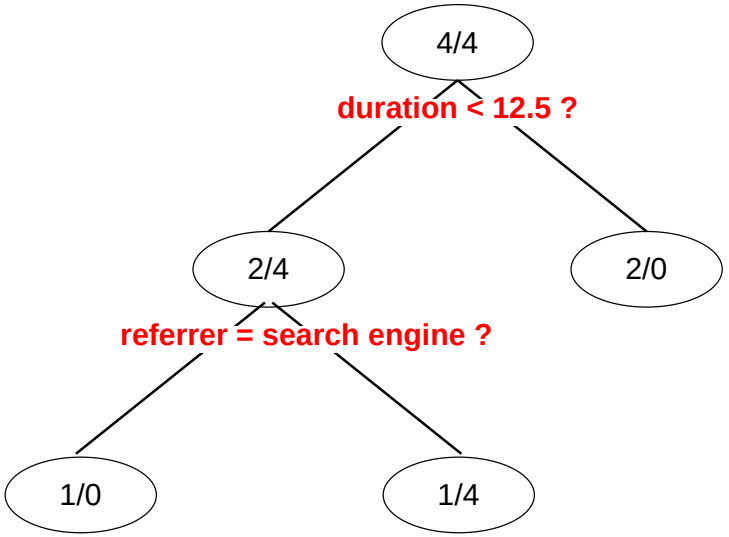
Again, the splits

- referrer = search engine ?
- num.visits = once ?

are locally optimal at node 2.

Example / Node 5 Split

We choose the split “referrer = search engine”:



The left node is pure and thus a leaf.

The right node (called ”node 5”) allows further splits.

Example / Node 5 Split

Step 3 (node 5): The right node (called ”node 5”) covers the following cases:

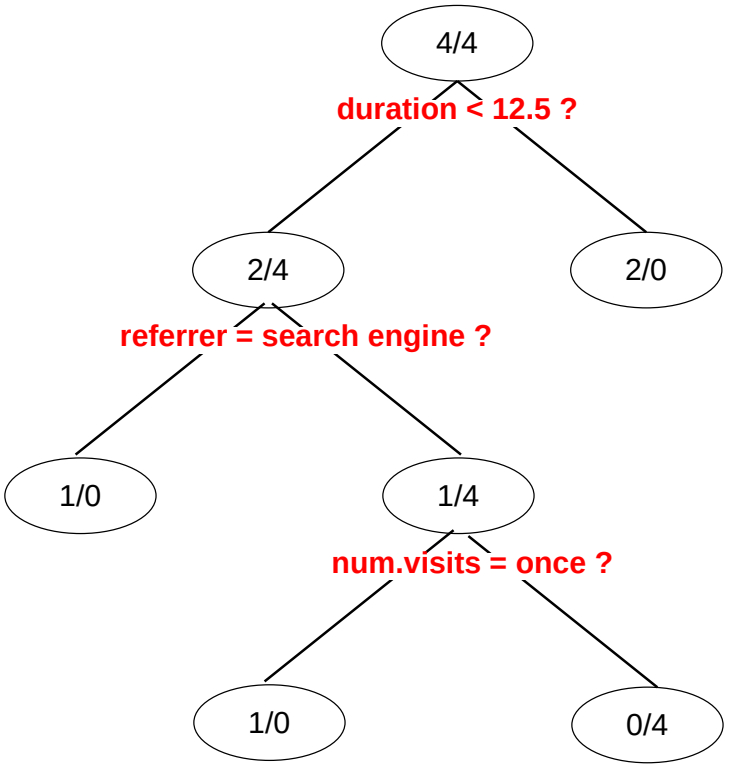
	referrer	num.visits	duration	buyer
3	other	several	5	yes
5	ad	once	10	no
6	other	once	10	no
7	other	once	5	no
8	ad	once	5	no

It allows the following splits:

		buyer		
variable	values	yes	no	errors
referrer	{a}	0	2	1
	{o}	1	2	
num.visits	once	1	0	0
	several	0	4	
duration	<7.5	1	2	1
	≥ 7.5	0	2	

Example / Node 5 Split

The split “num.visits = once” is locally optimal.



Both child nodes are pure thus leaf nodes.

The algorithm stops.

