# Lernen von Entscheidungsbäumen

Volker Tresp

# Anforderungen an Methoden zum Datamining

- Schnelle Verarbeitung großer Datenmengen
- Leichter Umgang mit hochdimensionalen Daten
- Das Lernergebnis soll vermittelbar sein
- Das Lernergebnis soll hohe Qualität (Vorhersagegenauigkeit) besitzen
- Das System soll gute Ergebnisse liefern, ohne dass der Anwender viel Domänenwissen besitzt ("off the shelf")
- Erlernte Entscheidungsbäume besitzen viele dieser Qualitäten, allerdings auf Kosten der Performanz
- Die Performanz kann durch Ensemble Methoden (Bagging, Boosting) wesentlich verbessert werden

# Entscheidungsbäume

- Entscheidungsbäume werden traditionell zur Fehlerfindung eingesetzt
- Aufgrund der Antworten auf eine Sequenz von Fragen wird entschieden, welche Fehlerklasse vorliegt und welche Aktion unternommen werden sollte
- Die Abfrage erfolgt nicht in einem starren Muster, sondern hängt von den gegebenen Antworten ab
- Diese Abfolge von Fragen stellt man üblicherweise als Baum dar, mit dem Wurzelknoten an oberster Stelle; an unterster Stelle befinden sich die Blätterknoten mit den Diagnosen (Klassifikationen) und Handlungsanweisungen

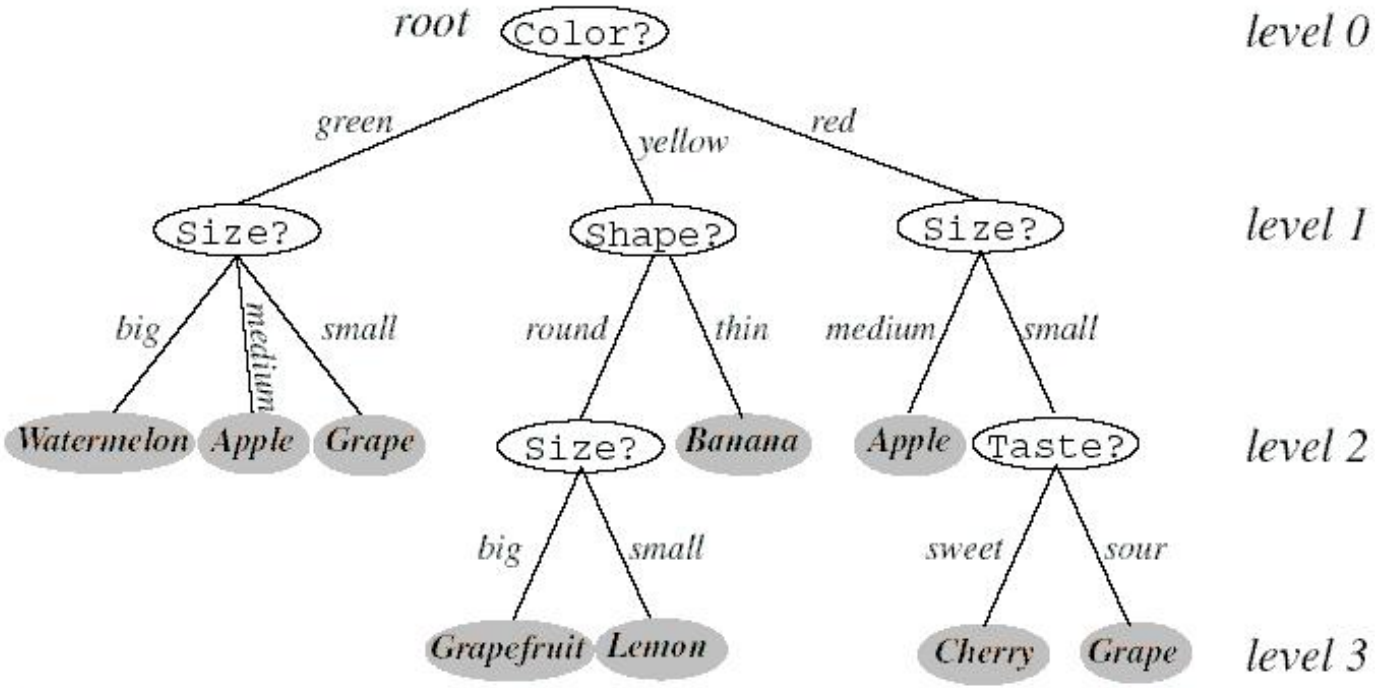



**FIGURE 8.1.** Classification in a basic decision tree proceeds from top to bottom. The questions asked at each node concern a particular property of the pattern, and the downward links correspond to the possible values. Successive nodes are visited until a terminal or leaf node is reached, where the category label is read. Note that the same question, `Size?`, appears in different places in the tree and that different questions can have different numbers of branches. Moreover, different leaf nodes, shown in pink, can be labeled by the same category (e.g., **Apple**). From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. 

