Technische
Hochschule
Wildau [FH]
Technical University
of Applied Sciences

Maschinelles Lernen

Klassifikation(-sbäume)

Prof. Dr. Rainer Stollhoff

Supervised Learning

Supervised Learning

1. Aufgabe A

   Vorhersage $\hat{Y} = A(X)$

2. Qualität Q

   Verlustfunktion $\mathrm{L}(\hat{Y}, \mathrm{Y})$

3. Erfahrung E

Datensatz

(x_i, y_i) für $i = 1, \cdots, n$

Eine Maschine ***lernt*** aus Erfahrung E eine Aufgabe A mit der Qualität Q, wenn die Qualität Q beim erfüllen der Aufgabe A mit Erfahrung E steigt (T. Mitchell, MIT, 1988)

Supervised Classification

Supervised Learning

1. Aufgabe A

   Vorhersage $\hat{Y} = A(X) \in \{\mathbf{0}, \mathbf{1}, \cdots, \boldsymbol{n}\}$

2. Qualität Q

   Verlustfunktion $\mathrm{L}(\hat{Y}, \mathrm{Y})$

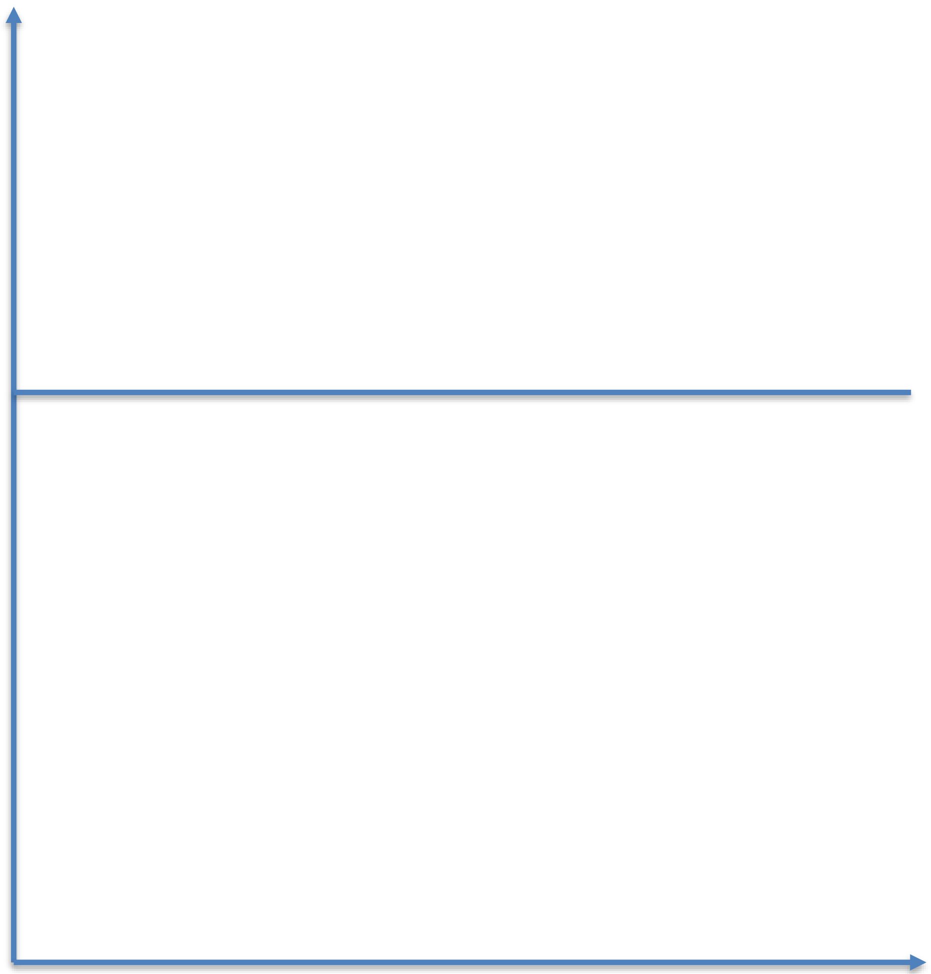
3. Erfahrung E

   Datensatz

   (x_i, y_i) für $i = 1, \cdots, n$ **mit** $\boldsymbol{y_i} \in \{\mathbf{0}, \mathbf{1}, \cdots, \boldsymbol{n}\}$

Eine Maschine ***lernt*** aus Erfahrung E eine Aufgabe A mit der Qualität Q, wenn die Qualität Q beim erfüllen der Aufgabe A mit Erfahrung E steigt (T. Mitchell, MIT, 1988)

Einfache univariate Klassifikation



- **A**ufgabe

  sage anhand von x den Zielwert y vorher

$$\hat{y} = f(x)$$

- **E**rfahrung

  Beobachtungen: (x_i, y_i)

- **Q**ualität

  sage den Zielwert <u>möglichst</u> <u>gut</u> vorher

  Gleichheit: $\hat{y} == y$

  Abstand: $(\hat{y} - y)^2$

Klassifikationsgüte

- Wahrheitsmatrix / Confusion-Matrix

	$y = 1$	$y = 0$	
$\hat{y} = 1$	Richtig-positiv (r_p)	Falsch-positiv (f_p)	Gesamtzahl Vorhersage Positiv r_p+f_p
$\hat{y} = 0$	Falsch-negativ (f_n)	Richtig-negativ (r_n)	Gesamtzahl Vorhersage Negativ r_n+f_n
	Gesamtzahl Echt Positiv r_p+f_n	Gesamtzahl Echt Negativ f_p+r_n	Gesamtzahl n

- Fehlklassifikationsrate / error rate

$$\mathrm{err} = \frac{1}{n}\sum_i \delta_{(y_i \neq \hat{y}_i)} = \#\{i: y_i \neq \hat{y}_i\} / \#\{i\}$$

- Korrektklassificationsrate / Accuracy

$$\mathrm{acc} = \frac{1}{n}\sum_i \delta_{(y_i = \hat{y}_i)} = \#\{i: y_i = \hat{y}_i\} / \#\{i\}$$

- **Sensitivität** / True-Positive-Rate / **Recall**:

$$sens = tpr = \frac{r_p}{r_p + f_n}$$

- **Spezifizität** / True-Negative-Rate:

$$spec = tnr = \frac{r_n}{r_n + f_p}$$

- Genauigkeit / **Precision** / Positive-Predictive-Value:

$$prec = ppv = \frac{r_p}{r_p + f_p}$$

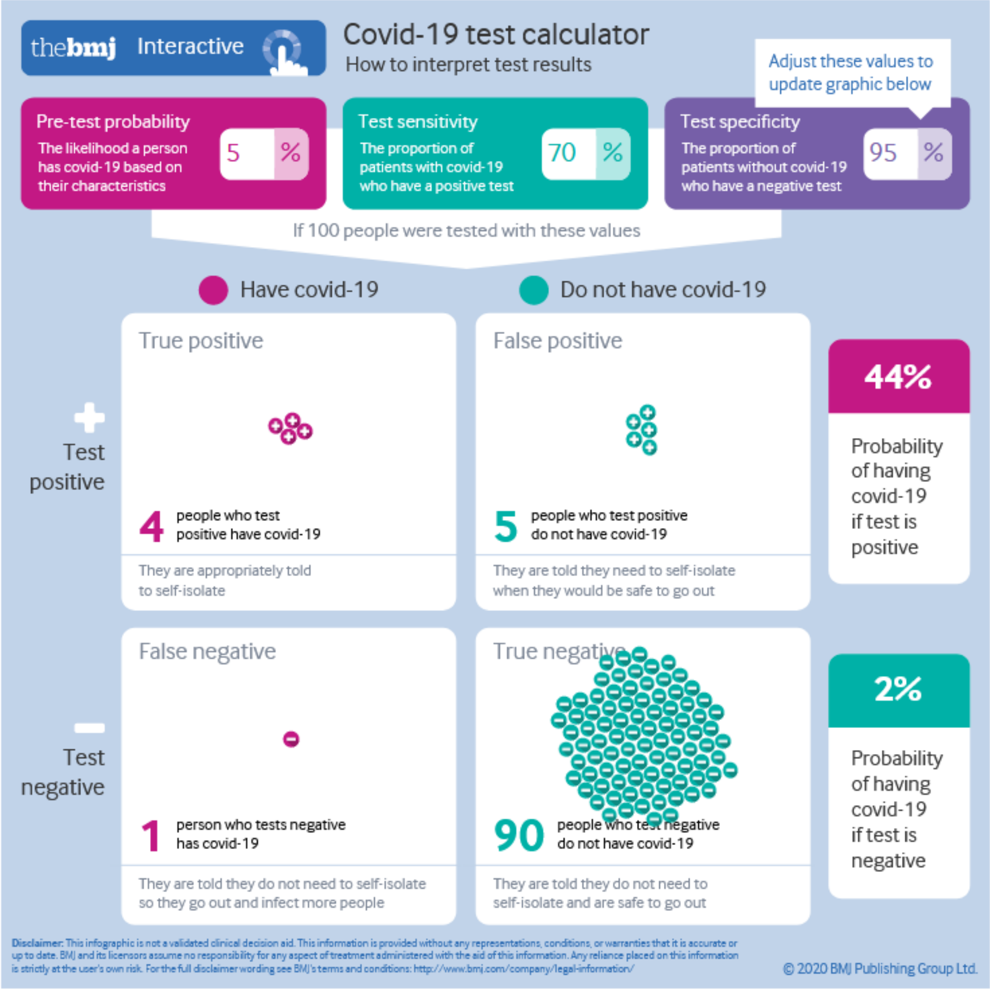
- Trennfähigkeit / Negative-Predictive-Value:

$$\mathrm{np}v = \frac{r_n}{r_n + f_n}$$

Klassifikationsgüte – Beispiel für Covid-19 tests

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

- Wahrheitsmatrix / Confusion-Matrix



https://www.bmj.com/content/369/bmj.m1808

– Fehlklassifikationsrate / error rate

$$\mathrm{err} = \frac{1}{n}\sum_i \delta_{(y_i \neq \hat{y}_i)} = \#\{i: y_i \neq \hat{y}_i\} \Big/ \#\{i\}$$

– Korrektklassifikationsrate / Accuracy

$$\mathrm{acc} = \frac{1}{n}\sum_i \delta_{(y_i = \hat{y}_i)} = \#\{i: y_i = \hat{y}_i\} \Big/ \#\{i\}$$

– **Sensitivität** / True-Positive-Rate / **Recall**:

$$sens = tpr = \frac{r_p}{r_p + f_n}$$

– **Spezifizität** / True-Negative-Rate:

$$spec = tnr = \frac{r_n}{r_n + f_p}$$

– Genauigkeit / **Precision** / Positive-Predictive-Value:

$$prec = ppv = \frac{r_p}{r_p + f_p}$$

– Trennfähigkeit / Negative-Predictive-Value:

$$\mathrm{npv} = \frac{r_n}{r_n + f_n}$$

Klassifikation

Aufgabe: Klassifikation, d.h. Vorhersage $\hat{y} = \hat{y}(x) \in \{0,1\}$

Erfahrung: Datensatz $(x_i, y_i)_{i=1}^{n}$

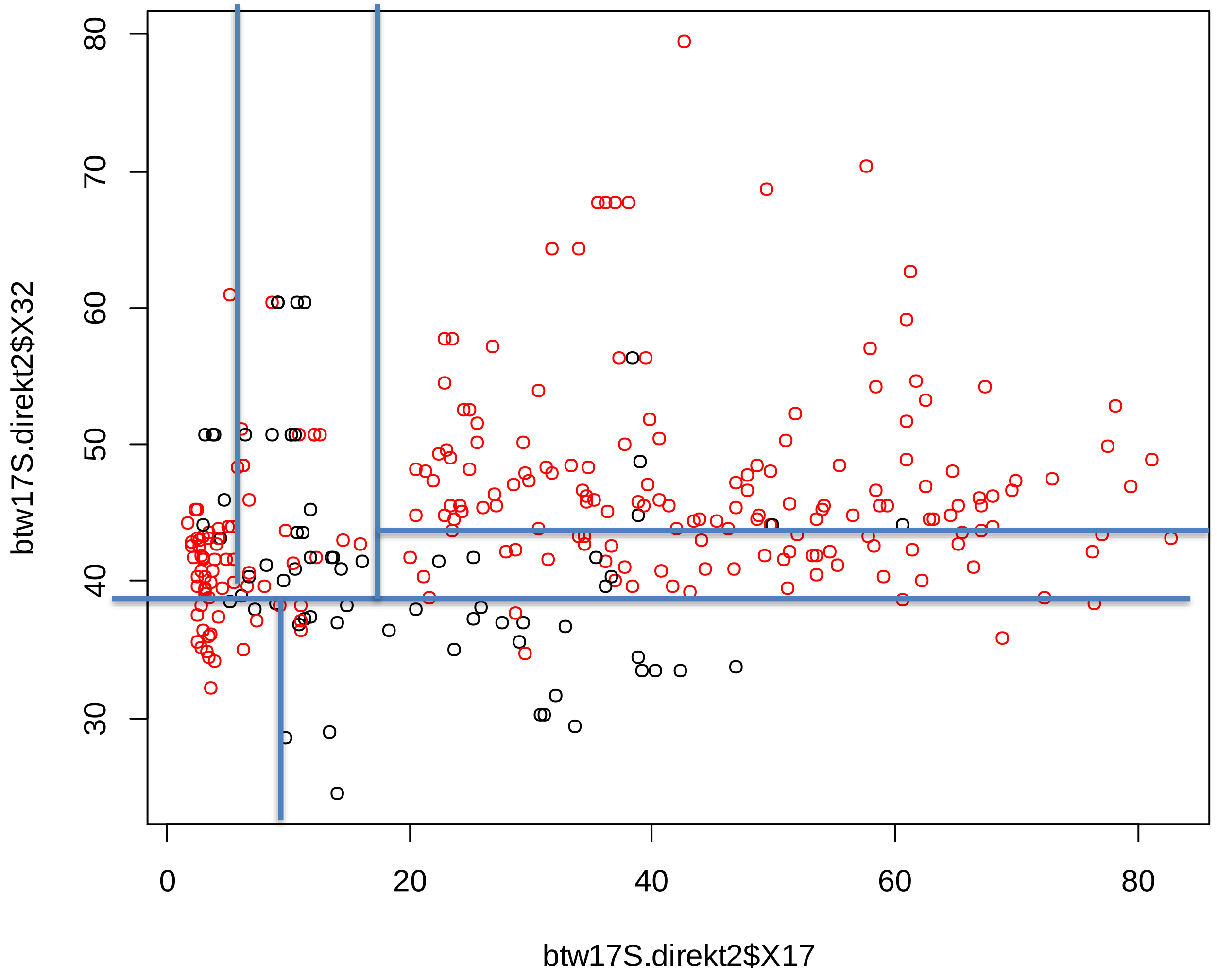
Qualität: Fehlerrate

$$\mathrm{err} = \frac{1}{n}\sum_{i} \delta_{(y_i \neq \hat{y}_i)} = {\#\{i: y_i \neq \hat{y}(x_i)\}}\big/{\#\{i\}}$$

