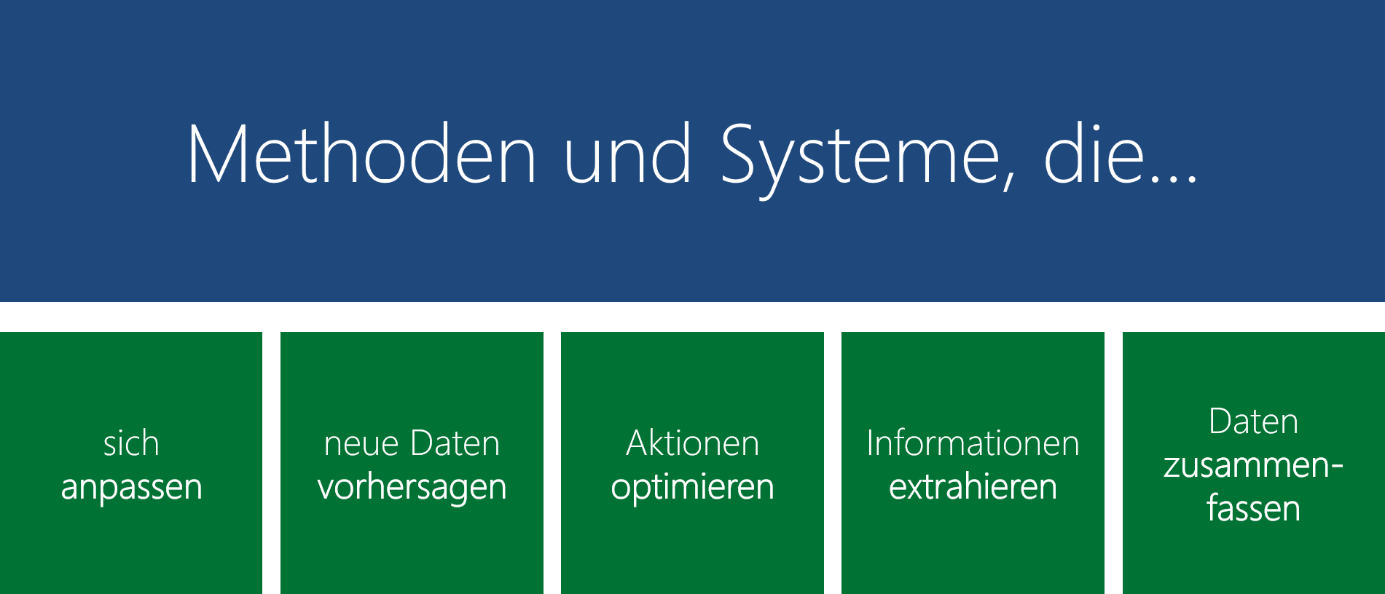
## Was ist Machine learning (ML) im Allgemeinen? Wovon hängt ein ML Algorithmus am stärksten ab und wodurch kann dieser daher verbessert werden? Nennen Sie Problemstellungen, die durch ML gelöst werden können.

Ist ein Konstrukt aus Algorithmen die aus einem Datenbestand lernen und daraus intelligente Aktionen generieren können. 

Am stärksten hängt Machine Learning von den Daten ab, je mehr aussagekräftigere Daten desto besser das Modell.

Problemstellungen: Erkennung von Spam Mails (Spamfilter), Spracherkennungssysteme, Erkennung von Kreditkartenbetrug

## Erkläre die MAPE loop. Was kann dadurch vermieden werden? Was passiert in den einzelnen Schritten?

---

## Was ist ein Peer-To-Peer Netzwerk? Welcher Peer hat besonderen Stellenwert in einen P2P Netzwerk und wer übernimmt die Organisation? Welche Vorteile hat P2P gegenüber Client-Server Systeme?