Decision Tree Learning Algorithm

(1) expand-decision-tree(node T, training data X) :
(2) if stopping-criterion(X)
(3) T.class $= \text{argmax}_{y'} |\{(x,y) \in X \mid y = y'\}|$
(4) return
(5) fi
(6) $s := \text{argmax}_{\text{split } s}$ quality-criterion(s)
(7) if s does not improve
(8) T.class $= \text{argmax}_{y'} |\{(x,y) \in X \mid y = y'\}|$
(9) return
(10) fi
(11) T.split $:= s$
(12) for $z \in \text{Im}(s)$ do
(13) create new node T'
(14) T.child$[z] := T'$
(15) expand-decision-tree($T', \{(x,y) \in X \mid s(x) = z\}$)
(16) od

Decision Tree Learning Algorithm / Remarks (1/2)

stopping-criterion(X):
e.g., all cases in X belong to the same class,
all cases in X have the same predictor values (for all variables),
there are less than the minimum number of cases per node to split.

split s:
all possible splits, e.g., all binary univariate interval splits.

quality-criterion(s):
e.g., misclassification rate in X after the split (i.e., if in each child node suggested by the split the majority class is predicted).

s does not improve:
e.g., if the misclassification rate is the same as in the actual node (without the split s).

Decision Tree Learning Algorithm / Remarks (2/2)

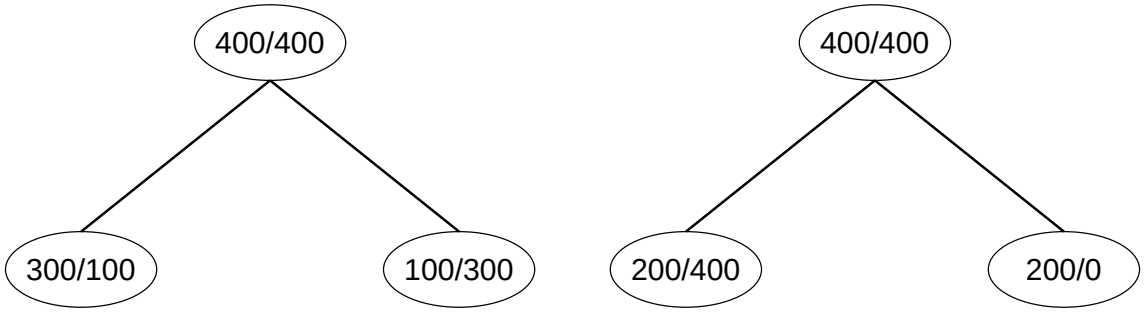
Im(s)**:**
all the possible outcomes of the split,
e.g., { 0, 1 } for a binary split.

T.**child**$(z) := T'$**:**
keep an array that maps all the possible outcomes of the split to the corresponding child node.

Why Misclassification Rate is a Bad Split Quality Criterion

Although it is possible to use misclassification rate as quality criterion, it usually is not a good idea.

Imagine a dataset with a binary target variable (zero/one) and 400 cases per class (400/400).
Assume there are two splits:



Both have 200 errors / misclassification rate 0.25.

But the right split may be preferred as it contains a pure node.

Split Contingency Tables

The effects of a split on training data can be described by a **contingency table** $(C_{j,k})_{j \in J, k \in K}$, i.e., a matrix

- with rows indexed by the different child nodes $j \in J$,
- with columns indexed by the different target classes $k \in K$,
- and cells $C_{j,k}$ containing the number of points in class k that the split assigns to child j:

$$C_{j,k} := |\{(x, y) \in X \mid s(x) = j \text{ and } y = k\}$$

Entropy

Let

$$P_n := \{(p_1, p_2, \ldots, p_n) \in [0,1]^n \mid \sum_i p_i = 1\}$$

be the set of multinomial probability distributions on the values $1, \ldots, n$.

An **entropy function** $q : P_n \to \mathbb{R}_0^+$ has the properties

- q is maximal for uniform $p = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})$.
- q is 0 iff p is deterministic
  (one of the $p_i = 1$ and all the others equal 0).

Entropy

Examples:

Cross-Entropy / Deviance:

$$H(p_1, \ldots, p_n) := -\sum_{i=1}^{n} p_i \log(p_i)$$

Shannons Entropy:

$$H(p_1, \ldots, p_n) := -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Quadratic Entropy:

$$H(p_1, \ldots, p_n) := \sum_{i=1}^{n} p_i(1 - p_i) = 1 - \sum_{i=1}^{n} p_i^2$$

Entropy measures can be extended to $\mathbb{R}_0^+$ via

$$q(x_1, \ldots, x_n) := q(\frac{x_1}{\sum_i x_i}, \frac{x_2}{\sum_i x_i}, \ldots, \frac{x_n}{\sum_i x_i})$$

