Block 3: Komplexität, Kausalität & Generalisierung

Einfach erklärt — Data Science and Strategy for Business

March 31, 2026

Für Einsteigerinnen und Einsteiger — keine Vorkenntnisse nötig

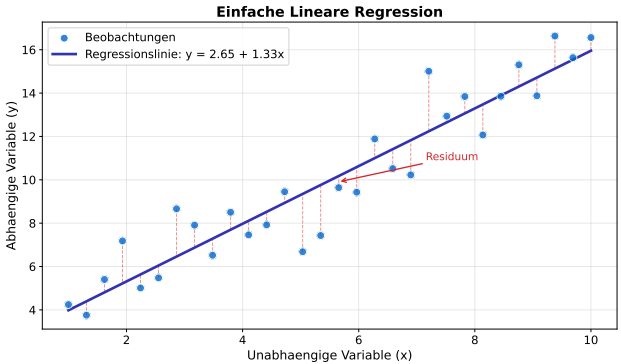
Was lernen wir heute?

Lena ist Datenanalystin. Sie hat ein Modell gebaut — aber es funktioniert nicht.

1. Warum **stures Routenspeichern** dem Modell nicht hilft
2. Wie man ein Modell **fair testet**
3. Warum manche Variablen sich **gegenseitig stören**
4. Wie man echte **Ursachen** von Zufall unterscheidet
5. Wie man mit **Ja/Nein-Fragen** Vorhersagen macht

Alle Konzepte werden mit Alltagsbeispielen erklärt — keine Formeln nötig.

Kurze Erinnerung: Was ist Regression?



Die Idee: Eine Linie durch die Daten legen.

- Jeder Punkt = ein Kunde, ein Produkt, eine Beobachtung
- Die Linie zeigt den **Trend**
- Je näher die Punkte an der Linie, desto besser das Modell

Block 3 fragt: Was kann schiefgehen?

Regression = eine Linie (oder Kurve) durch Datenpunkte legen, um Vorhersagen zu machen.

Lenas Problem: Perfekte Route, falsche Umleitung

Lena baut ein Modell zur Kundenvorhersage.

- Auf den **Trainingsdaten**: 99 % richtig
- Auf **neuen Daten**: nur 52 % richtig

Wie ein Navi, das alle Routen exakt gespeichert hat —
aber bei einer Umleitung komplett versagt.

Das Modell hat die **Routen** gelernt, nicht das **Straßennetz**.

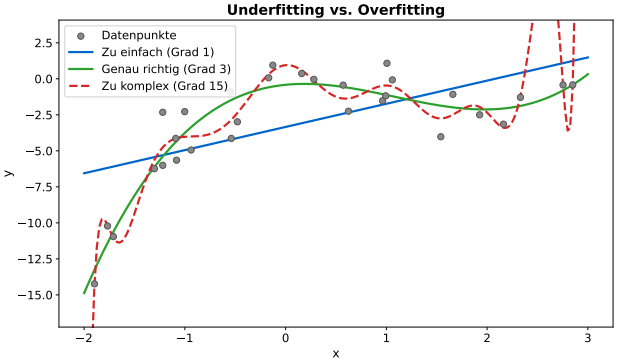
99 % auf Trainingsdaten klingt gut — ist aber ein Warnsignal.

Overfitting: Das Modell lernt Rauschen

Overfitting heißt:

- Das Modell merkt sich jedes Detail
- Auch zufällige Ausreißer und Fehler
- Es funktioniert nur auf alten Daten

Wie ein Navi, das exakte Routen speichert statt das Straßennetz zu verstehen.



Ein zu komplexes Modell passt perfekt auf alte Daten — und versagt bei neuen.

Underfitting: Das Modell ist zu faul

Underfitting ist das Gegenteil:

- Das Modell ist **zu einfach**
- Es erkennt nicht einmal die wichtigsten Muster
- Schlecht auf Trainingsdaten **und** neuen Daten

Wie ein Navi, das nur sagt „Fahr Richtung Norden" —
ohne Straßen oder Abbiegungen zu kennen.

	Trainingsdaten	**Neue Daten**
Underfitting	schlecht	schlecht
Overfitting	sehr gut	schlecht
Genau richtig	gut	gut

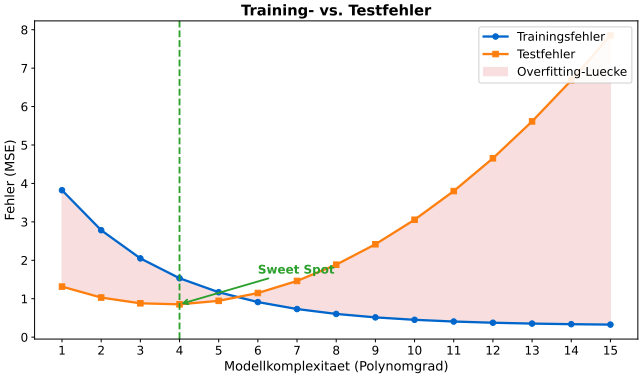
Zu einfach ist genauso schlecht wie zu komplex.

Der Sweet Spot: Genau richtig

Das beste Modell liegt **in der Mitte**:

- Nicht zu einfach (Underfitting)
- Nicht zu komplex (Overfitting)
- **Genau richtig** (Goldilocks)

Wie Goldilocks: Nicht zu heiß, nicht zu kalt.



Die Lücke zwischen Training und Test zeigt, wie stark das Modell nur alte Routen kennt.

Fünf Gegenmittel gegen Overfitting

Was kann Lena tun?

1. **Mehr Daten** — mehr Straßen kennenlernen
2. **Weniger Variablen** — nur die wichtigsten behalten
3. **Regularisierung** — dem Modell eine Leine anlegen
4. **Kreuzvalidierung** — das Modell fair prüfen

Alle fünf Methoden haben ein Ziel: Das Modell soll das Straßennetz verstehen, nicht nur Routen speichern.

Genau oder präzise? Die Zielscheibe

Stell dir eine **Zielscheibe** vor:

- **Genau** = nah am Zentrum
- **Präzise** = eng beieinander
- **Ideal:** beides!

Genau ist nicht das Gleiche wie präzise.

Ein gutes Modell trifft nah am Zentrum UND streut wenig.

Bias: Immer daneben

Bias bedeutet: Das Modell liegt **systematisch** falsch.

