Angelo
Bartłomiej
Inez
Kacper
Kamil
Kasia
Łukasz
~~**Pszemek**~~
Piotr

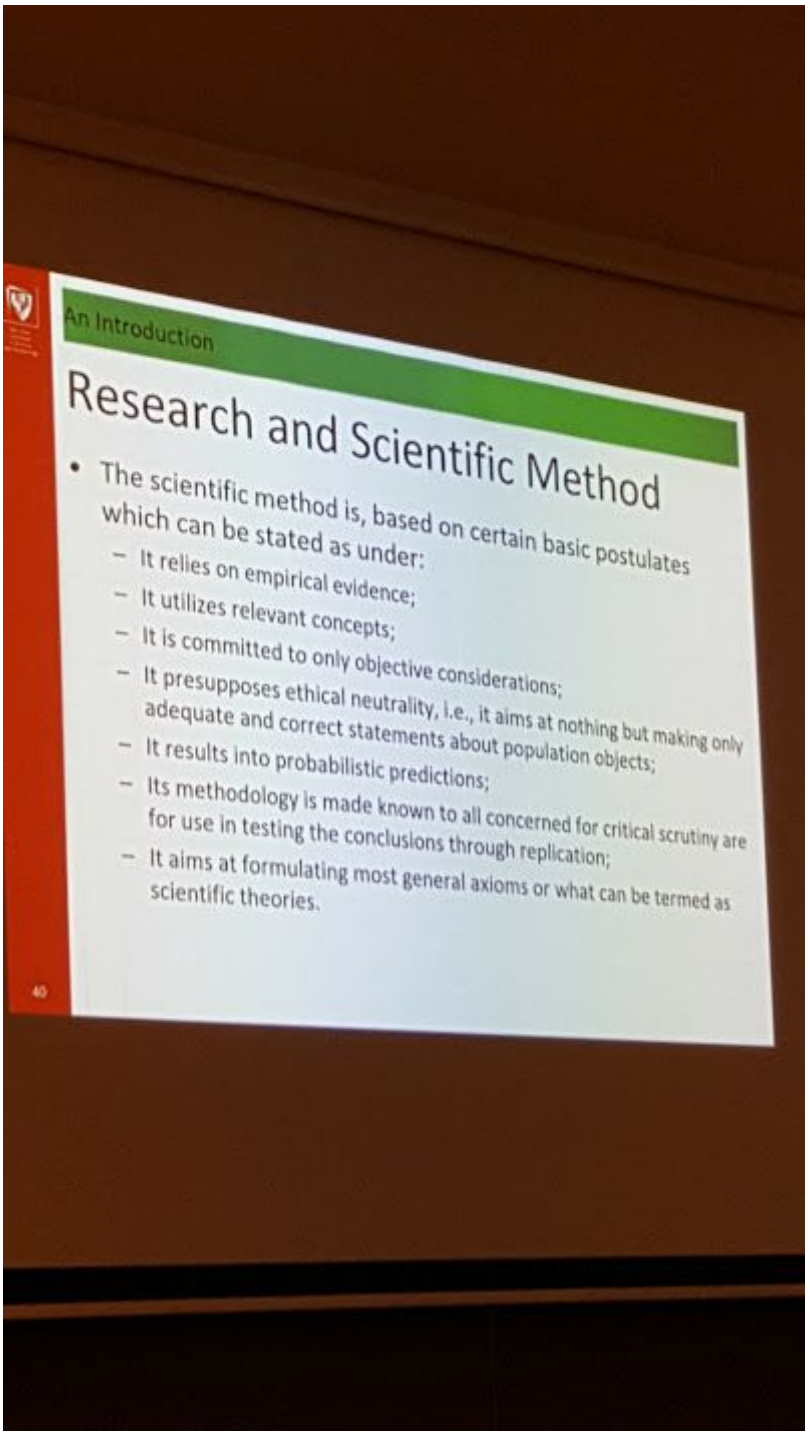
32 /40

Lista pytań: (żeby widzieć postęp, zaznaczajmy ✔ pytania, które są już zrobione)

Pytanie	Autor Opracowania	Status
1. Postulates of research methodology.	**Kamil**	✔
2. Modern methods used in research methodology.	**Kamil**	✔
3. Modeling and meta-modeling.	**Łukasz**	✔
4. Properties and scope of using UML.	**Bartłomiej**	✔
5. Problems with models transformation and consistency.	**Inez**	✔
6. Model-driven and quality-driven software development.	**Kasia**	✔
7. Use-cases, statecharts, sequence and activity diagrams.	**Bartłomiej**	✔
8. Software life cycle, different approaches.	**Inez**	✔
9. MDA approach to software development.	**Kasia**	✔
10. Basis of requirements engineering.	**Inez**	✔
11. Patterns (architectural, design, program).	**Bartłomiej**	✔
12. The effectiveness of information systems.	**Inez**	✔
13. Modeling of complex operation systems.	**help?**	
14. The concept of decision-making system and computerized decision support system.	***Kacper***	✔
15. Modeling, identification, and aiding of decision making process.	**Angelo**	✔
16. Basic problems, methods and algorithms of discrete optimization.	**Kasia**	✔
17. Basic methods of "soft computing?.	**Angelo**	✔

18. Rules for specification of the relational database model.	**Inez**	✔
19. Rules for mapping class diagrams onto relational models.	**Bartłomiej**	✔
20. The SQL 2003 standard.	**Piotr**	✔
21. Evolutionary Computation.	**Kasia**	✔
22. Introduction to machine learning, deduction versus induction.	**Inez**	✔
23. Artificial neural networks.	**Inez**	✔
24. Architecture of distributed and parallel systems, methods of parallel and distributed processing.	**Kasia**	✔
25. Grids and clusters. Exploitation and development problems.	**Bartłomiej**	✔
26. Static and dynamic interconnection networks, typical topologies, different routing strategies.	***Kacper***	
27. Automatic program parallelisation, dependencies in sequential programs, identification of parallelism,	**Bartłomiej**	✔
28. Evaluations of parallel systems: performance metrics, scalability of parallel systems, Amdhal, Gustafson and other laws.	**Kasia**	✔
29. Rule-based knowledge representations.	**Angelo**	✔
30. Knowledge based systems - inference mechanisms.	**Angelo**	✔
31. Incompleteness, inconsistency and uncertainty of knowledge.	**Angelo**	✔
32. Topologies of Computer Network.	**Inez**	✔
33. Internet and Web services Architecture. Web and P2P systems.	***Kacper***	
34. Measurement, estimation and prediction of communication time in the Internet.	**Angelo**	**+-**
35. The Web Server model. Access and scheduling algorithms for HTTP requests in a Web Server.	***Kacper***	
36. Differences between IPv4 and Ipv6.	**Kasia**	✔
37. Multimedia technologies used in information systems.	**Łukasz**	
38. Processing and access to multimedia data.	**Łukasz**	
39. Designing of multimedia interface of computer applications.	**Pszemek**	
40. Methods, techniques and tools used for designing and construction of mobile systems.	**Łukasz**	

1. Postulates of research methodology.



Research approach can be divided into three types: deductive, inductive, abductive. The relevance of hypotheses to the study is the main distinctive point between deductive and inductive approaches. Deductive approach tests the validity of assumptions (or theories/hypotheses) in hand, whereas inductive approach contributes to the emergence of new theories and generalizations. Abductive research, on the other hand, starts with ‘surprising facts’ or ‘puzzles’ and the research process is devoted their explanation. The following table illustrates the major differences between deductive, inductive and abductive research approaches in terms of logic, generalizability, use of data and theory.

Tabelka tutaj: http://research-methodology.net/research-methodology/research-approach/

	Deduction	Induction	Abduction
Logic	In a deductive inference, when the premises are true, the conclusion must also be true	In an inductive inference, known premises are used to generate untested conclusions	In an abductive inference, known premises are used to generate testable conclusions
Generalizability	Generalising from the general to the specific	Generalising from the specific to the general	Generalising from the interactions between the specific and the general
Use of data	Data collection is used to evaluate propositions or hypotheses related to an existing theory	Data collection is used to explore a phenomenon, identify themes and patterns and create a conceptual framework	Data collection is used to explore a phenomenon, identify themes and patterns, locate these in a conceptual framework and test this through subsequent data collection and so forth
Theory	Theory falsification or verification	Theory generation and building	Theory generation or modification; incorporating existing theory where appropriate, to build new theory or modify existing theory

2. Modern methods used in research methodology.

Your research will dictate the kinds of research methodologies you use to underpin your work and methods you use in order to collect data. If you wish to collect quantitative data you are probably measuring variables and verifying existing theories or hypotheses or questioning them. Data is often used to generate new hypotheses based on the results of data collected about different variables. One's colleagues are often much happier about the ability to verify quantitative data as many people feel safe only with numbers and statistics.
However, often collections of statistics and number crunching are not the answer to understanding meanings, beliefs and experience, which are better understood through qualitative data. And quantitative data, it must be remembered, are also collected in accordance with certain research vehicles and underlying research questions. Even the production of numbers is guided by the kinds of questions asked of the subjects, so is essentially subjective, although it appears less so than qualitative research data.

Więcej wiedzy:
https://he.palgrave.com/studentstudyskills/page/choosing-appropriate-research-methodologies/

3. Modeling and meta-modeling.

Modeling refers to using models – physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process. To express it is used modeling language.
Types of languages:

-Graphical eg. UML
-Textual eg. the Eiffel tower <is located in> Paris. Paris <is classified as a> city
-Algebraic eg. AML
-Behavioral - describe the observable behavior of complex systems
-Discipline-Specific - is focused on deliverables affiliated with a specific software development life cycle stage
-Domain-specific - is a software engineering methodology for designing and developing systems
-Framework-specific - for an object-oriented application framework
-Object -oriented - for designing object oriented soft
-Virtual reality - before 1995 known as the Virtual Reality Markup Language is a standard file format for representing 3-dimensional (3D) interactive vector graphics
-Others:
*Architecture Description Language
*Face Modeling Language
*Generative Modelling Language
*Java Modeling Language
*Promela
*Rebeca Modeling Language
*Service Modeling Language
*Web Services Modeling Language
*X3D

Metamodeling is a process of generating model of model(metamodel). Thus **metamodeling** or **meta-modeling** is the analysis, construction and development of the frames, rules, constraints, models and theories applicable and useful for modeling a predefined class of problems.