Ist eine spezielle Art eines Rechennetzwerkes bei welchen beliebig viele Rechner deren **Ressourcen bündeln** (Bandbreite, Rechenleistung, Speicher, …) und alle Rechner gleich berechtigt sind. Alle **Peers** sind in diesem System **gleichberechtigt**. Das zentrale Interesse eines P2P Netzwerkes liegt im Teilen von Ressourcen. Einfache P2P Netzwerke **organisieren sich selbst**, das heißt es gibt keinen zentralen Server. Von P2P Netzwerken gibt es 2 Typen. **Unstructured P2P** (keine besondere Struktur vorhanden) und **Structured P2P** (eine Struktur in der Kommunikation wird erzwungen). Vorteile von P2P Systemen gegenüber Client-Server Systemen sind: es gibt keinen single Point of failure, P2P Netzwerke sind in der Regel relativ Robust.

## Nennen Sie je einen Vorteil, Nachteil von P2P System. Erklären Sie diese konkret an einem Anwendungsfall und stellen Sie einen Vergleich zu Client-Server Architektur.

**Vorteil:** geringfügige Kosten 🡪 Rechner kann sofort ins Netzwerk integriert werden. Es fallen höchstens Kosten für Kabeln und Instandhaltung an. Im Gegensatz zu Client-Server Systemen die bei entsprechender Größe sehr teuren Server benötigen sind die Kosten für ein P2P Netzwerk relativ gering. Je nach Größe des Netzwerkes können die Kosten variieren.

**Nachteil:** kaum Sicherheit und schwer administrierbar 🡪 Da jeder Rechner im System gleich berechtigt ist und es keine zentrale Anlaufstelle gibt können keine komplexen Berechtigungs-Hierarchien aufgebaut werden. Bei Client-Server Systemen werden alle Zugriffe vom Server verwaltet. Diese Zugriffe können von ihm geregelt werden.

**Anwendungsfall:** File sharing, Voice-over-IP /Video-over-IP (Skype 🡪 hybrid P2P), Multiplayer games

## Was ist Ziel einer Distributed Hash Table? Vergleichen Sie die Anzahl der Zugriffe und 1 Vor-+ 1 Nachteil mit P2P Systemen ohne DHT.

Ein Schlüssel wird auf einen Wert gemappt. Ziel ist es Struktur zur Kommunikation einzelner Nodes in einem P2P Netzwerk einzuführen. Ein System ohne Distributed Hash Table ist bei der Suche nach einer Datei oder einem Node sehr ineffizient. Das System wird hier gefloodet, d.h. jeder Rechner schickt seinen unmittelbaren Nachbarn eine Anfrage ob er der gesuchte Rechner ist bzw. die gesuchte Datei besitzt.

**Vorteil P2P ohne DHT:** Kein Aufwand für Verwaltung der Informationen (🡪 Keine Routing- bzw. Fingertable vorhanden)

**Nachteil P2P ohne DHT:** Bei Suchen nach Rechnern oder Dateien in großen Netzen sehr inneffizient.

## Wie funktioniert DHT (Chord) und welche Funktionen gibt es?

Peers sind in Chord Ringförmig angeordnet. Jeder Peer kennt seinen direkten Nachfolger als auchseinen direkten Vorgänger

**Ein Peer speichert folgende Informationen:** Vorgänger, Nachfolger, FingertableDie Navigation erfolgt über einen Index in einem virtuellen Netzwerk 🡪 Nicht über die IP  
Damit die Navigation schneller vonstattengeht als ohne (Laufzeitverhalten *O(n)*), benötigt man die Fingertabelle, dazu speichert jeder Knoten weitere Nachfolger in dieser Tabelle.



**Funktionen von Chord:**

***Successor:*** Funktion lautet *find\_successor(k)* und gibt den nächsten direkten Nachfolger vom

Knoten *k* an.

***Hash Funktion:*** Um aus einen Key einen identifier zu erzeugen.

***Insert:*** Einfügen eines Key-Value-pair auf einem Knoten.

***Lookup:*** Anfragender Knoten sendet seine IP und seine Key an seinen Nachfolger. Diese Nachricht wird solange weitergereicht bis die Ressource gefunden ist. Der Knoten welche die Ressource hat schickt diese dann direkt zum Anfragenden Knoten.

***Join the swarm:*** Ein neuer Knoten wird ins Netzwerk eingefügt.

## Wie funktioniert der Insert in DHT? Skizzieren Sie die Schritte mit einem m = 2 Ring

Es wird ein Key-Value Pair (torrent, myIP) an einem index im Ring eingefügt.

