



**Prof. Dr. Adrian Ulges**

**Empolis Workshop “Machine Learning”**

# **Klassifikation**

**Hochschule RheinMain**

Department DCSM (*Design, Computer Science, Media*)

## ► Klassifikation

- Probabilistische Klassifikation (Naive Bayes)
- Maximum Entropy
- Entscheidungsbäume und Random Forests
- Feature Engineering
- **Python-Beispiel: News-Klassifikation**

# Generative vs. Diskriminative Ansätze

Prof. Adrian Ulges  
Fachbereich DCSM / Informatik  
Hochschule RheinMain

- Wir unterscheiden zwei **generelle Arten** von Klassifikatoren: **Generative Methoden** und **Diskriminative Methoden**
- Wir zerlegen  $P(\mathbf{c}|\mathbf{x})$  mit der **Bayes'schen Regel**

$$\text{posterior } P(\mathbf{c}|\mathbf{x}) = \frac{P(\mathbf{c}) \cdot P(\mathbf{x}|\mathbf{c})}{\cancel{P(\mathbf{x})}} \propto \text{prior } P(\mathbf{c}) \cdot \text{class-conditional density } P(\mathbf{x}|\mathbf{c})$$

- **Diskriminative Methoden**: Berechnen ein „direktes“ Modell für  $P(\mathbf{c}|\mathbf{x})$  (bzw. die **Entscheidungsgrenze**)
- **Generative Methoden**: Berechnen  $P(\mathbf{c}|\mathbf{x})$ , indem sie Modelle für  $P(\mathbf{c})$  und  $p(\mathbf{x}|\mathbf{c})$  aufstellen

- **Im Folgenden:** Eine einfache generative Methode (**Naive Bayes**) (später eine diskriminative (**logistische Regression**))
- Wir müssen also  $P(c)$  und  $P(x|c)$  berechnen
- Der **Prior  $P(c)$**  ist „einfach“ zu schätzen: Wir messen die Häufigkeit der Klassen in der Trainingsmenge
  - **Beispiel:** „Jede zehnte Mail ist SPAM“  
→  $P(„SPAM“)=0.1, P(„NO\_SPAM“)=0.9$
  - **Beispiel:** „relevante Dokumente sind sehr unwahrscheinlich“ →  $P(1) = 10^{-7}$
- Die **class-conditional Density  $P(x|c)$**  ist komplizierter zu schätzen.

- **Naive Bayes** ist ein generatives Lernverfahren (deshalb „**Bayes**“)

$$P(c|x) = \frac{P(c) \cdot P(x|c)}{P(x)}$$

- **Problem:** Ist der Merkmalsvektor **x** **hochdimensional**, ist die **class-conditional density** **P(x|c)** immer schwieriger zu lernen
- **Beispiel:** Sei **x** ein Boolescher Vektor mit **n** Merkmalen, **(x<sub>1</sub>, x<sub>2</sub>, ..., x<sub>n</sub>)**
- Dann entspricht **P(x|c)** einer **Wertetabelle**
- Die Wertetabelle hat **2<sup>n</sup> Einträge**

## Beispiel: Spamfilter

- Zwei Klassen (SPAM, HAM)
- Zwei Merkmale (n=2)
  - x<sub>1</sub>:** Absender ist bekannt
  - x<sub>2</sub>:** Term „viagra“ enthalten

**Klasse 1 (SPAM)**    **Klasse 2 (HAM)**

Klasse 1 (SPAM)			Klasse 2 (HAM)		
<b>x<sub>1</sub></b> (Absender)	<b>x<sub>2</sub></b> („Viagra“)	<b>P(x<sub>1</sub>, x<sub>2</sub>   spam)</b>	<b>x<sub>1</sub></b> (Absender)	<b>x<sub>2</sub></b> („Viagra“)	<b>P(x<sub>1</sub>, x<sub>2</sub>   ham)</b>
0	0	0.77	0	0	0.18
0	1	0.20	0	1	0.02
1	0	0.01	1	0	0.78
1	1	0.02	1	1	0.02

„20% aller Spam-Mails kommen von unbekannten Absendern und enthalten den Term „Viagra“

# Der „Curse of Dimensionality“

Prof. Adrian Ulges

Fachbereich DCSM / Informatik  
Hochschule RheinMain

- Ist **n groß**, können wir all diese Einträge nicht mehr zuverlässig lernen („**curse of dimensionality**“)
- Beispiel: Spam-Filter**  
*x als **Bag-of-Words**-Vektor (welche Terme kommen in einer E-Mail vor, welche nicht?)*

**n=1**

„Viagra“	$P(x \text{spam})$
0	0.78
1	0.22

**n=2**


„Viagra“	„nigeria“	$P(x \text{spam})$
0	0	0.58
0	1	0.22
1	0	0.16
1	1	0.04

**n=3**

„Viagra“	„nigeria“	„mom“	$P(x \text{spam})$
0	0	0	0.35
0	0	1	0.13
0	1	0	0.17
0	1	1	0.03
1	0	0	0.19
1	0	1	0.03
1	1	0	0.09
1	1	1	0.01

...

**n=10,000 ?**

Jeden dieser Einträge müssen wir aus einer Trainingsmenge **schätzen** (oder „lernen“)! 

- **Ziel:** **Vereinfache**  $P(\mathbf{x}|\mathbf{c})$ !
- **Ansatz:** Wir nehmen an, dass die einzelnen Einträge des Merkmalsvektors (gegeben die Klasse  $\mathbf{c}$ ) **unabhängig** sind (deshalb „naiv“)

- Dann folgt:

$$\begin{aligned} P(\mathbf{x}|\mathbf{c}) &= P(x_1, x_2, \dots, x_n | \mathbf{c}) \\ &= P(x_1 | \mathbf{c}) \cdot P(x_2 | \mathbf{c}) \cdot \dots \cdot P(x_n | \mathbf{c}) \end{aligned}$$

- Im **Beispiel**: Wir speichern nicht mehr eine **große Wertetabelle**, sondern für jedes Merkmal eine kleine ( $2^n$  Einträge  $\rightarrow$   ~~$2^n$~~   $2 \times n$  Einträge)
- Auch mit einer **begrenzten Trainingsmenge** können wir die benötigten Wahrscheinlichkeiten nun **zuverlässig** schätzen!

- Die **Entscheidungsregel** des Naive Bayes Klassifikators lautet also:

$$\begin{aligned} c^* &= \arg \max_c P(c|\mathbf{x}) \\ &= \arg \max_c \frac{P(c) \cdot P(\mathbf{x}|c)}{P(\mathbf{x})} && // \text{ ignoriere } P(\mathbf{x}), \text{ weil} \\ &= \arg \max_c P(c) \cdot P(\mathbf{x}|c) && \text{ nicht von } \mathbf{c} \text{ abhängig} \\ &= \arg \max_c P(c) \cdot P(x_1, x_2, \dots, x_n|c) \\ &= \arg \max_c P(c) \cdot \prod_{i=1}^n P(x_i|c) \end{aligned}$$

- Die **Form** von  $\mathbf{P}(\mathbf{x}_i|\mathbf{c})$  (z.B. *Bernoulli*-Verteilung) hängt vom konkreten Problem ab



# Naive Bayes: Beispiel

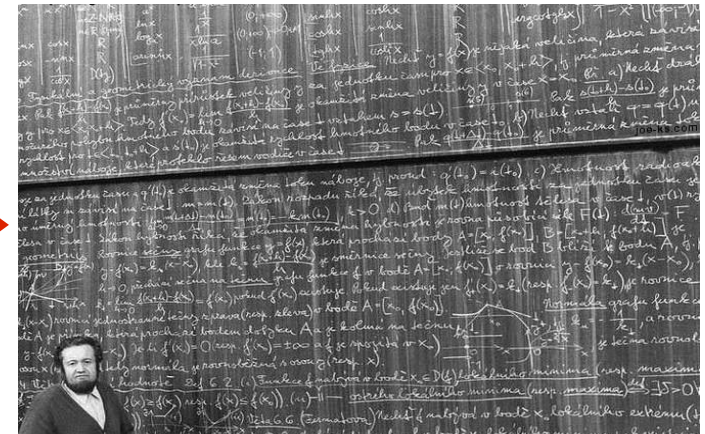
$$c^* = \arg \max_c P(c) \cdot \prod_{i=1}^n P(x_i|c)$$

- **SPAM-Filter**: E-Mails werden mittels **boolescher Bag-of-Kywords-Merkmale** repräsentiert
- Auf einer **Trainingsmenge** annotierter E-Mails haben wir die Daten rechts gelernt
- Wir wissen außerdem: **25%** aller E-Mails sind **SPAM**
- Wir klassifizieren folgende Mail:

term	P(term spam)	P(term ham)
viagra	0.05	0.001
money	0.10	0.01
bank	0.09	0.02
mom	0.02	0.05
exam	0.001	0.01
artificial	0.0001	0.001
intelligence	0.0001	0.002
please	0.08	0.07

i need urgently **money**.**please** help me i need 777USD money. i will give back your money.

if you are able to do my help .then i will give you my **bank** account no.



