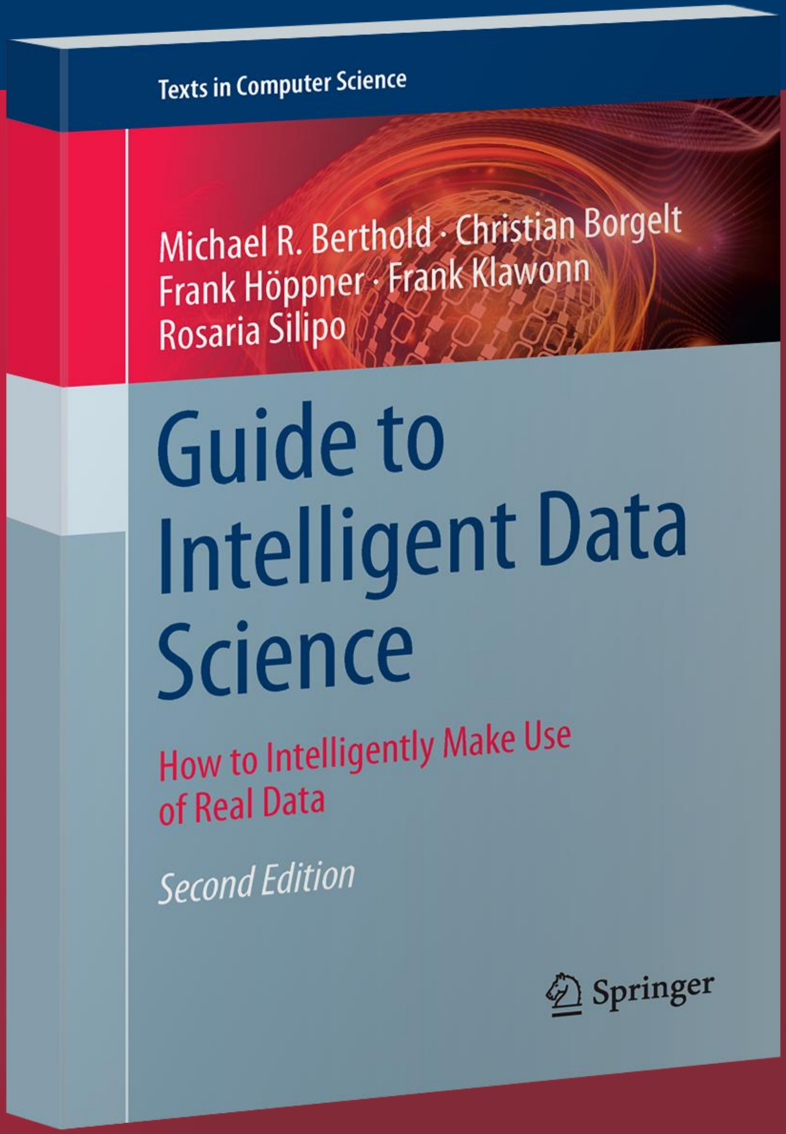
Decision and Regression Trees



Summary of this lesson

“Numbers have an important story to tell. They rely on you to give them a voice”
-Stephen Few

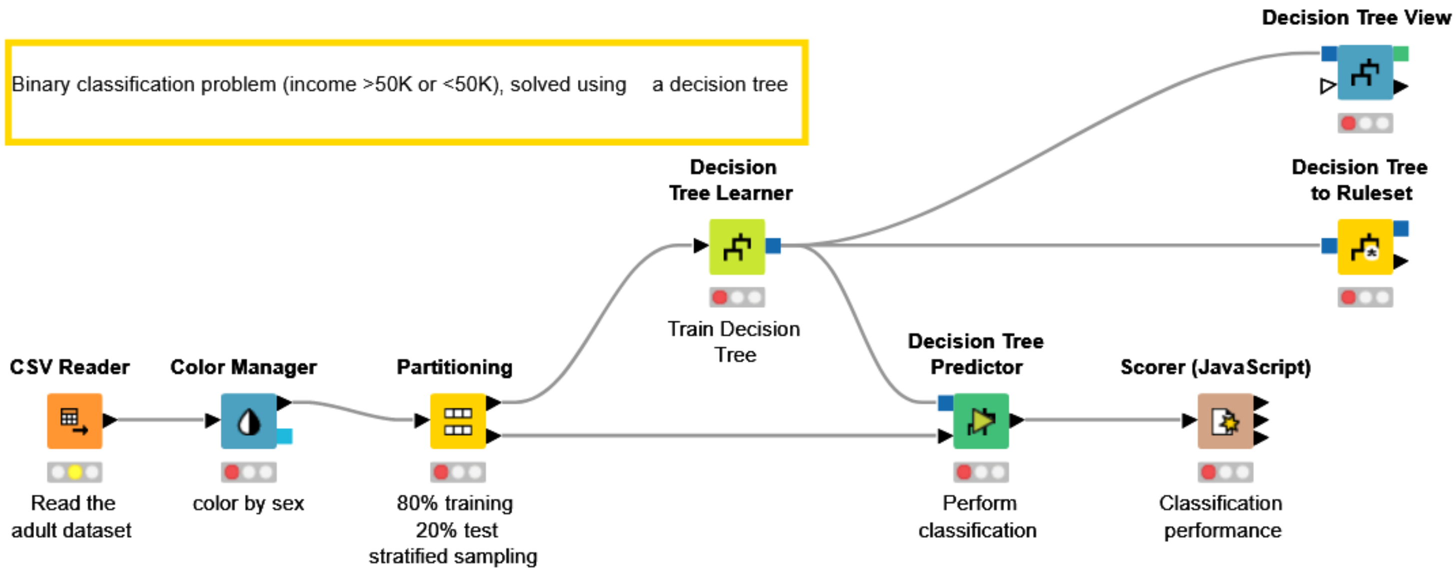
Can we *explain* the data?

**This lesson refers to chapter 8 of the GIDS book*

What you will learn

- Decision Trees
  - How to grow a decision tree
  - Possible evaluation measures
  - How to deal with numerical attributes
  - How to deal with Missing Values
- Pruning
  - Reduced Error Pruning
  - Pessimistic Pruning
  - Confidence Level Pruning
  - Minimum description length
- Regression Trees
  - Decision trees with numerical target

- Datasets used : adult dataset
- Example Workflows:
  - „Decision tree“ https://kni.me/w/PV-3WZ-ZquINMsI_
    - confusion matrix
    - accuracy measures



Supervised Learning

- A target attribute is predicted based on other attributes
- Assumption: in addition to the object description **x**, we have also the value for the target attribute **y**

Classification

- Categorical target attribute
- *Which category does a customer belong to?*

Regression

- Numerical target attribute
- *How much will a customer spend next month?*

Decision Trees

A bit of History of Decision Trees

1983

CART[1]
(Classification and Regression Trees)
- The leaves contain numeric constants
- Selection is based on the **Gini index**
- Produces only binary trees
- Can handle outliers

1986

ID3[2]
(Iterative DiChaudomiser 3)
- Greedy algorithm
- Chosen attributes and splits are based on the **information gain**
- Only nominal attributes allowed
- Missing values not supported

1993

C4.5[3]
ID3-based with improvements:
- Allowing continuous data
- Using missing values
- Using attributes with different weights
- Pruning tree after creation
- **Gain ratio** as selection measure

1997

See5/C5[3]
C4.5 improved
- Better performances
- Memory efficiency
- Smaller decision trees

1. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: CART: Classification and Regression Trees. Wadsworth, Belmont (1983)
2. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
3. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
4. Is See5/C5.0 Better Than C4.5? - https://www.rulequest.com/see5-comparison.html.

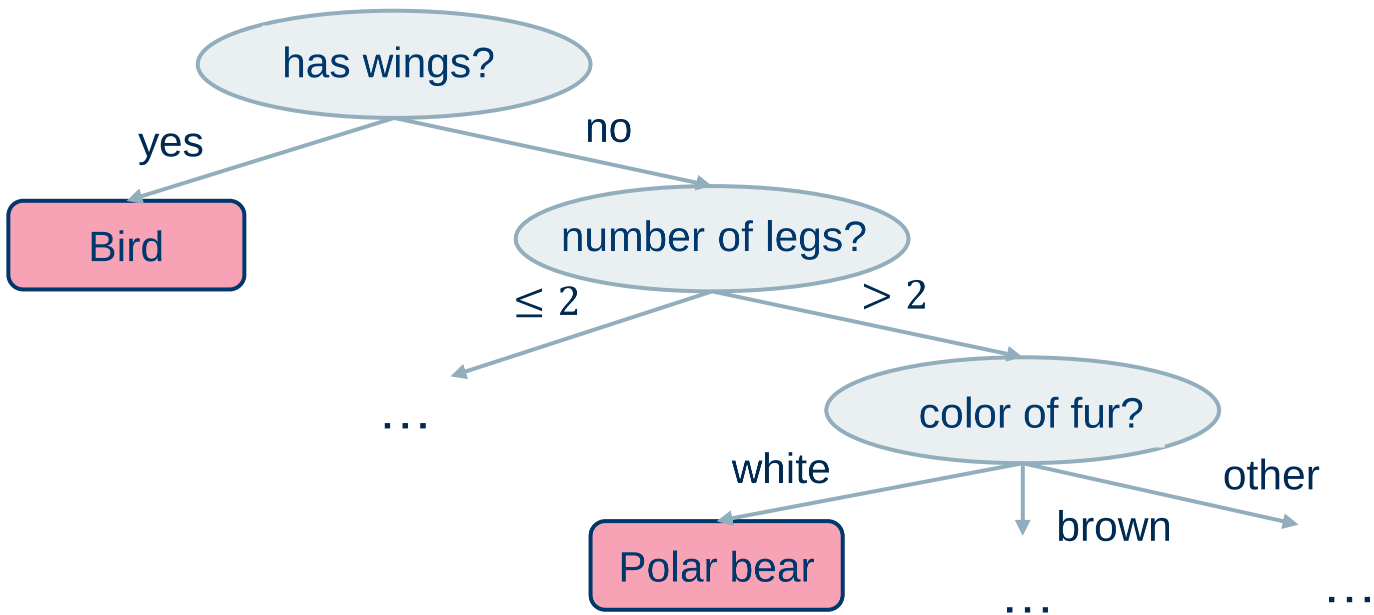
Decision Tree Algorithms comparison

	Splitting	Attributes	Missing values	Pruning	Outliers
ID3	Information Gain	Only Categorical	Not handled	No pruning	Susceptible
CART	Gini index / Towing Criteria	Categorical and Numeric	Handled	Cost-complexity pruning	Handled
C4.5	Gain Ratio	Categorical and Numeric	Handled	Error-based pruning	Susceptible

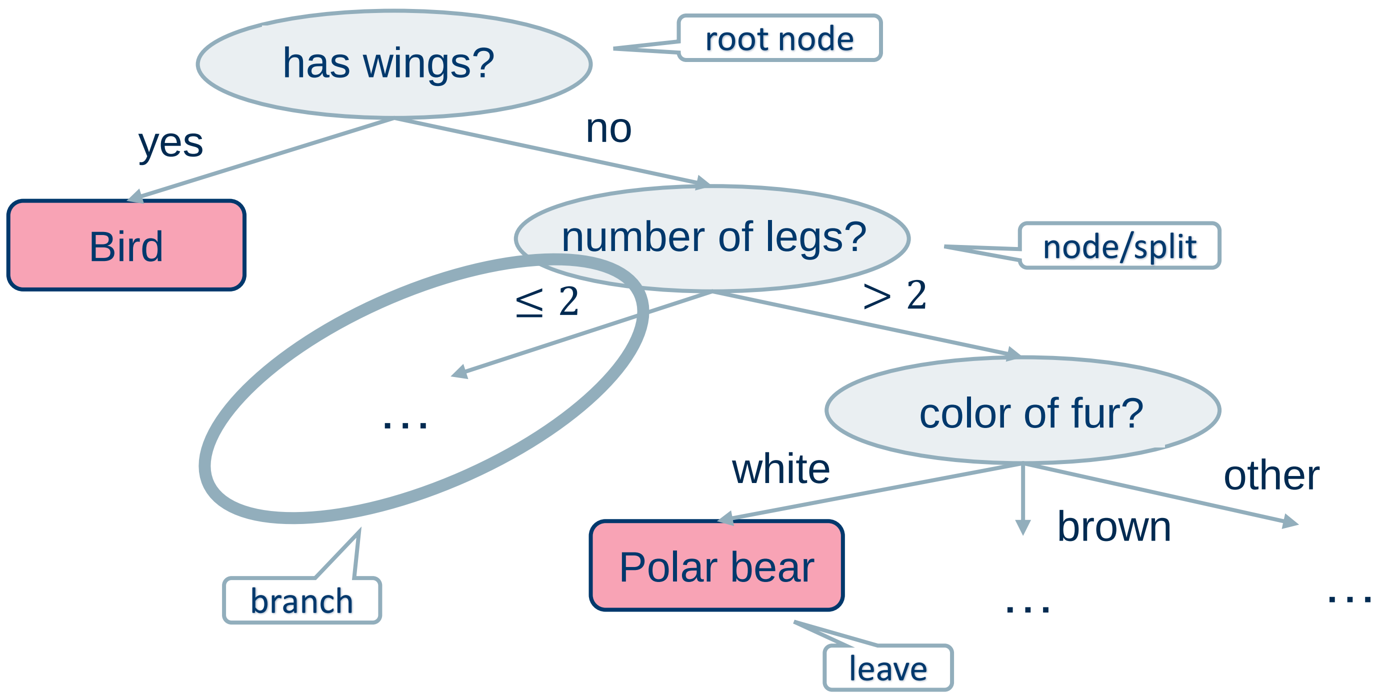
Source: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.685.4929&rep=rep1&type=pdf

- Decision trees aim to find a **hierarchical structure** to explain how different areas in the input space correspond to different outcomes.
- Useful for data with **a lot** of attributes of **unknown importance**
- The final decision tree often only uses a **small subset** of the available set of attributes ⇒ **Easy interpretation**
- They tend to be insensitive to normalization issues and **tolerant** toward many correlated or noisy attributes.
- Decision trees can reveal unexpected dependencies in the data which would otherwise be hidden in a more complex model.

