Block 3: Komplexität, Kausalität & Generalisierung

Data Science and Strategy for Business – Eine Lernreise mit Lena

March 31, 2026

Lernziele Block 3

- Multikollinearität erkennen und Auswirkungen erklären
- BLUE-Eigenschaften konzeptionell einordnen
- Overfitting und In-/Out-of-Sample Performance unterscheiden
- Kreuzvalidierung und Regularisierung erklären
- Korrelation von Kausalität unterscheiden
- A/B-Tests als kausale Inferenz verstehen
- Logistische Regression von OLS abgrenzen

Dieser Block bereitet auf fortgeschrittene ML-Methoden vor (Modul ER017).

Lena bei DataCo: Kann ich meinen Zahlen vertrauen?

Lenas Situation

Lena hat gerade bei DataCo angefangen – 50.000 Kundendatensätze, ein Auftrag: *„Sag mir, wer kündigt."*

- Sie öffnet RStudio und tippt `lm(churn ~ ., data = df)`
- Sofort: Koeffizienten, Standardfehler, p-Werte
- Alles sieht professionell aus – aber stimmen die Zahlen?
- **Kernfrage:** Wann kann ich meinem Modell vertrauen – und wann nicht?

Ziel dieser Sektion: Die Bedingungen verstehen, unter denen OLS optimale Ergebnisse liefert.

Vertrauen in Modelle erfordert Verständnis der Annahmen dahinter.

Entdeckungsfrage

Was macht eine gute Waage aus?

Stellen Sie sich vor, Sie wiegen denselben Gegenstand zehnmal hintereinander.

Was wünschen Sie sich von den Ergebnissen – und was wäre schlimm?

Denken Sie 30 Sekunden nach, bevor wir weitergehen.

Diese Frage führt direkt zu den drei Kerneigenschaften eines guten Schätzers.

Die Waage-Analogie: Drei Eigenschaften

Ein Päckchen Butter (250 g), zehnmal gewogen. Was wünschen wir uns?

Eigenschaft 1: Genau (Unbiased)

- Im Mittel den richtigen Wert – nicht systematisch 245 g oder 260 g
- Statistik: $E[\hat{\beta}] = \beta$ (Erwartungstreue)

Eigenschaft 2: Stabil (Low Variance)

- Messungen zwischen 248–252 g: gut. Zwischen 200–300 g: unbrauchbar
- Statistik: Der Schätzer hat eine kleine Varianz

Eigenschaft 3: Beste (Best)

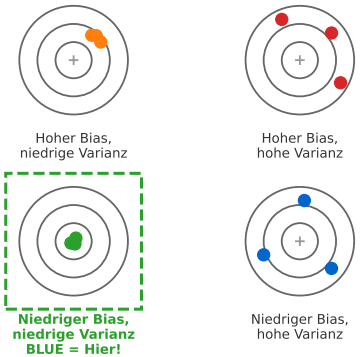
- Unter allen genauen Waagen: die stabilste wählen
- Statistik: Kleinste Varianz unter allen erwartungstreuen Schätzern

---

Genau + Stabil + Beste = die drei Säulen von BLUE.

Genauigkeit vs. Präzision: Die Zielscheibe

BLUE: Bias und Varianz als Zielscheibe



Vier Szenarien:

- Genau + Präzise – Idealfall
- Genau + Unpräzise – instabil
- Ungenau + Präzise – Bias
- Ungenau + Unpräzise – beides

OLS strebt den Idealfall an – unter bestimmten Bedingungen.

BLUE = genau UND so präzise wie möglich unter allen linearen Schätzern.

BLUE – Die vier Buchstaben

BLUE = Best Linear Unbiased Estimator

Der beste lineare erwartungstreue Schätzer

E – Estimator Eine Rechenvorschrift, die aus Daten einen Schätzwert berechnet.
OLS-Schätzer $\hat{\beta}$: nimmt (X, y) und liefert Koeffizienten.

U – Unbiased Bei 1000 Wiederholungen trifft der Mittelwert aller $\hat{\beta}$ den wahren Wert β.
Formal: $E[\hat{\beta}] = \beta$

L – Linear $\hat{\beta}$ ist eine gewichtete Summe der y-Werte:
$\hat{\beta} = \sum_i w_i \cdot y_i$, wobei w_i nur von x abhängt.

B – Best Unter allen linearen, erwartungstreuen Schätzern hat OLS die **kleinste Varianz**.

BLUE heißt: Kein anderer linearer, erwartungstreuer Schätzer ist präziser als OLS.

BLUE in der Praxis: Was bedeutet das?

Für Lena bei DataCo:

- Wenn Bedingungen erfüllt → `lm()` liefert das Bestmögliche
- Kein anderer linearer Schätzer wäre präziser
- Die Standardfehler sind korrekt
- p-Werte und Konfidenzintervalle stimmen

Aber Vorsicht:

- „Best" = nur Varianz, nicht Gesamtfehler
- Nichtlineare Schätzer können besser sein
- Bei kleinem n oft irrelevant
- Schätzer mit etwas Bias, aber weniger Varianz → niedrigerer MSE

$\text{MSE} = \text{Bias}^2 + \text{Varianz}$

→ Grundidee der Regularisierung (Abschnitt 4)

BLUE ist theoretisch elegant, praktisch aber nicht immer optimal.

Gauss-Markov-Annahmen (1–3)

Die BLUE-Eigenschaft gilt nur unter 5 Annahmen:

Annahme 1: Linearität in den Parametern

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \varepsilon_i$$

Linear in β, nicht notwendig in x. Beispiel: $y = \beta_0 + \beta_1 x^2$ ist erlaubt.

Annahme 2: Zufällige Stichprobe

- Beobachtungen unabhängig und aus derselben Grundgesamtheit
- Bei Lena: Kein Kunde beeinflusst einen anderen

Annahme 3: Keine perfekte Multikollinearität

- Kein Prädiktor ist exakte Linearkombination der anderen
- Approximative Korrelation ($r = 0{,}9$) erlaubt, aber problematisch

Annahmen 1–3 betreffen die Datenstruktur – prüfbar vor der Modellierung.

Gauss-Markov-Annahmen (4–5)

Annahme 4: Exogenität

$$E[\varepsilon_i \mid X] = 0$$

- Kein systematischer Zusammenhang zwischen Prädiktoren und Fehler
- Verletzung z. B. durch fehlende Variablen → verzerrte Schätzer

Annahme 5: Homoskedastizität

$$\mathrm{Var}(\varepsilon_i \mid X) = \sigma^2 \quad \text{für alle } i$$

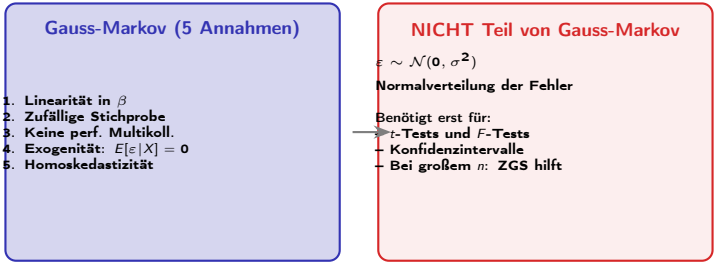
- Konstante Fehlervarianz für alle Beobachtungen
- Verletzung: „Trichter-Muster" in Residuenplots (Heteroskedastizität)

Gauss-Markov-Theorem

Wenn alle 5 Annahmen erfüllt → OLS ist BLUE. Kein besserer linearer Schätzer existiert.

Annahmen 4–5 betreffen die Fehlerstruktur – prüfbar mit Residuenanalyse.

Was gehört NICHT zu Gauss-Markov?



Merke: Für BLUE brauchen wir *keine* Normalverteilung. Normalität ist erst für die Inferenz nötig.

Eines der häufigsten Missverständnisse in der Statistik-Ausbildung.

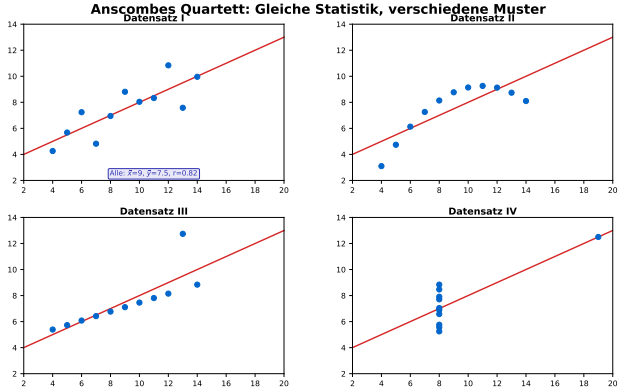
Annahmen systematisch prüfen

Annahme	Diagnostik	R-Funktion
Linearität	Residuen vs. Fitted Plot	`plot(model, 1)`
Zufällige Stichprobe	Studiendesign prüfen	(theoretisch)
Keine perf. Multikoll.	VIF berechnen	`car::vif()`
$E[\varepsilon \mid X] = 0$	Theorie + Residuen	`plot(model, 1)`
Homoskedastizität	Scale-Location Plot	`lmtest::bptest()`
Zusätzlich:		
Normalität	Q-Q Plot	`shapiro.test()`
Keine Autokorrelation	Durbin-Watson Test	`lmtest::dwtest()`

Faustregel: `plot(model)` in R erzeugt automatisch 4 Diagnose-Grafiken.

Immer Annahmen prüfen – blinde Anwendung von OLS ist gefährlich.

Anscombes Quartett: Warum Visualisierung zählt



Vier Datensätze, eine Statistik:

- Gleicher Mittelwert ($\bar{x} = 9$, $\bar{y} = 7{,}5$)
- Gleiche Korrelation ($r = 0{,}82$)
- Gleiche Regressionslinie

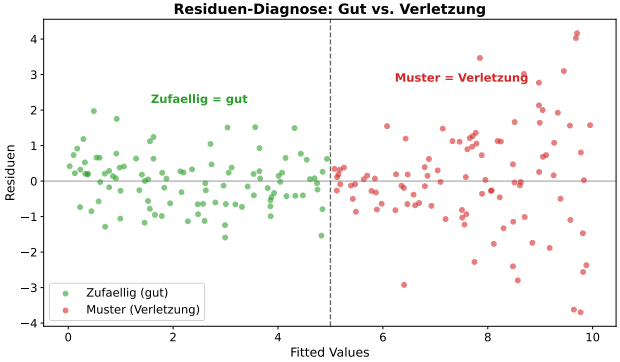
Aber völlig verschiedene Muster:

- Linear, kurvilinear, Ausreißer, einflussreiche Beobachtung

Lektion: Kennzahlen allein reichen nicht – man muss die Daten *sehen*.

Francis Anscombe (1973) konstruierte dieses Beispiel als Warnung vor blinder Statistik.

Was passiert bei Annahmenverletzungen?



Residuenplots zeigen Probleme:

- U-förmiges Muster → Linearität verletzt
- Trichter-Form → Heteroskedastizität
- Kein Muster → Annahmen erfüllt

In R: `plot(model)` erzeugt 4 Diagnose-Grafiken automatisch.

Residuenplots sind das wichtigste Werkzeug zur Annahmenprüfung.

Zahlenbeispiel: Was ändert Heteroskedastizität?

Lena schätzt $\hat{\beta}_1$ (Vertragslaufzeit $\rightarrow$ Churn) mit $n = 500$, $p = 4$:

Szenario	$\hat{\beta}_1$	*SE*	*t*	*p*
Korrekte Annahmen	2,3	0,4	5,75	$< 0{,}001$
Mit Heteroskedastizität	2,3	0,8	2,88	0,004

Was fällt auf?

- Koeffizient $\hat{\beta}_1 = 2{,}3$ bleibt gleich – OLS bleibt erwartungstreu
- Standardfehler **verdoppelt** sich: $0{,}4 \rightarrow 0{,}8$
- t-Wert **halbiert** sich: $5{,}75 \rightarrow 2{,}88$
- Bei kleinerem Effekt ($\hat{\beta}_1 = 1{,}0$): $t = 1{,}25$ – Signifikanz verschwindet!

Verletzte Annahmen ändern nicht die Schätzung, aber die Inferenz – das ist tückisch.

Geschichte: Gauss und der Asteroid Ceres (1801)

- **1801:** Giuseppe Piazzi entdeckt Ceres – nur 41 Tage Beobachtung
- Ceres verschwindet hinter der Sonne – nur 22 Messungen über 3° Bahnbogen
- **Carl Friedrich Gauss**, 24 Jahre alt, entwickelt die Methode der kleinsten Quadrate
- Seine Vorhersage: so präzise, dass Ceres am 31. Dezember 1801 genau dort gefunden wird

Von Gauss zu Lena:

- Gauss: 22 Messwerte, Papier
- Lena: 50.000 Zeilen, `lm()`
- Die Methode ist dieselbe!

∼100 Jahre später: Andrei Markov formalisiert die optimalen Eigenschaften → Gauss-Markov-Theorem

Die Methode der kleinsten Quadrate ist über 200 Jahre alt – und immer noch Standard.

Drei häufige Fehler über BLUE

Fehler 1: „BLUE heißt, OLS ist immer optimal."

- Nein! Gilt nur unter den 5 Annahmen. Nichtlineare Schätzer (Ridge, Lasso) können niedrigeren MSE haben.

Fehler 2: „Normalverteilung ist für BLUE nötig."

- Nein! Normalität brauchen wir erst für Inferenz (t-Tests, KI). Bei großem n hilft der ZGS.

Fehler 3: „Bei verletzten Annahmen ist OLS nutzlos."

- Nein! Z. B. bei Heteroskedastizität: OLS bleibt erwartungstreu. Nur Standardfehler werden falsch. Lösung: robuste SE (`sandwich`-Paket).

Diese drei Missverständnisse tauchen regelmäßig in Klausuren auf.

R-Code: Modelldiagnostik

```
# Modell schaetzen
model <- lm(churn_score ~ alter + vertragslaufzeit +
            monatl_kosten + nutzung_gb, data = df)

# Diagnose-Plots (4 auf einmal)
par(mfrow = c(2, 2))
plot(model)

# Breusch-Pagan Test (Heteroskedastizitaet)
library(lmtest)
bptest(model)    # p < 0.05 -> Heterosked. vorhanden

# VIF (Multikollinearitaet)
library(car)
vif(model)       # VIF > 5: problematisch
```

Diese drei Checks (Plot, Breusch-Pagan, VIF) decken die wichtigsten Probleme ab.

Übungsaufgaben: BLUE und Gauss-Markov

Aufgabe 1: Waage und Schätzer
Eine Waage zeigt bei 10 Messungen (wahres Gewicht 100 g): 98, 102, 99, 101, 100, 103, 97, 101, 99, 100.
(a) Mittelwert berechnen – ist die Waage erwartungstreu?
(b) Welcher BLUE-Eigenschaft entspricht (a)?

Aufgabe 2: Annahmen identifizieren
Welche Gauss-Markov-Annahme ist verletzt?
(a) Der Zusammenhang ist exponentiell, aber linear modelliert.
(b) „Monatl. Kosten" und „Jahreskosten $= 12\times$ Monatl. Kosten" im Modell.
(c) Residuenstreuung wächst mit steigendem Umsatz.

Aufgabe 3: Heteroskedastizität
$\hat{\beta}_1 = 3{,}5$, $SE = 0{,}7$ unter korrekten Annahmen. Heteroskedastizität verdreifacht SE auf 2,1.
Berechnen Sie beide t-Werte. Ist $\hat{\beta}_1$ noch signifikant?

Lösungen werden in der Übung besprochen.

Lena bei DataCo: Etwas stimmt nicht

Lenas Problem

Lena fügt zwei neue Variablen hinzu: Vertragslaufzeit und monatliche Kosten. Plötzlich verhält sich das Modell seltsam.

- Vertragslaufzeit wechselt das Vorzeichen (positiv $\rightarrow$ negativ!)
- Standardfehler explodieren
- Vorher signifikante Prädiktoren verlieren Signifikanz
- Aber R^2 steigt sogar leicht

Was ist passiert? Die neuen Variablen sind stark miteinander korreliert – das Modell kann ihre Effekte nicht trennen.

Instabile Koeffizienten sind das Warnsignal Nr. 1 für Multikollinearität.

Entdeckungsfrage

Was passiert, wenn wir eine Variable exakt kopieren und beide Kopien ins Modell stecken?

Kann das Modell die einzelnen Effekte noch trennen?

Und was ändert sich, wenn die Kopie nicht perfekt ist, sondern nur fast identisch?

Gedankenexperiment: Überlegen Sie 30 Sekunden.

Dieses Gedankenexperiment führt direkt zum Konzept der Multikollinearität.

Die Zwillings-Analogie: Zeugen vor Gericht

Unabhängige Zeugen:

- Zwei Zeugen berichten unabhängig
- Einer beschreibt den Täter, der andere den Fluchtweg
- Jeder trägt *eigene* Information bei
- Richter kann beide Aussagen nutzen

Eineiige Zwillinge:

- Immer zusammen, immer dasselbe gesehen
- Aussagen praktisch identisch
- „Was haben *Sie* gesehen?" – „Dasselbe wie er."
- Richter kann Beiträge nicht trennen

Wenn Prädiktoren korreliert sind, kann das Modell ihre Beiträge nicht trennen.

Von Zwillingen zu Prädiktoren

Bei Gericht:

- Zwillinge = korrelierte Prädiktoren
- Richter = OLS-Algorithmus
- Aussage = Information im Modell
- „Wer war's?" = Koeffizientenschätzung

Konsequenz:
Der Richter weiß, dass *einer* der Zwillinge Recht hat – aber nicht *welcher*. Genauso weiß OLS, dass die Prädiktoren erklären – aber nicht *wer wie viel*.

Wichtige Unterscheidung:

Vorhersage: Gesamtmodell funktioniert trotzdem! Die gemeinsame Information ist vorhanden.

Interpretation: Einzelne Koeffizienten sind unbrauchbar. Welcher Prädiktor wie viel beiträgt, bleibt unklar.

→ Ob Multikollinearität ein *Problem* ist, hängt vom Ziel ab.

Multikollinearität schadet der Interpretation, weniger der Vorhersage.

Multikollinearität: Exakt vs. Approximativ

Exakte (perfekte) Multikollinearität

Ein Prädiktor ist eine *exakte* Linearkombination der anderen: $x_3 = 2x_1 + x_2$.
OLS kann Koeffizienten **nicht berechnen**. R setzt einen Koeffizienten auf `NA`.

Approximative Multikollinearität

Prädiktoren sind *stark*, aber nicht perfekt korreliert: z. B. $r = 0{,}92$.
OLS funktioniert technisch, aber Schätzungen werden **unzuverlässig**.

In der Praxis:

- Exakte Multikollinearität → selten, sofort erkennbar
- Approximative Multikollinearität → häufig und tückisch
- Lenas Fall: Vertragslaufzeit und monatliche Kosten korrelieren mit $r \approx 0{,}9$

Der approximative Fall ist der gefährliche – weil das Modell läuft, aber falsche Schlüsse liefert.

Symptome der Multikollinearität (1/2)

Symptom 1: Hohes R^2, keine signifikanten Koeffizienten

- Gesamtmodell erklärt viel Varianz
- Aber kein einzelner Prädiktor ist signifikant
- Paradox? Nein – die Prädiktoren teilen sich die Erklärungskraft
- Keiner bekommt genug zugeteilt, um allein signifikant zu sein

Symptom 2: Aufgeblähte Standardfehler

- SE der Koeffizienten unverhältnismäßig groß
- Dadurch: t-Werte klein, p-Werte groß
- Lena findet keinen signifikanten Effekt – obwohl er existiert
- Typ-II-Fehler: Echter Effekt wird übersehen

Symptom 1 und 2 zusammen: Das typische Bild der Multikollinearität.

