Actuarial Machine Learning - Präsentation

GAMs und Baumbasierte Regressionsmodelle

S. Brand, T. Cvitan, A. Gaubrich und B. Höfling

Universität zu Köln

Wintersemester 2020/2021

21. Januar 2021

Inhalt

Übersicht

Allgemeines

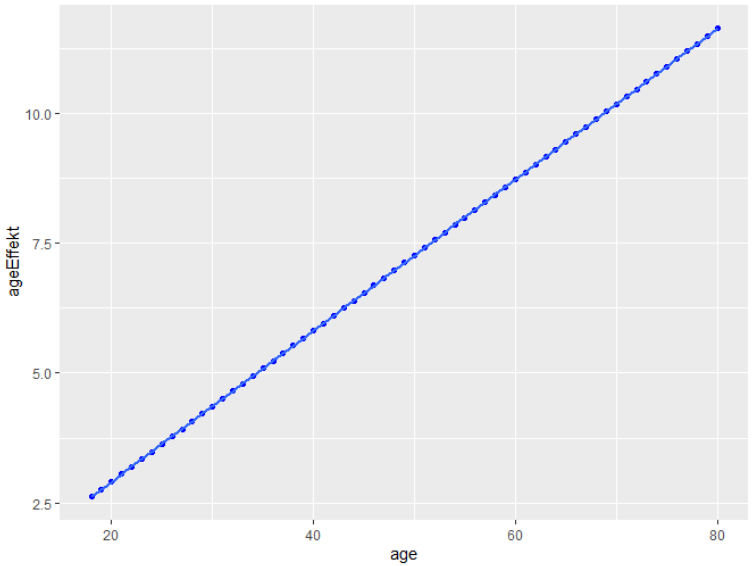
- Generalized Additive Models (GAMs) beschreiben eine Klasse von Regressions- und Klassifikationsmodellen.
- Sie fallen unter die Rubrik des überwachten Lernens (supervised learning).
- Sie sind eine Erweiterung der Generalized Linear Models (GLMs).

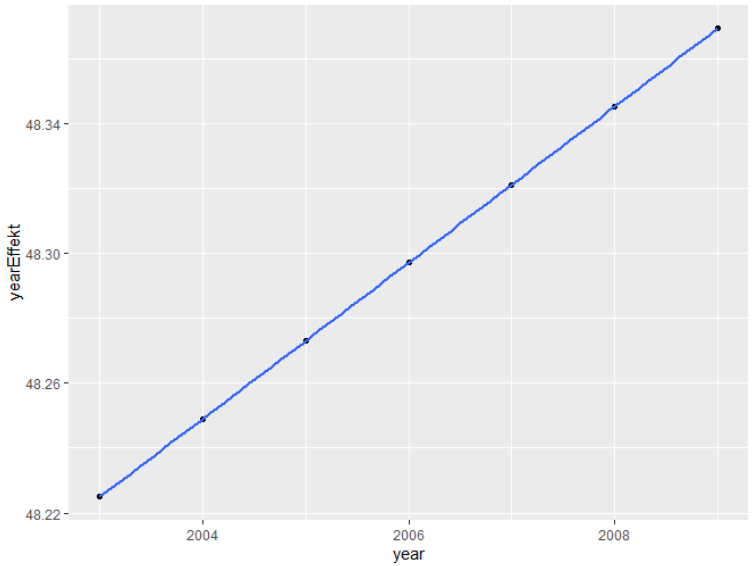
Herleitung der GAMs

GLMs beschreiben den Zusammenhang der Features zum Label durch die Summe der linearen Effekte der einzelnen Features auf den Output:

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \epsilon,$$

wobei Y die Prediction des Labels und $X_i, i \in 1, \ldots, p$ die Features sind.α ist der Y-Achsenabschnitt, ϵ ist ein Fehlerterm mit Erwartungswert 0, und β_i beschreibt den jeweiligen Effekt des Merkmals i.





Herleitung der GAMs

GAMs erweitern die GLMs

- erlaubt allgemeinere Effekte der Features,
- gleicher additiver Aufbau

Für ein klassischen Regressionsproblem wird das Modell beschrieben durch

$$Y = \alpha + f_1(X_1) + f_2(X_2) + \ldots + f_p(X_p) + \epsilon,$$

mit α und ϵ wie oben. Die f_i sind nicht-parametrische Funktionen.

$\Rightarrow$ Modell ist flexibler.

Link Funktionen

Allgemein wird ein GAM durch eine additive Link-Funktion g beschrieben, die in Beziehung zum bedingten Erwartungswert $\mu(X)$ von Y gesetzt wird:

$$g(\mu(X)) = \alpha + \sum_{i=1}^{p} f_i(X).$$

$\mu(X)$ ist der durch X bedingte Erwartungswert von Y, g ist die Link-Funktion. Beispiele von Link-Funktionen:

- $g(\mu) = logit(\mu) = log(\frac{\mu}{1-\mu})$
- $g(\mu) = \mu$
- $g(\mu) = log(\mu)$

Nicht-parametrische Funktionen

Wie bestimmt man eine nicht-parametrische Funktion?

Finde Funktion, welche den penalized RSS

$$\sum_{i=1}^{n}(y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}))^2 + \sum_{j=1}^{p} \lambda_j \int (f_j^{''}(t_j))^2 \, dt_j$$

minimiert. λ_j sind hierbei Tuningparameter.

Lösung ist ein kubischer Smoothing Spline:

- stückweise definiertes kubisches Polynom
- hat an jedem Wert x_i des Merkmals x einen Knoten
- stetig in den ersten beiden Ableitungen

Fittingmethode für Additive Modelle

Fitten eines GAMs durch fitten der Funktionen f_j.

Nicht-parametrische Funktionen lassen sich im Gegensatz zu parametrischen Funktionen nicht durch least-squared Methode fitten.

Lösung: Quelle (1)

Eine iterative Möglichkeit ein GAM zu fitten beschreibt der Backfittingalgorithmus für Additive Modelle.

Idee:

- Fitte zyklisch nacheinander jede Funktion f_j auf das Residuum der restlichen Funktionen $f_k, k \neq j$
- Wiederholen, bis sich die Funktion f_j um weniger als eine vorbestimmte Grenze verändert

Backfitting für Additive Modelle

Annahme: Durchschnitt der Funktionen f_j auf dem Datensatz ist 0: $\frac{1}{N}\sum_{i=1}^{N} f_j(x_{ij}) = 0, \forall j$.

(i) Initialisiere: $\hat{\alpha} = \frac{1}{N}\sum_{i=1}^{N} y_i, \hat{f}_j \equiv 0 \forall j$.

(ii) Iteriere zyklisch: $j = 1, 2, \ldots, p, 1, 2, \ldots, p, \ldots,$

$$\hat{f}_j \leftarrow S_j\left[\{y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik})\}_1^N\right], \tag{1}$$

$$\hat{f}_j \leftarrow \hat{f}_j - \frac{1}{N}\sum_{1}^{N} \hat{f}_j(x_{ij}), \tag{2}$$

bis sich $\hat{f}_j$ um weniger als eine vorbestimmte Grenze verändern.