Lernen: Finde einen Wert für θ , der die Fehlerrate minimiert

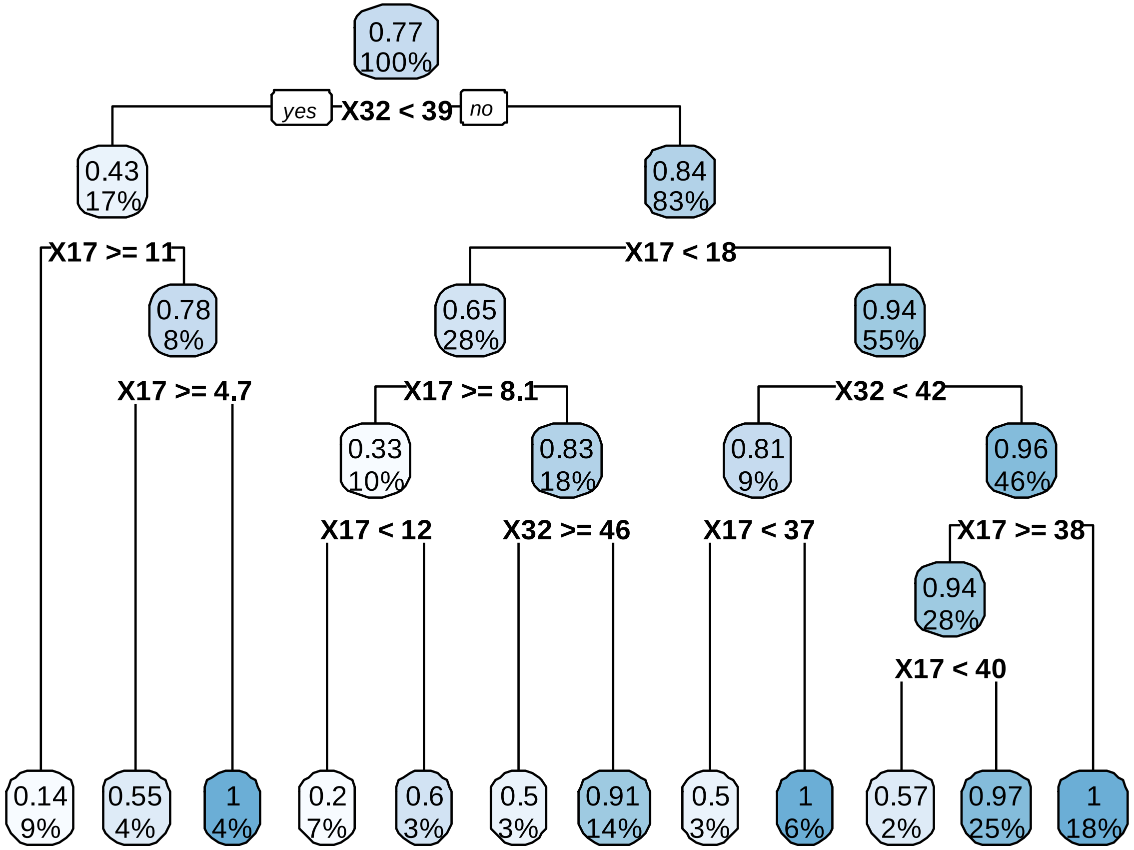
Klassifikationsbäume

- Idee: Rekursive Partitionierung / Wiederholtes Aufteilen



Klassifikationsbäume

- Idee: Rekursive Partitionierung / Wiederholtes Aufteilen
- Teilungsregel:
  In jeder neue Gruppe möglichst die gleiche Klasse, d.h. in jedem der beiden neuen Knoten K mit
  $$p_K = {}^{\#\{\text{i in K: } y_i = 1\}}\Big/_{\#\{i\ in\ K\}}$$
  minimiere
  $$p_K \cdot (1 - p_K)$$
- Vorhersage:
  Lasse eine neue Beobachtung den Baum durchlaufen. Angenommen sie landet im Knoten K. Die häufigste Klasse in K ist y_K, welche zur Vorhersage dient:
  $$\hat{y}(x_i) = y_{K(x_i)}$$



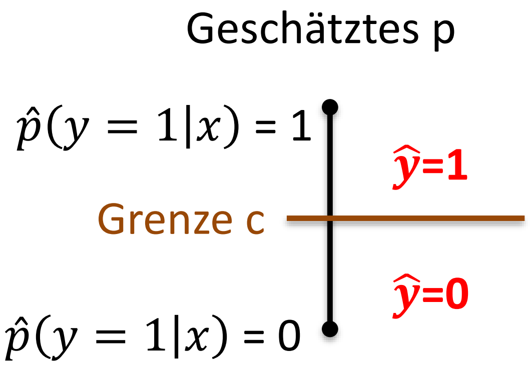
Klassifikationsregel mit Wahrscheinlichkeiten

- Vorhersage in der Regel durch Vergleich von Wahrscheinlichkeiten und Grenzwert

$$\hat{p}(y = 1|x) > c$$

oder scores und Grenzwert

$$f(x) > c$$

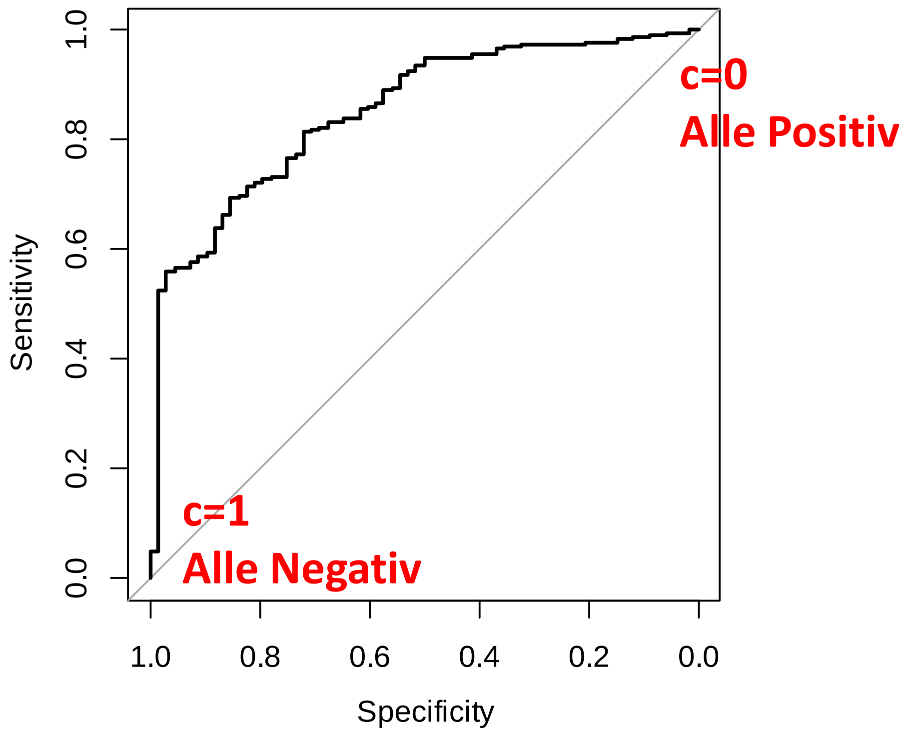


- Receiver Operating Characteristic Kurve ROC-Kurve

Trägt für alle möglichen Grenzen c
die Sensitivität gegen (1-Spezifizität) auf

- Area under the ROC-Curve – AUC bzw. `auc()`

Gibt die Fläche unterhalb der ROC-Kurve an



Vorhersage anhand von Wahrscheinlichkeiten

- Fehlklassifikationsrate

$$\text{err} = \frac{1}{n}\sum_i 1_{(y_i \neq \hat{y}_i)}$$

- Quadrat. Fehler der Wahrs.