A simple example for a decision tree to classify animals



A simple example for a decision tree to classify animals



Splits

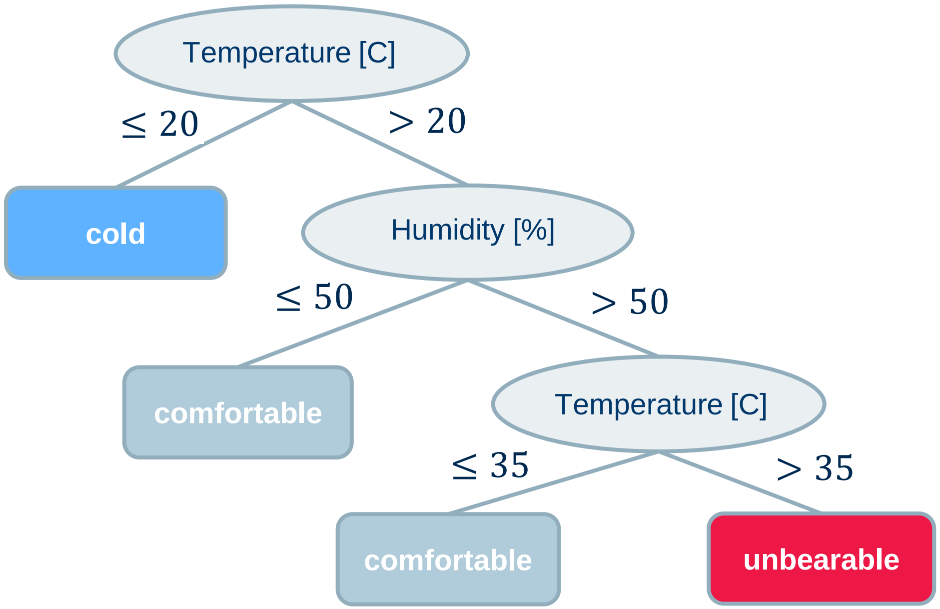
There are three options for splits in a Decision Tree

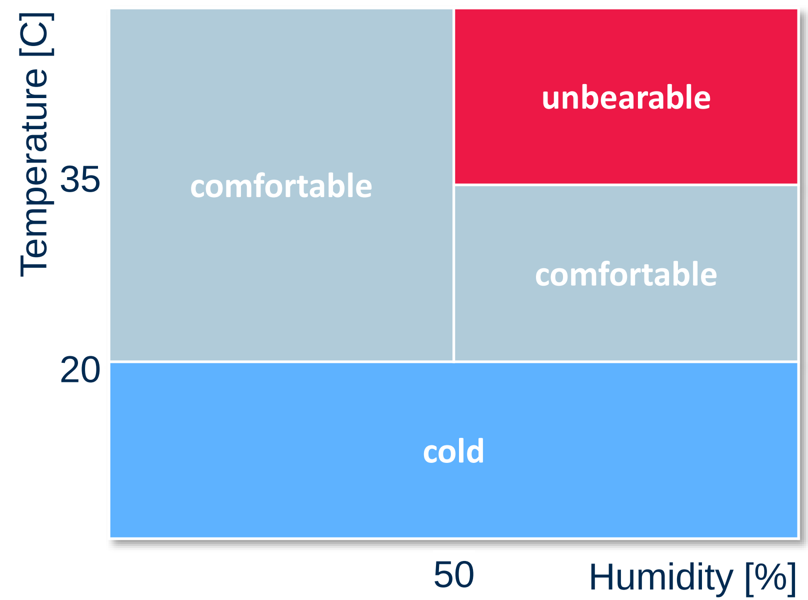
- **Boolean** splits: e.g. *married? yes/no*
- **Nominal** splits based on categorical attributes e.g. *color? red/blue/brown…*
- **Continuous** attributes splits based on numerical attributes e.g. *age?* ≤ 25 / > 25

– Each split divides the remaining space in two or more disjoint sub-partitions
– The resulting hierarchical structure is easy to read and interpret

- **Top-down** construction approach
  - Build the decision tree from top to bottom, from root to leaves
- **Greedy Selection** of a test attribute
  - Compute an evaluation measure for all attributes
  - Select the attribute with the best evaluation
- **Recursive Partitioning**
  - Divide the example cases according to the values of the test attribute
  - Apply the procedure recursively to subsets
  - Terminate the recursion if:
    - All cases belong to the same class
    - No more test attributes are available

Partitioning example



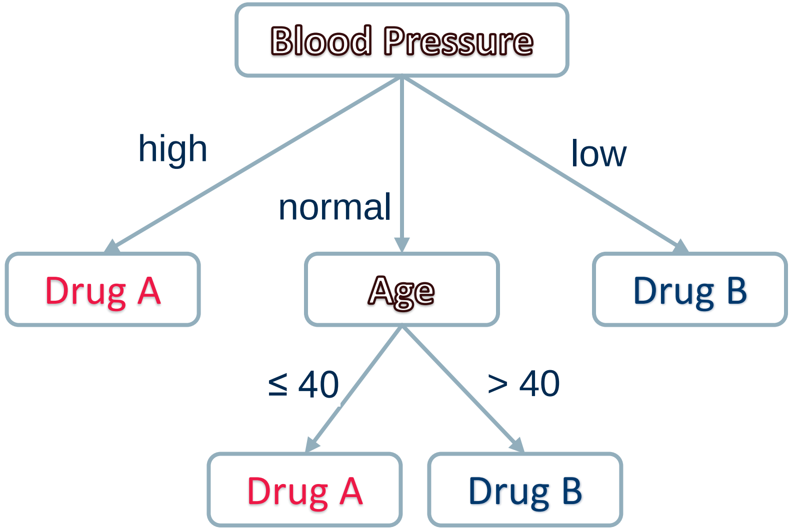


A simple example for a decision tree (left) using two numerical attributes and the corresponding partitioning (right) of the underlying feature space

Decision Trees: An Example

Another Example: Assignment of Drug to a Patient

Let‘s see how we constructed this tree



Building a tree: the baseline

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

Patient Database

12 example cases

- 3 descriptive attributes
- 1 class target

Building a tree: the baseline

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

Patient Database
12 example cases
  3 descriptive attributes
  1 class target

Assignment of drug
(without attributes)
Always assign majority drug
50% correct (6 /12 cases)

- **Sex** of the patient
  - Male vs. Female

Assignment of Drug

Male:	50% correct	(3/6 cases)
Female:	50% correct	(3/6 cases)
Total:	**50% correct**	(6/12 cases)

Not that much of an improvement

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

- Let‘s try with **Blood Pressure**
  - High vs. Normal vs. Low

Assignment of Drug

High:	100% correct	(3/3 cases)
Normal:	50% correct	(3/6 cases)
Low:	100% correct	(3/3 cases)
Total:	**75% correct**	(9/12 cases)

Better!

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

Building a tree: selecting test attributes

- Let‘s try with **Age**
  - Sort according to Age
  - Find best split on Age (ca. 40 years)

Assignment of Drug

<40:	67% correct	(4/6 cases)
>40:	67% correct	(4/6 cases)
Total:	**67% correct**	(8/12 cases)

Better, but not so much

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

Another Example: Assignment of Drug to a Patient

Let‘s stick to
Blood Pressure ...

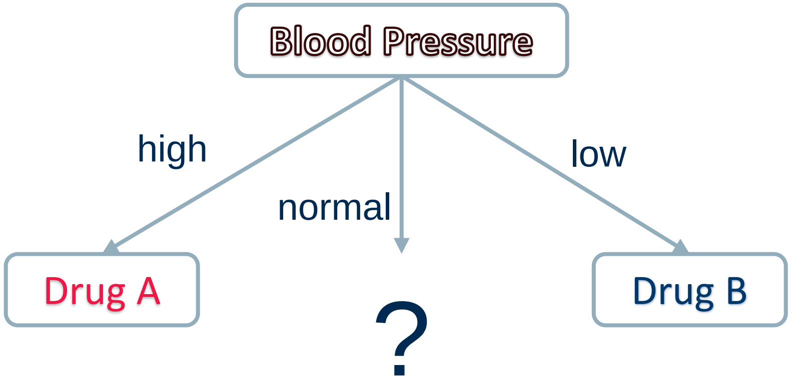
How do we determine which of the attributes is the “best” one?

Here we use **Class Frequency** as Evaluation Measure

Simple to compute, easy to understand

It works well only on two classes

We can do better ...



... and so on on the remaining data

Decision Tree construction algorithm

Algorithm BuildDecisionTree($\mathcal{D}, \mathcal{A}$)

input: training data $\mathcal{D}$, set $\mathcal{A}$ of available attributes
output: a decision tree matching $\mathcal{D}$, using all or a subset of $\mathcal{A}$

if all elements in $\mathcal{D}$ belong to one class
    return node with corresponding class label
elseif $\mathcal{A} = \emptyset$
    return node with majority class label in $\mathcal{D}$
else
    select attribute $A \in \mathcal{A}$ which best classifies $\mathcal{D}$
    create new node holding decision attribute A
    for each split v_A of A
        add new branch below with corresponding test for this split
        create $\mathcal{D}(v_A) \subset \mathcal{D}$ for which split condition holds
        if $\mathcal{D}(v_A) = \emptyset$
            return node with majority class label in $\mathcal{D}$
        else
            add subtree returned by calling
                BuildDecisionTree($\mathcal{D}(v_A), \mathcal{A} \backslash \{A\}$)
        endif
    endfor
    return node.
endif

- **Greedy strategy**
- Starting from the root node (top-down approach), recursively find the best split at each point
- The recursion stops if:
  - a subset of only one class objects is encountered, or
  - no further attributes for splits are available ⇒ in this case the leaf node is labelled with the majority class

How do we determine which of the attributes is the “best” one?

Which kind of splits are possible?

ID3 Learning Algorithm

Which kind of splits are possible?

- For nominal values:
  - Binary splits, some attributes on one side some on the other side
  - N-splits, ultimately creating as many branches as there exist attribute values

- **ID3** learning algorithm:
  - Considers only nominal attributes
  - Only N-splits into all possible values of an attribute within a single node
  - Each node has only one cut choice for splitting -> easier to implement

Decision Trees: Evaluation Measures

How do we determine which of the attributes is the “best” one?

- **Information Gain** (Kullback/Leibler 1951, Quinlan 1986)
- **Gain Ratio** (Quinlan 1986 / 1993)
- **Gini Index**
- χ^2 **Measure**

Shannon Entropy

- Intuition: the best split lets us create leaves as soon as possible
- Leaves are created if all the patterns of a branch belong to the same class
- We have to create subsets containing mostly data from one class ⇒ measure of **impurity** of the subset using the **Shannon entropy**

$$H_{\mathcal{D}}(\mathcal{C}) = -\sum_{k \in dom(\mathcal{C})} \frac{|\mathcal{D}_{\mathcal{C}=k}|}{|\mathcal{D}|} \log_2 \frac{|\mathcal{D}_{\mathcal{C}=k}|}{|\mathcal{D}|}$$

where $\mathcal{D}_{\mathcal{C}=k}$ is the number of data in class $\mathcal{C} = k$, with $0 \log 0 = 0$

Another Example: Predicting sex

ID	Height	Weight	Long Hair	Sex
1	medium	normal	no	male
2	short	low	yes	female
3	tall	high	no	male
4	short	normal	yes	female
5	tall	normal	yes	female
6	short	low	no	female
7	short	high	no	male
8	medium	normal	no	female
9	medium	low	yes	female
10	tall	normal	no	male

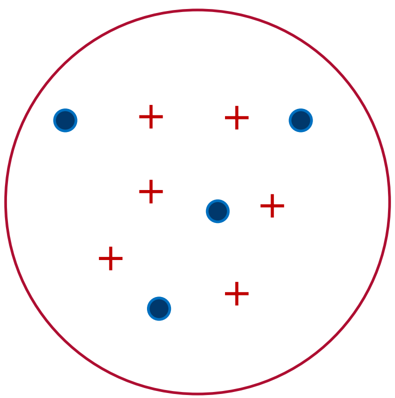
Shannon Entropy

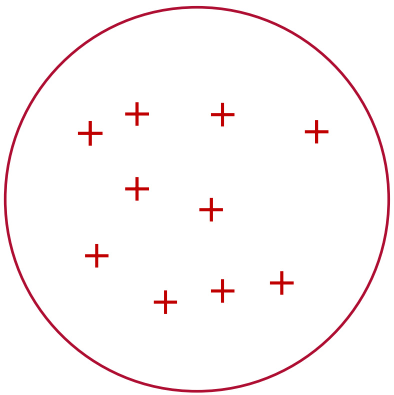
Based on Shannon entropy = measure for information / uncertainty

$$Entropy\ (p) = -\sum_{i=0}^{n} p_i \log_2 p_i \text{ for } p \in \mathbb{Q}^n$$

$p_1 = {}^{6}/_{10}$

$p_2 = {}^{4}/_{10}$





$p_1 = {}^{10}/_{10} = 1$

$p_2 = {}^{0}/_{10} = 0$

$$Entropy\ (p) = -\left({}^{6}/_{10} \log_2\left({}^{6}/_{10}\right) + {}^{4}/_{10} \log_2\left({}^{4}/_{10}\right)\right)$$
$$\sim 0{,}9710$$

$$Entropy\ (p) = -\left({}^{10}/_{10} \log_2\left({}^{10}/_{10}\right) + {}^{0}/_{10} \log_2\left({}^{0}/_{10}\right)\right)$$
$$= 0$$

Note. Shannon entropy ranges between 0 and 1

- 0 when all data points belong to one class
- 1 when data points are evenly distributed across all classes

- $H_{\mathcal{D}}(sex) \sim 0.9710$
- The best attribute to split on, is the attribute producing the biggest reduction in entropy from the original set.
- This reduction is called **Information Gain**

Entropy Reduction

$$I_{\mathcal{D}}(\mathcal{C}, A) = H_{\mathcal{D}}(\mathcal{C}) - H_{\mathcal{D}}(\mathcal{C}, A)$$

Expected Entropy after split on A

Entropy original dataset

Information Gain

- $H_{\mathcal{D}}(\mathcal{C}, A)$ is the entropy left after the split on attribute A, as the weighted sum of the entropy in the generated subsets

$$H_{\mathcal{D}}(\mathcal{C}, A) = \sum_{a \in dom(A)} \frac{|\mathcal{D}_{\mathcal{A}=a}|}{|\mathcal{D}|} H_{\mathcal{D}_{A=a}}(\mathcal{C})$$

- $\mathcal{D}_{\mathcal{A}=a}$ are the subsets of $\mathcal{D}$ where attribute A has value a
- $a \in dom(A)$ are all values in attribute A

Another Example: Predicting sex

ID	Height	Weight	Long Hair	Sex
1	medium	normal	no	**male**
2	**short**	low	yes	**female**
3	tall	high	no	**male**
4	**short**	normal	yes	**female**
5	tall	normal	yes	**female**
6	**short**	low	no	**female**
7	**short**	high	no	**male**
8	medium	normal	no	**female**
9	medium	low	yes	**female**
10	tall	normal	no	**male**

$$Entropy_1 = Entropy\left(\frac{3}{4}, \frac{1}{4}\right) \qquad w_1 = {}^{4}/_{10}$$

Another Example: Predicting sex

ID	Height	Weight	Long Hair	Sex
1	medium	normal	no	male
2	short	low	yes	female
3	tall	high	no	male
4	short	normal	yes	female
5	tall	normal	yes	female
6	short	low	no	female
7	short	high	no	male
8	medium	normal	no	female
9	medium	low	yes	female
10	tall	normal	no	male

$$Entropy_2 = Entropy\left(\frac{2}{3}, \frac{1}{3}\right) \qquad w_2 = {}^{3}/_{10}$$