1. Key = *hash(torrent)* 🡪 Erzeugt den Index
2. Frage den *successor(key) von*
3. Finde den zuständigen Knoten der für den key der zuständig ist für das Speichern: *successor(key)*

So wird der Index zufällig im Netzwerk verteilt. Der Key wird immer auf dem Knoten gespeichert der dem Key am nächsten ist.



## Wie funktioniert ein Join in DHT und welche Möglichkeiten gibt es als Node eine DHT wieder zu verlassen?

**Join:**

1. der einzufügenden Knoten hasht seine IP-Addresse und bekommt als Ergebnis einen Key 🡪 *Key = hash(torrent)*
2. Der neue eingefügte Knoten fragt seinen Nachfolger

🡪 *successor(key)*

wer ist mein Vorgänger.

1. Der eingefügte Knoten speichert sich danach diese Information.
2. Der successor Knoten wird nun benutzt um die Fingertable des eingefügten Knoten upzudaten.

In Chord können Knoten eine DHT entweder *Graceful* (gibt seine Keys an seinen Successor weiter) oder via *Crashing* (Knoten ist einfach weg) verlassen.

## Wie funktioniert Destination Sequenced Distance Vector? Welche Daten müssen pro Node gespeichert werden?

Ist ein Routing Algorithmus basierend auf den Bellman-Ford algorithmus. Es wird versucht den kürzesten Weg in einem Graphen (Netzwerk) zu finden. Jeder Knoten hat eine Tabelle mit dem besten und günstigsten Weg zu seinen Nachbar Knoten. Diese Tabellen werden upgedatet durch Informationsaustausch und Abgleich mit Nachbarknoten. Jeder Knoten ist so in der Lage die beste Verbindung zu jedem Ziel zu berechnen.

Pro Node müssen folgende Infos gespeichert werden: *next hop, number of hops* and *sequence number*

## Wie funktioniert Link State Routing? Warum sind berechnete Routen in LSR immer optimal?

Durch Link-State Routing wird der kürzeste Weg in einem Netzwerk von A nach B errechnet. In Link-State Routing werden 5 folgende Schritten pro Knoten angewandt:

1. Erforsche Nachbarn und lerne Ihre Adressen
2. Erstelle Distanz oder Kosten-Metrik zu jedem Nachbar Knoten
3. Erstelle ein Paket mit allen gerade gelernten Informationen.
4. Sende das Packet an alle Nachbarn des Knotens und empfange auch deren Informationen.
5. Berechne den kürzesten Pfad zu jedem anderen Knoten.

Berechnete Routen sind immer optimal weil der Dijkstra Algorithmus bei der Berechnung des kürzesten Pfades eingesetzt wird. Durch diesen wird immer der kürzeste Pfad errechnet.

## Wie funktioniert das optimierte Link-State-Routing? Was wird in OLSR optimiert und wodurch? Skizzieren Sie ein kleines Beispiel indem ein OLSR sinnvoll verbessert werden kann (im Gegensatz zu LSR)

Bei diesem verteilten flexiblen Routingverfahren ist allen Knoten (Routern) die vollständige Netztopologie bekannt, sodass sie von Fall zu Fall den kürzesten Weg zum Ziel festlegen können. Als proaktives Routingprotokoll hält es die dafür benötigten Informationen jederzeit bereit.

Die Topologie-Entdeckung erfolgt bei OLSR über zwei Arten von Nachrichten: HELLO- und Topology-Control (TC)-Nachrichten. HELLO-Nachrichten dienen zum Link Sensing, zur Nachbarentdeckung und zur Mitteilung der Multipoint-Relay-Wahl. Die TC-Nachrichten dienen dazu, die so gewonnenen Informationen über mögliche Verbindungen im Netz zu verteilen.