4. Properties and scope of using UML.

Properties:

- offers a way to visualize a system's architectural blueprints of: activities, components, system run, entities interaction, user interface
- compatible with the leading object-oriented software development
- diagram is a partial graphic representation of a system's model
- 2 different views of a system model:
    - static - class diagram, component diagram, deployment diagram, object diagram, package diagram
    - behavioral - activity diagram, sequence diagram, state diagram, use case diagram

Scope:

- software: applications :), enterprise systems, distributed web services, telecommunication, bla, bla…
- non-software: workflow in the legal systems, medical electronics, design of hardware

5. Problems with models transformation and consistency.

def. **Model transformation**, in model-driven engineering, is an automated way of modifying and creating models. An example use of model transformation is ensuring that a family of models is consistent, in a precise sense which the software engineer can define. The aim of using a model transformation is to save effort and reduce errors by automating the building and modification of models where possible.

Intro:

In model-driven software engineering, the central artefact in the development are models, rather than programs coded in a programming language. Beside the generally accepted usefulness of models as a means for communication, documentation, and requirements capture, the main objective of this idea is the abstraction from programming language.

It is generally acknowledged that maintenance and evolution of software are still among the main challenges of software engineering. Refactoring techniques help to overcome these problems at the code-level by defining (and supporting) software transformations that restructure a software system while preserving its behavior, which is validated by a number of test cases. If model-driven software engineering shall be successful, similar solutions are required at the model level.

Problem:

Proposals for refactoring of UML models [15] suffer from the problem that no generally accepted operational interpretation of UML models is available. Therefore, validation of UML refactorings through testing is difficult. At the same time, transformations of models are difficult to describe because, unlike in programming languages where techniques based on string and tree rewriting are suitable, UML models have a graphical representation. In such situations rule-based graph transformations have been used to specify data base schema evolution [8, 7], refactoring and evolution of object-oriented programs [10], or software architecture reconfiguration [16]

One reason for the first problem, beside the poor quality of language specifications and tools, is the fact that models describe a system from different, partly overlapping perspectives in order to reduce the complexity of the problem. This separation of concerns causes the need to ensure the consistency of different submodels or views in order to ensure that there exists an implementation satisfying all requirements expressed by these different submodels.

Proposed solution:

Recently, we have proposed a methodology for defining and checking semantic consistency requirements for UML models [3]. Rather than completely formalizing UML models [9], our approach consists of a partial formalization of those aspects of a model that lead to a consistency problem and can be summarized as follows: A potential consistency problem arises as a conceptual overlap between submodels dealing with common aspects of the system to be described. To formalize and analyze this potential consistency problem, a semantic domain is chosen which supports the relevant aspects, and a mapping of (these aspects of) the models into the semantic domain is defined. Within the semantic domain, consistency conditions are formulated to enable the formal verification of the consistency of individual models, e.g., by means of a model checker. A rule based notation has been proposed to describe flexible and extensible mappings of models to various semantic domains.

Źródło: *Consistency-Preserving Model Evolution through Transformations*, https://link.springer.com/content/pdf/10.1007%2F3-540-45800-X.pdf p. 212-227

6. Model-driven and quality-driven software development.

Model-Driven Software Development (MDSD or MDD) puts analysis and design models on par with code. Better integration of such models and code should significantly increase the opportunity to effect change through the models, rather than simply modifying the code directly. MDSD encompasses many different techniques:

- model-driven requirements engineering, approach using sub-models for all requirements/handling
- code generation from models, eg. auto generating source code for GUI from template in editor etc
- model-driven testing, tests based on UML models

MDSD therefore aims to find domain-specific abstractions and make them accessible through formal modeling. This procedure allows automation of software production. Moreover, both the quality and maintainability of software systems increase. Models can also be understood by domain experts. To successfully apply the 'domain-specific model' concept, three requirements must be met:

- Domain-specific languages are required to allow the actual formulating of models.
- Languages that can express the necessary model-to-code transformations are needed.
- Compilers, generators or transformers are required that can run the transformations to generate code executable on available platforms.

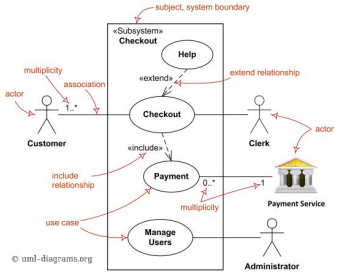
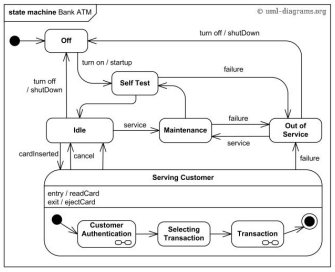
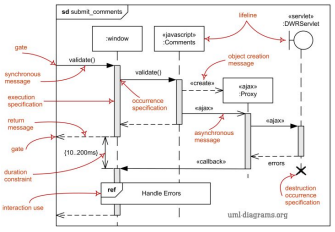
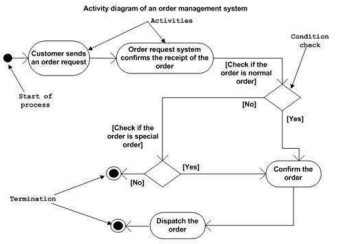
In the context of MDSD, models are graphical or textual. Typically, these models are translated into programming language source code to enable their subsequent compilation and execution.

Introducing Quality-Driven Development

Everything is made in high quality through the development lifecycle.

- Automated tests created by QA (quality assurance) run in the development environment.
- We fix issues on the fly. If a tester finds a problem, she can explain it to a programmer and get it fixed. That means if a tester finds a bug in a story, the story is not done. There is no need to file a ticket and get it assigned to someone later; the story is not done. Just fix it.
- We run manual test charters only once.
- No story is marked *done* until the automation itself can demo the feature, driving the browser and showing the product owner that the intended use case can actually happen.

7. Use-cases, statecharts, sequence and activity diagrams.

Use Case diagram	Statecharts (state diagram)	Sequence diagram	Activity diagram
Used to specify: - required usages of a system - the functionality offered by a subject - how environment should interact with the subject	Used to give an abstract description of the behavior of a system. This behavior is analyzed and represented as a series of events that can occur in one or more possible states.	Describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines.	Models the activity flow of the system. Captures application components / subsystems and describes the flow from one to another. For example between an application, database, external queues, or any other system.
(follow the link to enlarge) 	(follow the link to enlarge) 	(follow the link to enlarge) 	(follow the link to enlarge) 
Elements: - actor - use case - subject - include and extend relationships	Elements: - state (simple/composite) - pseudostate (initial , terminate, entry point, exit point, choice, fork, join, junction) - transitions - 3 parts: trigger [guard] / expression example: click(coordinates) [coordinates in window] / link.follow()	Elements: - lifeline, - execution specification, - message, - combined fragment, - interaction use, - state invariant, - continuation, - destruction occurrence	Elements: - Activities (akcja do zrobienia) - Associations / Edges (połączenia, mogą zawierać “guard’y” {zawarte w “[“ i “]”} określające warunek) - Partition - box określający komponent odpowiedzialny za daną czynność
More info: http://www.uml-diagrams.org/use-case-diagrams.html	More info: http://www.uml-diagrams.org/state-machine-diagrams.html	More info: http://www.uml-diagrams.org/sequence-diagrams.html	More info: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm - kwintesencja http://www.uml-diagrams.org/activity-diagrams.html - detale

Jeszcze krótki komentarz na temat różnicy pomiędzy State Diagram i Activity Diagram, bo na pierwszy rzut oka wyglądają podobnie, ale to tylko pozory. W obu główną rolę pełnią zaokrąglone kwadraty, jednak reprezentują inne rzeczy - stany i aktywności. Mamy też nody start i end.
W przypadku State Diagram interesują nas stany jakie posiada aplikacja - na diagramie chcemy je wszystkie zaprezentować i pokazać jakie zdarzenia (events) prowadzą do zmiany stanu. Przykład z palca (w zasadzie spod xD) - mamy klawiaturę - Stan1 = CAPS LOCK OFF, Stan 2 = CAPS LOCK ON, eventem trigerunącym przejścia między stanami jest wciśnięcie klawisza.

W przypadku Activity Diagram interesują nas poszczególne aktywności które są wykonywane w jakimś procesie i ich możliwy flow pomiędzy komponentami systemu. Np. żeby złożyć zamówienie user musi dodać produkty do koszyka w sklepie, kliknąć w “zamów”, dane zapisywane są w bazie danych, system powiadomień wysyła maila… itd.

8. Software life cycle, different approaches.

Spoko artykuł pod linkiem, nie ma sensu kopiować:

https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/

9. MDA approach to software development.

MDA (Model-driven architecture) is a software design approach for the development of software systems.
The model-driven architecture approach defines system functionality using a platform-independent model (PIM) using an appropriate domain-specific language (DSL).