Entropy for Contingency Tables

For a contingency table $C_{j,k}$ we use the following abbreviations:

$$C_{j,\cdot} := \sum_{k \in K} C_{j,k} \qquad \text{sum of row } j$$

$$C_{\cdot,k} := \sum_{j \in J} C_{j,k} \qquad \text{sum of column } k$$

$$C_{\cdot,\cdot} := \sum_{j \in J} \sum_{k \in K} C_{j,k} \qquad \text{sum of matrix}$$

and define the following entropies:

row entropy:

$$H_J(C) := H(C_{j,\cdot} \,|\, j \in J)$$

column entropy:

$$H_K(C) := H(C_{\cdot,k} \,|\, k \in K)$$

conditional column entropy:

$$H_{K|J}(C) := \sum_{j \in J} \frac{C_{j,\cdot}}{C_{\cdot,\cdot}} H(C_{j,k} \,|\, k \in K)$$

Entropy for Contingency Tables

Suitable split quality criteria are

entropy gain:

$$HG(C) := H_K(C) - H_{K|J}(C)$$

entropy gain ratio:

$$HG(C) := \frac{H_K(C) - H_{K|J}(C)}{H_J(C)}$$

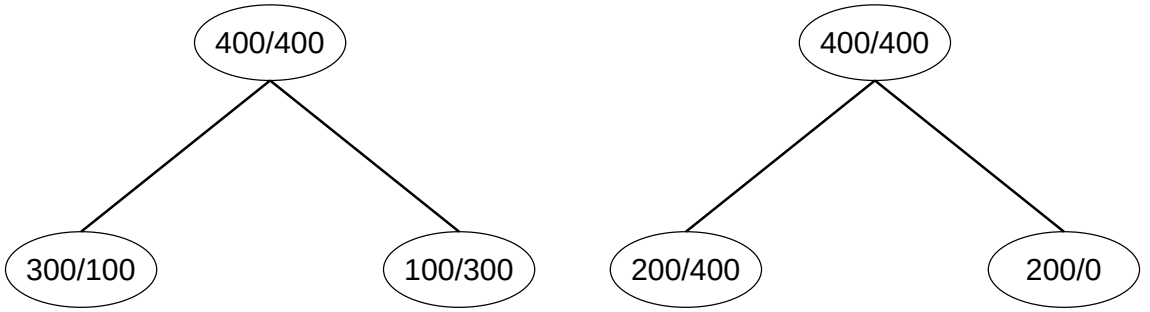
Shannon entropy gain is also called **information gain**:

$$\mathrm{IG}(C) := -\sum_k \frac{C_{.,k}}{C_{.,.}} \log_2 \frac{C_{.,k}}{C_{.,.}} + \sum_j \frac{C_{j,.}}{C_{.,.}} \sum_k \frac{C_{j,k}}{C_{j,.}} \log_2 \frac{C_{j,k}}{C_{j,.}}$$

Quadratic entropy gain is also called **Gini index**:

$$\mathrm{Gini}(C) := -\sum_k (\frac{C_{.,k}}{C_{.,.}})^2 + \sum_j \frac{C_{j,.}}{C_{.,.}} \sum_k (\frac{C_{j,k}}{C_{j,.}})^2$$

Entropy Measures as Split Quality Criterion



Both have 200 errors / misclassification rate 0.25.

But the right split may be preferred as it contains a pure node.

Gini-Impurity

$$= \frac{1}{2}((\frac{3}{4})^2 + (\frac{1}{4})^2) + \frac{1}{2}((\frac{3}{4})^2 + (\frac{1}{4})^2)$$
$$= 0.625$$

Gini-Impurity

$$= \frac{3}{4}((\frac{1}{3})^2 + (\frac{2}{3})^2) + \frac{1}{4}(1^2 + 0^2)$$
$$\approx 0.667$$

1. What is a Decision Tree?
2. Splits
3. Regularization
4. Learning Decision Trees
5. Digression: Incomplete Data
6. Properties of Decision Trees
7. Pruning Decision Trees

Complete and incomplete cases

If for some cases a variable is not observed, i.e., the value of the variable is not known, we say

the case is **incomplete** and
has **a missing value** w.r.t. that variable.

Variables with missing values cannot be used directly in models.

case	F	L	B	D	H
1	0	0	0	0	0
2	0	0	0	0	0
3	1	1	1	1	0
4	0	0	1	1	1
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	1	1
8	0	0	0	0	0
9	0	0	1	1	1
10	1	1	0	1	1

Complete data.

case	F	L	B	D	H
1	0	0	0	0	0
2	.	0	0	0	0
3	1	1	1	1	0
4	0	0	.	1	1
5	0	0	0	0	0
6	0	0	0	0	0
7	0	.	0	.	1
8	0	0	0	0	0
9	0	0	1	1	1
10	1	1	.	1	1

Incomplete data. Missing values are marked by a dot.