# Naive Bayes: Diskussion

Prof. Adrian Ulges

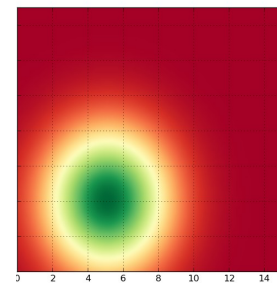
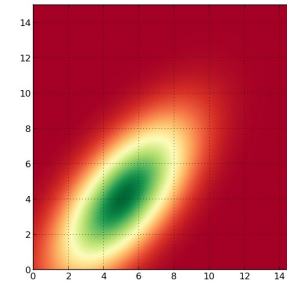
Fachbereich DCSM / Informatik  
Hochschule RheinMain

- Naive Bayes führt eine starke **Vereinfachung** mittels der **Unabhängigkeitsannahme** durch
- **Folgen:** siehe rechts
- Diese Annahme ist in der Praxis häufig verletzt (Beispiel: Klassifikation von **Münzen**: Sind **Durchmesser** und **Gewicht** unabhängig?)
- Naive Bayes erzielt bei sehr **hochdimensionalen Daten** oder sehr **kleinen Trainingsmengen** dennoch gute Ergebnisse
- Wird häufig als **Baseline** verwendet

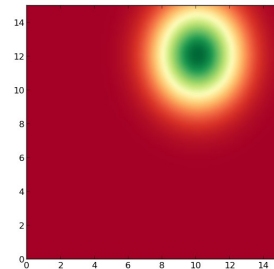
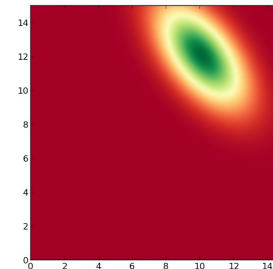
**Original**

**Naive Bayes**

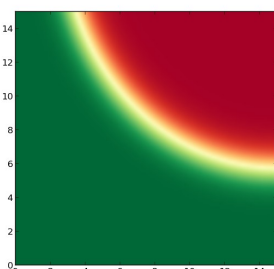
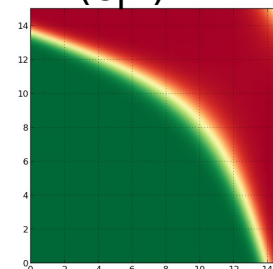
$P(x|C=1)$



$P(x|C=2)$



$P(c|x)$

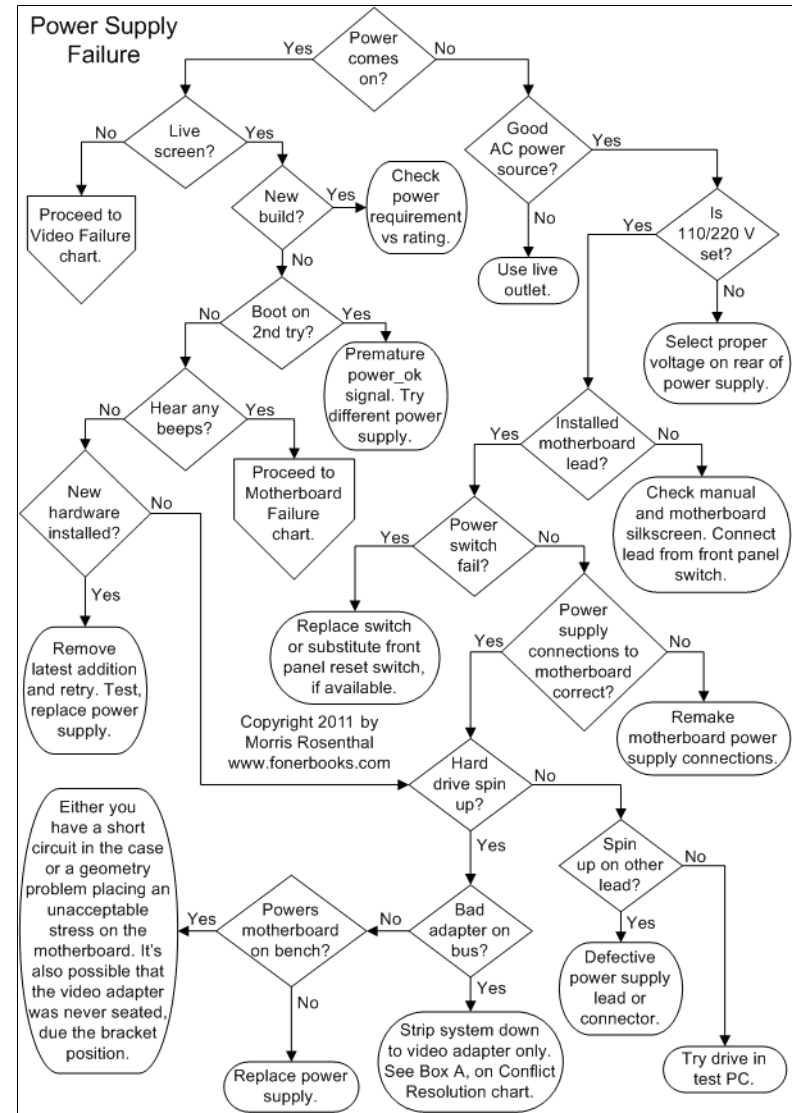
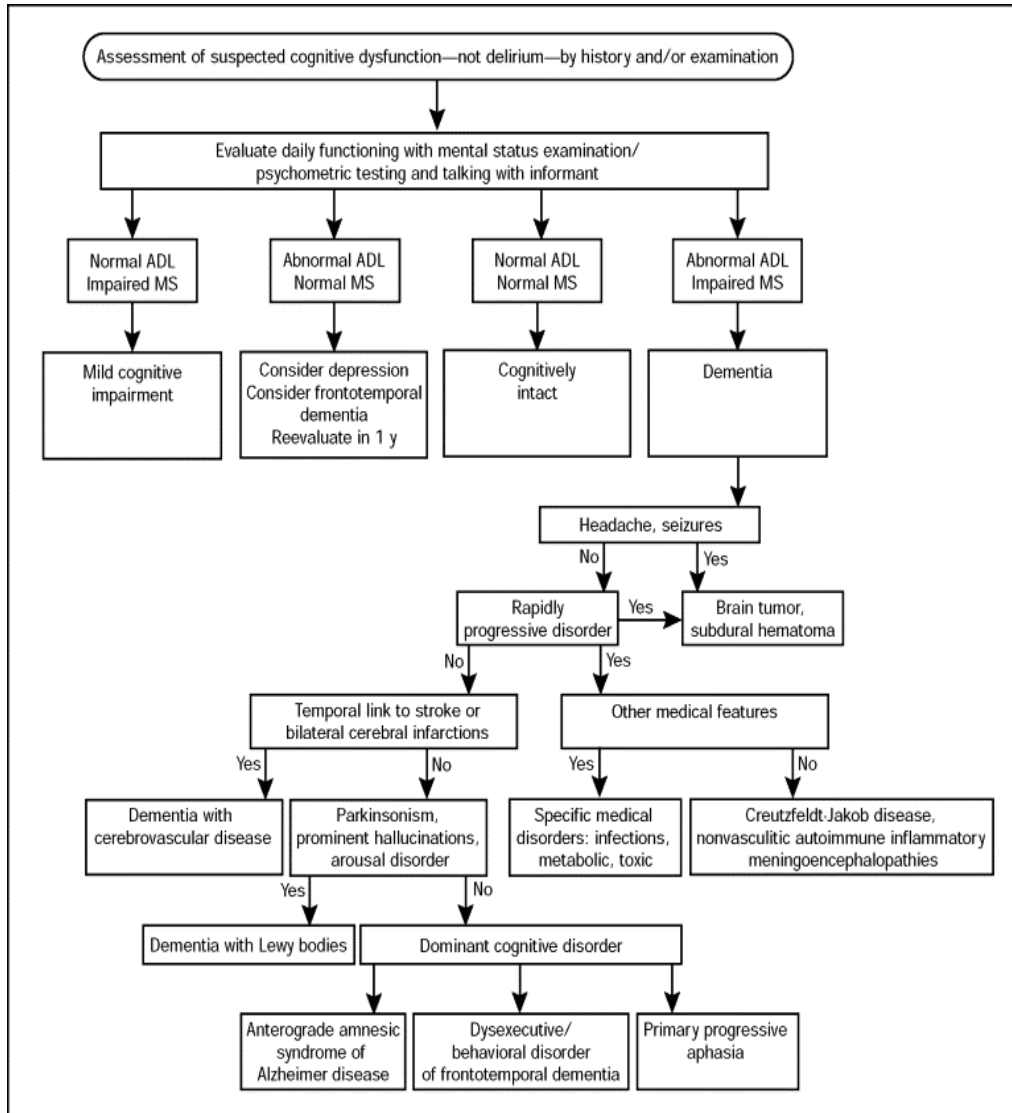