S_j ist ein kubischer smoothing Spline, der auf die jeweiligen Funktion $\hat{f}_j$ angewendet wird.

Bewertung und Ausblick

$$g(\mu(X)) = \alpha + \sum_{i=1}^{p} f_i(X)$$

Vorteile:

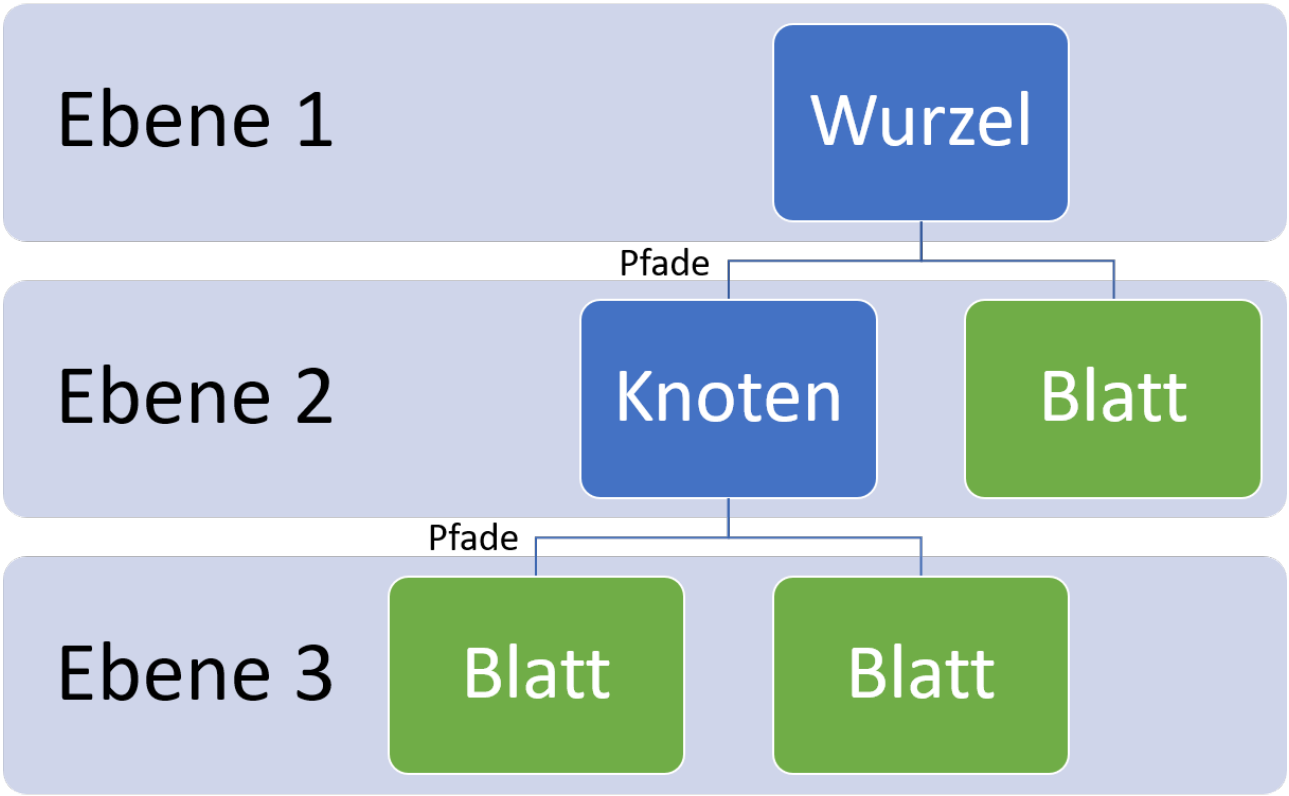
+ Additivität lässt viel Raum für Interpretation
+ Nichtlineare Effekte sorgen für bessere Ergebnisse als bei GLMs

Nachteile:

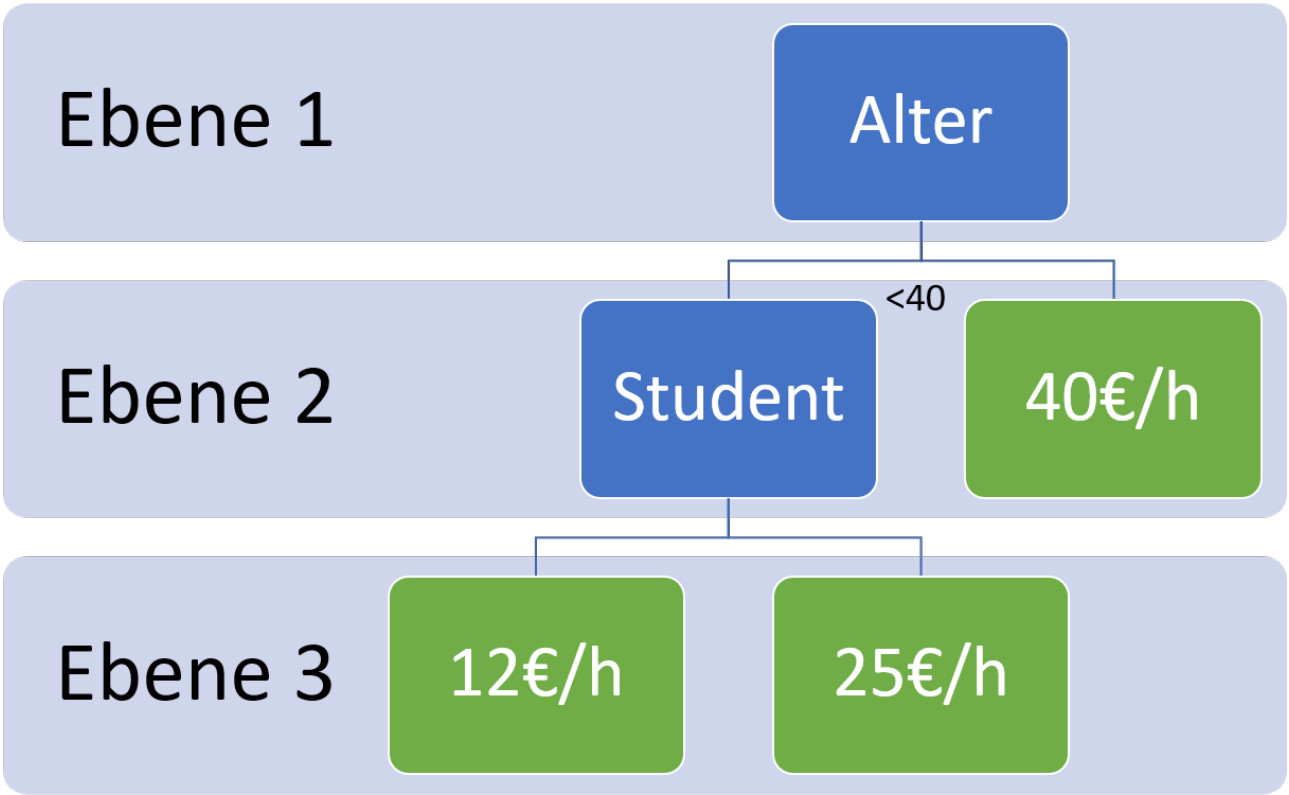
- Wichtige Effekte der Interaktion verschiedener Features unbeleuchtet (mögl. Interaktionsterm $f_{ij}(X_i X_j)$)
- Es gibt bessere Modelle