$$L(p,\hat{p}) = \frac{1}{n}\sum_i (p_i - \hat{p}_i)^2$$

- Brier-Score

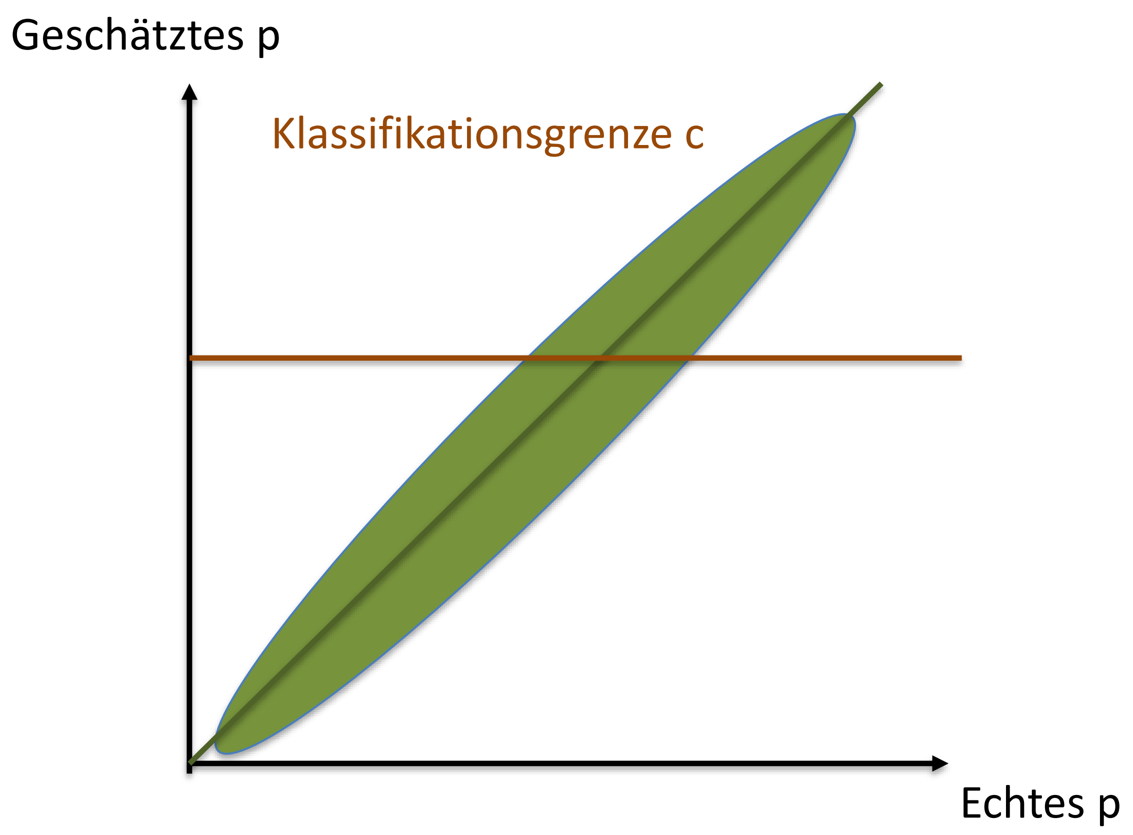
$$L(y,\hat{p}) = \frac{1}{n}\sum_i (y_i - \hat{p}_i)^2$$

- Likelihood

$$L(y,\hat{p}) = \prod_i y_i \cdot \hat{p}(x) + (1 - y_i) \cdot (1 - \hat{p}(x))$$

- Log-Likelihood

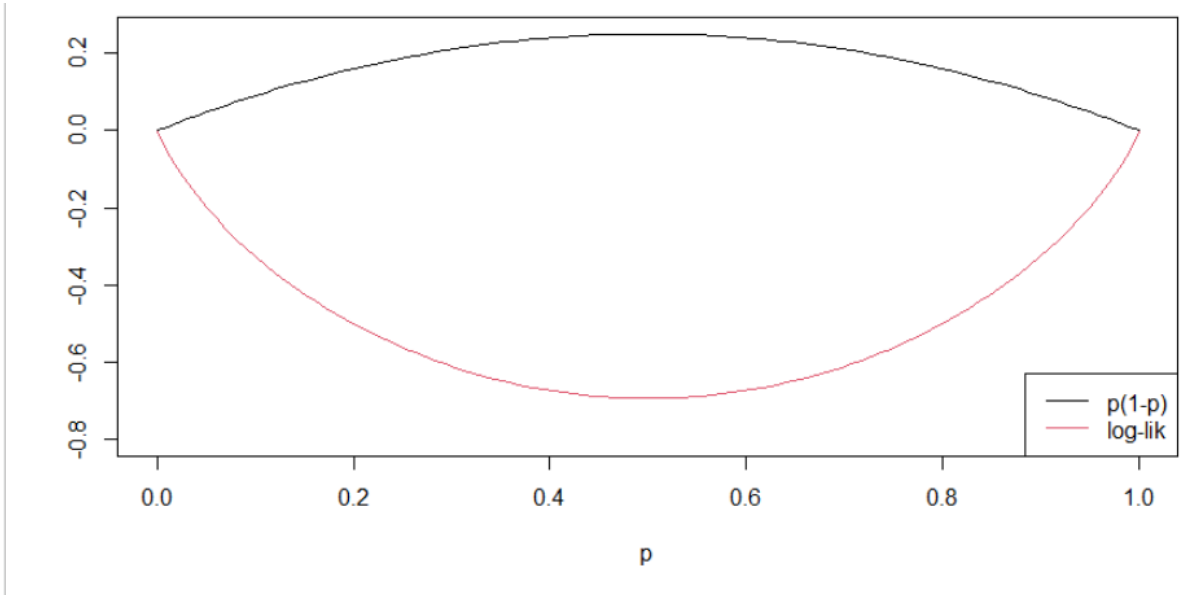
$$L(y,\hat{p}) = \sum_{i:y_i=1} \log \hat{p}(x_i) + \sum_{i:y_i=0} \log(1 - \hat{p}(x_i))$$

