Business Analytics

Prof. Dr. Jan Stehr

II. Quartal 2025

Dr. rer. nat. Jan Stehr

1971 in Kassel geboren, lebt seit 1994 in Paderborn.
Broterwerb in der Informatik seit ca. 30 Jahren.

Ausbildung zum Mathematisch-technischen Assistenten,
Studium Wirtschaftsinformatik,
Promotion in Informatik an der Uni Paderborn.

Professor für Informatik, AI und Cyber Security an der FHDW,
Systemarchitekt, Security Architect, Leitung Softwareentwicklung
in verschiedenen Unternehmen.

Verwendungshinweise

Bitte beachten Sie:

- Dieses Dokument und die im Rahmen der Veranstaltung ausgegebenen weiteren Dokumente sind nur für den internen Gebrauch an der FHDW durch Studierende und Mitarbeiter der Hochschule bestimmt.
- Sie dürfen Dritten, auch auszugsweise, nur mit vorheriger schriftlicher Zustimmung der FHDW zugänglich gemacht werden.
- Dies gilt für die Weitergabe in jedweder Form, ob elektronisch, z. B. Internet, oder Papier.

Organisatorisches

- 40 Kontaktstunden
- 85 Stunden Selbststudium
- 5 Credit Points
- Prüfungsleistung als Klausur, Dauer 90 Minuten
- Ihren Dozenten erreichen Sie unter jan.stehr@fhdw.de
- Warum findet diese Veranstaltung nur virtuell statt?

- Unterbrechen Sie bitte jederzeit bei Fragen, Unklarheiten, wenn Sie etwas vermissen oder ergänzen möchten.
- Geben Sie Bescheid, wenn Sie mehr Zeit für Übungen möchten.
- Im eLearning dürfen Sie die Kameras ausgeschaltet lassen.
- Melden Sie sich mit Fragen auch gerne außerhalb der Veranstaltung.

Literatur und Software

Shmueli et al.: Data Mining for Business Analytics: Concepts, Techniques and Applications in Python. Wiley, neueste Auflage.

Backhaus et al.: Multivariate Analysemethoden: Eine anwendungsorientierte Einführung. Springer Gabler, neueste Auflage.

Anaconda – anaconda.com

Eine Python-Distribution, Open Source mit Schwerpunkt auf Data Science. Daraus nutzen wir

- Python,
- Jupyter Lab,
- eine Reihe prominenter Bibliotheken.

Die Dokumentationen der einzelnen Produkte finden sie online immer in aktueller Fassung.

Inhalte und Abgrenzung Business Analytics

Business Analytics

- Einführung in eine Statistiksoftware.
- Implementierung grundlegender Verfahren (ein „Kanon“) zur Analyse und Prädiktion
  - numerischer und kategorischer Zielvariablen
  - auf Basis numerischer und kategorischer Prädiktoren.

Data Analysis and Machine Learning

- Grundlagen, Begriffe aus Data Science, Prozess- und Vorgehensmodelle
- Framing – vom Fachproblem zum Analytics-Problem
- Allocation – Architektur, Organisation
- Analytics – Überblick über Methoden, ML, deskriptiv, prediktiv
- Preparation – Präsentation und Verwendung

Grundlagen Data Science

- Zufallsvariablen und Wahrscheinlichkeitsverteilungen
- Stochastische Prozesse
- Gesetz der großen Zahlen
- Induktive Statistik

Data Lab

- Data Processes – Datenbereitstellung, Vorgehen
- Praxisprojekt

Business Analytics: Begriff und Scope

"Business Analytics (BA) verwendet […] prädiktive Analyse (z. B. mit Data Mining, statistischen Analysen und Machine Learning), um die Wahrscheinlichkeit künftiger Ergebnisse zu ermitteln. BA beantwortet Fragen nach dem „Warum“, d. h. damit können Sie fundierte Prognosen über das zukünftige Geschehen machen. Mit BA lassen sich Entwicklungen antizipieren und auf dieser Basis eventuell notwendige Änderungen für ein erfolgreiches Ergebnis frühzeitig in die Wege leiten."

Ansatz	Methoden
Verkleinerung numerischer Datensätze zur besseren Verarbeitbarkeit	Dimensionsreduktion
Numerische Prädiktoren, numerische Zielvariable, linearer Zusammenhang	Multiple lineare Regression
Numerische Prädiktoren, kategorische Zielvariable	Logistische Regression
Prädiktion eines numerischen Faktors mit zeitlichen Komponenten	Zeitreihenregression
Kategorische Prädiktoren, numerische Zielvariable	Varianzanalyse
Klassifikation einer kategorischen oder Prädiktion einer numerischen Zielvariable	k-Nächste-Nachbarn, Entscheidungsbäume
Kategorische Prädiktoren, Wahrscheinlichkeit einer Klassenzugehörigkeit	Naives Bayes-Verfahren
Flexible datengetriebene Klassifikation, Prädiktion oder Eigenschaftenextraktion	Künstliche Neuronale Netze

Auch hilfreich für ein „AI-Verzeichnis“ gemäß AI-Verordnung…

Ein paar Worte zu den Daten/-quellen

Wir wenden in dieser Veranstaltung ein breites Spektrum an Verfahren an. Als Beispiele nutzen wir dazu „Lehrbuchdaten-sätze“. Das sind bekannte Daten, an denen sich die jeweils behandelten Eigenschaften gut darstellen lassen.

Untersuchen Sie selbst auch andere Daten! Aus Ihrem eigenen Arbeitsumfeld, oder aus öffentlichen Quellen.

Beispiele für Datenquellen:

`kaggle.com`

`archive.ics.uci.edu/ml/`

`dwd.de`

`destatis.de`

`govdata.de`

`bund.dev`

`open.nrw`

`data.gov`

`transparenz.hamburg.de`

`github.com/gedeck/dmba/tree/master/src/dmba/csvFiles`

Kurze Wiederholung: Qualität von Vorhersagen

Der Vorhersagefehler für einen Datenpunkt i ist definiert als Differenz zwischen dem tatsächlichen und dem vorhergesagten Ergebnis: $e_i = y_i - \hat{y}_i$. Auf der Grundlage sind definiert:

Mittlerer absoluter Fehler (*mean absolute error*) $\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|e_i|$

Mittlerer Fehler (*mean error*) $\mathrm{ME} = \frac{1}{n}\sum_{i=1}^{n} e_i$. Hier bleiben die Vorzeichen der Fehler erhalten, d. h. Fehler können sich ausgleichen und wir bekommen einen Eindruck, ob die Vorhersagen im Mittel eher zu hoch oder zu niedrig liegen.

Mittlerer prozentualer Fehler (*mean percentage error*) $\mathrm{MPE} = 100 \cdot \frac{1}{n}\sum_{i=1}^{n}\frac{e_i}{y_i}$. Durchschnittliche prozentuale Abweichung der Vorhersagen nach oben oder unten.

Mittlerer absoluter prozentualer Fehler (*mean absolute percentage error*) $\mathrm{MAPE} = 100 \cdot \frac{1}{n}\sum_{i=1}^{n}\left|\frac{e_i}{y_i}\right|$

Wurzel des durchschnittlichen Fehlerquadrats (*root mean squared error*) $\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}$.

Standardfehler der Schätzung.

Kurze Wiederholung und Festlegungen: Begriffe

Datensatz – eine Sammlung von Datenpunkten, z. B. in Form einer CSV-Datei.

Datenpunkt – ein *record*, eine Beobachtung, Instanz, ein Eintrag oder eine Zeile/*row* in einem Datensatz, bzw. eine Koordinate in einem Datenraum mit *n*

Dimensionen – die Anzahl der Attribute pro Datenpunkt, bzw. der

Spalten im Datensatz – die damit auch die einzelnen **Variablen**, **Kategorien** oder **Merkmale** bilden. Daraus können wir

Prädiktoren und

Zielvariablen für eine Vorhersage auswählen.

Arbeit mit einer Statistiksoftware

Allgemeine Arten von Statistiksoftware

Was steht uns zur Auswahl?

Tabellenkalkulationen, Calc, Numbers, Excel & Co. – sehr leistungsfähig, aber nicht so gut bei dynamischen, komplexeren Prozessen.

Interaktive Umgebungen, z. B. KNIME, SPSS (der Dinosaurier, proprietär, UI-lastig, sperrige "Syntax" und damit für die heute übliche hoch flexible Integration und Automation in Datenpipelines weniger geeignet).

Programmiersprachen mit Data Science Frameworks und -Bibliotheken – etwas schwierigerer Zugang für Nicht-IT-ler, aber maximale Flexibilität hinsichtlich Use Cases, Datenformaten, Integration und Plattformen.

Etablierte Sprachen für Statistik, Data Science, ML, AI

Zunächst ist jede Programmiersprache nutzbar, aber es gibt prominente in diesem Umfeld.

R – multiparadigmatische Sprache, entwickelt für Statistik.

Julia – multiparadigmatische Sprache mit Schwerpunkt auf Numerik, manchmal als Alternative zu MATLAB.

Python – multiparadigmatische Sprache, sehr universell einsetzbar und eingesetzt.

Alle diese Sprachen bieten einen (für Programmiersprachen) niedrigschwelligen Zugang durch dynamische Typisierung. Programmieren geht kaum einfacher.

Durch die universelle Einsetzbarkeit und hohe Verbreitung nutzen wir in unserer Veranstaltung Python.

Data-Science-Werkzeuge und -Bibliotheken

Python/R-Distribution für Data Science/ML/AI
mit Paket-/Deployment-Manager

Interaktive Programmier- und
Rechenumgebung

Das Gleiche wie vorne für die Daten gilt auch hier: Durch eigenes Ausprobieren lernen Sie sehr viel. Die Jupyter Notebooks unserer Veranstaltung sind nicht read-only :-)

Extract, Transform, Load (ETL)
Data Exploration
Data Evaluation
Data Modeling
Data Presentation

Unsere Arbeitsumgebung: Python in JupyterLab

Anaconda Shell:

→ Ins Arbeitsverzeichnis wechseln

→ `jupyter lab`

→ `BSA_Einfuehrung.ipynb` öffnen

Dimensionsreduktion

Korrelationsanalyse
Principal Components Analysis

Dimensionalität von Daten…

Aus Data Analysis:

- Daten und Untersuchungen können viele Dimensionen haben.
- Für eine Analyse sind viele Sichtweisen und Kriterien interessant, also kann das zu Grunde liegende Datenmodell auch mehr- bis vieldimensional sein, damit es für alle(s) passt – einfacherer Zugang, alle Perspektiven finden sich wieder, nichts wird vergessen.

Was meinen wir hier mit *Dimension*:

- Anzahl an Variablen, auch Kategorien, die in unser Modell eingehen.
- Geometrische Interpretation: Wenn die Daten in Form von Tabellen vorliegen, sind die Variablen die Spalten/Columns, die unseren Datenraum definieren. Einzelne Datenpunkte in diesem Raum sind dann die Zeilen/Rows.

… und ihr „Fluch“

R. Bellman and R.E. Bellman. Adaptive Control Processes: A Guided Tour. Rand Corporation. Research Studies. Princeton University Press, 1961.

Anschaulich:

- Viele Dimensionen bilden einen sehr großen Raum.
- Die vielen Perspektiven bringen schnell Nachteile. Einfaches Beispiel: Visualisierung wird schnell schwierig, wenn Dimensionen > 3.