Symptome der Multikollinearität (2/2)

Symptom 3: Instabile Koeffizienten

- Eine Beobachtung entfernen → Koeffizienten ändern sich dramatisch
- Ein stabiles Modell sollte robust gegenüber kleinen Änderungen sein
- Bei Multikollinearität: kleine Datenänderung = große Schätzänderung

Symptom 4: Vorzeichenwechsel

- Ein Prädiktor, der theoretisch positiv wirken sollte, wird plötzlich negativ
- OLS „überkompensiert" bei der Aufteilung überlappender Information
- Genau das hat Lena beobachtet: Vertragslaufzeit wechselt das Vorzeichen

Zusammenfassung: Hohes R^2 + große SE + instabile Koeffizienten + Vorzeichenwechsel
= Multikollinearität verdächtigt

Wenn mehrere Symptome gleichzeitig auftreten: VIF berechnen!

Der Variance Inflation Factor (VIF)

Idee: Jeden Prädiktor X_j auf alle anderen regressieren. Hohes $R_j^2 \rightarrow$ Multikollinearität.

VIF-Formel

$$VIF_j = \frac{1}{1 - R_j^2}$$

Symbol für Symbol:

- VIF_j – Variance Inflation Factor für Prädiktor j
- R_j^2 – Bestimmtheitsmaß der Regression von X_j auf alle übrigen Prädiktoren
- Wenn $R_j^2 = 0$ (keine Korrelation): $VIF_j = 1$ – keine Inflation
- Wenn $R_j^2 = 0{,}9$: $VIF_j = 10$ – Varianz **zehnfach** aufgeblasen
- Wenn $R_j^2 \rightarrow 1$: $VIF_j \rightarrow \infty$ – perfekte Multikollinearität

VIF misst, um welchen Faktor die Varianz des Koeffizienten aufgeblasen wird.

VIF-Schwellenwerte und Interpretation

VIF-Wert	$\sqrt{VIF}$ (SE-Faktor)	Bewertung
$VIF = 1$	1,0×	Keine Korrelation – ideal
$VIF = 2$	1,4×	Unkritisch
$VIF = 5$	2,2×	Problematisch – genauer untersuchen
$VIF = 10$	3,2×	Schwer problematisch – Handlung nötig
$VIF = 50$	7,1×	Fast perfekte Multikollinearität

Faustregeln:

- $VIF < 5$: unkritisch
- $5 \leq VIF < 10$: problematisch – genauer untersuchen
- $VIF \geq 10$: schwer problematisch – Handlung erforderlich

Merke: $\sqrt{VIF}$ zeigt, um welchen Faktor der SE aufgeblasen ist.
$VIF = 10 \rightarrow SE$ ist 3,2× so groß wie ohne Multikollinearität.

VIF > 5 verdient Aufmerksamkeit, VIF > 10 erfordert Handlung.

Beispiel: Größe und Gewicht als Prädiktoren

Gesundheitsstudie, $n = 300$. Prädiktoren: Körpergröße und Gewicht ($r = 0{,}78$).

Modell	**Variable**	$\hat{\beta}$	SE	p
Nur Größe	Größe	0,45	0,12	$< 0{,}001$
Nur Gewicht	Gewicht	0,52	0,11	$< 0{,}001$
Beide zusammen	Größe	0,18	0,28	0,52
Beide zusammen	Gewicht	0,25	0,26	0,34

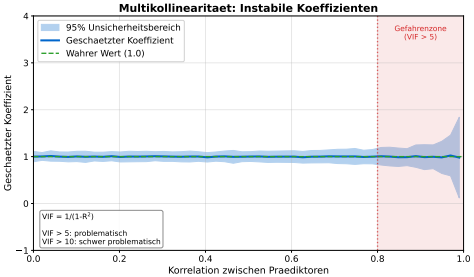
Was passiert?

- Einzeln: beide hochsignifikant
- Zusammen: keiner mehr signifikant – SE hat sich verdoppelt!
- Falsche Schlussfolgerung: „Weder Größe noch Gewicht beeinflussen Blutdruck"
- Richtig: Das Problem liegt an der Multikollinearität, nicht an fehlenden Effekten

Ein Klassiker in der Gesundheitsforschung – korrelierte Prädiktoren maskieren echte Effekte.

Lenas DataCo-Modell: VIF-Analyse

Feature	R_j^2	VIF	Bewertung
Vertragslaufzeit	0,82	5,6	problematisch
Monatl. Kosten	0,78	4,5	grenzwertig
Alter	0,12	1,14	unkritisch
Nutzung (GB)	0,25	1,33	unkritisch



Berechnung (Vertragslaufzeit):

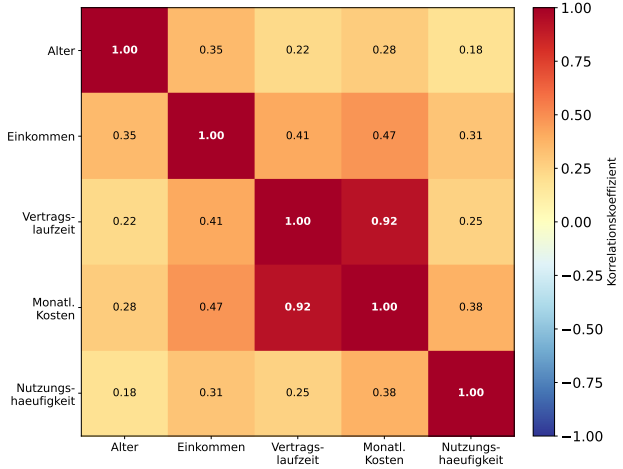
$$VIF = \frac{1}{1 - 0{,}82} = \frac{1}{0{,}18} = 5{,}6$$

$\sqrt{5{,}6} = 2{,}4 \rightarrow$ SE ist 2,4$\times$ aufgeblasen

Fazit: Vertragslaufzeit und monatliche Kosten überlappen stark. Alter und Nutzung sind unproblematisch.

VIF identifiziert die Problemvariablen – hier: Vertragslaufzeit und Kosten.

Korrelationsmatrix als Heatmap



Heatmap zeigt paarweise Korrelationen:

- Dunkelrot = hohe Korrelation → VIF prüfen
- Hellblau = geringe Korrelation → unproblematisch
- Muster auf einen Blick erkennen

Ergänzt den VIF: Heatmap zeigt *welche* Paare korreliert sind, VIF quantifiziert die Auswirkung.

Korrelationsmatrizen + VIF = vollständige Multikollinearitäts-Diagnostik.

Fünf Lösungen bei Multikollinearität

1. **Variable entfernen** – Einfach, aber: welche? Inhaltliche Gründe, nicht nur VIF!
2. **Variablen kombinieren** – Index bilden, z. B. Vertragswert = Laufzeit × Kosten. Keine Information verloren, aber schwerer interpretierbar.
3. **Regularisierung (Ridge/Lasso)** – Ridge schrumpft korrelierte Koeffizienten zueinander. Lasso setzt sie auf Null. → Abschnitt 4.
4. **Hauptkomponentenanalyse (PCA)** – Neue, unkorrelierte Variablen. Multikollinearität vollständig beseitigt, aber abstrakte Interpretation.
5. **Mehr Daten sammeln** – Größere n → kleinere SE. Oft nicht praktikabel, aber der sauberste Ansatz.

Entscheidend: Ziel bestimmt die Lösung.

- Vorhersage: Multikollinearität oft tolerierbar
- Interpretation: Multikollinearität muss behandelt werden

Die richtige Lösung hängt vom Analyseziel und der kausalen Struktur ab.

Geschichte: Hoerl & Kennard und Ridge Regression (1970)

- **Arthur Hoerl und Robert Kennard** – Chemiker bei DuPont
- Industrielle Chemie: Temperatur, Druck, Konzentration hängen physikalisch zusammen
- OLS-Koeffizienten schwankten wild, nahmen absurde Werte an
- **Idee (1970):** Kleinen Strafterm λ zur Diagonale der Datenmatrix addieren

Was Ridge bewirkt:

- „Drückt" Koeffizienten Richtung Null
- Stabilisiert die Schätzungen
- Opfert etwas Bias für viel weniger Varianz

Damals: Skeptisch aufgenommen – „BLUE wird verletzt!"
Heute: Standardmethode bei Multikollinearität.

Ridge Regression: bewusst Bias in Kauf nehmen, um Stabilität zu gewinnen.

Zwei häufige Fehler über Multikollinearität

Fehler 1: „Hohe Korrelation bedeutet immer Multikollinearität."

- Ob es ein *Problem* ist, hängt ab von: Korrelationshöhe, Stichprobengröße, Analyseziel
- Bei $n = 100.000$ kann $r = 0{,}85$ tolerierbar sein
- Bei $n = 50$ wäre dieselbe Korrelation katastrophal
- VIF gibt ein besseres Bild als die bloße Korrelation

Fehler 2: „Einfach die Variable mit dem höchsten VIF entfernen."

- Gefährlich! Die Variable mit dem höchsten VIF kann die *kausal relevanteste* sein
- Beispiel: Wenn „monatl. Kosten" kausal auf Churn wirkt und „Vertragslaufzeit" nur ein Proxy ist – falsche Variable entfernt!
- Entscheidung muss auf inhaltlichem Wissen basieren, nicht nur auf VIF

Kausales Denken schützt vor statistischen Fehlern – mehr dazu in Abschnitt 6.

R-Code: VIF-Analyse und Korrelationsplot

```
# VIF berechnen
library(car)
vif_werte <- vif(model)
print(vif_werte)
#  alter  vertragslaufzeit  monatl_kosten  nutzung_gb
#  1.14   5.60              4.50           1.33

# Korrelationsmatrix visualisieren
library(corrplot)
cor_matrix <- cor(df[, c("alter", "vertragslaufzeit",
                         "monatl_kosten", "nutzung_gb")])
corrplot(cor_matrix, method = "color",
         type = "upper", addCoef.col = "black")

# Modell ohne redundante Variable
model_red <- lm(churn_score ~ alter +
                vertragslaufzeit + nutzung_gb, data = df)
vif(model_red)   # Alle VIF jetzt unter 2
```

Zwei Befehle reichen: `vif()` **für die Diagnose,** `corrplot()` **für die Visualisierung.**

Übungsaufgaben: Multikollinearität und VIF

Aufgabe 1: VIF von Hand berechnen
Hilfsregression von X_1 auf X_2 und X_3 ergibt $R_1^2 = 0{,}75$.
(a) Berechnen Sie VIF_1.
(b) Um welchen Faktor ist der SE von $\hat{\beta}_1$ aufgeblasen?
(c) Ist dies nach den Faustregeln problematisch?

Aufgabe 2: Symptome erkennen
Modell A: `churn ~ alter + nutzung_gb` $\rightarrow R^2 = 0{,}35$, beide signifikant.
Modell B: + Vertragslaufzeit + monatl_kosten $\rightarrow R^2 = 0{,}38$, *kein* Koeffizient signifikant.
(a) Welches Symptom zeigt sich? (b) Warum steigt R^2 trotzdem?

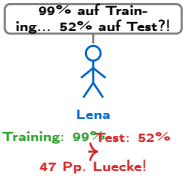
Aufgabe 3: Vorhersage vs. Interpretation
Warum schadet Multikollinearität der Interpretation, aber weniger der Vorhersage?

Lösungen werden in der Übung besprochen.

Lenas Schock: 99% Training, 52% Test

DataCo, Freitagmittag:

- Lena trainiert ihr Churn-Modell
- Trainingsdaten: **99% Genauigkeit**
- Sie praesentiert stolz dem Chef
- Chef fragt: "Und bei neuen Kunden?"
- Testdaten: **52% Genauigkeit**
- Kaum besser als eine Muenze werfen!



Was ist passiert? Das Modell hat die Trainingsdaten *Straße fuer Straße gespeichert* – inklusive Rauschen.

Die Luecke zwischen Training und Test ist das zentrale Warnsignal fuer Overfitting.

Entdeckungsfrage: Routen speichern vs. Netz verstehen

Denken Sie nach: Ein Navi kennt jede gespeicherte Route perfekt – jede Abbiegung, jede Einbahnstraße. Trotzdem versagt es bei der naechsten Umleitung.

Warum kennt das Navi nur alte Routen, obwohl es alles gespeichert hat?

Die Antwort enthuellt den wichtigsten Tradeoff in Machine Learning.

Die Navi-Analogie: Das starre Navi

Das starre Navi:

- Speichert *jede* Route Straße fuer Straße
- Lernt nicht die Logik, sondern den Weg
- "Hauptstr. → Bahnhofstr. → Marktplatz"
- Bekannte Strecken: perfekt

Bei einer Umleitung:

- Baustelle auf der Route: verloren
- Keine Alternative bekannt: ratlos
- Neue Strecke: Versagen

Lenas Modell:

- "Merkt sich" jede Trainingsbeobachtung
- Lernt auch das Rauschen mit
- Training: 99%
- Test: 52%

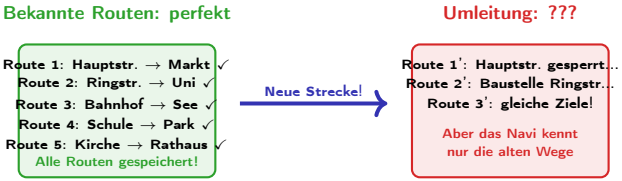
Der Unterschied:

- Routen speichern = Daten reproduzieren
- Netz verstehen = Muster erkennen

Overfitting: Ein Modell speichert die Trainingsdaten (inkl. Rauschen) statt das Muster zu erkennen.

Ein gutes Modell "versteht" das Straßennetz – es speichert nicht einzelne Routen.

Warum das beste Navi trotzdem versagt



Die Lektion fuer Modelle:

- Ein zu komplexes Modell "merkt sich" die Trainingsdaten mitsamt Rauschen
- Es kann perfekt reproduzieren, was es schon gesehen hat
- Aber es versagt bei auch nur leicht anderen Daten
- Der Kern: **Bias-Varianz-Tradeoff** – Routen speichern (zu viel Varianz) vs. zu stark vereinfachen (zu viel Bias)

Overfitting = zu viel Varianz. Underfitting = zu viel Bias.

Overfitting: Symptome und Ursachen

Overfitting: Modell zu komplex.

Symptome:

- Trainingsgenauigkeit verdaechtig hoch (>95%)
- Testgenauigkeit 20–40 Pp. darunter
- Koeffizienten sehr gross ($\beta_1 = 42$)
- Viele Parameter relativ zu n

Ursachen:

- Zu wenig Beobachtungen
- Zu viele Features
- Zu flexibles Modell
- Kein separates Testset

Underfitting: Modell zu einfach.

Symptome:

- Trainings- *und* Testfehler hoch
- Kein nennenswertes "Gap"
- Offensichtliche Strukturen ignoriert

Ursache:
Modell kann den wahren Zusammenhang nicht abbilden (z. B. linear fuer quadratisch).

Ziel:

Sweet Spot: Komplex genug fuer echte Muster, einfach genug gegen Rauschen.

Overfitting: grosse Luecke (Training gut, Test schlecht). Underfitting: beide schlecht.

Overfitting bekaempfen: Fuenf Strategien

Die fuenf Gegenmassnahmen:

1. **Mehr Daten** sammeln
2. **Feature Selection** – nur relevante Variablen
3. **Regularisierung** (Abschnitt 4)
4. **Kreuzvalidierung** (Abschnitt 5)
5. **Einfacheres Modell** waehlen

Faustregel:
Mindestens **10–20 Beobachtungen pro Praediktor** im Modell!

Lenas Szenarien bei DataCo:

Features	n/p	**Status**
$p = 4$	125	sicher
$p = 50$	10	kritisch
$p = 200$	2,5	gefaehrlich
$p = 1000$	0,5	unmoeglich*

*ohne Regularisierung bei $n = 500$

Das Verhaeltnis n/p ist ein schneller Indikator fuer das Overfitting-Risiko.

In-Sample vs. Out-of-Sample-Fehler

In-Sample-Fehler (Trainingsfehler): Wie gut passt das Modell auf die Trainingsdaten?
Out-of-Sample-Fehler (Testfehler): Wie gut bei *neuen*, ungesehenen Daten?

Trainingsfehler:

- Sinkt **monoton** mit Modellkomplexitaet
- Mehr Flexibilitaet = bessere Anpassung
- Optimistisch ueberschaetzt Qualitaet
- Lena: 99% → taeuschend gut

Testfehler:

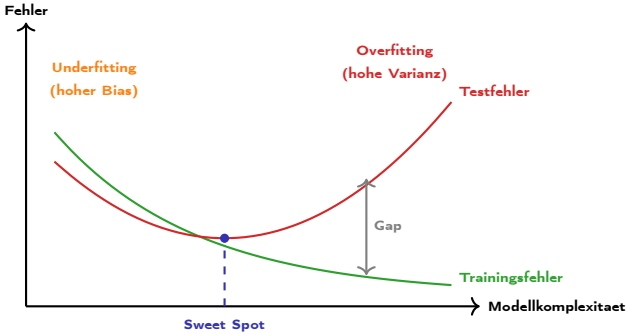
- Hat typisch eine **U-Form**
- Einfach: hoch (Underfitting)
- Sweet Spot: Minimum
- Zu komplex: steigt wieder (Overfitting)
- Lena: 52% → Realitaet

Diagnose ueber die Luecke:

- Kleine Luecke, beide schlecht → Underfitting
- Grosse Luecke (Training gut, Test schlecht) → Overfitting
- Lena: 99% − 52% = 47 Prozentpunkte → dramatisches Overfitting!

Was zaehlt, ist ausschliesslich die Out-of-Sample-Performance.

Die U-Kurve des Testfehlers



- Trainingsfehler: sinkt immer weiter – mehr Flexibilitaet heisst bessere Anpassung
- Testfehler: U-Form mit Minimum am Sweet Spot
- Die **Luecke** waechst bei Overfitting – Warnsignal!

Mehr Flexibilitaet heisst nicht automatisch bessere Vorhersagen auf neuen Daten.

Die Bias-Varianz-Zerlegung

Jeder Vorhersagefehler laesst sich in genau drei Quellen zerlegen:

$$E\left[(y - \hat{y})^2\right] = \underbrace{\text{Bias}^2}_{\text{Systematischer Fehler}} + \underbrace{\text{Varianz}}_{\text{Instabilitaet}} + \underbrace{\sigma^2}_{\text{Irreduzibler Fehler}}$$

Bias²
Wie weit liegt die *durchschnittliche* Vorhersage vom wahren Wert?

Modell zu einfach → hoher Bias

Varianz
Wie stark *schwankt* die Vorhersage bei verschiedenen Trainingssets?

Modell zu komplex → hohe Varianz

σ^2 (irreduzibel)
Grundrauschen in den Daten. Kein Modell kann das eliminieren.

Lena: Kuendigungen aus unvorhersehbaren Gruenden

Bias und Varianz sind gegenlaefig – das ist der Tradeoff.

Bias-Varianz: Lenas Zahlen

Lenas DataCo-Modell zerlegt:

Komponente	Wert	Beitrag zum Fehler
Bias	0,4	$0{,}4^2 = \mathbf{0{,}16}$
Varianz	0,3	**0,30**
σ (irreduzibel)	0,2	$0{,}2^2 = \mathbf{0{,}04}$
Gesamtfehler		**0,50**

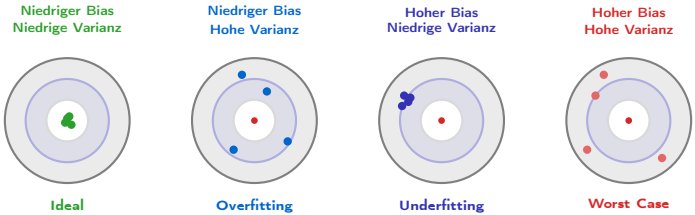
Was faellt auf?