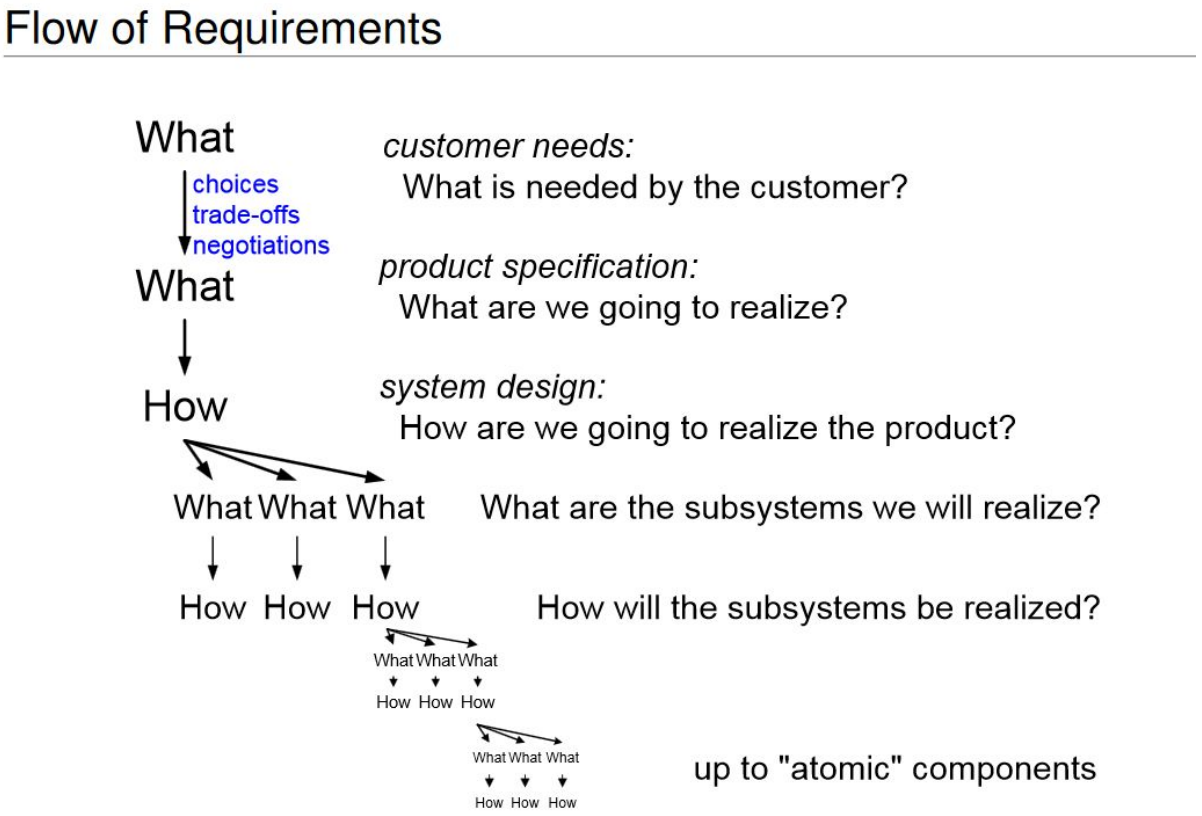
- PIMs are transformed into platform-specific model by using special model transformation language/tool.
- DSL is a language that servers specific role like PostScript, SQL, HTML.

Model-driven focuses architecture on forward engineering, i.e. producing code from abstract, human-elaborated modelling diagrams. One of the main aims of the MDA is to separate design from architecture. The idea is that PIM, which represents a conceptual design realizing the functional requirements, will survive changes in realization technologies and software architectures.
Idealistic: MDA is conceived as a forward engineering approach in which models that incorporate Action Language programming are transformed into implementation artifacts (e.g. executable code, database schema) in one direction via a fully or partially automated "generation" step. This aligns with OMG's vision that MDA should allow modelling of a problem domain's full complexity in UML (and related standards) with subsequent transformation to a complete (executable) application. This approach does, however, imply that changes to implementation artifacts (e.g. database schema tuning) are not supported . This constitutes a problem in situations where such post-transformation "adapting" of implementation artifacts is seen to be necessary.

10. Basis of requirements engineering.

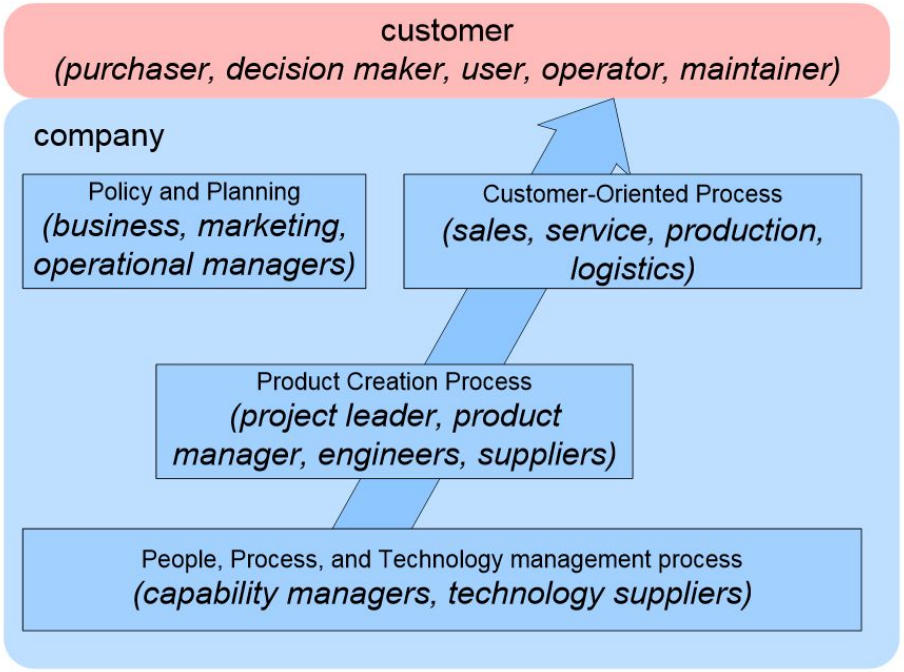
Easy:

Requirement definition:
1) Requirements describing the needs of the customer: *Customer Needs*
2) Requirements describing the characteristics of the final resulting product: *Product Specification*
   - the requirements management process recursively applies this definition for every level of decomposition
3) Requirements describing the needs of the company itself over the lifecycle: *Life Cycle Needs*



Requirements have to be:
- specific
- unambiguous
- verifiable
- quantifiable
- measurable
- complete
- traceable

Also, ‘to enable human use’, requirements shall be:
- accessible
- understandable
- low threshold



More specific:
Computer systems are designed and anything that's designed has an intended *purpose.* If a system is not satisfying, it was not designed properly or is used for other stuff than it was primarily designed for. **Requiremens engineering** provides a framework to understand the purpose.
In seeking to describe the purpose of a computer system, we need to look beyond the system itself, and into the human activities that it will support. For example, the purpose of a banking system is not to be found in the technology used to build such systems, but in the business activities of banks and day-to-day needs of their customers. The purpose of an online flight reservation system is not to be found in the details of the web technology used to implement it, but in the desire by passengers for more convenient ways of booking their travel, and the desire of airlines to provide competitive and profitable services.

Fitness for purpose
We define quality in terms of fitness-for-purpose. This means that we cannot assess quality as a measure of software by itself; we can only assess quality when we consider the software in the context of a set of human activities. In other words, software on its own cannot be said to be of 'high quality' or of 'low quality', because quality is an attribute of the relationship between software and the purposes for which it is used. This means that requirements engineering plays a critical role in our understanding and assessment of software quality.

def. **Requirements Engineering (RE)** is a set of activities concerned with identifying and communicating the purpose of a software-intensive system, and the contexts in which it will be used. Hence, RE acts as the bridge between the real-world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies.
(The term 'software-intensive system' describes systems that incorporate hardware, software and human activities.)

Importance of RE - why good analysis is crucial:

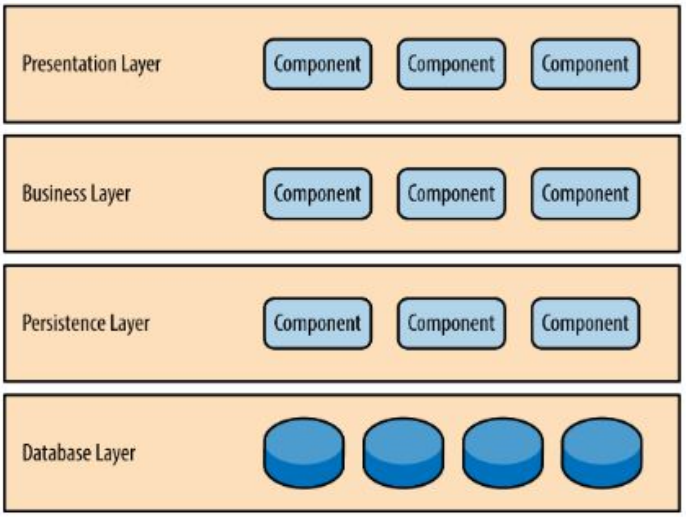
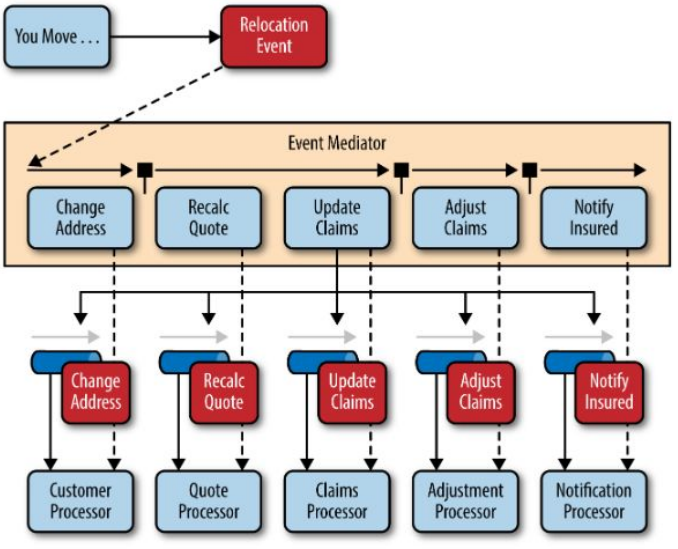
1) cost of fixing errors - One of the best-known investigations of the economics of software development was a series of studies conducted in the 1970's by Barry Boehm. Boehm investigated the cost to fix errors in the development of large software systems. Most of the projects followed a fairly standard sequence of phases: requirements analysis, design, coding (programming), development testing, acceptance testing and operation. Errors made in a particular phase cost more to fix the longer they are left undetected, so that, for example, a programming error that is not discovered until acceptance testing will cost ten times as much to fix than if it is found during programming. A requirements error not found until after delivery will cost a hundred times more to fix. The explanation is fairly simple – the longer a problem goes unnoticed, the more fix is going to cost
2) causes of projects failure - from other study - The most interesting part of this study was the list of success factors. When analyzing the causes of success or failure, the respondents to the survey cited the top three success factors as: user involvement; executive management support; and a clear statement of requirements. The top three factors leading to failure were: lack of user input; incomplete requirements and specifications; and changing requirements and specifications