Ein im Netz teilnehmendes Gerät (Knoten) entdeckt seine 1-Hop- und 2-Hop-Nachbarn über die periodisch verschickten HELLO-Nachrichten. Diese enthalten die IP-Adressen der bereits bekannten 1-Hop-Nachbarn sowie den Status der Verbindung zu ihnen und werden nicht weitergeleitet. Aus seinen 1-Hop-Nachbarn wählt jeder Knoten Multipoint Relays (MPRs), sodass er über sie jeden seiner 2-Hop-Nachbarn erreichen kann. Die MPRs sind die Knoten, die Broadcast-Nachrichten weiterleiten, was das Fluten effizienter macht. Sie sind es auch, die die TC-Nachrichten erstellen, die eine Liste mindestens der Knoten enthalten, von denen sie als MPRs gewählt wurden, sodass für jeden Knoten mindestens eine Möglichkeit bekannt ist, wie er erreicht werden kann. Diese TC-Nachrichten werden im gesamten Netzwerk verteilt. Auf diese Weise erhält jeder Knoten eine Vorstellung des Netzwerkes und kann Routingtabellen erstellen.

**Optimierung:** Das OLSR-Protokoll optimiert das Routing in dem es instabile Funkverbindungen ausspart und protokollbedingte Netzbelastungen durch Umgehung von Multipoint-Relays reduziert, was einer Minimierung der Hops entspricht. Jeder OLSR-Knoten ist durch seine IP-Adresse gekennzeichnet.

Im OLSR-Protokoll wird der Dijkstra-Algorithmus eingesetzt.

## Wie funktioniert ad-hoc on-demand distance vector? Was unterscheiden AODV grundsätzlich zu DSDV?

Der Ad-hoc On-demand Distance Vector-Routingalgorithmus (AODV) ist ein Algorithmus zum Weiterleiten von Daten durch ein mobiles Ad-hoc-Netz. Das Protokoll gehört zu den Topologie basierten, **reaktiven Routingverfahren**. Das heißt Routen zu bestimmten Zielen werden erst bei Bedarf ermittelt (hop bei hop).

Im AODV-Protokoll **verwaltet jeder Netzwerkknoten eine Routingtabelle**, die Schleifenfreiheit der Routen wird durch eine inkrementierende Sequenznummer erzeugt. In der Routingtabelle sind zusätzlich zur IP-Adresse des nächsten Knotens in Richtung Ziel (Next Hop), die Sequenznummer des Zielknotens, die Distanz zum Zielknoten (Anzahl Hops) und verschiedene Status-Flags gespeichert.

Eine AODV-Operation benötigt verschiedene Netzwerknachrichten, um Daten durchs Netzwerk zu verteilen. Der Algorithmus definiert dazu verschiedene Nachrichtentypen: **Route Requests** (RREQ), **Route Replies** (RREP) und **Route Errors** (RERR).

AODV ist ein reaktives Routing-Protokoll: Solange gültige Routen zwischen zwei Endpunkten einer Kommunikationsverbindung existieren, ist AODV inaktiv. Erst wenn eine neue Route zu einem (neuen) Ziel erstellt werden muss wird AODV aktiv: Der Ursprungsknoten sendet eine Anfragenachricht (RREQ) per Broadcast. Eine **Route** wird bestimmt, wenn der **RREQ entweder das Ziel erreicht oder einen Zwischenknoten mit einer Route findet, die "aktuell genug“ ist**. Eine Route ist "aktuell genug“, wenn der Routeneintrag gültig ist und die eingetragene Zielsequenznummer mindestens so groß ist wie die Zielsequenznummer in der Anfragenachricht. Diese Route wird per **Antwortnachricht (RREP) Unicast zurück an den Ursprung der RREQ** geschickt. Dazu hat jeder Knoten, der die Anfrage empfangen und weitergeleitet hat, den Knoten gespeichert, von dem er die RREQ-Nachricht erhalten hat.

Knoten beobachten den Verbindungsstatus der Nachbarknoten in aktiven Routen. Wenn eine Unterbrechung der Verbindung festgestellt wird, wird eine Fehlernachricht (RERR) zur Benachrichtigung des Linkbruchs gesendet. Diese Nachricht gibt die Knoten an, die nicht mehr länger durch die kaputte Verbindung erreichbar sind. Dieser Benachrichtigungsmechanismus wird durch Listen von „aktiven Nachbarn“ (Knoten, die innerhalb einer Zeitspanne ein Paket zur Weiterleitung an das Ziel empfangen haben) in den Routeneinträgen ermöglicht.