## Binäre Entscheidungsbäume

- Entscheidungsbäume mit beliebigem Verzweigungsfaktor können auf binäre Entscheidungsbäume reduziert werden

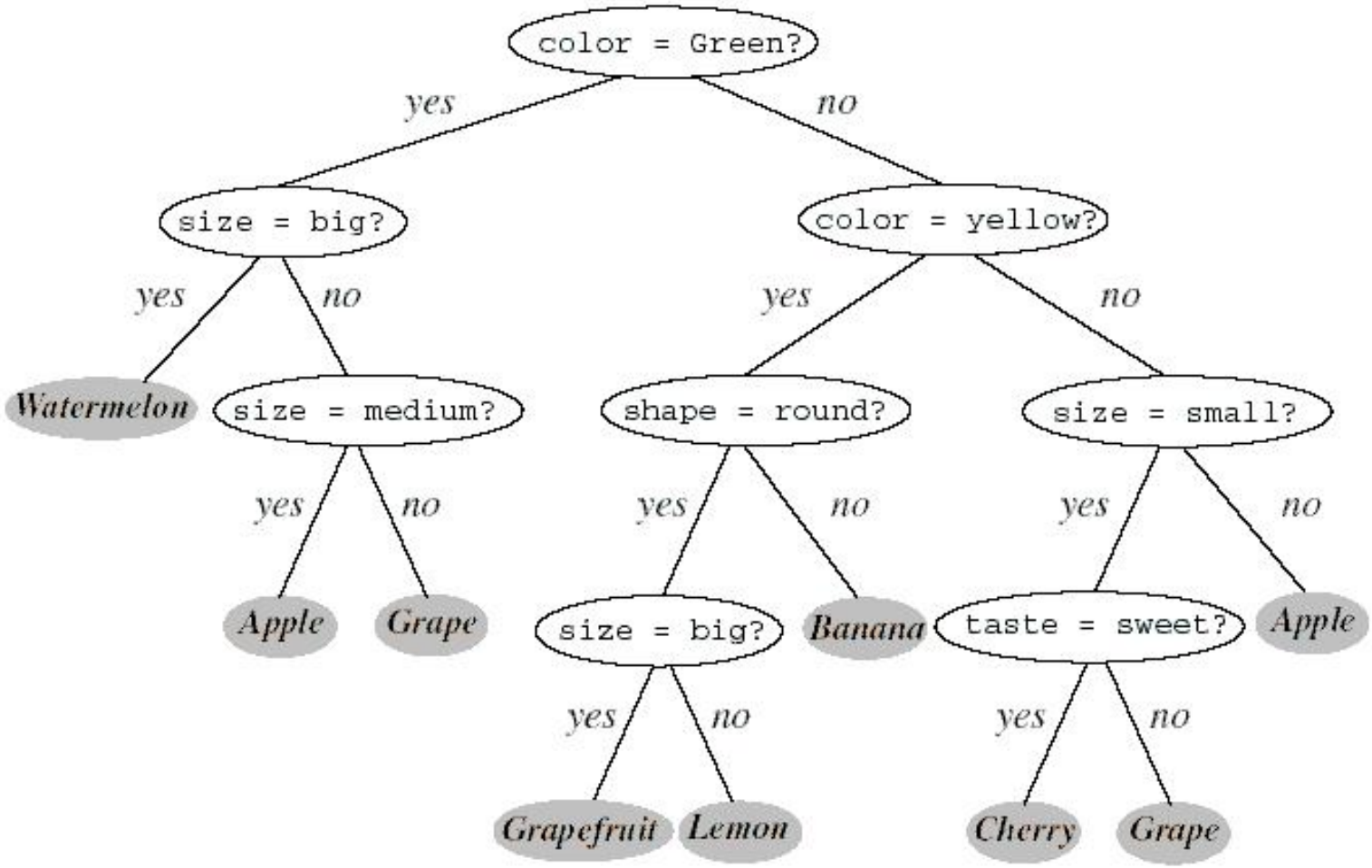


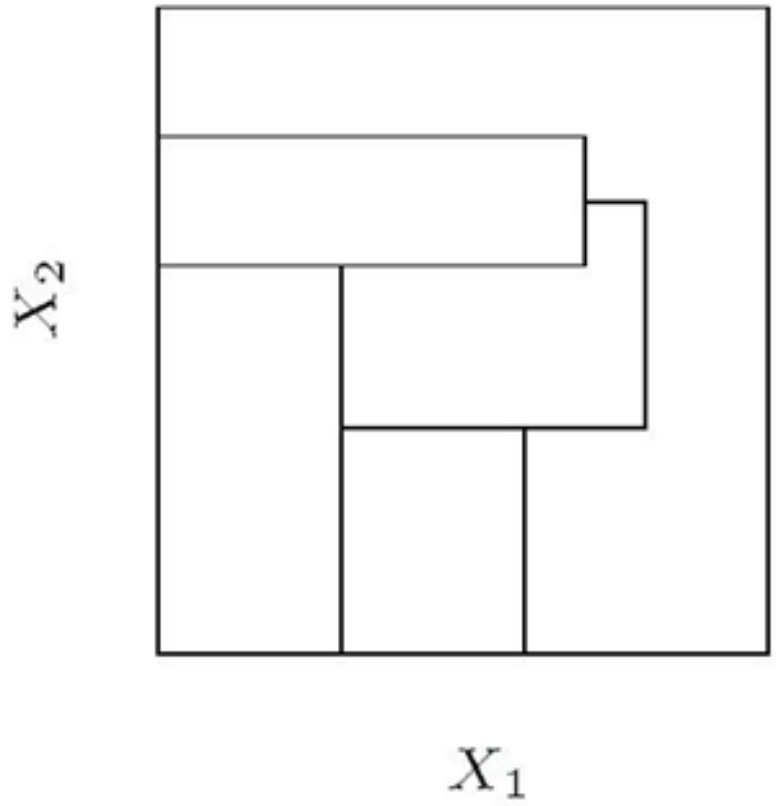

**FIGURE 8.2.** A tree with arbitrary branching factor at different nodes can always be represented by a functionally equivalent binary tree—that is, one having branching factor $B = 2$ throughout, as shown here. By convention the "yes" branch is on the left, the "no" branch on the right. This binary tree contains the same information and implements the same classification as that in Fig. 8.1. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

