Trainieren, prüfen, bestehen

Warum man ein Modell nie mit seinen Übungsaufgaben prüft: Teile die Daten selbst auf, trainiere live — und erlebe, was der „Schummeln“-Knopf wirklich kostet.

Kapitel online

Fortgeschritten · Kapitel 23 von 50 · ca. 10 Min. · 250 XP

Einfach gesagt: Eine Klausur mit den Übungsaufgaben von gestern besteht jeder — deshalb prüfst du dein Modell nur mit Daten, die es nie gesehen hat.

Stell dir eine Klausur vor, in der **exakt die Übungsaufgaben von gestern** drankommen. Jeder besteht — und keiner hat etwas bewiesen. Genau diesen Fehler machen Anfänger mit ihren Modellen ständig: Sie prüfen mit den Daten, mit denen sie trainiert haben. Bevor wir uns Overfitting ansehen, brauchst du deshalb dieses Handwerkszeug: **Daten richtig aufteilen.**

Was du hier lernst

- Warum du ein Modell **nie mit den Trainingsdaten** bewerten darfst
- Die **drei Töpfe** Training, Validierung und Test — und wer wofür zuständig ist
- Warum der **Test-Topf** genau EINMAL angefasst wird
- Woran du **Data Leakage** erkennst: Skalieren über alles, Dubletten, Zeitreisen
- Dass das Aufteilen selbst nur **Tensor-Zerschneiden** ist — zwei Schnitte, fertig

Für den Unterricht

Einstieg (2 Min): Kapitel gemeinsam lesen, Alltagsbeispiel aufgreifen.

Erkunden am Gerät (6 Min): die interaktiven Module auf herrki.de/235_daten_aufteilen ausprobieren.

Sichern mit Quiz (2 Min): „Kurzer Check" besprechen, „Das nimmst du mit" festhalten.

Drei Töpfe, drei Aufgaben

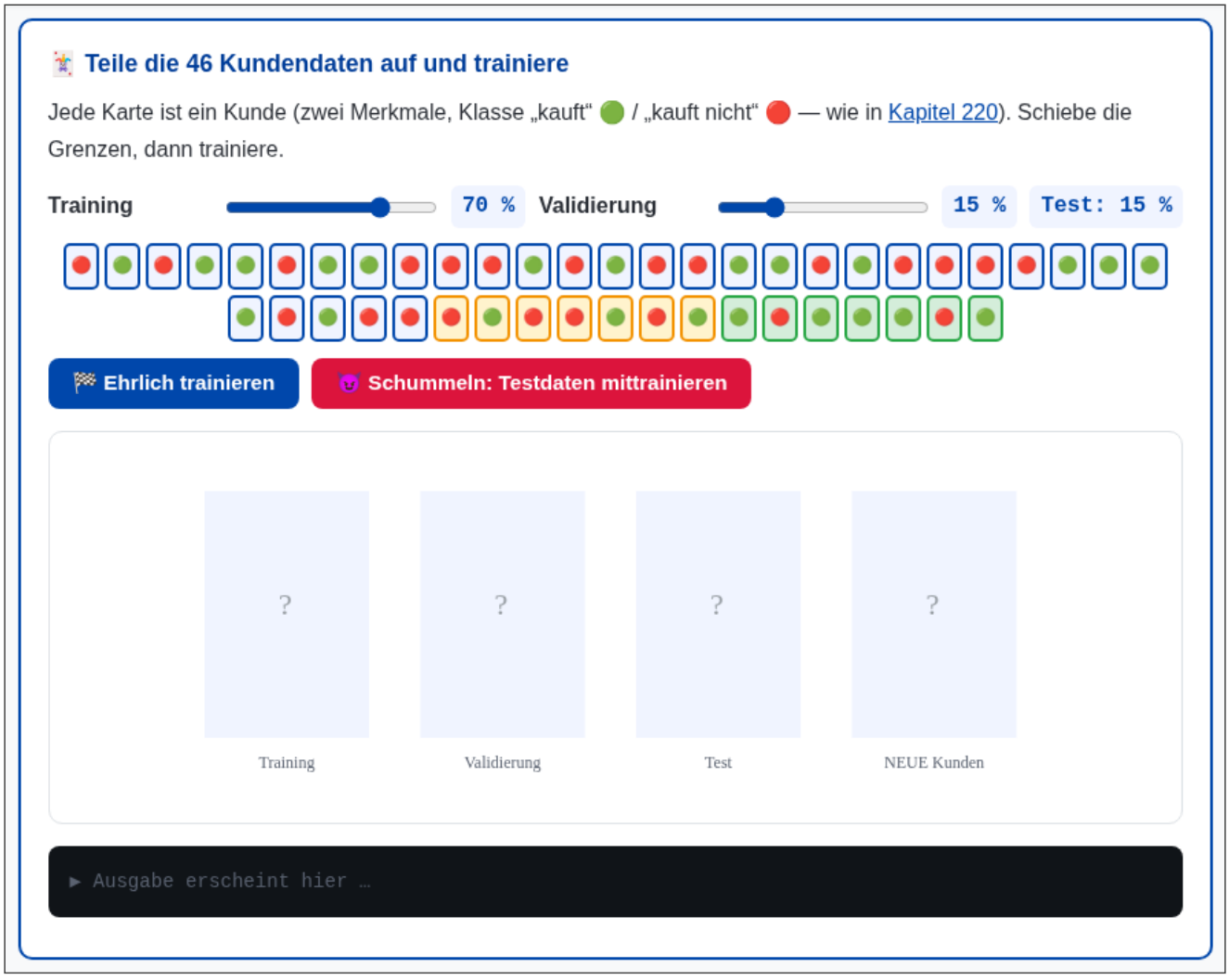
1**Training (~70 %)** — die Übungsaufgaben. NUR hiermit lernt das Modell.