Übersicht

Allgemeiner Aufbau



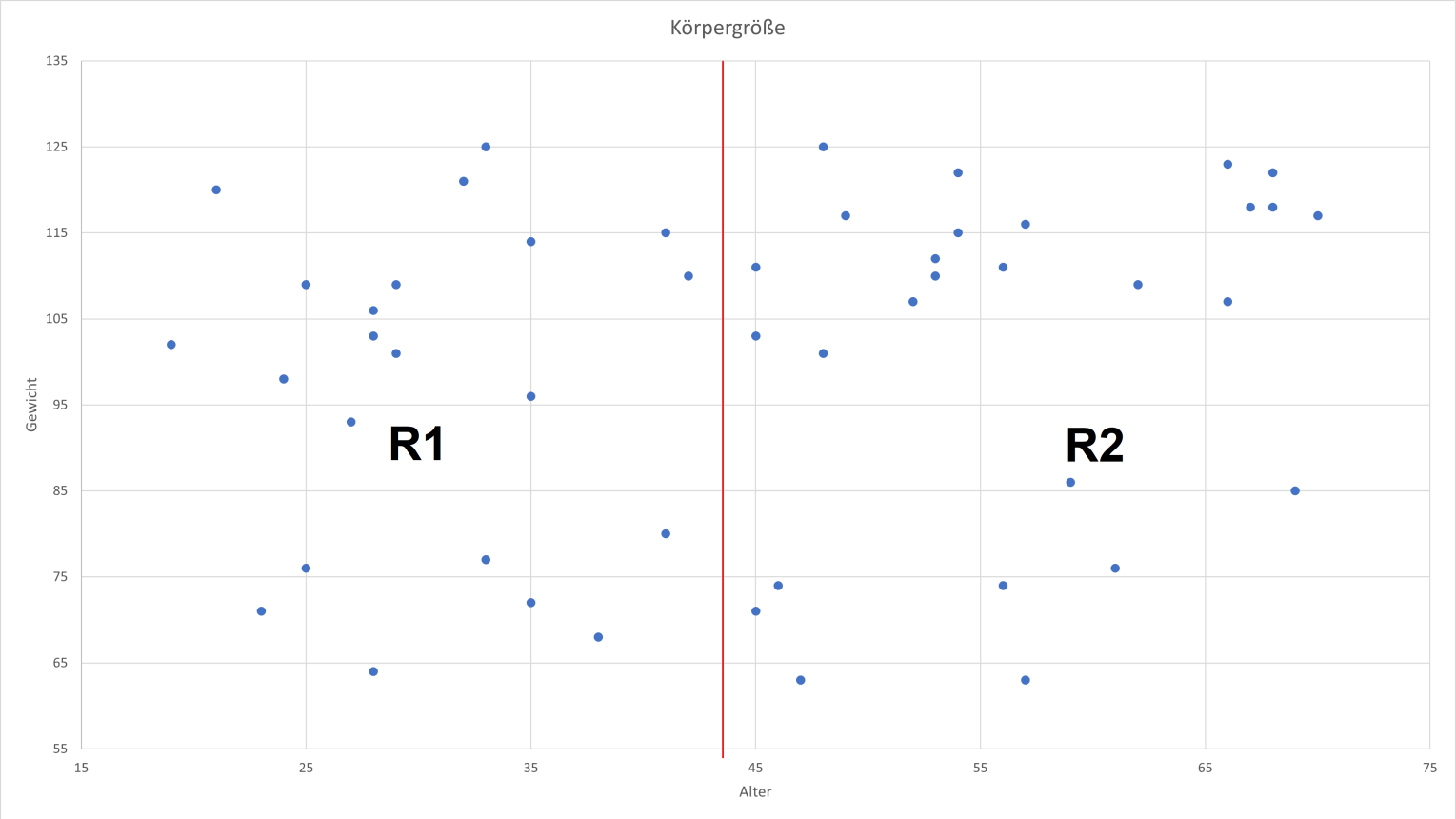
Beispielhafter Aufbau



$$f(X) = \sum_{m=1}^{M} c_m * 1_{x \in R_m}$$

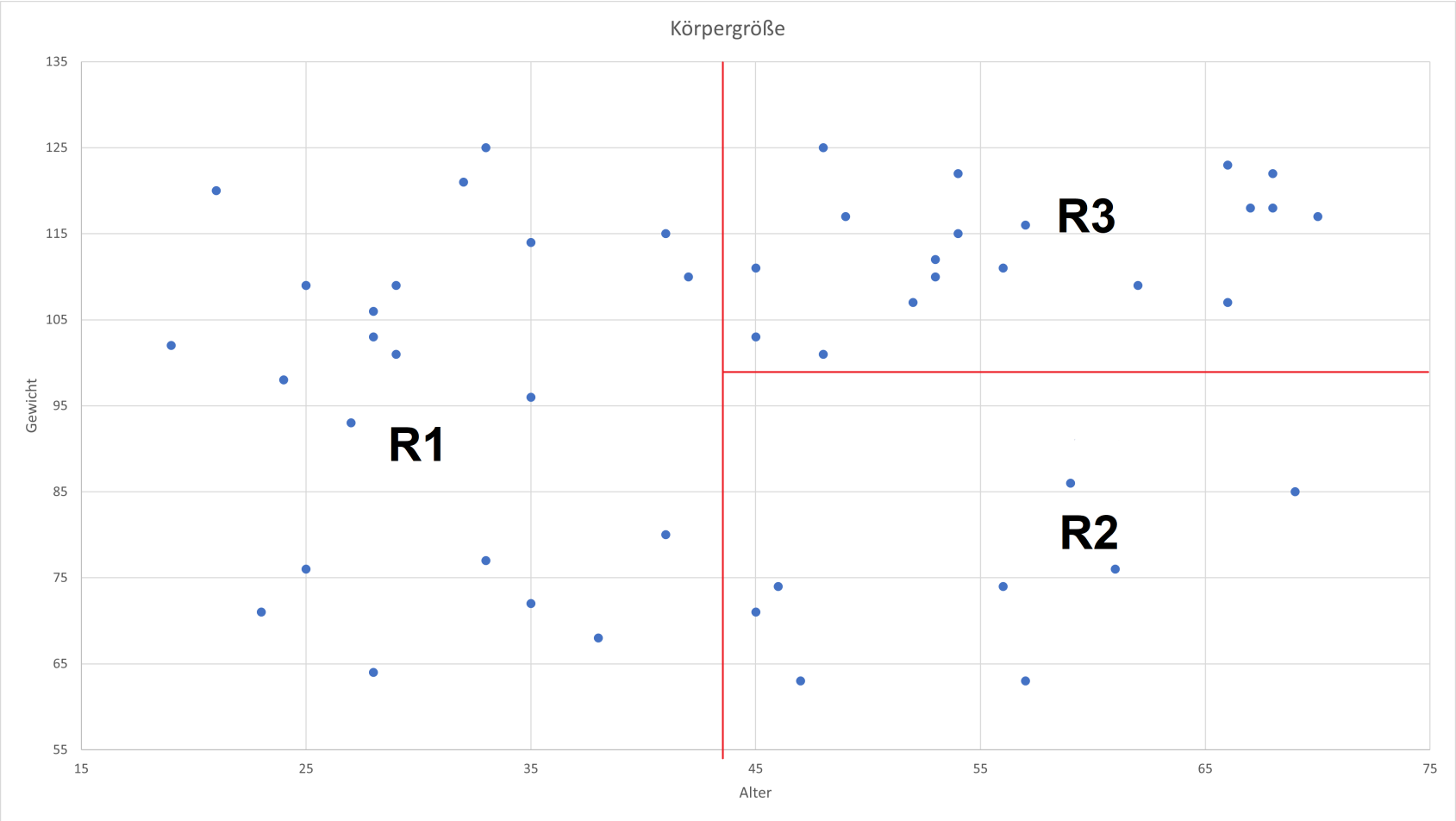
Regionen - Split