Es braucht sehr viele Daten, um ihn zu füllen, der Bedarf steigt exponentiell mit den Dimensionen. Haben wir zu wenig, ist unsere Stichprobe/Sample zu klein, um einen ML-Algorithmus damit valide zu trainieren. Es finden sich keine/zu wenig repräsentative Muster etc.

Liegen genügend Daten vor, behindert die Komplexität effiziente Verwertungen. Z. B. kosten redundante oder für die jeweilige Fragestellung irrelevante Variablen viel Rechenzeit, verbessern das Ergebnis aber nicht, oder verzerren bzw. verschlechtern es sogar.

Weitere praktische Aspekte

- Es kann aufwändig, teuer oder sogar unmöglich sein, die Daten für alle Variablen zu sammeln.
- Die Datenqualität kann besser sein, wenn weniger Variablen betrachtet werden (z. B. in Umfragen).
- Bei sehr vielen Variablen und Daten können schneller oder häufiger Lücken auftreten.
- Datensparsamkeit kann auch hier eine Rolle spielen: Der Einfluss einzelner Variablen auf die Ergebnisse wird bei einer kleineren Datenmenge deutlicher.

Dimensionsreduktion: Ansätze

1. Domänen- bzw. Fachkenntnisse nutzen, um Kategorien zu entfernen oder zusammenzufassen.
2. Daten aufsummieren/zusammenfassen, um Informationsüberlappungen zwischen Variablen zu behandeln und redundante Variablen zu entfernen oder mit anderen zusammenzufassen.
3. Daten umwandeln, z. B. kategorische in numerische (1 = *gehört zu Kategorie*, 0 = *gehört nicht zu Kategorie*).
4. Reduktionsverfahren einsetzen, die eine neue, unkorrelierte Variablenmenge erzeugen, bestehend aus gewichteten Durchschnitten der Ausgangsmenge. Daraus wählen wir eine Untermenge, die die für uns relevanten Informationen enthält.
5. Regressions- und Klassifikationsverfahren einsetzen, um redundante Variablen zu identifizieren und zu entfernen und ähnliche Kategorien von Variablen zu kombinieren.

Dimensionsreduktion: Allgemein

Die Ansätze und Verfahren sind in der Literatur oft zusammengefasst zu

Feature Selection – Reduktion der Variablenmenge und damit Dimension durch Auswahl von relevanten Variablen,

Feature Extraction – Reduktion der Variablenmenge und damit Dimension durch Kombination und Zusammenfassung von Variablen

oder auch

überwachte (supervised) Reduktion – mit interaktiven Ansätzen, die also menschliche Eingriffe erfordern, bzw. selektiv vorgehen und die Variablen entsprechend einordnen,

unüberwachte (unsupervised) Reduktion – behandelt alle Variablen gleich, mit einer abstrakten Definition, z. B. der beinhalteten Information.

Dimensionsreduktion

Korrelationsanalyse

Principal Components Analysis

Korrelationsanalyse: Ansatz

Pragmatischer Ansatz: Stark korrelierende Variablen auf jeweils einen Repräsentanten reduzieren.

In Datensätzen mit vielen Variablen, also potentiellen Prädiktoren, gibt es oft viele Überlappungen der Informationen. Wenn wir solche Redundanzen finden, erhalten wir Möglichkeiten zur Dimensionsreduktion. Dazu müssen wir aus den betreffenden Variablen passende mit ihren Daten aus dem Datensatz entfernen.

Vorgehen halbautomatisch: Korrelationsmatrix

Eine Korrelationsmatrix zeigt die paarweisen Korrelationen aller Variablen des Datensatzes. Paare mit starker – positiver oder negativer – Korrelation besitzen ähnliche Informationen, also Redundanz. Entsprechend können wir eine davon entfernen.

Zusätzlich können wir in Rohdatensätzen Duplikate von Daten ermitteln, z. B. gleiche Datenreihen unter verschiedenen Namen oder Maßeinheiten.

Praktisch: `Dimensionsreduktion_KA.ipynb`

Dimensionsreduktion

Korrelationsanalyse

Principal Components Analysis

Principal Components Analysis (PCA): Ansatz

Die Hauptkomponentenanalyse ist spätestens dann hilfreich, wenn wir sehr viele Variablen betrachten müssen, die vorherige manuelle Bearbeitung also unübersichtlich wird.

Besonders in Datensätzen mit hoch korrelierten Untermengen reduziert die PCA die Variablen auf wenige gewichtete Linearkombinationen der Ausgangsvariablen. Diese Hauptkomponenten

- behalten den Großteil der Informationen der ursprünglichen Variablen,
- sind unkorreliert, also nicht redundant.

Maß für die in einer Variable enthaltenen Information ist hier die Varianz (eine Variable mit einer Varianz von 0 enthält keine Information).

PCA ist geeignet für numerische Variablen, nicht für kategoriale.

Praktisch: `Dimensionsreduktion_PCA.ipynb`

Regressionsverfahren

Multiple lineare Regression
Logistische Regression

Regressionsverfahren

Multiple lineare Regression

Logistische Regression

Multiple Lineare Regression: Ansatz

Allgemein haben wir hier eine

- abhängige Ziel- oder Ergebnisvariable Y,
- die wir durch eine Menge unabhängiger Prädiktorvariablen X_j $(j = 1, \dots, p)$,
- für die wir jeweils eine Datenreihe haben (an die wir bestimmte Anforderungen stellen, dazu später mehr),

bestimmen wollen. Der Zusammenhang zwischen den Variablen soll hier linear sein:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$$

mit den Koeffizienten β_i und einer Stör- oder Fehlergröße ε.

Zur Bestimmung der β_i und ε benutzen wir N Datenpunkte und ein Kleinste-Quadrate-Verfahren (z. B. bekannt aus WSTA oder DAML).

$$KQ = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} \left[y_i - \left(\beta_0 + \beta_1 \cdot x_{1i} + \beta_2 \cdot x_{2i} + \ \dots + \beta_p \cdot x_{pi} \right) \right]^2 \rightarrow min$$

Deskription versus Prädiktion

In dieser Veranstaltung interessieren wir uns für die Anwendung der linearen Regression zur *Vorhersage* (Prädiktion). In allgemeinen Statistikveranstaltungen ist der Kontext aber oft die beschreibende bzw. deskriptive Statistik. Daher folgende Unterscheidung:

In der deskriptiven Statistik dient die LR dazu, die durchschnittliche Wirkung von Inputs auf Outputs zu erklären oder zu quantifizieren.

1. Modell approximiert die vorhandenen Daten und die Verhältnisse zwischen Eingaben und Ausgaben möglichst gut.
2. Nutzt den gesamten verfügbaren Datensatz.
3. Fokus auf den Koeffizienten.

In der Vorhersage dient die LR dazu, basierend auf Inputs neue dazu passende Outputs zu erzeugen.

1. Liefert ein möglichst genaues Ergebnis für neue Daten.
2. Teilt den verfügbaren Datensatz in Trainings- und Validierungsmenge.
3. Fokus auf den Prädiktionen, also der Zielvariable.

Auswahl von Variablen

Modellprämissen kennen Sie aus DAML (R-Beispiele lassen sich auch in Python implementieren).

Ein Verfahren wie die Regression profitiert von Dimensionsreduktion – aus den dort aufgeführten Gründen und mit den dort vorgestellten Methoden. Andererseits wollen wir die Variablen berücksichtigen, die uns die besten Ergebnisse der Prädiktionen liefern.

Gesucht ist ein Modell, das nicht zu einfach ist, also keine wichtigen Variablen auslässt (*under-fit*) und auch nicht zu komplex (*over-fit*).

Im Kontext der Regression können wir dabei schlicht Rechenleistung einsetzen:

1. Mit einer **erschöpfenden Suche** finden wir aus allen Untermengen von Prädiktoren die für unsere Regression optimale Kombination. Das stößt bei großen Mengen allerdings auch mit den Rechenkapazitäten einer Cloud an Grenzen (Kombinatorik…).
2. Alternativ **bestimmen wir Teilmengen** und suchen darin nach dem besten Modell.

Praktisch: `Multiple_Lineare_Regression.ipynb`

Regressionsverfahren

Multiple lineare Regression

Logistische Regression

Logistische Regression: Ansatz

Unsere abhängige Ziel- oder Ergebnisvariable Y soll nun diskret bzw. kategorisch sein, statt einer numerischen Prognose möchten wir allgemein

- eine **Klassifikation** eines neuen Datenpunktes erhalten, oder
- bei einer bekannten Klasse eines Datenpunktes ermitteln, welche Eigenschaften der Prediktoren zu dieser Einordnung geführt haben (**Profiling**).

Beispiele solcher Anwendungen sind

- die Einteilung von neuen Kunden in einmalig und regelmäßig,
- die Ermittlung der kritischen Faktoren für den Erfolg einer Innovation,
- die Beurteilung der Kreditwürdigkeit einer Person,
- die Einordnung von Emails als Spam, mit Prediktoren wie der Häufigkeit bestimmter Zeichenketten,
- die Identifikation von Einflüssen auf die Wahl einer Partei, eines Produkts etc.

Logistische Regression: Aufbau (1)

Gibt es nur zwei Alternativen der Klassifikation, so bildet Y eine binäre (dichotome) Zielvariable, die entsprechend eine **binäre logistische Regression** erfordert. Bei mehr als zwei Alternativen geht es um **multinomiale logistische Regression**.

Im binären Fall bezeichnen wir die beiden Alternativen allgemein mit 1 und 0, z. B. *Spam/kein Spam*, *Kredit ja/nein*. Dabei nehmen wir an, dass Y eine Zufallsvariable ist. Die Wahrscheinlichkeit für das Ereignis 1 ist

$$p = P\left(Y = 1 \middle| X_1, X_2, \dots, X_q\right)$$

als eine **bedingte Wahrscheinlichkeit durch die Werte der Prädiktoren** $\boldsymbol{X} = (X_1, X_2, \dots, X_q)$ der linearen Funktion

$$Z(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_q X_q$$

hier bezeichnet als **systematische Komponente** der logistischen Regression.

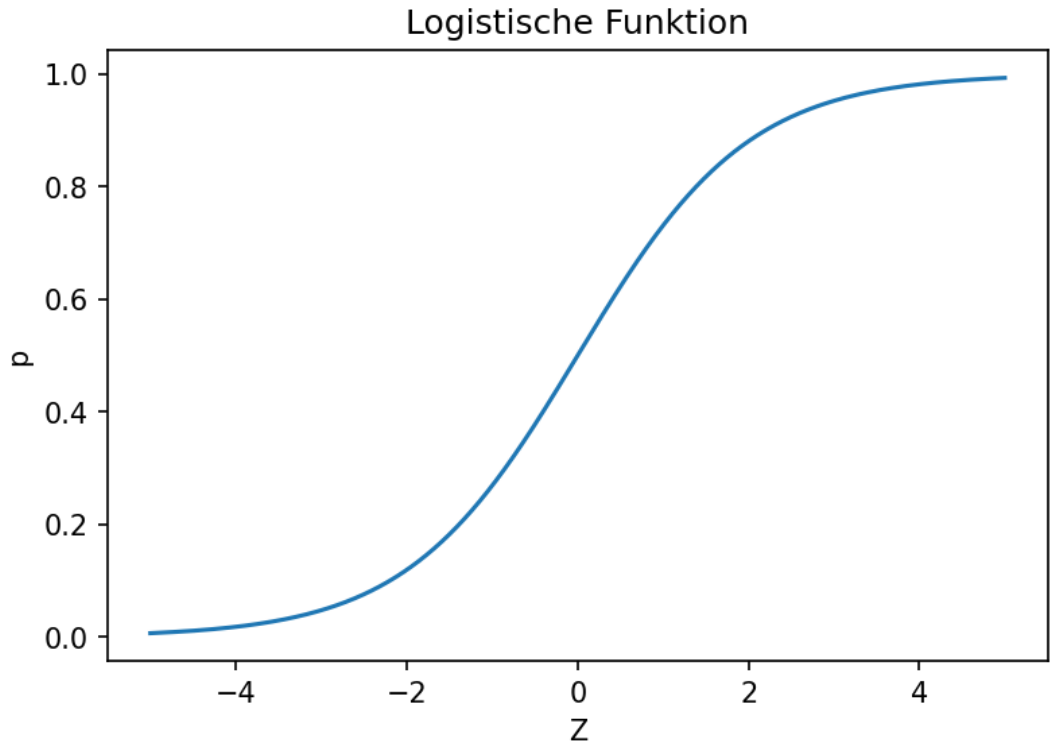
Logistische Regression: Aufbau (2)

Z als unsere bisherige, bekannte Regressionsfunktion besitzt allerdings einen Wertebereich von $-\infty$ bis $+\infty$. Den müssen wir in eine Wahrscheinlichkeit umwandeln, also einen Wertebereich von $[0,1]$.

