FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI

Einführung in Data Science – Zusammenfassung B1-B3

Block1
Big Data und NoSQL für die Datenanalyse

Data Science

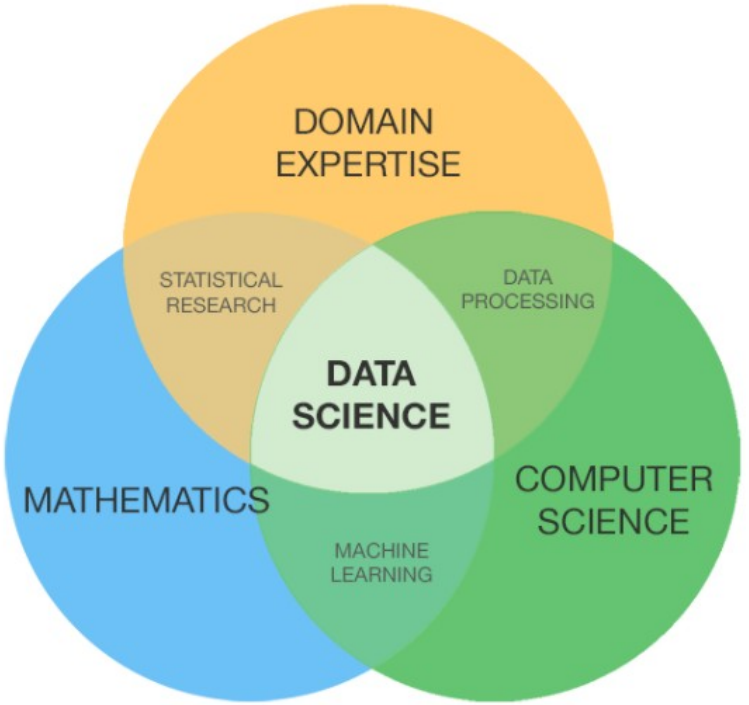
- **Data Science beschäftigt sich mit einer zweckorientierten Datenanalyse und der systematischen Generierung von Entscheidungshilfen und -grundlagen.**

- **Interdisziplinäres Forschungsfeld**

- Informatik, Statistik, Mathematik
- Naturwissenschaften
- Wirtschaftwissenschaften

- **Kompetenzen**

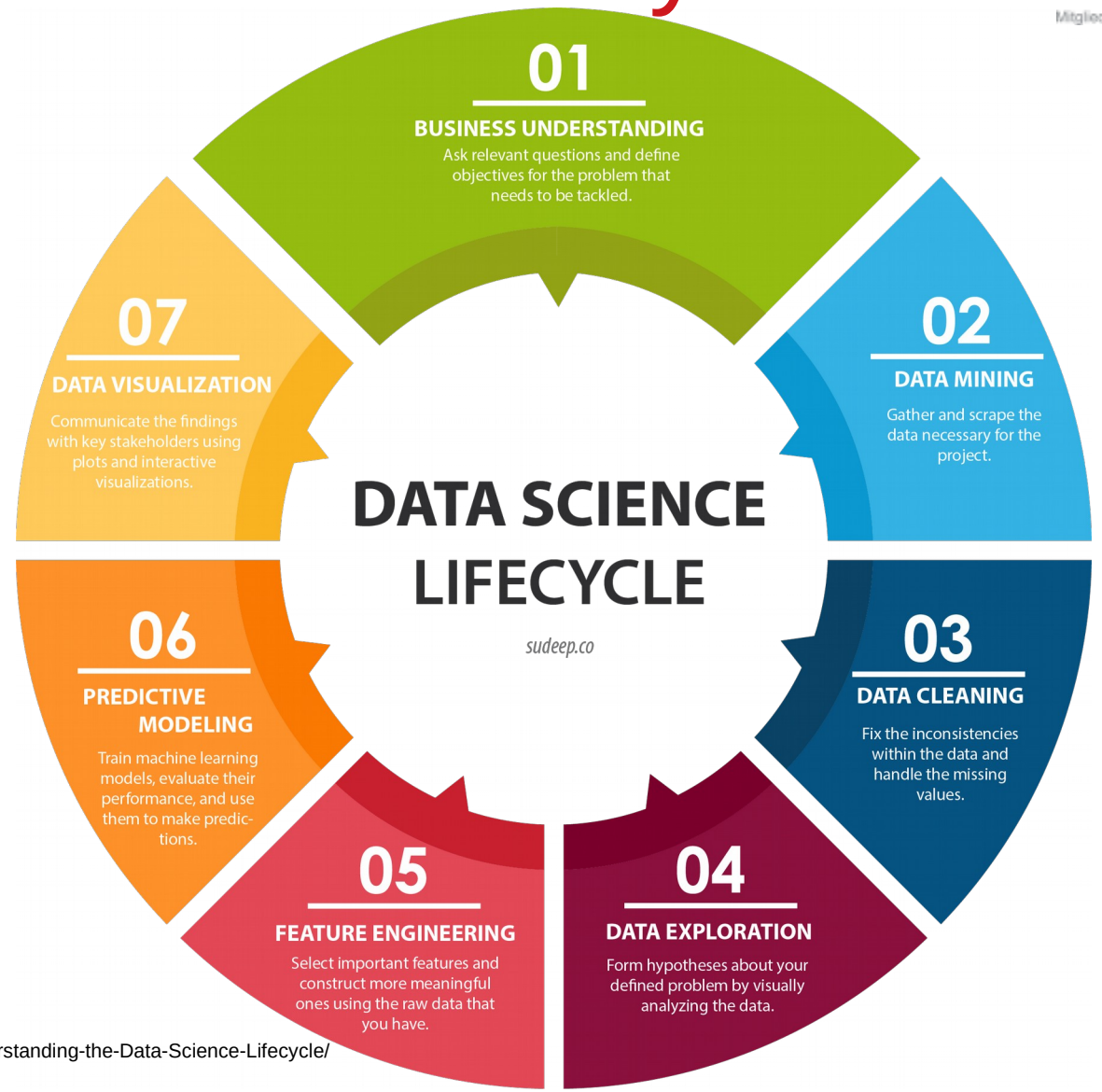
- Programmierung
- Datenanalyse
- Mustererkennung
- Prognostik
- Modellierung
- Sicherheit
- Ethik
- Datenlagerung



https://towardsdatascience.com/data-science-interview-guide-4ee9f5dc778

VIDEO

Data Science Lifecycle



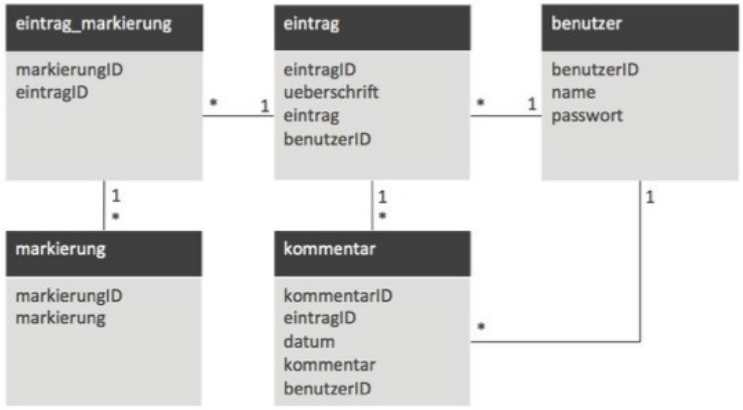
http://sudeep.co/data-science/Understanding-the-Data-Science-Lifecycle/

Big Data

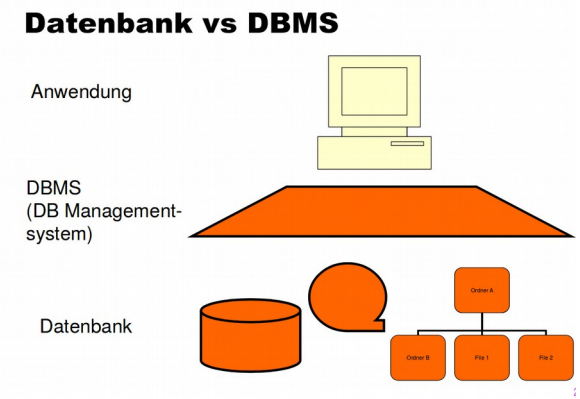
- Big Data bezeichnet Methoden und Technologien für die hochskalierbare Erfassung, Speicherung und Analyse polystrukturierter Daten

Relationale Datenbanken (SQL)

- Daten sind in Tabellen gespeichert (Spalten und Zeilen)
- SQL-Grenzen: Skalierbarkeit, Distribution, Umgang mit semi-strukturierten Daten.
- Reif und gut verstanden
- Es kann komplexe Transaktionen behandel



- **Transaktionen**
- Die Datenbankverarbeitung erfolgt durch Transaktionen
- Eine Folge von DB-Operationen (INSERT, UPDATE, DELETE,…)
- Die hinterlassen den Datenbestand nach fehlerfreier und vollständiger Ausführung in einem konsistenten Zustand.
- Transaktionen unterstützen: Datenkonsistenz, Mehrbenutzerbetrieb, Lesekonsistenz, Fehlererholung