Missing value indicators

For each variable v, we can interpret its missing of values as new random variable M_v,

$$M_v := \begin{cases} 1, & \text{if } v_{\text{obs}} = ., \\ 0, & \text{otherwise} \end{cases}$$

called **missing value indicator of** v.

case	F	M_F	L	M_L	B	M_B	D	M_D	H	M_H
1	0	0	0	0	0	0	0	0	0	0
2	.	1	0	0	0	0	0	0	0	0
3	1	0	1	0	1	0	1	0	0	0
4	0	0	0	0	.	1	1	0	1	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	.	1	0	0	.	1	1	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	1	0	1	0	1	0
10	1	0	1	0	.	1	1	0	1	0

Incomplete data and missing value indicators.

Types of missingness / MCAR

A variable v is called **missing completely at random** (MCAR), if the probability of a missing value is (unconditionally) independent of the (true, unobserved) value of v, i.e, if

$$I(M_v, v_{\text{true}})$$

(MCAR is also called **missing unconditionally at random**).

Example: think of an apparatus measuring the velocity v of wind that has a loose contact c. When the contact is closed, the measurement is recorded, otherwise it is skipped. If the contact c being closed does not depend on the velocity v of wind, v is MCAR.

Types of missingness / MCAR / Imputation

If a variable is MCAR, for each value the probability of missing is the same, and, e.g., the sample mean of v_{obs} is an unbiased estimator for the expectation of v_{true}; here

$$\hat{\mu}(v_{\text{obs}}) = \frac{1}{10}(2 \cdot 1 + 4 \cdot 3 + 2 \cdot 3 + 2 \cdot 4)$$
$$= \frac{1}{15}(3 \cdot 1 + 6 \cdot 3 + 3 \cdot 3 + 3 \cdot 4) = \hat{\mu}(v_{\text{true}})$$

Replacing the missing values by the sample mean is called **imputing**. Afterwards, the data is complete and can be used by all models.

case	v_{true}	v_{observed}
1	1	.
2	2	2
3	2	.
4	4	4
5	3	3
6	2	2
7	1	1
8	4	.
9	3	3
10	2	.
11	1	1
12	3	.
13	4	4
14	2	2
15	2	2

Data with a variable v MCAR. Missing values are stroken through.

Types of missingness / MAR

A variable v is called **missing at random** (MAR), if the probability of a missing value is conditionally independent of the (true, unobserved) value of v, i.e, if

$$I(M_v, v_{\text{true}} \mid W)$$

for some set of variables $W \subseteq V \setminus \{v\}$ (MAR is also called **missing conditionally at random**).

Example: think of an apparatus measuring the velocity v of wind. If we measure wind velocities at three different heights $h = 0, 1, 2$ and say the apparatus has problems with height not recording

- 1/3 of cases at height 0,
- 1/2 of cases at height 1,
- 2/3 of cases at height 2,

case	v_{true}	v_{observed}	h
1	1	.	0
2	2	2	0
3	3	.	0
4	3	3	0
5	1	1	0
6	3	3	0
7	1	1	0
8	2	.	0
9	2	2	0
10	3	.	1
11	4	4	1
12	4	.	1
13	3	3	1
14	3	.	2
15	4	4	2
16	4	.	2
17	5	5	2
18	3	.	2
19	5	.	2
20	3	3	2
21	4	.	2
22	5	.	2

Figure 28: Data with a variable v MAR (conditionally on h).

then v is missing at random (conditionally on h).

Types of missingness / MAR

If v depends on variables in W, then, e.g., the sample mean is not an unbiased estimator, but the weighted mean w.r.t. W has to be used; here:

$$\sum_{h=0}^{2} \hat{\mu}(v|H=h)p(H=h)$$
$$=2 \cdot \frac{9}{22} + 3.5 \cdot \frac{4}{22} + 4 \cdot \frac{9}{22}$$
$$\neq \frac{1}{11} \sum_{\substack{i=1,\ldots,22 \\ v_i \neq .}} v_i$$
$$=2 \cdot \frac{6}{11} + 3.5 \cdot \frac{2}{11} + 4 \cdot \frac{3}{11}$$

case	v_{true}	v_{observed}	h
1	1	.	0
2	2	2	0
3	3	.	0
4	3	3	0
5	1	1	0
6	3	3	0
7	1	1	0
8	2	.	0
9	2	2	0
10	3	.	1
11	4	4	1
12	4	.	1
13	3	3	1
14	3	.	2
15	4	4	2
16	4	.	2
17	5	5	2
18	3	.	2
19	5	.	2
20	3	3	2
21	4	.	2
22	5	.	2

Figure 28: Data with a variable v MAR (conditionally on h).