Das erreichen wir durch die **nichtlineare logistische Funktion**

$$p(Z) = \frac{e^Z}{1+e^Z} = \frac{1}{1+e^{-Z}}$$

Sie ähnelt der Verteilungsfunktion (kumulative Wahrscheinlichkeitsfunktion) der Normalverteilung.



Durch den nichtlinearen Zusammenhang funktioniert hier ein Kleinste-Quadrate-Ansatz zur Berechnung nicht. Die Anpassung erfolgt iterativ über *Maximum Likelihood*.

Logistische Regression: Odds

Zur Interpretation der Regressionsergebnisse und zur vereinfachten Darstellung des Modells benötigen wir noch zwei weitere Größen: *Odds* und *Logit*.

Die **Chancen (*odds*)** beschreiben das Verhältnis der Wahrscheinlichkeit von Klasse 1 zur Klasse 0, oder allgemein auch oft „Erfolg“ zu „Misserfolg“ mit

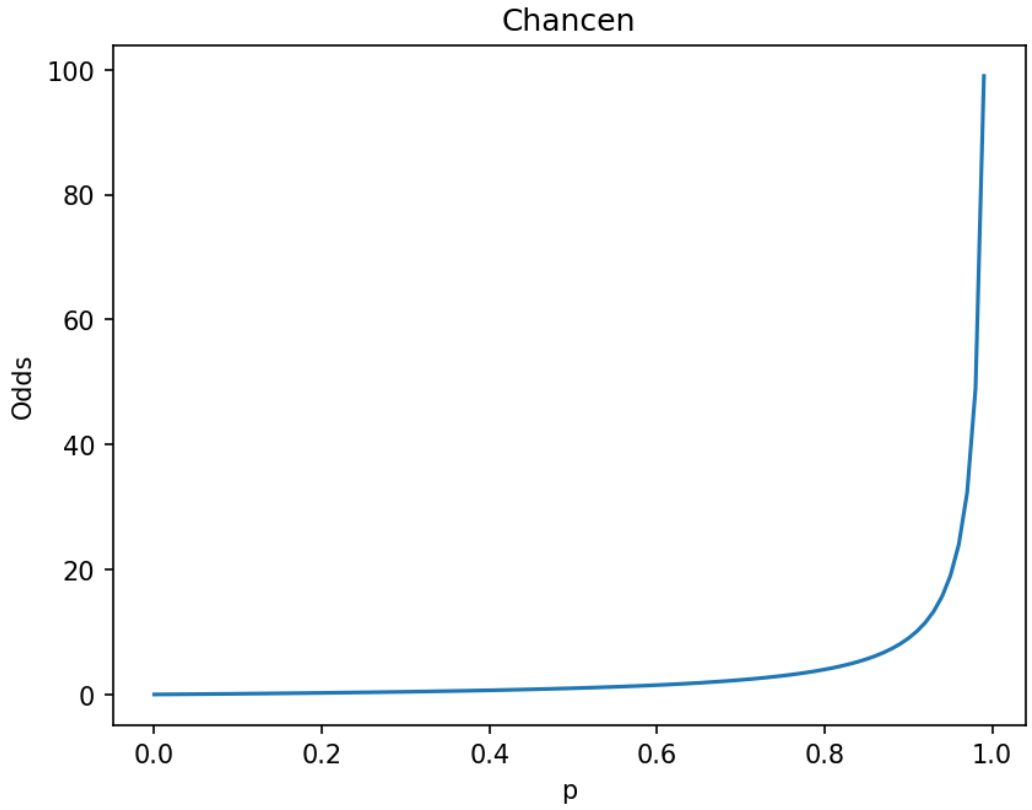
$$\text{Odds}(Y = 1) = \frac{p}{1 - p}$$

Daraus ergibt sich der bei z. B. Wetten übliche Ausdruck „die Chancen stehen X zu 1“.

Setzen wir dort unsere logistische Funktion ein, erhalten wir

$$\text{Odds}\left(p\big(Z(X)\big)\right) = e^{Z} = e^{\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_q X_q}$$

(Das bedingte $Y = 1$ ist hier nun implizit.)



Logistische Regression: Logit

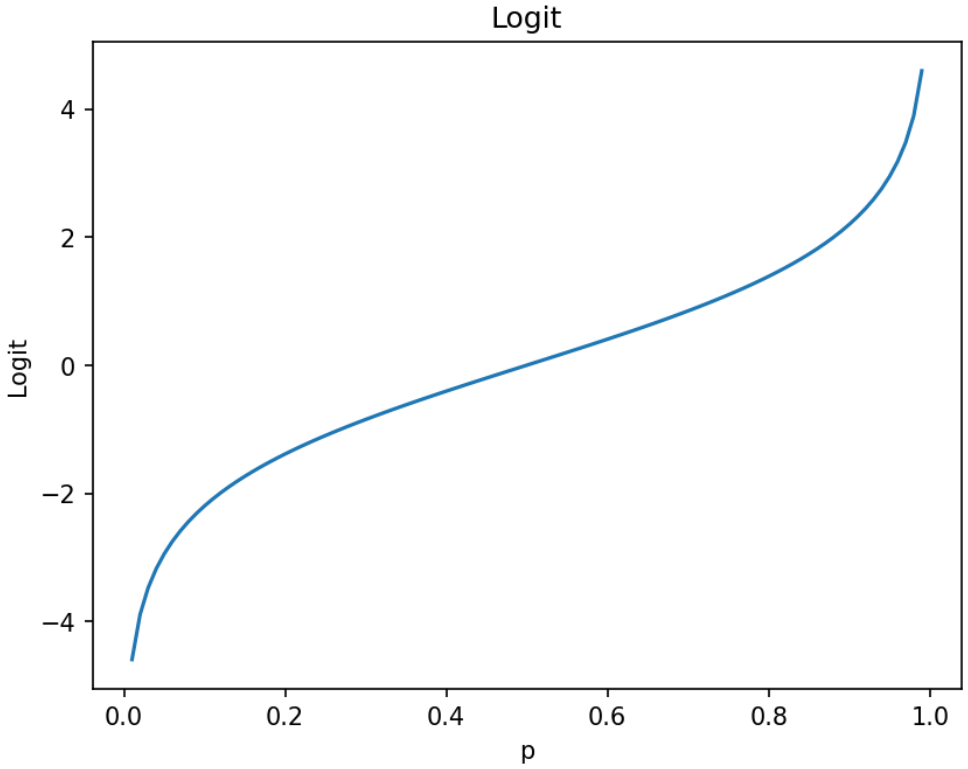
Der natürliche Logarithmus (ln) der *Odds* ist der sogenannte **Logit**:

$$\text{Logit}\left(p\left(Z(X)\right)\right)$$

$$= \ln\left(\text{Odds}\left(p\left(Z(X)\right)\right)\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_q X_q$$

$$= Z(X)$$

Wir linearisieren so das logistische Regressionsmodell mit seinen Prädiktoren als Eingabe und der Ausgabewahrscheinlichkeit. Die Logit-Transformation liefert auch die Umkehrfunktion der logistischen Funktion.



Logistische Regression: Cutoff

Die vorhergesagte Wahrscheinlichkeit wird auch als **propensity** bezeichnet, also Hang, Neigung i. S. v. Tendenz, dass ein Datenpunkt zu einer Klasse gehört. Für die eigentliche Fragestellung der logistischen Regression – der Klassifikation – stellt sie also nur einen Zwischenschritt dar.

Für die Zuordnung zu einer Klasse muss ein Modell einen Schwellenwert festlegen: Den **cutoff value**. Wenn die Wahrscheinlichkeit der Zugehörigkeit über dieser Schwelle liegt, wird der Datenpunkt der Klasse zugeordnet.

Der Standardwert für den cutoff ist 0.5. Den können wir aber ändern, allgemein zu Gunsten von Fehlklassifikationen in eine Richtung.

Warum sollte man das tun? Es können Asymmetrien in den Szenarien existieren: Eine bestimmte Einordnung könnte wichtiger sein, als eine andere, oder die Kosten für eine Fehlklassifikation in eine Richtung viel höher, als in die andere (Beispiele?).

In bestimmten Szenarien (*Triage*) gibt es zusätzlich noch einen unscharfen Bereich (*cannot say*), der durch zwei cutoffs definiert werden kann. Die Schwellen werden also erhöht. Im fragwürdigen Bereich könnten z. B. nicht genügend Informationen vorliegen.

Kleines Beispiel: `Logistische_Regression_Einfuehrung.ipynb`

Logistische Regression: Anforderungen an die Datensätze

- Hauptannahme: Kategoriale Zielvariable ist eine Bernoulli- oder multinomial-verteilte Zufallsvariable
- Größere Datenmengen erforderlich als z. B. für lineare Regression. Pro Kategorie der Zielvariable mindestens 25 Datenpunkte, d. h. mindestens 50 insgesamt.
- Bei größerer Anzahl der Prädiktoren sind größere Datenmengen pro Kategorie erforderlich.
- Prädiktoren sollten frei von Multikollinearität sein, d. h. nicht linear abhängig.

Größeres Beispiel: `Logistische_Regression_Profiling.ipynb`

Zeitreihen

Zeitreihen / Time Series: Ansatz

In der Vorhersage auf Basis von Daten spielen die Zeitreihen eine prominente Rolle. In unserem Alltag finden sich unzählige Beispiele:

- Entwicklung der Märkte, Preise, (Leit-)Zinsen
- Infektionen und damit verbundene Zahlen
- Versorgung durch erneuerbare Energien, Wetter
- Klimawandel

Vereinfacht haben wir bisher die Daten als *Querschnittsdaten* betrachtet. Sie setzen sich aus einer Reihe von Faktoren/Variablen/Datenreihen zusammen, in denen auch Zeit vorkommen kann (z. B. Baujahr), aber keine besondere Rolle spielt.

Mit **Zeitreihen** betrachten wir hier hingegen **einen Faktor in seiner zeitlichen Entwicklung**. Auch, wenn mehrere Faktoren über die Zeit untersucht werden, geschieht dies dann jeweils in Einzelbetrachtung. Multivariate Zeitreihen mit Korrelationen *zwischen* den Reihen benötigen eigene, komplexe Modelle, die nicht im Rahmen dieser Veranstaltung liegen.

