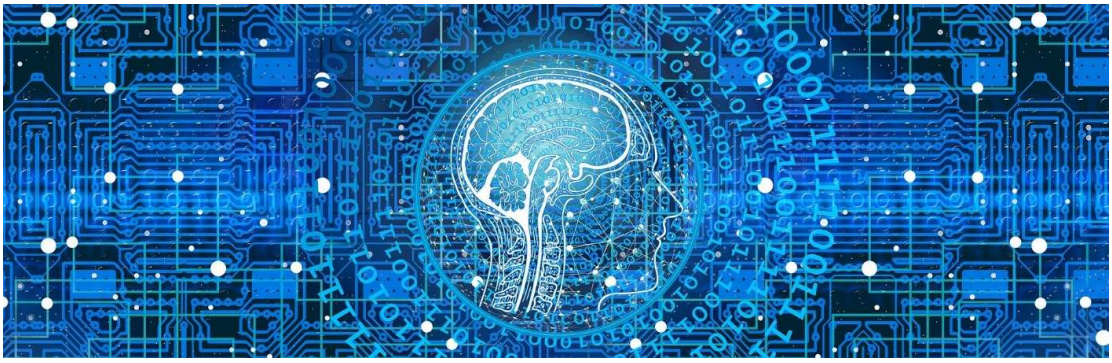
Data Science
4. Teil – Lernen und Bewerten

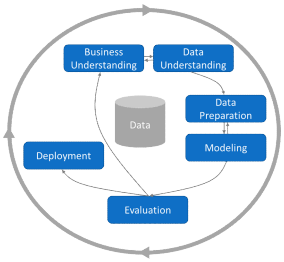
Vorlesung an der DHBW Stuttgart, Prof. Dr. Monika Kochanowski

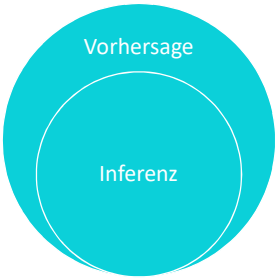


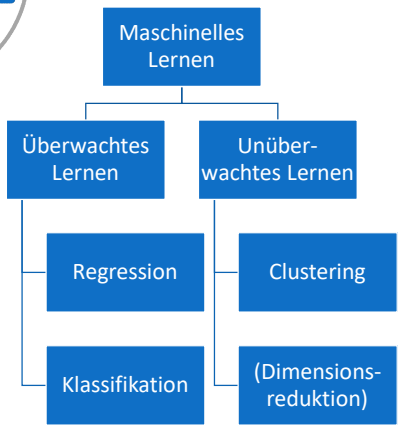
Agenda

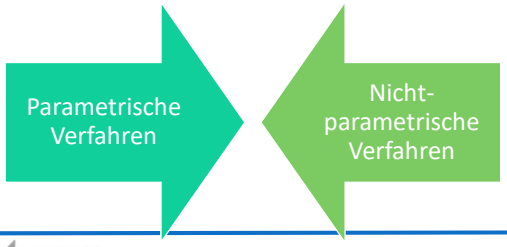
- Kahoot
- Besprechung der Aufgabe
- KNN als Beispielklassifikationsverfahren
- Ridgle/Lasso als Beispielregressionsverfahren
- Regressionsmetriken
- Bias-Variance-Trade-Off
- Train/Testsplit
- Kleine Python Aufgabe
- Vorstellung des Programmentwurfs
- Gruppeneinteilung

Wiederholung





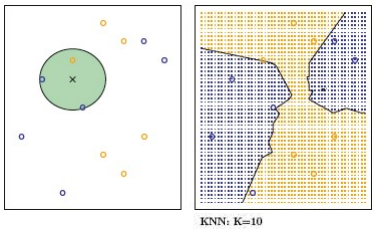


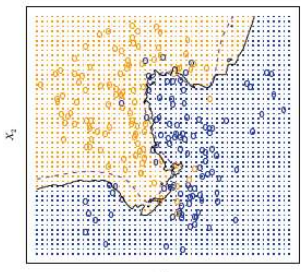


K-Nächste-Nachbarn (k-NN – k-Nearest Neighbors)