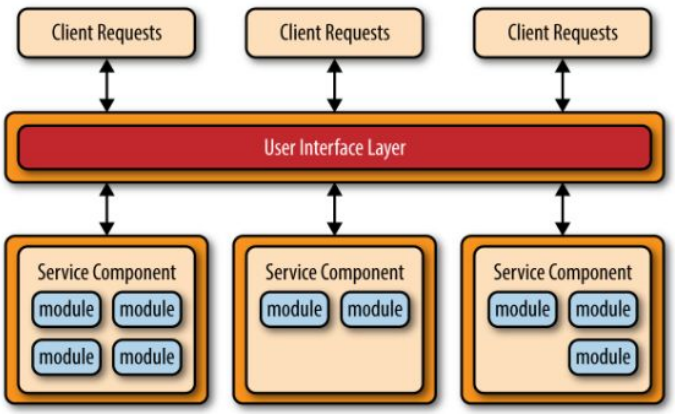
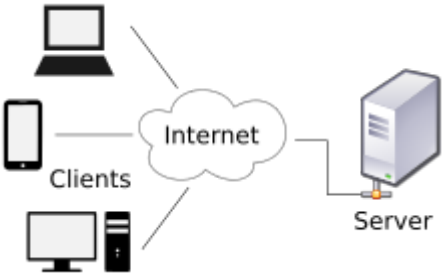
Źródło 1: http://www.gaudisite.nl/FundamentalsOfRequirementsPaper.pdf
Źródło 2: http://www.cs.toronto.edu/~sme/CSC340F/2005/readings/FoRE-chapter02-v7.pdf

11. Patterns (architectural, design, program).

Generalnie w necie można znaleźć miliardy podziałów i definicji tych patternów ze względu na model danych, integrację. aspekt biznesowy itd. Ja tutaj skupię się na perspektywie inżynierii oprogramowania (na to wskazuje kontekst pytania) i na patternach, które gdzieś się przewijały przez cały tok studiów.

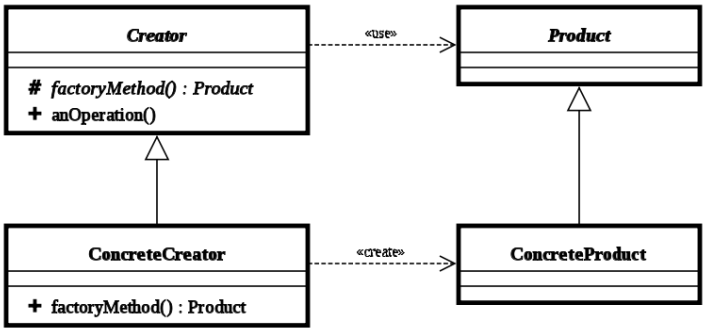
Architectural patterns (http://www.oreilly.com/programming/free/files/software-architecture-patterns.pdf):

Pattern + Opis	Schemat
Layered architecture - generalnie chodzi o rozdzielenie odpowiedzialności - warstwa widoku zajmuje się tylko prezentacją danych, business layer - logiką (decyzjami co wyświetlić, obliczeniami, przygotowanie danych), persistence odpowiada za komunikację z bazą, a database layer za fizyczne zapisanie danych. Zalety: łatwe testowanie poszczególnych warstw, Ułatwia development Wady: Niska skalowalność, performance nie najwyższych lotów Model View Controller - konkretny przykład architektury warstwowej. Model - utwardzanie danych, View - prezentacja, Controller - logika.	
Event-driven architecture - oparta na asynchronicznej komunikacji. Skalowalna, dobry performance ale skomplikowana w implementacji i testowaniu. Generalnie chodzi tutaj o to, że event jest “wyzwalaczem” dla serii dostępnych procesów. Przykład z prawej przedstawia schemat zmiany miejsca zamieszkania w systemie ubezpieczeń - jeden event uruchamia zbiór akcji procesorów (kolejność wykonywania procesorów jest zarządzana przez Event Mediatora, niektóre procesory mogą być zrównoleglone - Recalc Quote i Update Claims)	

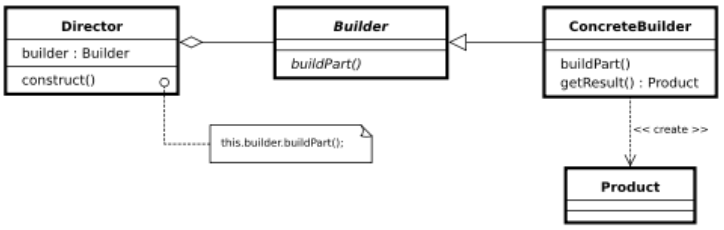
Microservices Architecture - architektura polegająca na rozbiciu systemu na mikroserwisy. Mikroserwis jest całkowicie niezależnym bytem co sprawia, że system staje się nisko związany (decoupled, czyli dobrze). Każdy mikroserwis udostępnia swoje zasoby/usługi przy użyciu jakiegoś protokołu (REST, SOAP, JMS, RMI…). Zalety: Elastyczny (bo decoupled, no i każdy serwisik może być napisany w innej technologii), Prosty w zarządzaniu (w przypadku zmiany w danym serwisie robimy deployment jednego serwisu zamiast całej kobyły), Prosty w testowaniu (serwisy odpowiadają za swoją robotę), Skalowalny (serwisy mogą być niezależnie skalowalne), Prosty w implementacji (mniejszy scope) Wady: Performance (overhead w związku z rozproszeniem, komunikacją)	
Client - server architecture - dodaję raczej jako przypomnienie, że to też się kwalifikuje jako pattern architektoniczny. Generalnie jaki jest to pattern każdy widzi. Dużo klientów, komunikacja z serwerem przy pomocy protokołu - Internet (TCP/IP żeby być bardziej precyzyjnym). Client tylko requestuje, cała praca i zasoby są po stronie serwera.	

Design patterns:

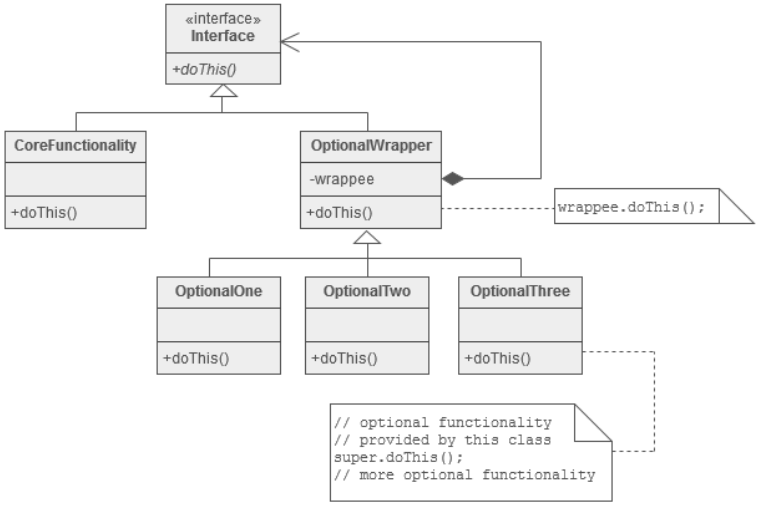
Pattern + Opis	Diagram

Factory method (creational)
Faktorka “chowa” logikę odpowiedzialną za tworzenie obiektu (wstrzykiwanie zależności, dobór parametrów itd). Dodatkowo zwracany jest interface obiektu przez co klient faktorki nie ma zależności na konkretny typ obiektu.
Przykład z kawą: zamiast tworzyć obiekt za pomocą:
`new Coffee(milk,sugar)`
gdzie obiekty `milk` i `sugar` muszą być przez nas stworzone, używamy faktorki, np.:
`CoffeeFactory.createCoffeeWithMi`
	

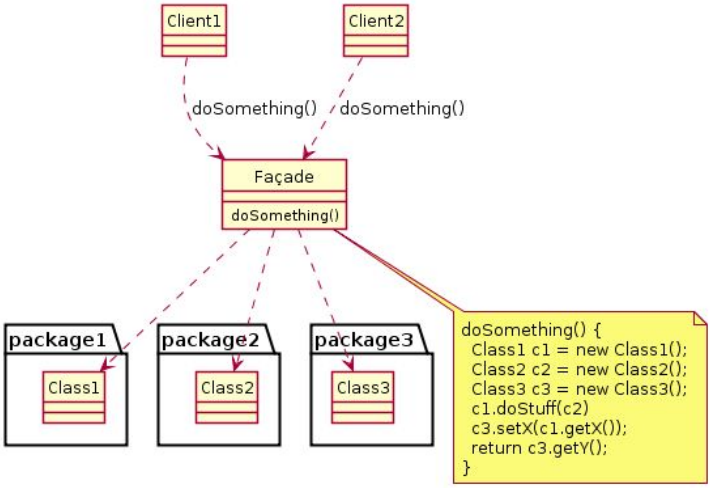
`lkAndSugar()`	

Builder (creational)
Chroni przed *telescoping constructor anti-pattern* czyli posiadaniem dużej ilości konstruktorów, w związku z dużą ilością parametrów i ich wariacjami.
Na przykładzie kawy - aby stworzyć kawę z cukrem wywołamy:
`CoffeeBuilder.setSuggar(true);`
`cws = CoffieBuilder.build();`
	