Types of missingness / missing systematically

A variable v is called **missing systematically** (or not at random), if the probability of a missing value does depend on its (unobserved, true) value.

Example: if the apparatus has problems measuring high velocities and say, e.g., misses

- 1/3 of all measurements of $v = 1$,
- 1/2 of all measurements of $v = 2$,
- 2/3 of all measurements of $v = 3$,

i.e., the probability of a missing value does depend on the velocity, v is missing systematically.

case	v_{true}	v_{observed}
1	1	.
2	1	1
3	2	.
4	3	.
5	3	3
6	2	2
7	1	1
8	2	.
9	3	.
10	2	2

Figure 29: Data with a variable v missing systematically.

Again, the sample mean is not unbiased; expectation can only be estimated if we have background knowledge about the probabilities of a missing value dependend on its true value.

Types of missingness / hidden variables

A variable v is called **hidden**, if the probability of a missing value is 1, i.e., it is missing in all cases.

Example: say we want to measure intelligence I of probands but cannot do this directly. We measure their level of education E and their income C instead. Then I is hidden.

case	I_{true}	I_{obs}	E	C
1	1	.	0	0
2	2	.	1	2
3	2	.	2	1
4	2	.	2	2
5	1	.	0	2
6	2	.	2	0
7	1	.	1	2
8	0	.	2	1
9	1	.	2	2
10	2	.	2	1

Figure 30: Data with a hidden variable I.

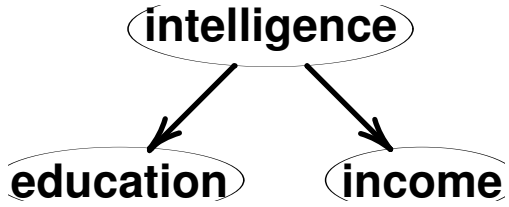


Figure 32: Suggested dependency of variables I, E, and C.

types of missingness

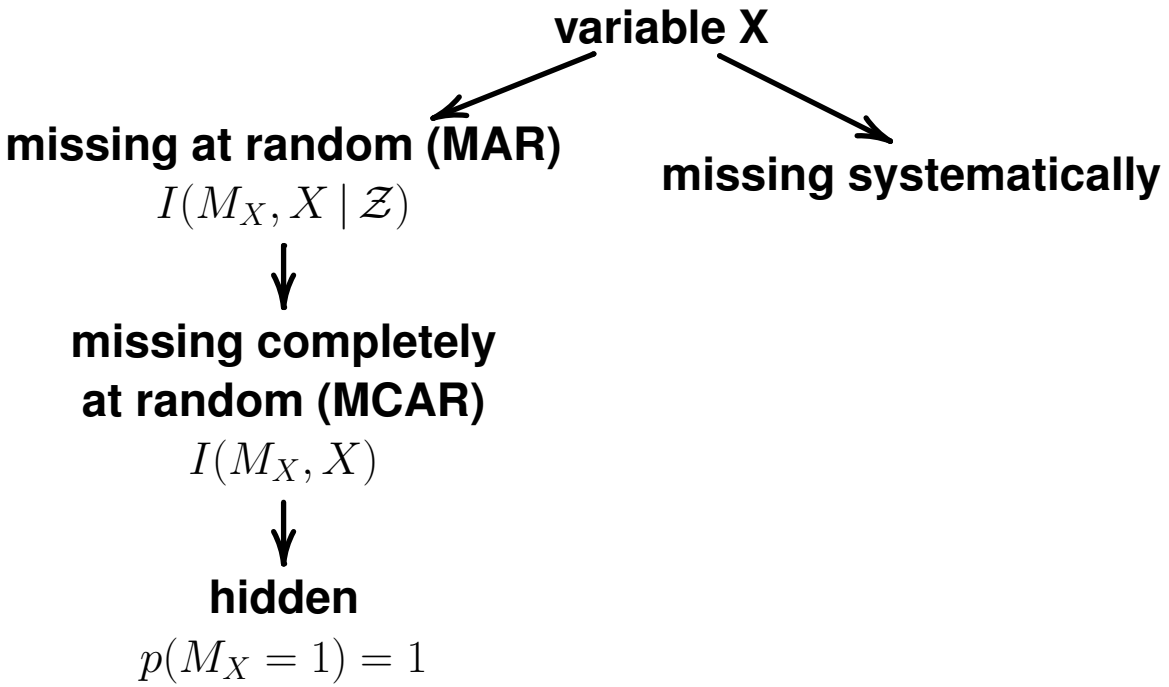


Figure 34: Types of missingness.