$$R_1(j,s) = \{X|X_j \leq s\} \quad , \quad R_2(j,s) = \{X|X_j > s\}$$

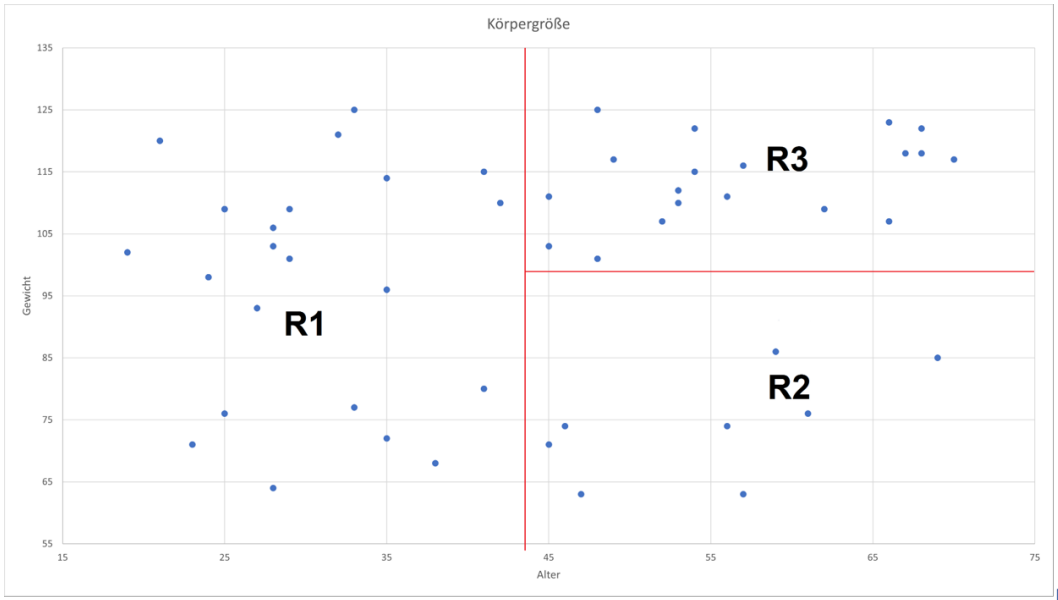


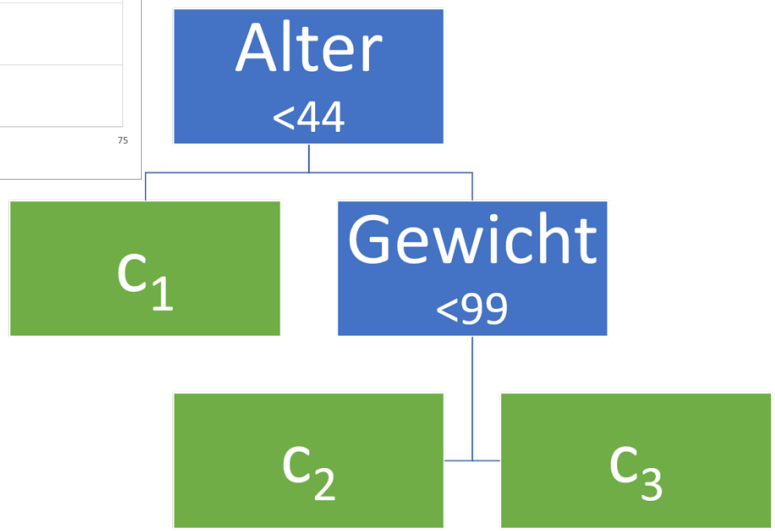
$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]$$