- Die **Varianz** dominiert mit 60% des Gesamtfehlers!
- Der irreduzible Fehler traegt nur 0,04 bei (8%)
- Lenas bester Hebel: **Varianz reduzieren**

Wie? Einfacheres Modell, weniger Features, oder **Regularisierung** (Abschnitt 4).

$\text{Bias}^2 + \text{Varianz} + \sigma^2 = 0{,}16 + 0{,}30 + 0{,}04 = 0{,}50$ – die Haelfte ist Varianz!

Zielscheiben-Analogie: Vier Szenarien



- **Ideal:** Treffer eng beieinander und nahe am Zentrum
- **Overfitting:** Im Schnitt richtig, aber instabil (Lenas Polynom-15)
- **Underfitting:** Konsistent daneben (lineares Modell fuer quadr. Daten)

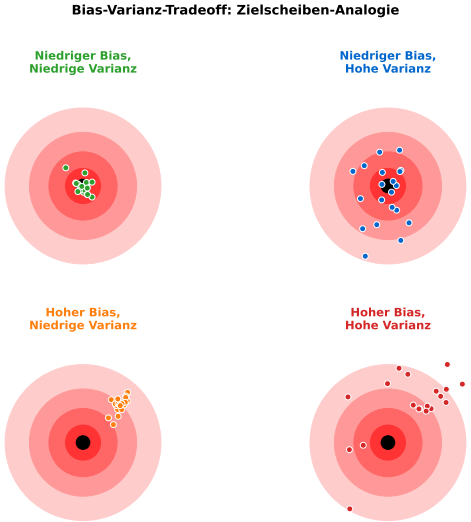
Jeder Wurf = eine Vorhersage auf einem anderen Trainingsdatensatz.

Zielscheiben: Was sie fuer Lena bedeuten

Zuordnung:

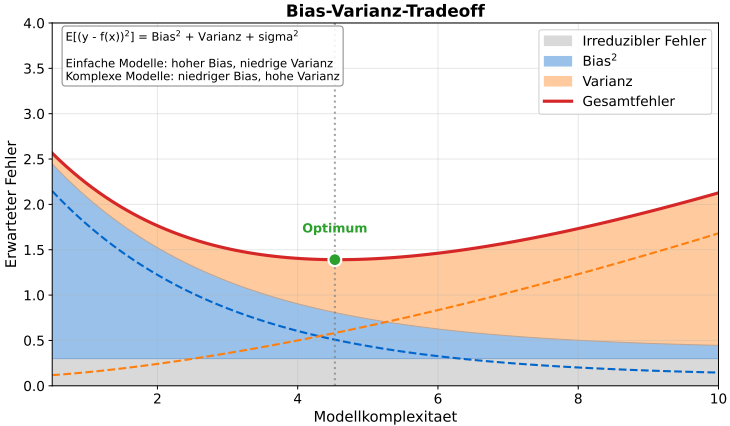
- Grad-15: niedrig Bias, hohe Varianz
- Linear fuer quadr.: hoher Bias, niedrige Varianz
- Grad-3: Sweet Spot!

Einfacher → weniger Varianz, mehr Bias
Komplexer → weniger Bias, mehr Varianz



Das optimale Modell minimiert Bias^2 + Varianz – nicht einen der beiden allein.

Bias-Varianz-Tradeoff



- Links: Underfitting (hoher Bias, niedrige Varianz)
- Rechts: Overfitting (niedriger Bias, hohe Varianz)
- **Sweet Spot:** minimaler Gesamtfehler = $\text{Bias}^2 + \text{Varianz} + \sigma^2$

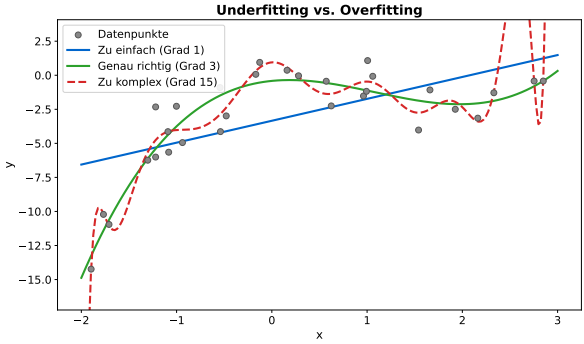
Das Ziel ist minimaler Gesamtfehler, nicht minimaler Bias oder minimale Varianz.

Polynomiale Regression: Grad 1 vs. 3 vs. 15

Wahrer Zusammenhang: $y = 2 + 0{,}5x + 0{,}2x^2 + \varepsilon$

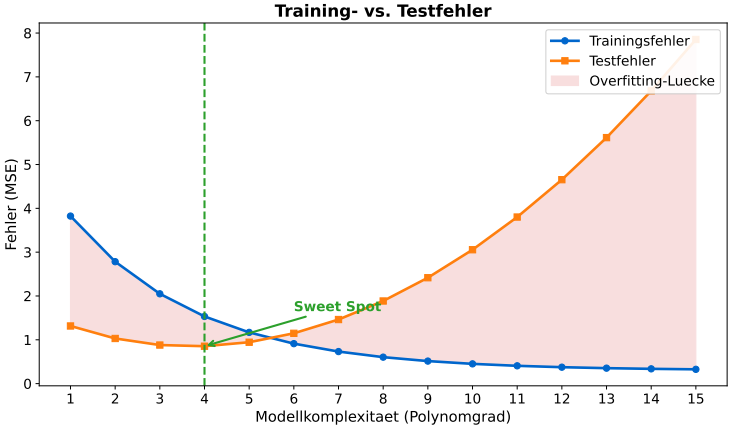
	Grad 1	Grad 3	Grad 15
Train-MSE	5,1	3,8	2,5
Test-MSE	5,3	4,1	28,7
Diagnose	Under	Sweet	Over
Bias	hoch	niedrig	niedrig
Varianz	niedrig	niedrig	enorm

Lektion: Mehr Flexibilitaet $\neq$ bessere Vorhersagen!



Der Testfehler hat ein U-foermiges Minimum – nicht der Trainingsfehler.

Training- vs. Testfehler: Die wachsende Luecke



- Trainingsfehler: sinkt monoton – Testfehler: U-Form mit Minimum am Sweet Spot
- Die **Luecke** zwischen beiden waechst bei Overfitting
- Lena: 99% − 52% = 47 Prozentpunkte Luecke = dramatisches Overfitting

Immer auf einem separaten Testset evaluieren und die Luecke beobachten!

Geschichte: Netflix Prize & Challenger

Netflix Prize (2009)

- $1 Mio. Preis fuer 10% Verbesserung
- Gewinner: Ensemble aus 100+ Modellen
- Ergebnis: 10,06% besser
- Pointe: Netflix nutzte es *nie* in Produktion!
- Zu komplex, zu langsam
- Einfacheres Modell (3 Algorithmen) fast genauso gut – 100× schneller

Challenger-Katastrophe (1986)

- Space Shuttle explodiert nach 73 Sek.
- Ursache: O-Ringe bei Kaelte sproede
- NASA analysierte *nur* Fluege mit Schaeden
- Selection Bias: Kein klarer Zusammenhang auf gefilterter Datenbasis
- Mit *allen* Fluegen: Zusammenhang offensichtlich unter 18°C

Fuer Lena: Nur gekuendigte Kunden analysieren = gleicher Fehler wie NASA!

Das theoretisch beste Modell ist nicht immer das praktisch nuetzlichste.

Drei verbreitete Irrtuemer

Irrtum 1: "Mehr Features = besseres Modell."
Nein. Jedes Feature braucht mehr Daten. Ab einem Punkt ueberwiegt die erhoehte Varianz. **Fluch der Dimensionalitaet:** In hohen Dimensionen wird der Datenraum so duenn besetzt, dass keine Methode Muster erkennt.

Irrtum 2: "Training-Genauigkeit = Modellqualitaet."
Nein. Was zaehlt, ist Out-of-Sample-Performance. Ein Modell mit 75% Training / 73% Test ist *besser* als eines mit 99% Training / 52% Test.

Irrtum 3: "Overfitting tritt nur bei kleinen Daten auf."
Nein. Entscheidend ist das Verhaeltnis n/p. Ein Datensatz mit 100.000 Beobachtungen und 50.000 Features ist genauso anfaellig wie einer mit 100 Beobachtungen und 50 Features.

Overfitting ist kein Problem der Datengroesse – es ist ein Problem des Verhaeltnisses n/p.

R-Code: Overfitting demonstrieren

```
set.seed(123)
x_train <- runif(50, 0, 10)
y_train <- 2 + 0.5*x_train + 0.2*x_train^2 + rnorm(50, 0, 2)
x_test  <- runif(100, 0, 10)
y_test  <- 2 + 0.5*x_test + 0.2*x_test^2 + rnorm(100, 0, 2)

train_mse <- test_mse <- numeric(15)
for (d in 1:15) {
  fit <- lm(y_train ~ poly(x_train, d, raw = TRUE))
  train_mse[d] <- mean((y_train - predict(fit))^2)
  test_mse[d]  <- mean((y_test - predict(fit,
                        newdata = data.frame(x_train = x_test)))^2)
}
cat("Optimaler Grad:", which.min(test_mse))
```

Ergebnis: Trainingsfehler sinkt monoton, Testfehler hat U-Form (Minimum bei Grad 2–3).

Trainings-MSE sinkt immer – nur der Testfehler zeigt den wahren Sweet Spot.

Uebung: Bias-Varianz-Tradeoff

Aufgabe 3.1: Lenas Kollegin zeigt ein Modell mit $R^2_{\text{Train}} = 0{,}98$ und $R^2_{\text{Test}} = 0{,}41$. Was ist das Problem? Nennen Sie drei Gegenmassnahmen.

Aufgabe 3.2: Fuer ein Modell gilt: Bias $= 0{,}6$, Varianz $= 0{,}1$, $\sigma = 0{,}3$.

a) Berechnen Sie den Gesamtfehler.

b) Welche Komponente dominiert?

c) Einfacher oder komplexer machen?

Aufgabe 3.3: Ordnen Sie den Zielscheiben-Quadranten zu:

a) Lineare Regression auf quadratischen Zusammenhang (viele Daten)

b) Polynom Grad 20 auf 15 Datenpunkte

c) Polynom Grad 3 auf quadratischen Zusammenhang (viele Daten)

Loesungen: 3.2a) $0{,}6^2 + 0{,}1 + 0{,}3^2 = 0{,}55$ **– Bias dominiert, komplexer machen!**

Zwischenfazit: Das starre Navi

Kernbotschaft: Das Ziel ist minimaler Gesamtfehler, nicht minimaler Bias oder minimale Varianz. Optimale Modellkomplexitaet balanciert beides.

Was Lena gelernt hat:

- 99% Training + 52% Test = Overfitting (Modell speichert Routen statt Muster)
- **Bias-Varianz-Zerlegung:** Fehler = Bias^2 + Varianz + σ^2
- Einfacher → mehr Bias, weniger Varianz
- Komplexer → weniger Bias, mehr Varianz
- **Faustregel:** Mindestens 10–20 Beobachtungen pro Praediktor

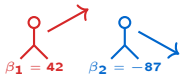
Naechster Schritt: Wie zaehmt man explodierende Koeffizienten?
→ **Abschnitt 4: Regularisierung**

Immer auf einem separaten Testset evaluieren und die Luecke beobachten.

Lenas wilde Koeffizienten

DataCo, Montagmorgen:
Lena schaut sich die Koeffizienten an:

Feature	$\hat{\beta}$ (OLS)
Vertragslaufzeit	42
Monatl. Kosten	−87
Beschwerden	63



Chaos!
Ohne Regularisierung

- Koeffizienten riesig und instabil
- Wechseln bei jedem Trainingsset die Vorzeichen
- Das Modell hat Rauschen statt Muster gespeichert

Lena braucht eine Leine fuer ihre wilden Koeffizienten.

Explodierende Koeffizienten sind das Symptom von Overfitting.

Entdeckungsfrage: Wie zaehmt man Koeffizienten?

Denken Sie nach: Was waere, wenn man ein "Budget" fuer die Koeffizientengroesse haette – wenn jeder zusaetzliche Betrag Kosten verursacht?

Ohne Budget: $\beta_1 = 42$, $\beta_2 = -87$ (alles erlaubt)
Mit Budget: Koeffizienten muessen "sparen" $\rightarrow$ kleine, stabile Werte

Regularisierung: Eine Technik, die grosse Koeffizienten **bestraft**, um Overfitting zu reduzieren und die Generalisierung zu verbessern.

Die Idee: Fuege zur Verlustfunktion einen **Strafterm** hinzu, der grosse Koeffizienten "teuer" macht. Das Modell muss zwei Ziele gleichzeitig optimieren:

1. Gute Datenanpassung (kleiner Fehler)
2. Kleine Koeffizienten (Einfachheit)

Regularisierung opfert bewusst etwas Bias, gewinnt dafuer Stabilitaet (weniger Varianz).

Die Leine: Hunde an der Leine = Regularisierung

Ohne Leine ($\lambda = 0$):

- Koeffizienten rennen in alle Richtungen
- β_1 springt auf 42, β_2 rast auf -87
- Passt sich an jedes Rauschen an

Mit Leine ($\lambda > 0$):

- Etwas Bewegungsfreiheit (nicht null)
- Aber Zurueckziehen bei zu grossem Ausscheren
- Je kuerzer die Leine (λ groesser), desto weniger Freiheit

Das λ-Spektrum:

$\lambda = 0$	Keine Leine (OLS)
λ klein	Lange Leine
λ mittel	Sweet Spot
λ gross	Kurze Leine
$\lambda \to \infty$	Alle $\beta \to 0$

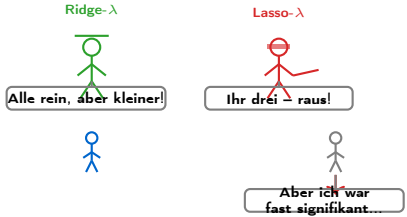
λ (Lambda): Steuert die Staerke der Bestrafung. Wird per Kreuzvalidierung gewaehlt.

Regularisierung = Leine fuer Koeffizienten. λ = Laenge der Leine.

Der Tuersteher: Ridge vs. Lasso

λ als Tuersteher vor dem Club:

- **Ridge** (hoeflicher Tuersteher): Laesst *alle* Koeffizienten rein, schrumpft aber jeden
- **Lasso** (strenger Tuersteher): Prueft jeden – wer unwichtig ist, wird *abgewiesen* (= Null gesetzt)
- Je groesser λ, desto strenger der Tuersteher



Ridge schrumpft alle Koeffizienten. Lasso setzt unwichtige auf exakt Null.

Die Strafidee: Allgemeine Formel

Kernidee in einem Satz: Fuege einen Strafterm hinzu, der grosse Koeffizienten bestraft.

$$\min_{\beta} \underbrace{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}_{\textbf{Anpassung (RSS)}} + \underbrace{\lambda \cdot \text{Strafe}(\beta)}_{\textbf{Komplexitaetskosten}}$$

Erster Term (RSS):

- Vertraute Residuenquadratsumme
- Wie gut passt das Modell?
- Allein: $\rightarrow$ OLS

Zweiter Term (Strafe):

- **Strafterm (Penalty):** Macht grosse β "teuer"
- λ steuert die Balance
- $\lambda = 0$: keine Strafe (OLS)
- $\lambda \rightarrow \infty$: alle $\beta \rightarrow 0$

Budget-Analogie: Ohne Budget kauft man alles (auch Rauschen). Mit Budget muss man priorisieren.

Zwei Ziele gleichzeitig: gute Datenanpassung UND kleine Koeffizienten.

Ridge Regression (L2-Strafe)

Ridge Regression: Regularisierung mit quadratischer Strafe.

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Eigenschaften:

- **Schrumpfung ohne Selektion:** Alle β_j werden kleiner, aber *keiner* wird exakt Null
- Alle Features bleiben im Modell – nur mit kleineren Koeffizienten
- **Stabilisierung bei Multikollinearitaet:** Korrelierte Features $\rightarrow$ instabile OLS-Koeffizienten. Ridge stabilisiert.
- **Geometrie:** L2-Strafe = kreisfoermige Nebenbedingung. Kreis hat keine Ecken $\rightarrow$ Koeffizienten treffen nie exakt eine Achse

Ridge-Analogie: "Gib ueberall etwas weniger aus" – kein Projekt wird komplett gestrichen.

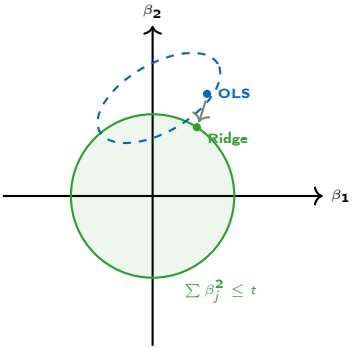
Ridge: Alle Koeffizienten bleiben, alle werden kleiner. Ideal bei vielen kleinen Effekten.

Ridge: Warum nie exakt Null?

Geometrische Intuition:

- OLS-Loesung: Ellipsen gleichen Fehlers
- Ridge-Nebenbedingung: **Kreis** ($\sum \beta_j^2 \leq t$)
- Loesung = Beruehrpunkt Ellipse–Kreis
- Kreis hat *keine Ecken*
- $\rightarrow$ Beruehrpunkt liegt fast nie auf einer Achse
- $\rightarrow$ Kein β_j wird exakt Null

Konsequenz: Ridge behaelt *alle* Features.



Kreisfoermige Nebenbedingung = keine Ecken = keine exakten Nullen.

Lasso Regression (L1-Strafe)

Lasso (Least Absolute Shrinkage and Selection Operator): Absolutwert-Strafe.

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Der entscheidende Unterschied zu Ridge:

- Setzt unwichtige Koeffizienten auf **exakt Null**
- Fuehrt *automatisch* Variablenselektion durch
- Erzeugt sparsame ("sparse") Modelle – ideal bei wenigen relevanten Features
- Verbessert Interpretierbarkeit: weniger β_j = einfacheres Modell

Nachteil: Bei stark korrelierten Features waehlt Lasso oft willkuerlich eines aus und setzt die anderen auf Null.
Lasso-Analogie: "Streich ganze Projekte" – an den Ecken wird mindestens ein Posten eliminiert.

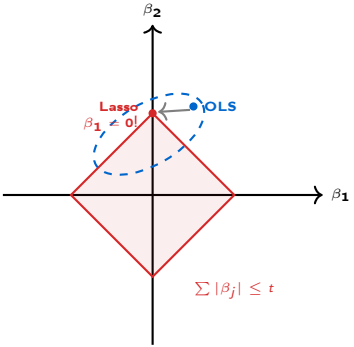
Lasso: Schrumpfung + Selektion. Ideal wenn nur wenige Features wirklich relevant sind.

Lasso: Warum exakt Null?