Das vermutlich bekannteste Verfahren für Klassifikation

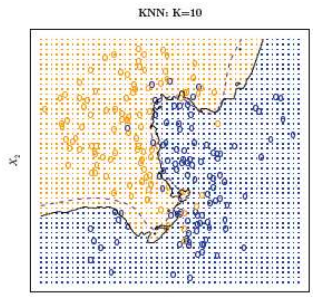
- Der nächste Nachbarn Schätzer sagt für »ähnliche« Eingaben die Ausgabe auf Basis der »ähnlichsten« Datenpunkte voraus – meistens verwendet für Klassifikation
  - »ähnlich« wird dabei definiert über eine **Distanzmetrik** (häufig: Euklidische Distanz, aber es gibt andere)
  - Annahme: näher beieinander liegende Punkte sind »ähnlich«
  - kNN ist eine *nicht-parametrische* Methode (geht nicht von einem Modell aus, welches über den gesamten Eingaberaum gültig ist) [Alpaydin 2008]
- **Fluch der Dimensionalität**
  - Bei sehr vielen Dimensionen in den Eingabewerte ist die durchschnittliche Distanz viel höher und die nächsten Nachbarn könnten von irrelevanten Eingabewerten dominiert werden (z. B. Rauschen)



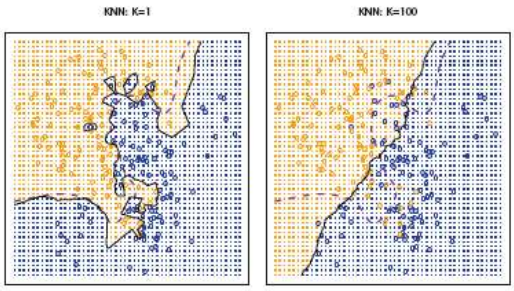


Bildquellen: [James et al. 2013]

Überanpassung und Unteranpassung
Beispiel k-NN



Optimaler Wert für K

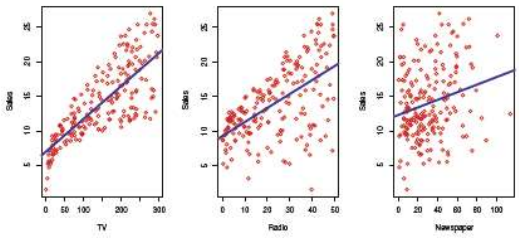


Überanpassung Unteranpassung

Bildquellen: [James et al. 2013]

Lineare Regression
Wiederholung

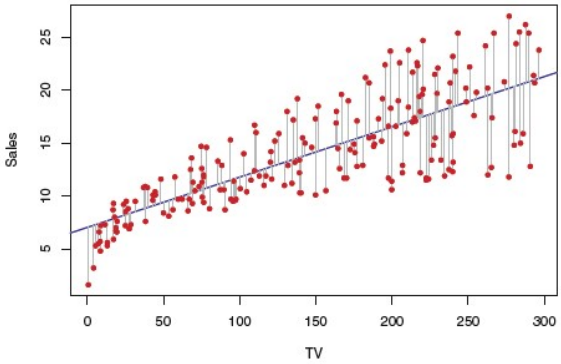
- Eines der ältesten Verfahren (wenn nicht das älteste Verfahren) des maschinellen Lernens
- Überwachtes Lernen
- Parametrische Methode
- **Einfache lineare Regression**
  - »simple linear regression« $y = ax + b$
- In »jeder« Software enthalten
- Liefert oft »erstaunlich gute« Ergebnisse
- Braucht »relativ wenige« Datenpunkte um dies zu tun
- Um viele Konzepte einzuführen, eignet sich die lineare Regression



Bildquelle: [G. James et al, 2013]

Ordinary Least Squares (Methode der kleinsten Quadrate)
und weitere Vorhersagearten