Another Example: Predicting sex

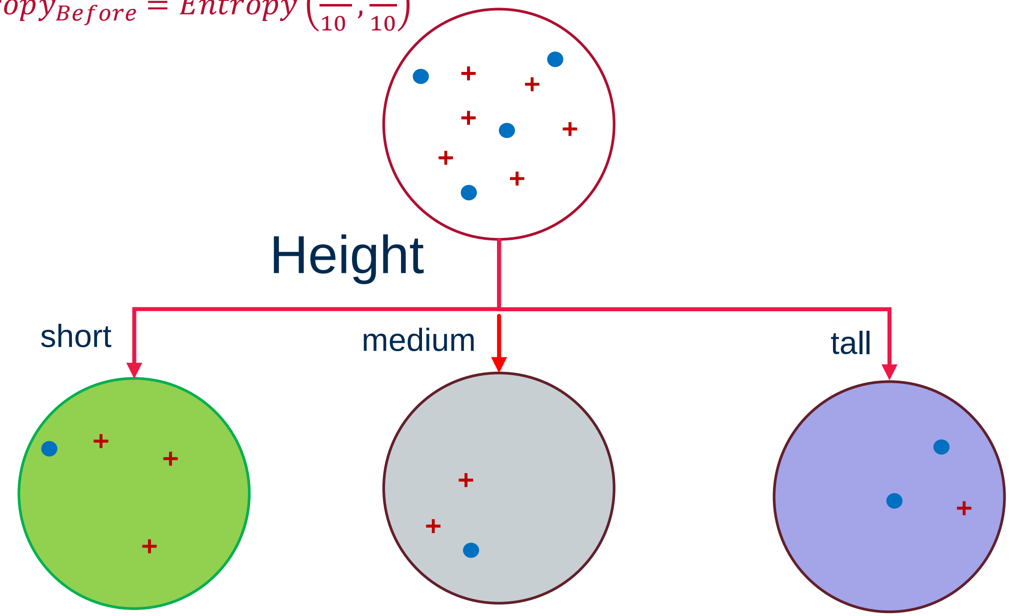
ID	Height	Weight	Long Hair	Sex
1	medium	normal	no	male
2	short	low	yes	female
3	tall	high	no	male
4	short	normal	yes	female
5	tall	normal	yes	female
6	short	low	no	female
7	short	high	no	male
8	medium	normal	no	female
9	medium	low	yes	female
10	tall	normal	no	male

$$Entropy_3 = Entropy\left(\frac{1}{3}, \frac{2}{3}\right) \qquad w_3 = {}^{3}/_{10}$$

Split Criterion: Information Gain

$$Entropy_{Before} = Entropy\left(\frac{6}{10}, \frac{4}{10}\right)$$

$$H(Sex|Height) = \frac{4}{10}\ Entropy_1 + \frac{3}{10} Entropy_2 + \frac{3}{10} Entropy_3 \sim 0.8755$$



$$Entropy_1 = Entropy\left(\frac{3}{4}, \frac{1}{4}\right)$$
$$w_1 = {}^4/_{10}$$

$$Entropy_2 = Entropy\left(\frac{2}{3}, \frac{1}{3}\right)$$
$$w_2 = {}^3/_{10}$$

$$Entropy_3 = Entropy\left(\frac{1}{3}, \frac{2}{3}\right)$$
$$w_3 = {}^3/_{10}$$

$$Inf.Gain = Entropy_{Before} - Entropy_{After}$$

$$Inf.Gain = 0.9710\ - 0.8755 = 0.0955$$

Next splitting:
Feature with
highest $Inf.Gain$

Another Example: Predicting sex

ID	Height	Weight	Long Hair	Sex
1	medium	normal	no	male
2	short	low	yes	female
3	tall	high	no	male
4	short	normal	yes	female
5	tall	normal	yes	female
6	short	low	no	female
7	short	high	no	male
8	medium	normal	no	female
9	medium	low	yes	female
10	t	normal	no	male

0

0

$$H(Sex|Weight) = \frac{3}{10}\, Entropy_1 + \frac{5}{10} Entropy_2 + \frac{2}{10} Entropy_3 \sim 0.4855$$

$$Inf.Gain = 0.9710\ - 0.4855 = 0.4855$$

Another Example: Predicting sex

ID	Height	Weight	Long Hair	Sex
1	medium	normal	no	**male**
2	short	low	yes	**female**
3	tall	high	no	**male**
4	short	normal	yes	**female**
5	tall	normal	yes	**female**
6	short	low	no	**female**
7	short	high	no	**male**
8	medium	normal	no	**female**
9	medium	low	yes	**female**
10	tall	rmal	no	**male**

0

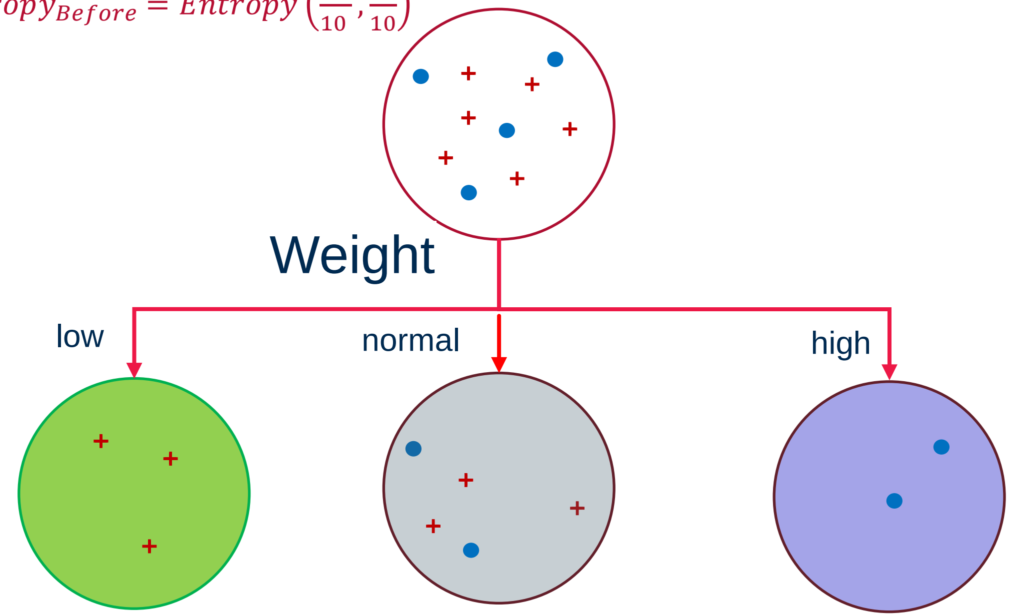
$$H(Sex|Long\ Hair) = \frac{4}{10}\ Entropy_1 + \frac{6}{10} Entropy_2 \sim 0.5510$$

$$Inf.Gain = 0.9710\ - 0.5510 = 0.4200$$

Feature with highest information gain: Weight

$$H(Sex|Weight) = \frac{3}{10}\,Entropy_1 + \frac{5}{10}Entropy_2 + \frac{2}{10}Entropy_3 \sim 0.4855$$

$$Entropy_{Before} = Entropy\left(\frac{6}{10}, \frac{4}{10}\right)$$



$$Entropy_1 = Entropy\left(\frac{0}{3}, \frac{3}{3}\right)$$
$$w_1 = {}^{3}/_{10}$$

$$Entropy_2 = Entropy\left(\frac{2}{5}, \frac{3}{5}\right)$$
$$w_2 = {}^{5}/_{10}$$

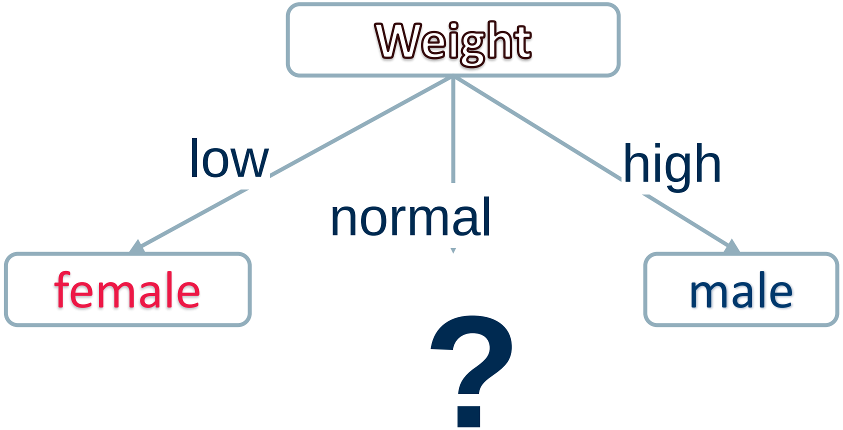
$$Entropy_3 = Entropy\left(\frac{0}{2}, \frac{2}{2}\right)$$
$$w_3 = {}^{2}/_{10}$$

$$Inf.Gain = Entropy_{Before} - Entropy_{After}$$

$$Inf.Gain = 0.9710\ - 0.4855 = 0.4855$$

Feature with highest $Inf.Gain$: **Weight**

Root node splitting on Weight



Repeat for

Weight = „normal“

Remaining Table to be considered

ID	Height	~~Weight~~	Long Hair	Sex
1	medium	~~normal~~	no	**male**
4	short	~~normal~~	yes	**female**
5	tall	~~normal~~	yes	**female**
8	medium	~~normal~~	no	**female**
10	tall	~~normal~~	no	**male**

$$H(Sex|Weight = normal) = -\left({}^{2}/_{5}\,\log_2({}^{2}/_{5})\; + \;{}^{3}/_{5}\,\log_2({}^{3}/_{5})\right) \sim 0.9710$$

Find best split for Weight=„normal“

ID	Height	Long Hair	Sex
1	medium	no	male
4	short	yes	female
5	tall	yes	female
8	medium	no	female
10	tall	no	male

$$H(Sex|Weight = normal, Height) = \frac{1}{5}\ Entropy_1 + \frac{2}{5} Entropy_2 + \frac{2}{5} Entropy_3 = \ 0.8$$

$$Inf.Gain = 0.9710\ - 0.8 = 0.1710$$

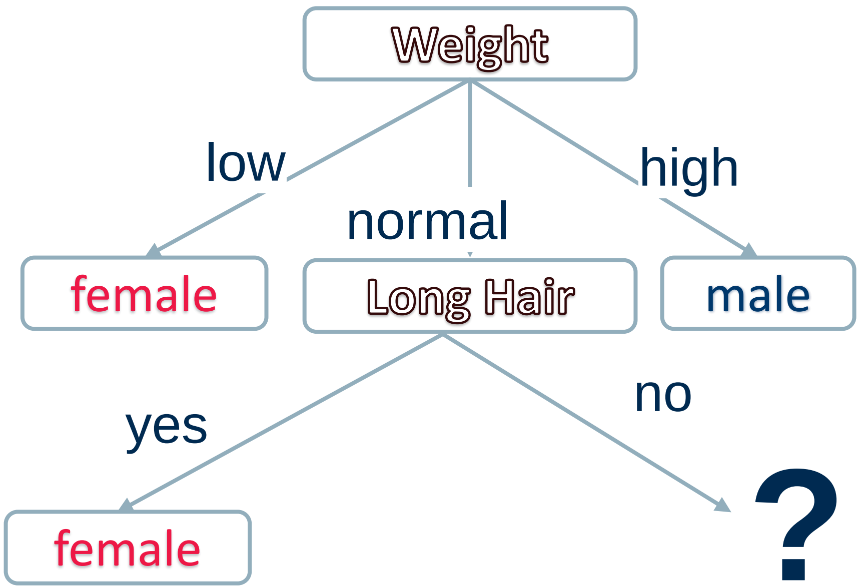
Remaining Table to be considered

ID	Height	Long Hair	Sex
1	medium	no	**male**
4	short	yes	**female**
5	tall	yes	**female**
8	medium	no	**female**
10	tall	no	**male**

$$H(Sex|Weight = normal, Long\ Hair) = \frac{2}{5}\ Entropy_1 + \frac{3}{5} Entropy_2 = \ 0.5510$$

$$Inf.Gain = 0.9710\ - 0.5510 = 0.4200$$

One more node on Long Hair



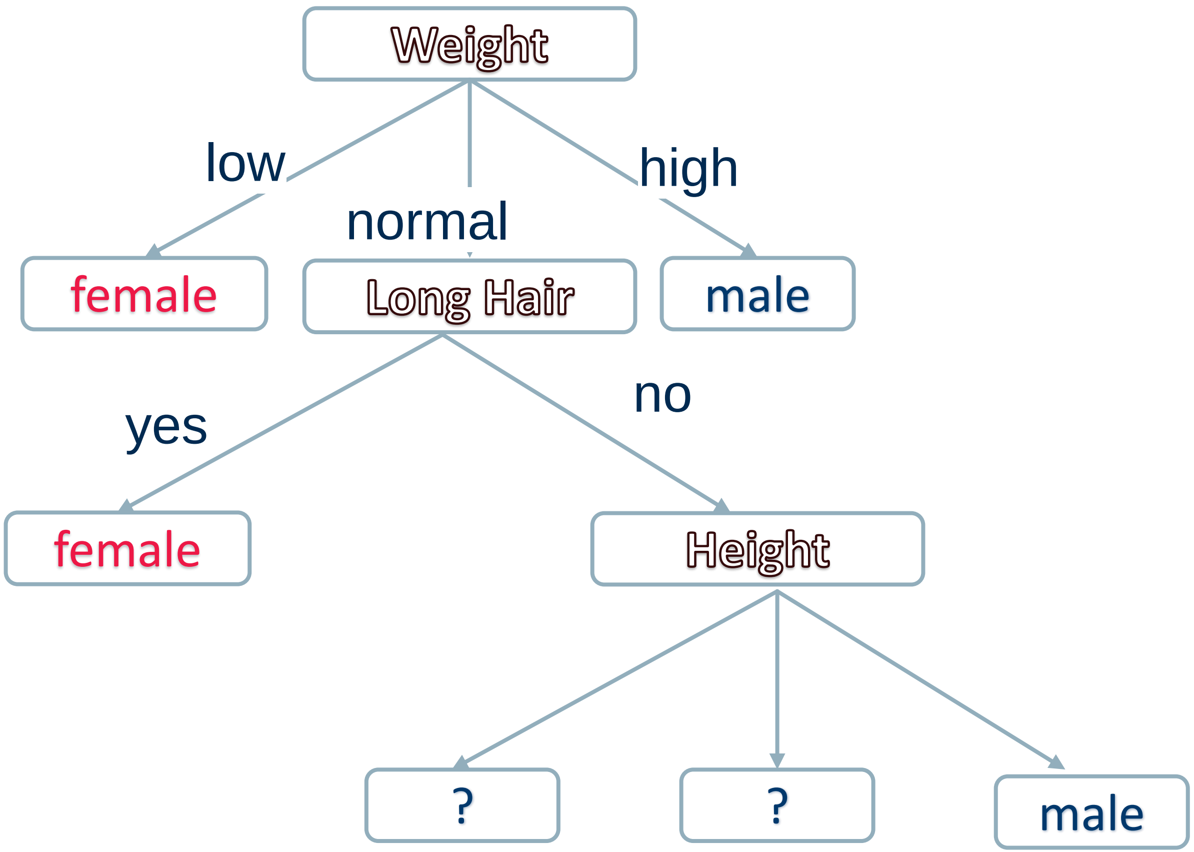
Repeat for
Weight = „normal“,
Long Hair = „no“

For the remaining node, only the attribute *Height* is left with the remaining data table:

ID	Height	~~Long Hair~~	Sex
1	medium	~~no~~	**male**
8	medium	~~no~~	**female**
10	tall	~~no~~	**male**

Therefore the resulting decision tree is:

Final Decision Tree



Other Evaluation Measures from Information Theory

- **Problem:** Information Gain favors attributes with many unique values, i.e. of two attributes having about the same information content it tends to select the one having more values.