# Logistische Regression

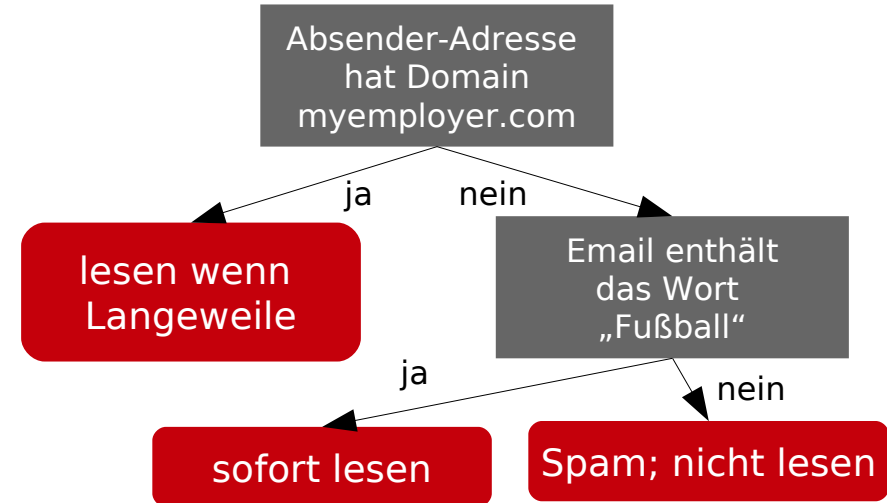
# Entscheidungs- bäume

# Entscheidungsbäume in Experten-Systemen

**Prof. Adrian Ulges**  
 Fachbereich DCSM / Informatik  
 Hochschule RheinMain

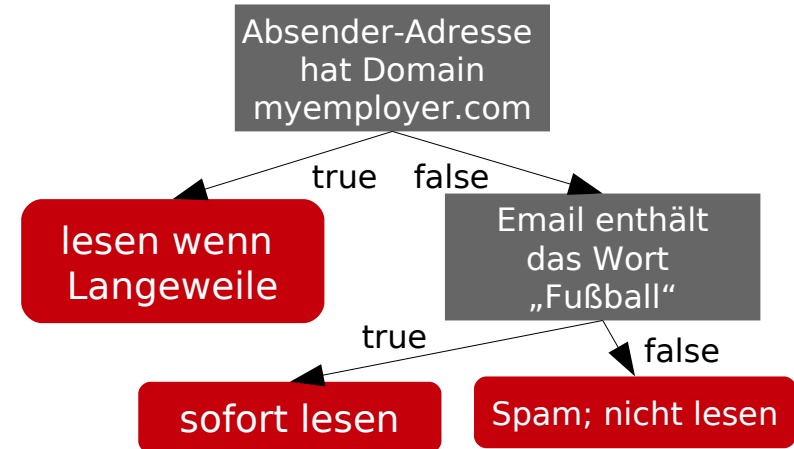


- Der **populärste** Klassifikator weltweit<sup>1</sup>
  - **flexibel** (fehlende und nicht-numerische Merkmale, Regression)
  - **einfach, schnell, transparent**
- **Ansatz**: Wähle Klasse auf Basis rekursiver einfacher Entscheidungen



- **Schlüsselfrage: Lernen** → Konstruiere Baumstruktur auf Basis gelabelter Samples





- **Ansatz: Rekursiver Aufbau** des Baums
- **Greedy-Strategie**: In jedem Schritt...
  - ... wähle das „beste“ **Merkmal**
  - ... splitte den Datensatz anhand des Merkmals in **Submengen**
- Verfahre rekursiv und breche ab, falls der Datensatz nur noch **Samples einer Klasse** enthält (wir bezeichnen den zugehörigen Knoten als „**rein**“ oder „**pure**“).

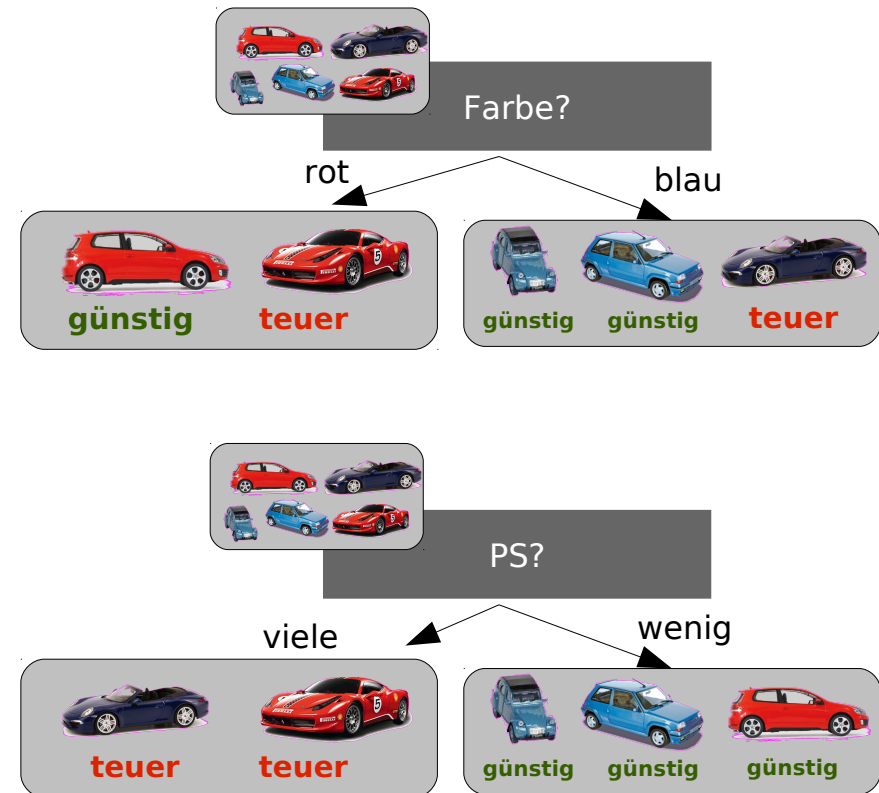


- Drei populäre Entscheidungsbaum-Modelle
  - **ID3**
  - **C4.5**
  - **CART**

# Beispiel

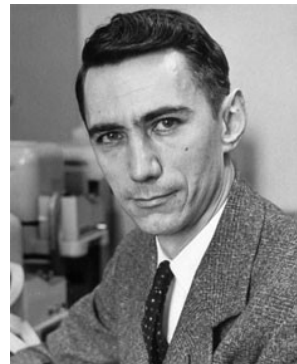
- Je „**reiner**“ („purer“) die **Verteilung der Klassen** in den Submengen, desto **besser** das Merkmal
- Wir wollen in den Kindknoten eine **niedrige Entropie** erreichen! Das resultierende Maß heißt „**Information Gain**“

Objekt	Farbe	PS	Klasse
	rot	viele	teuer
	blau	viele	teuer
	rot	wenig	günstig
	blau	wenig	günstig
	blau	wenig	günstig





- Wahrscheinlichkeitstheoretischer **Informationsbegriff**: Gegeben sei eine diskrete Verteilung mit **Wahrscheinlichkeiten**  $p_1, \dots, p_m$
- **Idee**: Die Verteilung  $\mathbf{p} = (p_1, \dots, p_m)$  enthält mehr Information, je mehr „**Unsicherheit**“ in ihr steckt
- **Informationsmaß**: Die Entropie  $H(p_1, \dots, p_m)$



Claude Shannon

# Entropie

$$H(p_1, \dots, p_m) = - \sum_{i=1}^m p_i \cdot \log_2(p_i) \quad (\text{mit } 0 \cdot \log_2(0) := 0)$$

Wir berechnen die Entropie für

a)  $(p_1, p_2, p_3, p_4) = (0, 0.5, 0.25, 0.25)$

b)  $(p_1, p_2, p_3, p_4) = (0, 1, 0, 0)$

- Sei **F** ein **Merkmal** (z.B. „Farbe“) mit Werten  $\mathbf{f}_1, \dots, \mathbf{f}_k$  (z.B. „rot“, „blau“, „silber“)
- Sei  $\mathbf{X} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$  die Menge von Samples ( $\mathbf{x}_i$  bezeichnet einen **Merkmalsvektor**,  $\mathbf{y}_i$  ein **Label**)
- Das Merkmal **F** zerlegt **X** in Teilmengen  $\mathbf{X}_1, \dots, \mathbf{X}_k$  mit

$$X_k := \{(x, y) \in X \mid F(x) = f_k\}$$

- **Ziel:** Finde das Merkmal, das zu den „**puresten**“ **Klassenlabels** in  $\mathbf{X}_1, \dots, \mathbf{X}_k$  führt!
- Gegeben eine Menge von Samples  $\mathbf{X}'$ , definieren wir die **Häufigkeitsverteilung der Klassen** in  $\mathbf{X}'$

$$(p_1, \dots, p_C) \text{ mit } p_c := \frac{\#\{(x, y) \in X' \mid y = c\}}{\#X'}$$

- Wir definieren die **Entropie** einer **Menge  $X'$**  als die Entropie der zugehörigen Klassenverteilung

$$H(X') := H(p_1, \dots, p_C)$$

- Dann lautet der **Information Gain**

$$Gain(X, F) := H(X) - \sum_{k=1}^K \frac{\#X_k}{\#X} \cdot H(X_k)$$

- Wir wählen für den Split das Merkmal  **$F^*$** , welches  **$Gain(X, F)$  maximiert**
- **$F^*$**  entspricht dem Merkmal, das die **Entropie der entstehenden Submengen** (gewichtet nach ihrer Mächtigkeit) minimiert

**Gegeben:**  $X$  (Menge von Samples) und  $M$  (Menge von Merkmalen)

## function **build\_tree**( $X$ , $M$ ):

Wenn alle Samples in  $X$  dasselbe Label  $L$  haben:

    return ( $L, -, \emptyset$ )      *// Blattknoten: Label L, kein Merkmal, keine Kinder*

Wenn  $M = \emptyset$ :      *// keine Merkmale mehr zum Splitten*

    Bestimme das häufigste Label  $L$  in  $X$

    return ( $L, -, \emptyset$ )

Finde das beste Merkmal  $F^* = \operatorname{argmax}_{F \in M} \text{Gain}(X, F)$

Splitte  $X$  gemäß  $F^*$  in Submengen  $X_1, \dots, X_k$

return ( $-, F^*, \{ \text{build\_tree}(X_1, M \setminus \{F^*\}),$

$\text{build\_tree}(X_2, M \setminus \{F^*\}),$

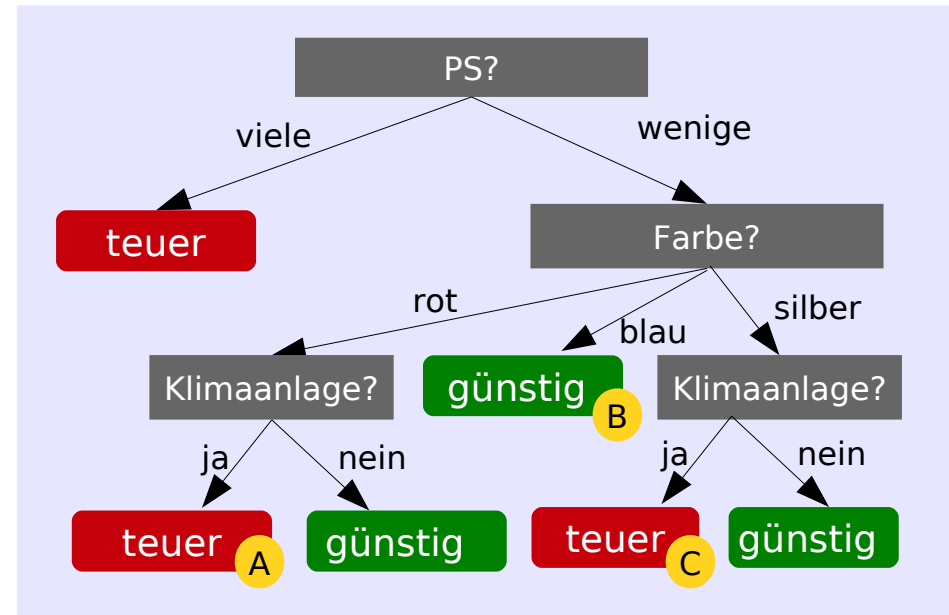
        ...