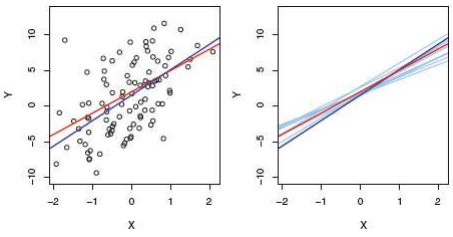
- Frage: Wie findet man die Parameter, die den Fehlerterm **bei linearer Regression** minimiert? $RSS = \sum_{i=1}^{n}(y_i - \hat{y})^2$
  - Im Bild: Quadrierter Abstand der Punkte von der Geraden (*Fehlerquadratsumme*, *Residuenquadratsumme*)
  - Lösung, um die Gerade zu finden: Lineare Algebra oder Arithmetik, z. B. in [O'Neil and Schutt 2013], Für einfache lineare Regression gibt es zudem eine Formel (http://uweziegenhagen.de/teaching/stat/linreg.pdf)
  - **Frage: woher weiß man dass die Linie richtig ist?**
- Alternative Methoden neben der Methode der kleinsten Quadrate zur Bestimmung der Koeffizienten
  - ***Ridge*** oder ***Lasso Regression***: Neben dem RSS spielen weitere **Tuning-Paramter** eine Rolle
  - Ziel: weniger Eingabewerte im Modell
  - **Frage: warum ist das gut?**



Bildquelle: [James et al. 2013]

Statistik angewendet auf die Vorhersage
Vorhersage ist selbst eine Stichprobe

- Woher weiß man dass die wie in der letzten Folie beschriebenen P
  (welche die Quadrate der Abweichungen minimieren) richtig sind?
  - Die Parameter sind am *wahrscheinlichsten*
  - Exkurs: Zudem geht man häufig davon aus, dass die Fehlerter
    normalverteilt sind. (Muss aber nicht!)
- Mit verschiedenen Datensets kann man somit eine *Stichprobe* erst
  - Daraus folgt: Man kann Konfidenzintervalle, p-Werte etc. dar
    berechnen. Im Gegensatz zu Python macht die Sprache R das
    »automatisch«.
- Dazu gibt es verschiedene Ansätze
  - ***Konfidenzintervall:*** man kann die Parameter damit angeben – kann sinnvoll sein.
  - ***p-Wert*** sagt (wenn er sehr niedrig ist) aus, ob die Parameter signifikant sind (Signifikanzniveau 5 %: $p < 0{,}05$), wie alle p-Werte Umsicht hierbei!
  - ***Bootstrapping*** ermöglicht es ebenfalls, die Auswirkungen der Auswahl des Trainingssets auf die Parameter abzuschätzen => „Ersatz" für Konfidenzintervall



Bildquelle: [James et al. 2013]

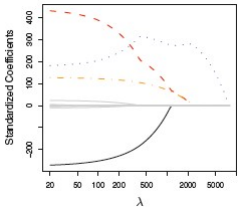
R-Quadrat-Wert (R^2)

oder auch: Wie gut passt mein Modell zu meinen Daten?

- Vorhersage der Form: $\hat{Y} = \hat{f}(X)$ dabei ist $\hat{f}$ unsere **Schätzung** (oder auch *Schätzer*) für f und $\hat{Y}$ die **Vorhersage** für Y (markiert durch das **Dach**)
  - Die **Abweichung** der Vorhersage von den tatsächlichen Werten kann dann wie folgt berechnet werden:
    - $E(Y - \hat{Y})^2 = E[f(X) + \varepsilon - \hat{f}(X)]^2 = \underbrace{[f(X) - \hat{f}(X)]^2}_{} + \underbrace{Var(\varepsilon)}_{}$
  - **Frage**: Wie trifft man die Unterscheidung zwischen ***reduzierbaren*** und ***nicht reduzierbaren*** Fehlern?
- Ziel ist es, den reduzierbaren Fehlerterm zu ***minimieren***. Dafür gibt es verschiedene Metriken.
- Der $\boldsymbol{R^2}$-Wert ist »der Anteil der Varianz in Y welche durch X erklärt wird« (auch: *Determinationskoeffizient* oder *Bestimmtheitsmaß*)
  - $R^2 = \frac{TSS - RSS}{TSS}$ mit $RSS = \sum_{i=1}^{n}(y_i - \hat{y})^2$ und $\text{TSS} = \sum_{i=1}^{n}(y_i - \bar{y})^2$
  - **R**esidual **S**um of **S**quares -> Maß für die Abweichung der vorhergesagten von den tatsächlichen Werten
  - **T**otal **S**um of **S**quares -> Maß für die Abweichung der tatsächlichen Werte vom Mittelwert
- **Diskussion anhand der Formel**