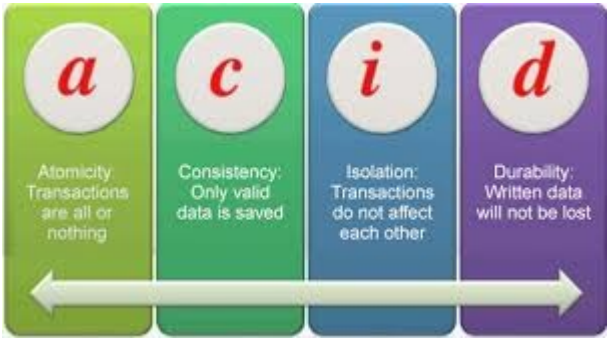


ACID Eigenschaften (RDB)

- Bezüglich der Ausführung von Transaktionen garantieren Datenbanksysteme die Einhaltung des sogenannten *Transaktionskonzept.* Dies betrifft die automatische Gewährleistung der folgenden vier Eigenschaften:

A	Atomarität (*atomicity*)
C	Konsistenz (*consistency*)
I	Isolation (*isolation*)
D	Dauerhaftigkeit (*durability*)

- **1) Atomarität (Alles oder nichts):** Änderungen einer Transaktion werden entweder vollkommen oder gar nicht in die Datenbank eingebracht.
- **2) Konsistenz:** Die Transaktion ist die Einheit der Datenbankkonsistenz. Dies bedeutet, dass bei Beginn und nach der Ende der Transaktion sämtliche Integritätsbedingungen erfüllt sind.
- **3) Isolation:** Datenbankensysteme unterstützen typischerweise eine grosse Anzahl von Benutzern, die gleichzeitig auf die Datenbank zugreifen können. Trotzdem wird es garantiert, dass dadurch keine unerwünschten Nebenwirkungen eintritt (z.B. gegenseitiges Überschreiben derselben Datenbankobjekt).
- **4) Dauerhaftigkeit:** Die Dauerhaftigkeit von erfolgreich beendeten Transaktionen wird garantiert. Die dauerhafte Speicherung der Daten muss auch nach einem Systemfehler (Software-Fehler oder Hardware-Ausfall) garantiert sein ohne Datenverlusten.

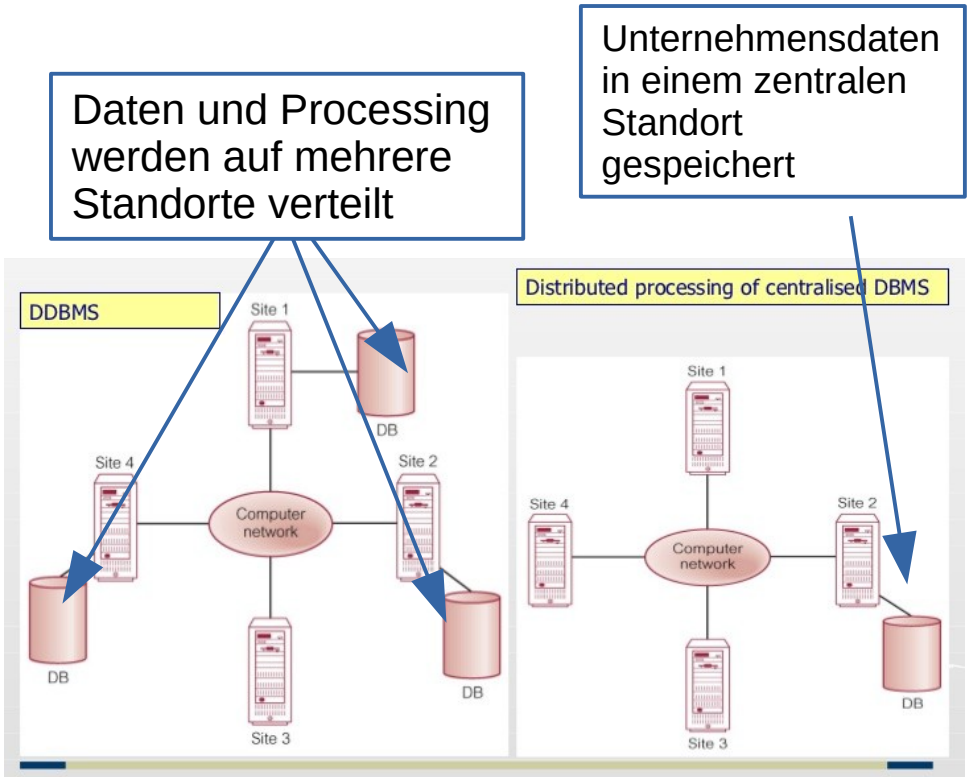


DBS-Einsatz: OLTP vs. OLAP

OLTP – Online Transaction Processing	**OLAP – Online Analytical Processing**
Viele kurze Transaktionen (queries + updates)	Lange Transaktionen (complexe queries)
Schnelle Antwortzeiten	Langsamere Antwortzeiten
Daten müssen upgedatet und konzistent sein	Daten müssen strukturiert und aggregiert werden, um sie zu analysieren
Zielt auf einen spezifischen Prozess, z.B. Bestellungen aus einem online store	Integriert Daten aus verschiedenen Prozesses, z.B. Verkäufe, Inventar und Käufe
Datenmengen in Mega- und Gigabereich	Datenmengen in Terabereich
Operational DBMS	Data Warehouse
Optimiert für Storage	Optimiert für Analyse

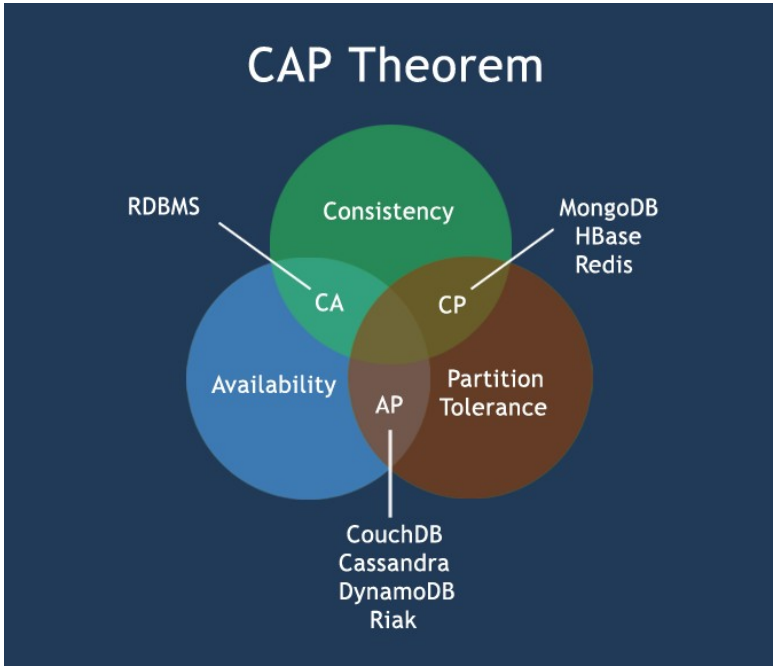
Verteilte DB

- Es dient zum Erstellen, Abrufen, Aktualisieren und Löschen.
- Es synchronisiert die Datenbank periodisch.
- Es stellt sicher, dass die an jeder Stelle geänderten Daten allgemein aktualisiert werden.
- Es wird in Anwendungsbereichen eingesetzt, bei denen grosse Datenmengen verarbeitet werden, und von zahlreichen Benutzern gleichzeitig zugegriffen.
- Es behält die Vertraulichkeit und die Integrität der Daten der Datenbanken.



Brewer’s CAP Theorem

- In einem verteilten System ist unmöglich, gleichzeitig die drei Eigenschaften Konzistenz (**C**onsistency), Verfügbarkeit (**A**vailability) und Ausfalltoleranz (**P**artition Tolerance) zu garantieren.
- Nur zwei Eigenschaften können gleichzeitig erfüllt werden



Consistency	Availability	Partition Tolerance
Konsistenz der gespeicherten Daten. Ein Update wird bei allen Usern gleichzeitig sichtbar.	Akzeptable Antwortzeiten. Das System ist jederzeit für jede gültige Anfrage verfügbar.	Das System arbeitet auch bei Ausfall eines Knotens.

BASE – Eigenschaften für verteilte DBMS

- In verteilten Datenbanken sollen alle ACID-Eigenschaften erfüllt werden?
- Diese Probleme wurden in dem CAP-Theorem von Brewer formuliert.
- Im Umfeld der NoSQL-Datenbanken wird daher häufig das BASE-Prinzip (Basically Available, Soft state, Eventual consistency) verfolgt.
- **B**asically **A**vailable (grunsätzlich Verfügbar): Das System garantiert die Verfügbarkeit im Sinne des CAP-Theorems.
- **S**oft State (loser Zustand): Der Zustand des Systems kann mit der Zeit ändern, auch ohne Eingabe. Dies ist wegen des schlussendlichen Konsistenzmodells.
- **E**ventually Consistent (schlussendlich Konsistent): Das System wird im Laufe der Zeit konsistent, angenommen dass, das System während dieser Zeit keine Eingabe erhält.
- Daten sind stets abrufbar aus der Datenbank, wobei nicht garantiert ist, dass sie sich im aktuellsten Zustand befinden, aber sie werden nach und nach aktualisiert, sodass in absehbarer Zeit Konsistenz erreicht wird.