- What happens if we use a unique ID attribute for the splits?
  - The information gain is maximized for the ID feature
  - Each ID results in its own branch (one split per value/object)
  - In most application this result is useless

- **Solutions**
  - Gain Ratio
  - Gini Index

Gain Ratio

– **Normalization** removes this bias towards attributes with many values

– Compute the **split information**

$$SI_{\mathcal{D}}(A) = -\sum_{a \in dom(A)} \frac{|\mathcal{D}_{A=a}|}{|\mathcal{D}|} \log \frac{|\mathcal{D}_{A=a}|}{|\mathcal{D}|}$$

Entropy distribution for subset $\mathcal{D}_{A=a}$

– Normalize information gain with $SI_{\mathcal{D}}(A)$ to obtain the **gain ratio** for a given split

$$GR_{\mathcal{D}}(\mathcal{C}, A) = \frac{I_{\mathcal{D}}(\mathcal{C}, A)}{SI_{\mathcal{D}}(A)}$$

Information Gain

Maximizing GR favors splits on few values

Normalization by Split Information

- Gini Index is based on Gini Impurity:

$$I_{Gini}(C) = 1 - \sum_{i=1}^{n} p_i^2$$

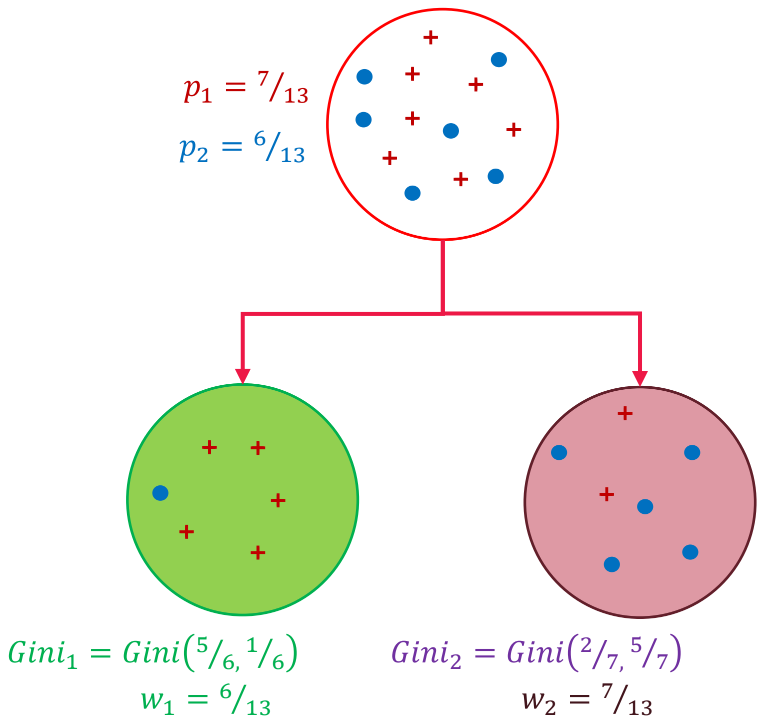
where $p_i \in \mathbb{Q}^n$ is the frequency of class i and n is the number of classes

- Gini Index can be interpreted as the expected error rate

$$I_{Gini\_Gain}(C) = I_{Gini}(C) - I_{Gini}(C|A)$$

$$I_{Gini\_Gain}(C,A) = \left(1 - \sum_{i=1}^{n_C} p_{i.}^2\right) - \sum_{j=1}^{n_A} p_{.j}^2 \left(1 - \sum_{i=1}^{n_C} p_{i|j}^2\right)$$

Possible Split Criterion: Gini Index



Gini index is based on Gini impurity:

$$Gini(p) = 1 - \sum_{i=1}^{n} p_i^2 \quad \text{for } p \in \mathbb{Q}^n$$

$$Gini(p) = 1 - \frac{7^2}{13^2} - \frac{6^2}{13^2}$$

Split criterion:

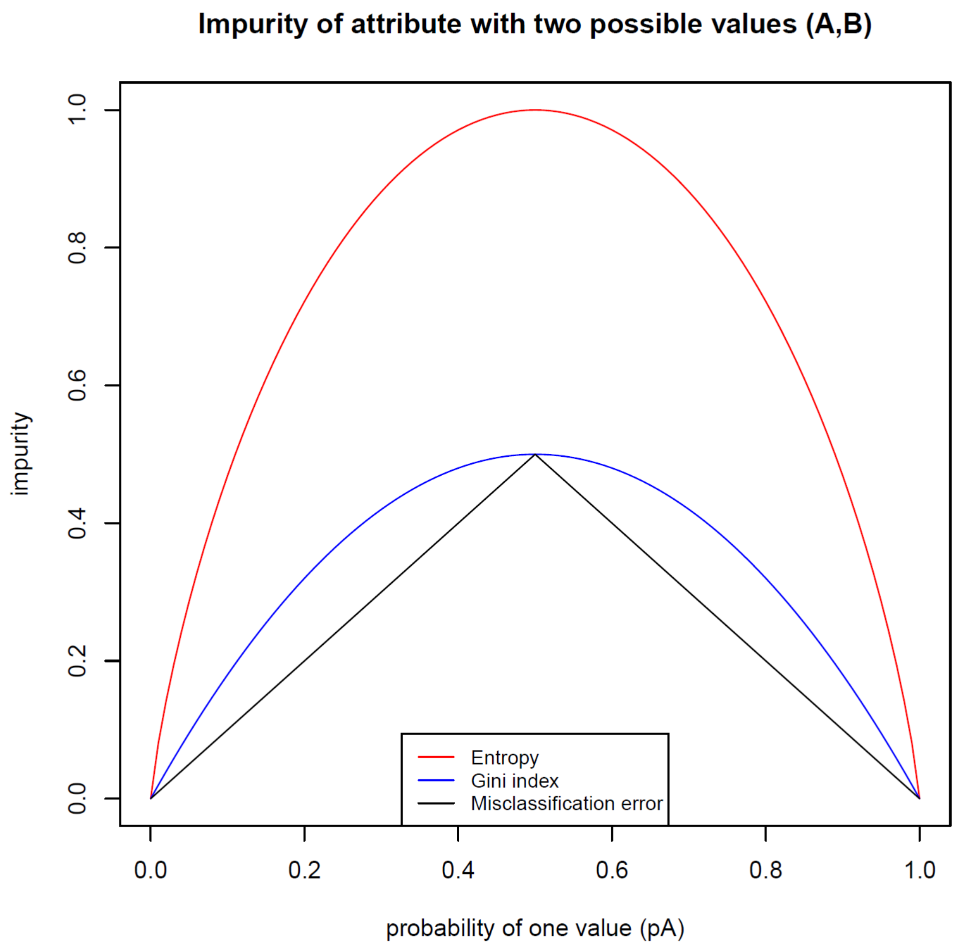
$$Gini_{Index} = \sum_{i=1}^{n} w_i Gini_i$$

$$Gini_{Index-after} = \frac{6}{13} Gini_1 + \frac{7}{13} Gini_2$$

Next splitting feature:

Feature with lowest $Gini_{Index}$

Comparison of Impurity Measures



χ^2 Measure

- Compares the actual joint distribution with a **hypotheitcal independent distribution**
- Uses absolute comparison
- Can be interpreted as a difference measure

$$\chi^2(C|A) = \sum_{i=1}^{n_C} \sum_{j=1}^{n_A} N_{..} \frac{\left(p_{i.}p_{.j} - p_{ij}\right)^2}{p_{i.}p_{.j}}$$

Decision Trees: Numerical Attributes

Discretization

- **Preprocessing I**
  - Form equally sized or equally populated intervals (**binning**)

- **Preprocessing II / Multi-splits during tree construction**
  - Build a decision tree using only the numeric attribute.
  - Flatten the tree to obtain a multi-interval discretization.
  - Let's see …

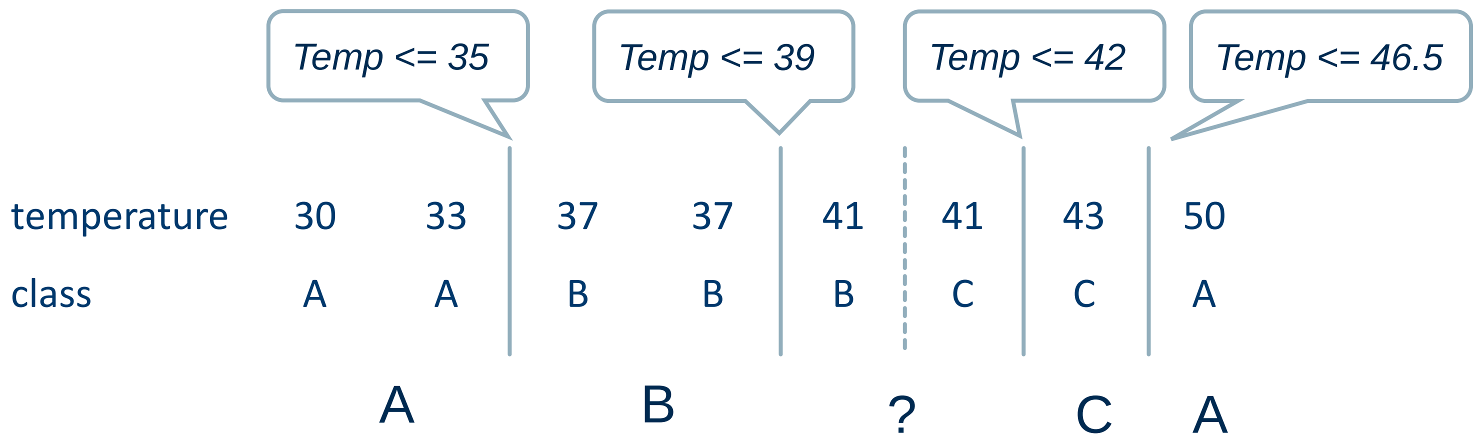
Splitting on Numerical Attributes

- ID3 only focuses on nominal values
- Introduce splits on numeric values
- We only care about splits occurring within class boundaries (i.e. it makes no sense to split a uniform subset into smaller ones)
- Instances are sorted by temperature values
- Three classes: A, B, C

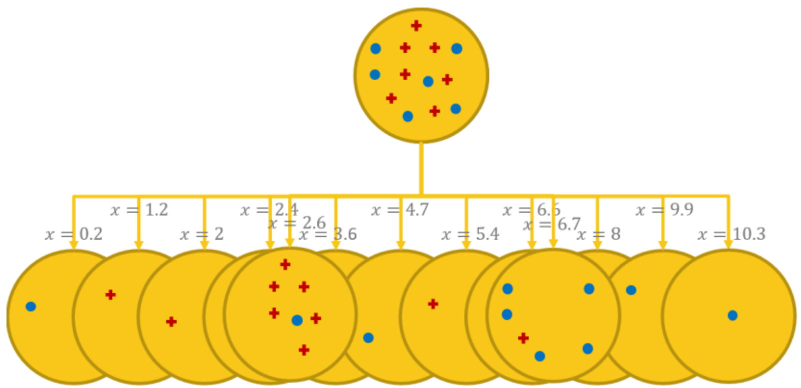
temperature	30	33	37	37	41	41	43	50
class	A	A	B	B	B	C	C	A

Splitting on Numerical Attributes

- Possible splits over temperature numerical values at boundary points
- Note that we cannot split on the dashed line since the same value holds for both classes.
- Define a binary variable for each possible split

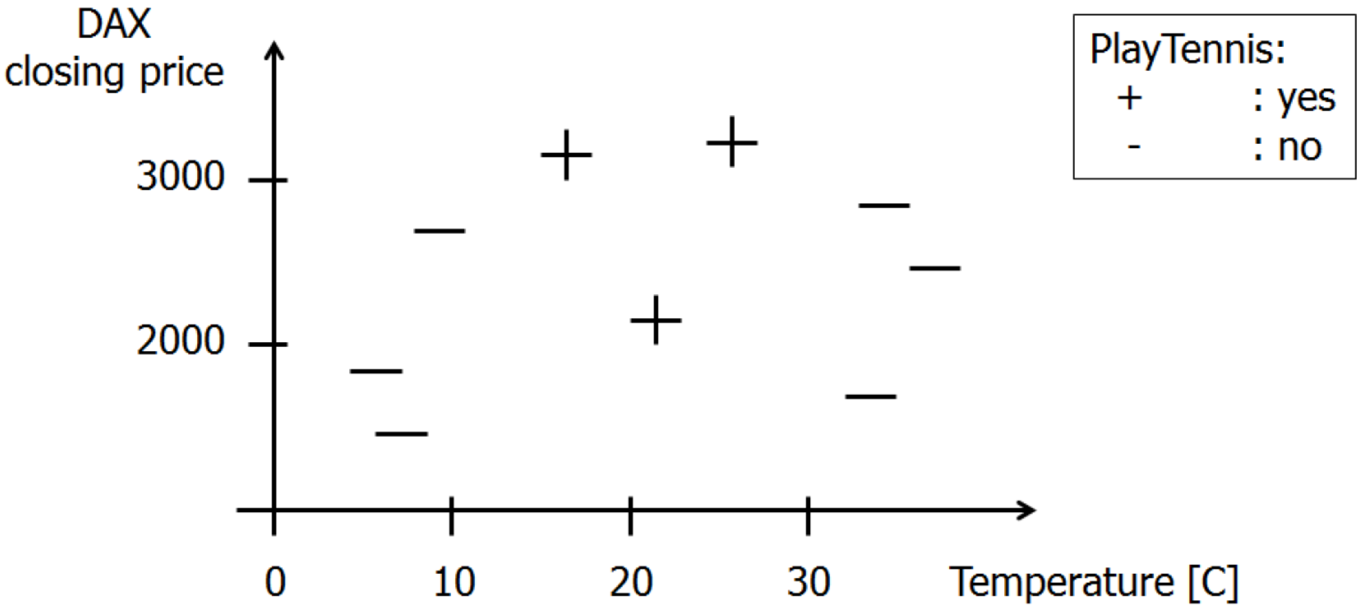


- Splits at boundary points minimize entropy.
- For binary splits (only one cut point) all boundary points are considered and the one with the smallest entropy is chosen.
- For multiple splits a recursive procedure is applied.