Ridge Regression und Lasso Regression

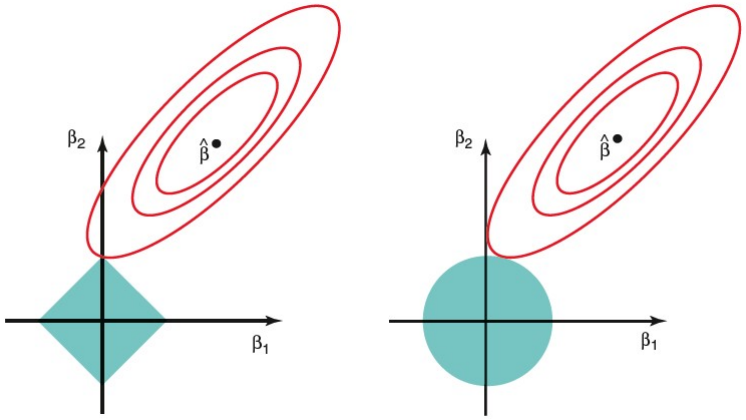
Weiterentwicklungen der linearen Regression



- Wiederholung: Lineare Regression nochmal anders formuliert $\hat{y} = \widehat{\beta_0} + \widehat{\beta_1}$ x
  - **Wofür steht das Dach?**
- **Ridge**: Statt $RSS = \sum_{i=1}^{n}(y_i - \hat{y})^2$ wird eine andere Gleichung minimiert $\text{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2$ mit $\lambda \geq 0$
  - Tuningparameter λ *schrumpft* alle Parameter bis hin zu 0
  - Alle Parameter bleiben erhalten
  - Leichte Vorteile gegenüber normaler linearer Regression bezüglich des Verzerrung-Varianz-Dilemmas
  - Besonders, wenn es sehr wenig Daten für sehr viele Dimensionen gibt $p \approx n$
- **Lasso**: Statt RSS wird eine andere Gleichung minimiert und zwar $\text{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|$
  - Ähnlich wie Ridge
  - Einige der Koeffizienten werden auf genau 0 gesetzt (und sind somit nicht mehr Teil des Modells)
  - => Auswahl der Vorhersageparameter => besser interpretierbar
  - Sparse: Wenige Variablen / Merkmale / Features werden verwendet für die Vorhersage
  - **Was ist besser – Lasso oder Ridge?**

Bildquelle und zum nachlesen: [James et al. 2013]

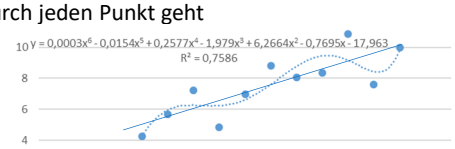
Optional
Lasso (Links), Ridge (Rechts)



Bildquelle und zum nachlesen: [James et al. 2013]

Bias-Varianz Trade-Off
Verzerrung-Varianz-Dilemma

- $E\left(x_0 - \hat{f}(\mathrm{x_o})\right)^2 = \mathrm{Var}\left(\hat{f}(\mathrm{x_o})\right) + \left[Bias(\hat{f}(\mathrm{x_o}))\right]^2 + Var(\varepsilon)$ [James et al. 2013]
  - Aussage bei der Formel: Man sieht die Funktion $\hat{f}$ selbst als »Stichprobe« und betrachtet den Erwartungswert des Fehlers der Vorhersage -> die Formel lässt sich dann zeigen
  - Textform: Für ein gegebenes x_0 ist der erwartete Testfehler $E\left(x_0 - \hat{f}(\mathrm{x_o})\right)^2$ die Summe aus der ***Varianz***, d. h. dem Wert den sich $\hat{f}$ durch die Auswahl des Training-Sets ändert, sowie der ***Verzerrung* (bias)**, also der Fehler der durch die (zu wenig genaue) Annährung eines realen Problems eingeführt wird (durch ***Unteranpassung***)
  - Extreme: man stelle sich die Vorhersage einer horizontalen Linie vor, oder einer Linie die genau durch jeden Punkt geht – oder alternativ eine Linie, die genau durch jeden Punkt geht
    - ***Trade-Off***: beides kann nicht minimiert werden
- Gegenteil von Unteranpassung ist ***Überanpassung***
  - Die Methode generealisiert nicht mehr
  - z. B. Daten 1:1 wiedergeben