- Es trifft immer die **gleiche falsche Stelle**
- Egal wie oft du misst — der Fehler bleibt
- Ein **einfaches Modell** hat oft hohen Bias

Wie eine Waage, die immer 2 kg zu viel anzeigt.
Egal was du draufstellst — sie liegt immer daneben.

Bias = systematischer Fehler. Einfachere Modelle haben mehr davon.

Varianz: Mal so, mal so

Varianz bedeutet: Das Modell ist **instabil**.

- Kleine Änderungen in den Daten — großer Unterschied im Ergebnis
- Mal trifft es perfekt, mal komplett daneben
- Ein **komplexes Modell** hat oft hohe Varianz

Wie ein GPS, das ständig hin und her springt.
Mal zeigt es 50 m links, mal 50 m rechts.

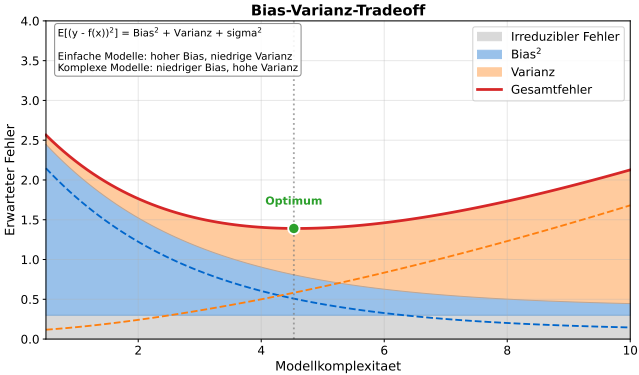
Varianz = Instabilität. Komplexere Modelle haben mehr davon.

Der Tradeoff: Man kann nicht beides haben

Das Dilemma:

- **Einfach** = stabil, aber daneben
- **Komplex** = nah dran, aber wackelig
- **Mittelweg** = am besten

Man muss den Punkt finden, wo der Gesamtfehler am kleinsten ist.



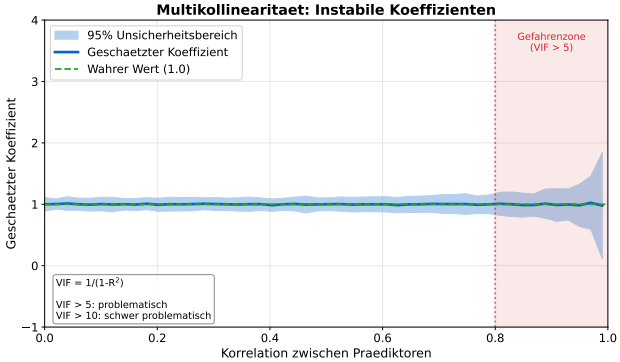
Der Bias-Varianz-Tradeoff erklärt, warum mehr Komplexität nicht immer besser ist.

Eineiige Zwillinge vor Gericht

Stell dir einen **Gerichtsprozess** vor:

- Zwei Zeugen sagen **immer das Gleiche**
- Der Richter kann nicht unterscheiden, wer was beiträgt
- **Multikollinearität** = Variablen, die fast identisch sind

Wie eineiige Zwillinge: Beide sagen dasselbe — der Richter ist verwirrt.



Multikollinearität = Variablen, die so stark zusammenhängen, dass das Modell sie nicht trennen kann.

Was passiert im Modell?

Wenn Variablen fast **identisch** sind:

- Die Gewichte (Koeffizienten) **springen wild hin und her**
- Kleine Datenänderungen — komplett anderes Ergebnis
- Das Modell **kann nicht sagen**, welche Variable wichtig ist

Die Vorhersage kann trotzdem gut sein.
Aber die Erklärung ist unbrauchbar.

Multikollinearität zerstört die Interpretation — nicht unbedingt die Vorhersage.

Was tun? Einen Zwilling entlassen

Drei einfache Lösungen:

1. **Einen entfernen** — eine der ähnlichen Variablen löschen
2. **Zusammenfassen** — aus beiden einen Durchschnitt bilden
3. **Regularisierung** — das Modell selbst regeln lassen

Wenn zwei Zeugen dasselbe sagen: Einen nach Hause schicken.

Regularisierung (nächstes Thema) löst Multikollinearität automatisch.

Wilde Koeffizienten an die Leine

Ohne Kontrolle laufen die Gewichte im Modell **wild**:

Variable	**Gewicht (ohne Leine)**
Alter	+42
Einkommen	−87
Klicks	+63

- Riesige Zahlen = das Modell reagiert **über**
- Kleine Datenänderung = komplett anderes Ergebnis

Wie ein Hund ohne Leine: Er rennt überall hin — auch ins Gebüsch.

Regularisierung = eine Leine für die Koeffizienten. Sie dürfen sich bewegen, aber nicht zu weit.

Ridge: Alle dürfen rein, aber leise

Ridge-Regularisierung funktioniert wie ein Türsteher:

- **Alle** Variablen dürfen ins Modell
- Aber alle müssen sich **benehmen** (kleine Gewichte)
- Kein Gewicht wird ganz auf Null gesetzt

Der Türsteher sagt: „Alle rein, aber benehmt euch!"

Wann Ridge? Wenn alle Variablen **ein bisschen** wichtig sind.

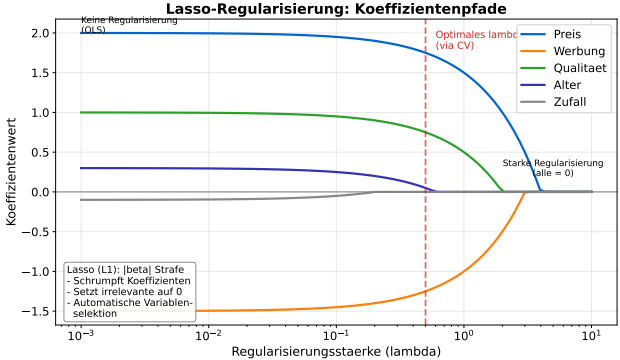
Ridge macht alle Gewichte kleiner — entfernt aber keine Variable komplett.

Lasso: Nur die Wichtigen dürfen rein

Lasso-Regularisierung ist strenger:

- Unwichtige Variablen werden **komplett entfernt**
- Ihr Gewicht wird auf **exakt Null** gesetzt
- Nur die wichtigsten bleiben übrig

Der strenge Türsteher: „Du bist unwichtig — raus!"