$\text{build\_tree}(X_k, M \setminus \{F^*\}) \} )$

- Spätere **Varianten** von Entscheidungsbäumen (namentlich, **C4.5** und **CART**) führen **Verbesserungen** und **Erweiterungen** ein
  - Behandlung **fehlender** Merkmale
  - Behandlung **kontinuierlicher** Merkmale
  - Bessere Generalisierung mittels **Pruning**
  - Anwendung für **Regressionsprobleme**

- Wir können mit Entscheidungsbäumen auch dann **klassifizieren**, wenn das Testsample **fehlende Merkmale** aufweist!
- Ansatz**: Traversiere in **alle Kindknoten**, sammle die Ergebnis-Labels ein, und führe ein **Voting** durch!
- Beispiel**: Klassifiziere

x = ( Farbe:           ?,  
      PS:            wenige,  
      Klimaanlage: ja )

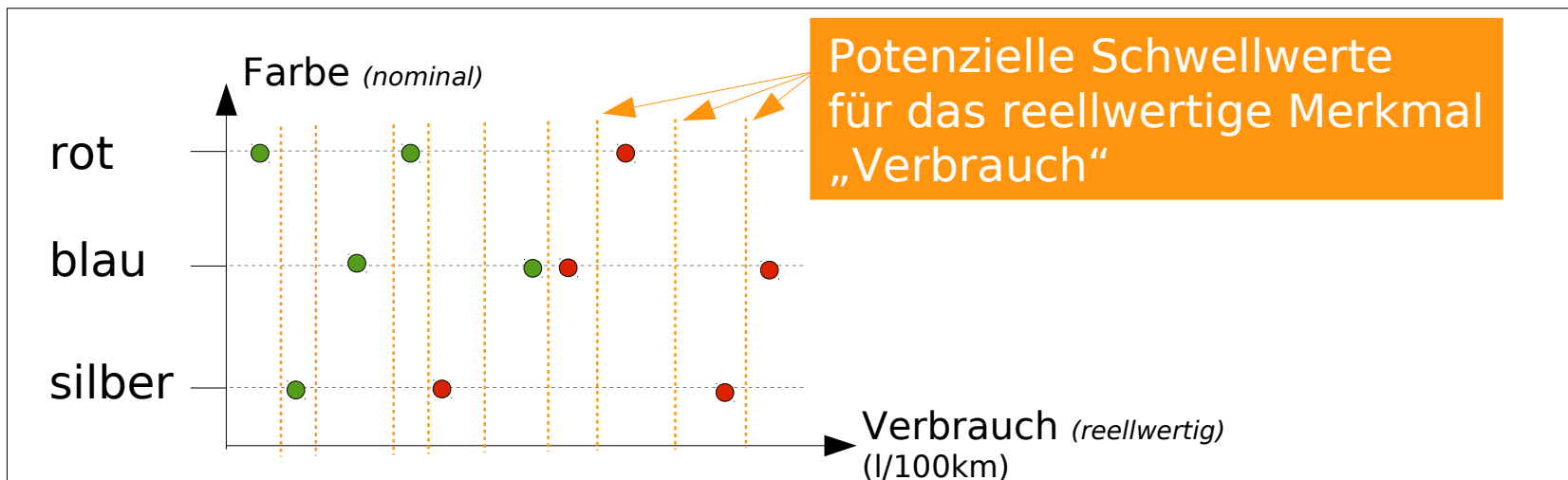


- Wir besuchen (*da Farbe unbekannt*) die Blätter **A**, **B** und **C**
- 2 Stimmen** für **teuer**,  
**1 Stimme** für **günstig**  
→ klassifiziere x als **teuer**

- Wir können Samples mit **fehlenden Merkmalen** auch zum **Training** verwenden!
- Fehlende Merkmale werden bei der Berechnung des Information Gain **ignoriert**
- Samples mit fehlenden Merkmalen werden aber auf **Kindknoten verteilt**. Das heißt: Das fehlende Merkmal wird **„geschätzt“**!
- **Heuristiken:**
  - Schätzung = häufigster Wert in der **Klasse**
  - Schätzung = häufigster Wert im aktuellen **Knoten**
  - Verteile das Trainingssample **anteilmäßig** auf alle Kindknoten (*gemäß der Anzahl der Samples in den Kindknoten*)



- **ID3** unterstützt nur Merkmale mit **endlich vielen Ausprägungen**. In der Praxis sind viele Merkmale aber **kontinuierlich** (d.h., **reellwertig**).
- **Ansatz**: Gegeben Merkmal **F**, wähle Schwelle **T** und führe einen binären Split durch:  **$F(x) \geq T$**  vs.  **$F(x) < T$**

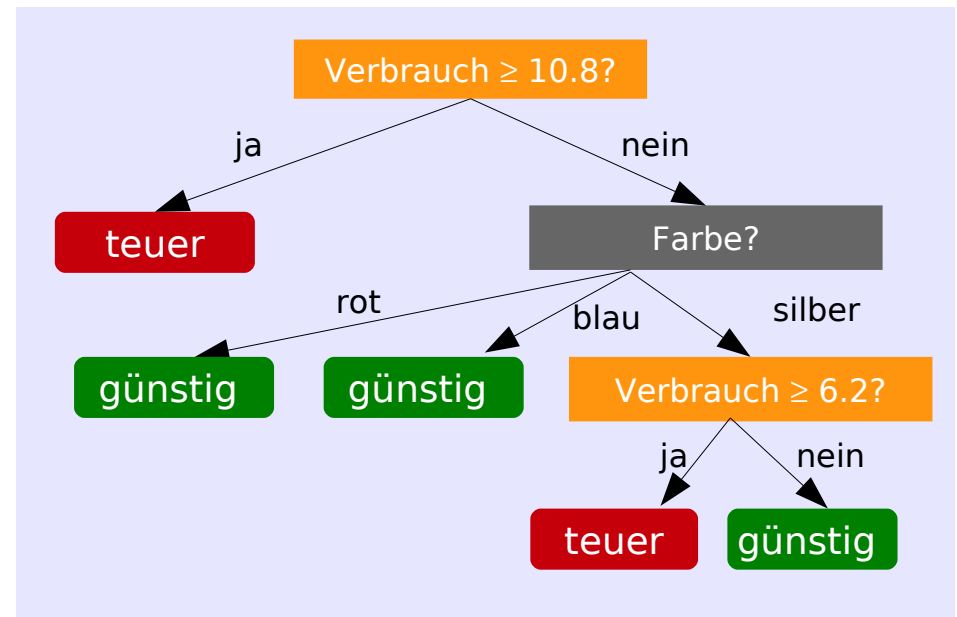
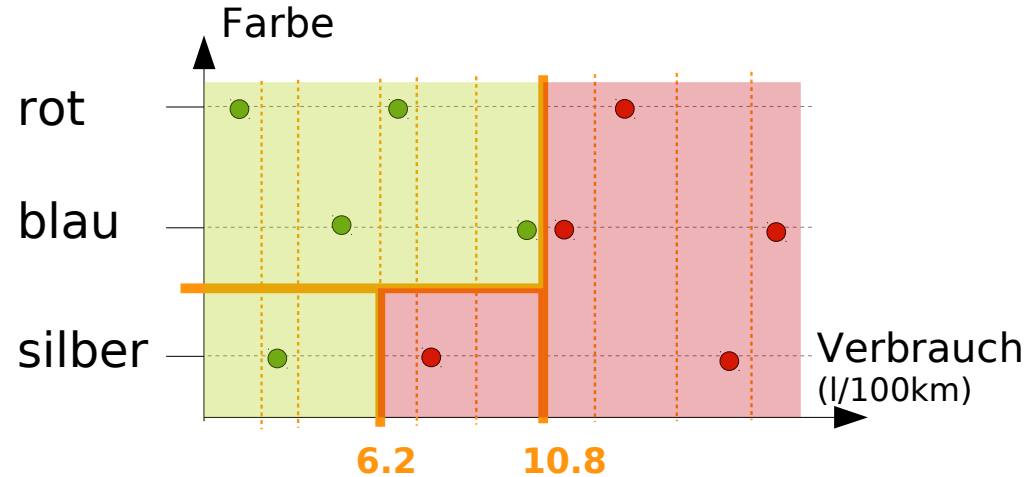


- Lernen wird **teurer**: Prüfe für jedes kontinuierliche Merkmal **alle** möglichen Schwellwerte **T**, die zwischen je zwei Merkmalswerten liegen