Training- und Testsets und Fehlermaße
Ein erster Einstieg in Validierung

- Um Überangepasstheit zu vermeiden (oder zumindest zu erkennen), wird (quasi) immer ein **Trainingsset** und ein **Testset** verwendet (**Frage: was war dies in der ersten Python Übung?**)
  - Naiver Ansatz: 50 % Trainingsdaten und 50 % Testdaten
- Mean Squared Error:
  - $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$
  - Ein gängiges Fehlermaß
  - Trainingsset: mit *steigender* Flexibilität (»lernen«) *fällt* der MSE *immer*
  - Testset: mit *steigender* Flexibilität *fällt* der MSE zunächst typischerweise (»lernen«), *steigt* dann aber *irgendwann* (Überanpassung)
- Andere Maße (**Übung separat, nicht jetzt**)
  - Root Mean Squared Error, Absolute Mean Squared Error
  - Mean Absolute Percentage Error, MIN/MAX

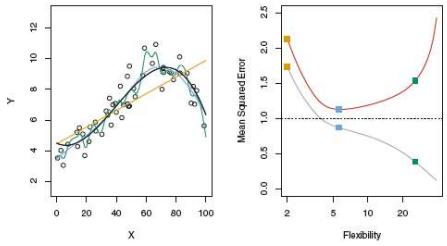


FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

Bias-Variance Trade-Off
Verzerrung-Varianz-Dilemma

- $E\left(x_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}\left(\hat{f}(x_0)\right) + \left[Bias(\hat{f}(x_0))\right]^2 + Var(\varepsilon)$ (Farben s. rechte Abbildung, ε konstant)
- Formel und Auszüge: [James et al. 2013]

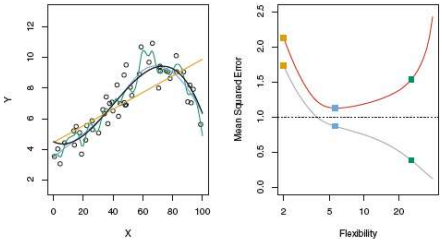


FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

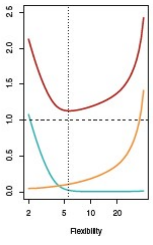
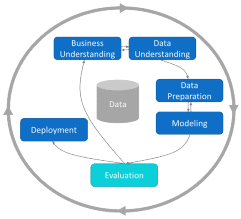
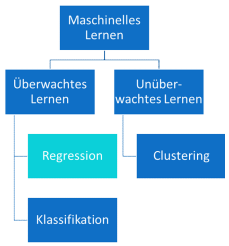


FIGURE 2.12. *Squared bias (blue curve), variance (orange curve), Var(ε) (dashed line), and test MSE (red curve) for the three data sets in Figures 2.9–2.11. The vertical dotted line indicates the flexibility level corresponding to the smallest test MSE.*

Gängige Bewertungsmetriken
Für Regressionsprobleme

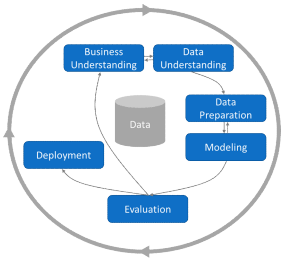


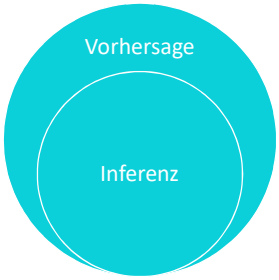
- Mean Squared Error (MSE) $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- Mean Absolute Error (MAE) $MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$
- Mean Absolute Percentage Error (MAPE) $MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{|y_i|}$
- Minimaler / maximaler Abstand (MIN / MAX) $MIN = \min_i |y_i - \hat{y}_i|$
- Root Mean Squared Error (RMSE) $RMSE = \sqrt{MSE}$
- R-Squared: $R^2 = \frac{TSS - RSS}{TSS}$ mit $RSS = \sum_{i=1}^{n}(y_i - \hat{y})^2$ und TSS $= \sum_{i=1}^{n}(y_i - \bar{y})^2$