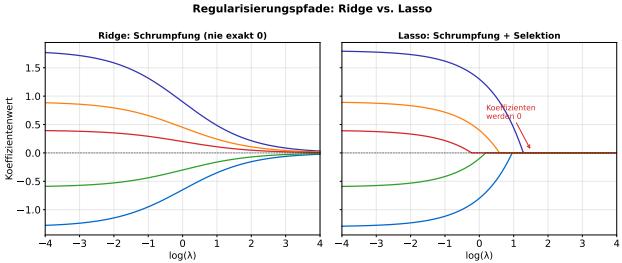
Lasso wählt automatisch die wichtigsten Variablen aus — wie ein Filter.

Vorher/Nachher: Was Lasso entfernt

Lenas Modell vorher und nachher:

Variable	OLS	Lasso
Alter	drin	drin
Premium	drin	drin
PLZ	drin	raus
Browser	drin	raus
Wochentag	drin	raus

Lasso erkennt: PLZ, Browser und Wochentag sind Rauschen.



Lasso findet automatisch die wichtigen Variablen — und entfernt den Rest.

Eine Klausur ist nicht fair

Lena testet ihr Modell — aber das Ergebnis **schwankt**:

Versuch	**Trefferquote**
Test 1	78 %
Test 2	71 %
Test 3	82 %

- Welches Ergebnis ist das **echte**?
- Ein einziger Test ist wie **eine Klausurfrage**
- Hatte Lena Glück oder Pech?

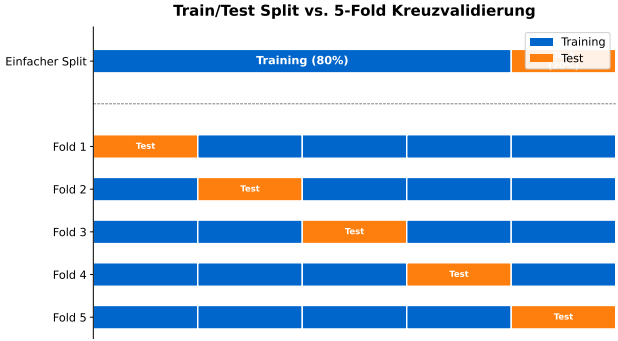
Ein einzelner Test ist nicht zuverlässig — wie eine Klausur mit nur einer Frage.

Rotierende Richter: Jeder kommt dran

Die Lösung: **Kreuzvalidierung** (Cross-Validation).

- Die Daten werden in Teile aufgeteilt
- Jeder Teil ist **einmal** der Test
- Am Ende: **Durchschnitt** aller Ergebnisse

Wie rotierende Richter: Jeder benotet einmal — das ist fairer als nur ein Richter.



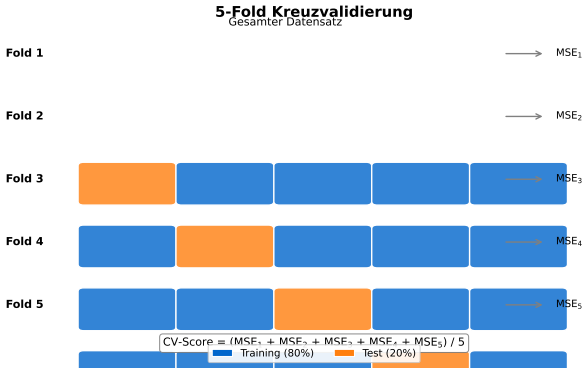
Kreuzvalidierung testet das Modell auf allen Daten — nicht nur auf einem zufälligen Teil.

So funktioniert K-Fold

K-Fold mit 5 Teilen:

- Runde 1: Teil 1 ist Test
- Runde 2: Teil 2 ist Test
- ... bis alle dran waren
- Endergebnis = Durchschnitt

Jeder Datenpunkt wird genau einmal getestet.



K-Fold heißt: K Runden, jede Runde ein anderer Testteil. Standard: K = 5 oder K = 10.

Warum 10 Folds?

Wie viele Runden braucht man?

- **5 Folds** = schnell, etwas ungenau
- **10 Folds** = guter Kompromiss (Standard)
- **Mehr Folds** = genauer, aber langsamer

Wie viele Klausuraufgaben braucht man für eine faire Note?
Eine Frage reicht nicht. Zehn sind besser als fünf.

Faustregel: 10 Folds sind fast immer eine gute Wahl.

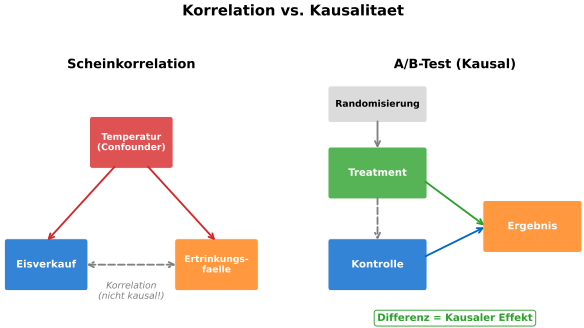
Mehr Folds = fairere Bewertung. 10 ist der Standard in der Praxis.

Eis und Ertrinken

Ein berühmtes Beispiel:

- Mehr Eisverkauf = mehr Ertrinkungsfälle
- Verursacht Eis das Ertrinken?
- **Nein!** Beide steigen im Sommer

Die Temperatur treibt beides.
Eis und Ertrinken haben nichts miteinander zu tun.



Korrelation (Zusammenhang) ist nicht Kausalität (Ursache). Der häufigste Fehler in der Datenanalyse.

Der versteckte Dritte: Confounding

Hinter vielen falschen Zusammenhängen steckt ein **Puppenspieler**:

- Eine **versteckte Variable** beeinflusst beide sichtbaren
- Man sieht den Zusammenhang — aber nicht die Ursache
- Der Fachbegriff: **Confounding** (Störvariable)

Wie ein Puppenspieler hinter dem Vorhang:
Er zieht an beiden Fäden — aber man sieht nur die Puppen.

Lenas Beispiel: Marketing und Kundenbindung steigen beide —
aber der **Wirtschaftsboom** treibt beides.

Confounding = eine versteckte Variable, die den echten Zusammenhang verzerrt.