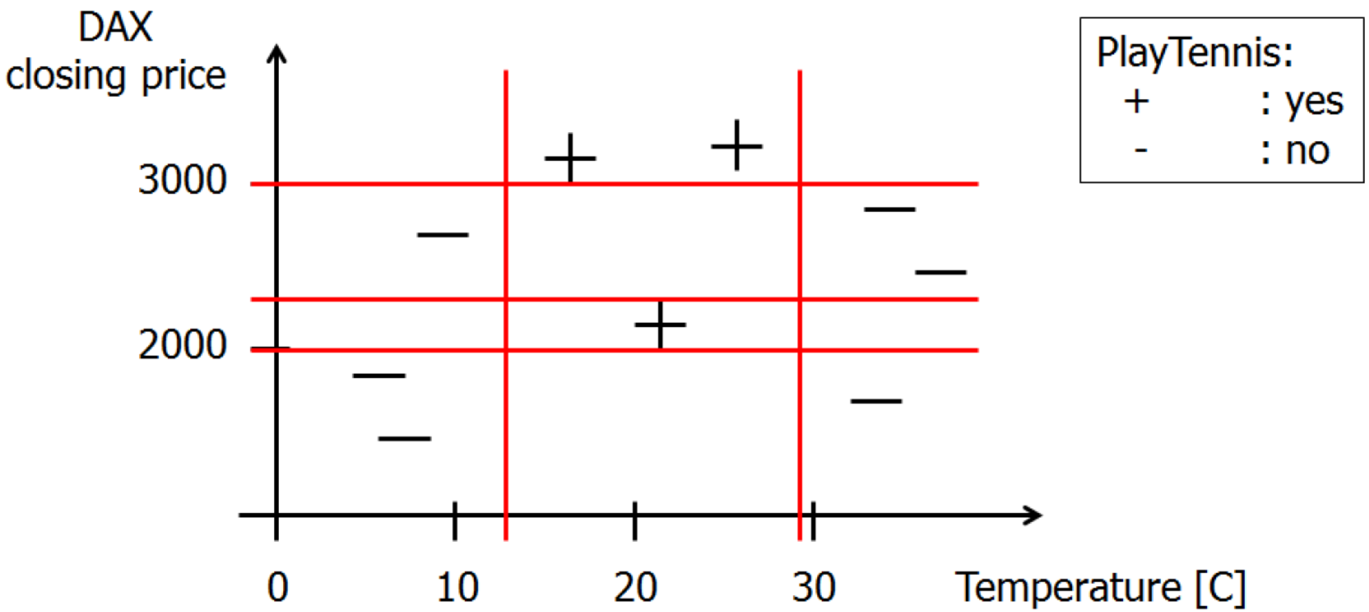


- Better restrict splits on numerical attributes to binary splits

– The Data


DAX
closing price
3000
2000
0
10
20
30
Temperature [C]
PlayTennis:
+ : yes
- : no

– The Solution

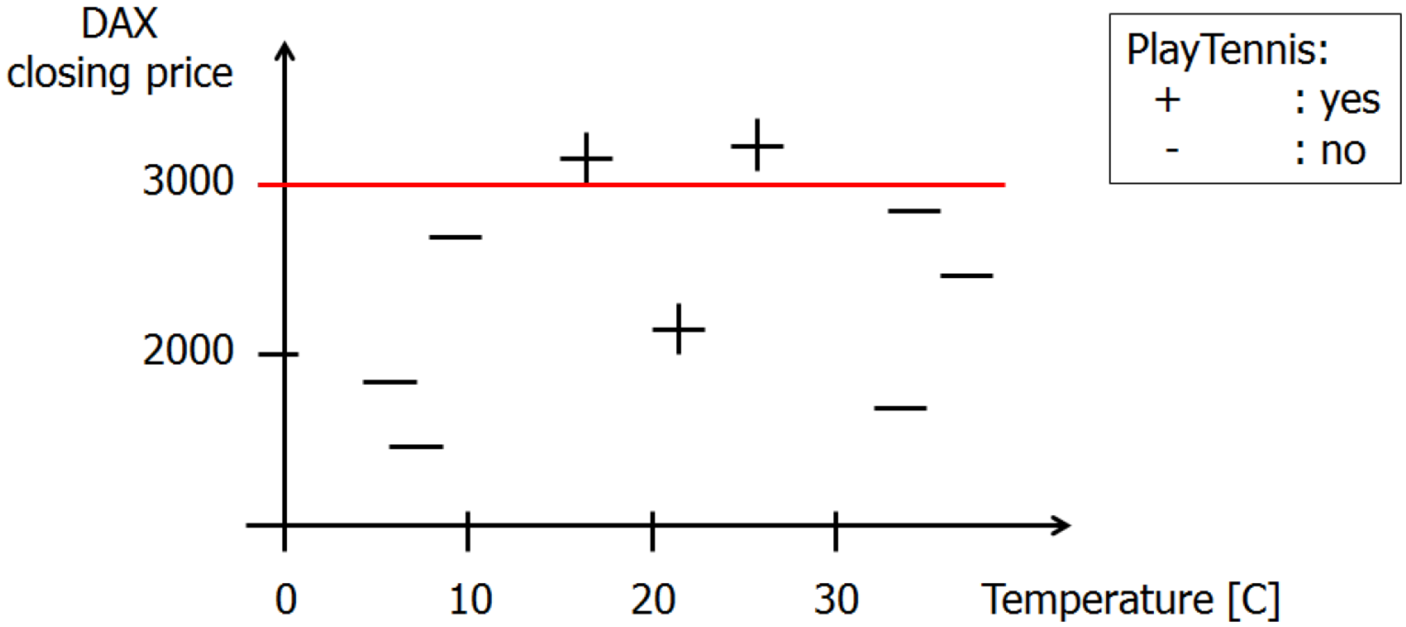


Example of Greedy Approach

- $H(S) = H(3+, 6-) = 0.918$
- $I_{Gain}(DAX > 3000) = H(S) - \left(\frac{2}{9} H(2+, 0-) + \frac{7}{9} H(1+, 6-)\right) = 0.918 - \left(\frac{2}{9} * 0.0 + \frac{7}{9} * 0.592\right) = 0.458$ 1
- $I_{Gain}(DAX > 2250) = H(S) - \left(\frac{5}{9} H(2+, 3-) + \frac{4}{9} H(1+, 3-)\right) = 0.918 - \left(\frac{5}{9} * 0.971 + \frac{4}{9} * 0.811\right) = 0.018$
- $I_{Gain}(DAX > 2000) = H(S) - \left(\frac{6}{9} H(3+, 3-) + \frac{3}{9} H(0+, 3-)\right) = 0.918 - \left(\frac{6}{9} * 1.0 + \frac{3}{9} * 0.0\right) = 0.251$
- $I_{Gain}(temp > 12) = H(S) - \left(\frac{6}{9} H(3+, 3-) + \frac{3}{9} H(0+, 3-)\right) = 0.918 - \left(\frac{6}{9} * 1.0 + \frac{3}{9} * 0.0\right) = 0.251$ 2
- $I_{Gain}(temp > 29) = H(S) - \left(\frac{6}{9} H(3+, 3-) + \frac{3}{9} H(0+, 3-)\right) = 0.918 - \left(\frac{6}{9} * 1.0 + \frac{3}{9} * 0.0\right) = 0.251$ 3

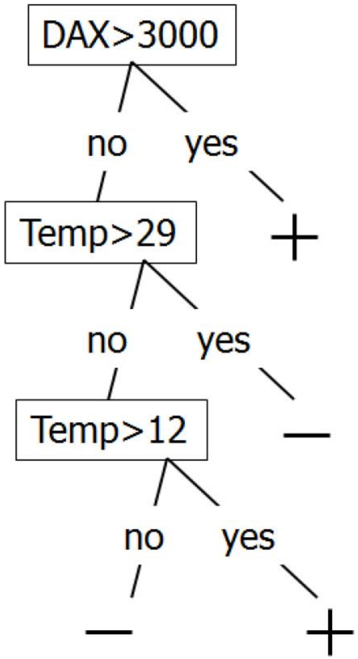
Example of Greedy Approach

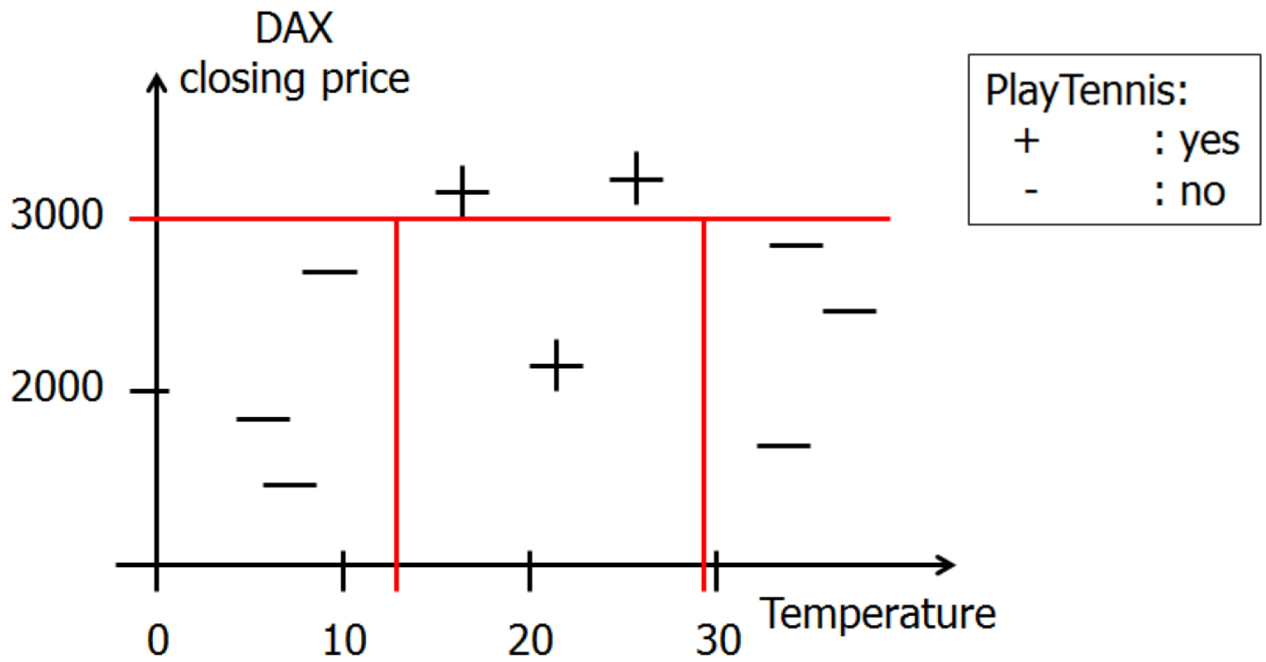
- First best split
- DAX > 3000



Example of Greedy Approach

- Second and third best split and final decision tree
- temp > 12, temp > 29





Decision Tree construction: Summary

- Greedly investigate all possible splits
- Weight possible splits by the evaluation measure and the respective size of the subsets
- Choose the split, resulting in the biggest reduction of error

Missing Values

- Dealing with missing values in decision trees is quite trivial

- **During training**
  - Estimate the impact of the missing value on the information gain measure
  - e.g. add a fraction of each class to each partition

- **During classification**
  - During the tree traversal, the node relies on a value missing for that record
  - Follow both branches
  - In the end, merge the results encountered in the two (or more) leaves

Fuzzy Decision Tree
can handle degrees of membership

Missing Value Strategies during Training

ID	...	Temp.	...	Play Tennis
ID1		Hot		No
ID2		Hot		No
ID3		Hot		Yes
ID4		Mild		Yes
ID5		???		Yes
ID6		Cool		No
ID7		Cool		Yes
ID8		???		No
ID9		Cool		Yes
ID10		Mild		Yes
ID11		Mild		Yes
ID12		Mild		Yes
ID13		Hot		Yes
ID14		Mild		Yes

Missing Value Strategies during Training

ID	...	Temp.	...	Play Tennis
ID1		Hot		No
ID2		Hot		No
ID3		Hot		Yes
ID4		Mild		Yes
ID5		**Mild**		Yes
ID6		Cool		No
ID7		Cool		Yes
ID8		**Mild**		No
ID9		Cool		Yes
ID10		Mild		Yes
ID11		Mild		Yes
ID12		Mild		Yes
ID13		Hot		Yes
ID14		Mild		Yes

Alternative 1
Replace with most frequent value

- 4* Hot
- **5* Mild**
- 3* Cool

Missing Value Strategies during Training

ID	...	Temp.	...	Play Tennis
ID1		Hot		No
ID2		Hot		No
ID3		Hot		Yes
ID4		Mild		Yes
ID5		**Mild**		Yes
ID6		Cool		No
ID7		Cool		Yes
ID8		**Hot**		No
ID9		Cool		Yes
ID10		Mild		Yes
ID11		Mild		Yes
ID12		Mild		Yes
ID13		Hot		Yes
ID14		Mild		Yes

Alternative 2
Replace with most frequent value of the same class

PlayTennis = Yes
- 2* Hot
- **4* Mild**
- 2* Cool

PlayTennis = No
- **2* Hot**
- 1* Mild
- 1* Cool

Missing Value Strategies during Training

ID	...	Temp.	...	Weight	Play Tennis
ID1		Hot		1.0	No
ID2		Hot		1.0	No
ID3		Hot		1.0	Yes
ID4		Mild		1.0	Yes
ID5a		**Hot**		**0.33**	Yes
ID5b		**Mild**		**0.42**	Yes
ID5c		**Cool**		**0.25**	Yes
ID6		Cool		1.0	No
ID7		Cool		1.0	Yes
ID8a		**Hot**		**0.33**	No
ID8b		**Mild**		**0.42**	No
ID8c		**Cool**		**0.25**	No
ID9		Cool		1.0	Yes
...		...		...	...

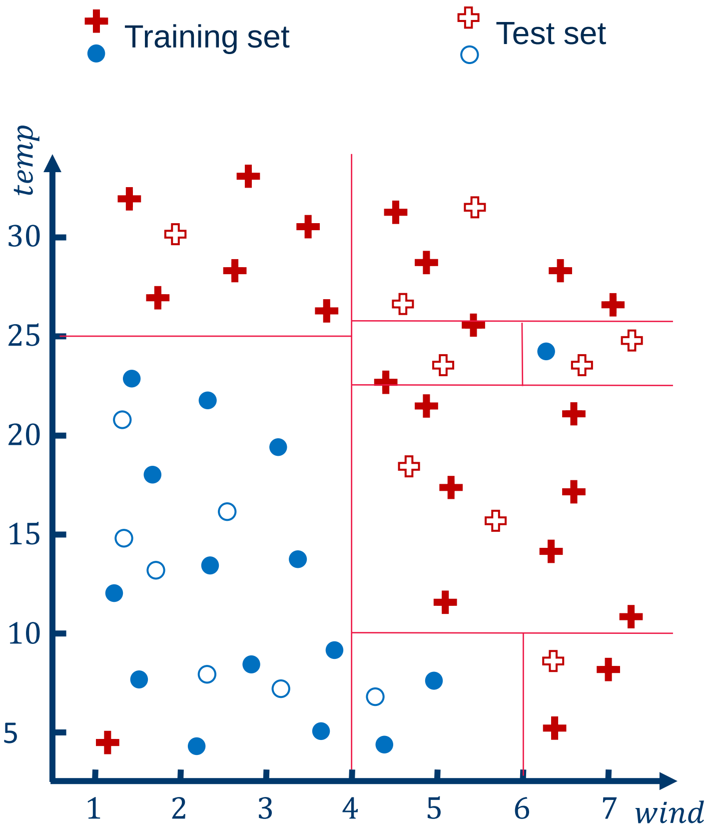
Fuzzy Decision Trees

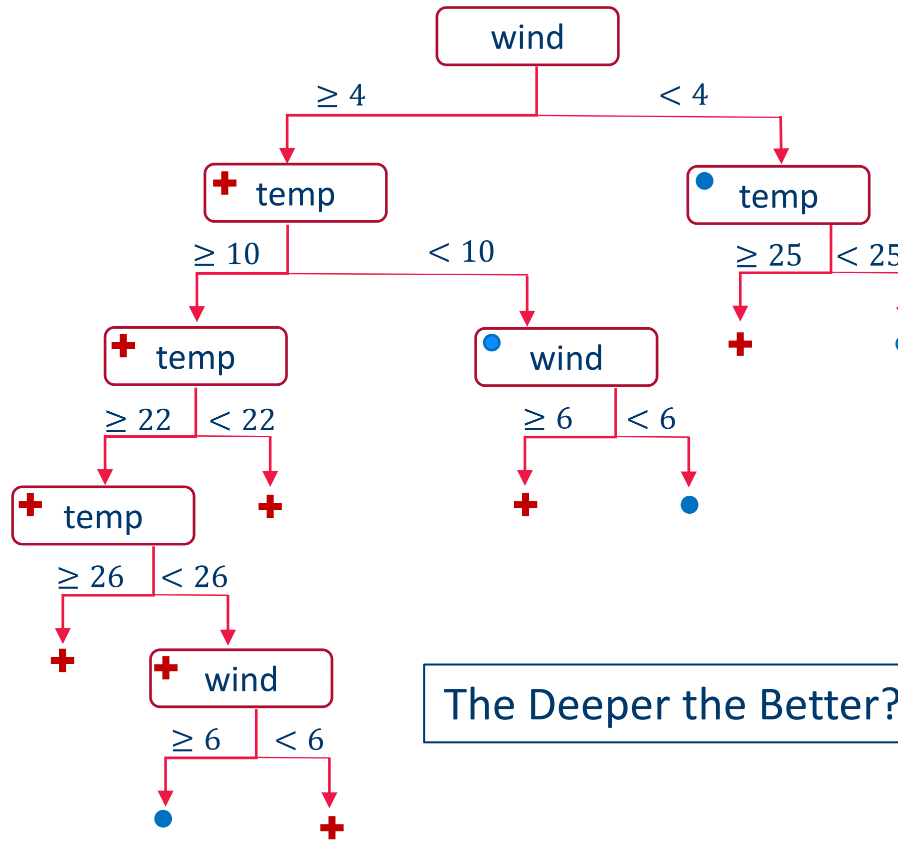
Alternative 3
Replace with all possible values and propagate with corresponding weights:

- 4* Hot -> w = 4/12 = 0.33
- 5* Mild -> w = 5/12 = 0.42
- 3* Cool -> w = 3/12 = 0.25

Decision Trees: Pruning

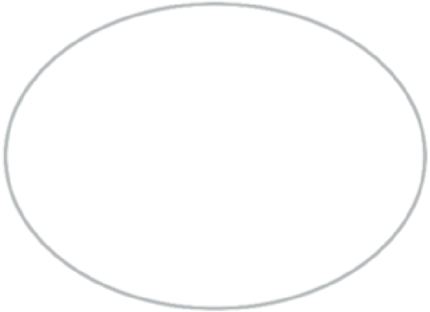
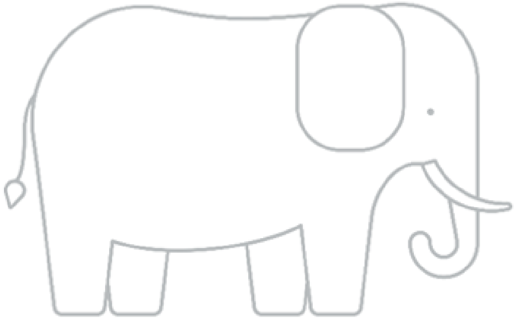
How deep should the tree be?





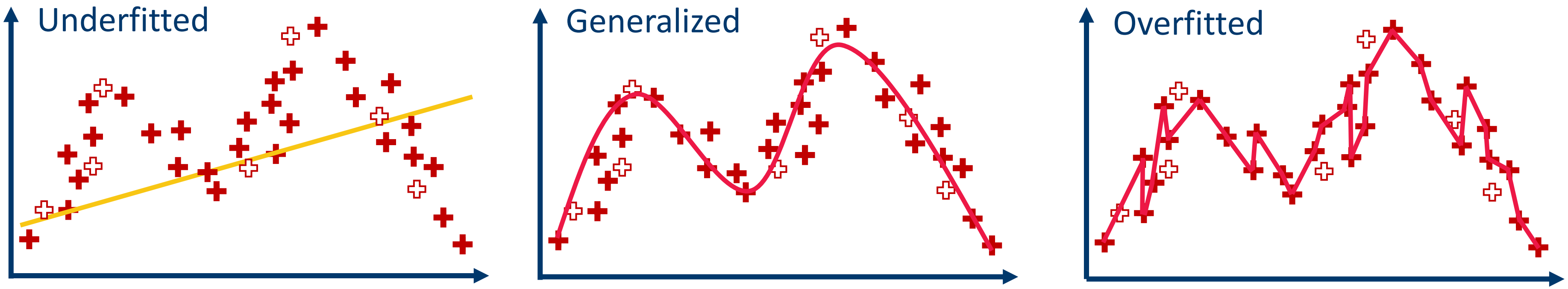
The Deeper the Better?!

Overfitting vs Underfitting

Underfitted	Generalized	Overfitted
		
Model overlooks underlying patterns in the training set	Model captures correlations in the training set	Model memorizes the training set rather then finding underlying patterns

Overfitting vs Underfitting

Overfitting	Underfitting
• Model that fits the training data too well, including details and noise • Negative impact on the model's ability to generalize	• A model that can neither model the training data nor generalize to new data



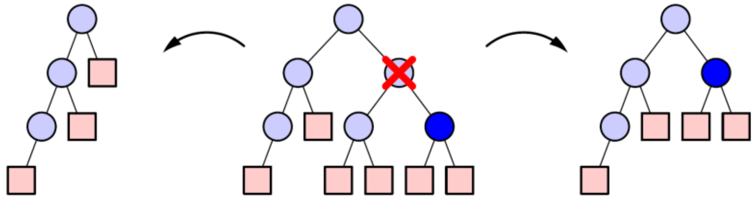
Pruning

- Why should we avoid overgrown trees?
  - Overfitting
  - Unnecessary complexity
  - Harder interpretation

The base algorithm would only stop if one node contains one pattern or there are no further attributes to be used for splits

- Two approaches:
  - **Pre-pruning**: stop the construction of the tree during training
  - **Post-pruning**: reduce the dimensions of an overgrown tree

- Basic ideas of post-pruning:
  - Replace “bad” branches (subtrees) by leaves
  - Replace a subtree by its largest branch if it is better



Controlling the Tree Depth

Goal: Tree that generalizes to new data and doesn't overfit

Early stopping

- Idea: Define a minimum size for the tree leaves

Pruning

- Idea: Cut branches that seem to overfit
- Techniques
  - Reduced Error Pruning
  - Pessimistic Pruning
  - Confidence Level Pruning
  - Minimum description length

Reduced Error Pruning

- Classify a set of **new example cases** with the decision tree. (These cases must not have been used for the learning!)
- Determine the **number of errors** for all leaves.
- The number of errors of a subtree is the sum of the errors of all of its leaves.
- Determine the number of errors for leaves that replace subtrees.
- If such a leaf leads to the same or fewer errors than the subtree, replace the subtree by the leaf.
- If a subtree has been replaced, recompute the number of errors of the subtrees it is part of.

- **Advantage:**
  Very good pruning, effective avoidance of overfitting.

- **Disadvantage:**
  Additional example cases needed. Number of cases in a leaf has no influence.

Pessimistic Pruning

- Classify a set of example cases with the decision tree. (These cases **may or may not** have been used for the learning.)
- Determine the number of errors for all leaves and **increase this number by a fixed, user-specified amount *r*.**
- The number of errors of a subtree is the sum of the errors of all of its leaves.
- Determine the number of errors for leaves that replace subtrees (also increased by *r*).
- If such a leaf leads to the same or fewer errors than the subtree, replace the subtree by the leaf and recompute subtree errors.

Pessimistic Pruning

- **Advantage:**
  No additional example cases needed.

- **Disadvantage:**
  Number of cases in a leaf has no influence.

Confidence Level Pruning

Like pessimistic pruning, but the number of errors is computed as follows:

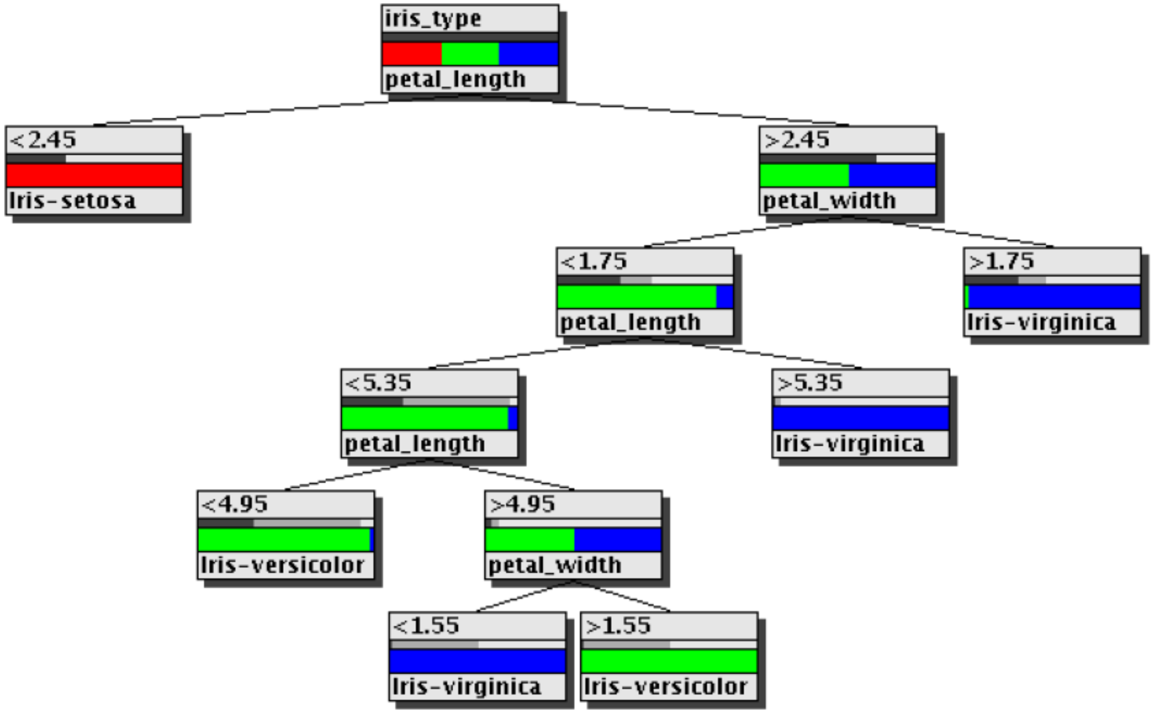
- See classification in a leaf as a Bernoulli experiment (error/no error): p, $p(1-p)$
  - Expected success rate: $f = \frac{no\ error}{error + no\ error}$
  - For a large enough number of classifications f follows a normal distribution
- Estimate an interval for the error probability $p(1-p)$ based on a user-specified confidence level α. (use approximation of the binomial distribution by a normal distribution)
- Increase error number to the upper level of the confidence interval times the number of cases assigned to the leaf.
- Formal problem: Classification is not a random experiment.

Confidence Level Pruning

- **Advantage:**
  No additional example cases needed. Good pruning.

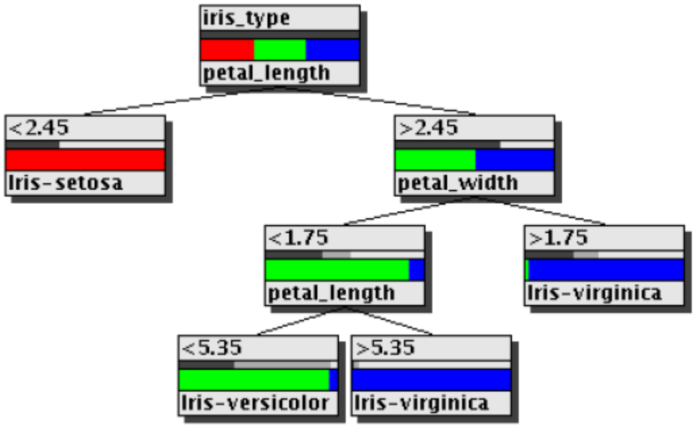
- **Disadvantage:**
  Statistically dubious foundation.

– A decision tree for iris data (induced with information gain ratio, unpruned)



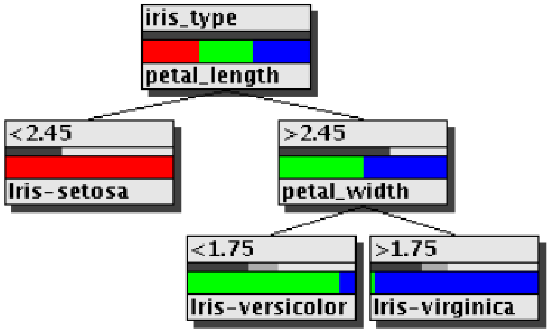
Decision tree pruning: An Example

Pruned with confidence level pruning α=0.8



7 instead of 11 nodes, 4 instead of 2 misclassifications.

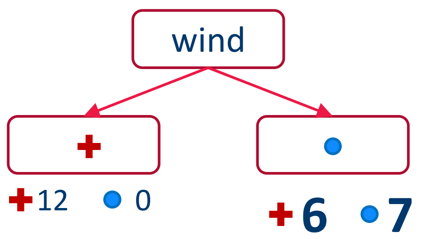
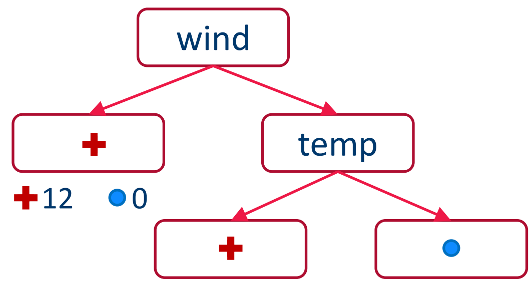
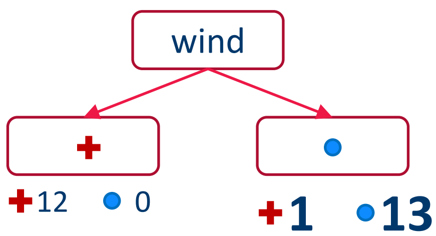
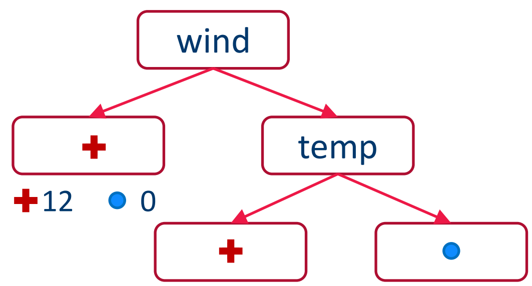
Pruned with pessimistic pruning r=2



5 instead of 11 nodes, 6 instead of 2 misclassifications.
This tree is “minimal” for the three classes.

Minimum Description Length Pruning (MDL)

Description length = #bits(tree) + #bits(misclassified samples)

	Tree 1	Tree 2	Note
Example 1			Many misclassified in samples in tree 1 => DL(Tree 1) > DL(Tree 2) => Select Tree 2
Example 2			Only 1 misclassified sample in tree 1 => DL(Tree 1) < DL(Tree 2) => Select Tree 1

- **Advantage:**

  No additional example cases needed. Good pruning.

- **Disadvantage:**

  Additional calculation needed.

- **Predictive tasks**:

  The decision tree (or more generally, the classifier) is constructed in order to apply it to new unclassified data.

- **Descriptive tasks:**

  The purpose of the tree construction is to understand, how classification has been carried out so far.