## Erlernen von binären Entscheidungsbäumen

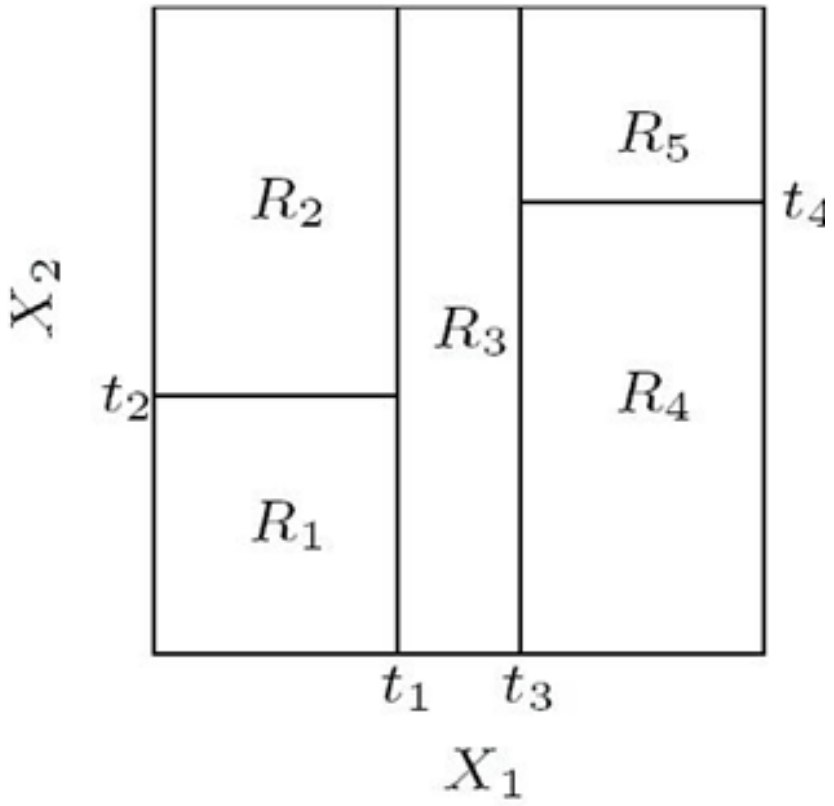
- Im Folgenden konzentrieren wir uns auf das Erlernen binärer Entscheidungsbäume können
- Der wichtigste Ansatz ist bekannt als CART (Classification and Regression Trees) (Breiman et al.)
- Mehr oder weniger equivalent sind ID3 (Interactive Dichotomizer, Version 3) und C4.5 (Quinlan et al.)
- Unser Fokus liegt auf CART

# CART: Grundidee

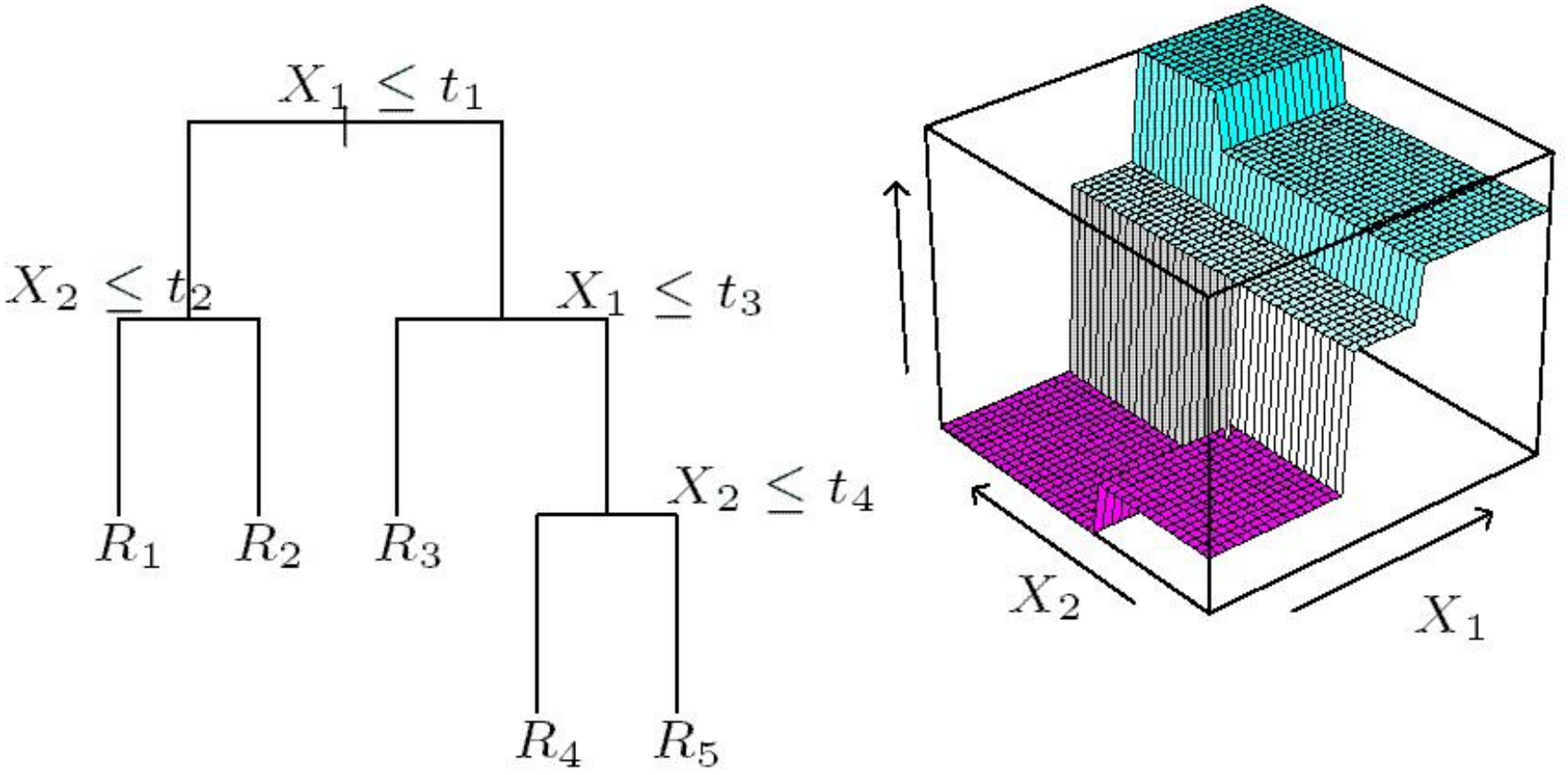
- Grundidee: Der Eingangsraum wird in Rechtecke partitioniert
- An die Daten eines Rechtecks wird ein einfaches Modell angepasst
- Regression: Das Modell ist eine Konstante
- Klassifikation: Das Modell ist eine konstante Klassenzuordnung
- Wir beginnen mit Regression