Drei Wege zur falschen Schlussfolgerung

Warum sieht man Zusammenhänge, die **keine Ursachen** sind?

1. **Versteckte dritte Variable**
   Eis und Ertrinken — die Temperatur steckt dahinter
2. **Umgekehrte Richtung**
   Kranke nehmen Medizin — Medizin verursacht nicht Krankheit
3. **Reiner Zufall**
   Scheidungsrate in Maine korreliert mit Margarineverbrauch

Daten zeigen Zusammenhänge. Nur Experimente zeigen Ursachen.

Drei Fallen: versteckte Variable, falsche Richtung, Zufall. Alle drei sehen in Daten gleich aus.

Wie beweist man Kausalität?

Der **Goldstandard** ist das Experiment:

- Zwei zufällige Gruppen bilden
- Nur **eine Sache** ändern
- Unterschied messen

Wenn kein Experiment möglich ist, braucht man mindestens:

1. Die Ursache kommt **zeitlich vorher**
2. Kein versteckter **Dritter** erklärt den Effekt
3. Ein **plausibler Mechanismus** existiert

Nur Experimente beweisen Kausalität sicher. Alles andere ist Indizien.

Der Geschmackstest

Kennst du den **Pepsi-vs-Coca-Cola-Test**?

- Beide Getränke — **blind** testen
- Niemand weiß, welches welches ist
- Am Ende: Welches schmeckt besser?

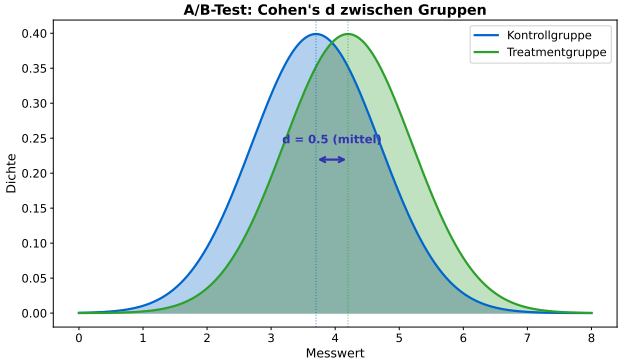
Zwei Varianten, zufällig zugewiesen, Ergebnis messen.

A/B-Test = Experiment mit zwei Gruppen. Die einfachste Art, Kausalität zu zeigen.

Vier Schritte zum A/B-Test

1. **Aufteilen**: Kunden zufällig in Gruppe A und B
2. **Ändern**: Nur **eine** Sache in Gruppe B
3. **Messen**: Ergebnis in beiden Gruppen
4. **Vergleichen**: Ist der Unterschied echt?

Wie ein Kochexperiment:
Nur eine Zutat ändern — sonst weißt du nicht, was gewirkt hat.



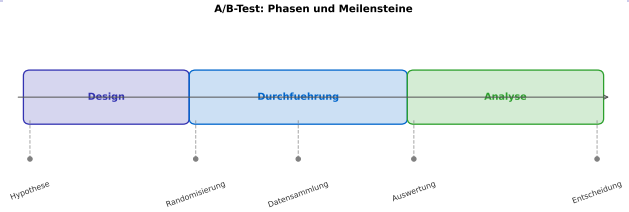
Die goldene Regel: Nur eine Sache ändern. Sonst weiß man nicht, was den Unterschied macht.

Lenas A/B-Test: Das Ergebnis

Lena testet das neue Onboarding:

- **Gruppe A** (alt): 68 % bleiben
- **Gruppe B** (neu): 76 % bleiben
- Unterschied: **+8 Prozentpunkte**

Ist das Zufall oder echt? *Der Unterschied ist **statistisch signifikant**.*



Signifikant heißt: Der Unterschied ist sehr wahrscheinlich echt — nicht nur Glück.

Drei Fallen beim A/B-Testen

Auch A/B-Tests können **schiefgehen**:

1. **Zu früh schauen** (Peeking)
   Ergebnisse schwanken am Anfang — man muss warten
2. **Zu viele Dinge gleichzeitig testen**
   Wenn du 20 Varianten testest, ist eine zufällig „besser"
3. **Zu wenig Testpersonen**
   Mit 10 Leuten kann man keinen kleinen Unterschied finden

Geduld, Fokus und genug Daten — das braucht ein guter Test.

Die drei häufigsten Fehler: zu früh schauen, zu viel testen, zu wenig Leute.

Kündigt der Kunde? Ja oder Nein

Lena will vorhersagen: **Bleibt der Kunde oder geht er?**
Das Ergebnis ist **Ja** oder **Nein**. Normale Regression gibt **unmögliche Antworten**:

Vorhersage	Problem
–12 % Kündigungsrisiko	Geht nicht: unter 0 %
135 % Kündigungsrisiko	Geht nicht: über 100 %

Wir brauchen etwas zwischen 0 % und 100 %.

Für Ja/Nein-Fragen braucht man eine spezielle Methode: die logistische Regression.

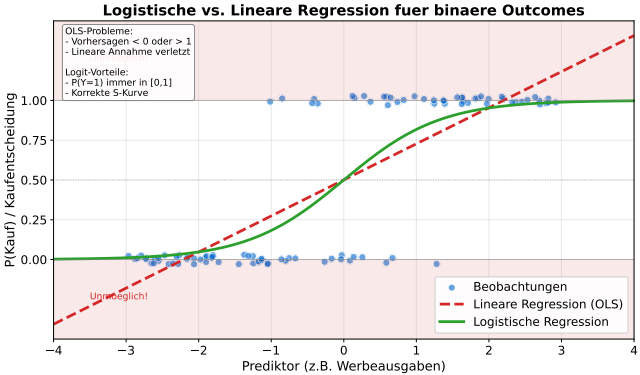
Die S-Kurve: Dimmer statt Schalter

Ein **Lichtschalter**: An oder Aus.

Ein **Dimmer**: Von 0 % bis 100 %.

Die logistische Regression ist wie ein Dimmer:

- Wenig Risikofaktoren = nahe **0 %**
- Viele Risikofaktoren = nahe **100 %**
- Die S-Kurve zeigt den sanften Übergang



Die S-Kurve hält die Vorhersage immer zwischen 0 % und 100 % — egal wie extrem die Daten sind.

Lenas Churn-Modell: Wer kündigt?