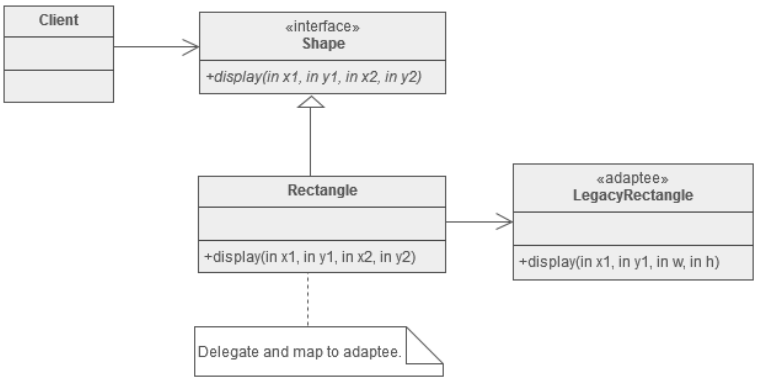
Singleton (creational) Chodzi o to żeby mieć tylko jeden obiekt danego typu (i pobieranie go za pomocą metody `Object.getInstance()`). Detale implementacyjne dla chętnych: - Konstruktor zmieniamy na prywatny, - Tworzymy statyczne pole z instancją - Tworzymy statyczną metodę getInstance() która zwraca instancję (lub tworzy ją przy pierwszym wywołaniu)	

Decorator (structural)
Dekorator rozszerza funkcjonalność obiektu bazowego. Z perspektywy implementacji dekorator zawiera obiekt podstawowy i jednocześnie rozszerza go (jest i ma). Cała zabawa polega na tym, że dekorator owrapowuje wywołanie metody obiektu bazowego o dodatkową funkcjonalność. Na przykładzie kawy:
w klasie *Coffee* mamy metodę:
`getPrice() {return 1;}`
w klasie *CoffeeWithMilkDecorator* mamy:
`getPrice() {return baseCoffee.getPrice() + 2;}`
Klasa *CoffeeWithMilkDecorator* dodaje zatem do ceny podstawowej kawy 2.
Częstym pytaniem na rozmowach kwalifikacyjnych (czyli nie spodziewajcie się takiego na egzaminie) jest podanie przykładów dekoratora w Java API. Przykładem jest *InputStream*, który może być udekorowany przez np. *FileInputStream*, *BufferedInputStream* itd.
	

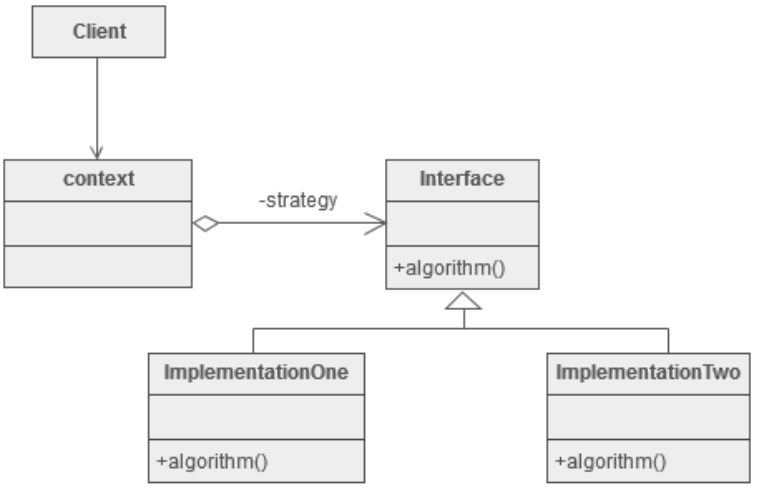
Facade (structural)
Fasada upraszcza użycie skomplikowanego API do prostych metod. Aby przygotować kawę musimy:
`boilWater()`
`putCoffeeIntoCup()`
`pourWater()`
`stir()`
kolejność tych metod będzie zawsze taka sama, więc sensownym wydaje się utworzenie fasady, która wykona to wszystko za nas, a my użyjemy tylko:
`CoffeeMakerFacade.makeCoffee()`



Adapter/Wrapper (structural)
Polega na konwersji danych/API tak aby umożliwić komunikację między klasami. W przykładzie po prawej klient oczekuje, że do wyświetlenia kształtu użyje metody gdzie poda współrzędne kształtu. Niestety, klasa *LegacyRectangle* oferuje API gdzie podajemy współrzędne wierzchołka + szerokość i wysokość. Tworzymy zatem adapter *Rectangle*, gdzie wystawiamy metodę przyjmującą współrzędne, w adapterze obliczamy wysokość i szerokość i delegujemy pracę do *LegacyRectangle*

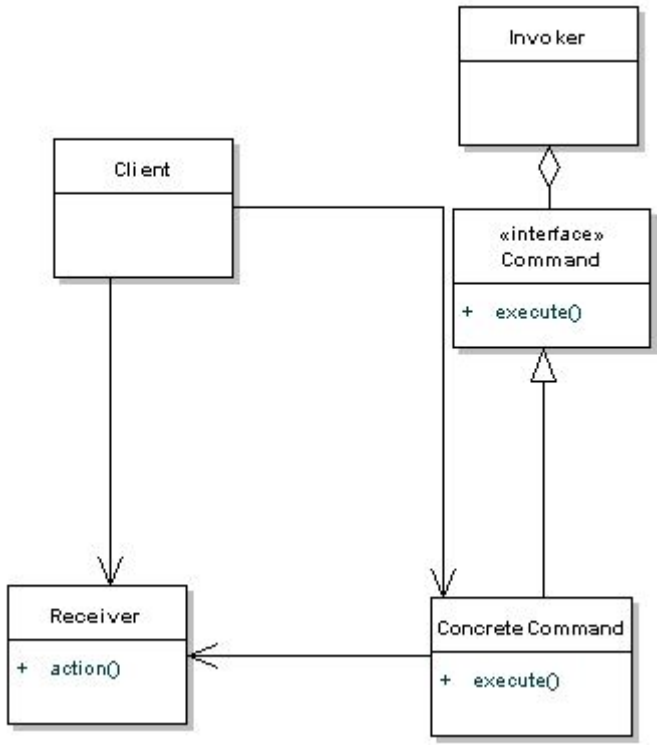


Strategy (behavioral)
Umożliwia na wybór innej strategii dla pewnego algorytmu w runtime'ie. Co więcej klient nie jest zależny od konkretnej strategii ale od interface'u.
Weźmy sobie klasę Calculator, która dokonuje operacji na 2 liczbach:
`calculate(int a,int b){operation.do(a,b)}`
gdzie operation jest strategią (interface *Operation*). Następnie możemy podmieniać strategie operacji w runtime'ie:
```
calculator.setOperation(new Add())
calculator.calculate(4,7) //11
calculator.setOperation(new Subtract())
calculator.calculate(4,7) // -3
```
gdzie klasy Add i Subtract implementują interface *Operation*

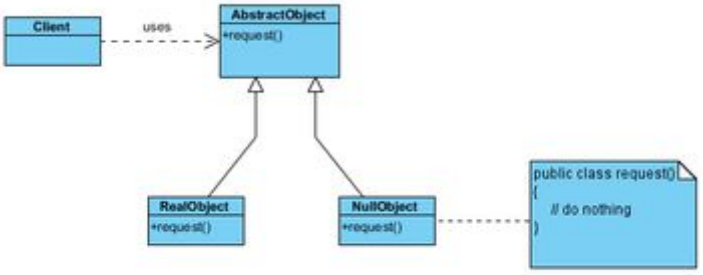


Command (behavioral)
Wzorzec podobny do strategii (wszystkie polecenia implementują interface *Command*, choć wykonują inne polecenia). Różnica polega na tym, że tutaj klient wie o konkretnych poleceniach, a nie o interface'ie. Invoker natomiast widzi sam interface *Command* na którym wykonuje execute(). *Receiver* jest natomiast modelem na którym wykonywane są operacje.
Przykład:

```
RemoteControl control = new
RemoteControl(); //Invoker
Light light = new Light(); //Receiver
// konkretne polecenia
Command lightsOn = new
LightsOnCommand(light);
Command lightsOff = new
LightsOffCommand(light);
//switch on
control.setCommand(lightsOn);
control.pressButton();
//switch off
control.setCommand(lightsOff);
control.pressButton();
```



Null Object (behavioral)
Zamiast zwracać null możemy dostarczyć obiekt z dummy implementacją, która nie wykonuje żadnych poleceń. Celem jest pozbycie się null checków z kodu.



Program patterns - w literaturze i internetach nie znalazłem takiego pojęcia. Prawdopodobnie synonim dla design patterns.

12. The effectiveness of information systems.

Why/when shall effectiveness be measured:

- when problems arise within the system or it's criticized
- when indicators of ineffectiveness are discovered
- before implementing a system to check its effectiveness and overall quality
- post-implementation review to determine whether a systems meets its purpose

Indicators of ineffectiveness:

- excessive down time and idle time
- slow system response time
- excessive maintenance costs
- inability to interface with new hardware/software
- unreliable system outputs
- data loss
- excessive run costs
- frequent need for program maintenance and modification
- user dissatisfaction with output format, content or timeliness.

Two approaches to measure effectiveness:

- Goal-centered view - does system achieve goals set out?
    - Conflicts as to priorities, timing etc. can lead to objectives met in the short run by sacrificing fundamental system qualities, leading to long run decline of effectiveness of the system
- System resource view - desirable qualities of a system are identified and their levels are measured.
    - If the qualities exist, then information system objectives, by inference, should be met. By measuring the qualities of the system may get a better, longer-term view of a system's effectiveness.

