Knowledge-Based Systems (KBS)

Machine Learning and Knowledge Engineering

Winter semester 2021/2022

Prof. Dr. Martin Atzmüller

Osnabrück University & DFKI
Semantic Information Systems Group
https://sis.cs.uos.de/

4 – Interpretable Machine Learning

1. Overview
2. Decision Trees
3. Decision Rule Models
4. Linear & Logistic Regression
5. Distance-Based Modeling & KNN

Notes: Initial presentation/intro roughly follows [Christoph Molnar (2021) Interpretable Machine Learning – A Guide for Making Black Box Models Explainable. `https://christophm.github.io/interpretable-ml-book/`].

Lecture slides also partially based on materials courtesy of:

- Prof. Dr. Dietmar Seipel, University of Würzburg, Germany
- Prof. Dr. Peter A. Flach, University of Bristol, UK

Knowledge-Based Systems – Overview/Contents

1. Introduction & Overview
2. Knowledge Engineering & Representation Formalisms
3. Knowledge Acquisition & Machine Learning
4. **Interpretable Learning**
5. Knowledge Discovery & Data Mining
6. Graph & Link Mining
7. Knowledge Graphs
8. Rule-Based Approaches and Case-Based Reasoning
9. Uncertainty Modeling & Answer-Set Programming
10. (Deep) Neural Networks
11. Interpretability, Explanation & Explainability
12. Informed Machine Learning

Motivation

Why do we not just trust a (well-learned) model when it makes a prediction, so just ignore why it made a certain decision?

- ... Human curiosity and learning
- ... Sensemaking: Finding meaning in the world
  $\rightsquigarrow$ Computational Sensemaking (via Machine Learning, Knowledge Discovery/Data Mining)
- ... Detecting bias in machine learning models
- ... Debugging and auditing
- ... By explaining decisions, you can also potentially check:
  - Fairness
  - Privacy
  - Reliability/Robustness
  - Causality
  - Trust

Overview

- Interpretable Machine Learing:
  - In general: Methods & models ...
  - ... that make the behavior and predictions of machine learning systems *understandable* to humans
- Machine learning task:
  - Combination of dataset with feature and target
  - E. g., classification, regression, clustering, outlier prediction ...
- Prediction:
  - Intuitively: What a machine learning model "guesses" what the target value should be ...
  - ... based on the given set of features

Taxonomy

- Intrinsic or post-hoc
  - intrinsically interpretable models, vs.
  - model-agnostic interpretation methods
- Model-specific or model-agnostic
- Local or global

Algorithm	Linear	Monotone	Interaction	Task
Decision Trees	No	Some	Yes	Classification, Regression
Decision Rules	Some	Some	Some	Classification, Regression
Linear regression	Yes	Yes	No	Regression
Logistic regression	Yes	Yes	No	Classification
Naive Bayes	No	Yes	No	Classification
K-Nearest Neighbor	No	No	No	Classification, Regression

Model Properties

- Linear: Association between features and target – linearly
- Monotonicity: Relationship between feature and target always in same direction (increase/decrease)
- Interaction:
  - Model able to capture feature interactions towards target
  - Alternative: Construct "interaction features"
- Task: Classification/regression, or both