Geometrische Erklaerung:

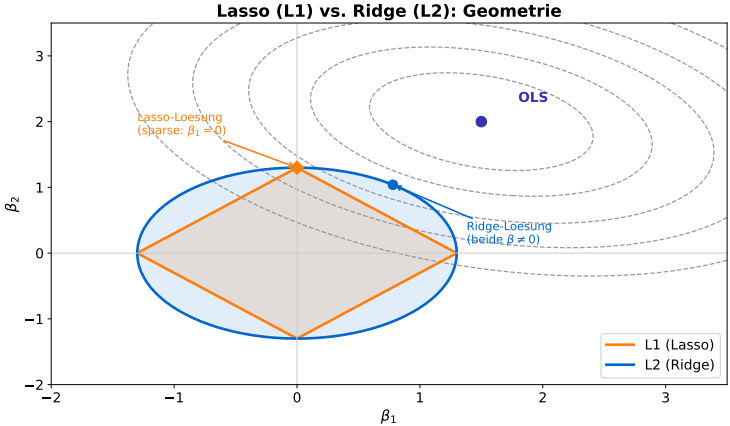
- Lasso-Nebenbedingung: **Diamant (Raute)** ($\sum |\beta_j| \leq t$)
- Diamant hat *Ecken*!
- An jeder Ecke: mindestens eine Koordinate = 0
- OLS-Ellipsen treffen den Diamant fast immer an einer Ecke
- $\rightarrow$ Koeffizienten werden exakt Null

Vergleich:
Ridge (Kreis) $\rightarrow$ keine Ecken $\rightarrow$ nie Null
Lasso (Diamant) $\rightarrow$ Ecken $\rightarrow$ exakte Nullen



Diamant-Ecken auf den Achsen = Variablenselektion. Das ist Lassos Superkraft.

Geometrie: Kreis vs. Diamant



Ridge (Kreis): glatt, Beruehrpunkt nie auf Achse $\rightarrow$ alle $\beta_j \neq 0$. **Lasso** (Diamant): Ecken auf Achsen $\rightarrow$ Beruehrpunkt an Ecke $\rightarrow$ manche $\beta_j = 0$.

L2 = Kreis = keine Selektion. L1 = Diamant = automatische Variablenselektion.

Elastic Net: Das Beste aus beiden Welten

Elastic Net: Kombination aus L1 (Lasso) und L2 (Ridge).

$$\min_{\beta} \sum (y_i - \hat{y}_i)^2 + \lambda \Big[\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \Big]$$

Der Mischungsparameter α:

- $\alpha = 1$: reines Lasso (nur L1)
- $\alpha = 0$: reines Ridge (nur L2)
- $\alpha = 0{,}5$: 50/50-Mischung beider Strafen

Wann Elastic Net?

- Korrelierte Feature-Gruppen $\rightarrow$ waehlt Gruppen gemeinsam aus (statt willkuerlich eine)
- Unsicher ob Ridge oder Lasso besser $\rightarrow$ sicherste Wahl
- In der Praxis: `cv.glmnet` optimiert λ per CV, α wird oft auf 0,5 gesetzt

Elastic Net verbindet die Selektion von Lasso mit der Stabilitaet von Ridge.

Vergleich: Ridge vs. Lasso vs. Elastic Net

Eigenschaft	Ridge	Lasso	Elastic Net
Strafterm	$\lambda \sum \beta_j^2$	$\lambda \sum \lvert\beta_j\rvert$	Mischung
Koeff. auf 0?	Nein	Ja	Ja
Variablenselektion?	Nein	Ja	Ja
Multikollinearitaet?	Sehr gut	Problematisch	Gut
Viele kleine Effekte?	Gut	Weniger gut	Gut
Wenige grosse Effekte?	Weniger gut	Sehr gut	Gut
`glmnet` α	$\alpha = 0$	$\alpha = 1$	$\alpha \in (0, 1)$

Kurzregel:

- **Ridge:** Viele Features, alle vermutlich relevant, korreliert
- **Lasso:** Wenige Features relevant, Rest ist Rauschen
- **Elastic Net:** Unsicher? Korrelierte Gruppen? $\rightarrow$ Elastic Net

Im Zweifel: Elastic Net mit $\alpha = 0{,}5$ und CV fuer λ – die sichere Wahl.

Lambda waehlen: Kreuzvalidierung mit cv.glmnet

Zentrale Frage: Welchen Wert soll λ haben?
Antwort: Kreuzvalidierung – niemals per Hand!

So funktioniert `cv.glmnet`**:**

1. Daten in k Folds aufteilen
2. Fuer viele λ-Werte: Trainiere auf $k-1$ Folds, teste auf dem letzten
3. Mittleren CV-Fehler pro λ berechnen
4. Waehle optimales λ

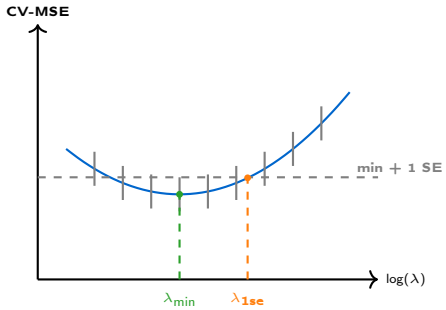
Zwei wichtige λ-Werte:

lambda.min:
Kleinstes mittleres MSE. Optimiert auf Vorhersagegenauigkeit.

lambda.1se:
Groesstes λ innerhalb 1 Standardfehler des Minimums. Einfacheres Modell, kaum schlechtere Vorhersage.

lambda.1se folgt Occam's Razor: das einfachste Modell, das fast genauso gut ist.

lambda.min vs. lambda.1se



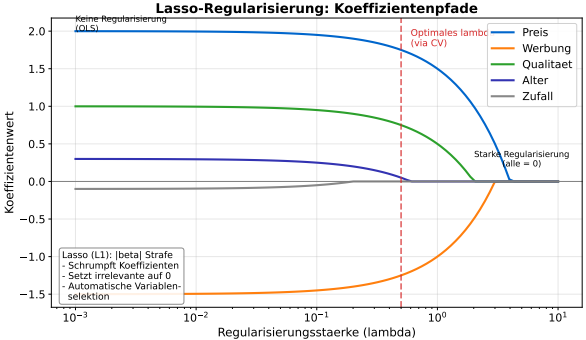
Lenas DataCo-Ergebnis:

	λ_{min}	$\lambda_{\mathbf{1se}}$
Wert	0,023	0,089
$\beta \neq 0$	9/12	5/12
CV-MSE	0,31	0,34

Fuer den Chef: $\lambda_{\mathbf{1se}}$ – einfacheres Modell mit nur 5 Features, kaum schlechtere Vorhersage.
Preis: Etwas hoehere MSE (0,03 mehr).

In der Praxis wird oft lambda.1se bevorzugt – Sparsamkeit vor Perfektion.

Koeffizientenpfade: Wie Koeffizienten schrumpfen



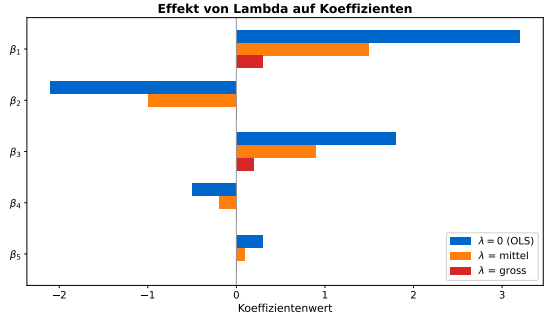
Was zeigt die Grafik?

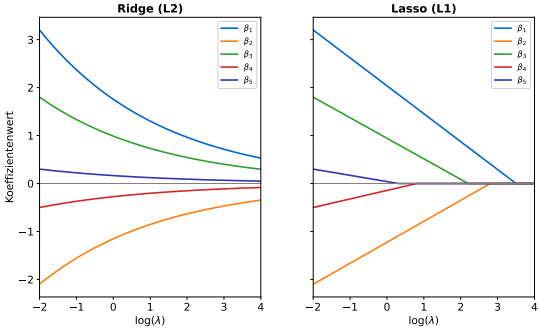
- Jede Linie = ein Koeffizient
- X-Achse: steigendes λ
- Koeffizienten schrumpfen mit λ
- Bei Lasso: fallen nacheinander auf exakt Null
- Weniger wichtige zuerst

Diagnosewerkzeug: Welche Features ueberleben bei welcher Regularisierungsstaerke?

Koeffizientenpfade zeigen die Rangfolge der Feature-Wichtigkeit.

Effekt von Lambda: Ridge vs. Lasso





Ridge-Pfade:

- Sanfte, stetige Schrumpfung
- Kein Koeffizient wird exakt Null
- Alle Features bleiben

Lasso-Pfade:

- Koeffizienten springen auf Null
- Weniger wichtige zuerst
- Automatische Selektion

Ridge: stetige Schrumpfung. Lasso: scharfe Selektion an den Knicken.

Lenas glmnet-Pipeline bei DataCo

500 Kunden, 12 Features, Lasso mit Kreuzvalidierung:

1. **Vorbereitung:** Modellmatrix mit `model.matrix()`, Zielvariable extrahieren
2. **CV:** `cv.glmnet(x, y, alpha = 1, nfolds = 10)` $\rightarrow$ `lambda.min` $= 0{,}023$
3. **Ergebnis:** 3 von 12 Koeffizienten auf exakt Null: PLZ, Browser-Typ, Anmelde-Wochentag
4. **Wichtigste Praediktoren:**

Feature	**Lasso** $\hat{\beta}$
Beschwerden	$+0{,}31$
Nutzungshaeufigkeit	$-0{,}28$
Vertragslaufzeit	$-0{,}19$

Ohne Lasso: $\beta_{\text{PLZ}} = 3{,}2$, $\beta_{\text{Browser}} = -7{,}8$ – absurd!
Mit Lasso: Schlankes, interpretierbares Modell mit besserer Out-of-Sample-Performance.

Lasso identifiziert automatisch irrelevante Features und entfernt sie.

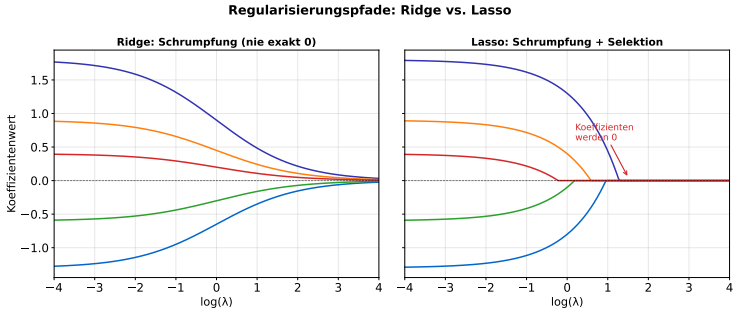
Vorher/Nachher: OLS vs. Lasso

Feature	OLS $\hat{\beta}$	Lasso $\hat{\beta}$	Status
Alter	0,02	0,01	geschrumpft
Vertragslaufzeit	−0,25	−0,19	geschrumpft
Monatl. Kosten	0,18	0,12	geschrumpft
Beschwerden	0,35	0,31	geschrumpft
Premium-Status	−0,42	−0,24	geschrumpft
Nutzungshaeufigkeit	−0,33	−0,28	geschrumpft
Support-Kontakte	0,21	0,15	geschrumpft
Online-Aktivitaet	−0,15	−0,08	geschrumpft
Vertragswert	0,11	0,04	geschrumpft
PLZ	3,20	0,00	entfernt
Browser-Typ	−7,80	0,00	entfernt
Anmelde-Wochentag	1,50	0,00	entfernt

OLS-Koeffizienten: teilweise absurd gross und instabil. Lasso: vernuenftige Groessenordnungen.

Lasso schrumpft alle Koeffizienten und entfernt drei irrelevante Features komplett.

Ridge vs. Lasso: Koeffizientenpfade im Vergleich



- **Ridge** (links): Stetige Schrumpfung aller Koeffizienten gegen Null
- **Lasso** (rechts): Koeffizienten fallen nacheinander auf exakt Null – Selektion!
- Feature, das am laengsten ueberlebt = wichtigstes Feature

Koeffizientenpfade sind ein wertvolles Diagnosewerkzeug in der Regularisierung.

Geschichte: Hoerl 1970 – Tibshirani 1996 – Zou 2005

1970: Ridge
Arthur Hoerl (Chemiker, DuPont): Korrelierte Messdaten → instabile OLS-Koeffizienten.
Idee: L2-Strafe schrumpft Koeffizienten.
Name: "Grat" (Ridge) in der Matrix $X^T X$.

1996: Lasso
Robert Tibshirani (Stanford): L1 statt L2-Strafe.
Ueberraschung: Diamant-Ecken → exakte Nullen → automatische Variablenselektion.
"Least Absolute Shrinkage and Selection Operator"

2005: Elastic Net
Zou & Hastie (Stanford): Lasso problematisch bei korrelierten Features.
Loesung: L1 + L2 kombinieren. Selektion + Stabilitaet.
Heute alle in `glmnet`.

35 Jahre Entwicklung – vom Chemieproblem zur Standardmethode im Machine Learning.

Alle drei Methoden sind heute in der R-Bibliothek glmnet vereint.

Drei verbreitete Irrtuemer

Irrtum 1: "Regularisierung verbessert immer das Modell."
Nein! Hilft nur bei Overfitting (zu viel Varianz). Bei Underfitting wuerden die Koeffizienten *weiter* schrumpfen $\rightarrow$ noch schlechter. Regularisierung ist ein Werkzeug gegen Varianz, nicht gegen Bias.

Irrtum 2: "Lasso ist immer besser als Ridge."
Nein! Bei vielen kleinen Effekten schneidet Ridge besser ab. Bei korrelierten Features waehlt Lasso willkuerlich eines aus – Ridge schrumpft alle gleichmaessig. Es kommt auf die *Datenstruktur* an.

Irrtum 3: "Lambda sollte so gross wie moeglich sein."
Nein! Zu grosses λ toetet alles Signal: alle $\beta \rightarrow 0 \rightarrow$ Vorhersage $\approx$ Mittelwert. Optimales λ per Kreuzvalidierung – nicht willkuerlich.

Regularisierung ist kein Allheilmittel – sie hilft gezielt gegen zu viel Varianz.

R-Code: glmnet-Pipeline (Daten & Lasso)

```
library(glmnet)
set.seed(42)
n <- 200; p <- 20
X <- matrix(rnorm(n * p), n, p)
colnames(X) <- paste0("x", 1:p)
# Wahre Koeff.: nur x1-x5 haben Effekt
true_beta <- c(3, -2, 1.5, -1, 0.5, rep(0, 15))
y <- X %*% true_beta + rnorm(n, 0, 2)

# Lasso mit 10-Fold CV
cv_lasso <- cv.glmnet(X, y, alpha = 1, nfolds = 10)
cat("lambda.min:", round(cv_lasso$lambda.min, 4))
cat("lambda.1se:", round(cv_lasso$lambda.1se, 4))
coef(cv_lasso, s = "lambda.min")
```

Ergebnis: Lasso erkennt, dass nur x_1–x_5 echte Effekte haben, und setzt die meisten der 15 Rausch-Features auf Null.

---

cv.glmnet waehlt automatisch das beste Lambda per Kreuzvalidierung.

R-Code: Ridge & Elastic Net Vergleich

```
# Ridge zum Vergleich (alpha = 0)
cv_ridge <- cv.glmnet(X, y, alpha = 0, nfolds = 10)
coef_ridge <- coef(cv_ridge, s = "lambda.min")
cat("Ridge Null-Koeff.:",
    sum(coef_ridge[-1] == 0), "von", p)  # -> 0

# Elastic Net (alpha = 0.5)
cv_enet <- cv.glmnet(X, y, alpha = 0.5, nfolds = 10)
coef_enet <- coef(cv_enet, s = "lambda.min")
cat("Elastic Net Null-Koeff.:",
    sum(coef_enet[-1] == 0), "von", p)

# Koeffizientenpfade plotten
lasso_fit <- glmnet(X, y, alpha = 1)
plot(lasso_fit, xvar = "lambda", label = TRUE)
title("Lasso: Koeffizientenpfade")
```

Vergleich: Ridge behaelt alle 20 Features ($\neq 0$). Elastic Net liegt dazwischen.

Ridge: 0 Nullen. Lasso: viele Nullen. Elastic Net: dazwischen.

Uebung: Regularisierung

Aufgabe 4.1: Erklaeren Sie, warum L1 (Lasso) Koeffizienten auf exakt Null setzt, L2 (Ridge) aber nicht. Nutzen Sie die Diamant-vs.-Kreis-Geometrie.

Aufgabe 4.2: Was passiert mit den Lasso-Koeffizienten, wenn λ von 0 auf einen sehr grossen Wert steigt? Welches Feature "ueberlebt" am laengsten?

Aufgabe 4.3: Ridge, Lasso, oder Elastic Net?

a) 500 Kunden, 8 Features, alle fachlich begruendet

b) 500 Kunden, 200 Features, die meisten irrelevant

c) 500 Kunden, 15 Features, davon 5 Gruppen mit je 3 korrelierten

Loesungen: a) Ridge – alle relevant, alle behalten. b) Lasso – Selektion! c) Elastic Net – korrelierte Gruppen gemeinsam.

Die richtige Methode haengt von der Datenstruktur ab, nicht von persoenlicher Praeferenz.

Zwischenfazit: Die Leine fuer wilde Koeffizienten

Kernbotschaft: Regularisierung opfert bewusst etwas Bias (Koeffizienten Richtung Null "verzerrt"), gewinnt dafuer deutlich an Stabilitaet (weniger Varianz). λ wird immer per CV gewaehlt.

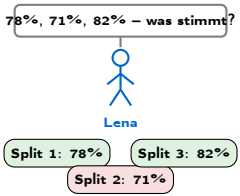
Was Lena gelernt hat:

- **Ridge:** Schrumpft alle, entfernt keine $\rightarrow$ viele kleine Effekte
- **Lasso:** Schrumpft *und* selektiert $\rightarrow$ wenige grosse Effekte
- **Elastic Net:** Kombiniert beides $\rightarrow$ korrelierte Features
- λ_{min}: bester CV-Fehler. $\lambda_{\texttt{1se}}$: einfacheres Modell, fast genauso gut
- Koeffizientenpfade als Diagnosewerkzeug

Naechster Schritt: Wie teilt man Daten fair auf?
$\rightarrow$ **Abschnitt 5: Kreuzvalidierung**

Ridge schrumpft alle, Lasso selektiert, Elastic Net kombiniert. Lambda per CV – nie per Hand.

Drei Splits, drei Zahlen – welche stimmt?



Lena hat ihr DataCo-Churn-Modell auf einem Testset geprüft:

- Split 1: Accuracy = **78 %**
- Split 2 (neu gemischt): Accuracy = **71 %**
- Split 3 (noch mal gemischt): Accuracy = **82 %**

Problem: Drei verschiedene Splits, drei verschiedene Zahlen. Welche Genauigkeit stimmt jetzt?

Ein einzelner Train/Test-Split ist wie eine Klausur mit nur einer Aufgabe – das Ergebnis hängt vom Zufall ab.

Warum reicht ein Split nicht?

Entdeckungsfrage: Stellen Sie sich vor, Sie bewerten eine Klausur mit nur einer einzigen Aufgabe. Könnte das Ergebnis zufällig besonders gut oder schlecht ausfallen?

Train/Test-Split: Einmalige Aufteilung der Daten in Trainings- und Testmenge (typisch 80/20).

- Der **Zufall** bestimmt, welche Punkte im Testset landen
- Dieses zufällige Testset kann die Performance nach oben *oder* unten verzerren
- Lena: 78%, 71%, 82% – Spanne von 11 Prozentpunkten!
- Wir brauchen eine Methode, die **alle** Datenpunkte zum Testen nutzt

Lösung: Die Daten systematisch rotieren – jeder Punkt wird einmal getestet.

Ein Split = eine Aufgabe. Kreuzvalidierung = die ganze Klausur.

Rotierende Richter

- Fünf Richter sitzen im Gericht
- In jeder Runde urteilt **ein** Richter, die anderen beobachten
- Nach 5 Runden hat **jeder** einmal geurteilt
- Das Mittel der 5 Urteile ist fairer als ein einzelnes Urteil

Übertragung: Jeder **Fold** (Datenteil) übernimmt einmal die Rolle des Richters (Testset). Am Ende hat jeder Datenpunkt **genau einmal** als Testpunkt gedient.