Regionen - zweiter Split

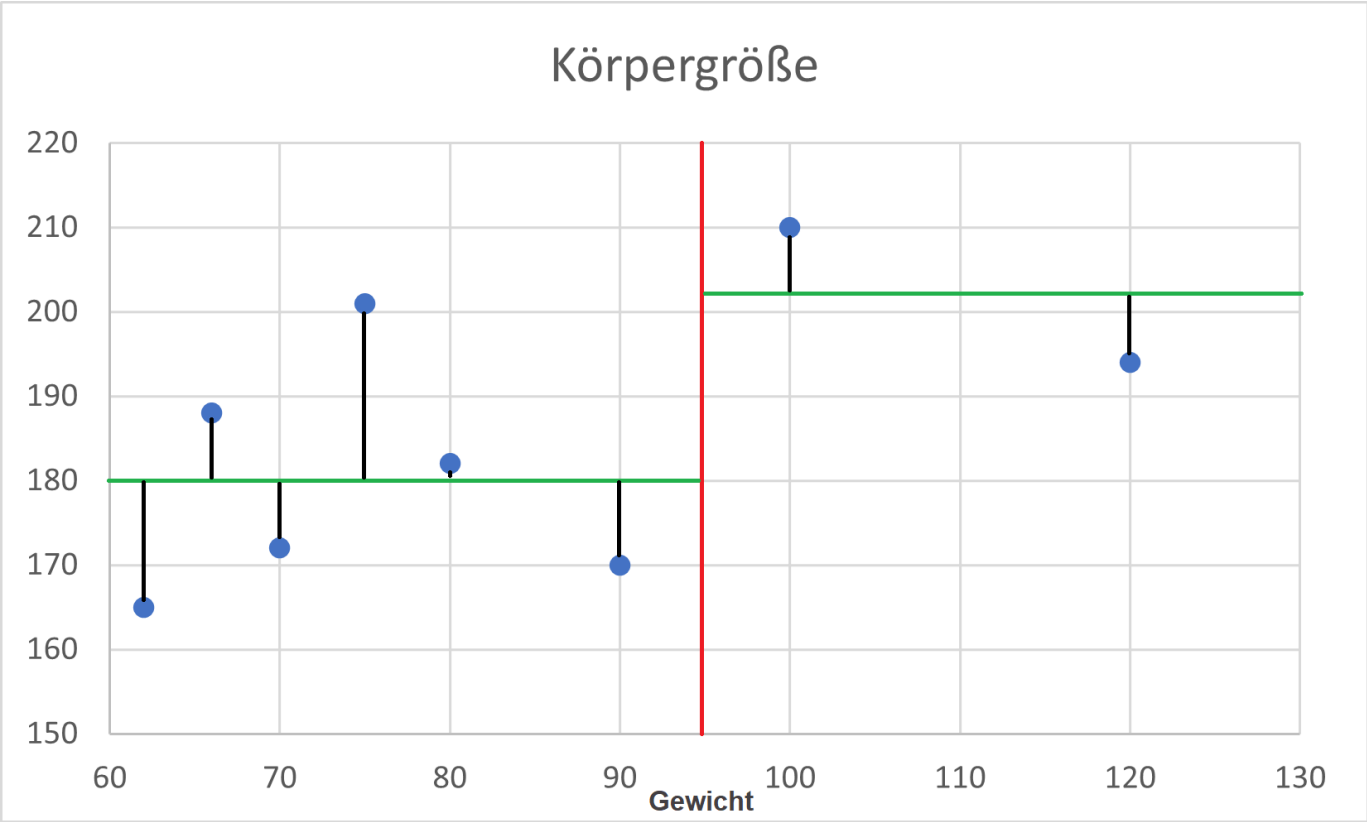


Regionen - zweiter Split



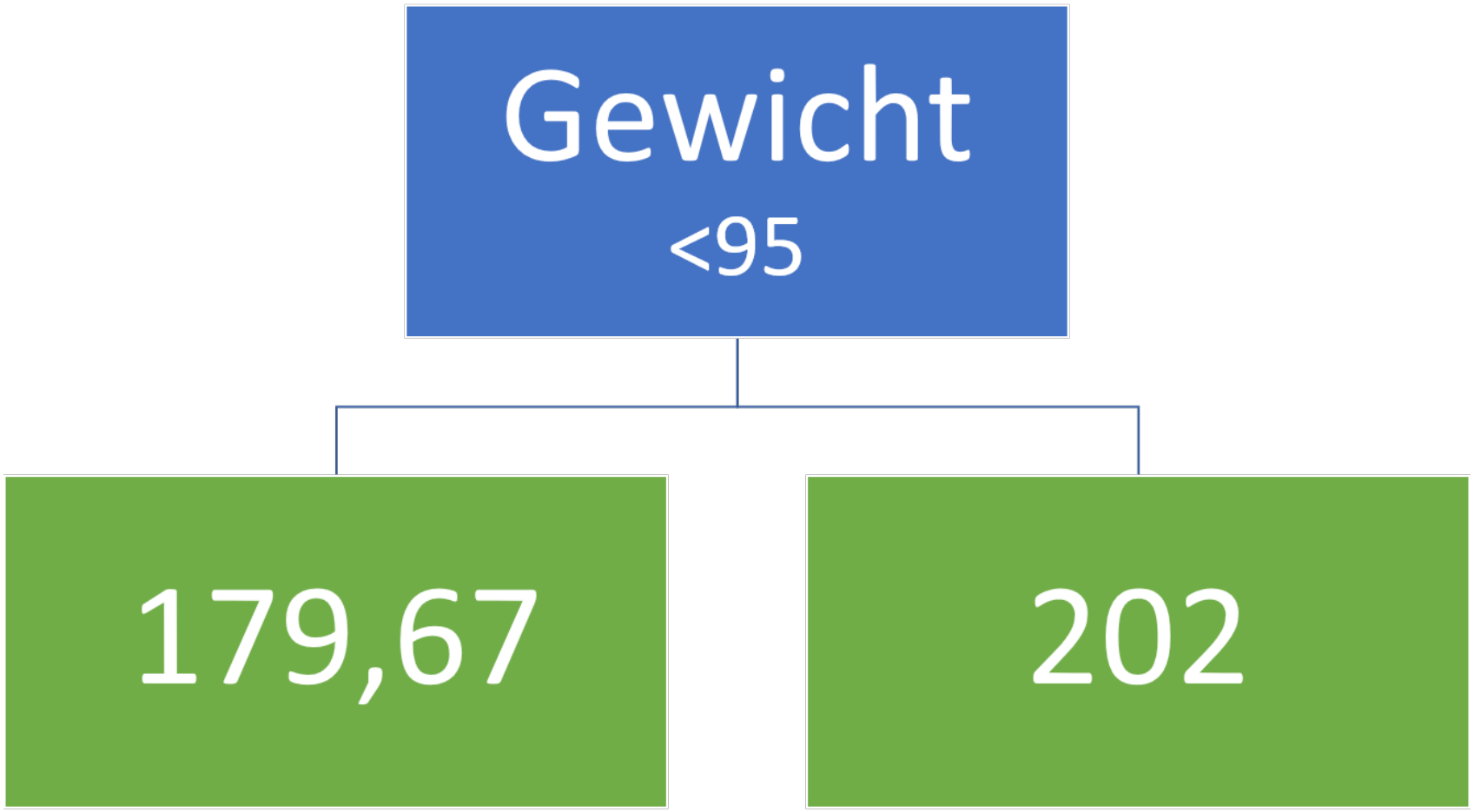


Beispiel - Regressionsbaum

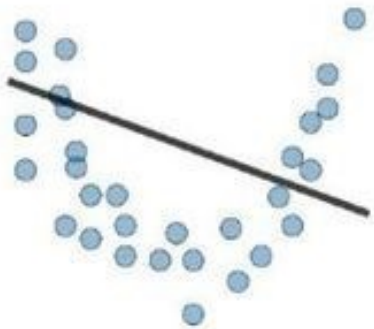


$$RSS = \sum_{x \in R_1} (f(x) - \hat{R}_1)^2 + \sum_{x \in R_2} (f(x) - \hat{R}_2)^2$$

Beispiel - Regressionsbaum



Overfitting

	Underfitting	Just right	Overfitting
Symptoms	- High training error - Training error close to test error - High bias	- Training error slightly lower than test error	- Low training error - Training error much lower than test error - High variance
Regression			

Pruning

- Entscheidungsbäume sind anfällig für Überanpassung
- Fachbegriff: Overfitting
- Pruning als Lösung
- kleinere Bäume sind vorteilhaft
- zurückschneiden vollständig entwickelter Bäume
- cost-complexity pruning: $\sum^{|T|} m = 1 \sum x_i \in R_m(yi - \hat{y}R_m)^2 + \alpha \cdot |T|$