# Reellwertige Merkmale (Beispiel)

Prof. Adrian Ulges  
Fachbereich DCSM / Informatik  
Hochschule RheinMain

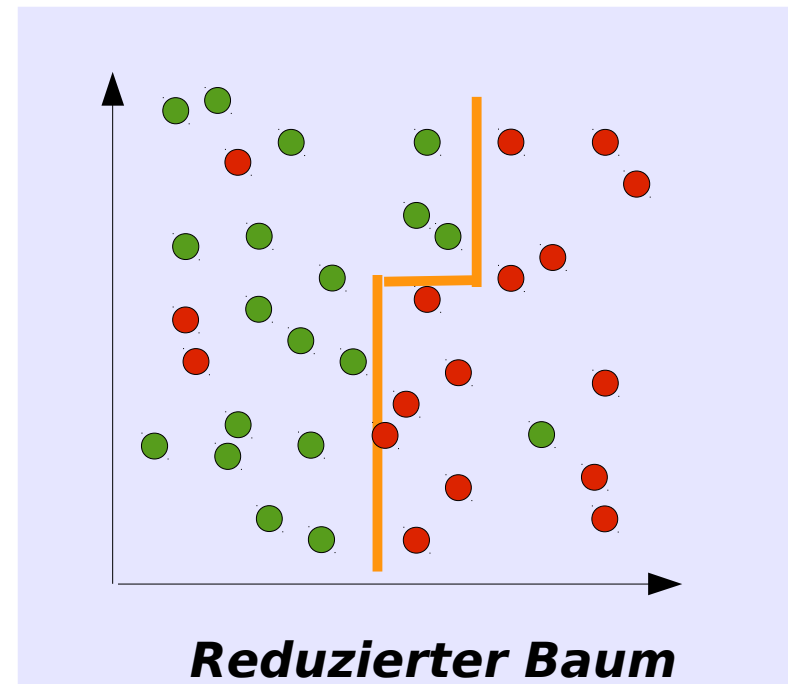
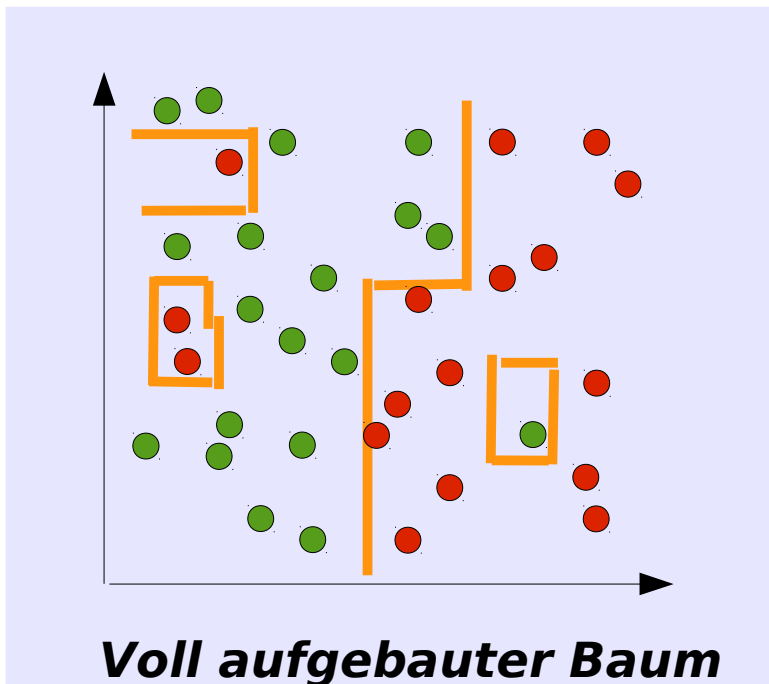
- Wir können innerhalb desselben Asts mehrere Splits für **dasselbe** reellwertige **Merkm**al durchführen (nämlich mit unterschiedlichen Schwellwerten **T**)
- Beispiel: *rechts***



# Entscheidungsbäume: Pruning

Prof. Adrian Ulges  
Fachbereich DCSM / Informatik  
Hochschule RheinMain

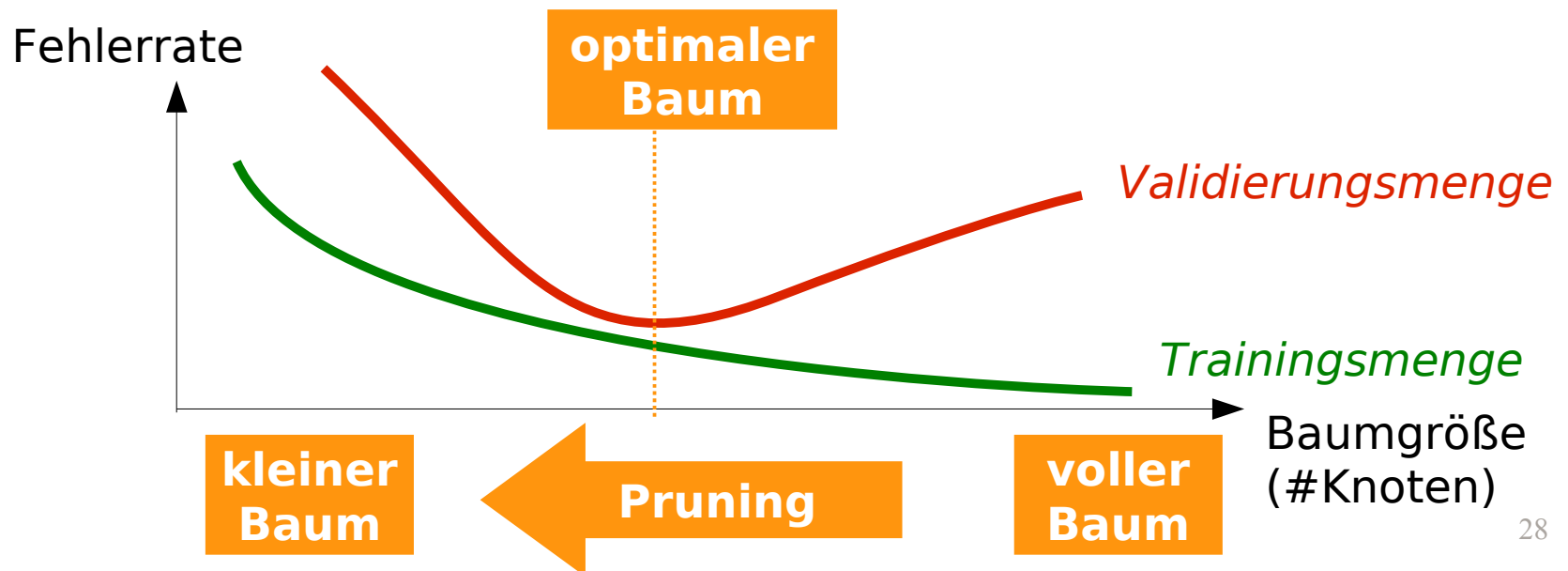
- Ein voller Aufbau des Baums führt zu **Overfitting**!
- **Ziel**: reduziere die Größe/Tiefe des Baums mittels **Pruning** → vereinfache die Entscheidungsgrenze!
- **Entferne Blätter** → es entstehen „**Mischnoten**“, in denen der Klassifikator die **häufigste Klasse** wählt.



# Entscheidungsbäume: Pruning

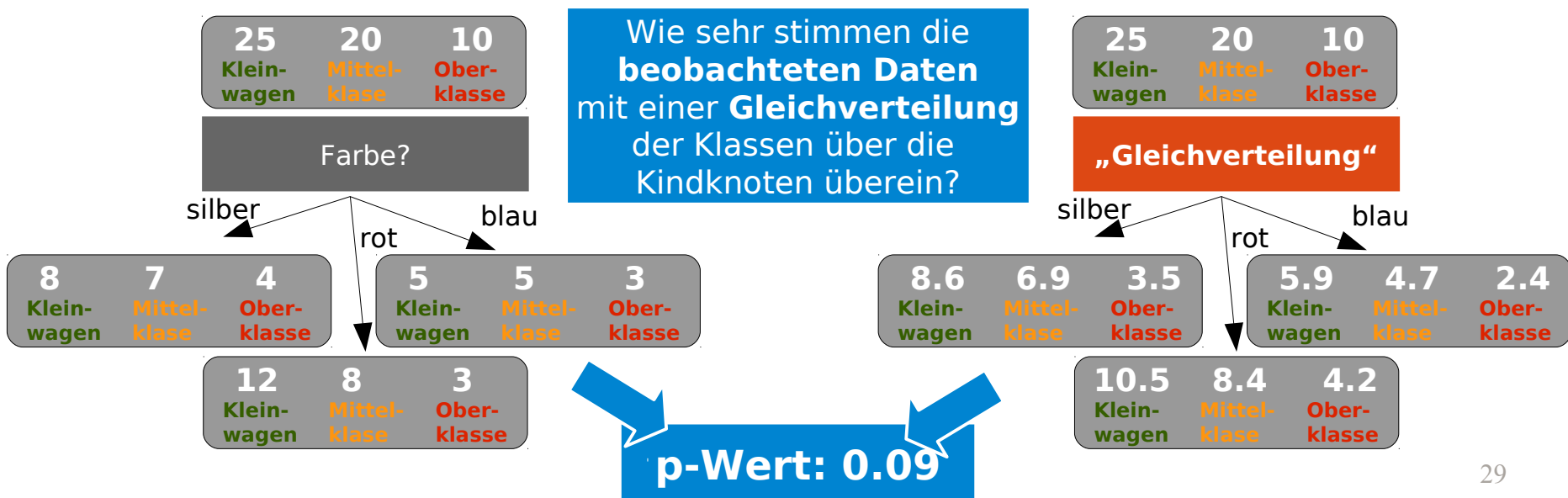
Prof. Adrian Ulges  
Fachbereich DCSM / Informatik  
Hochschule RheinMain