NoSQL Datenbanken

- Der Grossteil der Daten (Web, soziale Netze, mobile Geräten) ist teilstrukturiert (E-Mail Nachrichten, Webseiten, Benutzerprofile, Produktangebote, etc.) oder unstrukturiert (z.B. Fotos, Videos)
- Die Verarbeitung solcher Daten (bzw. Generell von Big Data) mit RDBS ist ineffizient und unflexibel
- Entwicklung von sogenannten NoSQL-Systeme
- Schemafrei
- Verteilte Realisierung
- Hohe Skalierbarkeit
- Repliezierte Datenspeicherung

  → Für höhe Verfügbarkeit
- Kein ACID
- Open-Source

Aggregat / Collection 1: »Einträge«

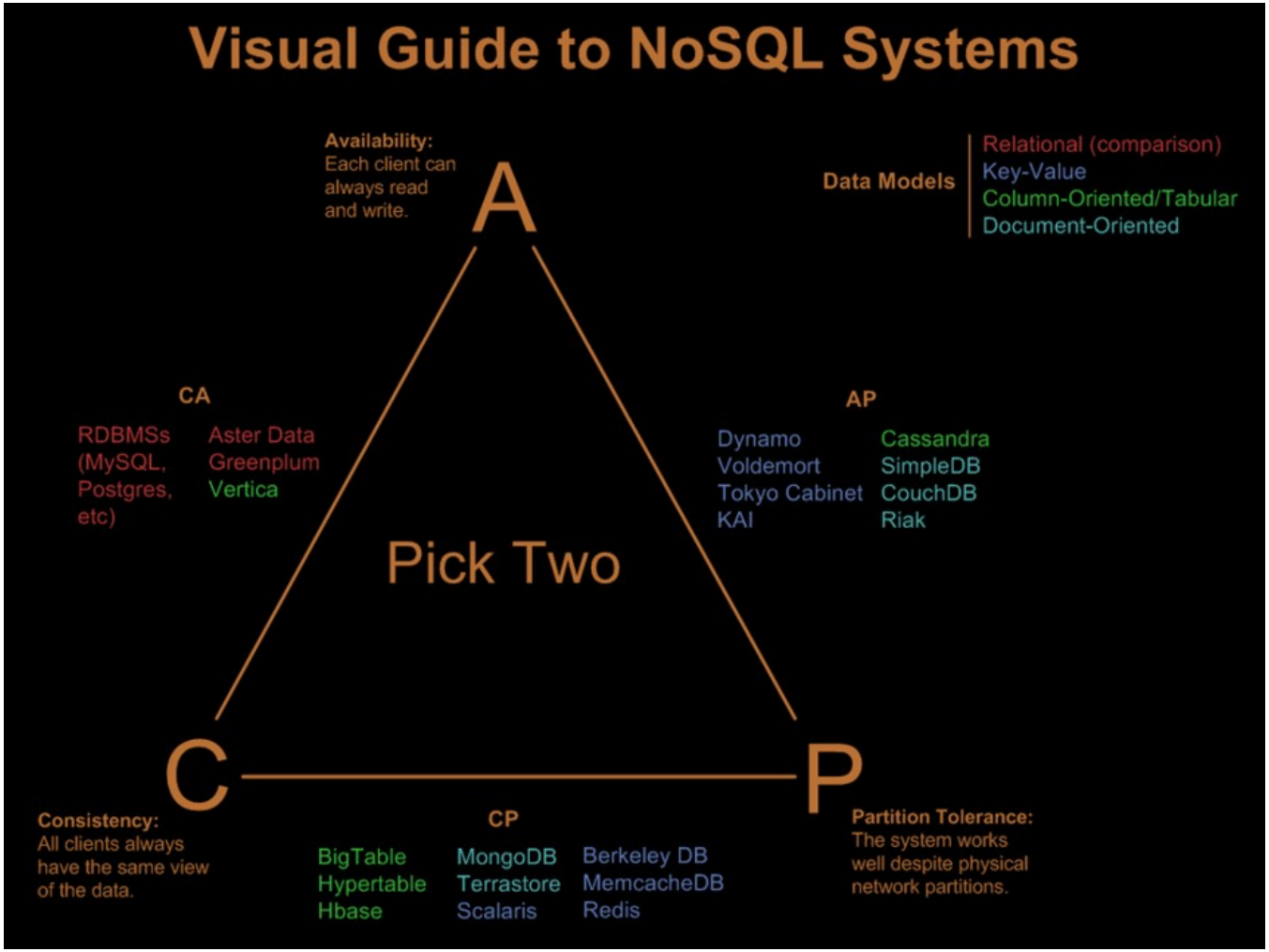
```
{
  "id": 1,
  "autor": "Michael",
  "datum": "2015-04-23T11:19:21.000Z",
  "ueberschrift": "Mein neuer Blog",
  "eintrag": "Hier der Text des Blogs…",
  "markierung": ["Allgemein", "News"],
  "kommentare": [{
    "autor": "klaus1",
    "datum": "2015-04-23T11:23:15.000Z",
    "kommentar": "Super Blog!"
  }, {
    "autor": "tom15",
    "datum": "2015-04-23T11:29:15.000Z",
    "kommentar": "Unbedingt lesen!"
  }],
  "id": 2,
  ...
}
```

Aggregat / Collection 2: »Benutzer«

```
{
  "benutzer": "Michael",
    "name": {
      "vorname": "Michael",
      "nachname": "Schwarze"
    },
    "password": "$5$6&7665!!223/34%4",
  "benutzer": "tom15",
  ...
}
```

NoSQL Datenbanken

- Key/Value Store (Redis)
- Column Store (Cassandra, HBase)
- Document Store (MongoDB, CouchDB)
- Graph Database (neo4j)



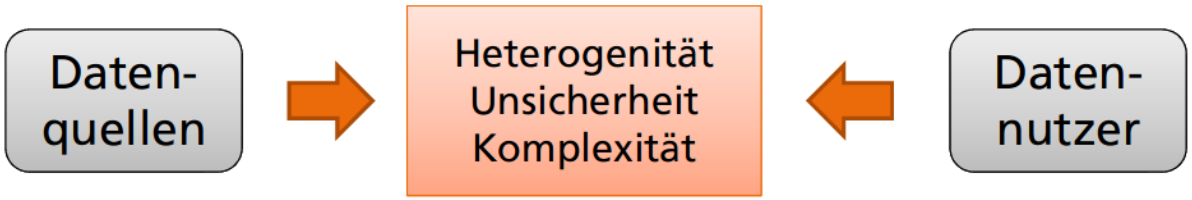
Data Lakes

Die üblichen Probleme eines Big-Data-Projekts

- Welche Datenquellen sind verfügbar?
- Wo sind die Daten, die ich brauche?
- Welche Schnittstellen bietet die Datenquellen?
- Mit welchem API kann ich effizient auf die Daten zugreifen?
- Wie kann ich meine Daten mit anderen Daten verknüpfen?
- Wie kann ich die Daten in meine gewünschte Struktur bringen?
- Wie kann man die Daten kontinuierlich analysieren?

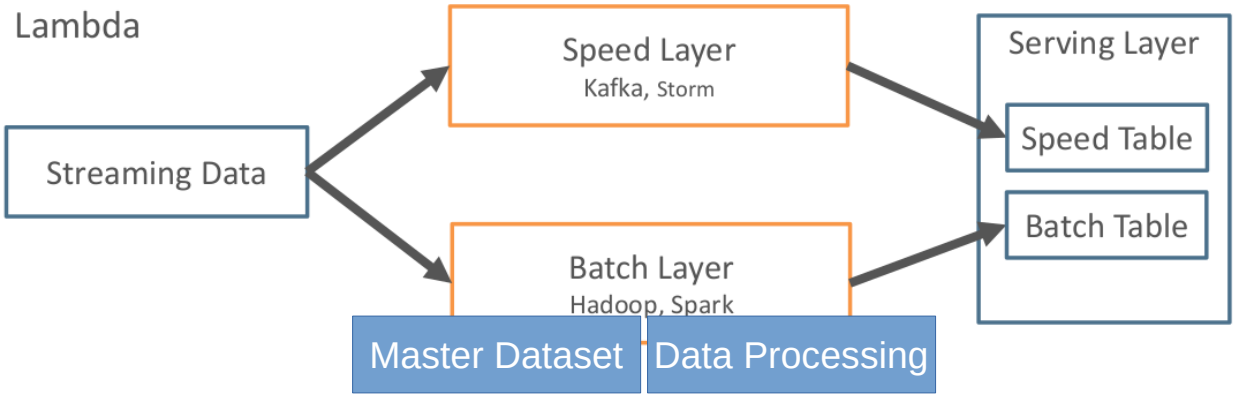
Daten Zugriff & Verfügbarkeit

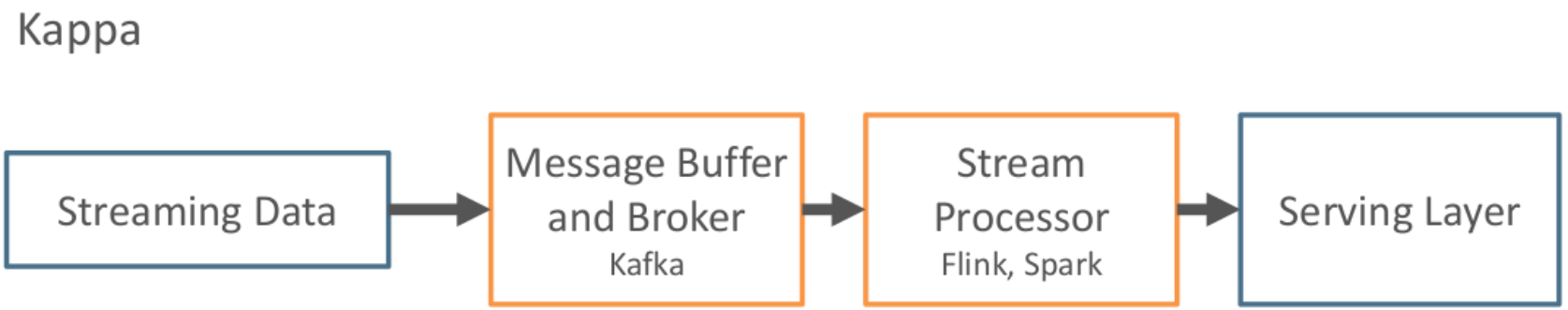
Data Lakes als universeller Datenspeicher