Entscheidungsgrenzen, die nicht durch CART gebildet werden können




Typische CART Entscheidungsgrenzen


$X_1 \leq t_1$
$X_2 \leq t_2$
$X_1 \leq t_3$
$X_2 \leq t_4$
$R_1$
$R_2$
$R_3$
$R_4$
$R_5$
$X_2$
$X_1$

# Regression: Darstellung der Lösung

- Der Eingangsraum wird partitioniert in Regionen $R_1, \ldots R_M$, die den Blättern im Baum entsprechen
- Gehört eine Datenpunkt $\mathbf{x}$ in Region $R_m$, so wird ihm die Zahl $c_m$ zugeordnet
- Im einfachsten Fall ist $c_m$ einfach der Mittelwert der Trainingsdaten aus $R_m$
- Man kann kompakt schreiben

$$f(\mathbf{x}) = \sum_{m=1}^{M} c_m I(\mathbf{x} \in R_m)$$

wobei $I(true) = 1$ und $I(false) = 0$

# Regression: Erlernen des Baumes

- Das Finden der besten Partitionierung ist exponentiell
- Wir beginnen mit einem gierigen (greedy) Ansatz
- Wir beginnen mit allen Daten (Wurzelknoten); alle Daten gehören dem Wurzelknoten;
- Angenommen, nach der Split-Strategie wird die $j$-te Variable ausgewählt; als Schwellwert sei $s$ definiert
- Wir teilen die Daten ein zwei Mengen ein, je nachdem ob die $j$-te Komponente kleiner oder größer als $s$ ist
- Für jede der beiden Mengen berechnen wir Mittelwerte $c_1$, $c_2$ und die Summe der Restfehler

$$\sum_{y_i | x_{i,j} \leq s} (y_i - c_1)^2 + \sum_{y_i | x_{i,j} > s} (y_i - c_2)^2$$

- Die optimale Aufteilung (Split) für Variable $j$ ist diejenige, für die die Summe der Restfehler minimal ist; $y_i$ ist die Zielgröße zu $\mathbf{x}_i$; $s$ wird dann der optimale Split.

# Regression: Erlernen des Baumes(2)

- Strategie: *Für jeden Blattknoten: Man berechnet den optimalen Split für alle Variablen und führt den Split aus, der für diesen Knoten optimal ist optimal ist*
- Anschließend werden die Daten den entsprechenden Regionen zugeordnet und der Prozess für die Daten jeder Region wie beschrieben rekursiv wiederholt
- Wie tief sollte der Baum werden? Auf jeden Fall nicht beliebig tief (Überanpassung)
- Man kann nicht ein einfaches Haltekriterium definieren: möglicherweise bringt ein einzelner Split nur eine geringe Verbesserung, eine Abfolge von zwei Splits jedoch eine dramatische Verbesserung (X-or)
- CART Strategie: erst einen übermäßig großen Baum Lernen (bis z.B. in jedem Knoten maximal 5 Datenpunkte zugeordnet werden)

# Knotenunreinheit (node impurity)

- Betrachten wir den Blattknoten $m$ im Baum $T$ mit $N_m$ Datenpunkte in Region $R_m$. Dann ist die node impurity (Knotenunreinheit) für Regression definiert als

$$Q_m(T) = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} (y_i - c_m)^2$$

- Man wählt diejenige Variable mit dem Splitkriterion $s$, die die Reduktion der Knotenunreinheit

$$Q_m(T, s, j) = Q_m(T) - \frac{N_{m_1}}{N_m} Q_{m_1}(T) - \frac{N_{m_2}}{N_m} Q_{m_2}(T)$$

maximiert. Hierbei sind $m_1$ und $m_2$, die Knoten, die sich durch das Splitten der $j$-ten Variablen nach Kriterion (Schwellwert) $s$ ergeben

- Man wählt Variable und Split, so dass $Q_m(T, s, j)$ maximiert wird

# Regression: Cost Complexity Pruning

- Anschließend: Cost-Complexity Pruning nach Kriterium

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|$$

- Hier ist $|T|$ die Anzahl der Blätter im Baum $T$ und $\alpha \geq 0$ ist ein Regularisierungsparameter
- $C_\alpha(T)$ ist somit quadratischer Restfehler plus $\alpha$ mal der Anzahl der Blätter
- Zu jedem $\alpha$ gibt es einen optimalen minimalen Unterbaum (subtree) $T_\alpha$ vom gewachsenen Baum