## Was ist Dynamic Source Routing? Wodurch unterscheidet es sich von anderen besprochenen Mechanismen? Erklären Sie das Prinzip von RREQ und RREP.

Dynamic Source Routing (DSR) ist ein Routing Protokoll für Wireless mesh network. Es ist ähnlich dem AODV-Protokoll, da die Routen erst dann gesucht werden, wenn ein Computer einen konkreten Verbindungswunsch hat (**reaktives Routen**).

Eine bemerkenswerte Optimierung von DSR ist, dass die Computer, welche die Nutzdaten weiterleiten, keine Routingtabellen haben müssen. Stattdessen wird **eine Liste aller Zieladressen** in **jedes Paket** gepackt. Das verringert die Notwendigkeit der weiterleitenden Rechner, immer aktuelle Routingtabellen zu haben. Dies verringert wiederum die Übertragung von Routingdaten wesentlich und die weiterleitenden Computer können einfacher aufgebaut sein (geringere Hardwareanforderung) und müssen auch keine großen Speicher für die Routingtabellen besitzen.

Teilnehmer belauschen den lokalen Netzwerkverkehr, um weitere Routinginformationen zu bekommen. Dies ist möglich, da in jeder übertragenen Nachricht eine Liste mit Adressen anderer Knoten steckt. Darüber hinaus erkennen sie Routenanfragen, Routenfehler und Informationen anderer Rechner. Diese können sie später selbst verwenden.

DSR hat ein ähnliches Protokoll zum Auffinden von Routen wie AODV. In kleinen, weniger ausgelasteten Funknetzwerken haben die beiden Protokolle ein ähnliches Performance-Verhalten. Ist die Auslastung dagegen höher, verursacht DSR nur etwa 1/3 des Datenaufkommens. Die DSR-Knoten bekommen sehr viele Informationen durch das Belauschen des Netzwerkverkehrs und müssen so das Netz mit weniger eigenen Routenwünschen belasten.

Auf der anderen Seite verursacht das Belauschen das Problem, dass viele Informationen gesammelt werden und ältere – nicht mehr aktuelle – Informationen erkannt und aus den Routingtabellen entfernt werden müssen.

**Dynamic Source Routing vs. AODV:**

* **Route discovery** 
  + Suche nach einer Route zu einem gewünschten Ziel;
  + Speichere festgestellte Routen in einen eigenen Zwischenspeicher)
* **Route maintenance**
  + Überprüfe ob die Verbindungen noch immer aufrecht sind und aktiv
  + Besondere Fehlermeldungen um Einträge im Zwischenspeicher zu bereinigen
* **Route caching**
  + Optimiert Routenfindung in Zwischenknoten
  + Knoten erlernen neue Routen von Route Request (RREQ) und Route Replies (RREP)
  + Knoten können auch von Zwischenrouten einer Route lernen

## Nennen Sie 2 Strategien für Kommunikation und vergleichen Sie diese. Nennen Sie je ein Beispiel für eine Anwendung und nennen Sie ein Beispiel, wo Sie die Kommunikation nicht einsetzen würden. Begründen Sie Ihre Antwort.

* **Message-Oriented Transient Communication** (Beispiel: **FTP (TCP), Video streaming (UDP)**)
* **Remote Procedure Call** (Beispiel **Network File System NFS**)

Message-Oriented Transient Communication:

Remote Procedure Call:

Ist ein Protokoll für die Ebenen 5 und 6 und gewährleistet einen entfernten Prozeduraufruf. Jeder Server im Netz stellt im Rahmen dieses Konzeptes eine Anzahl von Diensten zur Verfügung, die mit RPC angefordert werden können. Diese Funktionen sind als Prozeduren eines Programms realisiert und können unter Angabe von Serveradresse, Programmnummer und Prozedurnummer angesprochen werden. Der Server, der RPC-Dienste unterstützt, bekommt Anfragen von Clients, auf denen die Anwendung lokal verbreitet wird. Erhält der Server eine RPC-Anfrage vom Client, führt er eine lokale Prozedur aus. Nach Beendigung der Prozedur überträgt er die Ergebnisse zum Client und wartet auf die nächsten Anfragen. Um Sicherheitsaspekte zu gewährleisten, muss sich der Client beim Server ausweisen. Diese Berechtigung erstreckt sich auch auf die Nutzung anderer Dateien.   
Remote Procedure Call (RPC) ist anwendungsorientiert und setzt auf dem UDP- oder dem TCP-Protokoll auf.