MAR/MCAR terminology stems from Little and Rubin 1987.

complete case analysis

The simplest scheme to learn from incomplete data D, e.g., the vertex potentials $(p_v)_{v \in V}$ of a Bayesian network, is **complete case analysis** (also called **casewise deletion**): use only complete cases

$$D_{\text{compl}} := \{d \in D \mid d \text{ is complete}\}$$

case	F	L	B	D	H
1	0	0	0	0	0
2	.	0	0	0	0
3	1	1	1	1	0
4	0	0	.	1	1
5	0	0	0	0	0
6	0	0	0	0	0
7	0	.	0	.	1
8	0	0	0	0	0
9	0	0	1	1	1
10	1	1	.	1	1

Figure 35: Incomplete data and data used in complete case analysis (highlighted).

If D is MCAR, estimations based on the subsample D_{compl} are unbiased for D_{true}.

complete case analysis (2/2)

But for higher-dimensional data (i.e., with a larger number of variables), complete cases might become rare.

Let each variable have a probability for missing values of 0.05, then for 20 variables the probability of a case to be complete is

$$(1 - 0.05)^{20} \approx 0.36$$

for 50 variables it is ≈ 0.08, i.e., most cases are deleted.

available case analysis

A higher case rate can be achieved by **available case analysis**. If a quantity has to be estimated based on a subset $W \subseteq V$ of variables, e.g., the vertext potential p_v of a specific vertex $v \in V$ of a Bayesian network ($W = \mathrm{fam}(v)$), use only complete cases of $D|_W$

$$(D|_W)_{\text{compl}} = \{d \in D|_W \mid d \text{ is complete}\}$$

case	F	L	B	D	H
1	0	0	0	0	0
2	.	0	0	0	0
3	1	1	1	1	0
4	0	0	.	1	1
5	0	0	0	0	0
6	0	0	0	0	0
7	0	.	0	.	1
8	0	0	0	0	0
9	0	0	1	1	1
10	1	1	.	1	1

Figure 36: Incomplete data and data used in available case analysis for estimating the potential $p_L(L \mid F)$ (highlighted).

If D is MCAR, estimations based on the subsample $(D_W)_{\text{compl}}$ are unbiased for $(D_W)_{\text{true}}$.

1. What is a Decision Tree?
2. Splits
3. Regularization
4. Learning Decision Trees
5. Digression: Incomplete Data
6. Properties of Decision Trees
7. Pruning Decision Trees

Missing Values / Surrogate Splits

Decision trees can handle missing values intrinsically, i.e., without imputation, by using **surrogate splits**.

For this, one stores not only the best split (called **primary split**), but also several other splits

- on variables different from the one used in the primary split
- but that result in a similar assingment of training cases to child nodes as the primary split,

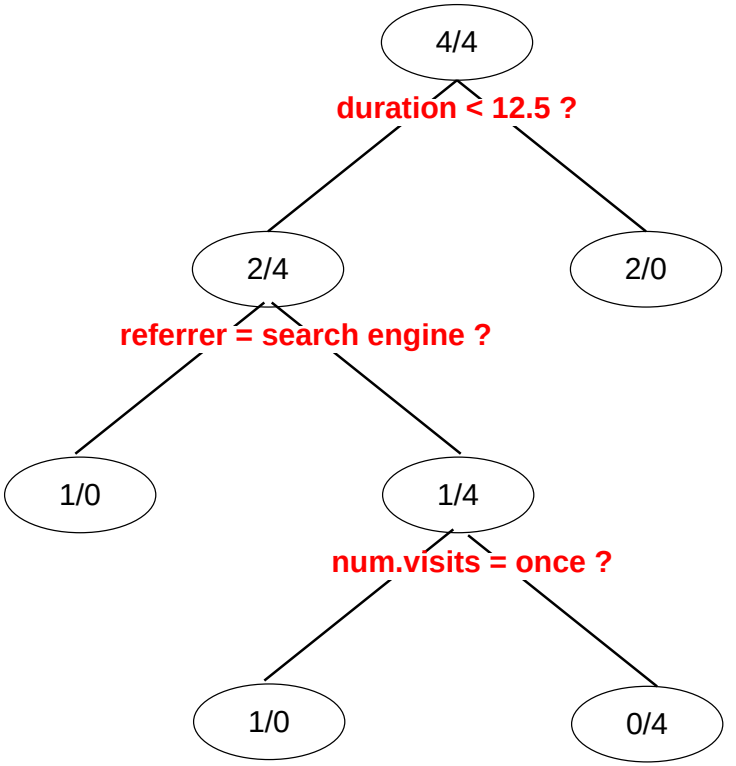
called **surrogate splits**.