Die persönliche Tendenz eines Richters mittelt sich heraus – die Varianz der Schätzung sinkt.

K-Fold Cross-Validation – Der Algorithmus

K-Fold Cross-Validation: Daten in K gleich große, nicht überlappende Teile (Folds) aufteilen. Jeder Fold dient einmal als Testset.

1. **Mische** die Daten zufällig und teile sie in K gleich große Folds auf
2. Für $k = 1, 2, \ldots, K$: Trainiere auf Folds $\{1, \ldots, K\} \setminus \{k\}$, teste auf Fold k, speichere Fehler e_k
3. Berechne den CV-Fehler als Durchschnitt: $\mathrm{CV}(K) = \frac{1}{K} \sum_{k=1}^{K} e_k$

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Runde 1	**Test**	Train	Train	Train	Train
Runde 2	Train	**Test**	Train	Train	Train
Runde 3	Train	Train	**Test**	Train	Train
Runde 4	Train	Train	Train	**Test**	Train
Runde 5	Train	Train	Train	Train	**Test**

Standard: $K = 5$ oder $K = 10$. Jeder Datenpunkt ist genau einmal im Testset.

Warum funktioniert K-Fold?

Fold: Einer der K Teilmengen mit je n/K Beobachtungen.

- Beim einfachen Split: 20 % bleiben **immer** im Test, 80 % **immer** im Training
- Bei 5-Fold CV: Jeder Block ist **einmal** Test und **viermal** Training
- Kein Punkt wird doppelt getestet, keiner wird vergessen
- Die Schätzung hängt nicht von einem einzigen glücklichen Split ab

Lenas DataCo-Beispiel ($n = 500$, $K = 5$):

- Jeder Fold enthält 100 Kunden
- In jedem Durchlauf: Training auf 400, Test auf 100
- Nach 5 Durchläufen: 5 Accuracy-Werte $\rightarrow$ stabiler Mittelwert

K-Fold nutzt alle Daten sowohl fürs Training als auch fürs Testen – das macht die Schätzung robust.

Wie viele Folds? $K = 5$ oder $K = 10$?

Bias-Varianz-Tradeoff bei K:

- **Kleines** K (z. B. 5): Weniger Rechenaufwand, aber etwas mehr Bias (weniger Trainingsdaten pro Fold)
- **Großes** K (z. B. 10): Weniger Bias, aber mehr Varianz und Rechenaufwand
- $K = n$ **(LOOCV):** Minimaler Bias, maximale Varianz

Empirische Empfehlungen:

- Breiman & Spector (1992): $K = 10$ bester Kompromiss
- Kohavi (1995): *Stratified 10-Fold* als Standard
- Praxis: $K = 5$ bei kleinen Datensätzen, $K = 10$ sonst

Lena wählt $K = 5$ – bei $n = 500$ ist der Unterschied zu $K = 10$ marginal.

$K = 10$ ist der allgemeine Standard. $K = 5$ ist schneller und bei kleinen Datensätzen gleichwertig.

CV-Fehler und Standardabweichung

$$\widehat{\mathrm{CV}} = \frac{1}{K}\sum_{k=1}^{K} e_k, \qquad \mathrm{SD}_{\mathrm{CV}} = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}(e_k - \widehat{\mathrm{CV}})^2}$$

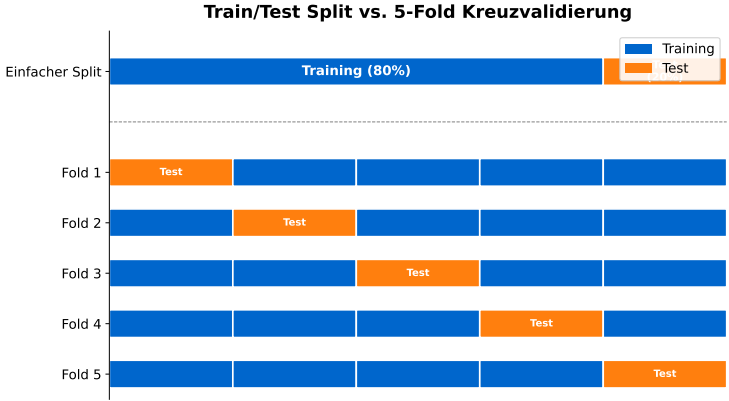
Symbol für Symbol:

- K – Anzahl der Folds (z. B. 5 oder 10)
- e_k – Testfehler (Accuracy, MSE oder AUC) auf Fold k
- $\widehat{\mathrm{CV}}$ – gemittelter CV-Fehler über alle Folds
- $\mathrm{SD}_{\mathrm{CV}}$ – Standardabweichung der Fold-Fehler; misst die **Stabilität**

Faustregel: Kleine $\mathrm{SD}_{\mathrm{CV}}$ → Modell performt auf allen Folds ähnlich → **robust**.

Der CV-Fehler ist unsere beste Schätzung für die Out-of-Sample Performance.

Train/Test-Split vs. K-Fold – Visualisierung



Links: Ein einzelner Split. Rechts: K-Fold Rotation – jeder Block wechselt die Rolle.

LOOCV – Leave-One-Out

LOOCV: Spezialfall mit $K = n$. Jeder einzelne Datenpunkt dient einmal als Testset.

Vorteile:

- Minimaler Bias – fast der gesamte Datensatz trainiert
- Nützlich bei **sehr kleinen** Datensätzen ($n < 50$)
- Bei linearer Regression: geschlossene Formel (Hat-Matrix)

Nachteile:

- **Hohe Varianz**: Die n Trainingsmengen unterscheiden sich nur um einen Punkt $\rightarrow$ Fehler korrelieren stark
- **Rechenintensiv**: n Modelle trainieren
- Lena: $n = 500 \rightarrow 500$ Modelle!

Empfehlung: LOOCV nur bei sehr kleinen Datensätzen. Sonst 10-Fold bevorzugen.

LOOCV hat minimalen Bias, aber die hohe Varianz und der Rechenaufwand sprechen meist für K-Fold.

Stratified K-Fold

Stratified K-Fold: Klassenverteilung wird in jedem Fold beibehalten. Unverzichtbar bei unbalancierten Daten.

Lenas Problem: Nur 15 % der DataCo-Kunden sind Kündiger.

Ohne Stratifizierung:

- Fold 3 enthält zufällig nur 5 % Kündiger
- Evaluation auf diesem Fold wenig aussagekräftig
- Große Varianz zwischen den Folds

Mit Stratifizierung:

- Jeder Fold enthält exakt 15 % Kündiger
- Alle Folds sind repräsentativ
- `caret` und `scikit-learn` nutzen dies standardmäßig

Merke: Bei Klassifikation immer Stratified K-Fold verwenden!

Moderne Software verwendet Stratified K-Fold automatisch für Klassifikationsprobleme.

Nested CV – Hyperparameter fair tunen

Nested CV: Doppelte Schleife. Äußere CV schätzt Performance, innere CV wählt Hyperparameter.

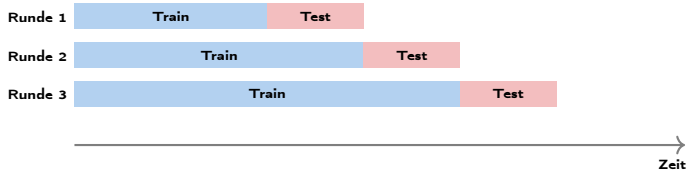
1. **Äußere Schleife** ($K_{\text{aussen}} = 5$): Teile Daten in äußeren Train/Test-Teil
2. **Innere Schleife** ($K_{\text{innen}} = 5$): Auf dem äußeren Trainingsteil CV durchführen, besten Hyperparameter wählen
3. Finales Modell mit bestem Hyperparameter auf äußerem Trainingsteil trainieren
4. Auf äußerem Testteil evaluieren. Für alle äußeren Folds wiederholen und mitteln.

Ohne Nested CV: Hyperparameter auf denselben Daten gewählt, auf denen auch die Performance bewertet wird → **optimistischer Bias**.

Nested CV verhindert, dass die Modellselektion Information über die Testdaten "sieht".

Time Series CV – Die Zeit respektieren

Time Series CV: Nur vergangene Daten zum Training, zukünftige zum Testen. Respektiert die zeitliche Ordnung.



Data Leakage: Verwendung von Informationen im Training, die zur Testzeit nicht verfügbar wären.

- Zufällige Fold-Zuteilung bei Zeitreihen = verboten
- Modell würde die "Zukunft sehen" → unrealistisch gute Performance

Time Series CV simuliert die reale Situation: Prognosen gelten nur für die Zukunft.

CV-Varianten im Überblick

Variante	K	Bias / Varianz	Wann?
K-Fold	5 oder 10	Guter Kompromiss	Standardwahl
LOOCV	n	Niedrig / Hoch	Sehr kleine Datensätze
Stratified	5 oder 10	Wie K-Fold	Unbalancierte Klassen
Nested	aussen + innen	Unverzerrt	Hyperparameter-Tuning
Time Series	variabel	Kontextabhängig	Zeitlich geordnete Daten

Entscheidungsregel:

- Klassifikation mit unbalancierten Klassen? $\rightarrow$ Stratified
- Zeitreihen? $\rightarrow$ Time Series CV
- Hyperparameter-Tuning? $\rightarrow$ Nested
- Sonst: $K = 10$ (oder $K = 5$ bei kleinen Daten)

Die richtige CV-Strategie hängt vom Datentyp und der Fragestellung ab.

CV + Regularisierung: Das Dream-Team

Regularisierung hat einen Hyperparameter λ – und der muss **auf den Daten** gewählt werden.

`cv.glmnet`: R-Funktion, die automatisch K-Fold CV durchführt und das optimale λ wählt.

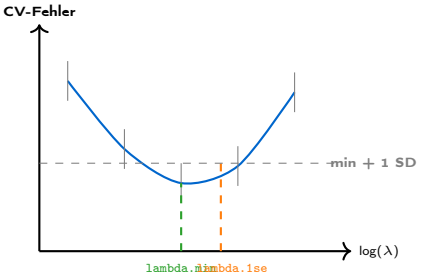
Pipeline:

1. Erzeuge 100 λ-Werte auf logarithmischer Skala
2. Für jedes λ: K-Fold CV $\rightarrow$ mittlerer CV-Fehler
3. Wähle `lambda.min` (kleinster Fehler) oder `lambda.1se` (sparsamstes Modell innerhalb 1 SD)
4. Finales Modell auf allen Trainingsdaten trainieren

`cv.glmnet` (innere CV) $\longrightarrow$ optimales λ $\longrightarrow$ finales Modell $\longrightarrow$ Holdout-Evaluation

Das Holdout-Set war nie an Modellselektion oder Tuning beteiligt – ehrliche Leistungsbewertung.

lambda.min vs. lambda.1se



Zwei Kandidaten:

- `lambda.min`: λ mit kleinstem CV-Fehler – beste Vorhersage
- `lambda.1se`: Sparsamstes Modell innerhalb 1 SD des Minimums – bessere Interpretierbarkeit

Empfehlung:

- Vorhersage wichtig? $\rightarrow$ `lambda.min`
- Interpretierbarkeit wichtig? $\rightarrow$ `lambda.1se`

Die 1-SE-Regel: Wähle das sparsamste Modell, das fast genauso gut ist wie das beste.

Lenas 5-Fold CV bei DataCo

Lena trainiert ein logistisches Lasso-Modell auf 5 Folds:

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	**Mittel**
Accuracy	0.74	0.78	0.77	0.73	0.78	**0.760**
AUC	0.81	0.84	0.83	0.80	0.82	**0.820**

Ergebnis: Mean Accuracy = 0.76 (SD = 0.024), Mean AUC = 0.82 (SD = 0.015).

- Niedrige SD → Modell performt **stabil** über alle Folds
- Lena berichtet dem Chef: Accuracy liegt bei ca. **76 %**
- Nicht bei den 78 % des glücklichen Splits – und nicht bei den 71 % des unglücklichen

Der CV-Mittelwert ist eine viel ehrlichere Schätzung als ein einzelner Split.

Modellvergleich via CV

Lena vergleicht drei Modelle mit denselben 5 Folds:

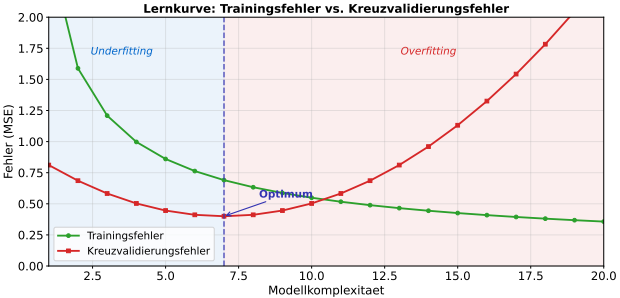
Modell	CV-Accuracy (SD)	CV-AUC (SD)
OLS (alle Features)	0.72 (0.04)	0.78 (0.03)
Ridge (λ_{min})	0.75 (0.03)	0.81 (0.02)
Lasso (λ_{min})	**0.76 (0.02)**	**0.82 (0.02)**

Lasso gewinnt:

- Höchste Accuracy und AUC
- **Kleinste Streuung** – robustestes Modell
- Automatische Feature-Selektion: 3 von 12 Prädiktoren auf Null gesetzt
- Weniger Varianz durch weniger irrelevante Features

CV-basierter Modellvergleich: Nicht nur den Mittelwert, sondern auch die Streuung beachten!

Lernkurve – Wann hilft mehr Komplexität?

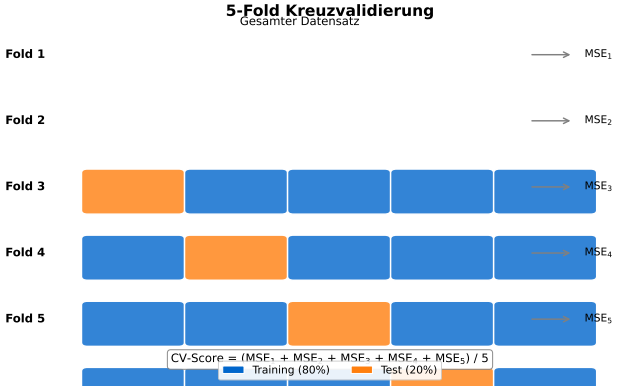


- **Trainingsfehler** sinkt mit steigender Komplexität – das Modell passt sich immer besser an
- **CV-Fehler** sinkt erst, steigt dann wieder – Overfitting!
- Der **Sweet Spot** liegt dort, wo der CV-Fehler minimal ist

---

Die Lernkurve zeigt den Bias-Varianz-Tradeoff: Zu einfach = Underfitting, zu komplex = Overfitting.

Kreuzvalidierung – Visualisierung



K-Fold CV: Jeder Datenpunkt ist genau einmal im Testset, $K-1$ Mal im Training.

Die Geburt der Kreuzvalidierung

1974 – Mervyn Stone

- Formalisiert die Idee, Daten systematisch aufzuteilen
- Erkennt: Gute Trainingsperformance $\neq$ gute Vorhersage

1975 – Seymour Geisser

- Nennt das Konzept "predictive sample reuse"
- Legt theoretische Grundlagen für LOOCV

1992 – Breiman & Spector

- Umfangreiche empirische Studie
- Ergebnis: $K = 10$ bester Kompromiss

1995 – Ron Kohavi

- Einflussreichste Vergleichsstudie
- Empfehlung: *Stratified 10-Fold* als Standard
- Hat sich bis heute gehalten

Von der theoretischen Idee (1974) zum industriellen Standard (1995) – in nur 20 Jahren.

Drei häufige CV-Irrtümer

Irrtum 1: "LOOCV ist immer am besten, weil der Bias minimal ist."

- Stimmt für den Bias, aber nicht für die Gesamtqualität
- Hohe Varianz: Trainingsmengen fast identisch $\rightarrow$ Fehler korrelieren
- 10-Fold ist in der Praxis fast immer besser

Irrtum 2: "CV ersetzt ein Holdout-Testset."

- CV = Werkzeug für **Modellselektion**
- Finale Leistungsbewertung: separates Holdout-Set, das **nie** für Training oder Modellauswahl verwendet wurde

Irrtum 3: "Mehr Folds sind immer besser."

- Grenznutzen nimmt ab: Zwischen $K = 10$ und $K = 20$ kaum Unterschied
- Rechenaufwand steigt linear mit K

Praxis-Standard: $K = 10$ (oder $K = 5$ bei Zeitdruck). LOOCV nur bei $n < 50$.

R-Code: 5-Fold CV mit caret

```
library(caret)

ctrl <- trainControl(
  method = "cv", number = 5,
  classProbs = TRUE,
  summaryFunction = twoClassSummary
)

model <- train(
  churn ~ ., data = dataco,
  method = "glm", family = "binomial",
  trControl = ctrl, metric = "ROC"
)
print(model)  # ROC 0.820, Sens 0.68, Spec 0.84
```

`caret` **nutzt automatisch Stratified K-Fold für Klassifikation.**

R-Code: Lasso mit cv.glmnet

```
library(glmnet)

x <- model.matrix(churn ~ ., data = dataco)[, -1]
y <- dataco$churn

cv_fit <- cv.glmnet(x, y, alpha = 1,
                    family = "binomial", nfolds = 5)

cat("lambda.min:", cv_fit$lambda.min)   # 0.023
cat("lambda.1se:", cv_fit$lambda.1se)   # 0.041
coef(cv_fit, s = "lambda.min")  # 3 von 12 = 0
```

`cv.glmnet` **testet automatisch 100 λ-Werte und wählt das optimale per CV.**

R-Code: Manuelle K-Fold-Implementierung

```
set.seed(42); K <- 5
folds <- sample(rep(1:K, length.out = nrow(dataco)))
errors <- numeric(K)

for (k in 1:K) {
  fit_k <- glm(churn ~ ., data = dataco[folds != k, ],
               family = binomial)
  pred_k <- predict(fit_k, dataco[folds == k, ],
                    type = "response")
  errors[k] <- mean((pred_k > 0.5) == dataco$churn[folds == k])
}
cat("Mean Accuracy:", mean(errors), "SD:", sd(errors))
```

Die manuelle Implementierung zeigt die innere Logik: Schleife über Folds, trainieren, testen, mitteln.

Übung: Kreuzvalidierung

Aufgabe 1 – Fold-Größen: Lena hat $n = 120$ Datenpunkte und wählt $K = 10$.

(a) Wie viele Datenpunkte enthält jeder Fold?

(b) Wie viele Datenpunkte werden pro Durchlauf zum Training verwendet?

(c) Wie oft wird jeder Punkt insgesamt zum Training herangezogen?

Aufgabe 2 – Modellwahl: Zwei Modelle ergeben:

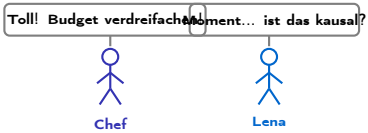
Modell	Mean Accuracy	SD
Modell A	0.80	0.08
Modell B	0.77	0.02

Welches empfehlen Sie? Beachten Sie Mittelwert *und* Streuung.

Aufgabe 3: Warum führt es zu optimistischem Bias, wenn man λ auf denselben Daten wählt, auf denen man auch die Performance bewertet?

Lösung Aufgabe 1: (a) 12, (b) 108, (c) 9 Mal.

r = 0,45 – Budget verdreifachen?



Lena präsentiert ihrem Chef:

- Marketing-Ausgaben und Kundenbindung korrelieren mit $r = 0{,}45$
- Der Chef will das Budget sofort **verdreifachen**
- Lena zögert: Bedeutet Korrelation wirklich **Kausalität**?