Brief Overview – Interpretable Models

First: Short overview – main features

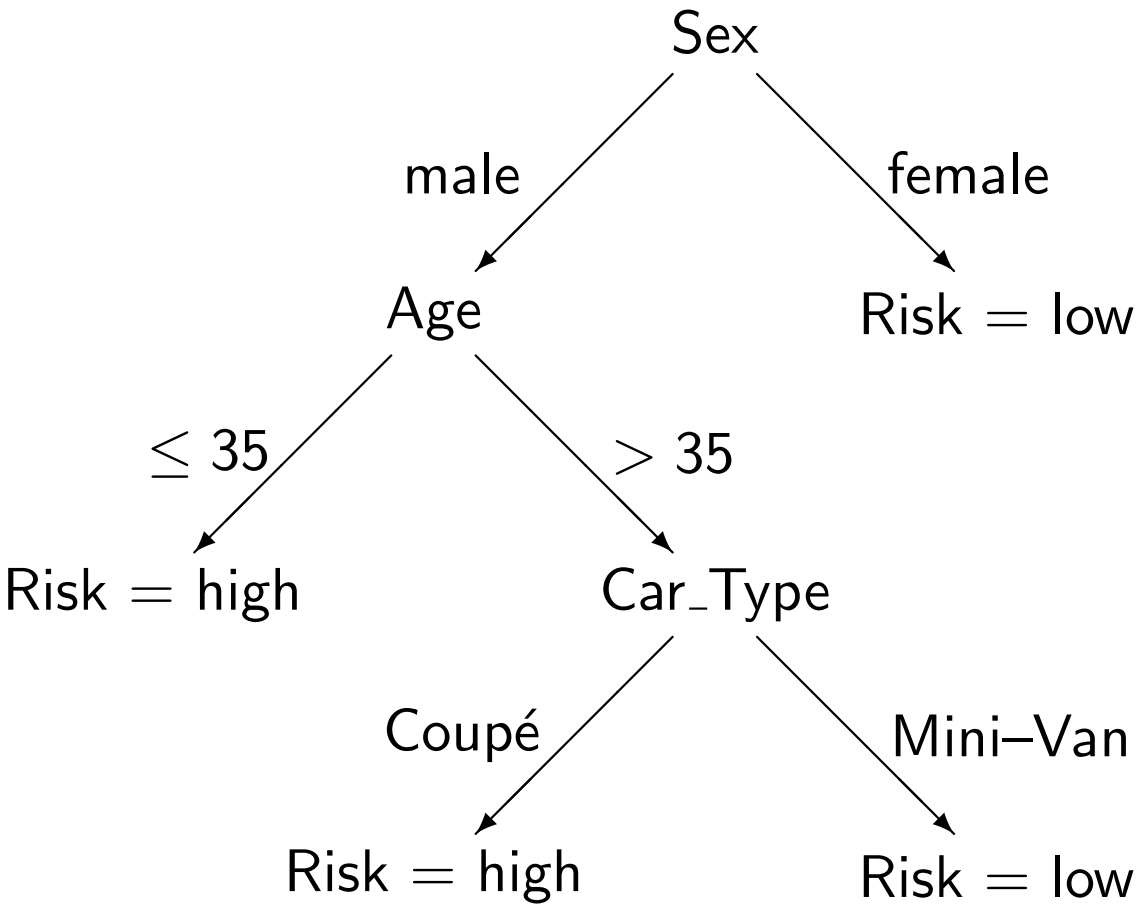
- Decision Trees: Splitting data in tree structure (cut-offs/decisions) until leaf node reached
- Linear Regression: Linear relationship modeling – Predict target as weighted sum of feature inputs
- Logistic Regression: Extension of linear regression, model probabilities for classification problems
- Naive Bayes: Based on Bayes' theorem, models feature contributions towards target
- K-Nearest-Neighbor: Nearest-neighbors used for prediction

After that: How to construct/learn these models.

Decision Tree (DT)

- DT splits data in tree structure (cut-offs/decisions) until leaf node reached
  $\rightsquigarrow$ different subsets are created, each instance belonging to exactly one
- Prediction: Outcome of each leaf node – majority/average in the node is used
- Can be used for classification and regression
- Interpretation: Feature importance, tree structure, visualization

DT – Example: Car Insurance



DT - Advantages/Disadvantages

- Advantages:
  - Capture interactions
  - Natural visualizations
  - Good explanation – tree structure
- Disadvantages:
  - Inefficient for linear relationships due to splits
  - Unstable, e. g., small variations in the training set
  - Interpretable specifically for small(er) trees

Decision Rules / Rule Models

- In general: Very versatile, many formalisms
- Already seen: Diagnostic scoring rules – different levels of complexity
- Basically: Some form of "IF-THEN" rules, e. g.,:

  IF $A = a_1$ AND $B > 3.5$ AND ... AND $Z = z_3$ THEN $class = true$
- Rule lists vs. rule sets
- Different approaches for learning, e. g., based on (greedy) heuristics, frequent patterns, class association rules, subgroup discovery
- One problem: Rules may overlap
- How to combine predictions of single rules?