## Regression: Weakest Link Pruning

- Weakest Link Pruning: Man kollabiere denjenigen Knoten, der den kleinsten Anstieg im Cost-Complexity-Pruning-Kriterium bewirkt (Regression: im quadratischen Restfehler bedeutet) (beachte: dies ist unabhängig von $\alpha$)
- In dieser Sequenz ist (zu jedem $\alpha$) der beste Unterbaum $T_\alpha$ enthalten
- Das optimale $\alpha$ wird über Kreuzvalidierung bestimmt

# CART: Klassifikation

- Sei

$$\widehat{p}_{m,k} = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} I(y_i = k)$$

der Anteil der Daten in $R_m$, der in Klasse $k$ fällt (empirische Wahrscheinlichkeit). $N_m$ ist die Anzahl der Datenpunkte in $R_m$

- Ein Datenpunkt in $R_m$ wird der Klasse zugeordnet mit

$$k(m) = \arg\max_k \widehat{p}_{m,k}$$

# Klassifikation: Knotenunreinheit (node impurity)

- Node impurity für $K$ Klassen
  - Klassifikationsfehler: $Q_m(T) = 1 - \widehat{p}_{k(m),m}$
  - Gini Index: $Q_m(T) = 1 - \sum_{k=1}^{K} \widehat{p}_{m,k}^2$
  - (Kreuz-) Entropie (*deviance*): $Q_m(T) = -\sum_{k=1}^{K} \widehat{p}_{m,k} \log \widehat{p}_{m,k}$
- Als Spezialfall für 2 Klassen mit

$$\widehat{p}_m = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = 2) :$$

  - Klassifikationsfehler: $Q_m(T) = 1 - max(\widehat{p}_m, 1 - \widehat{p}_m)$
  - Gini Index: $Q_m(T) = 2\widehat{p}_m(1 - \widehat{p}_m)$
  - Entropie: $Q_m(T) = -\widehat{p}_m \log \widehat{p}_m - (1 - \widehat{p}_m) \log(1 - \widehat{p}_m)$
- Der Gini Index wird typischerweise benutzt zum Wachsen des Baumes

- Ein Baum wird gewachsen, bis alle Daten eines Knoten zur selben Klasse gehören
- Der Klassifikationsfehler wird typischerweise zum Prunen (Beschneiden) benutzt