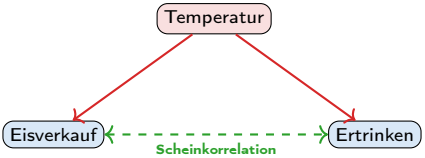
Die zentrale Frage: Verursacht Marketing die Bindung – oder steckt etwas anderes dahinter?

Korrelation $\neq$ Kausalität. Lenas Skepsis könnte dem Unternehmen viel Geld sparen.

Eiscreme verbieten, um Leben zu retten?

Entdeckungsfrage: Eiscremeverkauf und Ertrinkungsfälle steigen beide im Sommer. Sollen wir Eiscreme verbieten, um Menschenleben zu retten?

Natürlich nicht! Die **Hitze im Sommer** treibt beides:

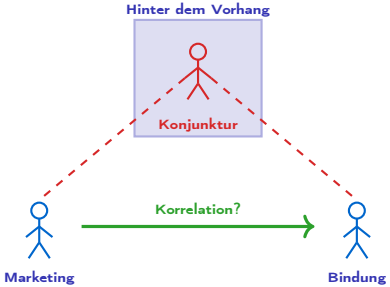


Die Korrelation ist **real** – aber sie entsteht durch eine dritte Variable, die auf beide einwirkt.

Hinter jeder Korrelation lauert die Frage: Gibt es einen versteckten Dritten?

Der Puppenspieler hinter dem Vorhang

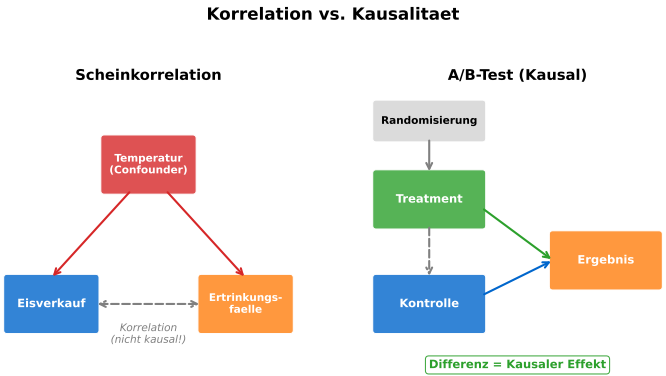
Korrelation: Statistischer Zusammenhang zwischen zwei Variablen. Sagt nichts über Richtung oder Ursache.



Confounder: Drittvariable, die sowohl X als auch Y beeinflusst und so eine Scheinkorrelation erzeugt.

Wer nur die Bühne sieht, glaubt X beeinflusse Y. Wer hinter den Vorhang schaut, sieht den Puppenspieler.

Korrelation vs. Kausalität



Korrelation kann durch Confounding, umgekehrte Kausalität oder reinen Zufall entstehen.

Drei Mechanismen hinter Korrelation

Kausalität: X verursacht Y, wenn eine Änderung in X – bei Konstanthaltung aller anderen Faktoren – zu einer Änderung in Y führt.

Wenn X und Y korrelieren, gibt es **genau drei** mögliche Erklärungen:

1. **Confounding (Drittvariable):** Ein verstecktes Z beeinflusst X und Y. Die Korrelation verschwindet, sobald man Z kontrolliert.
2. **Reverse Causation:** Nicht $X \to Y$, sondern $Y \to X$.
   Beispiel: Kranke nehmen mehr Medikamente – die Medikamente verursachen nicht die Krankheit.
3. **Zufall (Spurious):** Bei genug Variablen findet man immer Korrelationen.
   Tyler Vigen: Scheidungsrate in Maine korreliert $r = 0{,}99$ mit Margarineverbrauch.

In keinem Fall würde eine Intervention an X eine Änderung in Y bewirken!

Die Unterscheidung ist nicht akademisch – sie hat direkte Konsequenzen für Entscheidungen.

Fünf Kriterien für Kausalität

Angelehnt an Bradford Hill (1965) – fünf praxistaugliche Punkte:

1. **Korrelation vorhanden:** X und Y müssen statistisch zusammenhängen. Ohne Korrelation keine Kausalität – aber Korrelation allein reicht nicht.
2. **Zeitliche Reihenfolge:** Die Ursache muss **vor** der Wirkung eintreten. Marketing muss vor der Bindungsmessung stattfinden.
3. **Kein Confounding:** Alle relevanten Drittvariablen müssen kontrolliert sein. *Die härteste Bedingung.*
4. **Plausibler Mechanismus:** Es muss eine nachvollziehbare Erklärung geben, *warum* X auf Y wirkt. "Margarine verursacht Scheidungen" hat keinen.
5. **Konsistenz:** Der Zusammenhang zeigt sich wiederholt in verschiedenen Kontexten und Stichproben.

Hill-Kriterien (1965): Ursprünglich 9 Kriterien – hier auf 5 Kernpunkte verdichtet.

Der Goldstandard: Randomisiertes Experiment

Warum Randomisierung?

- Zufällige Zuteilung eliminiert **alle** Confounders
- Auch solche, die wir nicht kennen oder messen können
- Die 5 Kriterien helfen, wenn ein Experiment nicht möglich ist

Wann kein Experiment möglich?

- Ethische Gründe (z. B. Rauchen)
- Praktische Gründe (z. B. Konjunktur)
- Zeitliche Gründe (Effekt dauert Jahre)

A/B-Test bei DataCo:

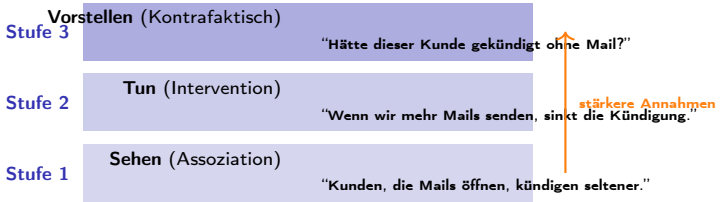
- Zufällig 50 % der Kunden: erhöhtes Marketing
- Kontrollgruppe: normales Marketing
- Nach 3 Monaten: Bindungsrate vergleichen
- Unterschied = **kausaler Effekt**

Merke: Der A/B-Test ist der einzige Weg, **alle** Confounders auszuschalten.

Ohne Experiment ist Kausalität schwer nachweisbar – aber nicht unmöglich (siehe kausale Methoden).

Pearls Leiter der Kausalität

Judea Pearl: Informatiker (UCLA), Turing Award 2011. Begründer der modernen kausalen Inferenz.



- Lenas $r = 0{,}45$ = **Stufe 1** (reine Beobachtung)
- Für Stufe 2 braucht sie ein **Experiment** (A/B-Test)
- Stufe 3 erfordert ein **kausales Modell** für kontrafaktische Fragen

Jede Stufe baut auf der vorherigen auf und erfordert stärkere Annahmen.

Pearls do-Operator

Der Unterschied zwischen Beobachtung und Intervention – formal:

$$\underbrace{P(Y \mid X = x)}_{\textbf{Stufe 1: Beobachtet}} \quad \neq \quad \underbrace{P(Y \mid \mathrm{do}(X = x))}_{\textbf{Stufe 2: Interveniert}}$$

Symbol für Symbol:

- $P(Y \mid X = x)$ – Wahrscheinlichkeit von Y, *gegeben wir beobachten* $X = x$
- $P(Y \mid \mathrm{do}(X = x))$ – Wahrscheinlichkeit von Y, *wenn wir aktiv setzen* $X = x$
- $\mathrm{do}(X = x)$ – fixiert X auf x und **kappt alle kausalen Einflüsse auf** X

Kernidee: In der Beobachtung verzerren Confounders. Der do-Operator eliminiert diese Verzerrungen – wie ein randomisiertes Experiment.

Der do-Operator ist Pearls mathematisches Werkzeug, um "Was passiert, wenn ich eingreife?" formal zu stellen.

Confounding: Das zentrale Problem

Confounder (formal): Variable Z ist Confounder für $X \to Y$, wenn:

1. Z beeinflusst X $\quad (Z \to X)$
2. Z beeinflusst Y $\quad (Z \to Y)$
3. Z liegt **nicht** auf dem kausalen Pfad $X \to Y$

Z erzeugt eine **Scheinkorrelation:** Korrelation zwischen X und Y, die verschwindet, sobald man Z kontrolliert.

Eiscreme und Ertrinken:

- Im Sommer steigt die Temperatur (Z)
- Mehr Eisverkauf (X) **und** mehr Schwimmbadbesuche $\to$ mehr Ertrinken (Y)
- Kontrolliere Temperatur (z. B. nur Tage mit 25°C vergleichen) $\to$ Korrelation verschwindet

Confounding ist der Hauptgrund, warum Korrelation nicht Kausalität impliziert.

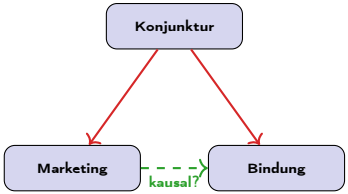
Lenas Konjunktur-Confounder

Die Situation bei DataCo:

- Gute Konjunktur → mehr Marketing-Budget (weil Geld da ist)
- Gute Konjunktur → Kunden bleiben länger (weniger preissensibel)
- Konjunktur beeinflusst **beides**

Konsequenz:

- $r = 0{,}45$ könnte großteils **Scheinkorrelation** sein
- Getrieben durch den Puppenspieler "Konjunktur"
- Budget verdreifachen → möglicherweise kein Effekt!



Rote Pfeile: Gesicherte Wirkungen.
Grüner Pfeil: Die offene Frage – gibt es einen **direkten** kausalen Effekt?

Um die offene Frage zu klären, muss Lena den Confounder "Konjunktur" kontrollieren.

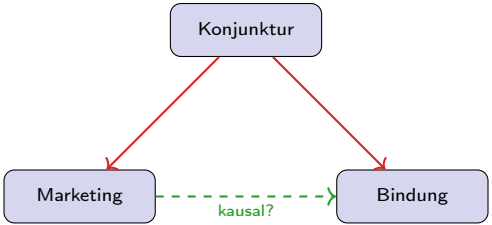
DAGs: Kausale Beziehungen zeichnen

DAG: Directed Acyclic Graph – Diagramm mit Knoten (Variablen) und gerichteten Pfeilen (kausale Wirkungen). "Azyklisch" = kein Pfeil führt im Kreis zurück.

Die drei Regeln:

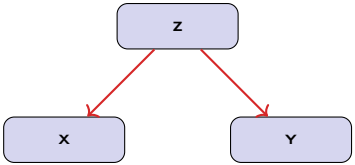
- Jeder **Knoten** = eine Variable
- Jeder **Pfeil** = eine direkte kausale Wirkung (Ursache → Wirkung)
- **Azyklisch**: Man kann den Pfeilen nicht im Kreis folgen

Lenas DAG für DataCo:

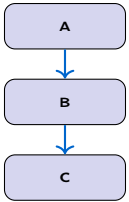


DAGs machen kausale Annahmen explizit und sichtbar – das Standardwerkzeug der kausalen Inferenz.

DAGs lesen und interpretieren



Confounding: **Z beeinflusst X und Y**



Kette: **A → B → C**

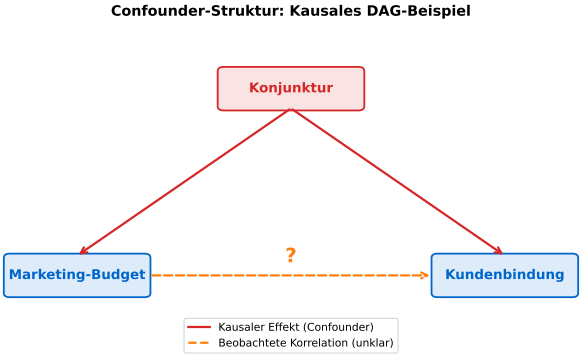
DAG-Bausteine:

- **Fork** (Confounding): $X \leftarrow Z \rightarrow Y$ – Kontrolle für Z nötig
- **Kette** (Mediation): $X \rightarrow M \rightarrow Y$ – M ist Mediator, **nicht** kontrollieren
- **Collider**: $X \rightarrow C \leftarrow Y$ – C **nicht** kontrollieren (sonst Bias!)

Adjustment Set: Variablen, die man kontrollieren muss, um den kausalen Effekt zu schätzen.

Den richtigen DAG zu zeichnen ist eine inhaltliche Entscheidung – keine statistische.

Confounding im DAG – Visualisierung



Der DAG zeigt: Konjunktur ist der Confounder. Kontrolliere Konjunktur, um den kausalen Effekt zu schätzen.

Kausale Methoden – Der Werkzeugkasten

Wenn kein Experiment möglich ist:

Methode	Kernidee	Kritische Annahme
Kontrollvariablen	Confounder ins Modell aufnehmen	Alle Confounders bekannt
Matching	Vergleichbare "Zwillinge" bilden	Gute Matches möglich
Instrumentvariablen	Exogene Variation nutzen	Gültiges Instrument existiert
Difference-in-Differences	Vorher/Nachher + Kontrollgruppe	Parallele Trends

- **Kontrollvariablen:** Einfachster Ansatz – funktioniert nur für *bekannte* Confounders
- **Matching:** Für jeden behandelten Kunden einen "Zwilling" suchen
- **IV:** Z. B. zufällig verteilter Gutschein als Instrument
- **DiD:** Vergleiche Treatment- vs. Kontrollgruppe, vorher vs. nachher

Jede Methode hat eine kritische Annahme – keine ist ein Allheilmittel.

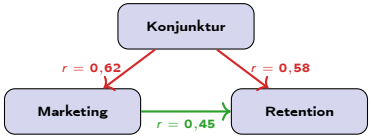
Lenas Marketing-Puzzle (1/2)

Schritt 1 – Rohe Korrelation:

- Marketing-Ausgaben vs. 30-Tage-Retention über 24 Monate
- $r = 0{,}45$, $p < 0{,}05$ – der Chef ist begeistert

Schritt 2 – Confounder identifizieren:

- Lena überlegt: Was könnte **beides** beeinflussen?
- Sie hat Zugang zum Konjunkturindex (BIP-Wachstum)
- Konjunktur korreliert mit Marketing: $r = 0{,}62$
- Konjunktur korreliert mit Retention: $r = 0{,}58$



Verdacht: Die Korrelation von 0,45 könnte durch den Confounder Konjunktur getrieben sein.

Lenas Marketing-Puzzle (2/2)

Schritt 3 – Partielle Korrelation:
Lena berechnet die Korrelation zwischen Marketing und Retention, **kontrolliert für Konjunktur**:

$$r_{\text{Marketing, Retention}|\text{Konjunktur}} = 0{,}12 \quad (p = 0{,}38,\ \text{n.s.})$$

Ergebnis:

- Rohe Korrelation: $r = 0{,}45$ (signifikant)
- Nach Kontrolle für Konjunktur: $r = 0{,}12$ (nicht signifikant!)
- Die ursprüngliche Korrelation war großteils **Scheinkorrelation**

Lenas Fazit an den Chef:
"Die Daten zeigen keine überzeugenden Belege, dass Marketing die Bindung direkt verbessert. Um das sicher zu wissen, brauchen wir einen **A/B-Test**."

Partielle Korrelation: Das einfachste Werkzeug, um einen Confounder zu kontrollieren.

Drei Geschichten über Kausalität

Judea Pearl (Turing Award 2011)

- KI-Forscher an der UCLA
- Erkenntnis: Maschinen "verstehen" Ursache und Wirkung nicht, wenn sie nur Korrelationen lernen
- Entwickelt den do-Kalkül und DAGs

Rauchen & Krebs (50 Jahre Debatte)

- 1950er: Doll & Hill zeigen starke Korrelation
- R.A. Fisher argumentiert dagegen: "Vielleicht ein Gen?"
- 1965: Hill formuliert seine 9 Kriterien
- Erst dann: Konsens über Kausalität

Tyler Vigen (2014)

- Harvard-Student startet "Spurious Correlations"
- Scheidungsrate Maine vs. Margarine: $r = 0{,}99$
- Nicolas-Cage-Filme vs. Ertrinkungstode
- Lektion: Bei genug Variablen findet man **immer** hohe Korrelationen

Verbindung zu Block 2: Verwandt mit dem Problem des multiplen Testens (p-Hacking).

Von Fishers Irrtum über Pearl's Revolution bis zu Vigens Humor – Kausalität hat eine reiche Geschichte.

Drei häufige Kausalitäts-Irrtümer

Irrtum 1: "Eine starke Korrelation bedeutet Kausalität."

- Selbst $r = 0{,}99$ kann spurious sein (Margarine/Scheidungen)
- Die **Stärke** sagt nichts über den kausalen Status
- Entscheidend: Kein Confounding + plausibler Mechanismus

Irrtum 2: "Wir können nie Kausalität zeigen."

- Zu pessimistisch! A/B-Tests zeigen kausale Effekte
- Auch aus Beobachtungsdaten: IV, DiD, Regression Discontinuity
- Richtige Frage: "Welche Annahmen brauche ich, und wie plausibel sind sie?"

Irrtum 3: "Kontrollvariablen lösen das Problem."

- Helfen nur gegen **bekannte und gemessene** Confounders
- "No unmeasured confounding" ist selten garantiert
- Nur Randomisierung eliminiert **alle** Confounders

Gesunde Skepsis + richtige Methodik = der Weg zu kausalen Aussagen.

R-Code: Korrelation und partielle Korrelation

```
# Rohe Korrelation
cor.test(dataco$marketing, dataco$retention)
# => r = 0.45, p = 0.027

# Korrelationsmatrix
cor(dataco[, c("marketing","retention","konjunktur")])

# Partielle Korrelation (ppcor)
library(ppcor)
pcor.test(x = dataco$marketing,
          y = dataco$retention,
          z = dataco$konjunktur)
# => r = 0.12, p = 0.38 (n.s.!)
```

`ppcor::pcor.test()` **berechnet die partielle Korrelation und testet auf Signifikanz.**

R-Code: DAGs mit ggdag

```
library(ggdag)

dag <- dagify(
  Retention ~ Marketing + Konjunktur,
  Marketing ~ Konjunktur,
  exposure = "Marketing",
  outcome  = "Retention"
)

ggdag(dag, text = TRUE, use_labels = "name") +
  theme_dag()

adjustmentSets(dag)   # => { Konjunktur }
```

Ergebnis: R identifiziert automatisch: Um den kausalen Effekt von Marketing auf Retention zu schätzen, muss Lena für **Konjunktur** kontrollieren.

`adjustmentSets()` **berechnet aus dem DAG, welche Variablen kontrolliert werden müssen.**

Übung: Kausalität

Aufgabe 1 – Korrelation oder Kausalität?

(a) Schüler, die frühstücken, haben bessere Noten. Kausal oder Confounder?

(b) Städte mit mehr Feuerwehrleuten haben mehr Brände. Warum?

(c) Patienten mit Medikament X haben niedrigeren Blutdruck. Kausal?

Aufgabe 2 – Pearls Leiter: Ordnen Sie zu (Sehen / Tun / Vorstellen):

(a) "Kunden, die den Newsletter lesen, kaufen mehr."

(b) "Wenn wir den Newsletter an alle schicken, steigt der Umsatz."

(c) "Hätte Kunde X mehr gekauft, wenn er den Newsletter erhalten hätte?"

Aufgabe 3 – DAG zeichnen: Homeoffice und Produktivität. Confounders: Berufserfahrung, Aufgabentyp. Zeichnen Sie den DAG und bestimmen Sie das Adjustment Set.

Lösung Aufgabe 2: (a) Sehen, (b) Tun, (c) Vorstellen.

Lena schlägt ein Experiment vor

Die Korrelation Marketing–Retention war Confounding. Nur ein Experiment klärt die Kausalität.

Entdeckungsfrage: Warum 1000 Kunden?

Lenas Chef runzelt die Stirn:

> "Warum brauchen wir 1000 Kunden für diesen Test? Reichen nicht 20?"