- **Ansatz: *Pruning mit Validierungsmenge***
  - Teile die Samples in eine **Trainingsmenge** und eine ***Validierungsmenge***
  - Trainiere einen vollen Baum auf der Trainingsmenge
  - Entferne sukzessive Blattknoten, solange der ***Fehler*** auf der Validierungsmenge reduziert wird



- **Ansatz 2: *Statistische Tests***

- Wir wollen entscheiden, ob ein Split „nützlich“ ist. **Schlüsselfrage**: Ist das zugehörige **Merkmal** mit den Klassen-**Labels korreliert**?
- Diesen Sachverhalt können wir z.B. mit dem **Chi-Quadrat-Unabhängigkeits-Test** prüfen



- **Ansatz 2: *Statistische Tests***
- ...
- Der ***p-Wert*** des Tests gibt die ***Wahrscheinlichkeit*** an, die gegebene Verteilung zu beobachten, falls Merkmal und Klassenlabels ***unabhängig wären***.
- **Strategie:** Entferne einen Split, falls  $p > p_T$
- Der Parameter  $p_T$  wird manuell eingestellt (oder kann seinerseits wieder auf einer Validierungsmenge gelernt werden)

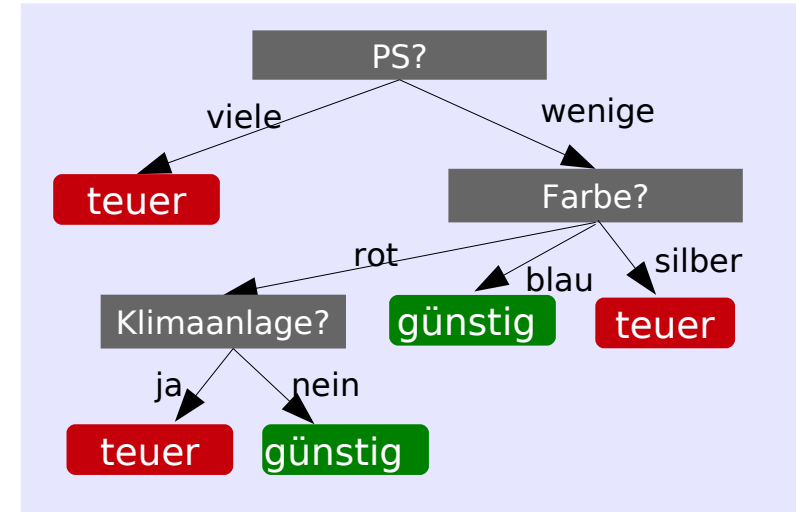
# Entscheidungsbäume: Pruning

Prof. Adrian Ulges

Fachbereich DCSM / Informatik  
Hochschule RheinMain

## Ansatz 3: Regelbasiertes **Post-Pruning** (C4.5)

- Transformiere den Baum in eine Menge von **if-else-Regeln** (jeder Pfad von der Wurzel zu einem Blatt-knoten wird eine Regel)
- Entferne Teile der if-Bedingung und prüfe ob sich die **Genauigkeit der Regel** (Validierungsmenge!) verbessert
- Sortiere die Regeln** nach ihrer Genauigkeit und wende sie der Reihe nach an



1)	( PS=viele )	-> teuer
2)	( PS=wenige ^ Farbe=blau )	-> günstig
3)	( PS=wenige ^ Farbe=silber )	-> teuer
4)	( PS=wenige ^ Farbe=rot ^ Klima=ja )	-> teuer
5)	( PS=wenige ^ Farbe=rot ^ Klima=nein )	-> günstig

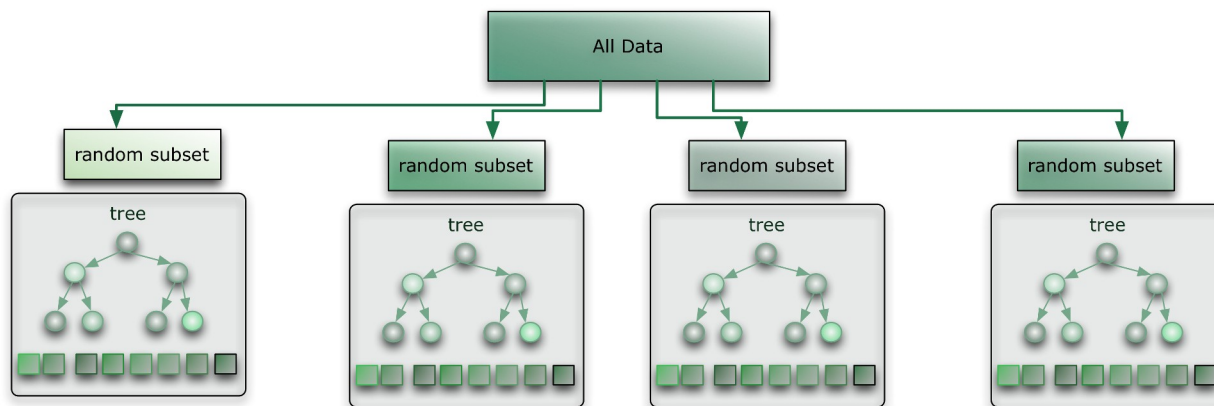
Evaluere Regel (3) gegen:

3a)	( PS=wenige )	-> teuer
3b)	( Farbe=silber )	-> teuer

und behalte die beste Variante.

## Alternativer Ansatz zu Pruning

- Verwende **voll ausgebaute** Bäume ... aber nicht nur einen, sondern **viele** (deshalb „random **forest**“)!
- Der Aufbau der einzelnen Bäume ist dabei **randomisiert** (deshalb „**random** forests“)
- Samples werden mit **jedem Baum klassifiziert**, und ein **Voting** über alle Bäume wird durchgeführt.
- Random Forests sind eine **Ensemble-Methode**. Das heißt, viele einfache Klassifikatoren (=Bäume) werden mit dem Ziel einer genaueren globalen Entscheidung kombiniert.





- **Ziel:** Die einzelnen Bäume sollten möglichst **genau klassifizieren** und möglichst **unabhängig** voneinander sein
- **Ansätze zur *Randomisierung***
  - Wähle jedes Split-Merkmal zufällig aus (***random split***)
  - Zufällige Auswahl der Trainingssamples (***bagging***)
  - Wähle in jedem Knoten eine Submenge von Merkmalen aus denen das Beste für den Split gewählt wird (***random input selection***)

# Random Forests: Evaluation

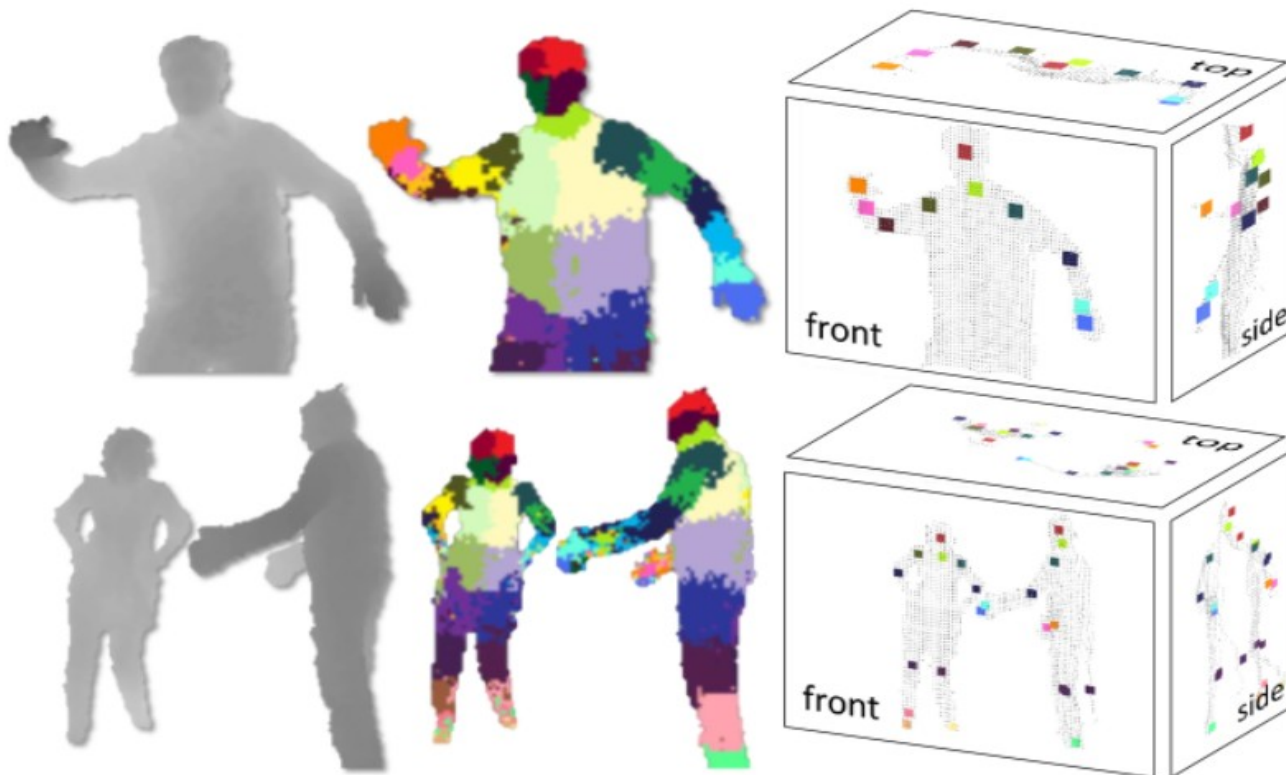
**Prof. Adrian Ulges**

Fachbereich DCSM / Informatik  
Hochschule RheinMain

- Fehlerraten auf diversen Datensätzen

Data set	Adaboost	Selection	Forest-RI single input	One tree
Glass	22.0	20.6	21.2	36.9
Breast cancer	3.2	2.9	2.7	6.3
Diabetes	26.6	24.2	24.3	33.1
Sonar	15.6	15.9	18.0	31.7
Vowel	4.1	3.4	3.3	30.4
Ionosphere	6.4	7.1	7.5	12.7
Vehicle	23.2	25.8	26.4	33.1
German credit	23.5	24.4	26.2	33.3
Image	1.6	2.1	2.7	6.4
Ecoli	14.8	12.8	13.0	24.5
Votes	4.8	4.1	4.6	7.4
Liver	30.7	25.1	24.7	40.6
Letters	3.4	3.5	4.7	19.8
Sat-images	8.8	8.6	10.5	17.2
Zip-code	6.2	6.3	7.8	20.6
Waveform	17.8	17.2	17.3	34.0
Twonorm	4.9	3.9	3.9	24.7
Threenorm	18.8	17.5	17.5	38.4
Ringnorm	6.9	4.9	4.9	25.7