Zeitreihen / Time Series: Praktische Aspekte

Zeitreihen entstehen auf sehr verschiedenen Skalen in unterschiedlichem Umfang:

- Aktienkurse mit einem „Ticker“ in definierter Frequenz,
- Käufe auf Webshops in unregelmäßiger „Echtzeit“,
- Sensorwerte im IoT von z. B. Sekundenwerten aus Verkehrsmessungen, Stundenwerten aus Temperaturfühlern etc.

Für Vorhersagen stellt sich dabei die Frage, ob die Skalierung übernommen, oder geändert werden sollte. Für eine tägliche Umsatzprognose beinhalten z. B. die minutenweisen Messdaten der einzelnen Einkäufe viele Verzerrungen (Spitzenzeiten etc.).

Entsprechend finden wir spezialisierte Technologien für Sammlung und Speicherung:

- Time Series Databases, die in ihren Strukturen auf die Verarbeitung von eher uniformen Massendaten ausgelegt sind,
- kompakte Binärkodierung von Sensordaten, wenn sie massenhaft über Luftschnittstellen gesammelt werden.

Zeitreihen / Time Series: Komponenten

Level – der Durchschnittswert der Reihe.

Trend – die Veränderung der Reihe von einer Periode bis zur nächsten.

Saisonalität – ein zyklisch wiederkehrendes Verhalten der Reihe über einen bestimmten Zeitraum.

Rauschen – zufällige Veränderungen durch Messfehler o. ä.

Zu den Begrifflichkeiten im Kontext hier:

Eine Periode ist eine Zeitspanne, die nicht zwingend zwischen zwei immer gleichen Zeitpunkten in einem Zyklus liegen muss.

Eine Saison ist eine Periode, die sich zyklisch in einer Zeitreihe wiederholt.

Erstes Beispiel: `Zeitreihen_Einfuehrung.ipynb`

Zeitreihen: Regressionsbasierte Vorhersagen

Um Regression für Zeitreihen einzusetzen, müssen wir zunächst lediglich den Prädiktor als Zeitindex definieren:

$$Y_t = \beta_0 + \beta_1 t + \varepsilon$$

Y_t ist damit der Vorhersagewert zum Zeitpunkt t. Drei der Komponenten sind auch bereits in diesem Modell enthalten: β_0 als *Level*, β_1 als *Trend* und ε als *Rauschen*.

Neben dem linearen finden wir Modelle für einen **exponentiellen** Trend mit $Y_t = ce^{\beta_1 t + \varepsilon}$ und **polynomiale**, z. B. **quadratische** Trends mit $Y_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \varepsilon$.

Saisonalität gibt es in zwei Formen:

- Bei *additiver Saisonalität* ist der Durchschnittswert von Y in einer bestimmten Saison um einen konstanten Wert erhöht oder vermindert. Das erfassen wir im Regressionsmodell durch eine *kategorische Variable, die jede Saison abbildet* (und für die Berechnung dann in Dummy-Variablen übersetzt wird).
- Bei multiplikativer Saisonalität bezieht sich die Veränderung einer Saison auf einen prozentualen Wert gegenüber einer anderen Saison. Das Modell erweitern wir dazu wie oben, passen aber $ln(Y_t)$ an bzw. multiplizieren mit der Saisonalitätskomponente.

Praktisch: `Zeitreihenregression.ipynb` bis Autokorrelation

Zeitreihen: Autokorrelation

Zusätzlich zu den bisher betrachteten Komponenten gibt es in Zeitreihen einen weiteren Einfluss auf den Verlauf: Die Werte benachbarter Perioden beeinflussen sich oft, sind also *korreliert*. Da der Einfluss sich hier auf die Variable selbst bezieht, bezeichnen wir das als **Autokorrelation**.

Um sie zu berechnen, ermitteln wir die Korrelation zwischen der Reihe und einer „verzögerten“ (*lagged*) Kopie. Verzögerungen können aus einer oder mehreren Verschiebungen bestehen – *lag-1*, *lag-2* etc.

Typisches Autokorrelationsverhalten ist:

- Starke Autokorrelation (positiv oder negativ) bei einem $lag\ k > 1$ und seiner Vielfachen ($2k$, $3k$, …) weist auf ein zyklisches Muster hin, z. B. *lag-12* bei monatlichen Daten auf eine jährliche Saisonalität.
- Positive lag-1-Autokorrelation beschreibt eine Reihe, wo sich aufeinander folgende Werte generell in eine Richtung bewegen. Bei einem starken linearen Trend erwarten wir entsprechend eine starke, positive lag-1-AK.
- Negative lag-1-Autokorrelation beschreibt eine schwankende Reihe, in der hohe Werte direkt von niedrigen gefolgt sind u. u.

Zeitreihen: Nutzung der Autokorrelation

Es gibt zwei Ansätze, um Autokorrelation in unseren Modellen zu berücksichtigen:

1. Autokorrelation direkt in das Regressionsmodell integrieren.
2. Ein nachgelagertes Vorhersagemodell für die Reihe der Residuen bauen.

Zeitreihen: Integrierte autoregressive (AR-) Modelle

Als Prädiktoren nutzen wir hier die vergangenen Werte der Zeitreihe. Ein Modell n-ter Ordnung, bezeichnet $AR(n)$ besitzt dann die Form

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_n Y_{t-n} + \varepsilon$$

Autoregressive Integrated Moving Average (ARIMA) repräsentieren Modelle, die für diese Art von Problemen besser geeignet sind, als einfache Regressionen. Sie sind aber auch schwierig anzuwenden, da ihre Modellierung und Parametrierung erhebliches Wissen über die Methode als auch die Daten erfordern.

Es gibt eine bestimmte, einfachere Einsatzmöglichkeit…

Zeitreihen: Nachgelagerte Autokorrelationsmodelle

Vorhersageverbesserung durch zusätzliche Vorhersage des Fehlers

1. Wir generieren eine *lag-k*-Vorhersage der Zeitreihe F_{t+k}
2. Wir generieren eine *lag-k*-Vorhersage des Vorhersagefehlers bzw. der Residuen E_{t+k} mit einem (gegebenen) AR-Modell
3. Wir verbessern die Vorhersage aus 1, indem wir sie gemäß ihres vorhergesagten Fehlers korrigieren: $F^*_{t+k} = F_{t+k} + E_{t+k}$

Praktisch: `Zeitreihenregression.ipynb` ab Autokorrelation

Varianzanalyse

Varianzanalyse: Ansatz

Bisher haben wir Methoden kennen gelernt, um aus numerischen Prädiktoren eine numerische und eine kategorische Zielvariable zu bestimmen. Die Varianzanalyse – *(Multivariate) Analysis of Variance*; (M)ANOVA – ermöglicht uns Aussagen über kategorische Einflussfaktoren auf numerische bzw. metrische Zielvariablen.

Praktische Fragestellungen z. B.:

- Für eine Softwareentwicklungspipeline gibt es vier verschiedene automatische Testwerkzeuge. Beeinflussen diese Produkte die Fehlerrate auf unterschiedliche Weise?
- Eine landwirtschaftliche Produktion soll die Wirksamkeit von drei verschiedenen Düngemitteln im Zusammenhang mit Wetterlagen prüfen. Dazu untersucht sie den Ernteertrag einer bestimmten Pflanzengattung in verschiedenen Regionen mit jeweils drei Düngesegmenten auf den Feldern.
- Eine Klinik kann verschiedene Therapien zur Behandlung einer Krankheit einsetzen. Sie möchte wissen, ob die Therapien zu unterschiedlichen Genesungszeiten führen und welche die schnellste ist.

Varianzanalyse: Modell

Es gibt für sie je nach Analyseszenario unterschiedliche Modelle. Für eine zweifaktorielle ANOVA mit Interaktionseffekten ist das

$$y_{ghi} = \mu + \alpha_g + \beta_h + (\alpha\beta)_{gh} + \varepsilon_{ghi}$$

mit

y_{ghi}	Beobachtungswert $i = 1, \dots, N$ der Zielvariable für Faktorstufen $g = 1, \dots, G$ und $h = 1, \dots, H$. Die „Stufen“ sind hier die Ausprägungen der Kategorien.
μ	Gesamtmittelwert der Grundgesamtheit
α_g, β_h	wahre Effekte der Faktoren g und h
$(\alpha\beta)_{gh}$	wahrer Interaktionseffekt von g und h
ε_{ghi}	Störgröße

wobei

$\alpha_g = \mu_g - \mu$ und $\beta_h = \mu_h - \mu$

μ_g und μ_h als Mittelwerte der Faktorstufen in der Grundgesamtheit

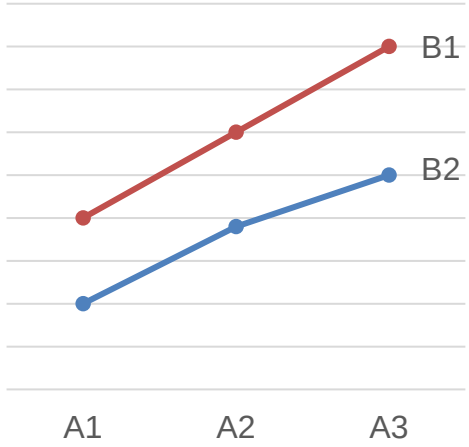
Varianzanalyse: Vorgehen

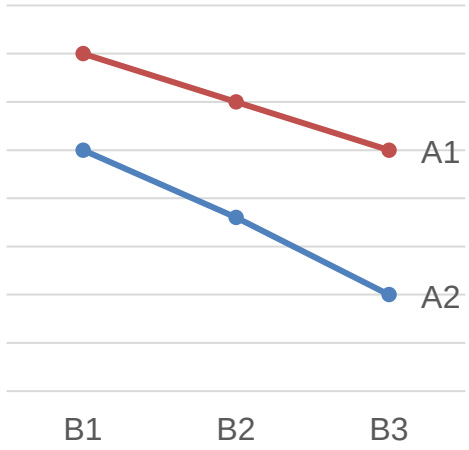
1. Modellformulierung – Aufstellung der relevanten Einflussgrößen, Parameter und Werte, Prüfung der Voraussetzungen.
2. Zerlegung der Streuung und Modellgüte – Ermittlung der Effekte der unterschiedlichen Faktoren, erklärte und nicht erklärte Streuungen.
3. Prüfung der statistischen Signifikanzen – Validität der einzelnen Effekte und Wechselwirkungen.
4. Interpretation der Ergebnisse – Bestätigung von Hypothesen (konfirmatorisch; a priori) oder Ermittlung von Wirkunterschieden (explorativ; ex post).