Entscheidungsbäume - Fazit

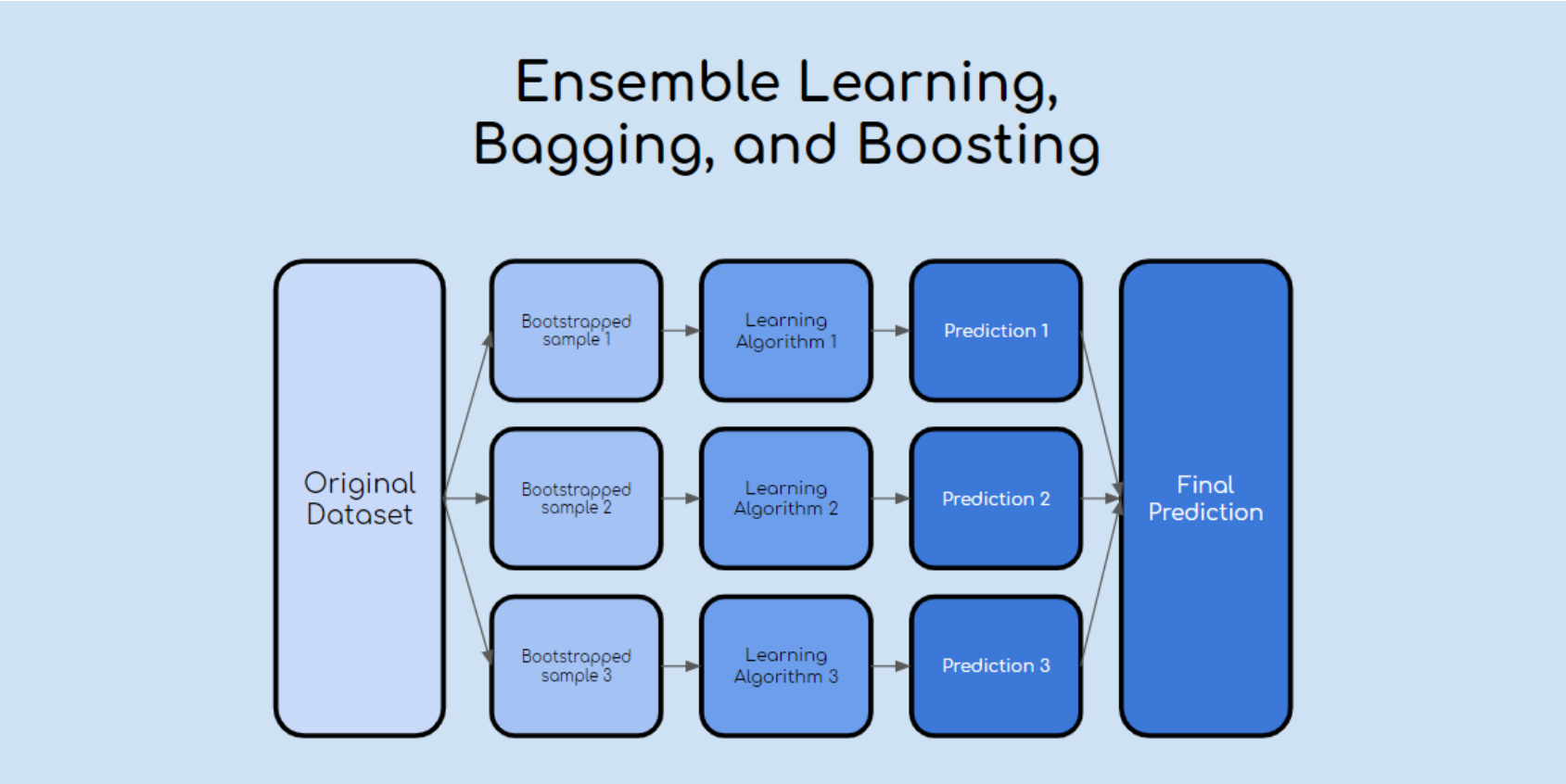
Vor- und Nachteile:

- ⊕ leicht zu verstehen
- ⊕ leicht zu interpretieren
- ⊕ berechnen simpel einen relativ guten Schätzer
- ⊖ Instabilität
- ⊖ Tendenz zu Overfitting
- ⊖ ungenau für sehr gute Schätzer

Übersicht

Motivation & Methodik

- Bagging macht ungenaue Methoden genauer
- Vor allem bei Bäumen hilfreich



Formel

$$\hat{f}_{bag} = \frac{1}{b} \sum_{i=1}^{b} f_i(x)$$

Durch Bagging wird die Varianz des Modells reduziert

$$\text{Varianz: } \sigma^2 \to \frac{\sigma^2}{b}$$

Von Bagging zu Random Forest

- Mit Bagging: 1 Baum pro Bootstrap (bei gleicher Methodik)
- Random Forest: Pro Bootstrap mehrere zufällig (Random) generierte Bäume (Forest)

Random Forest - Methodik

Analog zu Bagging

- Bootstraps generieren
- **Aus p Predictors werden nur $m < p$ (oft $\sqrt{p}$) betrachtet (pro Knoten)**
- Nach gewohntem System wird nach niedrigstem RSS gesucht
- Diese zufälligen Bäume werden wie bei Bagging ausgewertet

Übersicht

Boosting - Motivation

- Ensemble-Methode, welche weak learners kombiniert
- weak learner sind minimal besser als Raten
- weak learner haben keine komplexen Entscheidungsregeln
- Boosting senkt den Bias
- Boosting minimiert die Fehlerrate
- Differenz zwischen Regression und Klassifikation

Boosting - Allgemeine Verfahren

(1) Initialisiere einen Basisalgorithmus F_0

(2) Verstärke diesen schrittweise zu einem geboosteten Algorithmus F_i

(i) Regression: in F_{i-1} entstandene Fehler wirken sich auf F_i aus

(ii) Klassifikation: schlechte Schätzung von F_{i-1} erhält eine höhere Gewichtung in F_i

(3) Wiederhole das bilden neuer F_i bis Abbruchkriterium erfüllt ist

(4) Bilde deine Prediction aus der Kombination der F_i

Gradient Boosting - Vorgehen anhand eines Beispieles

Seien ein Trainingsdatensatz $\{(x_i, y_i)\}_{i=1}^{8}$ und die Lernrate $\ell = 0.1$ gegeben.

i	x_i			y_i
	Gewicht	Fußgröße	Handgröße	Körpergröße
1	80 kg	26 cm	18 cm	182 cm
2	62 kg	20 cm	12 cm	165 cm
3	120 kg	32 cm	24 cm	194 cm
4	70 kg	23 cm	13 cm	172 cm
5	90 kg	24 cm	15 cm	170 cm
6	66 kg	30 cm	20 cm	188 cm
7	75 kg	27 cm	19 cm	201 cm
8	100 kg	40 cm	30 cm	210 cm

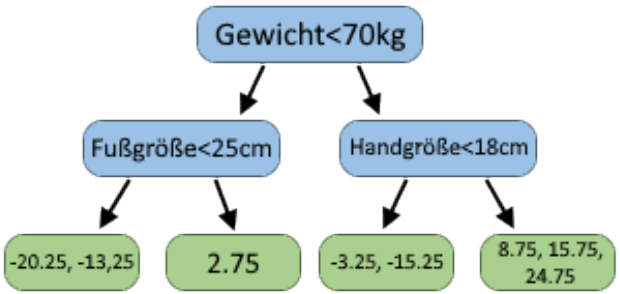
Wir berechnen den ersten Schätzer $F_0(x) = \frac{1}{8} \cdot \sum_{i=1}^{8} y_i = 185.25$.