Regression Trees

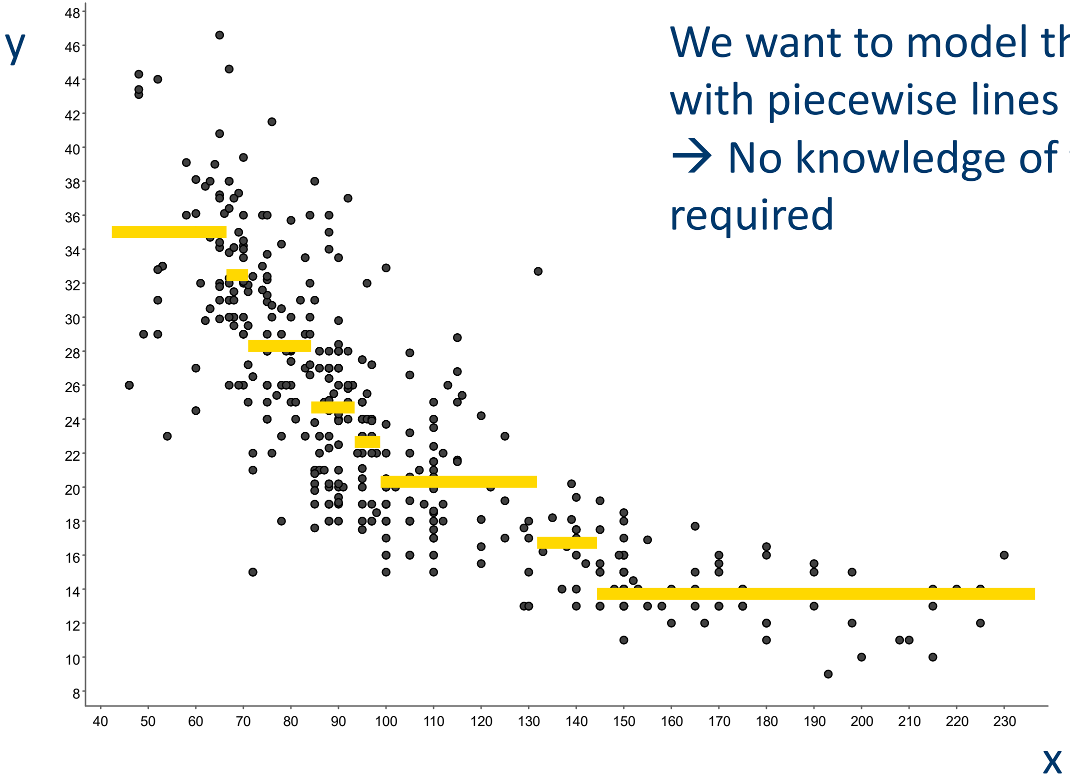
Numerical Target: Regression Trees

- Numerical Target = Predicting a **continuous** target value
- **CART** (Classification And Regression Trees) described by Breiman
- Leaves contain numerical values instead of class labels
- Entropy-based measurements no longer needed
- Fit measure of the tree is the sum of squared errors for each node n:

$$SME(\mathcal{D}_n) = \frac{1}{|\mathcal{D}_n|} \sum_{(\boldsymbol{X},Y)\in\mathcal{D}_n} (Y - c_n)^2$$

  - c_n is the constant value assigned to the node n,
  - $\mathcal{D}_n$ are the data points ending up at the node n,
  - $\boldsymbol{X}$ is the input feature vector
  - Y is the target value

Regression Tree: Goal



We want to model the target variable with piecewise lines

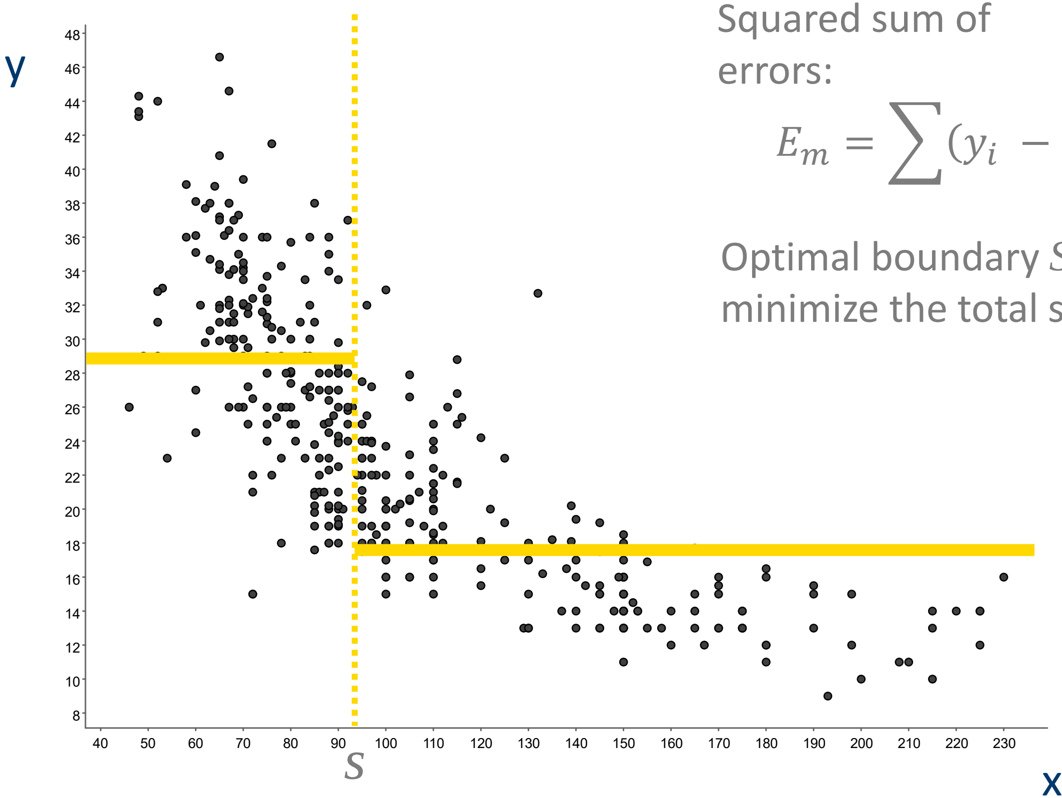
→ No knowledge of functional form required

Regression Tree: Initial Split

Local mean:

$$c_m = \frac{1}{n}\sum y_i$$

For observations in segment m



Squared sum of errors:

$$E_m = \sum (y_i - c_m)^2$$

Optimal boundary S should minimize the total squared sum:

$$\sum E_m$$

For all segments m

Regression Tree: the algorithm

- Split a feature x_j at threshold s:

$$R_1(j,s) = \{\boldsymbol{x} \mid x_j \leq s\} \qquad \text{and} \qquad R_2(j,s) = \{\boldsymbol{x} \mid x_j > s\}$$

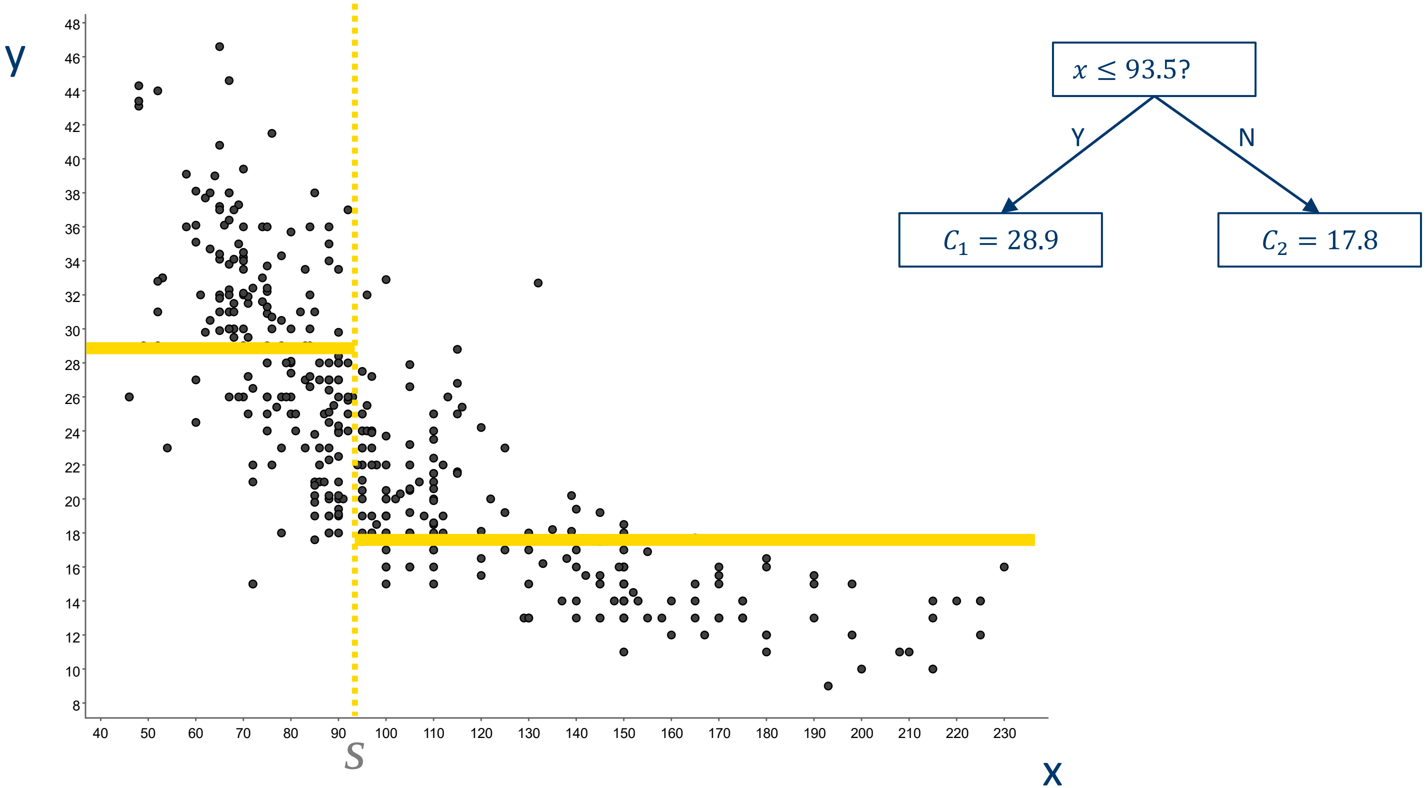
- Search for s that minimizes error of:

$$E_{j,s} = \sum_{\boldsymbol{x} \in R_1(j,s)} (y_i - c_1)^2 + \sum_{\boldsymbol{x} \in R_2(j,s)} (y_i - c_2)^2$$

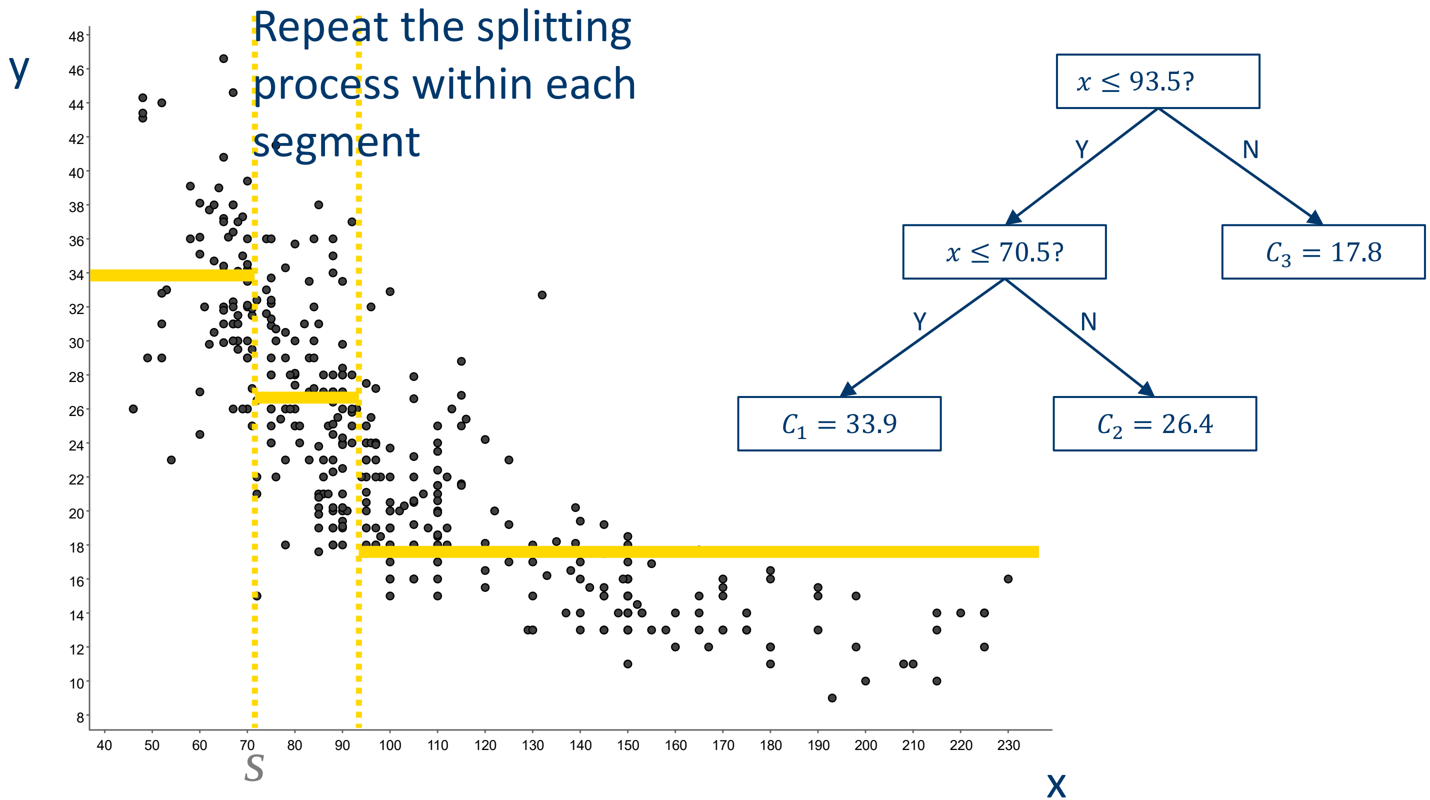
- where

$$c_k = \frac{1}{n} \sum_{\boldsymbol{x} \in R_k(j,s)} y_i \qquad \text{Where } (y_i \mid \boldsymbol{x} \in R_k, k = 1, 2)$$