Praktisch: `Einfaktorielle_ANOVA.ipynb`

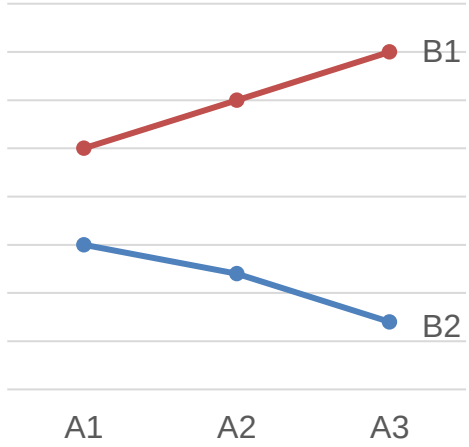
Varianzanalyse: Interaktionseffekte

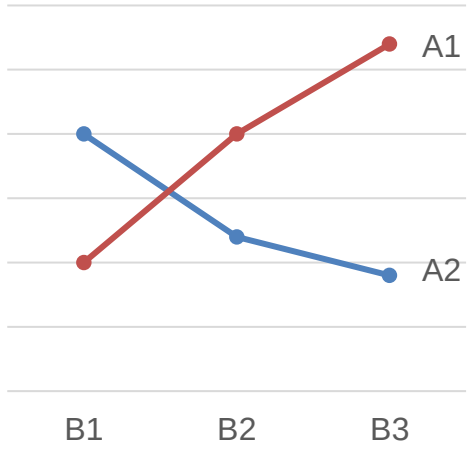
Ordinale Interaktion
(beide Haupteffekte interpretierbar)



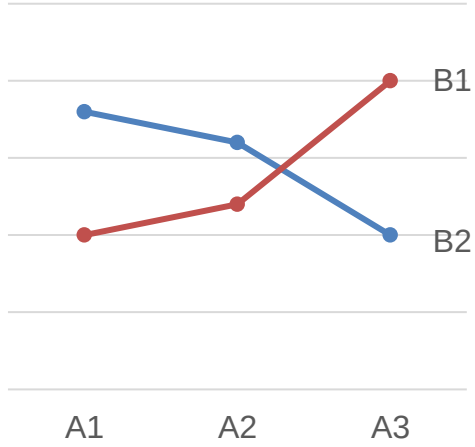


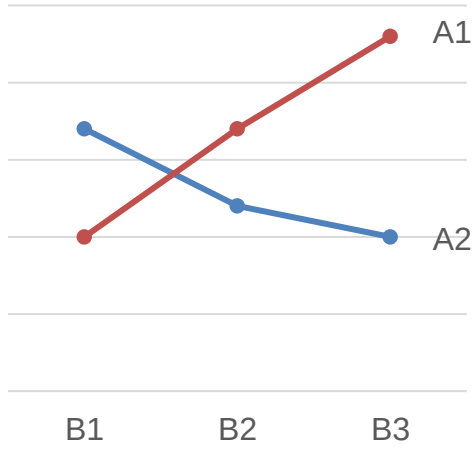
Hybride Interaktion
(nur Haupteffekt B interpretierbar)





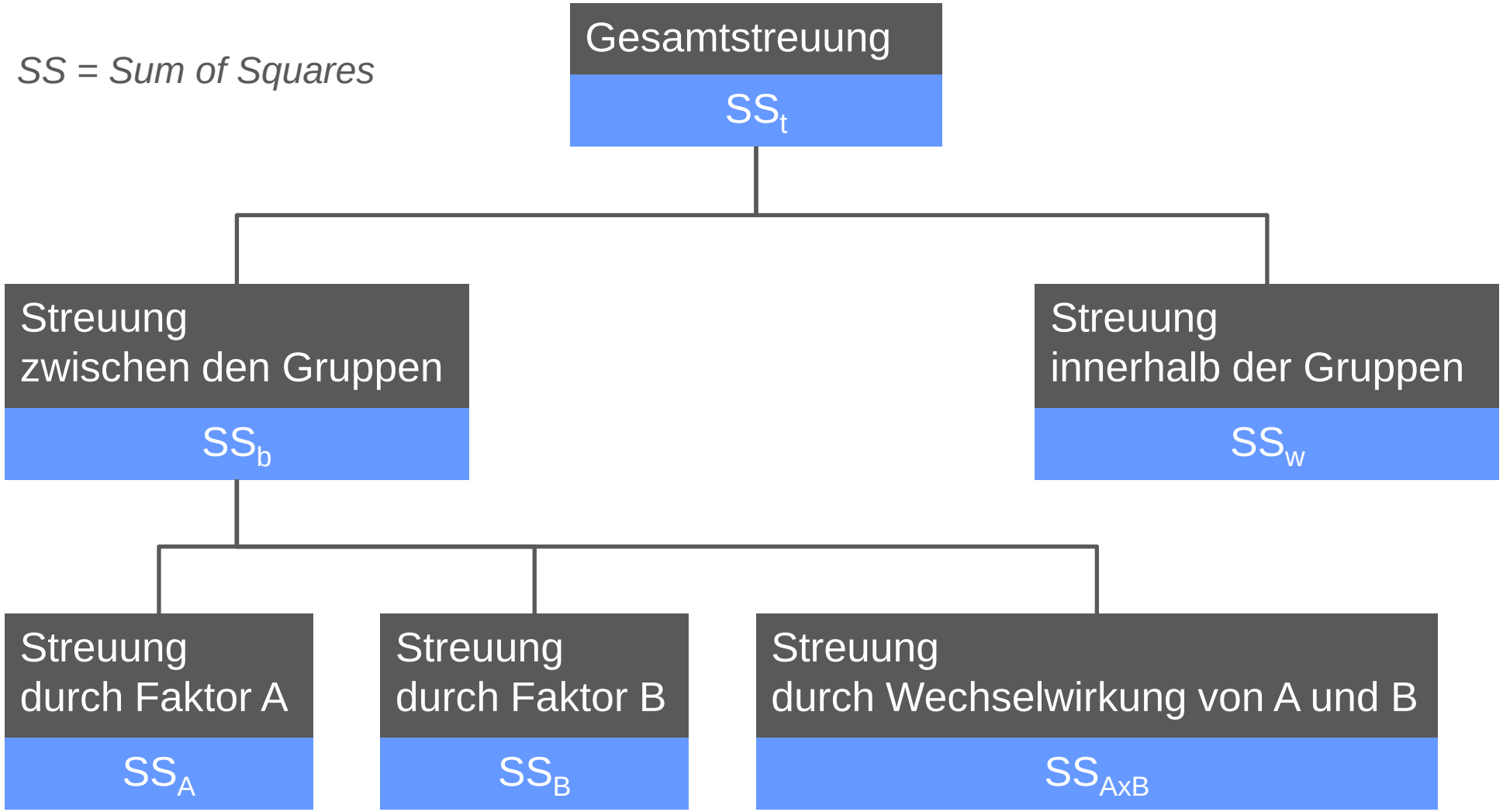
Disordinale Interaktion
(kein Haupteffekt interpretierbar)





Varianzanalyse: Streuungsaufteilung

SS = Sum of Squares



Praktisch: `Zweifaktorielle_ANOVA.ipynb`

Data-Mining-Verfahren

Nächste-Nachbarn-Algorithmus
Naives Bayes-Verfahren
Entscheidungsbaumalgorithmen
Künstliche Neuronale Netze

Data-Mining-Verfahren

Nächste-Nachbarn-Algorithmus

Naives Bayes-Verfahren

Entscheidungsbaumalgorithmen

Künstliche Neuronale Netze

k-Nächste Nachbarn (k-NN): Ansatz

k-NN dient sowohl der Klassifikation einer kategorischen Zielvariable, als auch der Prädiktion einer numerischen Zielvariable. Die Methode findet dazu „ähnliche“ Datenpunkte in den Trainingsdaten und wählt daraus eine Klassifikation oder ermittelt daraus einen Durchschnitt als Vorhersage.

Anders als z. B. die Regressionen trifft k-NN keine Annahmen über die Beziehungen zwischen Ziel- und Prädiktorvariablen und sie bildet auch keine Schätzungen in Form einer Funktion o. ä.

Ähnlichkeit ist hier der Abstand zwischen den Werten $(x_1, \dots, x_p)$ eines neuen Datenpunktes und den Werten $(u_1, \dots, u_p)$ eines Datenpunktes im Prädiktorraum. Es gibt viele Ansätze, diesen Abstand zu ermitteln, der populärste – braucht wenig Rechenzeit – ist der *euklidische*:

$$\sqrt{(x_1 - u_1)^2 + \cdots + (x_p - u_p)^2}$$

Dazu müssen wir i. d. R. die Prädiktoren normalisieren, damit die gleichen Skalen zu Grunde liegen.

k-Nächste Nachbarn (k-NN): Verfahren

Wenn wir die Abstände zwischen einem neuen Datenpunkt und den bestehenden Datenpunkten geeignet bestimmt haben, brauchen wir eine Regel zur Klassifikation: Zuordnung einer Klasse des neuen Datenpunktes, basierend auf den Klassen seiner Nachbarn.

Der einfachste Fall ist $k = 1$, in dem wir den nächsten Nachbarn und dessen Klasse auswählen. In der Praxis erweist sich schon dieser einfache, effiziente Ansatz als sehr effektiv.

Verallgemeinert ergibt sich $k > 1$:

1. Ermitteln der k nächsten Nachbarn des zu klassifizierenden Datenpunktes.
2. Klassifizieren nach der Klasse, der die Mehrheit der k Nachbarn angehört.

k-Nächste Nachbarn (k-NN): Erweiterungen

k-NN mit mehr als zwei Klassen

Aus den Verhältnissen der Klassenzugehörigkeiten der Nachbarn lassen sich Wahrscheinlichkeiten ableiten. Definieren wir eine Schwelle für die Zugehörigkeit, können wir auf dieser Basis mehr als zwei Klassen berücksichtigen.

k-NN mit einer numerischen Zielvariable

Auch hier bestimmen wir zunächst die nächsten Nachbarn wie gewohnt. Statt dann eine Mehrheitsklasse zu ermitteln, können wir aber auch Durchschnitte numerischer Werte der Nachbarn als Prädiktion bilden. Diese können über die zugehörigen Distanzen auch gewichtet werden (nahe Werte stärker berücksichtigen).

Praktisch: `k-Naechste_Nachbarn.ipynb`

Data-Mining-Verfahren

Nächste-Nachbarn-Algorithmus

Naives Bayes-Verfahren

Entscheidungsbaumalgorithmen

Künstliche Neuronale Netze

Bayes-Klassifikation: Ansatz (1)

Die naive Bayes-Methode stammt aus dem großen Werkzeugkasten der Bayes-Statistik.

Zunächst der **vollständige bzw. exakte Bayes-Klassifizierer, mit dem wir Datenpunkte bestimmten Klassen zuordnen wollen**. Für jeden Datenpunkt:

1. Alle Datenpunkte mit dem gleichen Prädiktorprofil ermitteln, d. h. *mit den gleichen Prädiktorwerten*.
2. Bestimmen, zu welchen Klassen diese Datenpunkte gehören und welche Klasse vorherrscht.
3. Diese Klasse wird dem Datenpunkt zugewiesen.

Bayes-Klassifikation: Ansatz (2)

Diese Klassifizierung lässt sich verfeinern auf die Fragestellung, **mit welcher Wahrscheinlichkeit Datenpunkte zu einer bestimmten Klasse gehören**, wenn uns nicht einfach die häufigste interessiert.

Eine solche Zuordnung können wir über Schwellenwerte/Cutoff Probabilities vornehmen:

1. Definieren eines Cutoff für die interessante Klasse, über dem wir ihr einen Datenpunkt zuordnen.
2. Alle Trainingsdatenpunkte finden, die das *gleiche Prädiktorprofil* besitzen, wie das eines neuen, zu klassifizierenden Datenpunktes.
3. Wahrscheinlichkeit ermitteln, dass diese Trainingsdatenpunkte zur interessanten Klasse gehören.
4. Ist diese Wahrscheinlichkeit > Cutoff, gehört der neue Datenpunkt zu dieser Klasse.