Hinweis: Es wird noch kein Baum, sondern quasi nur ein Blatt mit F_0 gebildet.

Gradient Boosting - Vorgehen anhand eines Beispieles

Nun berechnen wir die Residuen $r_{i,1} = y_i - F_0(x)$, s.d. wir anschließend einen Entscheidungsbaum bilden können, der auf den Residuen $r_{i,1}$ gefittet wird:

i	x_i			y_i	$r_{i,1}$
	Gewicht	Fußgröße	Handgröße	Körpergröße	
1	80 kg	26 cm	18 cm	182 cm	-3,25
2	62 kg	20 cm	12 cm	165 cm	-20,25
3	120 kg	32 cm	24 cm	194 cm	8,75
4	70 kg	23 cm	13 cm	172 cm	-13,25
5	90 kg	24 cm	15 cm	170 cm	-15,25
6	66 kg	30 cm	20 cm	188 cm	2,75
7	75 kg	27 cm	19 cm	201 cm	15,75
8	100 kg	40 cm	30 cm	210 cm	24,75

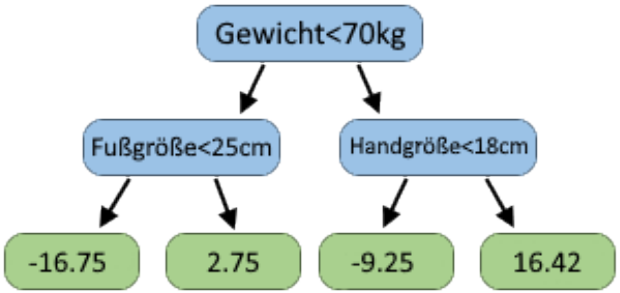


Hinweis: Falls mehrere Residuen in einem Blatt landen, bildet man den Durchschnitt dieser. So ergibt sich aus $r_{i,1}$ auch $\gamma_{j,1}$.

Gradient Boosting - Vorgehen anhand eines Beispieles

Wir haben gesehen, dass mehrere Residuen in einem Blatt gelandet sind. Daher berechnen wir jetzt die gefitteten Residuen $\gamma_{j,1}$ indem wir den Durchschnitt der Werte in einem Blatt berechnen:

i	x_i			y_i	$r_{i,1}$
	Gewicht	Fußgröße	Handgröße	Körpergröße	
1	80 kg	26 cm	18 cm	182 cm	-3,25
2	62 kg	20 cm	12 cm	165 cm	-20,25
3	120 kg	32 cm	24 cm	194 cm	8,75
4	70 kg	23 cm	13 cm	172 cm	-13,25
5	90 kg	24 cm	15 cm	170 cm	-15,25
6	66 kg	30 cm	20 cm	188 cm	2,75
7	75 kg	27 cm	19 cm	201 cm	15,75
8	100 kg	40 cm	30 cm	210 cm	24,75



Hinweis: $\gamma_{j,1}$ benötigen nur im nächsten Schritt.

Gradient Boosting - Vorgehen anhand eines Beispieles

Wir berechnen nun die neuen Residuen $r_{i,2} = r_{i,1} - \ell \cdot \gamma_{j,1}$, wobei $\gamma_{j,1}$ die gefitteten Residuen aus dem Baum sind:

i	x_i			y_i	$r_{i,1}$	$r_{i,2}$
	Gewicht	Fußgröße	Handgröße	Körpergröße		
1	80 kg	26 cm	18 cm	182 cm	-3,25	-2,32
2	62 kg	20 cm	12 cm	165 cm	-20,25	-18,58
3	120 kg	32 cm	24 cm	194 cm	8,75	7,11
4	70 kg	23 cm	13 cm	172 cm	-13,25	-11,58
5	90 kg	24 cm	15 cm	170 cm	-15,25	-14,33
6	66 kg	30 cm	20 cm	188 cm	2,75	2,47
7	75 kg	27 cm	19 cm	201 cm	15,75	14,11
8	100 kg	40 cm	30 cm	210 cm	24,75	23,11

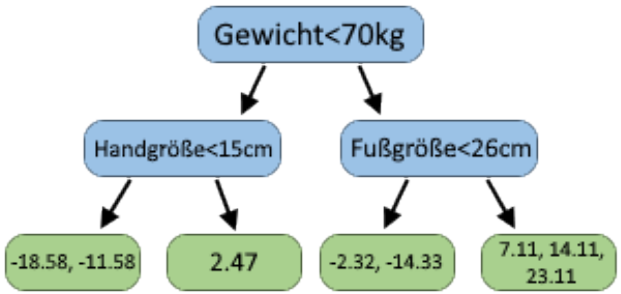
Hinweis: $r_{i,m} = (F_0 + r_{i,m-1}) - (F_0 + \ell \cdot \gamma_{j,m-1}) = F_0 + r_{i,m-1} - F_0 - \ell \cdot \gamma_{j,m-1}$

$= r_{i,m-1} - \ell \cdot \gamma_{j,m-1}$, *wobei m=1,...,M und* $m \in \mathbb{N}$

Gradient Boosting - Vorgehen anhand eines Beispieles

Wir bilden erneut einen Baum, der auf den neuen Residuen $r_{i,2}$ gefittet wird:

i	x_i			y_i	$r_{i,1}$	$r_{i,2}$
	Gewicht	Fußgröße	Handgröße	Körpergröße		
1	80 kg	26 cm	18 cm	182 cm	-3,25	-2,32
2	62 kg	20 cm	12 cm	165 cm	-20,25	-18,58
3	120 kg	32 cm	24 cm	194 cm	8,75	7,11
4	70 kg	23 cm	13 cm	172 cm	-13,25	-11,58
5	90 kg	24 cm	15 cm	170 cm	-15,25	-14,33
6	66 kg	30 cm	20 cm	188 cm	2,75	2,47
7	75 kg	27 cm	19 cm	201 cm	15,75	14,11
8	100 kg	40 cm	30 cm	210 cm	24,75	23,11

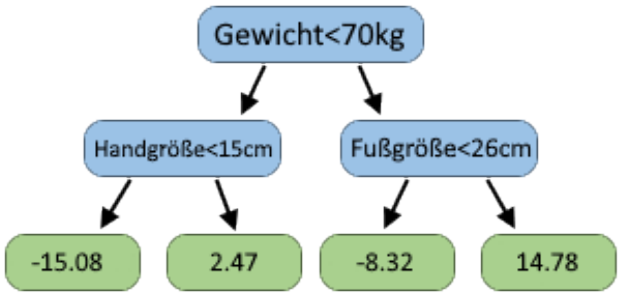


Hinweis: Der neue Baum kann, muss aber nicht die selbe Aufteilung haben.

