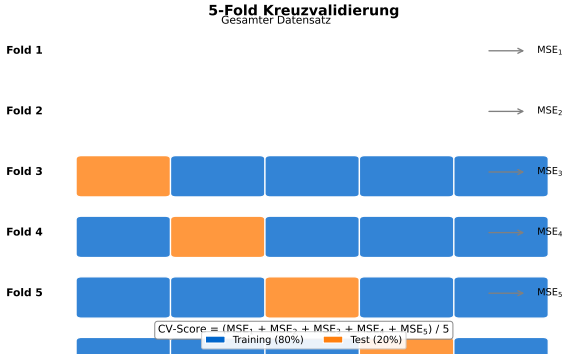
Topic 5: Kreuzvalidierung

Block 3: Komplexitaet, Kausalitaet & Generalisierung

January 25, 2026

Das Konzept: Kreuzvalidierung

- Daten in K gleich grosse Teile aufteilen
- K Iterationen durchfuehren
- Jeder Teil einmal als Testset
- Andere K-1 Teile als Trainingsset
- Jeder Datenpunkt genau einmal getestet



Kreuzvalidierung nutzt alle Daten sowohl zum Training als auch zum Testen

Warum Kreuzvalidierung?

- **Robustere Schaetzung:** Durchschnitt ueber K Testsets reduziert Zufallseinfluss
- **Bessere Datennutzung:** Jeder Datenpunkt traegt zum Training UND zur Evaluation bei
- **Vermeidung von Overfitting:** Modellleistung auf mehreren unabhaengigen Splits getestet
- **Zuverlaessige Modellauswahl:** Vergleich verschiedener Modelle auf gleicher Basis
- **Unsicherheitsquantifizierung:** Standardabweichung ueber K Folds zeigt Variabilitaet

Einzelner Train-Test-Split kann gluecklich oder ungluecklich sein – CV mittelt diesen Effekt heraus.

CV liefert realistischere und stabilere Performance-Schaetzungen als ein einzelner Split

K-Fold Cross-Validation

- **Standard-Wahl:** K = 5 oder K = 10
- **K = 5:** Schneller, hoehere Varianz der Schaetzung
- **K = 10:** Genauer, etwas rechenaufwendiger, etablierter Standard
- **Bias-Varianz-Trade-off:**
  - Kleines K: Groesserer Bias (weniger Trainingsdaten), kleinere Varianz (weniger Folds)
  - Grosses K: Kleinerer Bias (mehr Trainingsdaten), groessere Varianz (mehr Folds)
- **Rechenaufwand:** K Modelle trainieren – K=10 ist meist akzeptabel

Faustregel: Bei kleinen Datensaetzen K=10, bei grossen K=5 ausreichend.

K = 10 ist der Goldstandard fuer die meisten Anwendungen

Varianten der Kreuzvalidierung

- **Leave-One-Out CV (LOOCV):** K = n (Anzahl Beobachtungen)
  - Minimaler Bias, maximale Varianz
  - Sehr rechenintensiv bei grossen Datensaetzen
- **Stratified CV:** Klassenverhaeltnisse in jedem Fold erhalten
  - Wichtig bei unbalancierten Datensaetzen
- **Nested CV:** Aeussere CV fuer Modelltest, innere CV fuer Hyperparameter-Tuning
  - Vermeidet Overfitting bei Hyperparameter-Optimierung
- **Time Series CV:** Training nur auf vergangenen Daten
  - Respektiert zeitliche Abhaengigkeiten

Waehle die CV-Variante passend zur Datenstruktur und zum Problem

Kreuzvalidierung in R mit caret

```
library(caret)

# 10-Fold Cross-Validation definieren
ctrl <- trainControl(
  method = "cv",        # Kreuzvalidierung
  number = 10,          # 10 Folds
  savePredictions = "final"
)

# Modell mit CV trainieren
model <- train(
  y ~ x1 + x2 + x3,
  data = train_data,
  method = "lm",        # oder "glm", "rf", "xgbTree", ...
  trControl = ctrl
)

# Ergebnisse anzeigen
print(model)
model$results           # Durchschnittliche Performance
```

caret macht Kreuzvalidierung einfach und konsistent ueber alle Modelltypen hinweg