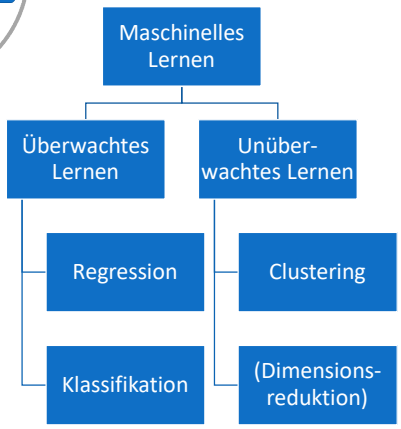


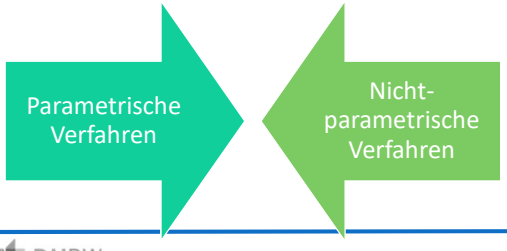
- **Beispiel zum selbst rechnen (in Python)**
- **Mögliche Probleme?**

Wiederholung



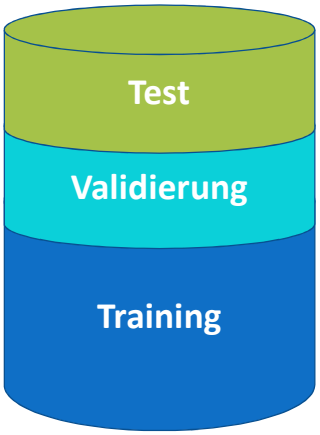


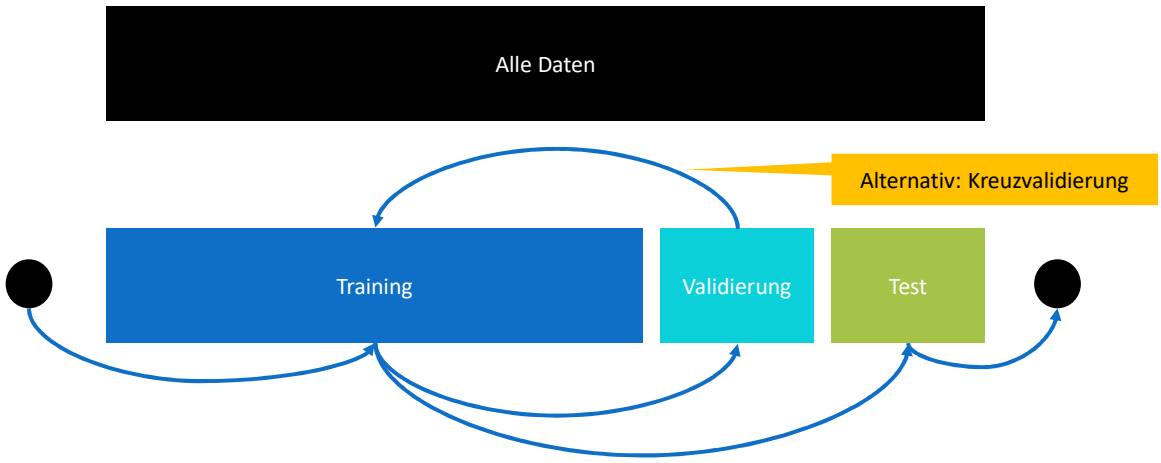


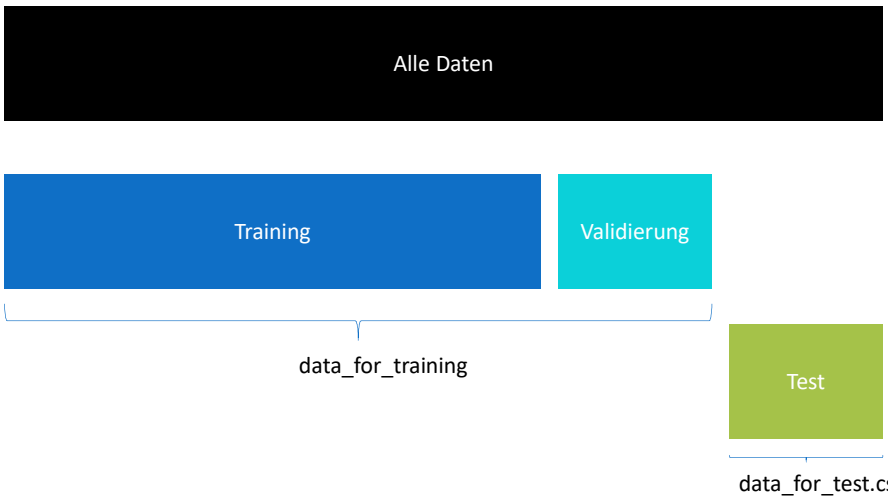


Validierungs- und Testdatensätze
Unterteilung der Daten in Training-, Validierungs- und Testdaten