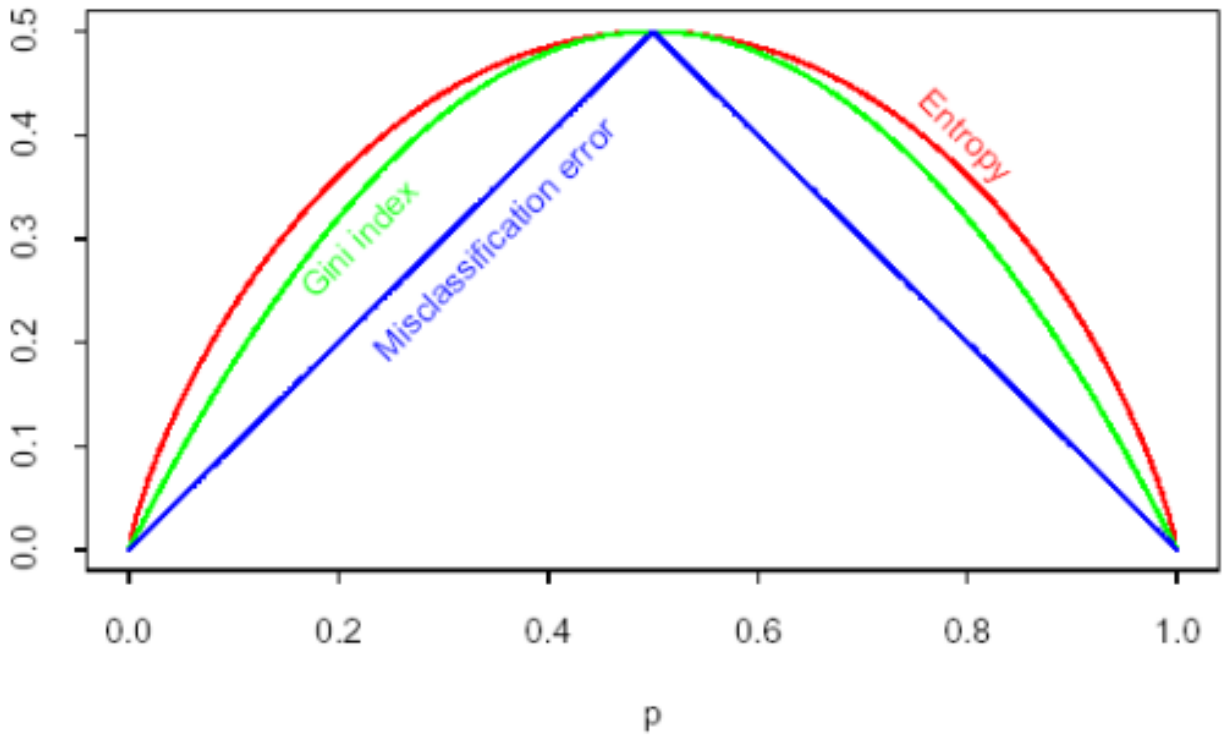


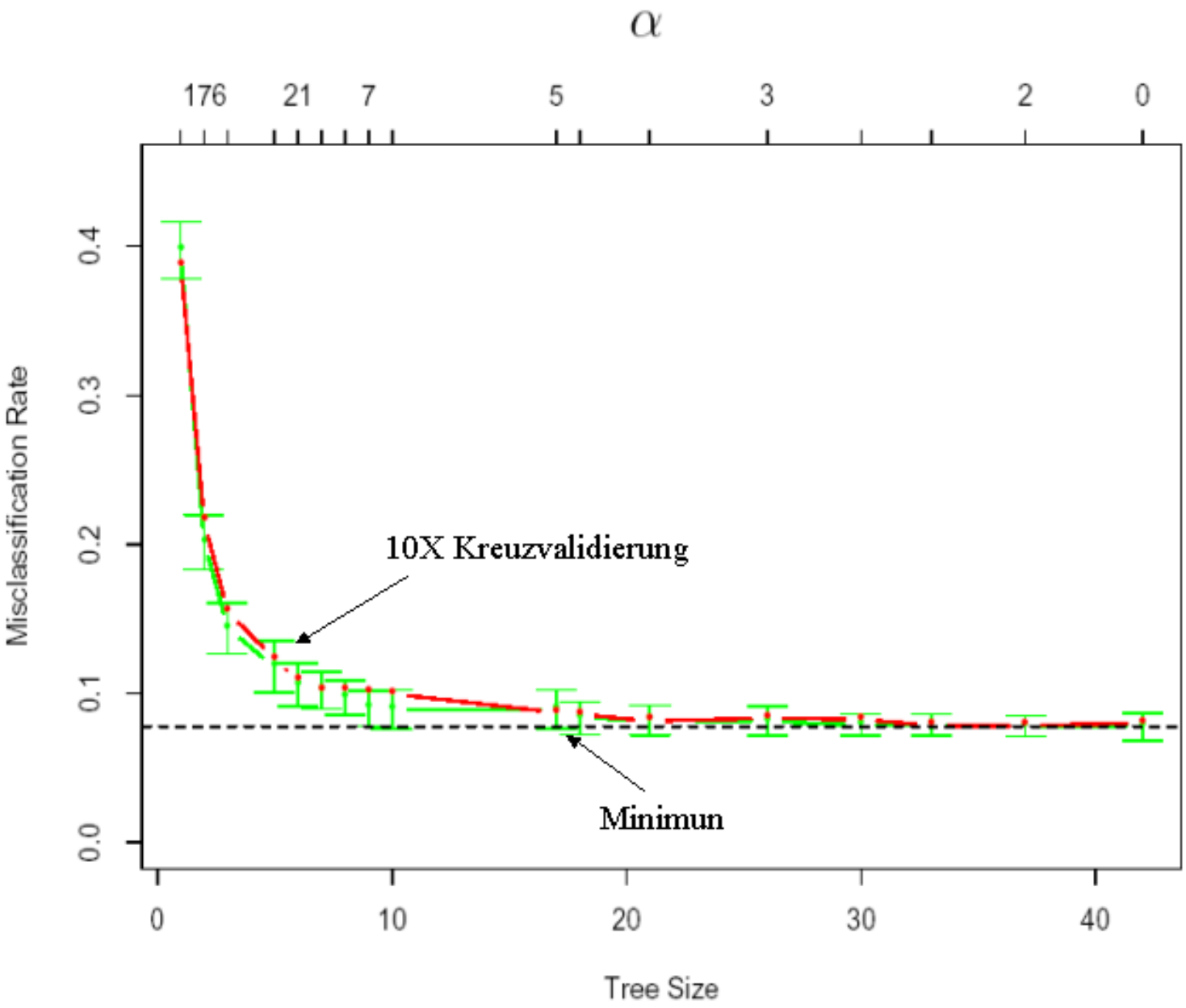

Figure 9.3: *Node impurity measures for two-class classification, as a function of the proportion $p$ in class 2. Cross-entropy has been scaled to pass through $(0.5, 0.5)$.*