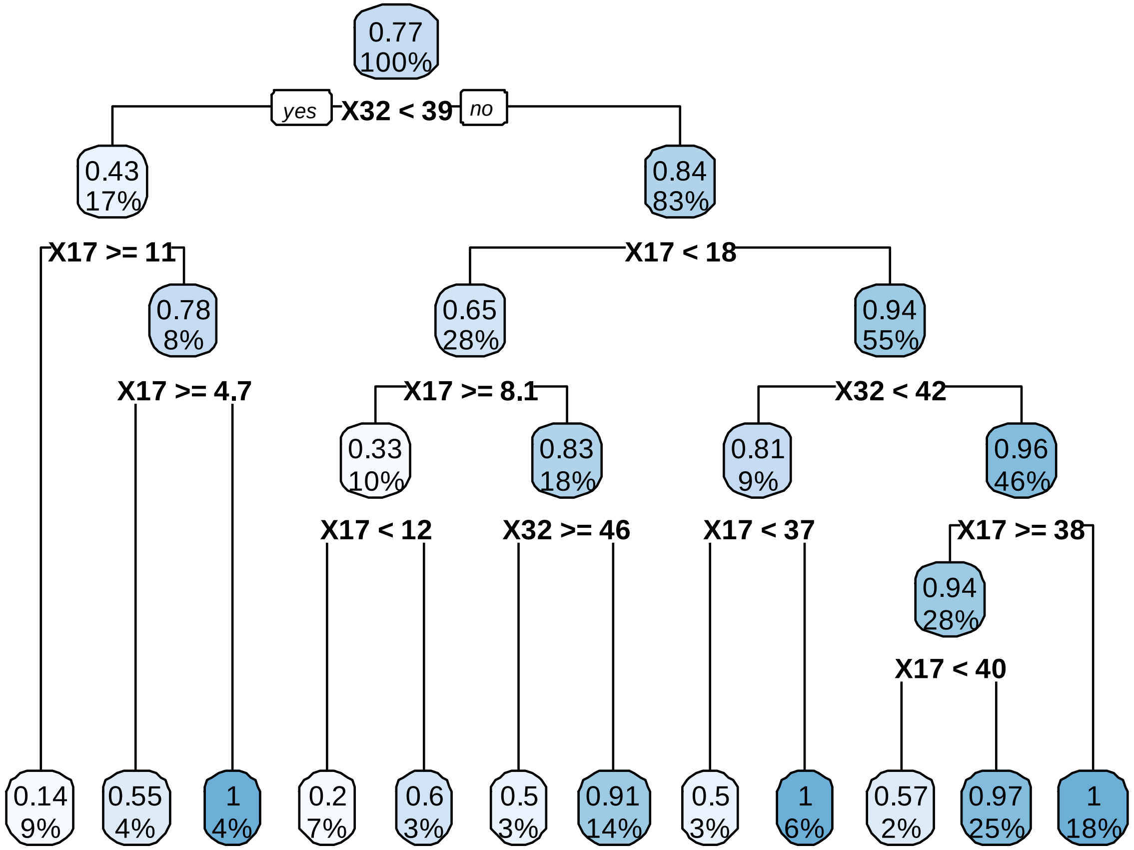


Klassifikationsbäume

- Idee: Rekursive Partitionierung / Wiederholtes Aufteilen
- Teilungsregel:
  In jedem neuen Knoten möglichst die gleiche Klasse, d.h. mit $p_K = {}^{\#\{\text{i in K}: y_i=1\}}/_{\#\{i \text{ in } K\}}$
  – Minimiere Brier score $\quad p_K \cdot (1 - p_K)$
  – Maximiere log-lik

$$\sum_{i \text{ in } K: y_i=1} \log p_K + \sum_{i \text{ in } K: y_i=0} \log(1 - p_K)$$





Ensemble Methoden

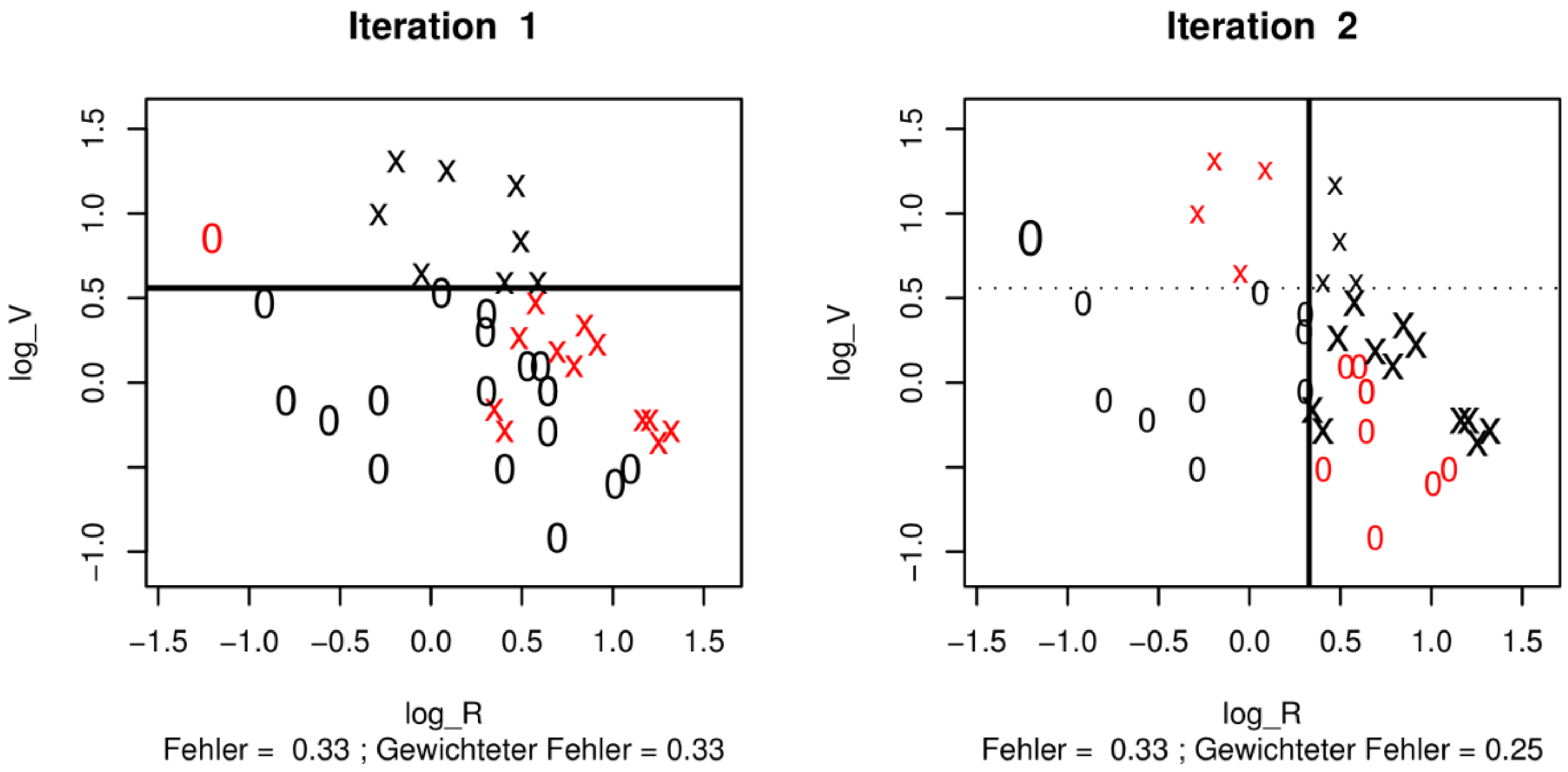
- Idee: „Swarm Intelligence“ oder „Wisdom of the Crowd“
  - Erzeuge wiederholt einfache Modelle
  - Aggregiere die einzelnen Vorhersagen z.B. mit Mittelwertbildung oder Mehrheitsentscheid
- Bagging = Bootstrap-Aggregating
  - Erzeuge Modelle auf Bootstrap Stichproben
- RandomForest
  - Erzeuge Bäume mit zufälligen Parametern (hier: Wahl der Partitionierung in einem Knoten) auf Bootstrap Stichproben
- Boosting
  - Erzeuge Modelle auf iterativ gewichteten Datensätzen
  - Erhöhe die Gewichte von Beobachtungen, die falsch vorhergesagt wurden