Überlegen Sie selbst, bevor Sie weiterblättern:

- Was passiert mit der Streuung bei nur 20 Beobachtungen?
- Wie breit wäre das Konfidenzintervall?
- Könnte ein echter Effekt im Rauschen untergehen?

$\Rightarrow$ Die Antwort führt zu einem zentralen Konzept: **Statistische Power**.

Mit $n = 20$ würde Lena einen echten Effekt in weniger als jedem achten Versuch entdecken.

Power = Wahrscheinlichkeit, einen tatsächlich vorhandenen Effekt zu erkennen. Standard: 80%.

A/B-Test: Vier Schritte zum kausalen Schluss

Das Protokoll eines randomisierten Experiments:

1. **Zufällige Zuteilung:** Jeder Kunde wird per Zufall A oder B zugewiesen – keine Selbstselektion, keine Vorauswahl
2. **Intervention nur in Gruppe B:** Nur B bekommt das neue Onboarding, A durchläuft den alten Prozess
3. **Ergebnis messen:** Nach 30 Tagen die primäre Metrik in beiden Gruppen erheben
4. **Differenz = kausaler Effekt:** $\Delta = \bar{Y}_B - \bar{Y}_A$ ist ein unverzerrter Schätzer des kausalen Effekts

Kausaler Effekt (ATE): $E[\bar{Y}_B - \bar{Y}_A]$ – unter Randomisierung unverzerrt, weil die Gruppen bis auf die Intervention identisch sind.

A/B-Tests sind das Standardtool für kausale Fragen im digitalen Business.

Warum Randomisierung funktioniert

Das Kernargument:

Randomisierung macht die Gruppen *im Erwartungswert identisch* – in **allen** Variablen, auch den unbekannten.

Beobachtungsstudie:

- Gruppen sind selbstselektiert
- Bekannte Confounders: kontrollierbar
- Unbekannte Confounders: **nicht kontrollierbar!**

Randomisiertes Experiment:

- Gruppen sind per Zufall eingeteilt
- Bekannte Confounders: balanciert
- Unbekannte Confounders: **auch balanciert!**

Gedankenexperiment: Zwei Kopien desselben Kundenuniversums – eine mit altem, eine mit neuem Onboarding. Die Differenz ist der kausale Effekt. Randomisierung approximiert diese kontrafaktische Welt.

Randomisierung eliminiert alle Confounders – auch die, die wir nicht kennen.

Testdesign: Fünf Entscheidungen vor dem Experiment

Bevor ein einziger Kunde in den Test kommt:

1. **Hypothese formulieren:** "Das neue Onboarding erhöht die 30-Tage-Retention" – präzise und testbar
2. **Primäre Metrik festlegen:** Retention nach 30 Tagen (binär). Nur *eine* primäre Metrik!
3. **MDE definieren:** Minimum Detectable Effect – wie groß muss der Effekt sein, damit er geschäftlich relevant ist? Lena wählt $d = 0{,}3$
4. **Stichprobengröße berechnen:** Basierend auf MDE, Power und α
5. **Laufzeit planen:** 30 Tage Intervention plus Puffer für Saisonalität

Pre-Registration: Alle Entscheidungen werden *vor* dem Test dokumentiert und nicht nachträglich angepasst.

MDE = der kleinste Effekt, den der Test mit geplanter Power entdecken kann.

Stichprobengröße: Die Formel

Benötigte Beobachtungen pro Gruppe:

$$n \approx \frac{2\,(z_{\alpha/2} + z_{\beta})^2}{d^2}$$

Symbol für Symbol:

- n – Anzahl pro *Gruppe* (nicht insgesamt)
- $z_{\alpha/2}$ – kritischer Wert für Signifikanzniveau (bei $\alpha = 0{,}05$: $z = 1{,}96$)
- z_{β} – kritischer Wert für Power (bei 80%: $z = 0{,}84$; bei 90%: $z = 1{,}28$)
- d – erwartete Effektstärke (Cohen's d), der minimale Effekt, den wir entdecken wollen

Gilt für zweiseitigen Zweistichproben-t-Test mit gleich großen Gruppen.

Power = Wahrscheinlichkeit, einen tatsächlich vorhandenen Effekt als signifikant zu erkennen.

Lenas Power-Berechnung: Schritt für Schritt

Lenas Werte: $d = 0{,}3$, Power $= 0{,}80$ ($z_\beta = 0{,}84$), $\alpha = 0{,}05$ ($z_{\alpha/2} = 1{,}96$)

$$n \approx \frac{2 \times (1{,}96 + 0{,}84)^2}{0{,}3^2} = \frac{2 \times 7{,}84}{0{,}09} = \frac{15{,}68}{0{,}09} \approx 174 \text{ pro Gruppe}$$

Warum plant Lena 500 pro Gruppe?

- Subgruppenanalysen (Premium vs. Free) brauchen zusätzliche Power
- Schutz gegen Drop-outs und technische Fehler
- Power steigt nichtlinear: bei $n = 500$ beträgt sie über 99% für $d = 0{,}3$

n pro Gruppe	Power
20	12%
100	48%
174	80%
500	99%

Bei $n = 20$ würde Lena den Effekt in 88% der Fälle übersehen.

Cohen's d: Wie groß ist der Effekt?

Standardisierte Effektstärke:

$$d = \frac{\bar{X}_T - \bar{X}_C}{s_p} \qquad \text{mit} \qquad s_p = \sqrt{\frac{(n_T - 1)\, s_T^2 + (n_C - 1)\, s_C^2}{n_T + n_C - 2}}$$

Cohen's Daumenregel (1988):

d	Interpretation
0.2	Klein – schwer mit bloßem Auge zu sehen
0.5	Mittel – deutlich spürbar (Aspirin gegen Kopfschmerz: $d \approx 0{,}5$)
0.8	Groß – offensichtlich

Warum nicht nur den p-Wert?

p hängt von n ab: Mit 1 Million Kunden wird selbst $d = 0{,}01$ signifikant. Cohen's d ist unabhängig von n und sagt, *wie groß* der Effekt ist.

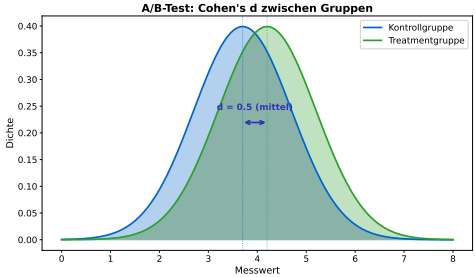
Effektstärke vor dem Experiment festlegen – wie groß muss der Effekt sein, um relevant zu sein?

Power und Signifikanz: Die zwei Verteilungen

Was zeigt die Grafik?

- **Links (blau):** Verteilung unter H_0 (kein Effekt)
- **Rechts (orange):** Verteilung unter H_1 (Effekt d)
- α **(rot):** Fälschlicherweise H_0 ablehnen (Fehler 1. Art)
- β **(grau):** Effekt übersehen (Fehler 2. Art)
- **Power** $= 1 - \beta$**:** Effekt korrekt erkennen

Mehr $n \Rightarrow$ schmalere Verteilungen $\Rightarrow$ weniger Überlappung $\Rightarrow$ höhere Power.



Power ist die Fläche rechts vom kritischen Wert unter der H_1-Verteilung.

Drei gefährliche Fehler bei A/B-Tests

Fehler 1: Peeking – zu früh reinschauen
Lena schaut nach einer Woche: $p = 0{,}03$ – signifikant! Aber bei täglichem Prüfen über 30 Tage steigt die Falsch-Positiv-Rate von 5% auf bis zu **30%**. Lösung: Erst am Ende auswerten oder sequentielle Testverfahren nutzen.

Fehler 2: Multiple Testing – zu viele Metriken
10 Metriken auf 5%-Niveau: $1 - 0{,}95^{\mathbf{10}} \approx 0{,}40$ – **40%** Chance auf ein Falsch-Positives! Lösung: *Eine* primäre Metrik vorher festlegen, Rest mit Bonferroni korrigieren (α/k).

Fehler 3: Simpson's Paradoxon
Gesamteffekt positiv, aber in *jeder* Subgruppe negativ? Möglich, wenn die Gruppenzusammensetzung unterschiedlich ist. Lösung: Nach dem Test Subgruppenanalysen durchführen und Randomisierung auf Balance prüfen.

Lena: "Ich prüfe NUR meine primäre Metrik und schaue erst am geplanten Ende in die Daten."

Peeking, Multiple Testing, Simpson's Paradox – die drei größten A/B-Test-Fallen.

Lenas A/B-Test bei DataCo: Design und Durchführung

Phase 1 – Design:

- H_0: "Das neue Onboarding hat keinen Effekt auf die 30-Tage-Retention"
- Primäre Metrik: Retention (binär). MDE: $d = 0{,}3$. Power: 80%. $\alpha = 0{,}05$
- Benötigt: $n = 174$ pro Gruppe, geplant: 500 pro Gruppe

Phase 2 – Durchführung:

- 1000 Neukunden per Zufallsgenerator in Gruppe A (alt) und B (neu) eingeteilt
- Randomisierung geprüft: Alter, Geschlecht, Premium-Status fast identisch verteilt
- Während der 30 Tage schaut Lena *nicht* in die Daten

Pre-Registration: Hypothese, Metrik, MDE, Sample Size und Laufzeit wurden *vor* dem Test dokumentiert.

Die Planung ist die halbe Miete – das Experiment läuft nur 30 Tage, die Planung dauert Wochen.

Lenas A/B-Test: Ergebnis und Board-Präsentation

Phase 3 – Auswertung:

	Kontrolle (A)	Treatment (B)
30-Tage-Retention	68%	76%
Zufriedenheits-Score	3,7	4,2

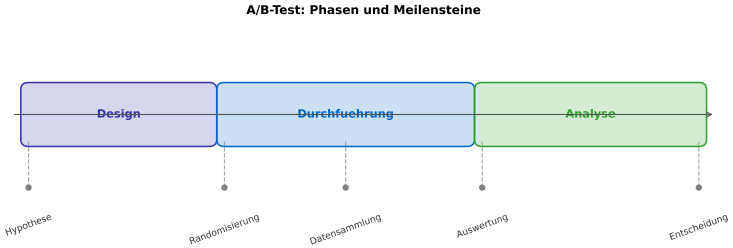
$d = 0{,}45$ (mittlerer Effekt), $p = 0{,}003$, 95%-KI für d: [0,18; 0,72]
KI schließt Null nicht ein $\Rightarrow$ statistisch signifikant **und** praktisch relevant.

Phase 4 – Board-Präsentation:

- Neues Onboarding erhöht Retention um 8 Prozentpunkte (68% $\rightarrow$ 76%)
- Bei 10.000 Neukunden/Monat und CLV = 500 CHF: **+400.000 CHF/Monat**
- Empfehlung: Sofort für alle Kunden ausrollen

Von der Statistik zur Geschäftsentscheidung: Lena überzeugt das Board mit Zahlen.

Zeitlicher Ablauf eines A/B-Tests



Drei Phasen:

1. **Design** (Wochen): Hypothese, Metrik, MDE, Sample Size, Pre-Registration
2. **Durchführung** (Tage–Wochen): Randomisierung, Monitoring, *kein* Peeking
3. **Analyse** (Tage): Primäre Metrik, Effektstärke, Subgruppenanalyse, Dokumentation

---

Die Planungsphase dauert oft länger als das Experiment selbst.

275 Jahre kontrollierte Experimente

Von Skorbut bis Silicon Valley:

- **1747 – James Lind:** Testet 6 Behandlungen gegen Skorbut an 12 Matrosen auf der HMS Salisbury. Nur Zitrusfrüchte helfen. Erste kontrollierte Studie der Geschichte.
- **1920er – Ronald Fisher:** Formalisiert Randomisierung in Agrarexperimenten. Seine "Lady Tasting Tea" führt zum exakten Test.
- **2008 – Obama:** 24 Versionen der Spendenseite im A/B-Test – die Gewinnerversion brachte **60 Millionen Dollar** mehr Spenden.
- **Heute:** Google führt über **10.000 A/B-Tests pro Jahr** durch. Amazon, Netflix, Booking.com testen praktisch jede Änderung.

Lena: "Wenn A/B-Tests für Präsidentschaftswahlen funktionieren, dann auch für DataCos Onboarding!"

Von Skorbut (1747) bis Silicon Valley – A/B-Tests sind 275 Jahre alt und aktueller denn je.

A/B-Tests: Häufige Irrtümer

Irrtum 1: "A/B-Tests geben immer klare Antworten."
Nein. Bei kleinen Effekten und kleinen Stichproben liefern sie oft nicht-signifikante Ergebnisse. Das bedeutet nicht, dass kein Effekt existiert – nur dass der Test zu wenig Power hatte.

Irrtum 2: "Statistisch signifikant = praktisch wichtig."
Bei sehr großen n wird selbst $d = 0{,}02$ signifikant. Deshalb immer die Effektstärke berichten: $p = 0{,}001$ aber $d = 0{,}02$ ist statistisch signifikant, aber geschäftlich irrelevant.

Irrtum 3: "Früh reinschauen ist okay, solange man nicht stoppt."
Falsch. Allein das Anschauen beeinflusst die Entscheidungsfindung, und jedes Prüfen erhöht die kumulative Fehlerrate. Entweder am Ende auswerten oder ein sequentielles Testverfahren verwenden.

Nicht-signifikant heißt nicht "kein Effekt" – es heißt "nicht genug Power".

R-Code: A/B-Test auswerten

```
# Zweistichproben-t-Test
t.test(retention ~ gruppe, data = ab_daten)

# Cohen's d mit effsize-Paket
library(effsize)
cohen.d(retention ~ gruppe, data = ab_daten)

# Power-Analyse: Wie viele Kunden brauchen wir?
power.t.test(delta = 0.3, sd = 1,
  sig.level = 0.05, power = 0.80,
  type = "two.sample")  # n = 176 pro Gruppe
```

t.test für Signifikanz, cohen.d für Effektstärke, power.t.test für Planung.

Übung: A/B-Testing

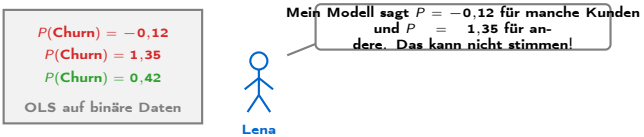
Aufgabe 1 – Power-Analyse:
Ein Online-Shop testet eine neue Produktseite (erwarteter Effekt $d = 0{,}4$). Wie viele Besucher braucht man pro Gruppe bei $\alpha = 0{,}05$ und Power $= 0{,}80$? Setzen Sie die Werte in die Formel ein.

Aufgabe 2 – Cohen's d berechnen:
Kontrollgruppe: $\bar{X}_C = 48$ CHF, $s_C = 12$ ($n = 200$). Treatment: $\bar{X}_T = 52$ CHF, $s_T = 14$ ($n = 200$). Berechnen Sie s_p und d. Wie ordnen Sie das Ergebnis ein?

Aufgabe 3 – Peeking:
Ein PM stoppt nach 8 Tagen bei $p = 0{,}04$ (geplant: 30 Tage). Warum ist das Ergebnis nicht verlässlich? Schätzen Sie die effektive Falsch-Positiv-Rate bei 8 Zwischenanalysen: $1 - 0{,}95^k$.

Tipp: Die Sample-Size-Formel ergibt $n \approx 2(1{,}96 + 0{,}84)^2 / d^2$.

Lena und das Churn-Problem



Churn ist binär: der Kunde kündigt – oder nicht. Negative Wahrscheinlichkeiten und $P > 1$ sind mathematischer Unsinn. Lena braucht ein anderes Modell.

Wenn OLS auf 0/1-Daten angewendet wird, liegen die Vorhersagen nicht zwingend in $[0, 1]$.

Entdeckungsfrage: Warum versagt OLS bei Ja/Nein?

Zeichnen Sie eine Gerade durch Datenpunkte, die nur bei $Y = 0$ und $Y = 1$ liegen. Was passiert an den Rändern?

Drei Probleme der OLS auf binäre Daten:

1. **Vorhersagen außerhalb** $[0, 1]$**:** Eine Gerade ist unbeschränkt – für extreme X-Werte sagt das Modell $P < 0$ oder $P > 1$ vorher
2. **Linearität verletzt:** Die wahre Beziehung $X \to P(Y = 1)$ ist S-förmig, nicht linear
3. **Heteroskedastizität:** Bei binärem Y gilt $\text{Var}(Y|X) = P(1 - P)$ – die Varianz hängt vom Mittelwert ab

$\Rightarrow$ Wir brauchen eine Funktion, die *jeden* Input in $[0, 1]$ abbildet.

OLS modelliert eine Gerade – Wahrscheinlichkeiten brauchen eine S-Kurve.

Lichtschalter vs. Dimmer

Die Analogie:

Lichtschalter (Outcome):

- Kunde kündigt oder bleibt
- Nur zwei Zustände: 0 oder 1
- Das *Ergebnis* ist binär

Dimmer (Modell):

- Wahrscheinlichkeit von 0 bis 1
- Je weiter aufgedreht, desto wahrscheinlicher die Kündigung
- Das *Modell* ist kontinuierlich

Ziel der logistischen Regression:

Nicht die binäre Entscheidung selbst modellieren, sondern:

$$P(Y = 1 \mid X)$$

die Wahrscheinlichkeit der Kündigung, gegeben die Merkmale des Kunden.

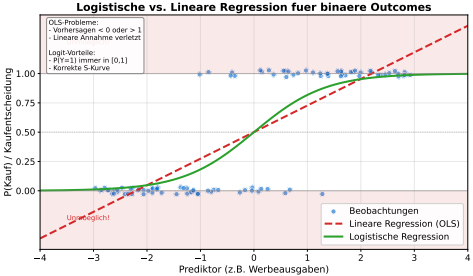
Diese Wahrscheinlichkeit muss *immer* zwischen 0 und 1 liegen.

Logistische Regression modelliert den Dimmer, nicht den Schalter.

OLS vs. Logit: Der Vergleich

Was zeigt die Grafik?

- **Rote Gerade (OLS):** Überschreitet die Grenzen – $P < 0$ und $P > 1$ möglich
- **Blaue S-Kurve (Logit):** Bleibt immer in $(0, 1)$
- Am steilsten bei $P = 0{,}5$ – dort haben Änderungen in X den größten Effekt
- Flacht zu den Rändern ab – wer schon fast sicher kündigt, wird kaum noch beeinflusst



Die logistische Funktion garantiert Vorhersagen in $(0, 1)$ – egal wie extrem die X-Werte.

Die logistische Funktion (Sigmoid)

Die Formel:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

Symbol für Symbol:

- $P(Y = 1|X)$ – Wahrscheinlichkeit der Kündigung, gegeben X
- $e \approx 2{,}718$ – Eulersche Zahl
- β_0 – Intercept (auf der Log-Odds-Skala)
- β_1 – Koeffizient von X (Änderung der Log-Odds pro Einheit X)
- $\beta_0 + \beta_1 X$ – linearer Prädiktor, von $-\infty$ bis $+\infty$

Drei Eigenschaften der S-Kurve:

1. Beschränkt auf $(0, 1)$ – gültige Wahrscheinlichkeiten
2. Symmetrisch um den Wendepunkt bei $P = 0{,}5$
3. Am steilsten bei $P = 0{,}5$, flacht zu den Rändern ab

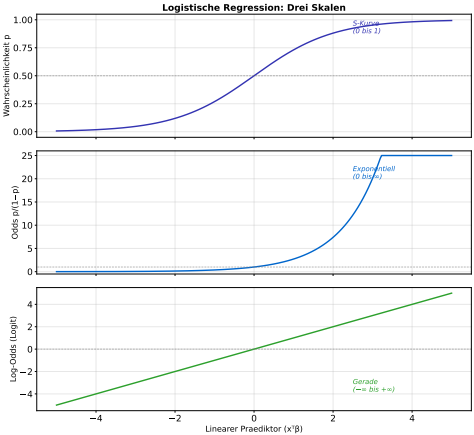
Die Sigmoid-Funktion bildet jeden reellen Wert in das Intervall $(0, 1)$ ab.