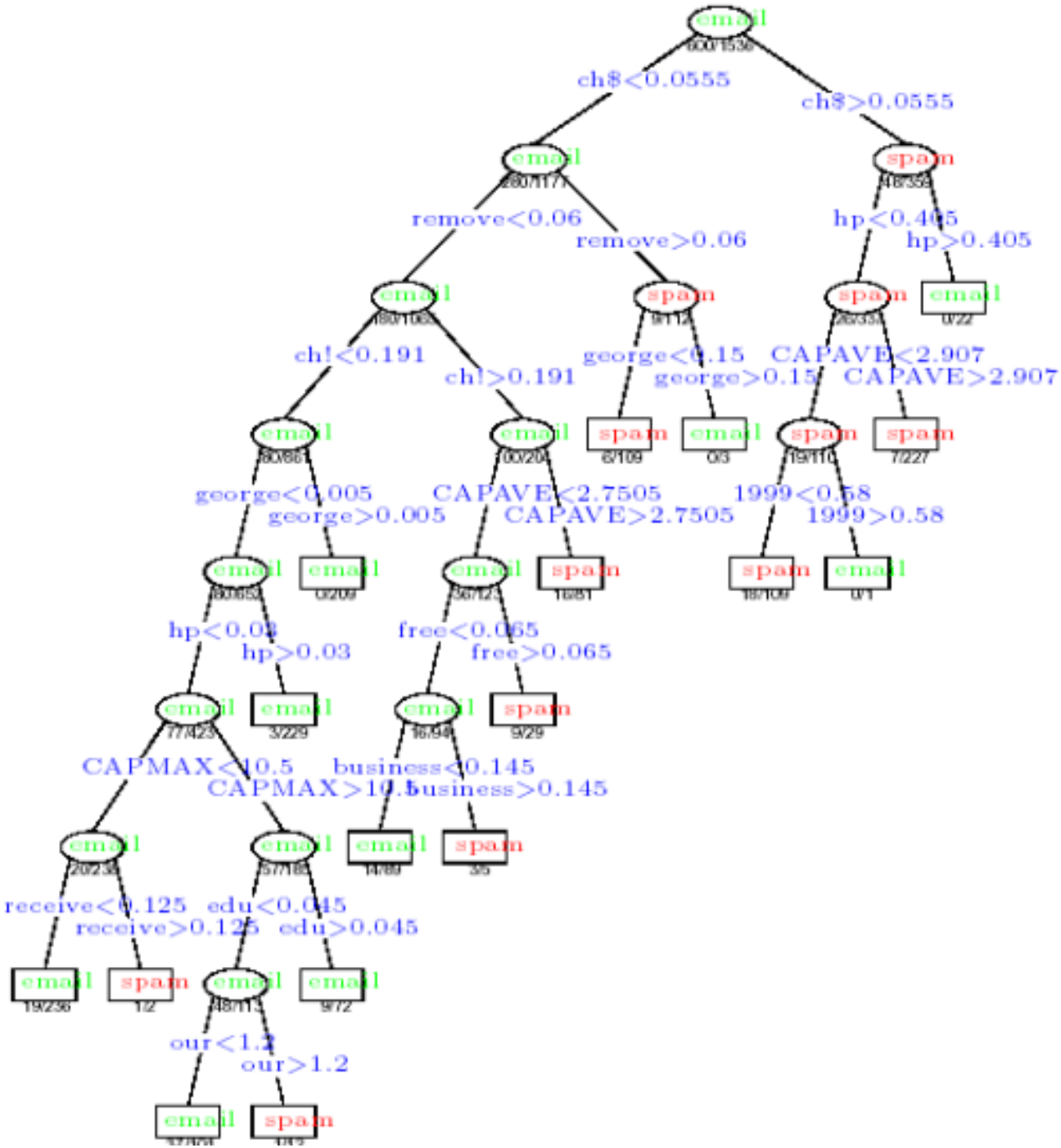
## CART: Eigenschaften

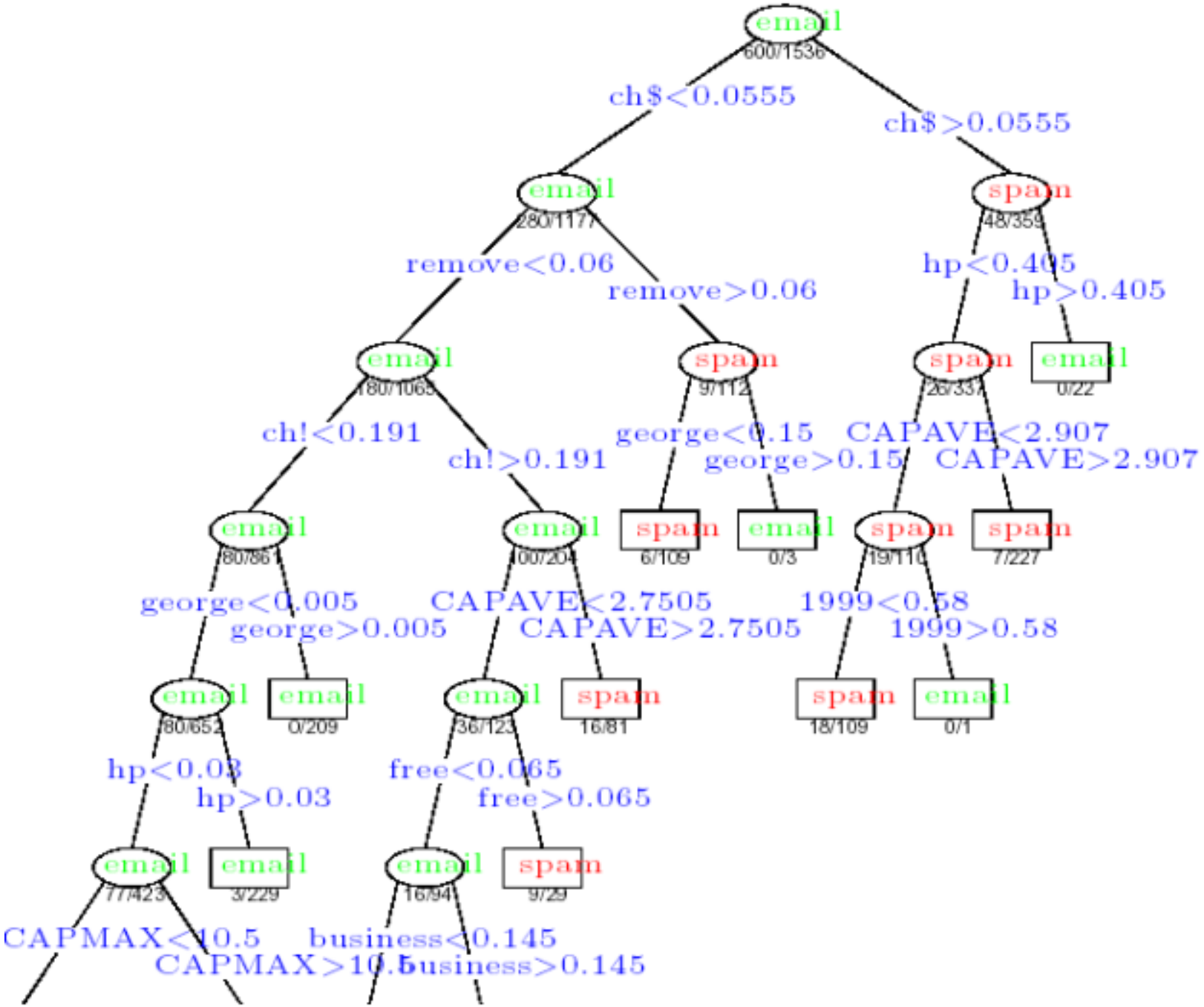
- Das Ergebnis ist ein leicht-verständlicher Entscheidungsbaum
- Allerdings: Baum ist häufig empfindlich gegenüber kleinen Änderungen in den Daten (instabil)
- Insbesondere für Regression sind die sprunghaften Übergänge unschön