Gradient Boosting - Vorgehen anhand eines Beispieles

Anschließend bestimmen wir wieder $\gamma_{j,2}$ und tragen es in den Blättern ein:

i	x_i			y_i	$r_{i,1}$	$r_{i,2}$
	Gewicht	Fußgröße	Handgröße	Körpergröße		
1	80 kg	26 cm	18 cm	182 cm	-3,25	-2,32
2	62 kg	20 cm	12 cm	165 cm	-20,25	-18,58
3	120 kg	32 cm	24 cm	194 cm	8,75	7,11
4	70 kg	23 cm	13 cm	172 cm	-13,25	-11,58
5	90 kg	24 cm	15 cm	170 cm	-15,25	-14,33
6	66 kg	30 cm	20 cm	188 cm	2,75	2,47
7	75 kg	27 cm	19 cm	201 cm	15,75	14,11
8	100 kg	40 cm	30 cm	210 cm	24,75	23,11



Gradient Boosting - Vorgehen anhand eines Beispieles

Diese Schritte wiederholt man bis das Abbruchkriterium erfüllt ist. Wir können also wieder $r_{i,3}$ bestimmen usw. Wie sehen aber unsere Schätzungen nun aus? Wir bestimmen $F_2(x)$:

$$\Rightarrow F_2(x) = F_0(x) + \ell \cdot \gamma_{j,1} + \ell \cdot \gamma_{j,2}$$

i	x_i			y_i	$r_{i,1}$	$r_{i,2}$	$r_{i,3}$	$F_2(x_i)$
	Gewicht	Fußgröße	Handgröße	Körpergröße				
1	80 kg	26 cm	18 cm	182 cm	-3,25	-2,32	-1,49	183,49 cm
2	62 kg	20 cm	12 cm	165 cm	-20,25	-18,58	-17,07	182,07 cm
3	120 kg	32 cm	24 cm	194 cm	8,75	7,11	5,63	188,37 cm
4	70 kg	23 cm	13 cm	172 cm	-13,25	-11,58	-10,07	182,07 cm
5	90 kg	24 cm	15 cm	170 cm	-15,25	-14,33	-13,49	183,49 cm
6	66 kg	30 cm	20 cm	188 cm	2,75	2,47	2,23	185,77 cm
7	75 kg	27 cm	19 cm	201 cm	15,75	14,11	12,63	188,37 cm
8	100 kg	40 cm	30 cm	210 cm	24,75	23,11	21,63	188,37 cm

Man sieht, dass sich die Schätzungen langsam an unsere Beobachtungen y_i annähern.

Gradient Boosting - Algorithmus

Input: Trainingsdaten $\{(x_i, y_i)\}_{i=1}^n$ und eine differenzierbare Verlustfunktion $L(y_i, F(x))$

Schritt 1: Initialisiere Modell mit $F_0(x) = \underset{\gamma}{\text{argmin}} \sum_{i=1}^n L(y_i, \gamma)$

Schritt 2: Wiederhole für m=1,...,M:

(i) Berechne $r_{i,m} = -[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{F(x)=F_{m-1}(x)}$ für i=1,...,n

(ii) Bilde einen Entscheidungsbaum, der auf den $r_{i,m}$ fittet und bilde die Regionen $R_{j,m}$ für $j = 1, ..., J_m$

(iii) Berechne $\gamma_{j,m} = \underset{\gamma}{\text{argmin}} \sum_{x_i \in R_{j,i}} L(y_i, F_{m-1}(x_i) + \gamma)$ für $j = 1, ..., J_m$

(iv) Bestimme $F_m(x) = F_{m-1}(x) + \ell \cdot \sum_{j=1}^J \gamma_{j,m} I_{(x \in R_{j,m})}$

Output: $F_M(x)$

Gradient Boosting - Verlustfunktionen

Fehlerart	Verlustfunktion (engl. Loss Function)	Gradient der Verlustfunktion
Quadratisch	$L(y_i, F(x_i)) = \frac{1}{2} \cdot [y_i - F(x_i)]^2$	$y_i - F(x_i)$
Absolut	$L(y_i, F(x_i)) = \lvert y_i - F(x_i) \rvert$	$sign(y_i - F(x_i))$
Huber	$L(y_i, F(x_i)) = \begin{cases} \frac{1}{2} \cdot (y_i - F(x_i))^2, & falls \lvert y_i - F(x_i) \rvert \leq \delta \\ \delta \cdot \lvert y_i - F(x_i) \rvert - \frac{\delta^2}{2}, & falls \lvert y_i - F(x_i) \rvert > \delta \end{cases}$	$\begin{cases} y_i - F(x_i), & falls \lvert y_i - F(x_i) \rvert \leq \delta \\ \delta \cdot sign(y_i - F(x_i)), & falls \lvert y_i - F(x_i) \rvert > \delta \end{cases}$

Abbildung: Gradienten für häufig benutzte Loss-Funktionen

Adaptive Boost - Algorithmus

Schritt 1: Initialisiere Stichproben Gewichte $w_i = \frac{1}{n}$, wobei i=1,...,n

Schritt 2: Wiederhole für m=1,...,M:

(i) Fitte einen Klassifizierer $G_m(x)$ für die Trainingsdaten in Abhängigkeit von w_i

(ii) Berechne totale Fehlerrate $err_m = \frac{\sum_{i=1}^{n} w_i \cdot I_{(y_i \neq G_m(x_i))}}{\sum_{i=1}^{n} w_i}$

(iii) Berechne Stimmgewicht $\alpha_m = log(\frac{1-err_m}{err_m})$

(iv) Setze $w_i \leftarrow w_i \cdot exp(\alpha_m \cdot I_{(y_i \neq G_m(x_i))})$, für i=1,...,n

Output: $G(x) = sign[\sum_{m=1}^{M} \alpha_m \cdot G_m(x)]$

Gradient Boosting vs. AdaBoost

- Adaptive Boost war ursprünglich für Klassifikation gedacht
- AdaBoost wandelt Regressionsproblem in Klassifikation um
- AdaBoost bildet nur Baumstümpfe, Gradient Boost hingegen bildet Bäume meist mit 8-32 Blättern
- AdaBoost gewichtet die Stichproben, Gradient Boost nutzt eine Verlustfunktion

Boosting vs. Bagging - Fazit

(1) Bagging:

→ unabhängiges Training

→ Reduktion von Varianz

→ Vermeidung von Overfitting

→ keine Gewichtung der Bäume

(2) Boosting:

→ sequentielles Training

→ Reduktion von Varianz und Bias

→ anfällig für Overfitting

→ Gewichtung der Bäume

Literatur

(1) The Elements of Statistical Learning; T.Hastie, R.Tibshirani, J.Friedmann; Springer Verlag (2008)

(2) An Introduction to Statistical Learning; G.James, D.Witten, T.Hastie, R.Tibshirani; Springer Verlag (2017)

(3) Statistisches und Maschinelles Lernen; S.Richter; Springer Verlag (2019)