Lambda und Kappa Architektur

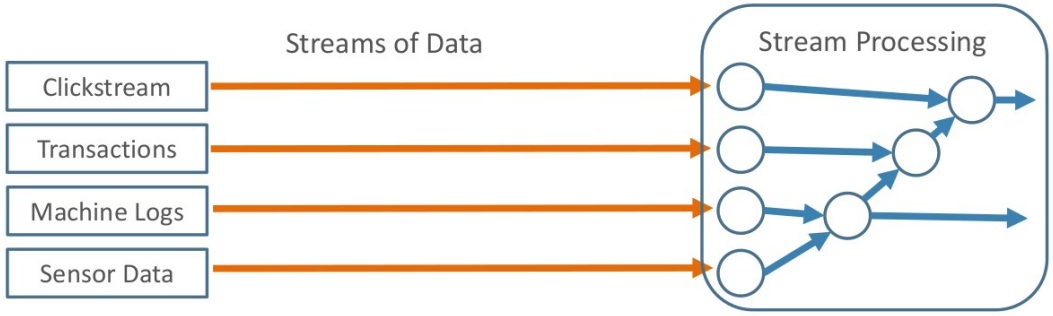
- Ziel: Schelle Datenauswertung





Batch vs. Stream

- Zugriff auf alle Daten
- Split in Batches
- Verarbeitung aller Daten gleichzeitig
- Antwort am Ende
- Complex Analytics (z.B. Model Training)



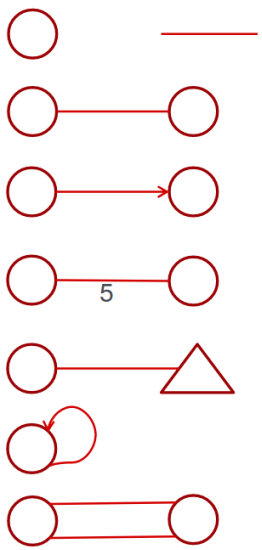
- Verarbeitung eines eingehenden Datenstroms
- Sofortige Antwort
- Die Ergebnisse basieren sich auf aktuellen Daten
- Optimierung der Latenzzeit (z.B. durchschnittliche Zeit für ein Record)
- Die Berechnung muss in Echtzeit abgeschlossen werden
- Berechnet etwas relativ Einfaches (z.B. die Verwendung vordefinierter Modelle zur Labeling eines Datensatzes)

Block2
Netzwerkanalyse

Grundbegriffe der Graphentheorie

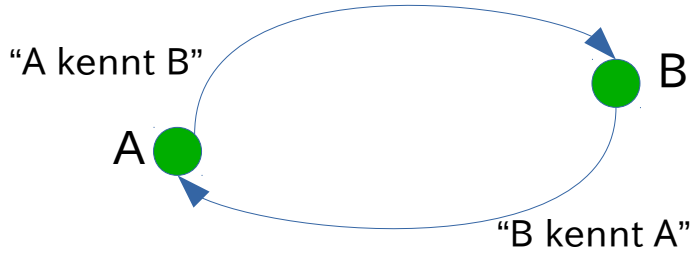
- Netzwerke lassen sich mit Graphen beschreiben
- Mathematisches Modell, bestehend aus Knoten und Kanten
- In einem sozialen Netz entsprechen die Knoten den Individuen einer Gesellschaft, die Kanten repräsentieren ihre Beziehungen

- Knoten und Kanten
- Ungerichtete Graphen
- Gerichtete Graphen (digraph)
- Gewichtete Graphen
- Bipartite Graphen
- Schleife (Loop)
- Mehrfachkanten (multi-edges)

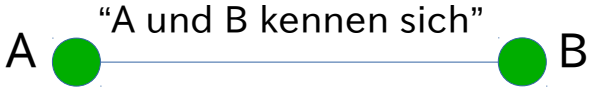


Knotengrade

- Grad eines Knotens (node-degree): Die Anzahl seiner Kanten
- Eingangsgrad (in-degree): Die Anzahl Kanten, die zu einem Knoten hinführen
- Ausgangsgrad (out-degree): Die Anzahl Kanten, die von einem Knoten wegführen



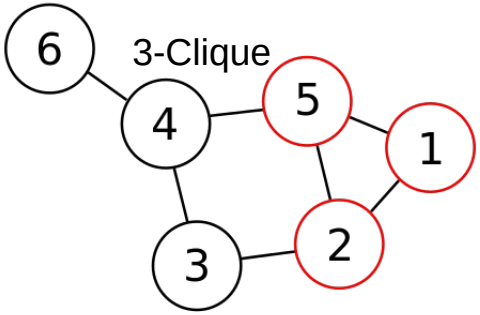
gerichtet



ungerichtet

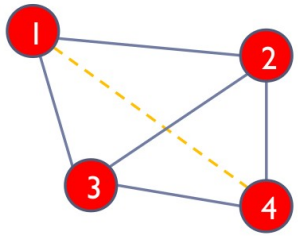
Clique

- Teilmenge von Knoten in einem ungerichteten Graphen, bei der jedes Knotenpaar durch eine Kante verbunden ist.
- Eine Clique ist ein Teilgraph, der vollständig ist.

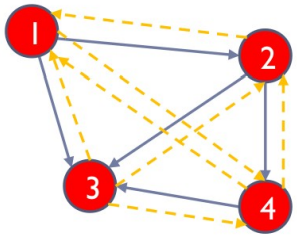


Density

Anzahl vorhandener Kanten / Anzahl möglicher Kanten



density = 5/6 = 0.83

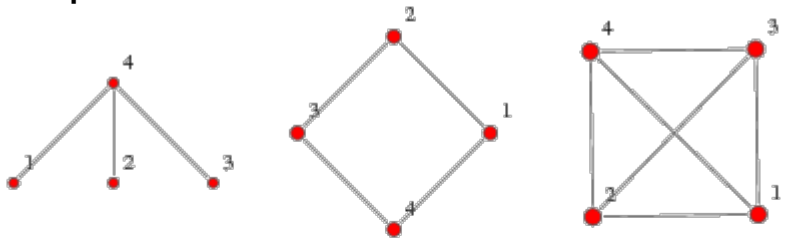


density = 5/12 = 0.42

Ein gerichteter Graph hat die hälfte Dichte seiner ungerichtetes Äquivalent, weil es doppelt soviele mögliche Kanten hat.

Adjazenz-Matrizen

- Darstellung eines Graphs als quadratische Matrix
- Falls eine Kante existiert zwischen mi und mj, dann ist mij = 1
- Anzahl Kanten, resp. Gewicht als Matrix-Element

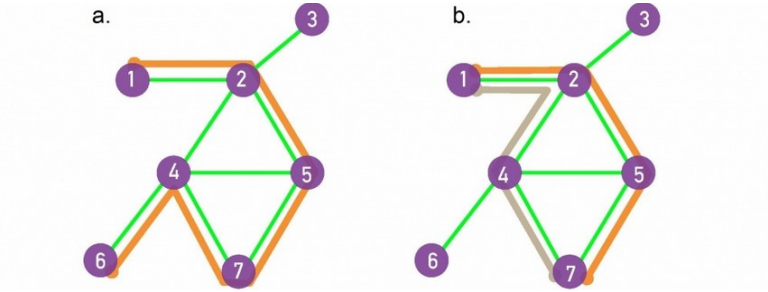


$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

- Die Matrizen-Form ermöglicht die Berechnug vieler Masszahlen, welche Auskunft über die Charakteristik des Graphen geben

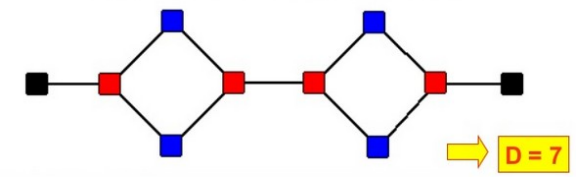
Pfade und Pfadlängen

- Pfad = Verbindung zweier Knoten in einem Graphen (geordnete List von Links)
- Ein kürzester Pfad zwischen zwei Knoten ist ein Pfad mit minimaler Länge
- Pfadlänge: Anzahl der Kanten in einem Pfad