Advantages/Disadvantages of Rules

- Advantages
  - Easy to interpret
  - As expressive as decision trees, but can be more compact
  - Prediction (process) is fast
  - Robustness (transformations of features, outliers)
  - Only relevant features are included in the model
- Disadvantages
  - Regression mostly via discretization of the target (intervals)
  - Same goes often for the features (discretization required)
  - Safeguards for overfitting required (!)
  - Linear relationships not well captured
    - No "smooth curves" for prediction
    - Instead: Step functions for predictions (like DTs)

Linear & Logistic Regression

- Linear regression: Capture linear relationship:
  $y = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$
  - Predicted outcome – weighted sum of its features
  - β_0: Intercept
  - Linearity makes interpretation easy
- Numerical vs. binary vs. categorical features: transformations – one-hot-encoding, normalization
- Advantages
  - Weighted sum – transparent prediction
  - Efficient estimation
  - Statistically well founded
- Disadvantages
  - Only linear relationships can be represented
  - Interactions need to be hand-crafted
  - Interpretation of weights only in relation to other weights
- Logistic regression: Extension towards (probabilistic) classification

Naive Bayes

- Applies Bayes' theorem – conditional probability model: $P(C_k \mid x) = \frac{1}{Z} P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)$
  - Z: scaling parameter, ensures that sum of probabilities (for all classes) is 1
  - Otherwise: individual conditional probabilities, class probability
- Interpretability due to *strong* independence assumption between features
- Modular: Contribution of each feature clear, due to conditional probability

K-Nearest Neighbor (KNN)

- KNN: Classification + regression
- Uses the (set of) nearest neighbors of a data point (instance) for prediction
- Classification: Most common class
- Regression: Average of the k-nearest neighbors
- Important difference to other approaches: Instance-based method (!)
- Local interpretability – comes down to interpretation of a single instance

Interpretable Machine Learning modeling ... in more detail

Decision Trees

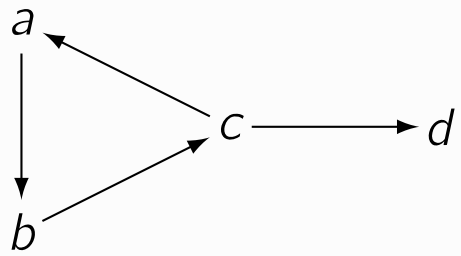
Decision Trees A decision tree (or classification tree) groups many classification rules to a classification schema.

Every branch represents a classification rule $r = L \Rightarrow R$, where $L = L_1 \wedge \ldots \wedge L_n$:

1. A leaf is marked by the conclusion R of r together with the pair (f, c) consisting of the frequency and the confidence of r.
2. The inner nodes and the edges of the branch are marked based on the conjuncts L_i.

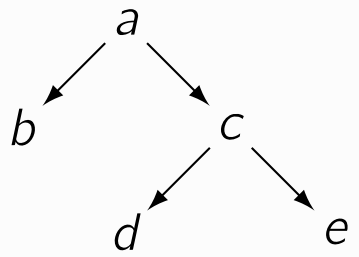
Notions from Graph Theory

$\mathcal{G}$:

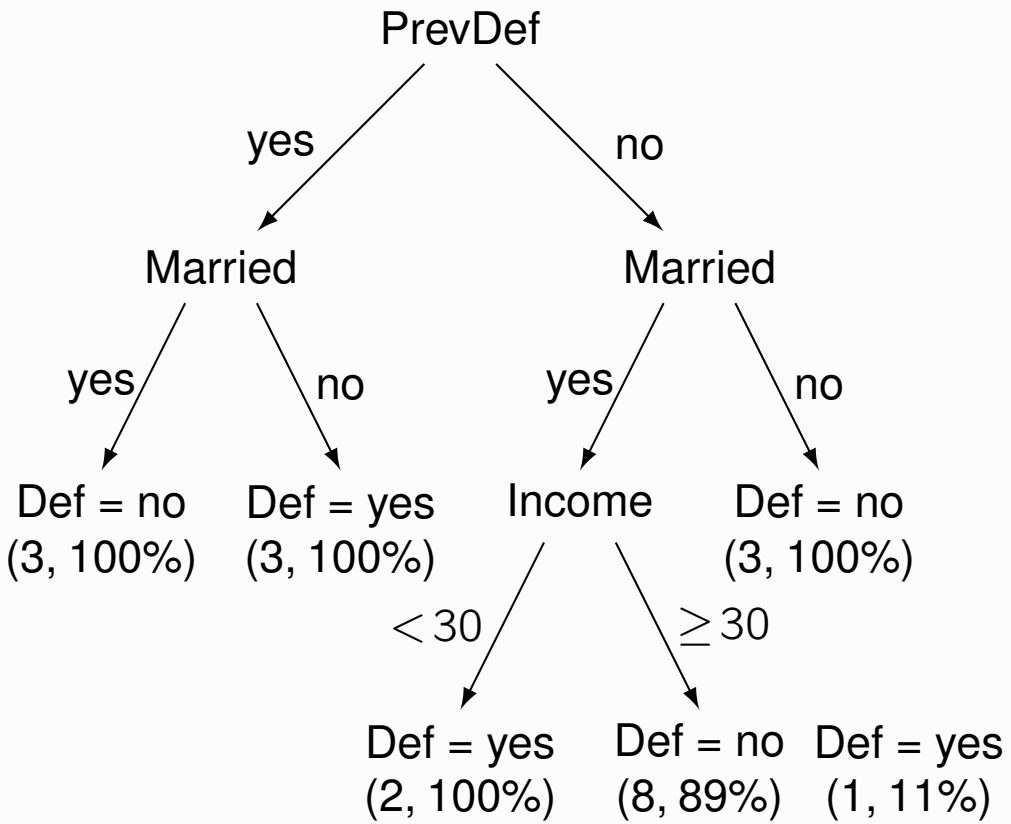


1. *Graph* $\mathcal{G} = (N, E)$:
   - nodes $N = \{ a, \ldots, d \}$,
   - edges $E = \{ (a, b), (b, c), (c, a), (c, d) \}$.
2. A *tree* $\mathcal{T} = (N, E)$ is a special graph,
   where all nodes except the root have one incoming edge;
   the root has no incoming edges:
   - nodes $N = \{ a, \ldots, e \}$,
   - edges $E = \{ (a, b), (a, c), (c, d), (c, e) \}$,
   - root a, leaves b, d, e,
   - branch $a \to c \to d$.

$\mathcal{T}$:

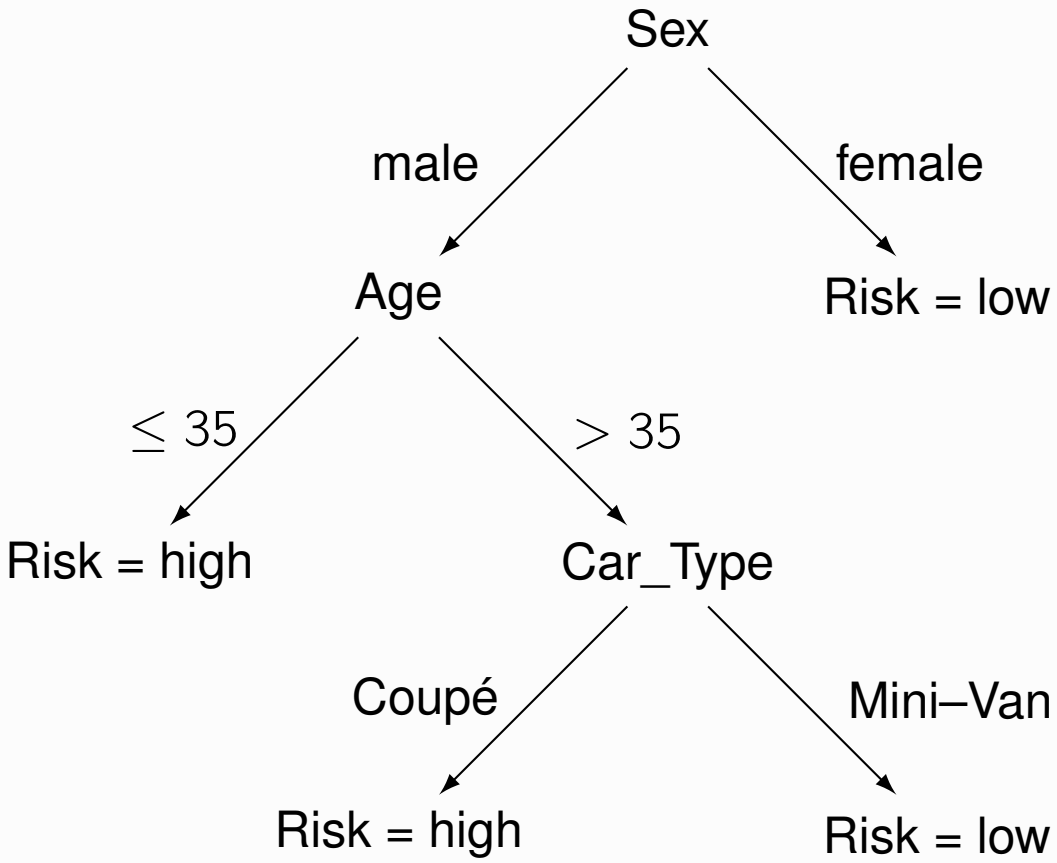


Example (Creditworthiness)