2**Validierung (~15 %)** — die Probeklausur. Hiermit vergleichst du Varianten und entscheidest, wann Schluss ist.

3**Test (~15 %)** — die echte Prüfung. Wird **genau EINMAL** angefasst, ganz am Ende. Wer mehrfach auf den Test schaut und danach am Modell dreht, hat ihn zur zweiten Validierung gemacht — und keine ehrliche Zahl mehr.

Am Gerät ausprobieren: herrki.de/235_daten_aufteilen#abschnitt-2

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

Data Leakage — Schummeln aus Versehen

Niemand schummelt absichtlich. In der Praxis passiert es **versehentlich**, zum Beispiel so:

⚠ **Skalieren über ALLE Daten:** Du berechnest min/max über den kompletten Datensatz und teilst erst danach — schon sind Informationen aus dem Test ins Training gesickert (die Regel aus Kapitel 135!).

⚠ **Doppelte Datensätze:** Derselbe Kunde landet in Training UND Test.

⚠ **Zeitreisen:** Du sagst den Aktienkurs von Montag vorher — mit Daten, die es erst am Dienstag gab.

Runter zum Tensor

Das Aufteilen selbst ist unspektakulär — Tensor-Zerschneiden (Kapitel 060), nach dem Mischen. Deine 46 Kunden von oben liegen als Merkmals-Tensor [46, 2] mit den Klassen [46, 1] daneben — eine Zeile pro Kunde (Beispielzeilen):

```
// zwei Merkmale, auf 0–1 skaliert (wie in Kapitel 220), plus die Klasse
             Merkmal 1 Merkmal 2 kauft?
Zeile 0 0.62 0.58 1 ●
Zeile 1 0.31 0.44 0 ●
Zeile 2 0.68 0.71 1 ●
   ▯
Zeile 45 0.39 0.35 0 ●
```

Der Schnitt bei 70/15/15 ist dann schlichtes Kopfrechnen auf der Zeilenzahl:

```
46 × 70 % = 32.2 → 32 Zeilen Training (Zeile 0 – 31)
46 × 15 % = 6.9 → 7 Zeilen Validierung (Zeile 32 – 38)
Rest → 7 Zeilen Test (Zeile 39 – 45)
```

Und genau diese drei Zahlenpaare stehen im Code — `slice` nimmt Startzeile und Anzahl:

```
// 46 Kunden, gemischt. 70/15/15 aufteilen = einfach den Tensor zerschneiden:
const training = daten.slice(0, 32); // Zeile 0-31
const validierung = daten.slice(32, 7); // Zeile 32-38
const test = daten.slice(39, 7); // Zeile 39-45 – anfassen: EINMAL
```

Mehr ist es nicht: Ein Schnitt an Zeile 32, einer an Zeile 39 — und aus „Übungsaufgaben“ werden „Probeklausur“ und „Prüfung“.