Lenas Ergebnis in einfacher Sprache:

- **Premium-Kunden** kündigen seltener
- Kunden mit **vielen Beschwerden** kündigen häufiger
- **Lange Vertragsdauer** schützt vor Kündigung

Das Modell sagt nicht Ja oder Nein.
Es sagt: „Dieser Kunde hat 73 % Risiko zu kündigen."

Lenas Empfehlung: Kunden mit Risiko über 60 % gezielt ansprechen.

Logistische Regression gibt Wahrscheinlichkeiten — nicht nur Ja oder Nein.

Lenas Reise: Was wir gelernt haben

1. **Overfitting** = gespeicherte Routen helfen nicht bei Umleitungen
2. **Bias** = systematisch daneben, **Varianz** = instabil
3. **Multikollinearität** = ähnliche Variablen verwirren das Modell
4. **Regularisierung** = eine Leine für wilde Koeffizienten
5. **Kreuzvalidierung** = faire Prüfung mit rotierenden Tests
6. **Korrelation** ist nicht **Kausalität**
7. **A/B-Tests** beweisen, was wirklich wirkt
8. **Logistische Regression** beantwortet Ja/Nein-Fragen

Acht Konzepte — jedes mit einer Alltagsanalogie. Die Formeln stehen im Anhang.

Vielen Dank!

Block 3: Komplexität, Kausalität & Generalisierung

Für Neugierige:
Im Anhang finden Sie alle Formeln und R-Code-Beispiele.

Data Science and Strategy for Business

Fragen? Jederzeit melden. Die Konzepte sind wichtiger als die Formeln.

Anhang: Für Neugierige

Formeln, Algorithmen und R-Code

Die folgenden Folien vertiefen die Konzepte aus dem Hauptteil.

Anhang: BLUE – Best Linear Unbiased Estimator

Was bedeuten die vier Buchstaben?

Buchstabe	Bedeutung	Erklärung
B	Best	Kleinste Varianz unter allen linearen, erwartungstreuen Schätzern
L	Linear	Schätzer ist Linearkombination: $\hat{\beta} = \sum_i w_i \cdot y_i$
U	Unbiased	Erwartungstreu: $E[\hat{\beta}] = \beta$ (im Mittel korrekt)
E	Estimator	Funktion, die aus Daten (X, y) den Parameter $\hat{\beta}$ berechnet

Waage-Analogie:

- **Genau** (Unbiased): Zeigt im Durchschnitt das richtige Gewicht
- **Stabil** (Best): Schwankt am wenigsten von Messung zu Messung
- **Beste Waage** = genau UND am stabilsten

$$\text{MSE} = \text{Bias}^2 + \text{Varianz}. \qquad \text{BLUE minimiert Varianz bei Bias} = 0.$$

BLUE gilt nur unter den 5 Gauss-Markov-Annahmen – siehe nächste Folie.

Anhang: Gauss-Markov – Fünf Annahmen

Alle fünf müssen gelten, damit OLS = BLUE:

Nr.	Annahme	Bedeutung	Prüfbar?
1	Linearität in β	$y = X\beta + \varepsilon$	Residuenplot
2	Zufällige Stichprobe	Beobachtungen unabhängig	Studiendesign
3	Keine perfekte Multikollinearität	Kein x_j als Linearkomb. der anderen	VIF
4	$E[\varepsilon_i \mid X] = 0$	Fehler im Mittel null	Residuenplot
5	$\text{Var}(\varepsilon_i \mid X) = \sigma^2$	Konstante Fehlervarianz	Breusch-Pagan

Wenn erfüllt:

- `lm()` liefert optimale Schätzungen
- Keine bessere lineare Methode möglich

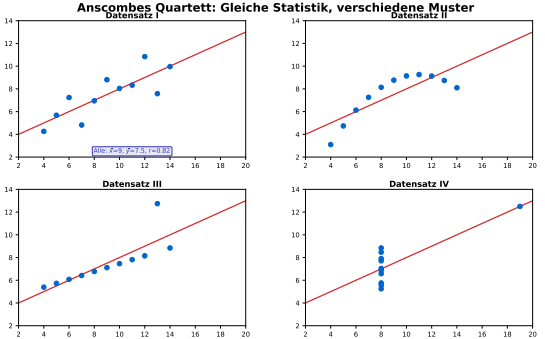
Häufige Irrtümer:

- Normalverteilung ist NICHT Teil von Gauss-Markov
- Normalität braucht man erst für p-Werte und Konfidenzintervalle

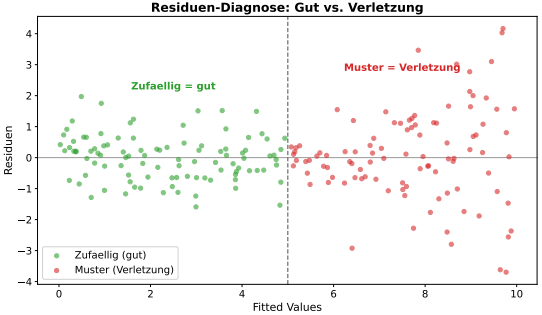
Annahmen 1–3: Datenstruktur (vor Modellierung prüfbar). Annahmen 4–5: Fehlerstruktur (Residuenanalyse).

Verletzte Annahmen ändern nicht die Schätzung, aber machen die Inferenz unzuverlässig.

Anhang: Diagnostik – Anscombe-Quartett



Anscombe (1973): 4 Datensätze mit gleicher Statistik ($\bar{x}$, $\bar{y}$, r, Regressionslinie) – aber völlig verschiedene Muster.
Kennzahlen allein reichen nie!