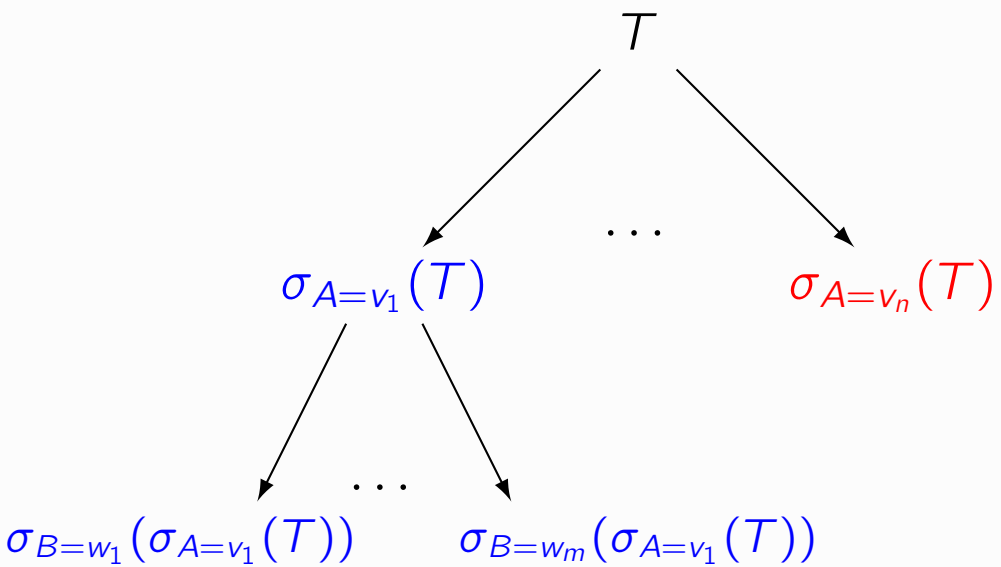


The quality of a decision tree and its corresponding classification rules is measured by the weighted average homogeneity of the leaf tables.

Example (Car Insurance)

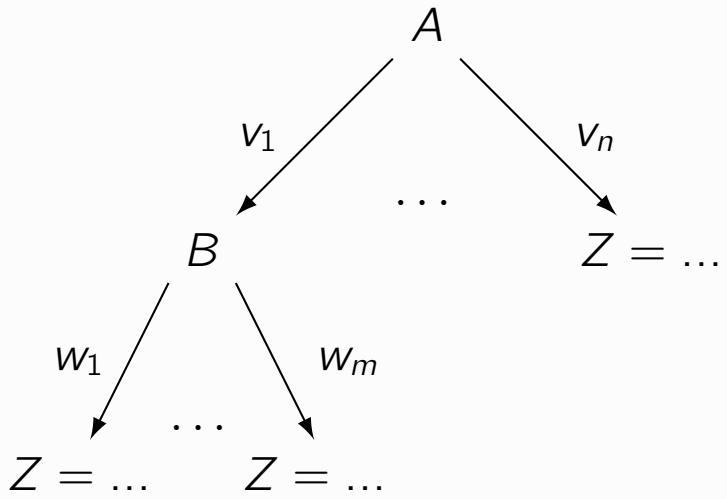


Computation of Decision Trees We recursively split a given table T w.r.t. an attribute A into sub–tables $\sigma_{A=v}(T)$ for all the possible values v of the attribute A in T. $\rightsquigarrow$ Recursive Split



In each step, we choose an attribute that maximizes the quality of the resulting classification rules w.r.t. their confidence and support.

In the corresponding decision tree, the nodes are labelled by the attributes used for the split, and the outgoing edges are labelled by their possible values:



Here, the classification rules for the leaves of the decision tree are:

$(A = v_1) \wedge (B = w_1) \Rightarrow (Z = ...)$ ...

$(A = v_1) \wedge (B = w_m) \Rightarrow (Z = ...)$

...

$(A = v_n) \Rightarrow (Z = ...)$

Example (Computation of Decision) – Dataset (Creditworthiness)

Id	Married	PrevDef	Income	Def
C1	yes	no	50	no
C2	yes	no	100	no
C3	no	yes	135	yes
C4	yes	no	125	no
C5	yes	no	50	no
C6	no	no	30	no
C7	yes	yes	10	no
C8	yes	no	10	yes
C9	yes	no	75	no
C10	yes	yes	45	no

Id	Married	PrevDef	Income	Def
C11	yes	no	60	yes
C12	no	yes	125	yes
C13	yes	yes	20	no
C14	no	no	15	no
C15	no	no	60	no
C16	yes	no	15	yes
C17	yes	no	35	no
C18	no	yes	160	yes
C19	yes	no	40	no
C20	yes	no	30	no

Recap: Confidence, Support, and Frequency

Given a classification rule $r = L \Rightarrow R$ in a table T.