If a case with a missing variable used in a primary split has to be predicted, a surrogate split is used instead.

Missing Values / Surrogate Splits / Example

Predict buying behavior of the following customer:

	referrer	num.visits	duration	buyer
9	search engine	once	.	?



Missing Values / Surrogate Splits / Example

The primary split at node 1 is

duration < 12.5

sending only cases 1 and 4 to the right.

A surrogate split mimicks this split as closely as possible, i.e., is the best split for the surrogate target "duration < 12.5".

	referrer	num.visits	duration	buyer	surrogate target
1	search engine	several	15	yes	true
2	search engine	once	10	yes	false
3	other	several	5	yes	false
4	ad	once	15	yes	true
5	ad	once	10	no	false
6	other	once	10	no	false
7	other	once	5	no	false
8	ad	once	5	no	false

Missing Values / Surrogate Splits / Example

At node 1 are the following surrogate splits:

variable	values	surrogate target true	surrogate target false	errors	gini impurity
referrer	{s}	1	1	2	0.667
	{a, o}	5	1		
referrer	{s, a}	3	2	2	0.7
	{o}	3	0		
referrer	{s, o}	3	1	2	0.625
	{a}	3	1		
num.visits	once	5	1	2	0.667
	several	1	1		

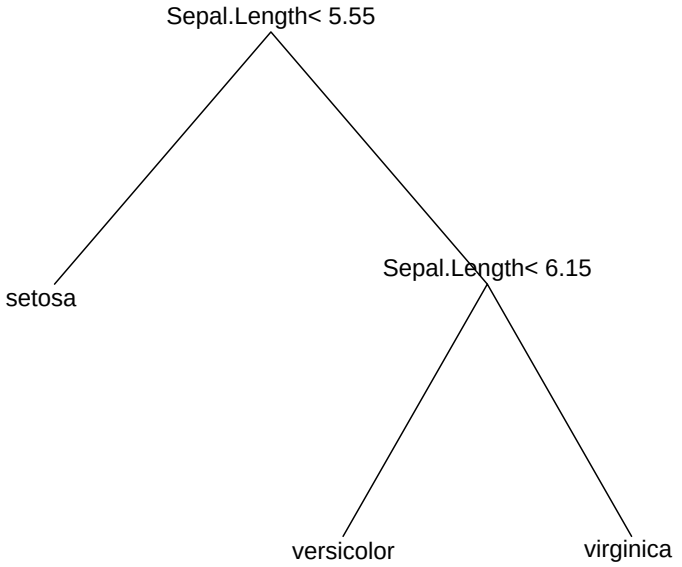
In principle, referrer $\in \{s, a\}$ is the best surrogate split.
But as the majority class in both rows is “true”, a missing value always is sent down to the left.