Table 1 Criteria for evaluating the performance functionality of information system

No.	Criterions	Agreeing completely				Disagreeing completely
1.	Users (and managers) perceive IS as satisfactory	5	4	3	2	1
2.	IS is approachable to users whenever and whereever they need it	5	4	3	2	1
3.	IS is compatible to other parts of organization	5	4	3	2	1
4.	Existing good documentation that describes IS and formal procedures for usage	5	4	3	2	1
5.	Good user's training for usage of IS	5	4	3	2	1
6.	IS can be easily altered and adapted to new conditions and demands	5	4	3	2	1
7.	High compliance between user's demands and IS abilities	5	4	3	2	1
8.	No large amount of effort needed for maintaining satisfactory functioning of IS	5	4	3	2	1
9.	Appearance of mistakes is minimizad	5	4	3	2	1
10.	Usage of IS is very simple	5	4	3	2	1
11.	High safety of data and the model in IS	5	4	3	2	1
12.	IS in very good manner provides achievment of business and organizacional goals	5	4	3	2	1
13.	IS helps organization to achieve high benefits with relatively small investments	5	4	3	2	1
14.	Detail check can be done in IS in order to minimize operational mistakes and unsatisfaction of user	5	4	3	2	1
15.	Clarity of IS output information is high	5	4	3	2	1
16.	Output information of IS are consistent	5	4	3	2	1
17.	Output information of IS are accúrate enough for their purpose	5	4	3	2	1
18.	Necessary output information of IS are prompt	5	4	3	2	1
19.	Content of output information of IS is elaborated enough	5	4	3	2	1
20.	Output information of IS are very important for solving business problems and achievment of organizacional goals	5	4	3	2	1
21.	Presentation of output information of IS is in appropriate form	5	4	3	2	1
22.	IS in great deal provides information necessary for decision making in managing organizationstrategically	5	4	3	2	1

Źródło 1: http://www.ef.uns.ac.rs/mis/archive-pdf/2009%20-%20No2/MIS2009_2_2.pdf
Źródło 2: Evaluating information system effectiveness and efficiency

13. Modeling of complex operation systems.

Slajdy z przedmiotu System Modeling and Analysis:
http://modeleisystemy.pl/wp-content/uploads/2014/02/lecture-15.pdf
http://modeleisystemy.pl/wp-content/uploads/2014/02/lecture-14.pdf

A complex input output system with M elementary subsystems

Example: task allocation for a complex of parallel operations. The completion time of complex of operations is optimal as long as all operations are completed at the same moment.

14. The concept of decision-making system and computerized decision support system. K

Decision making system (DM software) is the name of computer applications that are used to help individuals and organisations make choices and take decisions, typically by ranking, prioritizing or choosing from a number of options.

First DM systems appeared on the market in 1970s and the main bloom was in 1990s. Most DM software focuses on ranking, prioritizing or choosing from among alternatives characterized on multiple criteria or attributes. Examples of software: 1000Minds, Ahoona, Analytica, D-Sight, DecideIT, M-Macbeth, Priest…

Decision making methods commonly used in the software include:

- Analytic network process,
- Measuring Attractiveness by a Categorical Based Evaluation Technique (MACBETH),
- Analytic Hierarchy Process (AHP)
- and others.

A decision support system (DSS) is a computer-based information system that supports business or organizational decision-making activities, typically resulting in ranking, sorting, or choosing from among alternatives. DSSs serve the management, operations, and planning levels of an organization

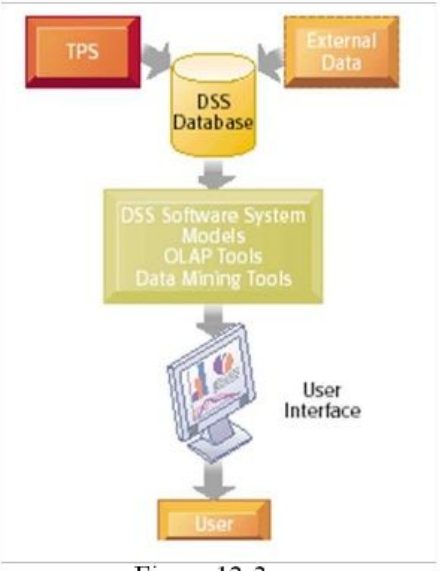
.

Another thing is an Expert System (może chodziło o to?), in artificial intelligence, a computer system that emulates the decision-making ability of a human expert. Expert systems are designed to solve complex problems by reasoning about knowledge. An expert system is an example of a knowledge-based system. Expert systems were the first commercial systems to use a knowledge-based architecture. A knowledge-based system is essentially composed of two sub-systems: the knowledge base (set of facts, information provided or gathered before) and the inference engine (uses the base to analyse the current situation and give a result).

A distinction between Decision Support Systems (DSS) and Decision Making Systems (DMS) is that the former supports a human decision maker in some decision-making tasks whereas the latter also assumes the role of decision making. An oft used example to illustrate this distinction is that whereas the Financial What-If Analysis tool in the example above is a DSS (it allows the decision maker (DM) to explore alternatives, but the DM makes the final decision based on such analysis), an Expert System (ES) that does such exploration and also makes a recommendation would be a Decision Making System.

3 Components of DSS:

- Model Management
- Data Management
- User Interface Management



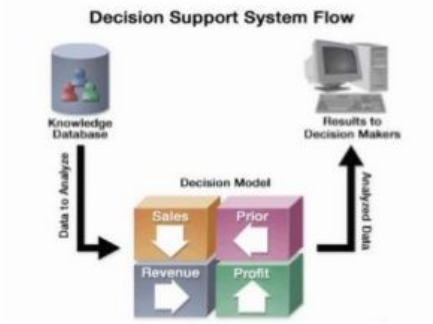


Figure 12-3

Examples…of DSS:

Company or Application	Description
Cinergy Corporation	The electric utility developed a DSS to reduce lead time and effort required to make decisions in purchasing coal.
RCA	The company developed a DSS called Industrial Relation Information System (IRIS) to help solve personnel problems and issues.
U.S. Army	It developed a DSS to help recruit, train, and educate enlisted forces. The DSS uses simulation that incorporates what-if features.
National Audubon Society	It developed a DSS called Energy Plan (EPLAN) to analyze the impact of U.S. energy policy on the environment.
Hewlett-Packard	The computer company developed a DSS called Quality Decision Management to help improve the quality of its products and services.
Virginia	The state of Virginia developed the Transportation Evacuation Decision Support System (TEDSS) to determine the best way to evacuate people in case of a nuclear disaster at its nuclear power plants.

DSS

- Facilitates decision making
- Unstructured environment
- Alternatives may not be fully realized yet
- Extract or gain knowledge from a computer system
- Use goals and the system data to establish alternatives and outcomes, so a good decision can be made

Expert Systems

- Automate decision making.
- The decision environments have structure
- The alternatives and goals are often established in advance.
- Inject expert knowledge in to a computer system.
- The expert system can eventually replace the human decision maker.

Anita Johri - INFS 401

Source: https://www.slideshare.net/anitajohri/dss-vs-expert-system

15. Modeling, identification, and aiding of decision making process.

For a decision maker (DM) it is important to take due regard of the expert judgements, current science and respected theories and evidence that might be summarised within a probabilistic expert system. However he needs to apply such domain knowledge to the actual problem he faces. He will usually only need to use a small subset of the expert information available. He therefore needs not only to draw on that small subset of the expert information that is relevant to the problem at hand - augmenting and complementing this as necessary with other context specific information but also to use this probabilistic information to help him make the best decision he can on the basis of the information available

Models for decision making process:

- Diagnostic Hypothesis generation

Diagnostic hypothesis-generation processes are ubiquitous in human reasoning. For example, clinicians generate disease hypotheses to explain symptoms and help guide treatment.
The example given above is only one of many real-world inductive inference tasks in which the decision maker is required to generate hypotheses on the basis of data. Indeed, hypothesis generation processes occur in any task that involves taking data and formulating possible explanations of those data, including clinicians' generation of diagnoses on the basis of symptoms.
To begin, we make the distinction between events external to the decision maker and events internal to the decision maker. External events represent what we call the universe of possible states. This is the exhaustive collection of events that are related to data. We use the term hypothesis to refer to one's mental representation

of an external event. The distinction between external events and one's mental representation of the external events (i.e., one's hypotheses) is represented by the two largest concentric circles in the Venn diagram in Figure 1: Non-represented events are denoted with a ?, indicating that these states of the universe are unknown to the decision maker. Although these unknown events exist in the ecology, they have not been experienced or registered by the semantic memory system. Represented hypotheses are denoted by H, which we take (for convenience) to be part of the semantic memory system. We assume that semantic memory is populated with representations of experienced hypotheses (H) and the data (D) with which they typically are associated. humans generate sets of hypotheses, form probability judgments based on those sets, and select hypotheses from the generated set as explanations of observable data

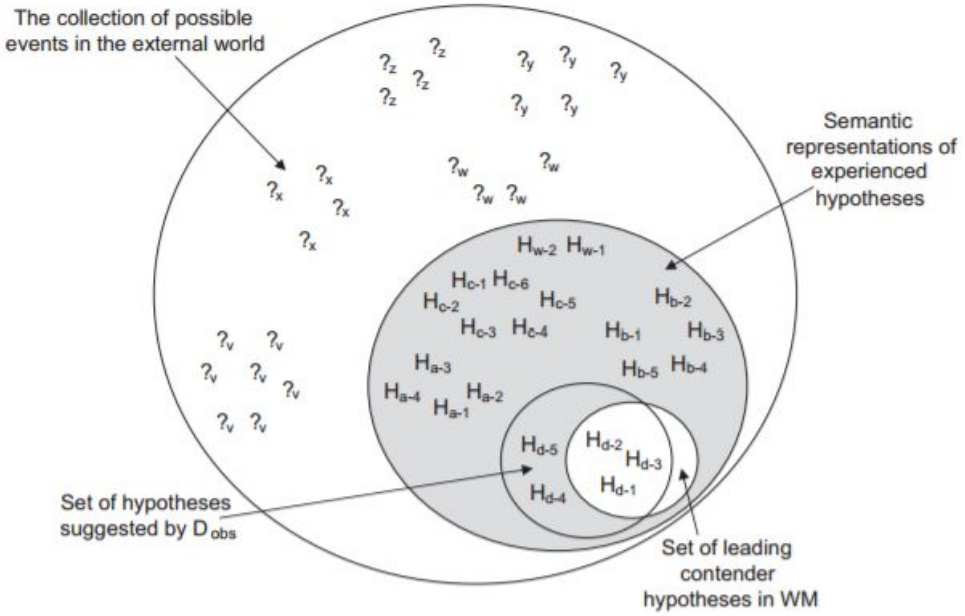


Figure 1. Venn diagram of semantics as defined within HyGene. Elements denoted with common alphabetic subscripts are members of the same cluster of hypotheses. Elements denoted by a ? are possible explanations of the D_{obs} that are not within the observer's semantic knowledge. Elements denoted with an H are within the observer's semantic knowledge. D_{obs} = pattern of observable data; WM = working memory.