Bayes-Klassifikation: Bedingte Wahrscheinlichkeit

In beiden Fällen finden wir das Konzept der bedingten Wahrscheinlichkeit eines Ereignisses A unter der Voraussetzung, dass B stattgefunden hat: $P(A|B)$.

Im hier gegebenen Fall ist das die Wahrscheinlichkeit, dass ein Datenpunkt mit den Prädiktorwerten $x_1, \dots, x_p$ zu einer Klasse aus $C_1, \dots, C_m$ gehört: $P\left(C_i|x_1, \dots, x_p\right)$

Um einen Datenpunkt auf diese Weise zu klassifizieren, würden wir für jede Klasse die Zugehörigkeitswahrscheinlichkeit bestimmen und ihn dann entweder der Klasse mit der höchsten Wahrscheinlichkeit, oder einer Klasse, die einen definierten Cutoff überschreitet zuordnen.

Wir sehen hier, dass die Bayessche Klassifizierung nur mit kategorischen Prädiktoren funktioniert.

Bayes-Klassifikation: Beispiel 1

In wie weit können frühere rechtliche Probleme auf betrügerische Bilanzen hinweisen?

	Rechtliche Probleme ($X = 1$)	**Keine rechtlichen Probleme ($X = 0$)**	**Summe**
Betrügerisch (C_1)	50	50	100
Ehrlich (C_2)	180	720	900
Summe	230	770	1000

Bekommen wir eine neue Bilanz, klassifizieren wir sie, indem wir die Wahrscheinlichkeiten der Klassenzugehörigkeiten berechnen: Wenn sie von einem Unternehmen mit rechtlichen Problemen kommt $P(Betrug|Problem) = 50/230$, sonst $P(Ehrlich|Problem) = 180/230$.

Würden wir hier also den Klassifikationsansatz der wahrscheinlichsten Klasse wählen, wäre die Einordnung immer *ehrlich*.

Bayes-Klassifikation: Beispiel 1

Hier interessieren wir uns insbesondere für die potentiellen Betrüger. Im gegebenen Fall lassen sich die bedingten Wahrscheinlichkeiten einfach aus der Tabelle ermitteln. Allgemein gilt aber der *Satz von Bayes*:

$$P(Betrug|Problem) = \frac{P(Problem|Betrug)P(Betrug)}{P(Problem|Betrug)P(Betrug) + P(Problem|Ehrlich)P(Ehrlich)}$$

$$= \frac{50/100 \cdot 100/1000}{\frac{50}{100} \cdot \frac{100}{1000} + 180/900 \cdot 900/1000} = \frac{0.5 \cdot 0.1}{0.5 \cdot 0.1 + 0.2 \cdot 0.9} = \frac{0.05}{0.23} = 0.22$$

Legen wir den Cutoff z. B. bei 0.2 fest, würden wir die juristischen Problemfälle nun als potentiellen Betrug klassifizieren. Je nach Modellperformance und Kosten für Fehlentscheidungen können wir diesen Wert verschieben.

Problem dieser exakten Bayes-Prozedur: Es ist in diesem einfachen Beispiel und mit wenigen Prädiktorvariablen möglich, alle Datenpunkte zu ermitteln, die *genau die gleichen* Werte bzw. Eigenschaften wie ein zu bewertender Datenpunkt besitzen. Schon bei 20 Prädiktoren gibt es aber schnell Lücken – und nur "ähnlich" reicht nicht.

Naives Bayes-Verfahren: Ansatz

Beim naiven Bayes-Verfahren beschränken wir uns nicht mehr auf die Datenpunkte, die *genau* mit dem zu klassifizierenden übereinstimmen, sondern wir betrachten die gesamte Datenmenge, um die Klassifikation vorzunehmen. Das Vorgehen ist nun:

1. Für eine Klasse C_1 die bedingten Wahrscheinlichkeiten $P(x_j|C_1)$ für jeden einzelnen Prädiktor ermitteln. Das ist die Wahrscheinlichkeit, dass der Prädiktorwert im zu klassifizierenden Datenpunkt in Klasse C_1 vorkommt.
2. Diese Wahrscheinlichkeiten miteinander multiplizieren, dann mit dem Verhältnis der Datenpunkte, die zu C_1 gehören.
3. Schritte 1 und 2 für alle Klassen wiederholen.
4. Eine Wahrscheinlichkeit für C_i schätzen als Division des Wertes für C_i aus Schritt 2 durch die Summe dieser Werte für alle Klassen.
5. Zuordnung der Klasse mit der höchsten Wahrscheinlichkeit für die Prädiktorwerte des zu klassifizierenden Datenpunktes.

Der Ansatz ist effizient, erfordert in der Praxis aber große Datenmengen.

Bayes-Klassifikation: Beispiel 2

Wir erweitern unser Beispiel auf zwei Prädiktoren.

Unternehmen	Rechtliche Probleme	Unternehmens-größe	Status
1	Ja	Klein	Ehrlich
2	Nein	Klein	Ehrlich
3	Nein	Groß	Ehrlich
4	Nein	Groß	Ehrlich
5	Nein	Klein	Ehrlich
6	Nein	Klein	Ehrlich
7	Ja	Klein	Betrug
8	Ja	Groß	Betrug
9	Nein	Groß	Betrug
10	Ja	Groß	Betrug

Exakter Bayes:

$$P(Betrug|Problem = Ja, Groesse = Klein) = 1/2 = 0.5$$
$$P(Betrug|Problem = Ja, Groesse = Gross) = 2/2 = 1$$
$$P(Betrug|Problem = Nein, Groesse = Klein) = 0/3 = 0$$
$$P(Betrug|Problem = Nein, Groesse = Gross) = 1/3 = 0.33$$

Naiver Bayes:

$$P_{nb}(Betrug|Problem, Klein) = \frac{P(Betrug)[P(Problem|Betrug)P(Klein|Betrug)]}{P(Betrug)[P(Problem|Betrug)P(Klein|Betrug)]+P(Ehrlich)[P(Problem|Ehrlich)P(Klein|Ehrlich)]} = \frac{(4/10)(3/4)(1/4)}{(4/10)(3/4)(1/4)+(6/10)(1/6)(4/6)} = 0.53$$

$$P_{nb}(Betrug|Problem, Gross) = 0.87$$
$$P_{nb}(Betrug|kein\ Problem, Klein) = 0.07$$
$$P_{nb}(Betrug|kein\ Problem, Gross) = 0.31$$

Die Werte beider Ansätze sind nahe beieinander. Mit einem Cutoff von z. B. 0.5 würden sie die gleichen Klassifikationen liefern.

Praktisch: `Naiver_Bayes.ipynb`

Data-Mining-Verfahren

Nächste-Nachbarn-Algorithmus

Naives Bayes-Verfahren

Entscheidungsbaumalgorithmen

Künstliche Neuronale Netze

Klassifikations- und Regressionsbäume: Ansatz

Entscheidungsbäume (*Classification And Regression Trees, CART*, Breiman et al. 1984) sind ein gut automatisierbarer Ansatz, dessen Ergebnisse auch ohne Kenntnisse der Methodik recht einfach nachvollziehbar sind. Daher sind sie prominent im Bereich der *eXplainable Artificial Intelligence* (XAI).

Sie teilen die Prädiktordaten in Gruppen auf, aus denen WENN-DANN-Bedingungen entstehen, z. B. „WENN qm > 100 UND Ort = Paderborn DANN Klasse = 1“. Entsprechend flexibel ist der Typ der Prädiktorvariablen.

Gute Bäume benötigen viele Daten und sind rechenintensiv zu generieren. Aber einmal generiert, sind Bäume auch im Falle großer Datensätze laufzeiteffizient. Wir unterscheiden

- **Klassifikationsbäume** – die neue Datenpunkte einer Kategorie zuordnen (kategorische Zielvariable) und
- **Regressionsbäume** – die neue Datenpunkte einem numerischen Wert (numerische Zielvariable) zuordnen.

Beide Arten bestehen aus *Entscheidungsknoten*, die einen Datenpunkt einer bestimmten Datenteilmenge zuordnen und *Endknoten*, die die Zuordnung mit einem Wert abschließen.

Bäume berücksichtigen nicht mögliche Beziehungen zwischen Prädiktoren, was Vorteile (anwendbar für viele solche Zusammenhänge), aber auch Nachteile haben kann (Auswirkungen auf die Vorhersagen werden nicht gut erfasst).

Praktisch: `Entscheidungsbaeume.ipynb`

Data-Mining-Verfahren

Nächste-Nachbarn-Algorithmus

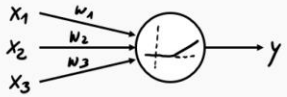
Naives Bayes-Verfahren

Entscheidungsbaumalgorithmen

Künstliche Neuronale Netze

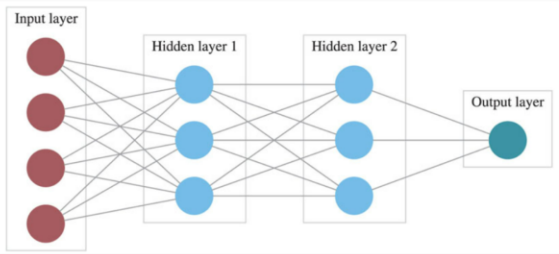
Künstliche Neuronale Netze: Kennen Sie bereits

Inspiration

Neuron (Biologie)	Künstliches Neuron
	
Das menschliche Gehirn besteht aus vielen verbundenen Neuronen.	KNNs bestehen aus vielen verbunden künstlichen Neuronen
• Kommunikation über elektrische Signale • Addition von ankommenden Signalen und „Feuern“, wenn Signal stark genug • Lernen durch Verstärken oder Abschwächen der Signale	• Kommunikation über verknüpfte Berechnungen • Addition von (gewichteten) ankommenden Werten und angepasste Weiterleitung • Lernen durch Anpassen der verwendeten Gewichte

Feedforward Netze

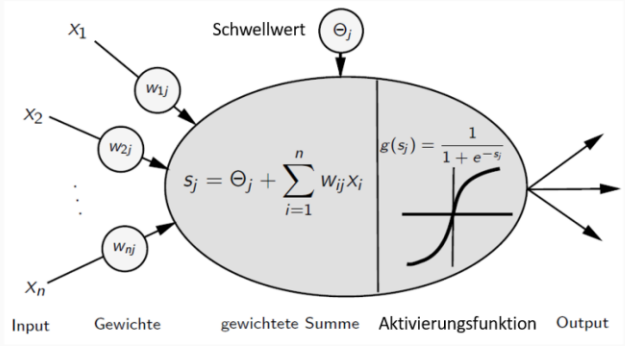
Viele erfolgreichen Anwendungen von KNNs basieren auf „Feedforward“ Netzwerken (FNNs).



Quelle: Machine Learning for Business Analytics (Shmueli et. al.)

Der Input-Layer wird mit Zahlen zwischen 0 und 1 belegt, welche von Schicht zu Schicht weiter verarbeitet werden. Ergebnis ist eine Zahl zwischen 0 und 1 im Output Layer.

Die Berechnung in einen Perzeptron

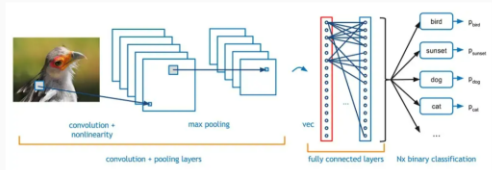


Lernprozesse benötigen einen Output, der bei geringen Änderungen an den Parametern selbst auch nur eine geringe Änderung zeigt. Dieses wird durch den Einsatz von Aktivierungsfunktionen (Squashing Function, Transferfunktion) gesteuert.