Drei Skalen: Wahrscheinlichkeit, Odds, Log-Odds

Dasselbe Modell, drei Perspektiven:

- **Wahrscheinlichkeit** P: S-förmig, in $(0, 1)$ – was wir wollen
- **Odds** $\frac{P}{1-P}$: $(0, \infty)$ – multiplikativ interpretierbar
- **Log-Odds** $\ln \frac{P}{1-P}$: $(-\infty, +\infty)$ – hier ist das Modell *linear*!

Beispiel: Bei $P = 0{,}80$ sind die Odds $\frac{0{,}80}{0{,}20} = 4$ ("4 zu 1") und die Log-Odds $\ln(4) = 1{,}39$.



Auf der Log-Odds-Skala ist die logistische Regression linear – dort leben die Koeffizienten.

Log-Odds und Odds Ratios

Logit-Transformation (Umkehrung der Sigmoid):

$$\ln \frac{p}{1-p} = \beta_0 + \beta_1 X$$

Odds Ratio:

$$OR = e^{\beta_1}$$

Was bedeutet das?

- $\frac{p}{1-p}$ sind die **Odds**: Verhältnis für vs. gegen das Ereignis
- β_1 = Änderung der Log-Odds pro Einheit X
- e^{β_1} = Faktor, um den sich die **Odds multiplizieren** bei $+1$ Einheit X
- $OR = 1$: kein Effekt. $OR > 1$: erhöhte Chance. $OR < 1$: verringerte Chance

Logit-Koeffizienten sind Log-Odds – exp() gibt die interpretierbaren Odds Ratios.

Odds Ratios: Symbol für Symbol

Sportwetten-Analogie:
"3 zu 1 gegen" = Odds von $1/3 = 0{,}33$. Bei $P = 0{,}75$: Odds $= 0{,}75/0{,}25 = 3$ ("dreimal so wahrscheinlich wie unwahrscheinlich").

Der Clou der logistischen Regression:

- Auf der Log-Odds-Skala ist das Modell **linear**
- β-Koeffizienten beschreiben Änderung der Log-Odds
- e^{β} übersetzt in einen **multiplikativen Faktor** für die Odds

Achtung:

- OR ist **nicht** das Relative Risiko! Bei häufigen Ereignissen überschätzt OR den Effekt
- Nur bei seltenen Ereignissen ($p < 0{,}10$) gilt $OR \approx RR$
- Immer Konfidenzintervall mit angeben

OR $= 2{,}5$ heißt nicht "2,5-mal so wahrscheinlich" – sondern "2,5-fache Odds".

Beispiel: Lenas Premium-Kunden

Koeffizient für "Premium-Abo": $\beta_{\textbf{premium}} = 0{,}916$

Schritt 1 – Odds Ratio:

$$OR = e^{\mathbf{0{,}916}} = 2{,}5$$

Premium-Kunden haben 2,5-fach höhere Odds zu *bleiben*.

Schritt 2 – Von Odds zur Wahrscheinlichkeit:

Basis-Bleibewahrscheinlichkeit (Non-Premium): $p = 0{,}40$

Basis-Odds:	$\frac{\mathbf{0{,}40}}{\mathbf{0{,}60}} = 0{,}667$
Premium-Odds:	$0{,}667 \times 2{,}5 = 1{,}667$
Premium-Wahrscheinlichkeit:	$\frac{\mathbf{1{,}667}}{\mathbf{1+1{,}667}} = \frac{\mathbf{1{,}667}}{\mathbf{2{,}667}} \approx 0{,}63$

Fazit: Das Premium-Abo erhöht die Bleibewahrscheinlichkeit von 40% auf 63%.

Odds → OR multiplizieren → zurück zur Wahrscheinlichkeit: $p = \text{Odds}/(1 + \text{Odds})$.

Confusion Matrix: Die vier Felder

Vorhersage in binäre Entscheidung übersetzen (z. B. Churn wenn $P > 0{,}5$):

	Pred. Bleibt	**Pred. Kündigt**
Tats. Bleibt	TN (True Negative)	FP (False Positive)
Tats. Kündigt	FN (False Negative)	TP (True Positive)

Vier Metriken:

- **Accuracy** $= \frac{TP+TN}{N}$ – Anteil korrekt klassifiziert (irreführend bei Unbalance!)
- **Precision** $= \frac{TP}{TP+FP}$ – von allen "Kündiger"-Vorhersagen: wie viele stimmen?
- **Recall** $= \frac{TP}{TP+FN}$ – von allen echten Kündigern: wie viele erkannt?
- **F1** $= 2 \cdot \frac{\textbf{Precision} \cdot \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}}$ – harmonisches Mittel

Bei 5% Churn-Rate erreicht "immer Bleibt" 95% Accuracy – ohne einen Kündiger zu erkennen.

Lenas Confusion Matrix bei DataCo

$n = 500$ **Testkunden, davon 75 tatsächliche Kündiger (15%):**

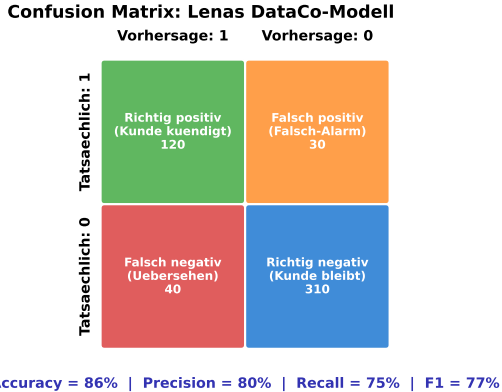
	Pred. Bleibt	Pred. Kündigt	Summe
Bleibt	400 (TN)	25 (FP)	425
Kündigt	19 (FN)	56 (TP)	75
Summe	419	81	500

- Accuracy $= 456/500 = 91{,}2\%$
- Precision $= 56/81 = 69{,}1\%$
- Recall $= 56/75 = 74{,}7\%$
- F1 $= 71{,}8\%$

Achtung: "Immer Bleibt" erreicht $425/500 = 85\%$ Accuracy! Bei unbalancierten Klassen ist Accuracy irreführend.

Precision und Recall sind aussagekräftiger als Accuracy bei unbalancierten Klassen.

Confusion Matrix: Visualisierung



Accuracy = 86% | Precision = 80% | Recall = 75% | F1 = 77%

Die teuersten Fehler bei DataCo:

- **FN (False Negative):** Kündiger übersehen – der teuerste Fehler, weil keine Intervention möglich
- **FP (False Positive):** Zufriedene als Kündiger markiert – verschwendete Retention-Ressourcen

Welche Metrik wichtig ist, hängt vom Business-Problem ab: FN-Kosten vs. FP-Kosten.

ROC-Kurve und AUC

Das Schwellenwert-Problem:

Warum genau $P > 0{,}5$? Niedrigerer Schwellenwert $\rightarrow$ mehr Recall, aber mehr FP. Höherer $\rightarrow$ weniger FP, aber mehr FN.

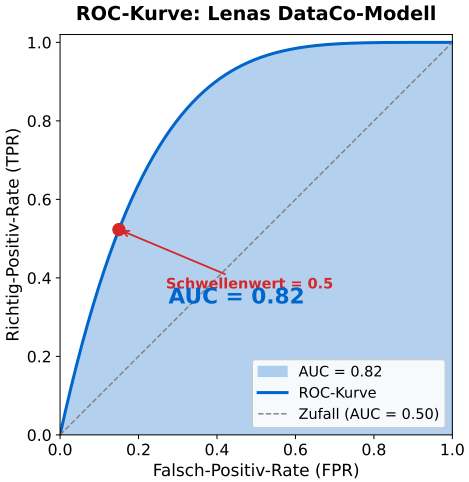
ROC-Kurve: Für *jeden* Schwellenwert:

- TPR (Recall) auf der Y-Achse
- FPR $= FP/(FP + TN)$ auf der X-Achse

AUC-Interpretation:

- 0,5: Zufall (Münzwurf)
- 0,7–0,8: Akzeptabel
- 0,8–0,9: Gut
- $> 0{,}9$: Exzellent

Lenas AUC $= 0{,}82$ (gut).



AUC = Wahrscheinlichkeit, dass das Modell einen Kündiger höher rankt als einen Bleiber.

Lenas DataCo-Modell: glm()-Output

Das Modell: `glm(churn ~ premium + alter + nutzung + beschwerden, family = binomial)`

Variable	β	OR (e^{β})	95%-KI	p
(Intercept)	$-2{,}00$	–	–	$< 0{,}001$
Premium	$-0{,}92$	0,40	[0,27; 0,59]	$< 0{,}001$
Alter (pro Jahr)	0,02	1,02	[1,01; 1,03]	0,018
Nutzung (Monate)	$-0{,}05$	0,95	[0,93; 0,97]	0,002
Beschwerden	0,30	1,35	[1,18; 1,54]	$< 0{,}001$

Interpretation Zeile für Zeile:

- **Premium** ($OR = 0{,}40$): Nur 40% der Kündigungs-Odds von Non-Premium – stärkster Schutzfaktor
- **Alter** ($OR = 1{,}02$): Pro Jahr +2% Odds – klein, aber kumulativ relevant
- **Nutzung** ($OR = 0{,}95$): Pro Monat -5% Odds – Loyalität wächst mit der Zeit
- **Beschwerden** ($OR = 1{,}35$): Pro Beschwerde +35% Odds; bei 3: $1{,}35^{\mathbf{3}} = 2{,}46$-fach!

Lenas Empfehlung: Premium-Kunden binden und Beschwerdemanagement verbessern.

DataCo-Modell: Von Koeffizienten zur Entscheidung

Lenas Profil eines Risikokunden:

- Kein Premium-Abo ($OR = 0{,}40$ entfällt $\rightarrow$ höhere Basis-Odds)
- Relativ jung (höheres Alter-OR kumuliert weniger)
- Wenige Nutzungsmonate (Schutzeffekt gering)
- 3+ Beschwerden ($1{,}35^{\mathbf{3}} = 2{,}46$-fache Odds)

Lenas Empfehlung an das Management:

1. Premium-Konversion pushen – stärkster Schutzfaktor
2. Beschwerdemanagement verbessern – jede Beschwerde erhöht die Kündigungsgefahr
3. Frühwarnsystem einrichten: Kunden mit $P(\text{Churn}) > 0{,}6$ proaktiv kontaktieren

Modellgüte: AUC $= 0{,}82$ (gut), Recall $= 74{,}7\%$ (3 von 4 Kündigern erkannt).

Das Modell sagt nicht nur wer kündigt, sondern erklärt warum – durch interpretierbare Odds Ratios.

Titanic: Das "Hello World" der Logistischen Regression

Die Titanic (1912): 2224 Passagiere, 710 überlebten.

Feature	OR	Interpretation
Geschlecht (weiblich)	10,5	10-fach höhere Überlebens-Odds als Männer
1. Klasse (vs. 3.)	3,8	Reiche überlebten deutlich öfter
Alter (pro Jahr)	0,97	Ältere leicht benachteiligt
Kinder ($<$ 10 J.)	2,1	"Frauen und Kinder zuerst" – in den Daten sichtbar

Die Parallele zu DataCo:
Was bei der Titanic Klasse und Geschlecht sind, sind bei Lena Premium-Abo und Nutzungsintensität. Die logistische Regression deckt auf, *welche* Faktoren den Ausgang beeinflussen – und *wie stark*.

Gesellschaftliche Strukturen werden in Daten sichtbar – Titanic zeigt es eindrücklich.

Anwendungsgebiete der Logistischen Regression

- **Churn Prediction:** Welche Kunden kündigen in den nächsten 30 Tagen? → Gezielte Retention
- **Conversion:** Welcher Besucher kauft? → Rabatt-Banner nur bei mittlerer Kaufwahrscheinlichkeit
- **Credit Scoring:** Ausfallwahrscheinlichkeit von Krediten – regulatorisch vorgeschrieben (Basel), weil OR interpretierbar
- **Fraud Detection:** Betrügerische Transaktionen erkennen – bei 0,1% Betrugsrate ist Accuracy nutzlos, AUC entscheidend

Warum trotz moderner ML-Methoden so beliebt?

1. **Interpretierbar:** Odds Ratios erklären jede Variable
2. **Schnell:** Millionen Datenpunkte in Sekunden
3. **Robuste Baseline:** Jedes komplexere Modell muss mindestens so gut sein – sonst lohnt sich die Komplexität nicht

---

Logistische Regression ist die Basis für Klassifikation – einfach, schnell, interpretierbar.

Geschichte: Berkson erfindet "Logit" (1944)

Von Bevölkerungswachstum zur Churn-Prediction:

- **1838 – Verhulst:** Belgischer Mathematiker nutzt die logistische Funktion für Bevölkerungswachstum – die S-Kurve war schon bekannt
- **1934 – Bliss:** Probit-Modell (Normalverteilungs-CDF) – mathematisch korrekt, aber schwer handhabbar
- **1944 – Berkson:** Statistiker an der Mayo Clinic sucht eine einfachere Alternative. Sein Logit-Modell (log +unit, analog zu prob + unit = Probit) liefert fast identische Ergebnisse bei einfacherer Rechnung

Heute: Weltweit meistgenutztes Klassifikationsmodell. Was Berkson für Patienten an der Mayo Clinic entwickelte, nutzt Lena 80 Jahre später für Kundendaten bei DataCo.

Logit = log + unit. Berkson wollte eine praktikable Alternative zum Probit-Modell.

Logistische Regression: Häufige Irrtümer

Irrtum 1: "Logistische Regression ist keine Regression."
Doch – auf der Log-Odds-Skala ist sie linear. Die Koeffizienten beschreiben eine lineare Beziehung zwischen Prädiktoren und Log-Odds. Der Name ist korrekt.

Irrtum 2: "Accuracy ist die beste Metrik."
Bei 5% Churn erreicht "immer Bleibt" 95% Accuracy. Precision, Recall, F1 und AUC-ROC sind fast immer aussagekräftiger.

Irrtum 3: "Odds Ratio = Wahrscheinlichkeitsverhältnis."
$OR = 2{,}5$ heißt *nicht* "2,5-mal so wahrscheinlich". OR operiert auf Odds, nicht auf Wahrscheinlichkeiten. Nur bei seltenen Ereignissen ($p < 0{,}10$) gilt die Näherung $OR \approx RR$.

Faustregel: Wenn jemand "doppelt so wahrscheinlich" sagt, fragen Sie: "Meinen Sie die Odds oder die Wahrscheinlichkeit?"

OR und Relatives Risiko sind nur bei seltenen Ereignissen annähernd gleich.

R-Code: Logistische Regression schätzen

```
# Modell schaetzen
logit_model <- glm(churn ~ premium + alter +
  nutzung_monate + beschwerden,
  data = dataco, family = binomial)

# Odds Ratios mit 95%-KI
exp(cbind(OR = coef(logit_model),
          confint(logit_model)))

# Confusion Matrix
library(caret)
pred <- ifelse(predict(logit_model,
  type = "response") > 0.5, 1, 0)
confusionMatrix(factor(pred), factor(dataco$churn))
```

glm(family = binomial) schätzt logistische Regression. exp(coef) gibt Odds Ratios.

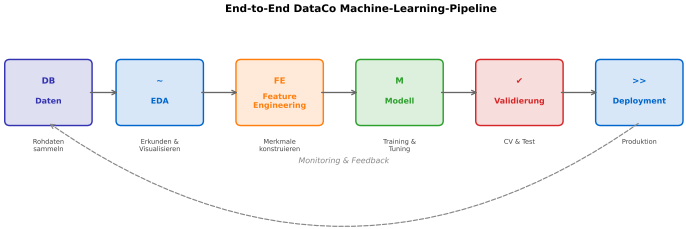
R-Code: ROC-Kurve und AUC

```
# ROC-Kurve und AUC mit pROC
library(pROC)

pred_prob <- predict(logit_model,
                     type = "response")

roc_obj <- roc(dataco$churn, pred_prob)
plot(roc_obj,
     main = "ROC: DataCo Churn-Modell")
auc(roc_obj)
# Lenas Ergebnis: AUC = 0.82
```

`pROC` berechnet die ROC-Kurve für jeden möglichen Schwellenwert und die Fläche darunter (AUC).

AUC = 0,82: In 82% der Fälle rankt Lenas Modell einen Kündiger höher als einen Bleiber.

Die vollständige Data-Science-Pipeline



Lenas Weg durch Block 3:

1. Daten sammeln → 2. Annahmen prüfen (BLUE) → 3. Multikollinearität erkennen (VIF)
2. Regularisieren (Lasso/Ridge) → 5. Validieren (CV) → 6. Kausalität klären
3. Experimentieren (A/B-Test) → 8. Klassifizieren (Logit) → **Entscheidung**

Von den Rohdaten zur Geschäftsentscheidung – das ist die vollständige Pipeline.

Übung: Logistische Regression

Aufgabe 1 – Odds Ratio:
$\beta_{\textbf{Beschwerden}} = 0{,}47$. Berechnen Sie die OR. Ein Kunde hat $p = 0{,}30$ Churn-Wahrscheinlichkeit. Wie hoch ist p nach einer zusätzlichen Beschwerde? Weg: $p \to$ Odds $\to$ neue Odds $\to p$.

Aufgabe 2 – Confusion Matrix:
10.000 Transaktionen, 50 Betrug. Modell: TN=9920, FP=30, FN=10, TP=40. Berechnen Sie Accuracy, Precision, Recall, F1. Warum ist Accuracy irreführend?

Aufgabe 3 – AUC:
Zwei Churn-Modelle: AUC = 0,72 vs. 0,88. Was bedeutet der Unterschied praktisch? Wann lohnt sich das komplexere Modell?

Tipp: $\textbf{Odds} = p/(1-p)$, $\textbf{neue Odds} = \textbf{Odds} \times OR$, $\textbf{neues } p = \textbf{Odds}/(1+\textbf{Odds})$.

Lenas Reise: Zusammenfassung

Acht Kapitel – acht Erkenntnisse:

1. **BLUE & Gauss-Markov:** OLS-Schätzer sind nur unter bestimmten Annahmen optimal
2. **Multikollinearität:** Korrelierte Prädiktoren machen Koeffizienten instabil $\rightarrow$ VIF prüfen
3. **Bias-Varianz-Tradeoff:** Perfekte Anpassung $\neq$ gute Vorhersage $\rightarrow$ Komplexität kontrollieren
4. **Regularisierung:** Lasso/Ridge bestrafen Komplexität $\rightarrow$ robustere Modelle
5. **Kreuzvalidierung:** Out-of-Sample-Performance ist der wahre Maßstab
6. **Kausalität:** Korrelation $\neq$ Kausalität $\rightarrow$ Confounders, DAGs, natürliche Experimente
7. **A/B-Testing:** Randomisierte Experimente sind der Goldstandard für kausale Fragen
8. **Logistische Regression:** Binäre Outcomes brauchen die S-Kurve $\rightarrow$ Odds Ratios interpretieren

Lenas Bilanz: AUC $= 0{,}82$ für Churn-Prediction, A/B-Test beweist $+8$pp Retention, und sie kann dem Board jede Zahl erklären.

Von der deskriptiven Analyse über die Inferenz zur kausalen Schlussfolgerung – das ist Data Science.

Vielen Dank für Ihre Aufmerksamkeit!

Fragen?