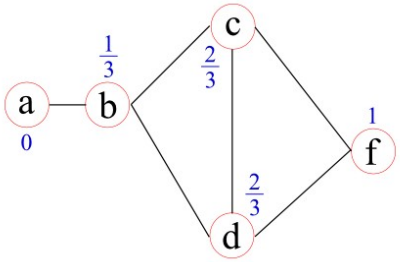
Durchmesser

Der Durchmesser eines Netzwerks ist der längste kürzeste Pfad



Cluster-Coefficient

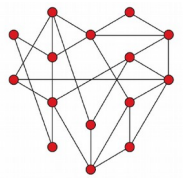
- Lokake Kompaktheit oder Dichte eines Graphen
- Anzahl verbundene Nachbarn /Mögliche Nachbarverbindungen



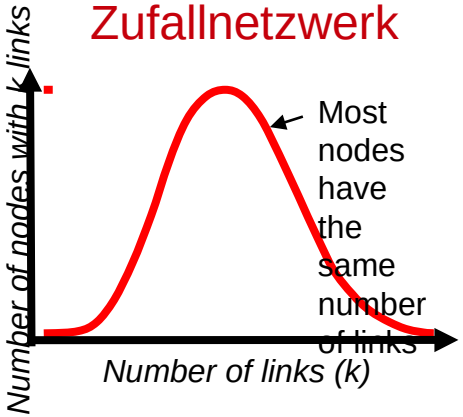
$$C_i = \frac{2L_i}{k_i(k_i-1)}$$

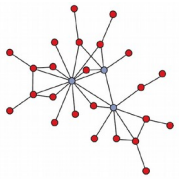
Gradverteilung

P(K) = Prob(Knoten hat genau Grad k)

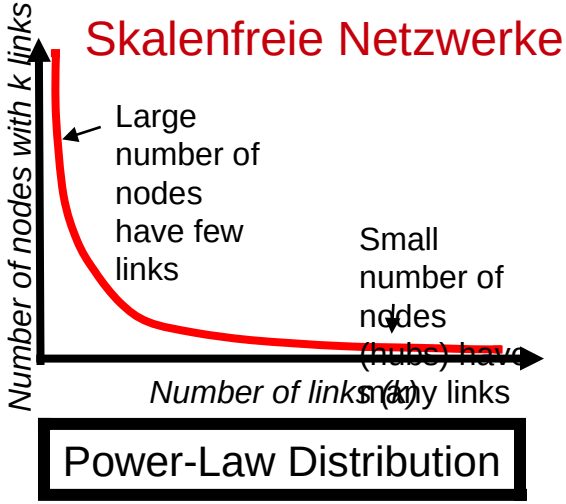


Zufallnetzwerk





Skalenfreie Netzwerke

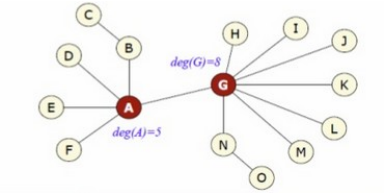


Zentralitätsmassen

- Wie gut die Kommunikation zwischen Knoten ist, lässt dich durch die Pfadlänge beurteilen
- Welche Knoten besonders wichtig für den Zusammenhalt des Netzwerkes oder für den Informationsfluss sind, kann man mit Zentralitätsmassen bestimmen
- Grad-basierte Zentralität (degree centrality)
- Nähe-basierte Zentralität (closeness centrality)
- Zwischenzentralität (betweeness centrality)

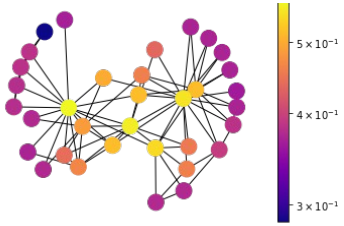
Degree Centrality: “Wer hat viele Freunde?”

Die Anzahl der direkten Beziehungen, die ein Knoten hat



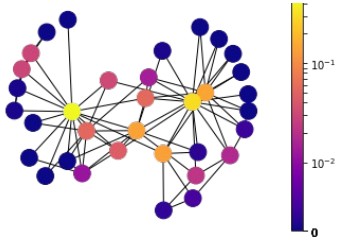
Closeness Centrality: ”Wo sind die Hubs?”

Misst, wie schnell ein Knoten auf andere Knoten zugreifen kann



Betweeness Centrality: “Wo sind die Brücke?”

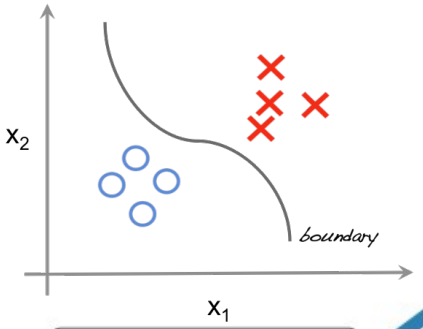
Identifiziert die Position eines Knotens in Bezug auf sein Fähigkeit, Verbindungen zu anderen Gruppen herzustellen



Block3
Überwachtes Lernen

Supervised vs Unsupervised

Supervised learning

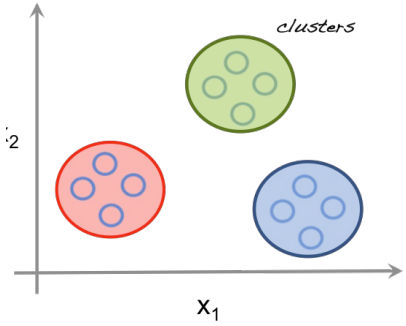


Vorhersage Attribute bekannt (Ziel: Klassen Vorhersage)

Daten haben Labels

- Klassifikation: Die Ausgangsvariable nimmt Klassenbezeichnungen. (Beispiel: ist eine Email Spam oder nicht?)
- Regression: Die Ausgangsvariable nimmt kontinuierliche Werte an. (Beispiel: geeignete Hauspreise kennen)

Unsupervised learning

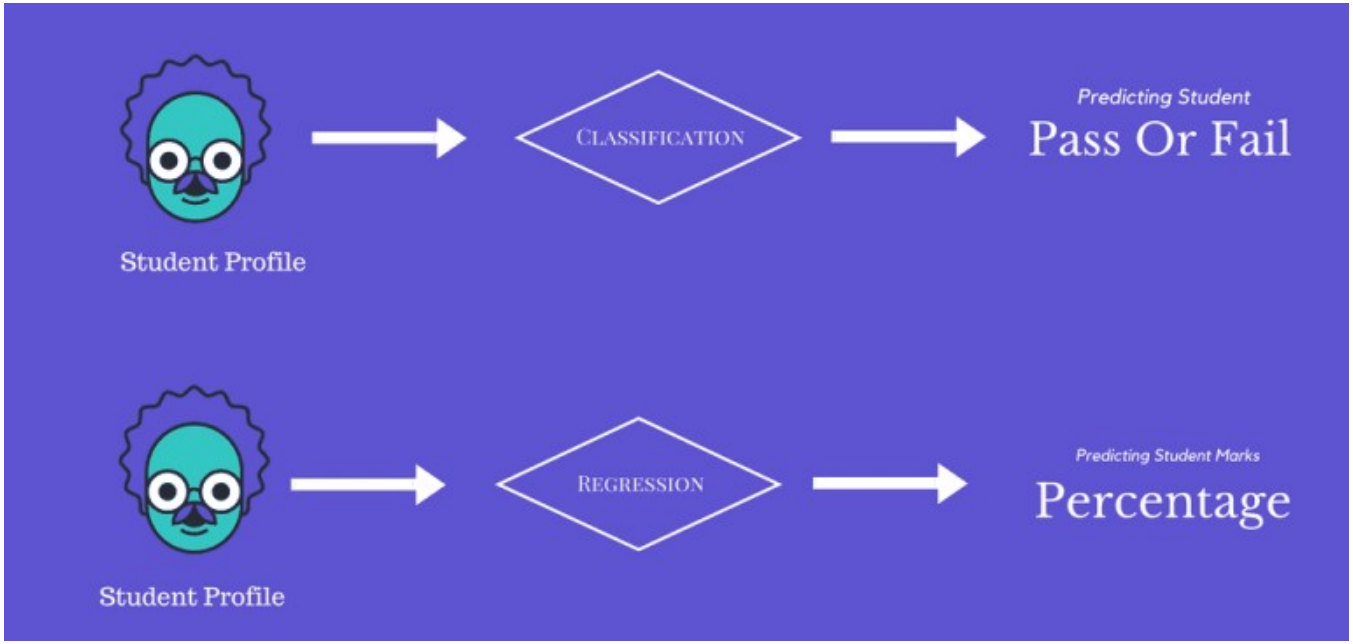


Daten haben keine Labels

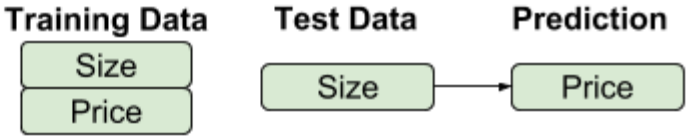
Vorhersage Attribute nicht bekannt (Ziel: Muster finden)