Deep Learning und Convolutional Neural Networks

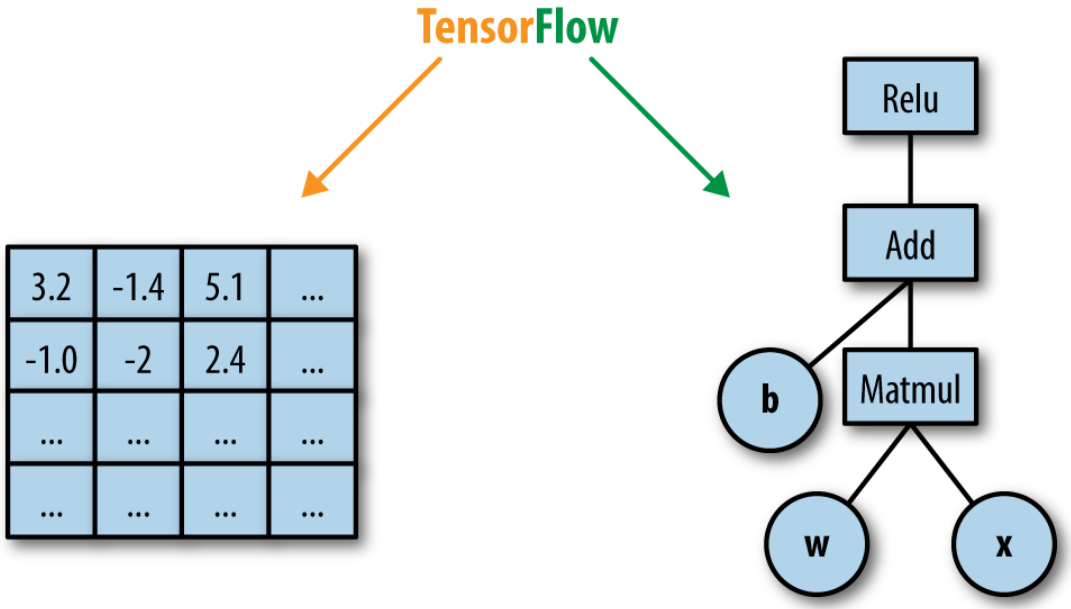
Enthält ein KNN viele verborgene Schichten, nennt man es ein *tiefes neuronales Netz* (Deep Neural Network, DNN). Wird ein DNN eingesetzt, spricht man von **Deep Learning**.

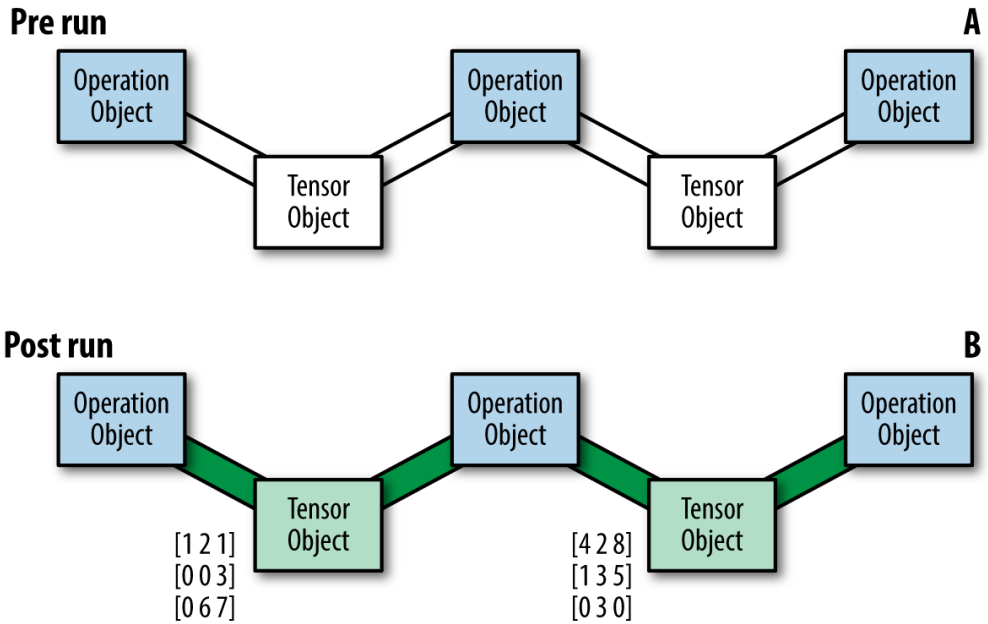
Die betrachteten Feed Forward Netze (FNN) funktionieren für größere Bilder nicht, da zu viele Verbindungen erforderlich wären. Abhilfe schaffen die **Convolutional Neural Networks** (CNN). Schichten sind in 2-D angeordnet und nur teilweise verbunden.



https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

Künstliche Neuronale Netze: TensorFlow





Hope et al.: Learning Tensor Flow. O'Reilly, 2018.

Künstliche Neuronale Netze: Keras

High-level Python-Schnittstelle für TensorFlow, vereinfacht Erstellung, Training und Deployment von Modellen. Zusätzlich Werkzeuge für Optimierung, domänenspezifische Modelle und Cloud-Nutzung.

Modelle sind azyklische, gerichtete Graphen aus Schichten, um Eingaben in die gewünschten Ausgaben umzuwandeln. Keras bietet zwei Formen von Modellen:

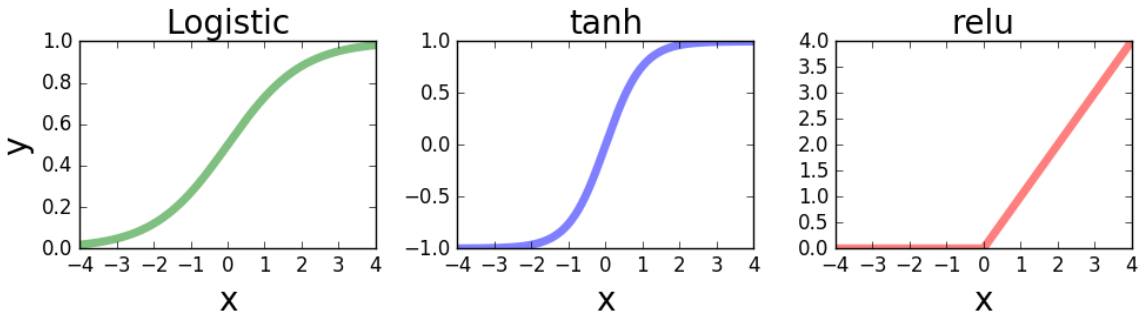
- Sequential API – einfacher Layer Stack.
- Functional API – erlaubt komplexere, auch parallele Flows.

Schichten entsprechen Transformationen von Tensoren. Keras bringt viele schnell einsetzbare, komplexe Schichten mit, von Preprocessing mit *Normalisierung* und *Rescaling* über *Density* bis *Dropout*.

Die *Dense*-Schicht repräsentiert die bekannte Aktivierungsfunktion der KNN:

$$y = \text{activation}(W \cdot x + b)$$

mit dem (durch das KNN ermittelte) Gewicht W, Bias b, Matrixmultiplikation $\cdot$



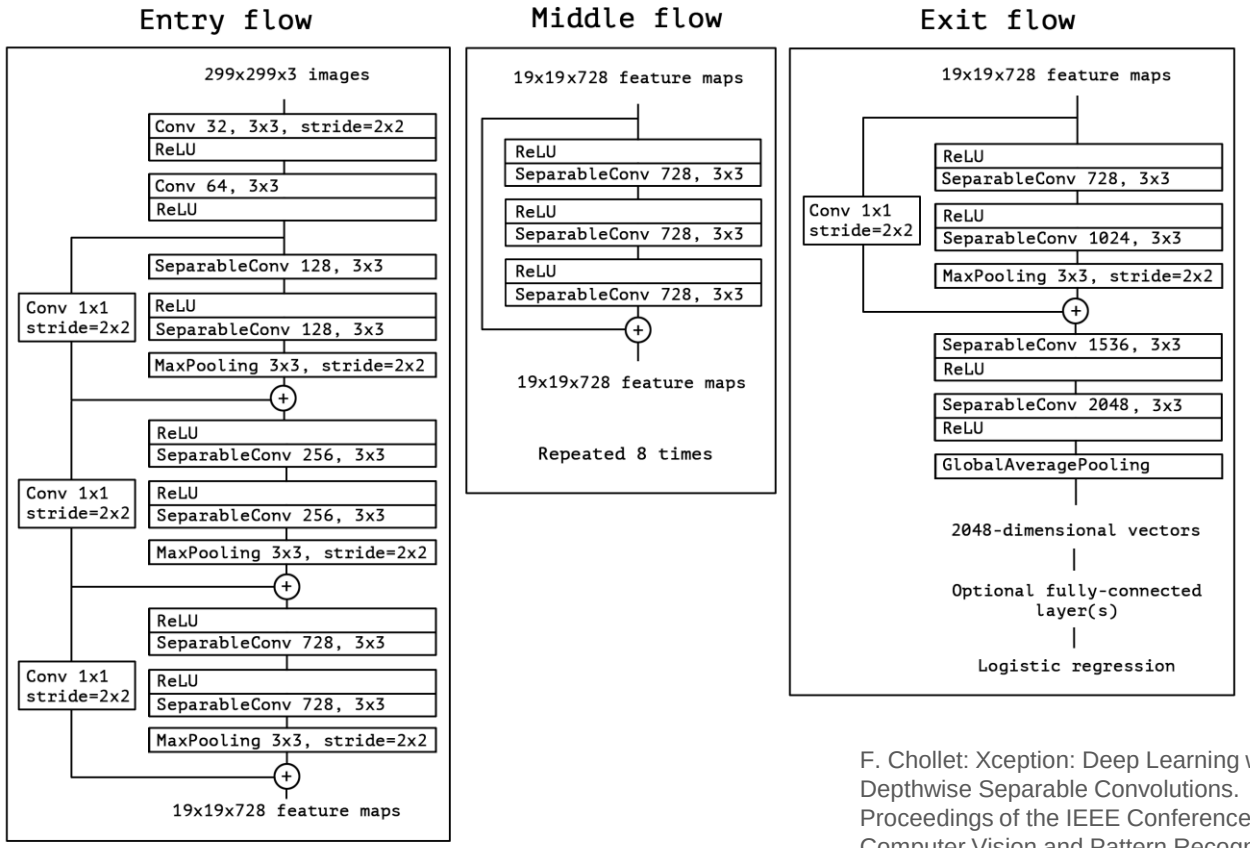
Praktisch: `Neuronale_Netze_Einfuehrung.ipynb`

Künstliche Neuronale Netze: Inception Networks

Deep Neural Network (DNN) mit sich wiederholenden Modulen, als eine Weiterentwicklung der *Convolutional NN*.

Inception Modules führen mehrfache, verschiedene Transformationen über den gleichen Input aus und lassen das Modell entscheiden, welche relevanten Features der kombinierte Output besitzt.

Die Dimensionen der Convolutions werden dabei reduziert und die Transformationen so angeordnet, dass sie parallel erfolgen können, also auch die Tiefe des Netzwerks reduziert wird.