- Bayesian modeling

In any given decision-making situation, there are relevant quantities or outcomes we have observed and recorded, and other relevant quantities or outcomes we have not yet observed and recorded – and are therefore uncertain about. In order to make rational decisions, we need to weigh up all relevant uncertainties and provide some form of quantification. Bayesian statistics provides this quantification in the form of probability statements, having taken into account all relevant evidence provided by the quantities and outcomes we have observed and recorded.
Bayes Theorem is a way to update beliefs in light of new evidence (a posteriori). Humans are rational inference machines…we are testing hypotheses based on evidence from the environment. Assume you are trying to infer the process that was responsible for generating observable data (D). H is the hypothesis about the true generating process.
By Bayes Theorem: $P(H \mid D) = P(D \mid H) * P(H) / P(D)$
• P(H|D) = posterior (Hyp given Data)
• P(H) = prior
• P(H|D) = (Data given Hyp) conditional or likelihood

• P(D) = marginal or normalizing constant (ensures prob is normalized to sum one)
The posterior is proportional to the product of the prior probability and the likelihood

The a priori or prior probability reflects our knowledge of how likely we expect a certain state of nature before we can actually observe said state of nature.

16. Basic problems, methods and algorithms of discrete optimization.

Discrete optimization or **combinatorial optimization** means **searching for an optimal solution** in a set of potential solutions. Optimality is defined with respect to some **criterion function** (see: fitness function in question 21), which is to be **minimizes** or **maximized**.

Examples of **minimization**: cost, distance, travel time, energy consumption, number of objects.
Examples of **maximization**: profit, value, efficiency, capacity, number of objects.

The solutions may be combinatorial structures like: combinations, sequences, subsets, subgraphs, routes in a network, schedules of jobs, etc.

Examples of **optimization problems**:

1) **Traveling salesman problem** (TSP)

Find a minimum weight Hamiltonian (visiting each vertex exactly once) cycle in a given complete weighted graph.
Application: a salesman has to visit all cities and come back home. Distances between cities are given. Find the shortest route.

2) **Postman problem** (or chinese postman problem)

In a weighted graph, find a minimum weight closed walk that includes every edge at least once.
Application: a postman has to deliver mail to all streets in a certain neighbourhood. Find the shortest route.

3) **Knapsack problem**

Given: n objects (each has value v_i and weight w_i assigned) and maximum capacity M of the knapsack.
Choose a subset of objects such that the total weight does not exceed the capacity M of the knapsack and the total value is maximized.

4) **Scheduling problem**

Given n jobs are to be completed using m machines. Every job consists of a sequence of operations on different machines. The total schedule of operations is to be determined such that the total **makespan (total length** of the schedule = when all jobs have finished processing) of all operations is minimized.

The main **categories of solution methods**:

- **Exact** methods - are guaranteed to find an optimal solution in finite computational time (example of exact method - *brute force* - checking all solutions).
- **Heuristic** methods (or heuristics) - have no guarantee for optimality but they are based on reasonable, justifiable principles leading to good or near-optimal solutions. These methods are often easy to implement and efficient.

Heuristics are preferred over exact methods **when**:

1) the problem is so large that exact methods would take too long
2) no exact solution algorithms exist
3) a feasible and functional solution is needed quickly
4) problem data is uncertain, inaccurate or fuzzy - exact solution isn't of much worth

Examples of heuristic methods:

1) local search (neighbourhood search) - initial feasible solution modified to improve fitness function value, trial solutions taken from the specified neighbourhood of the current solution

2) random search - checking random trial solutions are making random changes in a current solution
3) genetic algorithm - several solutions are checked simultaneously, modified, improved, compared with each other

An **approximation algorithm** - an algorithm that produces a solution that its fitness function value is within specified distance from the optimum (**g-approximation** if value at most g times worse).

More info: http://www.mafy.lut.fi/study/DiscreteOpt/CH1.pdf

17. Basic methods of "soft computing?.

In computer science, **soft computing** (sometimes referred to as computational intelligence, though CI does not have an agreed definition) is the use of inexact solutions to computationally hard tasks such as the solution of NP-complete problems, for which there is no known algorithm that can compute an exact solution in polynomial time. Soft computing differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth, and approximation. In effect, the role model for soft computing is the human mind.

Components of soft computing include:

- Machine learning, including:
    - Neural networks (NN)
        - Perceptron
    - Support Vector Machines (SVM)
- Fuzzy logic (FL)
- Evolutionary computation (EC), including:
    - Evolutionary algorithms
        - Genetic algorithms
        - Differential evolution
    - Metaheuristic and Swarm Intelligence
        - Ant colony optimization
        - Particle swarm optimization
- Ideas about probability including:
    - Bayesian network

Soft computing can also be seen as a foundation for the growing field of computational intelligence (CI). The difference between traditional artificial intelligence (AI) and computational intelligence is that AI is based on hard computing whereas CI is based on soft computing

18. Rules for specification of the relational database model.

Codd's twelve rules are a set of thirteen rules (numbered zero to twelve) proposed by Edgar F. Codd, a pioneer of the relational model for databases, designed to define what is required from a database management system in order for it to be considered *relational*, i.e., a relational database management system (RDBMS) They are sometimes jokingly referred to as "Codd's Twelve Commandments".

Rule 0: The *foundation rule*:

For any system that is advertised as, or claimed to be, a relational database management system, that system must be able to manage databases entirely through its relational capabilities.

Rule 1: The *information rule*:

All information in a relational database is represented explicitly at the logical level and in exactly one way – by values in tables.

Rule 2: The *guaranteed access rule*:

Each and every datum (atomic value) in a relational database is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name.

Rule 3: Systematic treatment of null values:

Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type.

Rule 4: *Dynamic online catalog based on the relational model*:

The data base description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data.

Rule 5: The *comprehensive data sublanguage rule*:

A relational system may support several languages and various modes of terminal use (for example, the fill-in-the-blanks mode). However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings and that is comprehensive in supporting all of the following items:

1. Data definition.
2. View definition.
3. Data manipulation (interactive and by program).
4. Integrity constraints.
5. Authorization.
6. Transaction boundaries (begin, commit and rollback).

Rule 6: The *view updating rule*:

All views that are theoretically updatable are also updatable by the system.

Rule 7: *High-level insert, update, and delete*:

The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data.

Rule 8: *Physical data independence*:

Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representations or access methods.

Rule 9: *Logical data independence*:

Application programs and terminal activities remain logically unimpaired when information-preserving changes of any kind that theoretically permit unimpairment are made to the base tables.

Rule 10: *Integrity independence*:

Integrity constraints specific to a particular relational data base must be definable in the relational data sublanguage and storable in the catalog, not in the application programs.

Rule 11: *Distribution independence*:

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only.

Rule 12: The *nonsubversion rule*:

If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher level relational language (multiple-records-at-a-time).

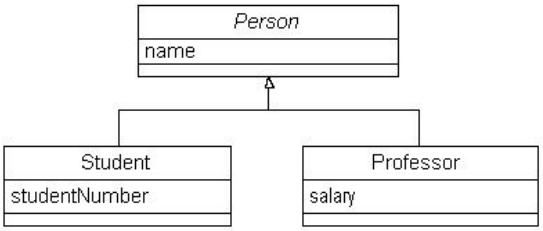
19. Rules for mapping class diagrams onto relational models.

https://www.ibm.com/developerworks/library/ws-mapping-to-rdb/

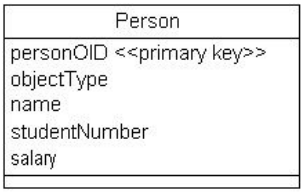
Class diagram - focused on class data and behaviour.
Relational model - focused on data.

Rules:

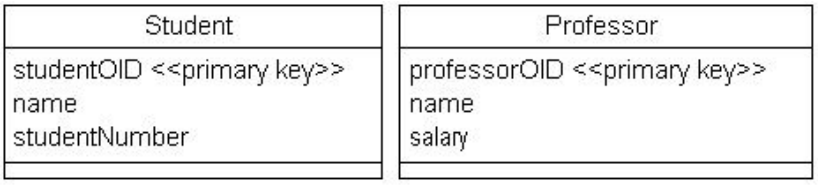
- Mapping attributes to columns
  - not all class attributes are persistent (some may be calculated - for example - invoiceTotal)
  - attribute may be an object represented by the whole table (for example Course has TextBook which is persisted in other table, see point “Relationships between classes”)
- Inheritance



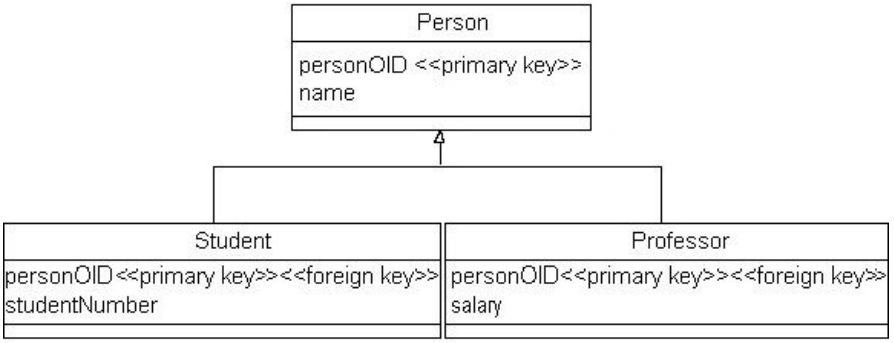
  - No inheritance in relational model - 3 possible solutions to handle it:
    - One table - there is one table for whole hierarchy - fields not applicable for a given type are set to null, one additional column - objectType - is needed in order to recognise type of persisted object



    - One table per concrete class - table is created only for child classes. Attributes of parent class are moved to each table with concrete class representation.