- in den Daten Muster (Clusters) zu finden

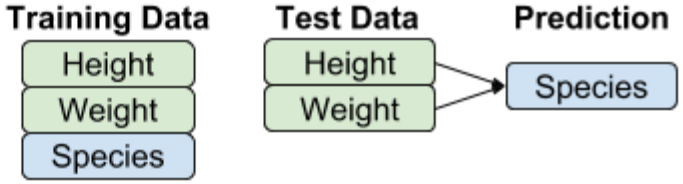
Classification vs Regression



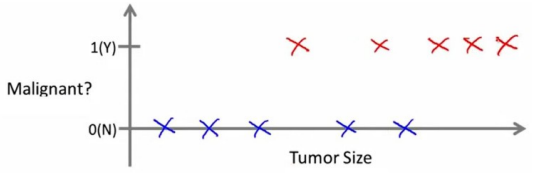
Regression Schema



Classification Schema

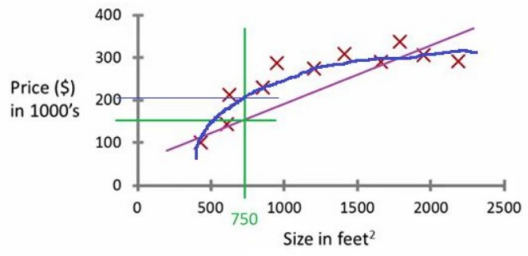


Beispiele

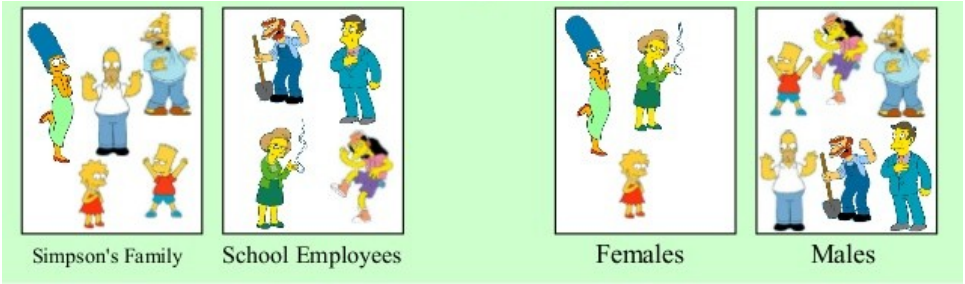


Klassification von Spam

Vorhersage von Kreditwürdigkeit

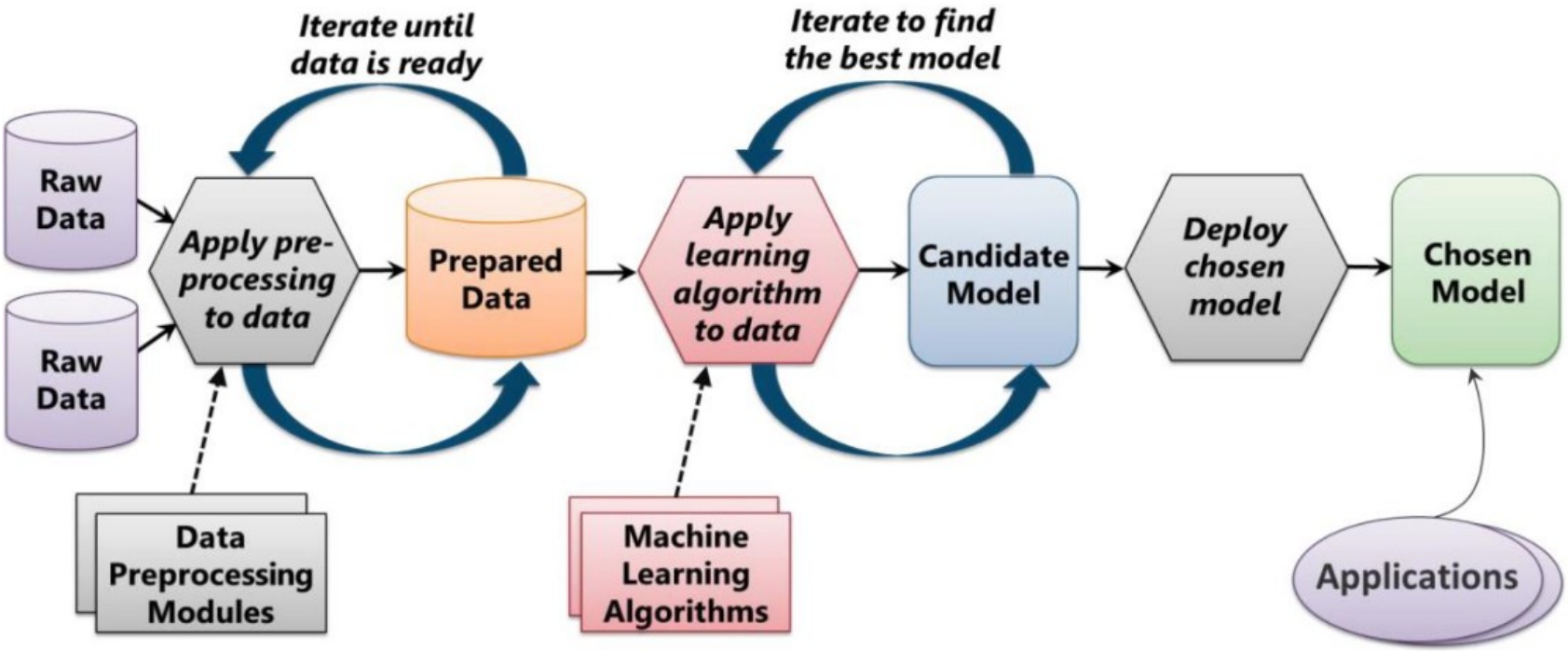


Vorhersage Kundenverhalten →
Zeit bis Kündigung
Hauspreis bestimmen



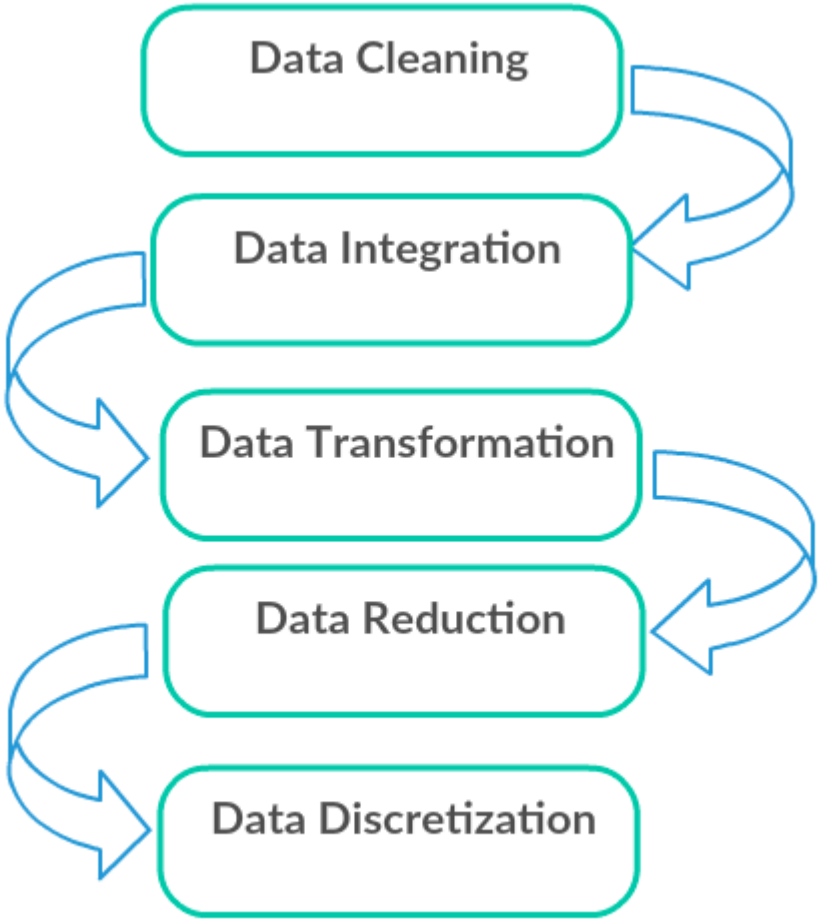
Computer Cluster
Social Network Analyse
Markt Segmentierung
Astronomische Daten