- Anwendungsbeispiel: **Body Part Recognition** in der **Kinect-Konsole**



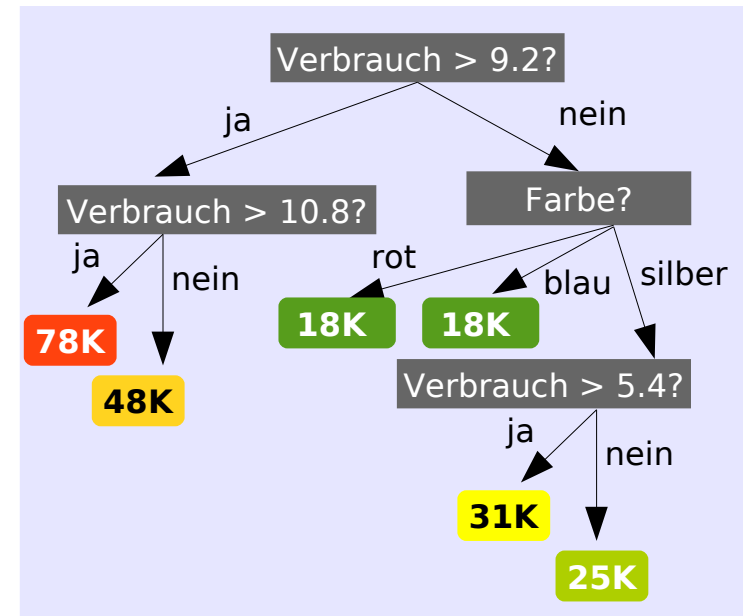
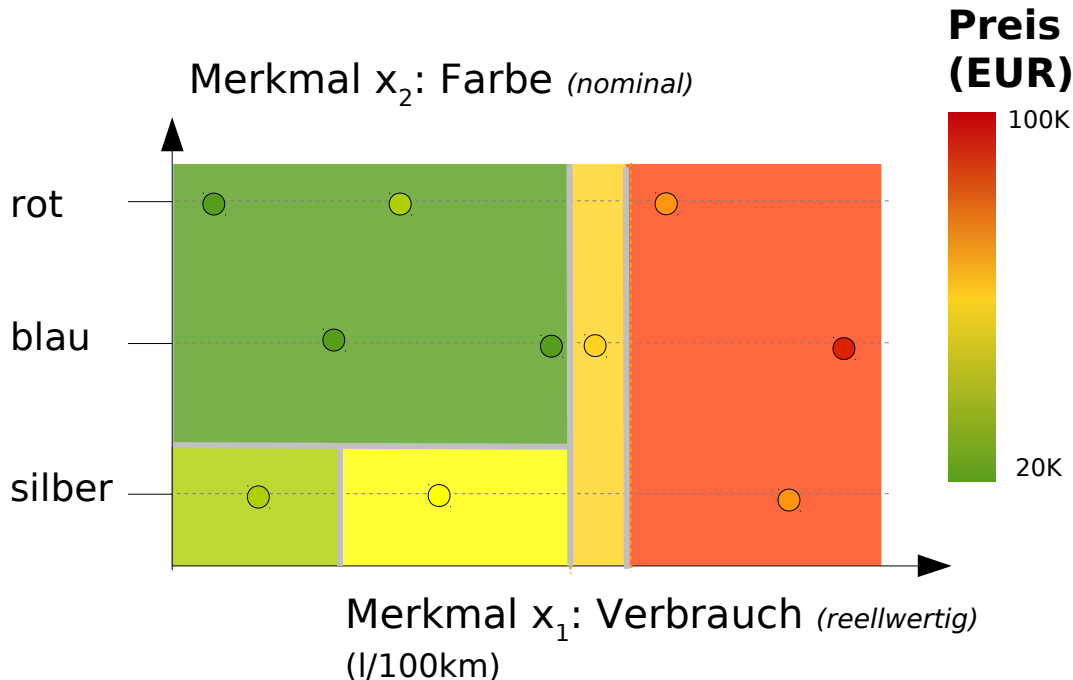
- Wir können Bäume auch zur Regression verwenden (**CART**: „**Classification and Regression Trees**“)
- **Gegeben**: Samples  $(\mathbf{x}, y)$  mit Merkmalsvektor  $\mathbf{x}$  (wie vorher) und  $y$  (jetzt kein **Klassen-Label** mehr, sondern ein **numerischer Wert**)
- Berechnung des **Rückgabewertes**
  - **Klassifikation**: Ein **Klassenlabel** pro Blatt (per Voting)
  - **Regression**: Ein **numerischer Wert** pro Blatt (berechnet durch Mittlung aller Werte im Blatt)
- Wahl des **besten Merkmals** für einen Split
  - **Klassifikation**: maximaler **Information Gain**
  - **Regression**: Minimaler **Sum-of-Squares-Error**

$$F^* := \arg \min_F \sum_{k=1}^K \sum_{(x,y) \in X_k} (y - \bar{y}_k)^2$$

# Entscheidungsbäume: Regression

Prof. Adrian Ulges  
Fachbereich DCSM / Informatik  
Hochschule RheinMain

- **Pruning:** (wieder) per **Validierungsmenge**
- **Beispiel:** Ein Regressionsbaum, der den Preis von Autos vorhersagt



- **Vorteile**

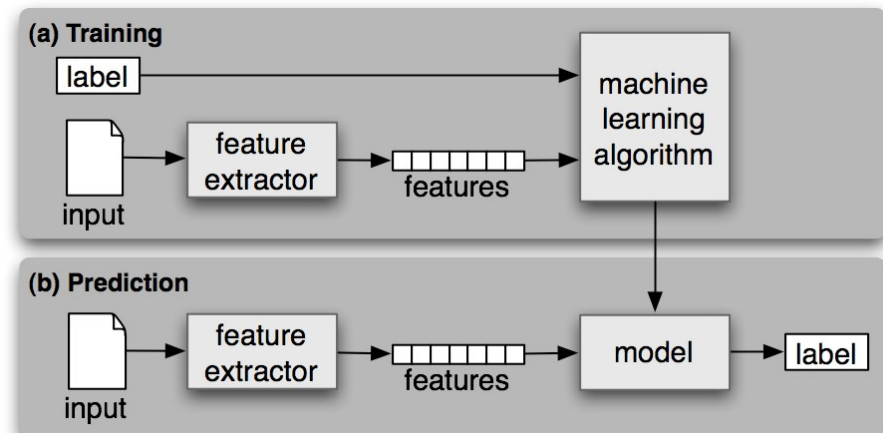
- (sehr) **schnelle** Klassifikation
- **Transparenz** der Ergebnisse
- Behandlung **fehlender Merkmale** möglich
- Behandlung **nicht-numerischer** Merkmale möglich

- **Nachteile**

- **Potenzielles Overfitting**  
(Beispiel: Nicht-lineare Entscheidungsgrenzen)
- Decision Trees sind nur dann brauchbar, wenn **wenige entscheidende Merkmale** eine gute Entscheidungsgrenze liefern.