Residuenplots erkennen Probleme:

- U-Form → Linearität verletzt
- Trichter → Heteroskedastizität
- Kein Muster → alles OK

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

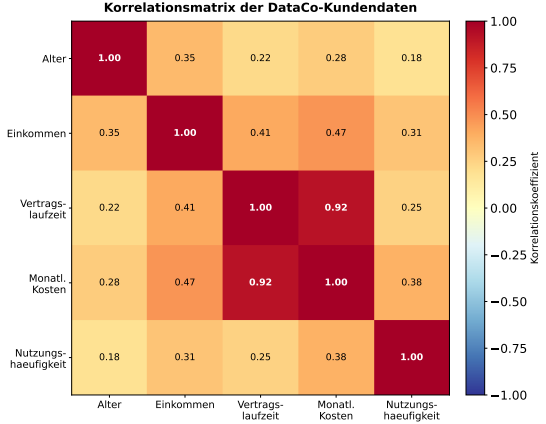
Anhang: VIF – Formel und Schwellenwerte

VIF-Formel: Regressiere X_j auf alle anderen Prädiktoren:

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

- $R_j^2 = 0$: $VIF = 1$ (keine Inflation)
- $R_j^2 = 0{,}9$: $VIF = 10$ (10-fach aufgebläht)
- $R_j^2 \to 1$: $VIF \to \infty$

VIF	$\sqrt{VIF}$	Bewertung
1	1,0×	ideal
5	2,2×	problematisch
10	3,2×	Handlung nötig



Heatmap lesen: Dunkelrot = hohe Korrelation zwischen Variablen. Die Heatmap zeigt *welche* Paare korrelieren. VIF quantifiziert die Auswirkung auf die Standardfehler.

$\sqrt{VIF}$ = Faktor, um den der Standardfehler aufgebläht wird.

VIF $>$ 5 verdient Aufmerksamkeit. $\sqrt{VIF}$ zeigt direkt den SE-Aufblähungsfaktor.

Anhang: Ridge, Lasso und Elastic Net – Formeln

Kernidee: Ein Strafterm bestraft große Koeffizienten.

Methode	Zielfunktion	Eigenschaft
Ridge (L2)	$\min_\beta \sum(y_i - \hat{y}_i)^2 + \lambda \sum \beta_j^2$	Schrumpft alle
Lasso (L1)	$\min_\beta \sum(y_i - \hat{y}_i)^2 + \lambda \sum \lvert\beta_j\rvert$	Schrumpft + selektiert
Elastic Net	$\min_\beta \text{RSS} + \lambda\left[\alpha \sum \lvert\beta_j\rvert + (1-\alpha) \sum \beta_j^2\right]$	Kombination L1+L2

Zwei Teile:

- **RSS** (Residual Sum of Squares): Wie gut passt das Modell? Allein = OLS.
- **Strafterm**: Große β werden „teuer". Je größer λ, desto stärker die Strafe.

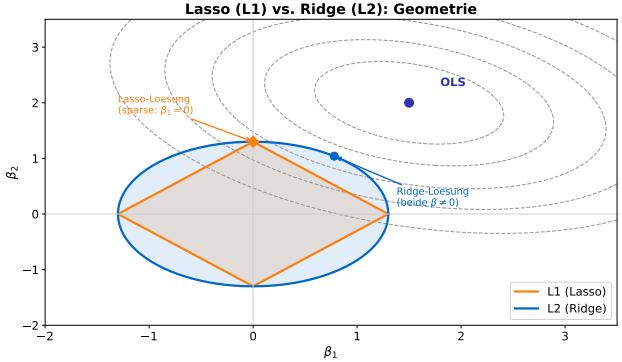
Lambda-Effekt:

- $\lambda = 0$: Keine Strafe = OLS
- λ mittel: Sweet Spot
- $\lambda \to \infty$: Alle $\beta \to 0$

Elastic Net: $\alpha \in (0, 1)$. Korrelierte Feature-Gruppen werden gemeinsam ausgewählt.

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

Anhang: Geometrie – Warum Lasso Nullen erzeugt



Die Ellipsen zeigen die RSS-Höhenlinien (OLS-Lösung im Zentrum). Der Strafbereich begrenzt die erlaubten Koeffizienten.

Ridge (Kreis = L2):

- Keine Ecken → Berührpunkt nie auf einer Achse
- Kein β_j wird exakt null
- Behält *alle* Features

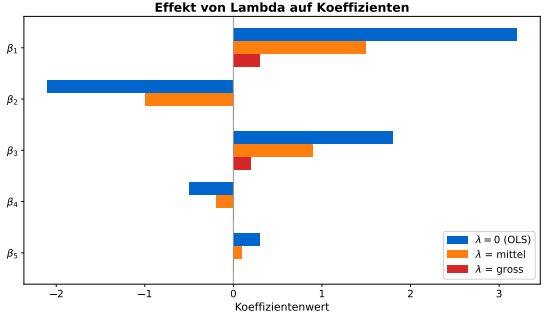
Lasso (Diamant = L1):

- Ecken liegen auf den Achsen
- Ellipsen treffen fast immer eine Ecke
- → β_j wird exakt null
- Automatische Variablenselektion

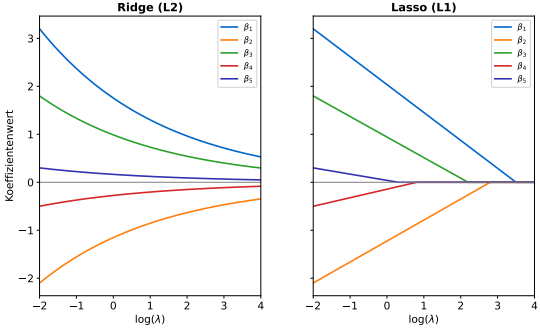
Merke: Die Form des Strafbereichs bestimmt, ob Features eliminiert werden.

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

Anhang: Koeffizientenpfade und Lambda-Effekt



Lambda-Effekt: Das Feature, das am längsten überlebt, ist das wichtigste. Unwichtige verschwinden zuerst.



Vergleich der Pfade:

- **Ridge:** Sanfte, stetige Schrumpfung. Kein β exakt null. Alle Features bleiben.
- **Lasso:** Koeffizienten springen auf null. Scharfe Knicke = Selektionspunkte.

Diagnose: Ridge-Pfade zeigen relative Wichtigkeit. Lasso-Pfade zeigen, welche Features bei welchem λ eliminiert werden. Die Reihenfolge des Verschwindens ergibt ein Feature-Ranking.

Koeffizientenpfade zeigen die Rangfolge der Feature-Wichtigkeit über den gesamten λ-Bereich.