Machine Learning Vorgehen

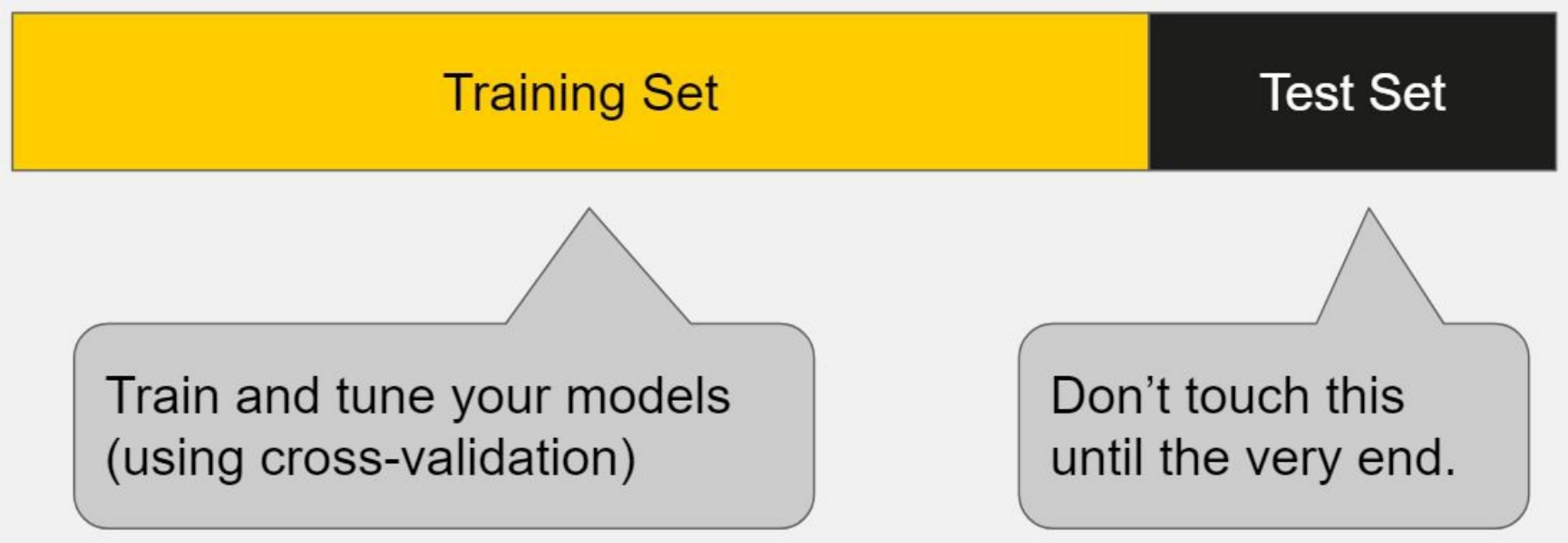


- Quelle: Chappell: Introduction to Machine Learning, 2015" p.5

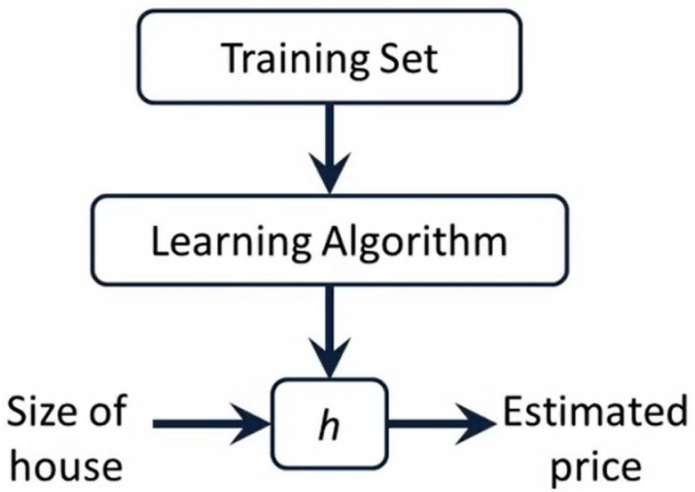
Data Preprocessing



Training Set und Test Set



Kostenfunktion



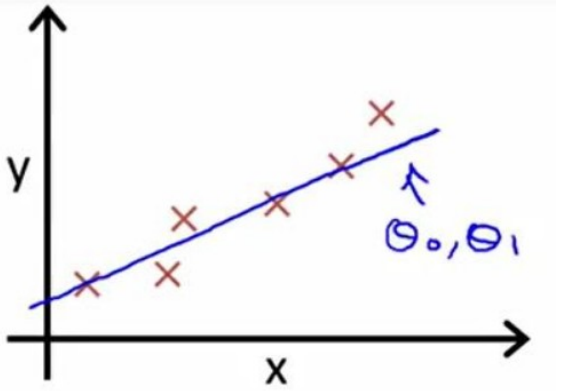
- Problem: minimierung einer Zielfunktion (Kostenfunktion)
- Die Kostenfunktion beschreibt die Abweichung des gewählten Modells von den vorliegenden Daten
- Aufgabe der Optimierungsverfahren ist es, den Fehler iterativ zu minimieren.

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

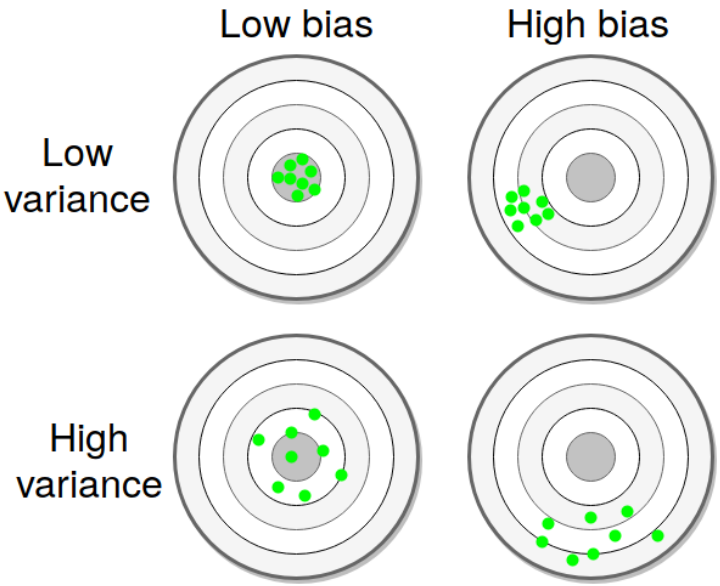
Goal: $\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$



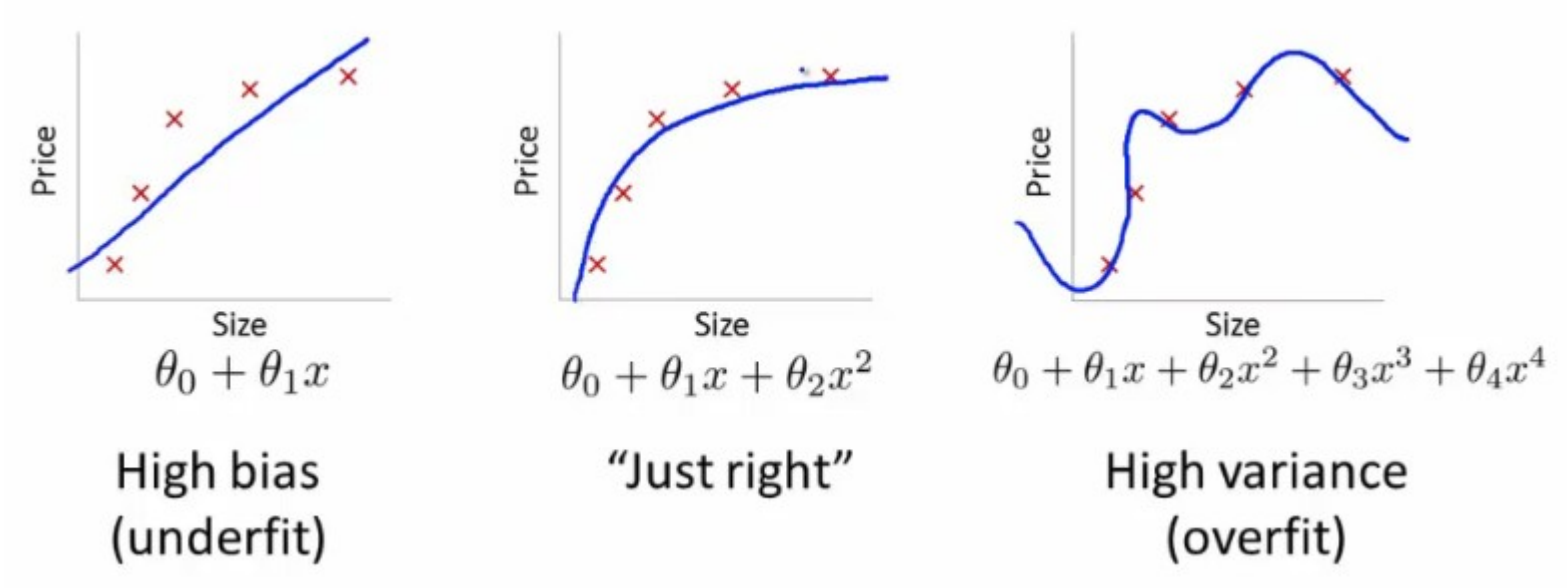
Bias-Variance Dilemma

- Bias: Fehler von falschen Annahmen im Modell. Matematisch ist der Erwartungswert, mit dem sich das Modell von der “Realität” unterscheidet.

- Variance: Misst, wie sich die Vorhersagen über verschiedene Realisierungen vom Modell (aus verschiedenen Trainings sets) voneinander unterscheiden. Im Gegenstatz zu Bias misst nicht, ob wir überhaupt richtig sind, in einer Situation mit hoher Varianz sind die Vorhersagen aus verschiedene Realisierungen von Modell sehr ausgebreitet.