(Adaptive)Boosting

AdaBoost.M1

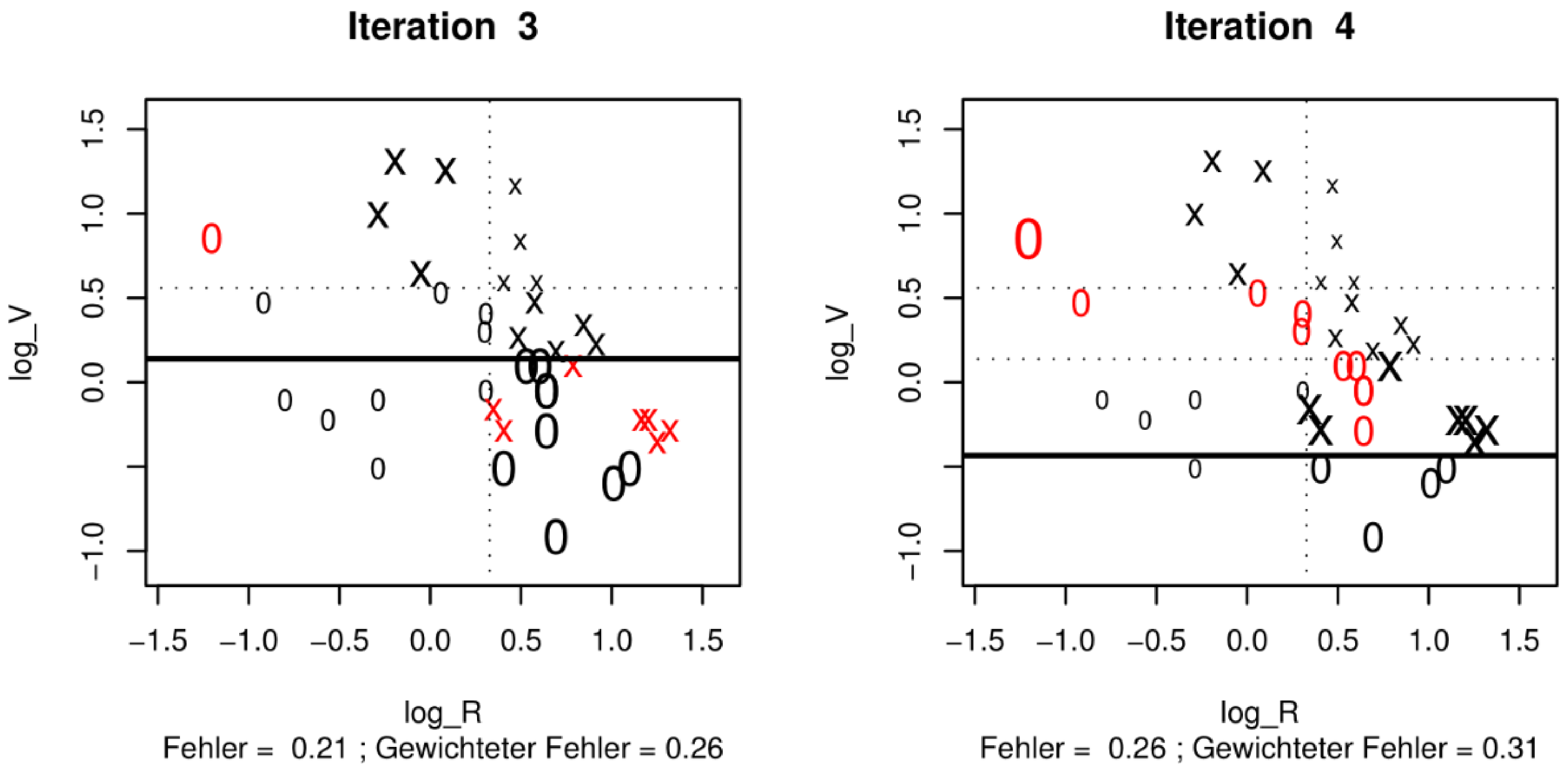
1. Gegeben einen Trainingsdatensatz $T^n = ((x_i, y_i))_{i=1,\dots,n}$
2. Initialisiere die Gewichtung $w_m(i) = \frac{1}{N}$, $(i = 1, \dots, n)$
3. Iteriere für $m = 1, \dots, M$:
   (a) Konstruiere mit w_m gewichteten Datensatz T^n_m.
   (b) Erhalte eine Klassifikationsregel $\hat{y}_m = \hat{Y}(T^n_m)$
   (c) mit gewichtetem Fehler $err^w_m = \sum_{i:\hat{y}(x_i) \neq (y_i)} w_m(i)$.
   (d) Setze $\beta_m = \frac{err^w_m}{1-err^w_m}$
   (e) und $w_{m+1}(i) = \begin{cases} w_m(i) & \text{, falls } y_i \neq \hat{y}_m(x_i) \\ w_m(i)\ \beta_m & \text{, sonst} \end{cases} \qquad (i = 1, \dots, n).$
   (f) Normalisiere die Gewichtung, so dass $\sum_{i=1}^n w_{m+1}(i) = 1$.
4. Scoring rule $\hat{f} := \sum_{m=1}^M \log(\frac{1}{\beta_m})\ \hat{y}_m$
5. Klassifikationsregel $\hat{y} := sign(\hat{f})$
6. geschätzte Faktorisierung $\hat{p} := \frac{e^{\hat{f}}}{e^{\hat{f}} + e^{-\hat{f}}}$

Beispiel AdaBoost



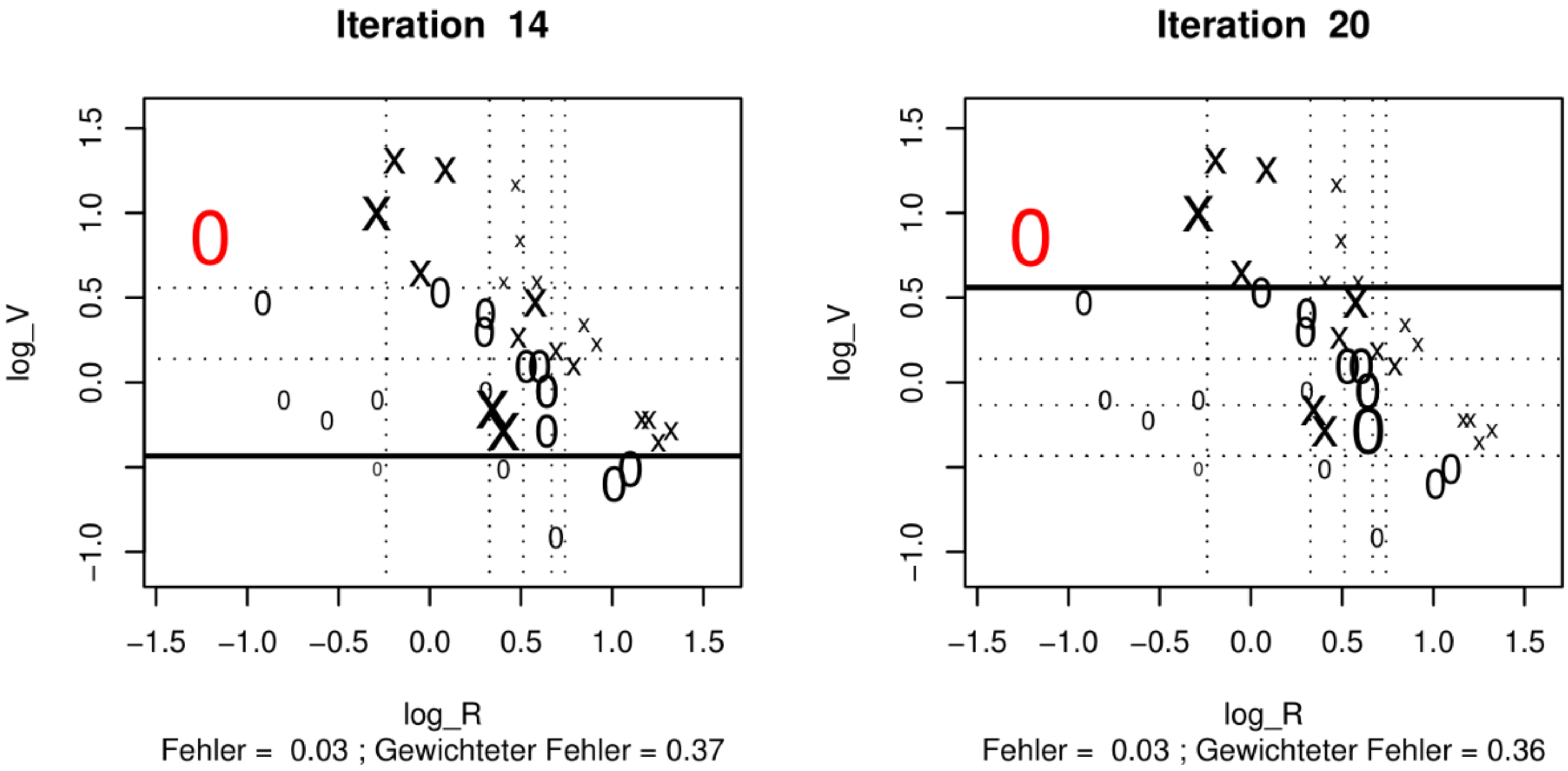
(Stollhoff, 2005, Verbesserung von Klassifikationsverfahren durch Boosting)