## 32. Was versteht man unter k-fold cross validation? Was versteht man unter data splitting?

## Wozu gibt es AOP im Allgemeinen? Was wird dadurch gelöst, was in „herkömmlichen“ OOP nur mit sehr viel Aufwand verbunden wäre?

Aspekt orientierte Programmierung (AOP) ist ein Programmierparadigma für die objektorientierte Programmierung, um generische Funktionalitäten über mehrere Klassen hinweg zu verwenden (**Cross-Cutting Concern**). Beispiele für solche Cross-Cutting Concerns sind: Security, logging, caching, etc.

Mit AOP wird die Trennung von **Core-Level-Concerns** (Businesslogic 🡪 Anforderungen an ein Programm 🡪 funktionalen Anforderungen) und **System-Level-Concerns** (betreffen das gesamte System/ technische Randbedingungen 🡪 Logging, Security, etc) ermöglicht, welches durch die herkömmliche OOP nur mit viel Aufwand gelöst werden kann.

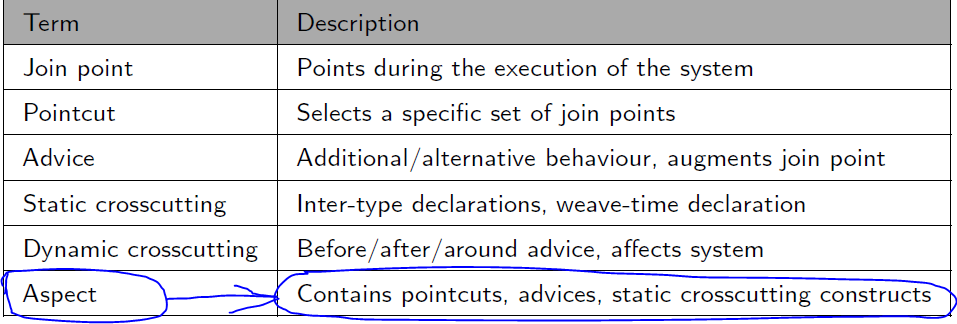
Die beiden Teile Core-Level-Concerns und System-Level-Concerns sind miteinander „verwoben“. Die Core-Level-Concerns kann man als Komponenten bezeichnen und die System-Level-Concerns sind die Aspekte. Core-Level-Concerns werden überlicherweise als Module oder Objekte implementiert. Für die Aspekte gab es vor der aspektorientierten Programmierung keine elegante Lösung.

Das Problem der miteinander verwobenen Anforderungen wird auch als Cross-Cutting Concerns bezeichnet, denn sie „schneiden“ quer durch alle logischen Schichten des Systems. AOP ist das Werkzeug, um die logisch unabhängigen Belange auch physisch voneinander zu trennen. Dabei wird angestrebt, Code zu erzeugen, der besser wartbar und wiederverwendbar ist.

## Wozu dient ein Aspekt und aus welchen Bestandteilen setzt er sich zusammen?

Ein Aspekt eines Programmes ist als zum Beispiel eine Funktionalität wie Logging, die über mehrere Module/Teile eines Programmes hinweg verwendet wird aber eigentlich nicht zur primären Funktionalität eines Programmes zählt. Dadurch wird das „separation of concerns“-Konzept der OOP verletzt, welche die Kapselung von nicht zusammenhängenden Funktionalitäten verfolgt.

Das heißt mittels eines Aspekts werden System-Level-Concerns von den Core-Level-Concerns gekapselt.



Ein Aspekt beinhaltet also „pointcuts“ (Abfrage die überprüft/feststellt ob ein definierter/gegebener „join point“ übereinstimmt), „advices“ (zusätzliches Verhalten das zu einem definierten „join point“ hineingewoben werden soll) und „static crosscutting constructs“ (statische Struktur von Klassen, Interfaces und Aspekte verändern)