## Beispiel: Spam

- 4601 Emails: Klassifiziere, ob Spam = Junk Email oder reguläre Email
- Merkmale: 57 häufigste Wörter und Punktuationen
- Beispiel: Wörter wie “free” oder auch “!” sind in Spams weitaus häufiger
- Mehr als 17 Blätter sind nicht notwendig
- Fehlerrate von 8.7 % (nicht sehr gut)
- Root split: mehrheitlich SPAM wenn “$” mehr als 5.5% mal im Text vorkommt; falls zusätzlich “hp” mit mehr als 40% mal vorkommt: Firmen-Email
- Sensitivity (P(pred=spam| spam) ): 86.3 %
- Specifity (P(pred=not spam| not spam) ): 93.4 %


α
176
21
7
5
3
2
0
Misclassification Rate
0.0
0.1
0.2
0.3
0.4
10X Kreuzvalidierung
Minimun
0
10
20
30
40
Tree Size


email
600/1536
ch$<0.0555
ch$>0.0555
email
280/1177
spam
48/359
remove<0.06
remove>0.06
hp<0.405
hp>0.405
email
180/1065
spam
9/112
spam
26/333
email
0/22
ch!<0.191
ch!>0.191
george<0.15
george>0.15
CAPAVE<2.907
CAPAVE>2.907
email
80/861
email
100/204
spam
6/109
email
0/3
spam
19/110
spam
7/227
george<0.005
george>0.005
CAPAVE<2.7505
CAPAVE>2.7505
1999<0.58
1999>0.58
email
80/652
email
0/209
email
36/123
spam
16/81
spam
18/109
email
0/1
hp<0.03
hp>0.03
free<0.065
free>0.065
email
77/423
email
3/229
email
16/94
spam
9/29
CAPMAX<10.5
CAPMAX>10.5
business<0.145
business>0.145
email
20/238
email
57/185
email
14/89
spam
3/5
receive<0.125
receive>0.125
edu<0.045
edu>0.045
email
19/236
spam
1/2
email
48/113
email
9/72
our<1.2
our>1.2
email
spam


email
600/1536
ch$<0.0555
ch$>0.0555
email
280/1177
spam
48/359
remove<0.06
remove>0.06
hp<0.405
hp>0.405
email
180/1065
spam
9/112
spam
26/337
email
0/22
ch!<0.191
ch!>0.191
george<0.15
george>0.15
CAPAVE<2.907
CAPAVE>2.907
email
80/861
email
100/204
spam
6/109
email
0/3
spam
19/110
spam
7/227
george<0.005
george>0.005
CAPAVE<2.7505
CAPAVE>2.7505
1999<0.58
1999>0.58
email
80/652
email
0/209
email
36/123
spam
16/81
spam
18/109
email
0/1
hp<0.03
hp>0.03
free<0.065
free>0.065
email
77/423
email
3/229
email
16/94
spam
9/29
CAPMAX<10.5
CAPMAX>10.5
business<0.145
business>0.145

# CART: Vorteile im Datamining

- Der Algorithmus ist schnell und das Ergebnis ist interpretierbar
- CART kann mit stetigen und diskreten Eingangsdaten umgehen und besitzt Techniken zum Umgang mit fehlenden Daten
- CART ist invariant zu monotonen Transformationen der Eingänge
- Daher ist CART unempfindlich gegenüber Ausreißern
- Die Merkmalsselektion ist automatisch Teil des Algorithmus
- Daher: unempfindlich gegenüber irrelevanten Eingängen (oft Mehrzahl im Datamining)

Table 1: *Some characteristics of different learning methods. Key:* ● *= good,* ● *=fair, and* ● *=poor.*

| Characteristic | Neural nets | SVM | Trees | MARS | k-NN, kernels |
|---|---|---|---|---|---|
| Natural handling of data of "mixed" type | ● | ● | ● | ● | ● |
| Handling of missing values | ● | ● | ● | ● | ● |
| Robustness to outliers in input space | ● | ● | ● | ● | ● |
| Insensitive to monotone transformations of inputs | ● | ● | ● | ● | ● |
| Computational scalability (large $N$) | ● | ● | ● | ● | ● |
| Ability to deal with irrelevant inputs | ● | ● | ● | ● | ● |
| Ability to extract linear combinations of features | ● | ● | ● | ● | ● |
| Interpretability | ● | ● | ● | ● | ● |
| Predictive power | ● | ● | ● | ● | ● |

## Mehr Bäume

- ID3, C4.5, C5.0 (Quinlan) (C5.0 sehr ähnlich zu CART); verfeinerte Extraktion von Regeln
- PRIM: Bump Hunting (finden von Regionen hoher Datendichte)
- MARS: Multivariate Adaptive Regression Splines (Vorteile für Regression)