# Feature Engineering

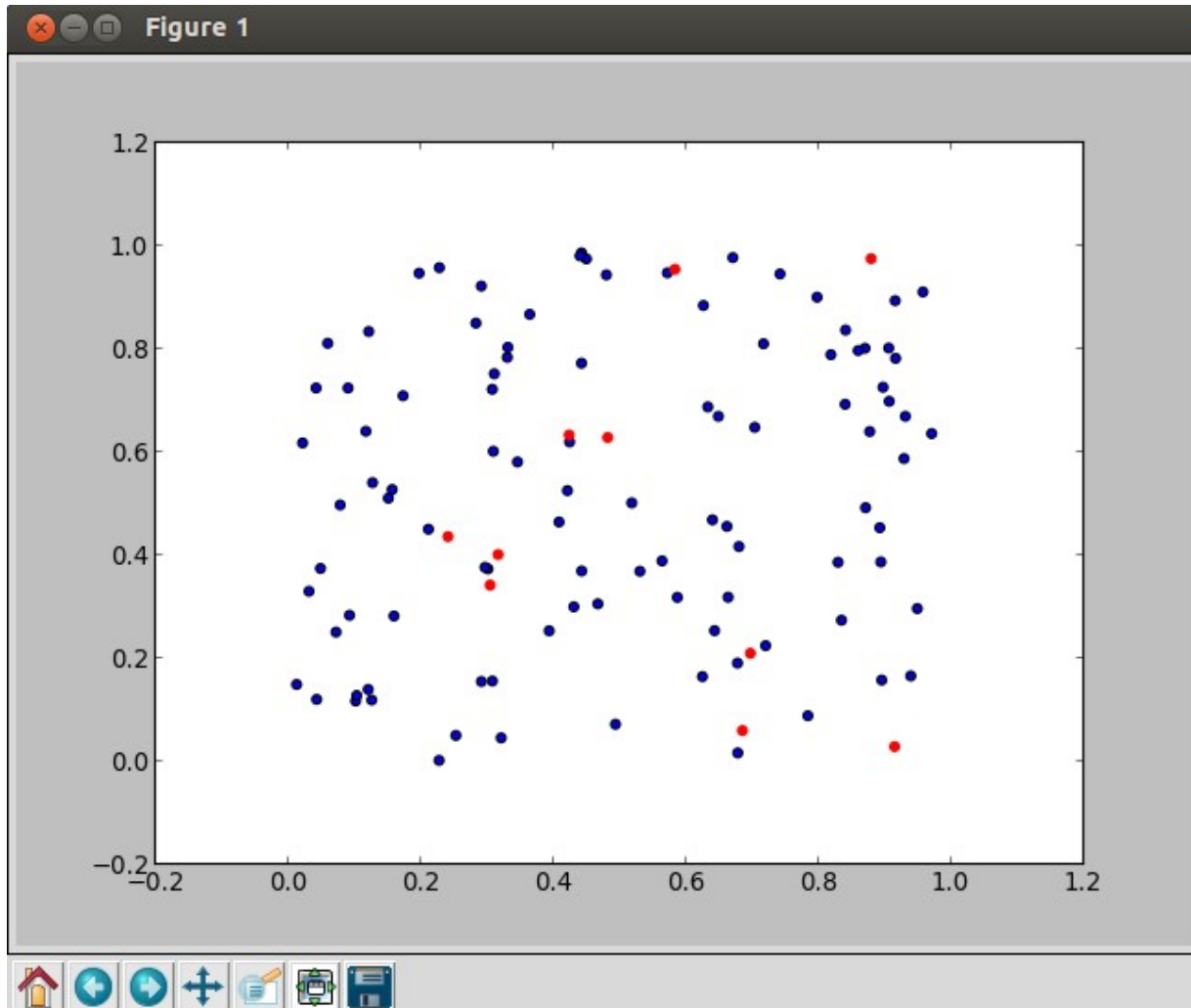
- **Merkmalsextraktion** ist der wichtigste Teil vieler ML-Anwendungen („**garbage-in-garbage-out**“)
- **Herausforderungen:** Gute Features sind (1) **anwendungsabhängig** und (2) abhängig vom **Klassifikator**
- **Im Folgenden:**
  - Einige Eigenschaften guter Features
  - Anmerkungen zur Merkmals-Selektion
  - Merkmale für Text





# Der „Curse of Dimensionality“

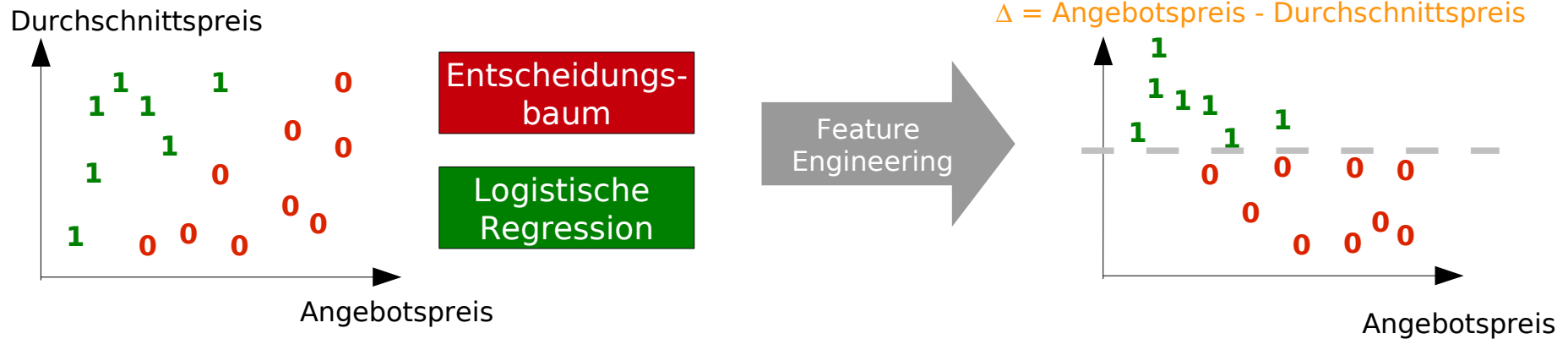
**Prof. Adrian Ulges**  
Fachbereich DCSM / Informatik  
Hochschule RheinMain



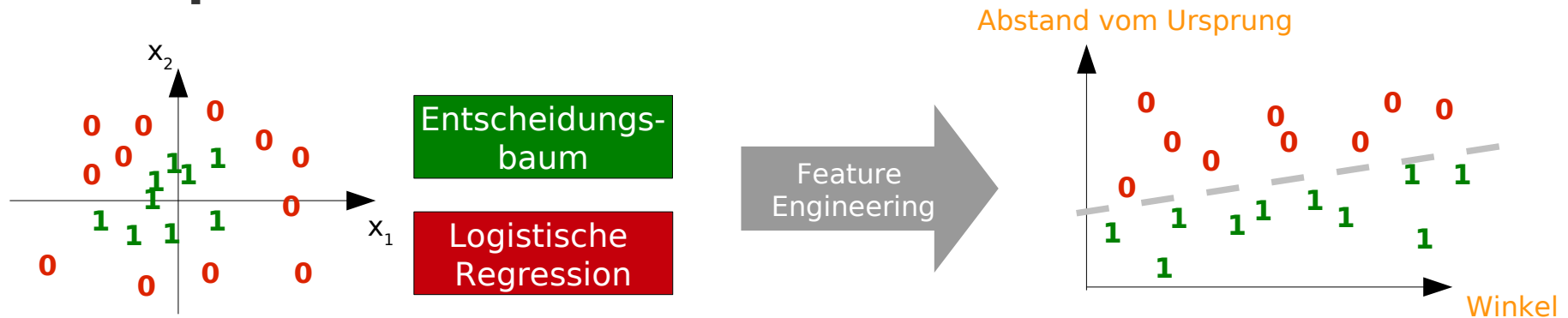
# Was „gute“ Merkmale sind hängt vom Klassifikator ab

Prof. Adrian Ulges  
Fachbereich DCSM / Informatik  
Hochschule RheinMain

- **Beispiel 1:** Kauft ein Kunde auf ebay?



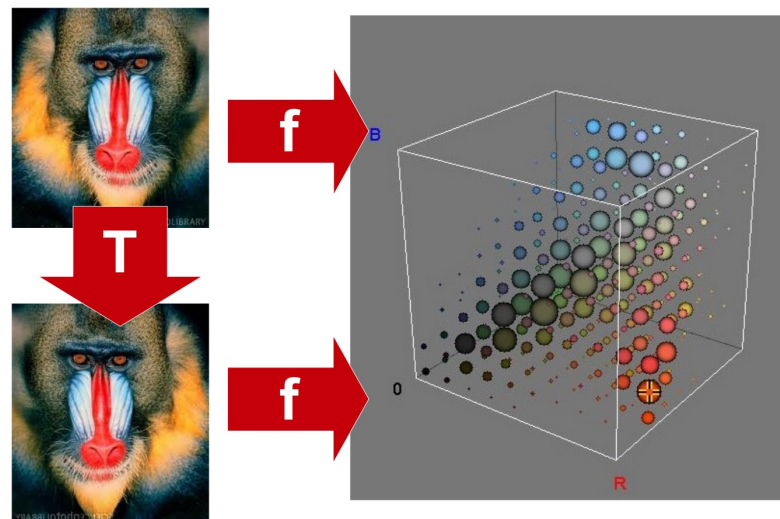
- **Beispiel 2**



# Eigenschaften „guter“ Features

Prof. Adrian Ulges  
Fachbereich DCSM / Informatik  
Hochschule RheinMain

- **Kompaktheit**: Die **Anzahl** der Merkmale sollte gering sein
  - Grund 1: Effizienz
  - Grund 2: Overfitting
- **Diskriminativität**: Die Merkmale sollten uns erlauben, Klassen voneinander zu **trennen**
- **Invarianz**: Die Merkmale sollten robust gegenüber **Transformationen** der Eingabedaten sein:  $\mathbf{f}(\mathbf{T}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$



- Es existieren zahlreiche Techniken zur **Reduktion von Merkmalen („dimensionality reduction“)**
  - PCA, LDA, PLSA, MDS, Auto-Encoder-Networks...
  - Einige sind überwacht, die meisten **unüberwacht**
  - **Ziel: Projektion** hochdimensionaler Eingabedaten auf niedrigdimensionalen Raum
- **Klassifikatoren** bieten oft eine intrinsische Merkmalsselektion
- **Beispiel 1 - Logistische Regression**
  - Durch die Regularisierung werden die **Gewichte** vieler Variablen 0
  - Das bedeutet: Die zugehörigen Variable spielen keine Rolle für die Entscheidung des Klassifikators

- **Beispiel 2: *Entscheidungsbäume***
  - Entscheidungsbäume verwenden **nicht jedes** Feature zum Split!
  - Auch die verwendeten Features können wir nach ihrer Bedeutung („**feature importance**“) bewerten
- **Ansatz 1: *permutation importance***
  - Idee: Um die Bedeutung eines **Features F** zu messen, lasse es **wegfallen**
  - **Vertausche** zufällig die Werte von **F** zwischen den Samples der Testmenge
  - Messe wie stark der **Klassifikationsfehler** ansteigt
- **Ansatz 2: *Gini importance***
  - Jeder **Split** im Baum **erhöht die Purity** um einen Wert  $\Delta$
  - Für jedes Merkmal **summieren** wir sämtliche  $\Delta$ -Werte aller Splits in allen Bäumen eines Forests auf
  - Je **höher dieser Wert**, desto besser das Merkmal

# Merkmale zur Beschreibung von Text