- $$confidence(r) = \frac{n_{L \wedge R}(T)}{n_L(T)},$$

  where $n_\ominus(T)$ is the number of tuples satisfying the condition $\ominus$.

- $$support(r) = \frac{n_{L \wedge R}(T)}{n(T)},$$
  $$freq(r) = n_{L \wedge R}(T),$$

  where $n(T) = |T|$ is the number of tuples in T.

We denote a classification rule with support s and confidence c as:

$$r = L \stackrel{s,c}{\Longrightarrow} R.$$

Example (Computation of Decision Trees)

Split of the table T in two sub–tables $\sigma_{PrevDef=no}(T)$ and $\sigma_{PrevDef=yes}(T)$

Id	Married	PrevDef	Income	Def
C1	yes	no	50	no
C2	yes	no	100	no
C4	yes	no	125	no
C5	yes	no	50	no
C6	no	no	30	no
C8	yes	no	10	yes
C9	yes	no	75	no
C11	yes	no	60	yes
C14	no	no	15	no
C15	no	no	60	no
C16	yes	no	15	yes
C17	yes	no	35	no
C19	yes	no	40	no
C20	yes	no	30	no

Id	Married	PrevDef	Income	Def
C3	no	yes	135	yes
C7	yes	yes	10	no
C10	yes	yes	45	no
C12	no	yes	125	yes
C13	yes	yes	20	no
C18	no	yes	160	yes

The left (red) sub–table yields two classification rules:

$$r_1 = (PrevDef = no) \overset{11/20,\,11/14}{\Longrightarrow} (Def = no)$$
$$r_2 = (PrevDef = no) \overset{3/20,\,3/14}{\Longrightarrow} (Def = yes)$$

The right (blue) sub–table yields two classification rules:

$$r_3 = (PrevDef = yes) \overset{3/20,\,50\%}{\Longrightarrow} (Def = no)$$
$$r_4 = (PrevDef = yes) \overset{3/20,\,50\%}{\Longrightarrow} (Def = yes)$$

The two classification rules for the same sub–table – i.e., r_1, r_2 and r_3, r_4, respectively – are equivalent, since there are exactly two possible values for the target attribute Def.

Thus, we would use the rule r_1 with the higher confidence. The rules r_3 and r_4 are not useful for predicting the value of Def, since both of their confidences are 50%.

Double split w.r.t. *Married* and *PrevDef* leads to more homogeneous tables for predicting $Z = Def$:

$T_{yes,no}$

Id	Married	PrevDef	Income	Def
C1	yes	no	50	no
C2	yes	no	100	no
C4	yes	no	125	no
C5	yes	no	50	no
C8	yes	no	10	yes
C9	yes	no	75	no
C11	yes	no	60	yes
C16	yes	no	15	yes
C17	yes	no	35	no
C19	yes	no	40	no
C20	yes	no	30	no

$T_{yes,yes}$

Id	Married	PrevDef	Income	Def
C7	yes	yes	10	no
C10	yes	yes	45	no
C13	yes	yes	20	no