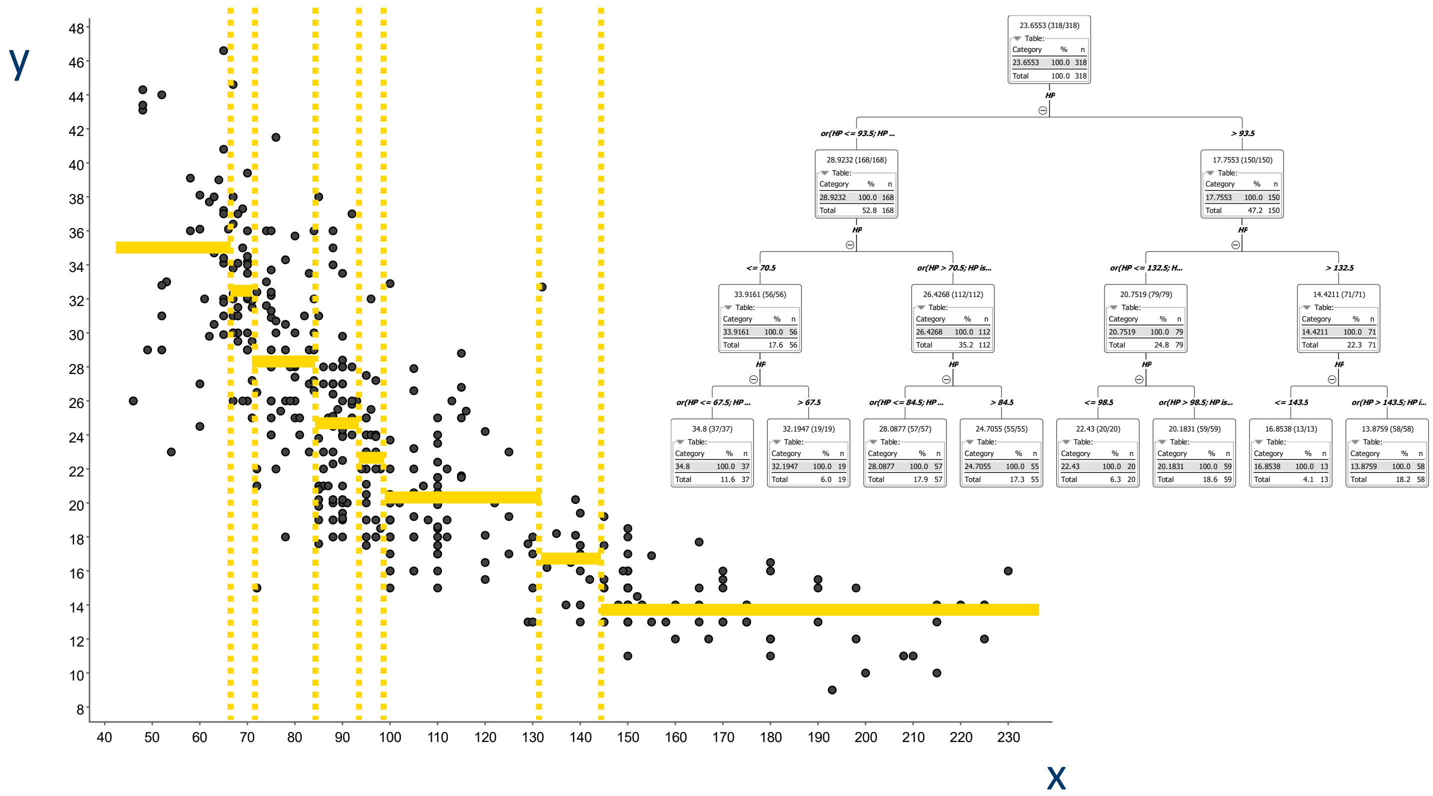
Regression Tree: Initial Split



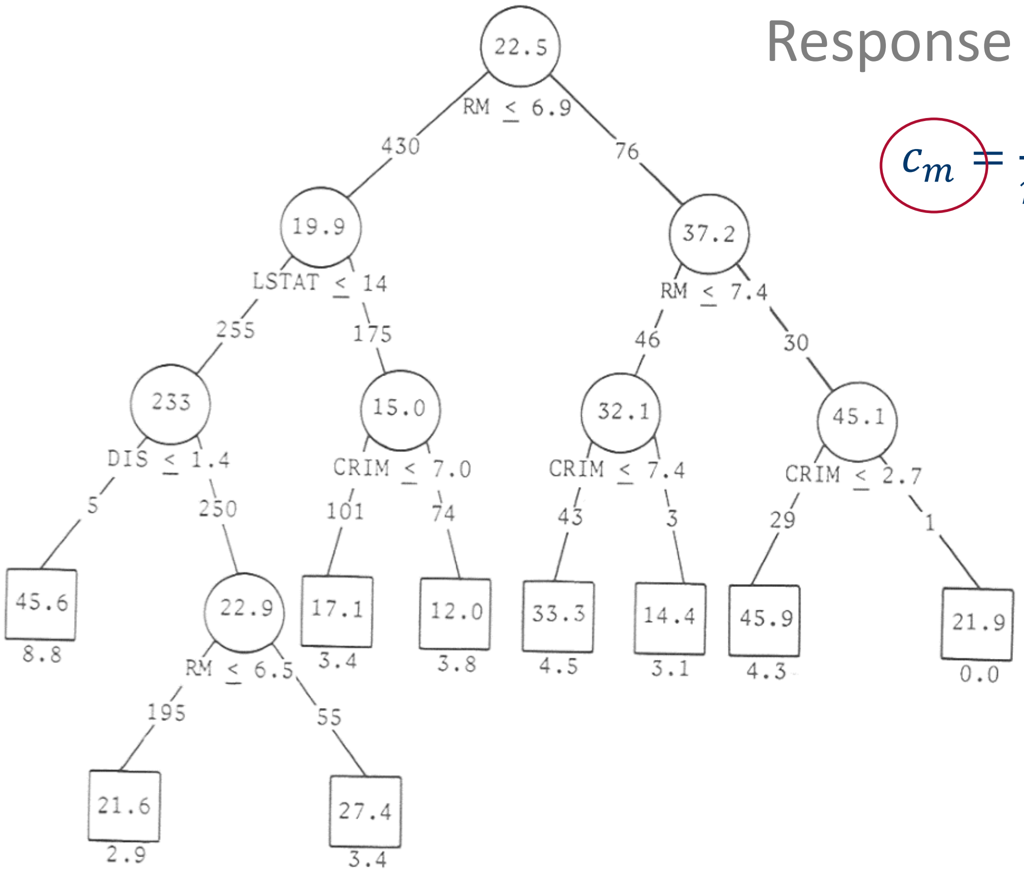
Regression Tree: Growing the Tree



Regression Tree: Final Model



Regression Tree: The Tree



Response in each leaf is a constant:

$$c_m = \frac{1}{n}\sum y_i \qquad \text{Where } (y_i \mid \boldsymbol{x} \in Leaf\ m)$$

$$E_m = \sum (y_i - c_m)^2$$

More complex regression trees have functions in the leaves rather than constant values. Rarely used due to harder interpretation

Start with a single node containing all points.

1. Calculate c_i and E_i.
2. If all points have the same value for feature x_j, stop.
3. Otherwise, find the best binary splits that reduces $E_{j,s}$ as much as possible.
   - $E_{j,s}$ doesn't reduce as much → stop
   - A node contains less than the minimum node size → stop
   - Otherwise, take that split, creating two new nodes.
   - In each new node, go back to step 1.

- Differences to decision trees:
  - Splitting criterion: minimizing intra-subset variation (error)
  - Pruning criterion: based on numeric error measure
  - Leaf node predicts average target values of training instances reaching that node

- Can approximate piecewise constant functions
- Easy to interpret

Regression Trees: Pros & Cons

- Finding of (local) regression values (average)
- Problems:
  - No interpolation across borders
  - Heuristic algorithm: unstable and not optimal.

- Extensions:
  - Fuzzy trees (better interpolation)
  - Local models for each leaf (linear, quadratic)

Additional Notes

- Despite their wide application, Decision Trees are notoriously unstable
- Small changes in the training data can heavily change the resulting decision tree

This is due to the underlying greedy algorithm

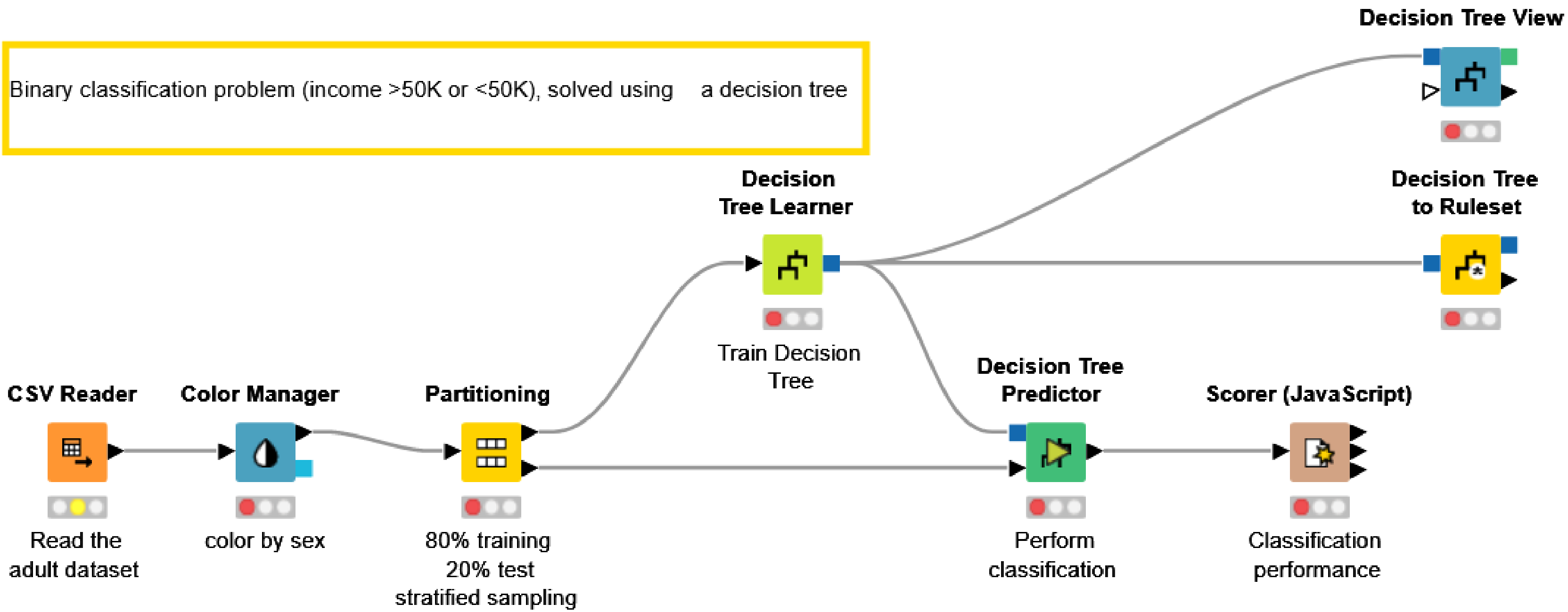
- **Forests of decision trees**
- Build a number of smaller, differently initialized decision trees
- Compute the classification by committee voting
- Can solve the stability problem
- Lead to less interpretability

What you should remember from this lesson

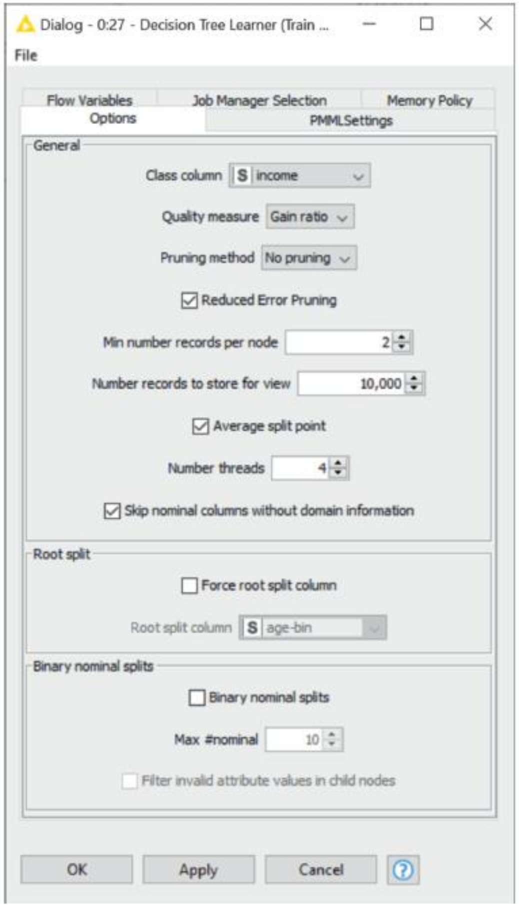
- What is a decision tree and what is it used for?
- How is a decision tree built from data?
- Which kind of splitting measurements exist?
- How can we handle missing values in the data?
- Why are decision trees pruned and how?
- What is a regression tree?

Practical Example

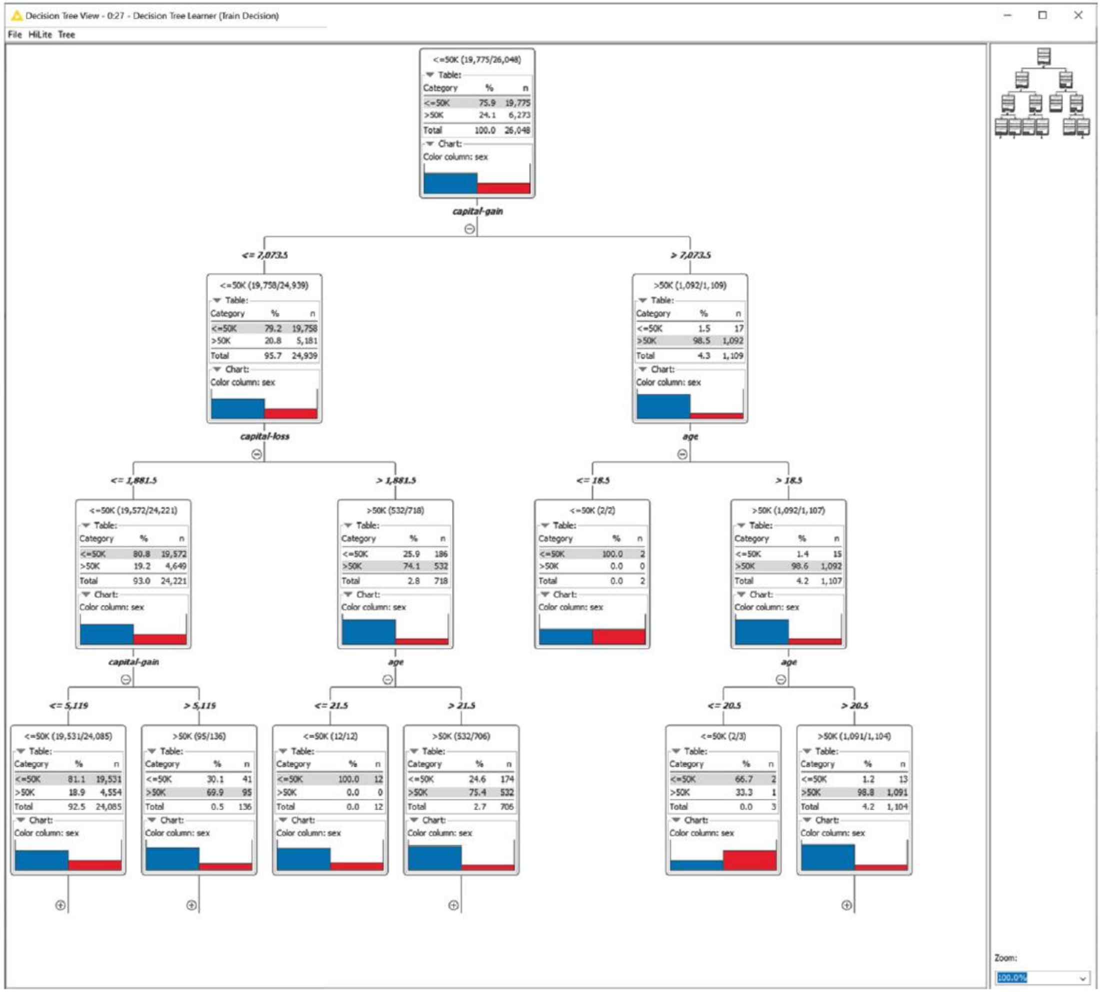
– Decision Tree



– Decision tree learner configuration window



– Trained tree with the tree view



Thank you

For any questions please contact: education@knime.com