Zurück zur Anwendung

Ab jetzt liest du jede Modell-Meldung anders: „99 % Genauigkeit“ ist so lange Werbung, bis dabei steht, **auf welchem Topf** sie gemessen wurde. Und im nächsten Kapitel wirst du sehen, wie die Trainings- und die Validierungs-Kurve auseinanderlaufen — jetzt weißt du, wer diese beiden Linien sind.

Zum Weiterlesen

- **Die drei Töpfe Schritt für Schritt:** Google Machine Learning Crash Course, *Dividing the original dataset* — developers.google.com → Dividing the dataset (developers.google.com/machine-learning/crash-course/overfitting/dividing-datasets)
- **Data Leakage in der Praxis vermeiden:** scikit-learn, *Common pitfalls and recommended practices* (mit Code-Beispielen zum falschen und richtigen Skalieren) — scikit-learn.org/stable/common_pitfalls.html
- **Wenn die Daten knapp sind — Cross-Validation:** James, Witten, Hastie, Tibshirani, *An Introduction to Statistical Learning*, Kapitel 5 (PDF frei) — statlearning.com

Kurzer Check – mit Lösung

Lösungen für Lehrkräfte – für Lernende die Online-Version nutzen.

Frage 1: Warum darf man ein Modell nicht mit den Trainingsdaten bewerten?

- Weil das zu lange dauert.
- ✔ **Weil es die Beispiele schon kennt — die Note beweist Erinnerung, nicht Können.**
- Weil Trainingsdaten dabei gelöscht werden.

Die Klausur mit den Übungsaufgaben von gestern: Bestehen heißt nichts. Können zeigt sich nur an UNGESEHENEN Daten.

Frage 2: Wofür ist der Validierungs-Topf da — im Unterschied zum Test-Topf?

- ✔ **Validierung: laufend Varianten vergleichen und Entscheidungen treffen. Test: EINMAL ganz am Ende die ehrliche Zahl.**
- Beide sind identisch, nur unterschiedlich groß.
- Validierung ist für schnelle, Test für langsame Modelle.

Wer während der Entwicklung immer wieder auf den Test schaut und danach am Modell dreht, „verbraucht“ ihn — er wird zur zweiten Validierung, und die finale Zahl ist geschönt.

Frage 3: Du skalierst mit min/max über ALLE Daten und teilst erst danach auf. Was ist passiert?

- Nichts — Skalieren ist immer erlaubt.
- ✔ **Data Leakage: Information aus den Prüfdaten ist ins Training gesickert — die Bewertung wird zu gut ausfallen.**
- Das Modell kann danach nicht mehr trainiert werden.

Min und Max „verraten“ dem Training etwas über die Testverteilung. Richtig: erst teilen, dann NUR aus dem Trainings-Topf die Skalierungswerte berechnen (Kapitel 135).

Das nimmst du mit

- **Nie mit den Trainingsdaten bewerten.** Die Klausur mit den Aufgaben von gestern beweist Erinnerung, nicht Können.
- **Drei Töpfe, drei Aufgaben.** Training (~70 %) zum Üben, Validierung (~15 %) zum Vergleichen und Entscheiden, Test (~15 %) für die ehrliche Zahl.
- **Der Test-Topf wird genau EINMAL angefasst.** Wer mehrfach hinschaut und danach am Modell dreht, hat ihn zur zweiten Validierung gemacht.
- **Data Leakage passiert aus Versehen.** Skalieren über alle Daten, doppelte Datensätze, Zeitreisen — drei Klassiker, die die Bewertung heimlich schönen.
- **Aufteilen ist nur Tensor-Zerschneiden.** Mischen, ein Schnitt bei Zeile 32, einer bei Zeile 39 — mehr steckt technisch nicht dahinter.

So zitierst du dieses Kapitel

HerrKI: Trainieren, prüfen, bestehen. herrki.de/235_daten_aufteilen, ein privates Hobbyprojekt von Terragon.de, abgerufen am 09.09.2026.