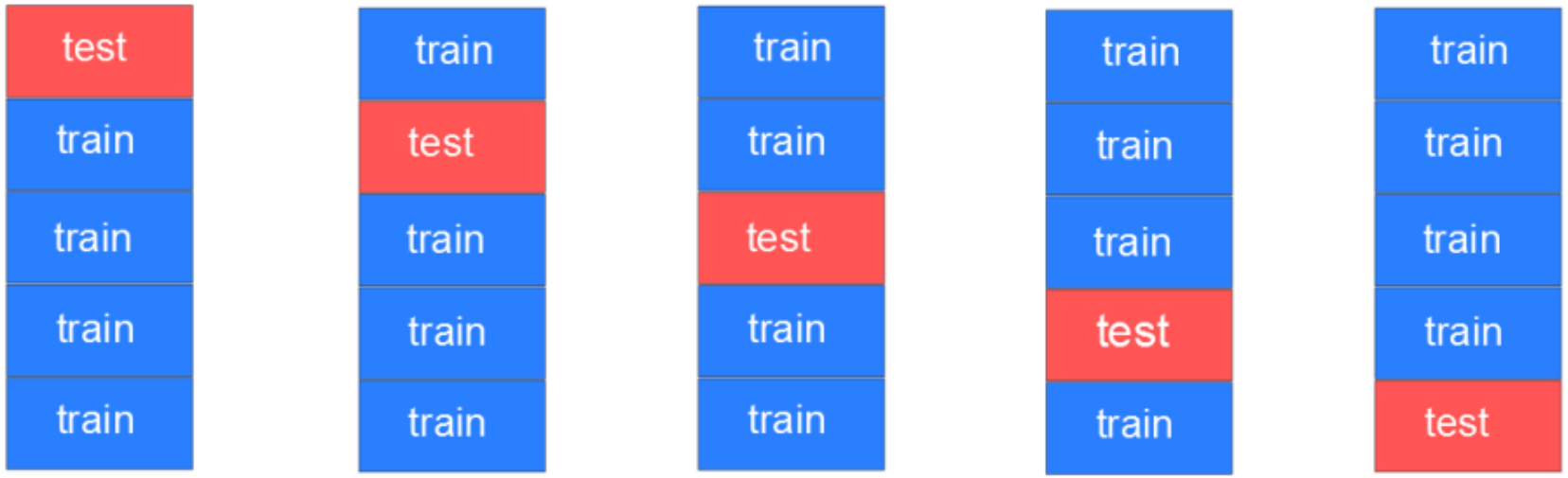


Bias-Variance Tradeoff

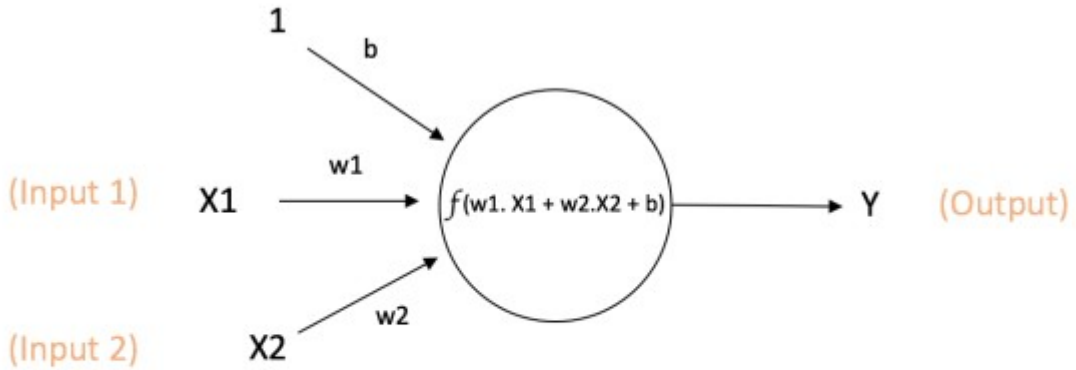


- Wie wählen wir zwischen 2 Modelle, eine mit hoher Varianz und ein mit hohem Bias?
- Der Kompromiss ist die Verwendung von Kreuzvalidierung, bei der dem Training Set in k gleiche Untermenge aufgeteilt wird. Ein Teil der Datensatz wird als Test Set benutzt. Das Modell wird auf die k Untermenge Traininiert und die Resultate werden gemittelt, um einen Generalisierungsfehler zu erzeugen.

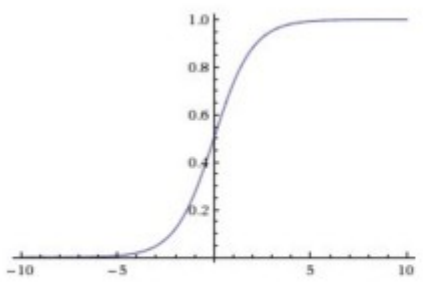
Kreuzvalidierung



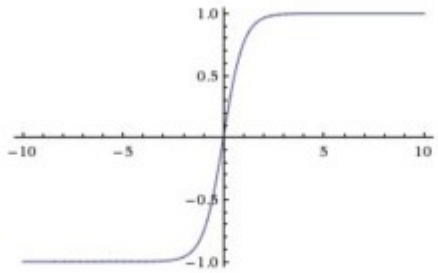
Neural Networks



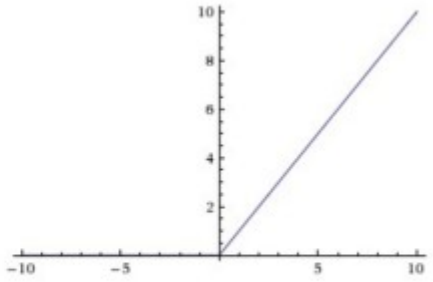
Output of neuron = Y= $f(w1.\ X1 + w2.X2 + b)$



Sigmoid

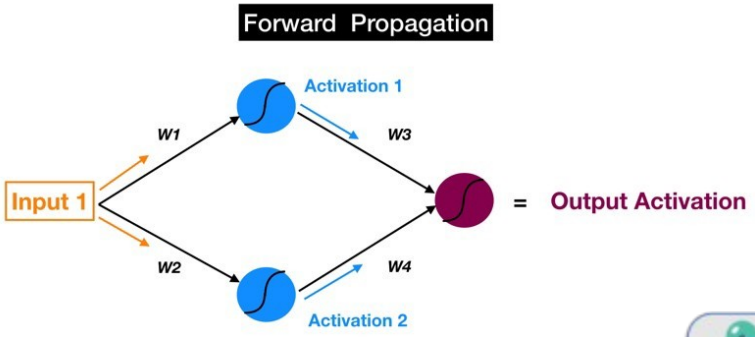


tanh



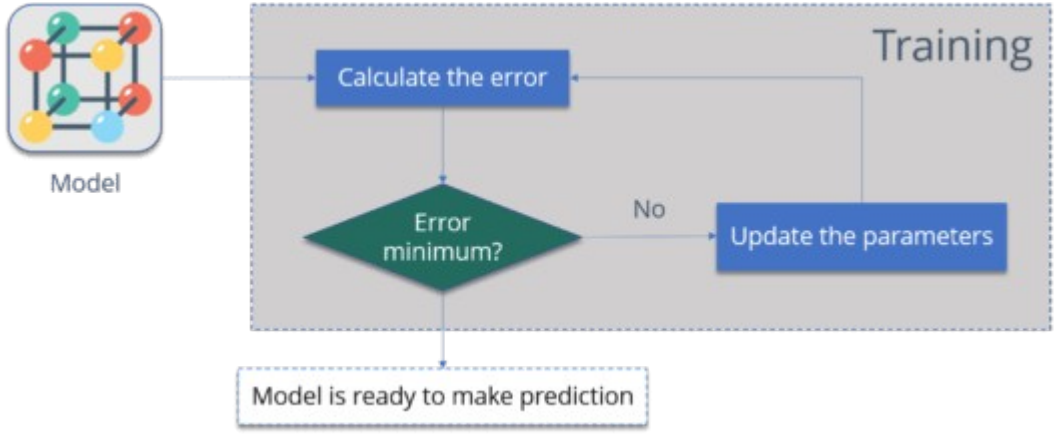
ReLU

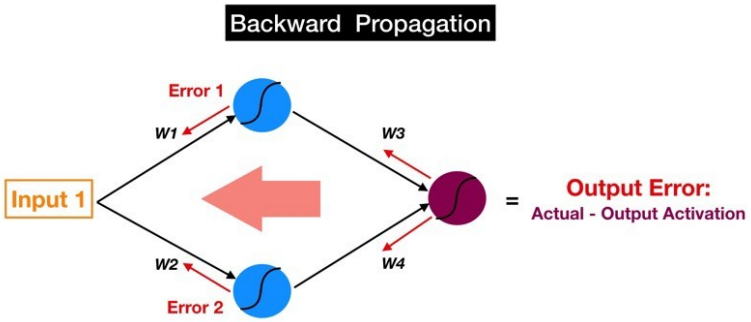
Feed Forward Neural Network



Move Signal forward

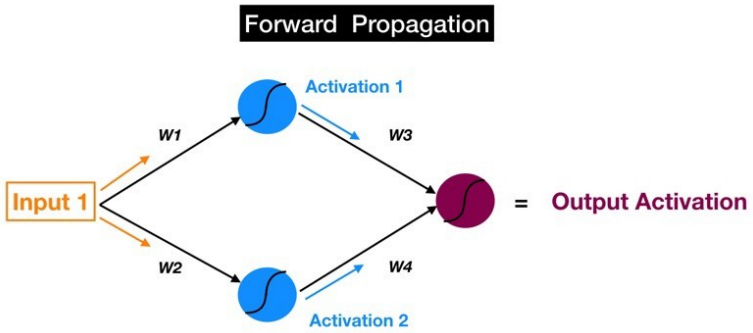
Backpropagation





Move Error Backwards

Backpropagation



Move Signal forward