- Modelle sollten mittels nicht im Training verwendeter Daten **geprüft** werden
  -> kein Overfitting, Generalisierbarkeit sicherstellen
  -> es ist häufig schlechter auf Testdaten
- **Training**: ~80 % der Daten für Erzeugung der Modelle
  Verschiedene Algorithmen
  Verschiedene Parameter
  ggs. Cross-Validation (wenn sinnvoll!)
- **Validierung**: Messung des Fehlers der Modellkandidaten und Auswahl des besten Modells
- **Test**: Überprüfung der Qualität des gewählten Modells **am Ende**







Übungsaufgaben

- Sie haben einen Datensatz mit 100 Datenpunkten und 50 Dimensionen. Worauf müssen Sie bei der Wahl ihres Verfahren achten? Welche Varianz und Verzerrung erwarten Sie?
- Sie haben einen Datensatz mit 1. Mio Datenpunkte und 2 Dimensionen. Was ändert sich zum ersten Szenario?
- Ihre Regressionsmetriken sind im Trainingsset besser als im Testset. Welche Schlüsse ziehen Sie?
- Ihre Regressionsmetriken sind im Testset besser als im Trainingsset. Welche Schlüsse ziehen Sie?
- Sie haben zu wenig Daten für ein Testset. Was tun Sie?
- Welche Vorteile haben nicht-parametrische Verfahren?
- Welche Vorteile haben parametrische Verfahren?
- Warum sollte man ggfs. mehr als eine Metrik nutzen?

Literaturliste

- [James et al. 2013] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani: An introduction to statistical learning
  - Favorit: Sehr gut gemachte Einführung, jedoch Beispiele in R, verständlich mit Mathematik, als pdf frei erhältlich
- [Hastie et al. 2008] Trevor Hastie, Robert Tibshirani, Jerome Friedman: The elements of statistical learning
  - DIE Referenz, für Mathematiker geschrieben, als pdf frei erhältlich
- [O'Neil and Schutt 2013] Cathy O'Neil and Rachel Schutt: Doing Data Science
  - Spannend zu lesen, teilweise Erfahrungsberichte (durch Drittautoren)
- [Mueller and Guido 2017] Andreas C. Müller & Sasha Guido: An Introduction to Machine Learning with Python
  - Interessant da Python 3 tatsächlich genutzt wird für die Einführung inklusive der üblichen Bibliotheken
- [Grues 2016] Joel Grues (übersetzt von Kristian Rother): Einführung in Data Science
  - Auf deutsch gut übersetzt, nutzt Python für grundlegendes Verständnis ohne die üblichen Bibliotheken, extrem leicht lesbar
- [Alpaydin 2008]: Ethem Alpaydin (übersetzt von Simone linke): Maschinelles Lernen
  - Auf deutsch gut übersetzt, relativ viel Mathematik, in Deutschland scheint das weit verbreitet zu sein
- [Bruce et al. 2020]: Peter Bruce, Andrew Bruce, Peter Gedeck: Practical Statistics for Data Scientists
  - Das einzig wahre Statistikbuch was keines ist
- [Reinhart 2016]: Alex Reinhart (übersetzt von Knut Lorenzen): Statistics done wrong
  - Bevor man wirklich Konfidenzintervalle oder p-Werte angibt und über „Signifikanz" spricht, sollte man das gelesen haben

Literaturliste contd.

- Online-Ressource zu Visualisierung
  - https://www.visualisingdata.com/
- Storytelling with Data [Buch]: Klassiker für Überzeugungsarbeit in Präsentationen von Ergebnissen
  - http://www.bdbanalytics.ir/media/1123/storytelling-with-data-cole-nussbaumer-knaflic.pdf
- Show Me the Numbers [Buch]: Ganz konkrete Tipps für die Praxis
  - https://courses.washington.edu/info424/2007/readings/Show_Me_the_Numbers_v2.pdf
- Now you see it [Buch]: Ebenfalls ganz konkrete Inhalte