Anhang: K-Fold CV – Algorithmus und Formel

Algorithmus in 3 Schritten:

1. **Aufteilen:** Mische die Daten zufällig und teile sie in K gleich große Folds
2. **Rotieren:** Für jeden Fold $k = 1, \ldots, K$: Trainiere auf den $K-1$ anderen Folds, teste auf Fold k, speichere den Fehler e_k
3. **Mitteln:** Berechne den CV-Schätzer als Durchschnitt aller Fold-Fehler

$$\widehat{\mathrm{CV}} = \frac{1}{K} \sum_{k=1}^{K} e_k$$

Standardabweichung:

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

Wahl von K:

K	**Eigenschaft**
$K = 5$	Weniger Aufwand, etwas mehr Bias
$K = 10$	Standard (Breiman 1992)
$K = n$	LOOCV: Min. Bias, max. Varianz

Kleine $\mathrm{SD}_{\mathrm{CV}}$ bedeutet: das Modell verhält sich stabil über verschiedene Datenaufteilungen.

Standard: K=10. Kleine $\mathrm{SD}_{\mathrm{CV}}$ = robustes Modell. Große $\mathrm{SD}_{\mathrm{CV}}$ = instabil.

Anhang: CV-Varianten im Überblick

Variante	K	Wann einsetzen?	Details
K-Fold	5 / 10	Standardfall	Jeder Punkt genau $1\times$ im Testset
LOOCV	n	Kleine Daten ($n < 50$)	n Modelle trainiert; hohe Varianz
Stratified	5 / 10	Unbalancierte Klassen	Klassenverteilung in jedem Fold beibehalten
Nested	äußere + innere	Hyperparameter-Tuning	Äußere: Performance; innere: λ-Wahl
Time Series	variabel	Zeitlich sortierte Daten	Nur Vergangenheit zum Trainieren

Entscheidungsbaum:

- Zeitreihen? $\rightarrow$ **Time Series CV** (keine Zukunft im Training)
- Unbalancierte Klassen (z.B. 5% Kündiger)? $\rightarrow$ **Stratified**
- Hyperparameter zu tunen (λ, α)? $\rightarrow$ **Nested CV**
- Sehr wenig Daten ($n < 50$)? $\rightarrow$ **LOOCV**
- Sonst: K=10 **Standard-CV**

Achtung: Ohne Nested CV bei Hyperparameter-Tuning entsteht Optimismus-Bias!

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

Anhang: Pearls Leiter und do-Operator

Drei Stufen kausalen Denkens (Judea Pearl, Turing Award 2011):

Stufe	Name	Frage	Beispiel (Lena)
1	**Sehen**	Was beobachte ich?	Kunden, die Mails öffnen, kündigen seltener
2	**Tun**	Was passiert, wenn ich eingreife?	Wenn ich mehr Mails schicke, kündigen weniger?
3	**Vorstellen**	Was wäre gewesen, wenn...?	Hätte der Kunde ohne Mail gekündigt?

Der do-Operator:

$$P(Y \mid X{=}x) \neq P(Y \mid \mathrm{do}(X{=}x))$$

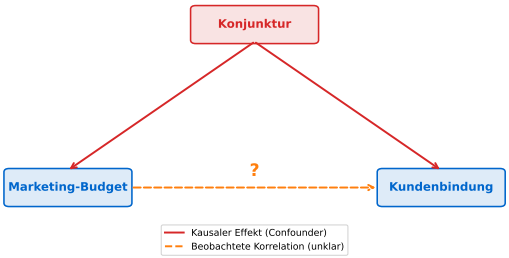
- $P(Y \mid X{=}x)$: Beobachtung – was passiert, wenn wir $X{=}x$ *sehen*?
- $P(Y \mid \mathrm{do}(X{=}x))$: Intervention – was passiert, wenn wir $X{=}x$ *erzwingen*?
- $\mathrm{do}()$ kappt alle kausalen Einflüsse auf X und setzt es direkt

Warum wichtig? ML-Modelle lernen Stufe 1 (Korrelation). Für Entscheidungen brauchen wir Stufe 2 (Kausalität).

Pearls Leiter: Sehen (Korrelation) → Tun (Intervention) → Vorstellen (Kontrafaktisch).

Anhang: DAGs – Kausale Graphen

Confounder-Struktur: Kausales DAG-Beispiel



DAG = Directed Acyclic Graph. Pfeile zeigen kausale Richtung. Keine Zyklen erlaubt.

Drei Bausteine:

- **Fork**: $X \leftarrow Z \rightarrow Y$ – Z kontrollieren!
- **Kette**: $X \rightarrow M \rightarrow Y$ – M **nicht** kontrollieren
- **Collider**: $X \rightarrow C \leftarrow Y$ – C **nicht** kontrollieren!

Kausale Methoden ohne Experiment:

Methode	Kernidee
Kontrollvariablen	Confounder ins Modell
Matching	Vergleichbare Paare bilden
IV	Exogene Variation nutzen
DiD	Vorher/Nachher + Kontrollgruppe

Lenas Puzzle:

1. Roh: Marketing $\leftrightarrow$ Retention: $r = 0{,}45$
2. Konjunktur korreliert mit beiden
3. Partiell: $r = 0{,}12$ (n.s.!)

Fazit: Scheinkorrelation durch Confounder.

DAGs machen kausale Annahmen explizit. Jede Methode hat eine kritische Annahme.

Anhang: Stichprobengröße und Cohen's d

Sample-Size-Formel (pro Gruppe):

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

Standardwerte:

- $\alpha = 0{,}05 \rightarrow z_{\alpha/2} = 1{,}96$
- Power $= 0{,}80 \rightarrow z_{\beta} = 0{,}84$

Lenas Berechnung:

$d = 0{,}3 \rightarrow n \approx \frac{2 \cdot (1{,}96 + 0{,}84)^2}{0{,}3^2} = \frac{2 \cdot 7{,}84}{0{,}09} \approx 174$ pro Gruppe.

Cohen's d – Effektstärke:

$$d = \frac{\bar{X}_T - \bar{X}_C}{s_p}$$

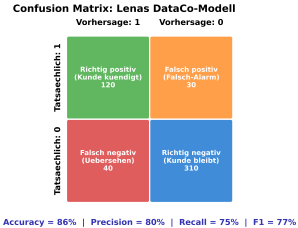
mit $s_p = \sqrt{\frac{(n_T - 1)s_T^2 + (n_C - 1)s_C^2}{n_T + n_C - 2}}$

d	**Interpretation**
0,2	Klein
0,5	Mittel
0,8	Groß

Achtung: d ist unabhängig von n. Bei 1 Mio. Beobachtungen wird selbst $d = 0{,}01$ signifikant!