Beispiel AdaBoost



(Stollhoff, 2005, Verbesserung von Klassifikationsverfahren durch Boosting)

Beispiel AdaBoost



(Stollhoff, 2005, Verbesserung von Klassifikationsverfahren durch Boosting)

Gradient-Boosting

Logit-Boost

1. Gegeben einen Trainingsdatensatz $T^n = ((x_i, y_i))_{i=1,\dots,n}$

2. Initialisiere die scoring rule $\hat{f} \equiv 0$, die geschätzten bedingten Wahrscheinlichkeiten $\hat{p}_1(x_i) = \frac{1}{2}$ und die Gewichtung $w_m(i) = \frac{1}{N}$, $(i = 1, \dots, n)$

3. Iteriere für $m = 1, \dots, M$:

   (a) $z_m(i) = \frac{y_i - \hat{p}_m(x_i)}{\hat{p}_m(x_i)(1-\hat{p}_m(x_i))}$

   (b) $w_m(i) = \hat{p}_m(x_i)(1 - \hat{p}_m(x_i))$

   (c) Bezeichne $T_m^n(z)$ den mit w_m gewichteten Trainingsdatensatz $T_m^n(z) = ((x_i, z_m(i)))_{i=1,\dots,n}$

   (d) Wende das Regressionsverfahren $\hat{F}$ unter Minimierung des quadratischen Verlustes auf $T_m^n(z)$ an und erhalte die Funktion $\hat{r}_m$

   (e) Setze $\hat{f}_m = \sum_{k=1}^{m} \hat{r}_k$

   (f) und $\hat{p}_m(x) = \frac{e^{\hat{f}_m(x)}}{1+e^{\hat{f}_m(x)}}$

4. Scoring Rule $\hat{f} = \sum_{k=1}^{M} \hat{r}_k$

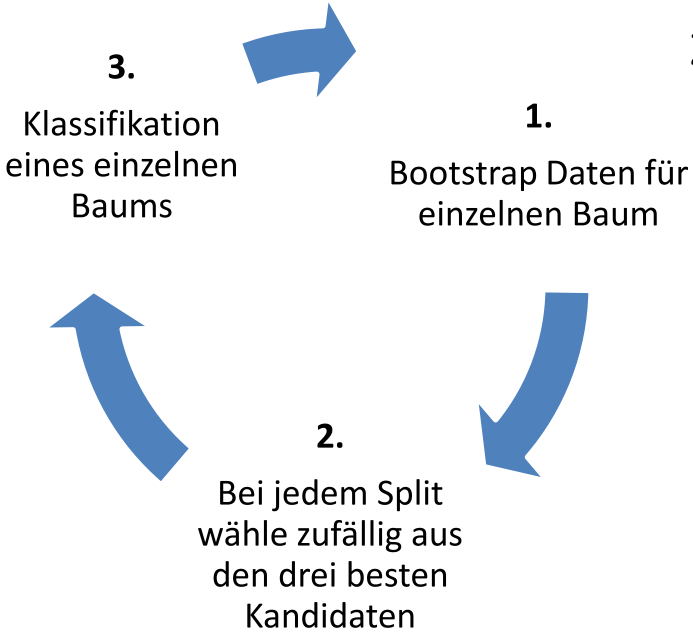
5. Klassifikationsregel $\hat{y}(x) = \begin{cases} 1 & \text{, falls } \hat{f}(x) \geq 0 \\ 0 & \text{, sonst} \end{cases}$

6. geschätzte Faktorisierung $\hat{p}(x) = \frac{e^{\hat{f}(x)}}{1+e^{\hat{f}(x)}}$

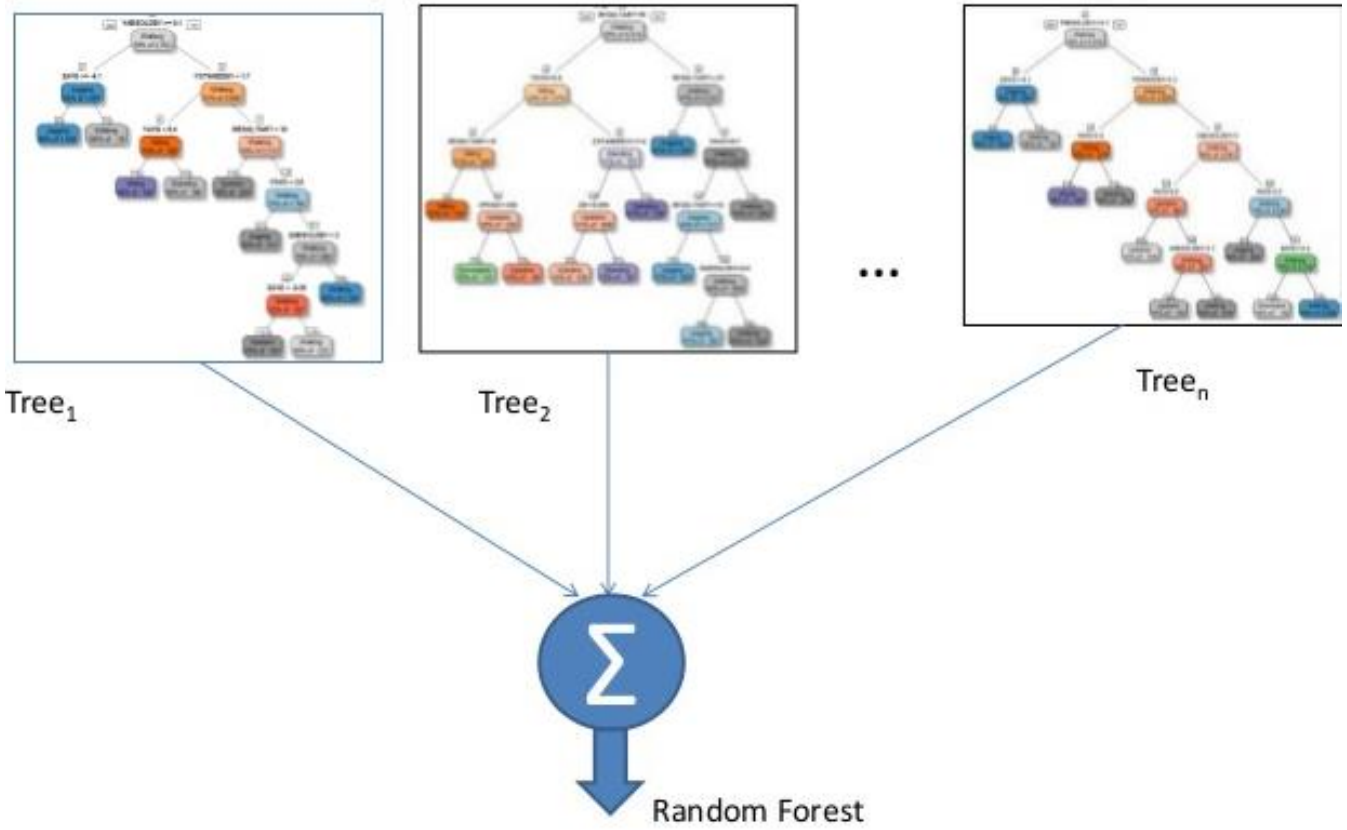
Random Forest

Technische Hochschule Wildau
Technical University of Applied Sciences

1) Erzeugen mittels wiederholter
Zufallsstichproben und teilweise zufälliger Parameterwahl