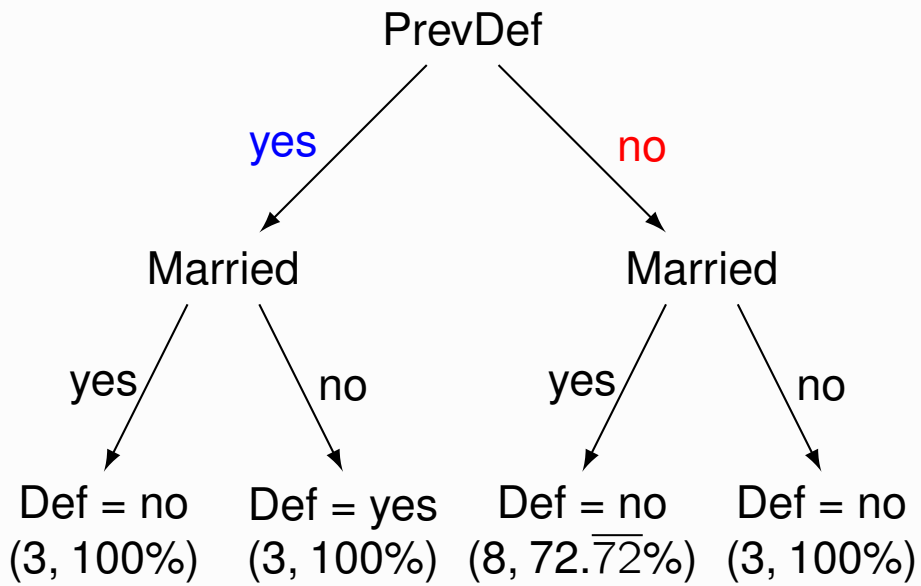
$T_{no,no}$

Id	Married	PrevDef	Income	Def
C6	no	no	30	no
C14	no	no	15	no
C15	no	no	60	no

$T_{no,yes}$

Id	Married	PrevDef	Income	Def
C3	no	yes	135	yes
C12	no	yes	125	yes
C18	no	yes	160	yes

We obtain the following decision tree:



Further splitting the third leaf w.r.t. the attribute Income could improve the quality of the tree and its corresponding classification rules.

The other leaves already yield predictions with a confidence of 100%.

The third branch yields two classification rules, since there are two possible values *yes* and *no* for the target attribute $Z = Def$, whereas the other branches yield just one classification rule:

1. $(PrevDef = yes) \wedge (Married = yes) \overset{3/20,\,1}{\Longrightarrow} (Def = no)$.
2. $(PrevDef = yes) \wedge (Married = no) \overset{3/20,\,1}{\Longrightarrow} (Def = yes)$.
3. $(PrevDef = no) \wedge (Married = yes) \overset{8/20,\,72.\overline{72}\%}{\Longrightarrow} (Def = no)$,
   $(PrevDef = no) \wedge (Married = yes) \overset{3/20,\,27.\overline{27}\%}{\Longrightarrow} (Def = yes)$.
4. $(PrevDef = no) \wedge (Married = no) \overset{3/20,\,1}{\Longrightarrow} (Def = no)$.

For the third branch, the classical algorithm ID3 only produces the classification rule with the higher confidence $72.\overline{72}\%$.

For the target attribute $Z = Income$ (instead of $Z = Def$), we would have 13 possible values $v_1, \ldots, v_{13}$, namely $10, 15, 30, 35, 40, 45, 50, 60, 75, 100, 125, 135, 160$

Quality of Classification Rules and Decision Trees For measuring the quality of the classification rules derived from a single table T, we apply a so–called entropy function ε. It combines the relative frequencies p_i for Z:

$$entropy_Z(T) = \varepsilon(p_1, \ldots, p_n).$$

For measuring the quality of the classification rules derived from a collection $T_1, \ldots, T_n$ of sub–tables of T, we will use a weighted averaging of entropy, where every table contributes according to its number of tuples:

$\rightsquigarrow$ Weighted average for a collection $T_1, \ldots, T_n$ of sub–tables of T:

$$entropy_Z(T_1, \ldots, T_n) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \cdot entropy_Z(T_i).$$

In the previous example, the two classification rules for the third table – with the relative frequencies $p_1 = 72.\overline{72}\%$ and $p_2 = 27.\overline{27}\%$ would contribute most, namely with the weight $\frac{|T_3|}{|T|} = 9/20$.

Entropy, Homogeneity of a Table

Given a table T of size $|T| = N$ with n different values $v_1, \ldots, v_n$ for Z.

Let N_i be the number of occurrences of the value v_i for Z in T.

- N_i: number of ocurrences of v_i
- $p_i = N_i/N$: relative frequency.

We aim to investigate measures $entropy_Z(T)$ for the homogeneity of the values for Z in T depending on the relative frequencies $p_i = N_i/N$:

$$entropy_Z(T) = \varepsilon(p_1, \ldots, p_n).$$

Properties of the Entropy

1. The function ε should be *continuous* on the domain of all tuples $(p_1, \ldots, p_n) \in [0,1]^n$, satisfying $\sum_{i=1}^{n} p_i = 1$.
2. Lower values should be assigned to more homogeneous tables.
3. ε should be independent of the order of its arguments:
   i.e., for every *permutation* $\pi : \langle 1, n \rangle \longrightarrow \langle 1, n \rangle$ it should hold
   $$\varepsilon(p_1, \ldots, p_n) = \varepsilon(p_{\pi(1)}, \ldots, p_{\pi(n)}).$$

Moreover, the following should hold:

$$\begin{aligned} \varepsilon(p_i, p_j, \dots) &\geq \varepsilon(p_i + p_j, \dots), \\ \varepsilon(p_i, p_j, \dots) &\leq \varepsilon((p_i + p_j)/2, (p_i + p_j)/2, \dots), \end{aligned}$$

since

4. *merging* increases the homogeneity, and
5. *averaging* decreases the homogeneity.