	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	**0.790**	**0.945**

F. Chollet: Xception: Deep Learning with Depthwise Separable Convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1251-1258

Praktisch: `Neuronale_Netze_Bilderkennung.ipynb`

Cluster-Analyse

Cluster-Analyse: Ansatz

Ziel allgemein: Aus Datenpunkten **homogene Gruppierungen** bilden, um Einblicke in die Strukturen der Daten zu gewinnen. „Homogene Gruppierung“ = Datenpunkte mit gleichen oder ähnlichen Eigenschaften sind sich „nahe“.

Im Rahmen unserer Vorhersagen können wir so

- **spezialisiertere Modelle** für bestimmte Gruppen, also Teilmengen von Daten konstruieren und ihre Eigenschaften besser ausnutzen und
- **neue Gruppen als Klassen** von Daten für eine spätere Klassifikation ermitteln.

Allgemein gibt es zwei Arten von Gruppierungsalgorithmen für Datensätze:

1. **Hierarchische Methoden** – *agglomerativ*, *n* Cluster sukzessiv zusammenlegen, oder *divisiv*, ein Cluster mit allen Daten sukzessiv zerlegen.
2. **Partitionierende Methoden** – Datenpunkte werden auf eine vorgegebene Cluster-Anzahl aufgeteilt.

Cluster-Analyse: Abstände zwischen numerischen Datenpunkten

Sei d_{ij} eine *Distanzmetrik*, oder ein *Ungleichsmaß* zwischen Datenpunkten i und j. Für i haben wir den Vektor aus p Werten $(x_{i1}, x_{i2}, \dots, x_{ip})$, für j $(x_{j1}, x_{j2}, \dots, x_{jp})$.

Euklidische Distanz $d_{ij} = \sqrt{(x_{i1} - x_{j1})^2 + \cdots + (x_{ip} - x_{jp})^2}$

Korrelationsbasierte Ähnlichkeit $d_{ij} = 1 - r_{ij}^2$
mit dem *Pearsonschen Korrelationskoeffizienten* r.

Statistische oder **Mahalanobis-Distanz** $d_{ij} = \sqrt{(x_i - x_j)^T S^{-1} (x_i - x_j)}$, mit x_i, x_j den p-dimensionalen Vektoren der Werte der Datenpunkte i und j, S ihrer Kovarianzmatrix.

Manhattan-Distanz ("city block") $d_{ij} = \sum_{m=1}^{p} |x_{im} - x_{jm}|$

Maximaler Koordinatenabstand $d_{ij} = max_{m=1,\dots,p} |x_{im} - x_{jm}|$

Cluster-Analyse: Abstände zwischen kategorischen Datenpunkten

Im binären Fall nutzen wir Ähnlichkeit statt Distanz. Wenn wir p binäre Werte für beide Datenpunkte i und j haben, können wir folgende Tabelle bilden:

		$Record\ j$		
		0	1	
$Record\ i$	0	a	b	$a+b$
	1	c	d	$c+d$
		$a+c$	$b+d$	p

a bezeichnet die Anzahl der Variablen, deren Attribut weder i noch j besitzen (d. h. bei beiden ist der Wert 0), d hingegen die Anzahl an Variablen, deren Attribut beide besitzen usw.

Kennzahlen für Ähnlichkeit sind dann

- **Übereinstimmungskoeffizient** $(a+d)\ /\ p$
- **Jaquards Koeffizient** $d\ /\ (b+c+d)$, ignoriert Übereinstimmungen mit 0, da die Abwesenheit von Attributen nicht unbedingt eine Übereinstimmung implizieren muss.

Zur Behandlung von nominal skalierten Variablen gibt es zwei Möglichkeiten:

1. Transformation in binäre Variablen – ähnlich wie die Dummy-Variablen als eine Art Bitmuster.
2. Analyse der Häufigkeiten von Merkmalsausprägungen – zur Prüfung von Abhängigkeiten zwischen nominalen Variablen.

Cluster-Analyse: Abstände zwischen gemischten Datenpunkten

Kommen im Datensatz sowohl numerische, als auch kategorische Variablen vor, können wir **Gowers Ähnlichkeitsmaß** nutzen:

$$s_{ij} = \frac{\sum_{m=1}^{p} w_{ijm} s_{ijm}}{\sum_{m=1}^{p} w_{ijm}}$$

Es definiert einen Durchschnitt aus den gewichteten Abständen jeder auf $[0,1]$ skalierten Variable. s_{ijm} ist die Ähnlichkeit von Merkmal m zwischen i und j, w_{ijm} ist das binäre Gewicht dieses Werts. Berechnung:

1. Für numerische Variablen $s_{ijm} = 1 - \frac{|x_{im} - x_{jm}|}{max(x_m) - min(x_m)}$ und $w_{ijm} = 1$, wenn alle Merkmalswerte bekannt sind, sonst ist $w_{ijm} = 0$.
2. Für binäre kategorische Variablen $s_{ijm} = 1$, wenn $x_{im} = x_{jm} = 1$, sonst 0. $w_{ijm} = 1$, außer $x_{im} = x_{jm} = 0$.
3. Für nicht-binäre kategorische Variablen $s_{ijm} = 1$, wenn beide Datenpunkte der gleichen Kategorie angehören, sonst $s_{ijm} = 0$. $w_{ijm} = 1$, wenn alle Kategorien bekannt sind, sonst $w_{ijm} = 0$ (also wie unter 1).

Cluster-Analyse: Abstände zwischen Gruppierungen

Angenommen seien Cluster A mit m Datenpunkten $A_1, A_2, \dots, A_m$ und Cluster B mit n Datenpunkten $B_1, B_2, \dots, B_n$.

Minimaler Abstand

Die Distanz zwischen dem Paar A_i und B_j, das sich am nächsten ist:

$$min\left(distance\left(A_i, B_j\right)\right), \mathrm{i} = 1, \dots, \mathrm{m}; \mathrm{j} = 1, \dots, \mathrm{n}$$

Maximaler Abstand

Die Distanz zwischen dem Paar A_i und B_j, das am weitesten voneinander entfernt ist:

$$max\left(distance\left(A_i, B_j\right)\right), \mathrm{i} = 1, \dots, \mathrm{m}; \mathrm{j} = 1, \dots, \mathrm{n}$$

Durchschnittlicher Abstand

Durchschnittliche Distanz aller möglicher Abstände zwischen Punkten in einem Cluster von denen im anderen Cluster:

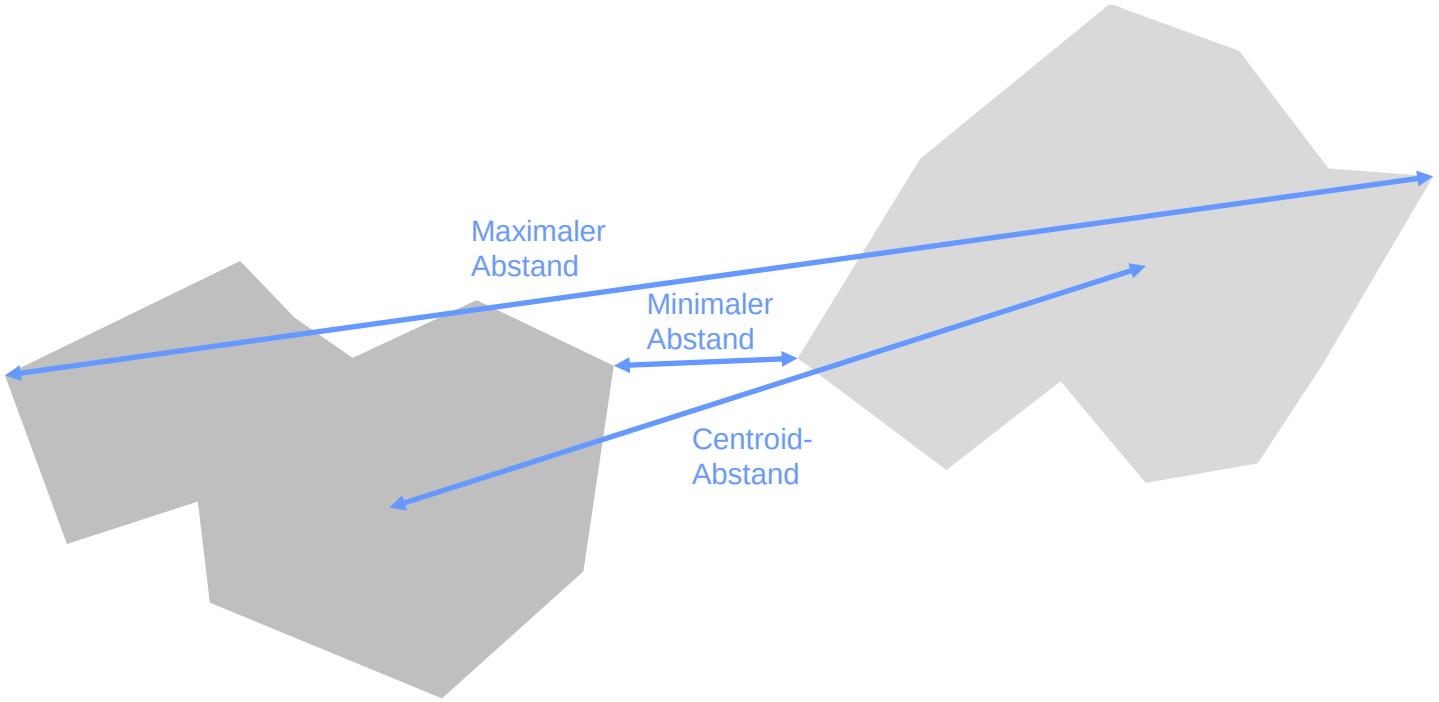
$$average\left(distance\left(A_i, B_j\right)\right), \mathrm{i} = 1, \dots, \mathrm{m}; \mathrm{j} = 1, \dots, \mathrm{n}$$

Cluster-Analyse: Abstände zwischen Gruppierungen

Centroid-Abstand

Ein Cluster-Centroid ist der Vektor der Variablendurchschnitte über alle Punkte in einem Cluster. Der Centroid-Abstand $distance(\bar{x}_A, \bar{x}_B)$ ergibt sich aus

$\bar{x}_A = \left(1/m \sum_{i=1}^{m} x_{i1}, \ldots, 1/m \sum_{i=1}^{m} x_{ip}\right)$ und $\bar{x}_B = \left(1/n \sum_{j=1}^{n} x_{j1}, \ldots, 1/n \sum_{j=1}^{n} x_{jp}\right)$



Cluster-Analyse: Wahl der Abstandsberechnung

Allgemein gibt es keine „besseren“ oder „schlechteren“ Ansätze der Abstandsermittlung.

Im Speziellen aber natürlich schon, d. h. hier ist domänenspezifisches Fachwissen entscheidend.

Gibt es z. B. plausible Annahmen über eine Kettenform der Cluster, ist der minimale Abstand eine gute Wahl: Es müssen nicht alle Punkte nah beieinander sein, sondern nur jeweils ein neuer Punkt nah an irgend einem bestehenden. Beispiele: Acker-Pflanzungen, Minensuchen, Infektionen über Verkehrsrouten.

Maximum und Durchschnitte funktionieren besser, wenn Cluster als eher kugelförmig angenommen werden. Allgemein sind sie eine gute Ausgangsposition für Analysen von noch unbekannten Datensätzen, da Cluster oft in Richtung Kugel tendieren.

Praktisch: `Cluster-Analyse.ipynb`