2) Vorhersage mittels Aggregation / Mehrheitsentscheid



Exkurs: Klassifikation – Maximum-Likelihood-Schätzer mit Gradientenabstieg

Technische Hochschule Wildau
Technical University of Applied Sciences

Aufgabe: Klassifikation, d.h. Vorhersage $\hat{y} = \hat{y}(x) \in \{0,1\}$
mittels Vorhersage der bedingten Wahrscheinlichkeit $\mathrm{f}(x;\theta) = \hat{p}(y=1|x;\theta)$ und Grenzwert

Erfahrung: Datensatz $(x_i, y_i)_{i=1}^n$

Qualität: (Likelihood) oder log-likelihood

$$L(\theta) = \sum_{i:y_i=1} \log \hat{p}(x_i;\theta) + \sum_{i:y_i=0} \log(1-\hat{p}(x_i;\theta))$$

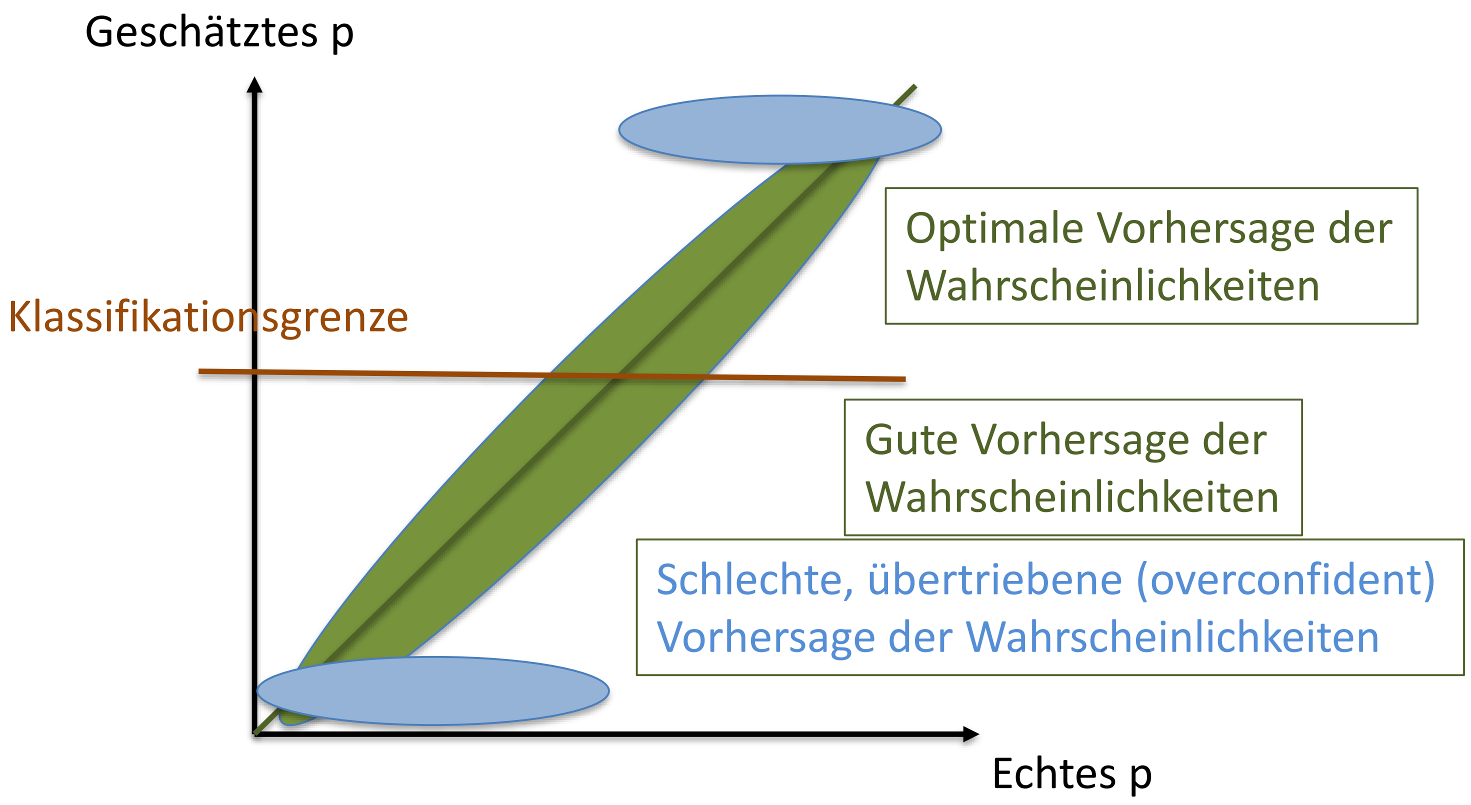
Lernen: Finde einen Wert für θ , der die log-likelihood minimiert (bzw. likelihood maximiert)
Durch geeignete Wahl von θ in einem iterativen Prozess (Gradientenabstiegsverfahren):

1. Wähle Startwert z.B. $\theta^0 = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$

2. Berechne Gradienten $\nabla L(\theta^0) = \begin{pmatrix} \frac{\partial}{\partial\theta_0} L(\theta^0) \\ \frac{\partial}{\partial\theta_1} L(\theta^0) \\ \vdots \\ \frac{\partial}{\partial\theta_n} L(\theta^0) \end{pmatrix} = \begin{pmatrix} \frac{\boldsymbol{d}}{\boldsymbol{df}} \boldsymbol{L}(\boldsymbol{f}(\boldsymbol{\theta^0})) \cdot \frac{\boldsymbol{\partial}}{\boldsymbol{\partial\theta_0}} \boldsymbol{f}(\boldsymbol{x};\boldsymbol{\theta^0}) \\ \frac{\boldsymbol{d}}{\boldsymbol{df}} \boldsymbol{L}(\boldsymbol{f}(\boldsymbol{\theta^0})) \cdot \frac{\boldsymbol{\partial}}{\boldsymbol{\partial\theta_1}} \boldsymbol{f}(\boldsymbol{x};\boldsymbol{\theta^0})) \\ \vdots \\ \frac{\boldsymbol{d}}{\boldsymbol{df}} \boldsymbol{L}(\boldsymbol{f}(\boldsymbol{\theta^0})) \cdot \frac{\boldsymbol{\partial}}{\boldsymbol{\partial\theta_n}} \boldsymbol{f}(\boldsymbol{x};\boldsymbol{\theta^0})) \end{pmatrix}$

3. Update $\theta^{t+1} = \theta^t + \boldsymbol{A} \cdot \nabla L(\theta^t)$

Klassifikationsgüte und (Klassen-)wahrscheinlichkeiten



Klassifikationsgüte und (Klassen-)wahrscheinlichkeiten