Observe, that $\varepsilon(p_i + p_j, \dots)$ has one argument less than $\varepsilon(p_i, p_j, \dots)$, since "..." stands for the same arguments.

From these conditions, it follows that ε becomes

4. minimal, if all p_i are 0, except one, which is 1, and
5. maximal, if all p_i are identical, i.e. $p_i = 1/n$.

Candidates for Entropy Functions We can think of the following candidates for an entropy function:

$$\begin{aligned}
\varepsilon(p_1, \ldots, p_n) &= \sum_{i=1}^{n} -p_i \cdot \log_2 p_i, \\
\varepsilon'(p_1, \ldots, p_n) &= \prod_{i=1}^{n} p_i = p_1 \cdot \ldots \cdot p_n, \\
\varepsilon''(p_1, \ldots, p_n) &= 1 - \sum_{i=1}^{n} p_i^2 = 1 - (p_1^2 + \ldots + p_n^2).
\end{aligned}$$

More precisely, for ε, the relative frequencies $p_i = 0$ have to be omitted, since $\log_2 p_i$ is not defined; this will be taken care of later.

We will analyze these three functions later and show that

- ε and ε'' fullfil all listed conditions, whereas
- ε' violates the merging condition.

According to literature, the functions ε and ε'' are used frequently.

Example (Permutation, Merging, and Averaging)

For $n = 2$ we get

$$\varepsilon(p_1, p_2) \;=\; \varepsilon(p_2, p_1),$$

and for $n = 3$ we get

$$\varepsilon(p_1, p_2, p_3) \;\geq\; \varepsilon(p_1 + p_2, 0, p_3) \;\geq$$
$$\varepsilon(\underbrace{p_1 + p_2 + p_3}_{=1}, 0, 0) \;=\; \varepsilon(1, 0, 0).$$

For many entropy functions, the minimal value $\varepsilon(1, 0, 0)$ is 0.
Moreover, from the averaging condition it follows, that

$$\varepsilon(p_1, p_2, p_3) \;\leq\; \varepsilon(p_{\varnothing}, p_{\varnothing}, p_{\varnothing}),$$

where $p_{\varnothing} = (p_1 + p_2 + p_3)/3$ is the average of the frequencies p_i.
For $n = 3$, we get $p_{\varnothing} = 1/3$, and $\varepsilon(1/3, 1/3, 1/3)$ is the maximal value.

... a brief Excursus

Consider a second table T' with the same size $|T| = |T'| = N$ and values $v_1, \ldots, v_n$ for the attribute Z, but possibly with different frequencies $p'_i = N'_i/N$, where N'_i is the number of ocurrences of the Z–value v_i in T'.

Merge: If $N'_1 = N_1 + N_2$, $N'_2 = 0$, and $N'_i = N_i$, for all $3 \leq i \leq n$, then T' is more homogeneous, and it should hold: $entropy_Z(T) \geq entropy_Z(T')$.

Below, we have $N_1 = 1$, $N_2 = 3$, $N'_1 = 4$, $N'_2 = 0$:

T :

A	Z
...	v_1
...	v_2
...	v_2
...	v_2
...	v_3
...	...
...	v_n

T' :

A	Z
...	v_1
...	v_1
...	v_1
...	v_1
...	v_3
...	...
...	v_n

Then $p'_2 = N'_2/N = 0/N = 0$ and

$$p'_1 = N'_1/N = (N_1 + N_2)/N = N_1/N + N_2/N = p_1 + p_2.$$

Average: If $N'_1 = N'_2 = (N_1 + N_2)/2$, and $N'_i = N_i$, for all $3 \leq i \leq n$, then T is more homogeneous, and it should hold $entropy_Z(T) \leq entropy_Z(T')$.

Below, we have $N_1 = 1$, $N_2 = 3$, $N'_1 = N'_2 = 2$:

T :

A	Z
...	v_1
...	v_2
...	v_2
...	v_2
...	v_3
...	...
...	v_n

T' :

A	Z
...	v_1
...	v_1
...	v_2
...	v_2
...	v_3
...	...
...	v_n