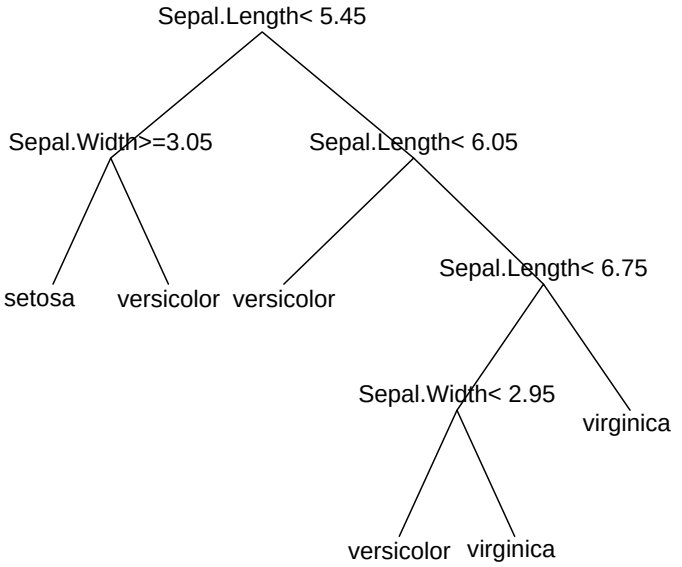
Instability

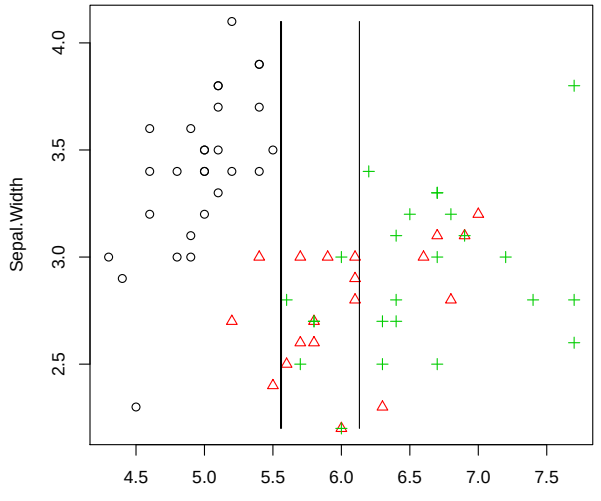
Decision trees often are used to visually explain models.

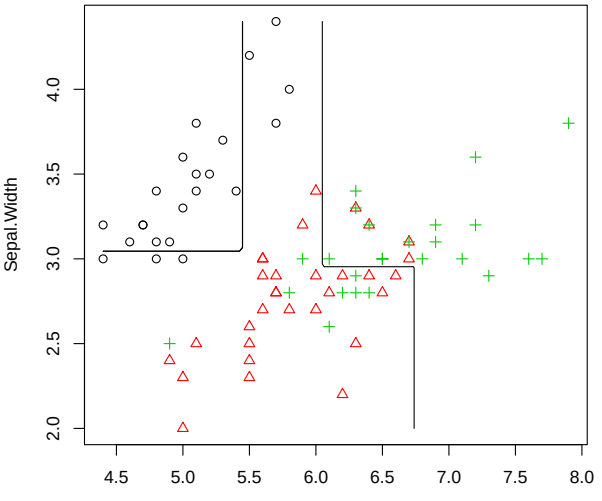
Nevertheless, usually there are many candidates for the primary split with very similar values of the quality criterion. So the choice of the primary split shown in the tree is somewhat arbitrary: the split may change, if the data changes a bit. The tree is said to be **instable**.

Instability / Example / Iris (50/50)









1. What is a Decision Tree?
2. Splits
3. Regularization
4. Learning Decision Trees
5. Digression: Incomplete Data
6. Properties of Decision Trees
7. Pruning Decision Trees

Subtree Selection (Pruning)

If a decision tree has been overgrown, i.e., overfits,
it can be regularized by selecting the best subtree (**pruning**).

Subtrees

Let nodes$(\hat{y})$ denote the **set of nodes** of a decision tree $\hat{y}$.

A subset $\mathcal{T} \subseteq$ nodes$(\hat{y})$ of its nodes is called a **subtree**, if it is closed under parents (i.e., it contains for each node $T \in \mathcal{T}$ also its parent node).

A subtree $\mathcal{T}$ can be interpreted as decision tree, if for leaf nodes $T \in \mathcal{T}$ that have been interior nodes in the original tree $\hat{y}$, the split is replaced by a prediction:

$$T.\text{class} := \text{argmax}_{y'} |\{(x, y) \in X \mid y = y'\}|$$

where X denotes the subset of the training data that is passed down to node T.

Model Selection Measures

Cost Complexity Criterion: (minimize)

$$\text{CCC}(\hat{y}) := \text{RSS}(\hat{y}) + \lambda \cdot |\text{nodes}(\hat{y})|$$

where

$\hat{y}$ is a regression tree,

$\lambda \geq 0$ the **complexity parameter** / **regularization parameter**, and

$\text{nodes}(\hat{y})$ denotes the set of nodes of tree $\hat{y}$.

λ is a hyperparameter that needs to be estimated by cross-validation.

Note: many texts on decision trees use the symbol α instead of λ to denote the regularization parameter.

...

...

Popular Decision Tree Configurations

name	ChAID	CART	ID3	C4.5
author	Kass 1980	Breiman et al. 1984	Quinlan 1986	Quinlan 1993
selection measure	χ^2	Gini index, twoing index	information gain	information gain ratio
splits	all	binary nominal, binary quantitative, binary bivariate quantitative	complete	complete, binary nominal, binary quantitative
stopping criterion	χ^2 independence test	minimum number of cases/node	χ^2 independence test	lower bound on selection measure
pruning technique	none	error complexity pruning	pessimistic error pruning	pessimistic error pruning, error based pruning