Power = Wahrscheinlichkeit, einen echten Effekt zu erkennen. Effektstärke *vorher* festlegen!

Anhang: Confusion Matrix und Metriken



Vier Metriken mit Formeln:

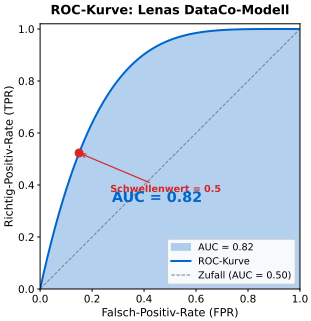
Metrik	Formel	Lena
Accuracy	$\frac{TP+TN}{N}$	91,2%
Precision	$\frac{TP}{TP+FP}$	69,1%
Recall	$\frac{TP}{TP+FN}$	74,7%
F1	$\frac{2 \cdot P \cdot R}{P+R}$	71,8%

Lenas Modell:

	Bleibt	Kündigt
Vorhersage: Bleibt	400 (TN)	19 (FN)
Vorhersage: Kündigt	25 (FP)	56 (TP)

Bei unbalancierten Daten: Precision und Recall wichtiger als Accuracy. FN = teuerster Fehler.

Anhang: ROC-Kurve und AUC



ROC = Receiver Operating Characteristic.

Was zeigt die Kurve?

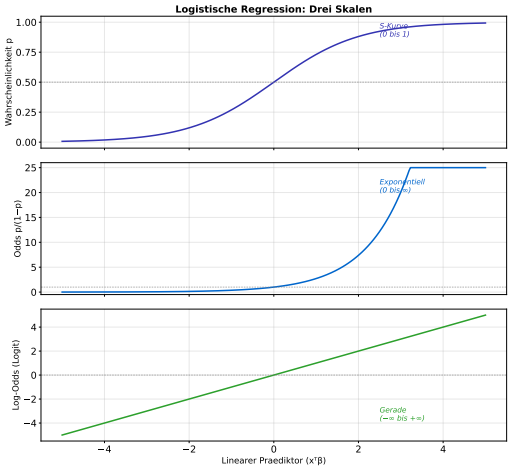
- x-Achse: False Positive Rate (FPR)
- y-Achse: True Positive Rate (TPR)
- Diagonale = Zufall; links oben = besser

AUC-Werte:

AUC	Bewertung
0,5	Zufall
0,7–0,9	Gut
$> 0{,}9$	Exzellent

AUC = Fläche unter der ROC-Kurve. Lenas Modell: AUC = 0,82 (gut).

Anhang: Odds Ratios und Log-Odds



Drei Perspektiven auf dasselbe Modell:

Skala	Eigenschaft
P	$(0, 1)$, S-förmig, nicht-linear
Odds $= \frac{P}{1-P}$	$(0, \infty)$, multiplikativ
Log-Odds $= \ln \frac{P}{1-P}$	$(-\infty, +\infty)$, **linear**

Logit-Modell: $\ln \frac{P}{1-P} = \beta_0 + \beta_1 X$

Odds Ratio: $OR = e^{\beta_1}$

- $OR = 1$: kein Effekt
- $OR > 1$: erhöht die Odds
- $OR < 1$: verringert die Odds

Drei Skalen, ein Zusammenhang.

Achtung: $OR \neq$ Relatives Risiko. Nur bei seltenen Ereignissen ($p < 0{,}10$) gilt $OR \approx RR$.

Log-Odds-Skala: Regression ist linear. $\exp(\beta)$ **liefert interpretierbare Odds Ratios.**

Anhang: R-Code – BLUE und Diagnostik

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

# 4 Diagnose-Plots: Residuen, QQ, Scale-Location, Leverage
par(mfrow = c(2, 2))
plot(model)

# Breusch-Pagan-Test auf Heteroskedastizität
library(lmtest)
bptest(model)       # p < 0.05 -> Heteroskedastizität!

# VIF berechnen (Multikollinearität)
library(car)
vif(model)          # VIF > 5: problematisch

# 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")
```

Drei Checks: `plot(model)` **für Residuen,** `bptest()` **für Varianz,** `vif()` **für Kollinearität.**

Anhang: R-Code – Regularisierung und CV

```
# --- Lasso mit glmnet ---
library(glmnet)
X <- model.matrix(churn_score ~ ., data = df)[, -1]
y <- df$churn_score

# Lambda per Kreuzvalidierung wählen
cv_lasso <- cv.glmnet(X, y, alpha = 1, nfolds = 10)
cat("Optimales Lambda:", round(cv_lasso$lambda.min, 4))
coef(cv_lasso, s = "lambda.min")   # Welche Features bleiben?

# Ridge zum Vergleich (alpha = 0)
cv_ridge <- cv.glmnet(X, y, alpha = 0, nfolds = 10)

# --- K-Fold CV mit caret ---
library(caret)
ctrl <- trainControl(method = "cv", number = 10)
model_cv <- train(churn_score ~ ., data = df,
                  method = "lm", trControl = ctrl)
print(model_cv$results)   # RMSE, R-squared, MAE

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

cv.glmnet wählt λ per CV. caret::train vergleicht Modelle. alpha: 0 = Ridge, 1 = Lasso.

Anhang: R-Code – Kausalität, A/B-Test und Logit

```
# --- Korrelation und partielle Korrelation ---
cor.test(dataco$marketing, dataco$retention)  # r=0.45
library(ppcor)
pcor.test(dataco$marketing, dataco$retention,
          dataco$konjunktur)   # r=0.12, p=0.38 (n.s.!)

# --- A/B-Test: t-Test und Effektstärke ---
t.test(retention ~ gruppe, data = ab_daten)
library(effsize)
cohen.d(retention ~ gruppe, data = ab_daten)  # d=0.45
power.t.test(delta = 0.3, sd = 1, sig.level = 0.05,
             power = 0.80, type = "two.sample")  # n=176

# --- Logistische Regression ---
logit <- glm(churn ~ premium + alter + nutzung_monate
             + beschwerden, data = dataco, family = binomial)
exp(cbind(OR = coef(logit), confint(logit)))  # Odds Ratios
library(pROC)
auc(roc(dataco$churn, predict(logit, type = "response")))
```

cor.test: Korrelation. t.test: Gruppenvergleich. glm(family=binomial): Logistische Regression.