    - One table per class - each class in the hierarchy is represented in separate table



- Relationships between classes

- Composition, Aggregation, One to many, One to one - represented as foreign key to another table which persists entity.
- Many to many relationship - associative table need to be introduced.

Existing frameworks for object-relational mapping for different languages:
JPA (Java), Doctrine (PHP), NPA (C#), Django ORM (Python)

20. The SQL 2003 standard.

SQL 2003 is based on previous revisions of SQL standard, the most important – SQL 92.
SQL statements are grouped into following categories:

- Data manipulation : The Data Manipulation Language (DML) is the subset of SQL which is used to add, update and delete data.
- Data definition : The Data Definition Language (DDL) is used to manage table and index structure. CREATE, ALTER, RENAME, DROP and TRUNCATE statements are to name a few data definition elements.
- Data control : The Data Control Language (DCL) is used to set permissions to users and groups of users whether they can access and manipulate data.
- Transaction : A transaction contains a number of SQL statements. After the transaction begins, all of the SQL statements are executed and at the end of the transaction, permanent changes are made in the associated tables.
- Procedure : Using a stored procedure, a method is created which contains source code for performing repetitive tasks.

Introduced XML-related features (SQL/XML), window functions, standardized sequences, and columns with auto-generated values (including identity-columns), new MERGE statement, extended CREATE TABLE statement and enhancements to SQL-invoked routines.

http://web.archive.org/web/20071011220454/http://www.sigmod.org/sigmod/record/issues/0403/E.JimAndrew-standard.pdf

21. Evolutionary Computation.

https://en.wikipedia.org/wiki/Evolutionary_computation

In computer science, evolutionary computation is a family of algorithms for global optimization inspired by biological evolution.

Evolutionary computation - a general name for a group of problem-solving techniques whose principles are based on the theory of biological evolution, such as genetic inheritance and natural selection.

In evolutionary computation, an initial set of candidate solutions is generated and iteratively updated. Each new **generation** is produced by removing not desired solutions and introducing small random changes. In biological terminology, a **population** of solutions is subjected to **selection** and **mutation**. As a result, the value of the **fitness function** of the algorithm will increase for the population.

Examples of algorithms using evolutionary computation: genetic algorithm, ant colony optimization, evolutionary algorithms.

The **evaluation function** (**fitness function**) represents the requirements to adapt to. It is a function that assigns a quality measure to a solution.

Possible algorithm **termination conditions**:
(source: http://www.cs.vu.nl/~gusz/ecbook/Eiben-Smith-Intro2EC-Ch2.pdf)

- fitness function reaches a desired level,
- max CPU time elapsed,
- total number of fitness function evaluations reached,
- population diversity dropped under a given threshold.

22. Introduction to machine learning, deduction versus induction. (reasoning)

def. 1. Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.

def.2. *"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E." -- Tom Mitchell, Carnegie Mellon University*

Two types:

- Supervised machine learning (= function approximation): The program is "trained" on a predefined set of "training examples"(labelled data), which then facilitate its ability to reach an accurate conclusion when given new data.
  - **Supervised machine learning algorithms** can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

Classification	Regression
Taking some kind of input x, and mapping it to discrete number of labels example: mapping portrait pictures to male or female label	Mapping from input space to some real number. example: have bunch of points, got a new one, map it to some real value
output: discrete set	output: continuous
You use classification to predict a category. There is a wide variety of classification applications from medicine to marketing. Classification models include linear models like Logistic Regression, SVM, and nonlinear ones like K-NN, Kernel SVM and Random Forests.	Regression models (both linear and non-linear) are used for predicting a real value, like salary for example. If your independent variable is time, then you are forecasting future values, otherwise your model is predicting present but unknown values. Regression technique vary from Linear Regression to SVR and Random Forests Regression.
1. Logistic Regression 2. K-Nearest Neighbors (K-NN) 3. Support Vector Machine (SVM) 4. Kernel SVM 5. Naive Bayes 6. Decision Tree Classification 7. Random Forest Classification	1. Simple Linear Regression 2. Multiple Linear Regression 3. Polynomial Regression 4. Support Vector for Regression (SVR) 5. Decision Tree Classification 6. Random Forest Classification

  -

- Unsupervised machine learning(=function description): The program is given a bunch of data and must find patterns and relationships therein
  - **Unsupervised machine learning algorithms** are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

Clustering
Clustering is similar to classification, but the basis is different. In Clustering you don't know what you are looking for, and you are trying to identify some segments or clusters in your data. def. **Cluster analysis** or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.
● K-Means clustering ● Hierarchical clustering

- ALSO:
  - **Semi-supervised machine learning algorithms** fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiringunlabeled data generally doesn't require additional resources.
  - **Reinforcement machine learning algorithms (=learning from delayed reward)** is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

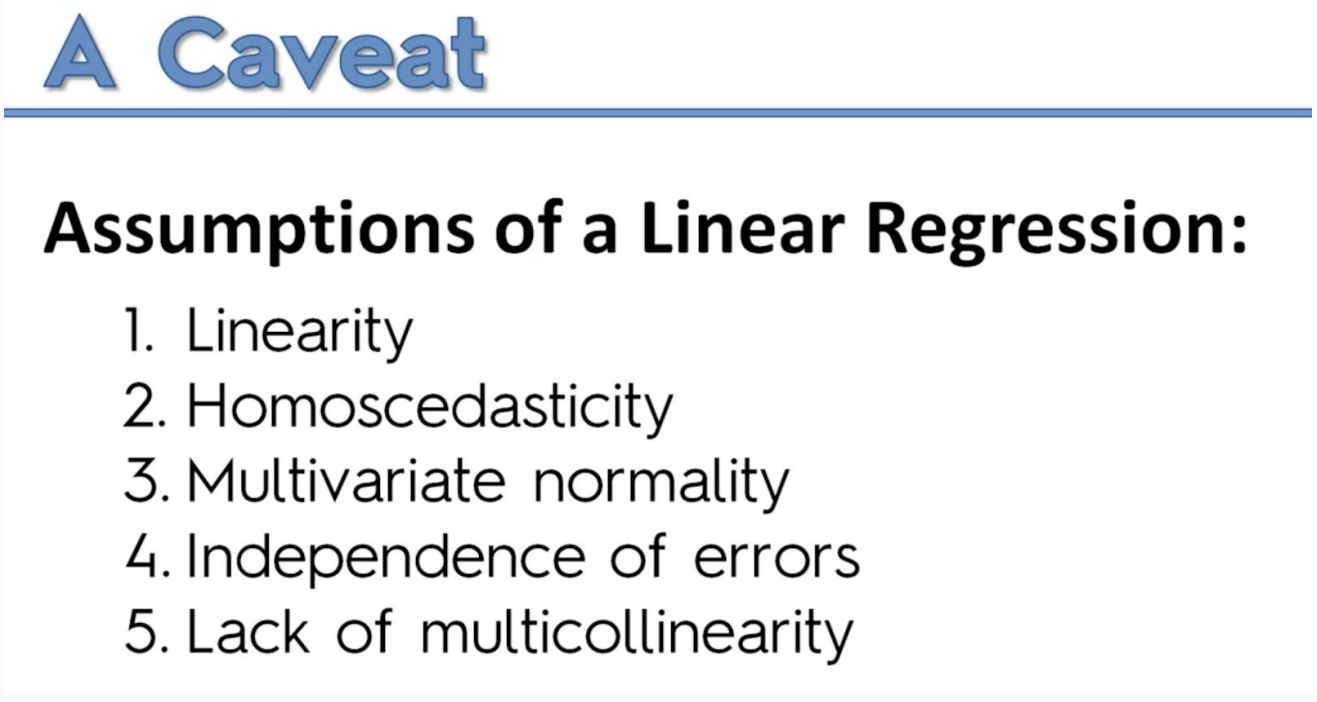
- INDUCTION - specifics to generalities - reasoning
- DEDUCTION - generalities to specifics - i.e. rule-based approach

Supervised learning is about induction
Glossary:

- instances - set of input
- concept - function that maps input to class (like pics to female, male), intuitively an idea that describes a set of things, for example with pictures a concept is an idea of what femaleness is and what maleness is
- target concept - answer
- hypothesis - all possible functions
- training set (sample) - set of inputs paired with label, explaining what a target concept is
- candidate - concept that you think might be a target concept
- testing set - input set w/o labels, for tests if candidate is a target concept

Algorithms overview

Regression



Assumptions of Linear Regression

Linear regression is an analysis that assesses whether one or more predictor variables explains the dependent (criterion) variable. The regression has five key assumptions:

- Linear relationship
- Multivariate normality
- No or little multicollinearity
- No auto-correlation

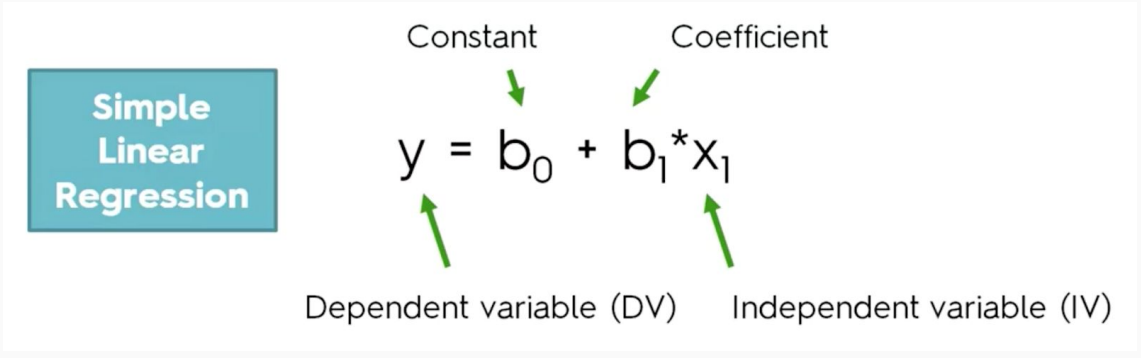
- Homoscedasticity

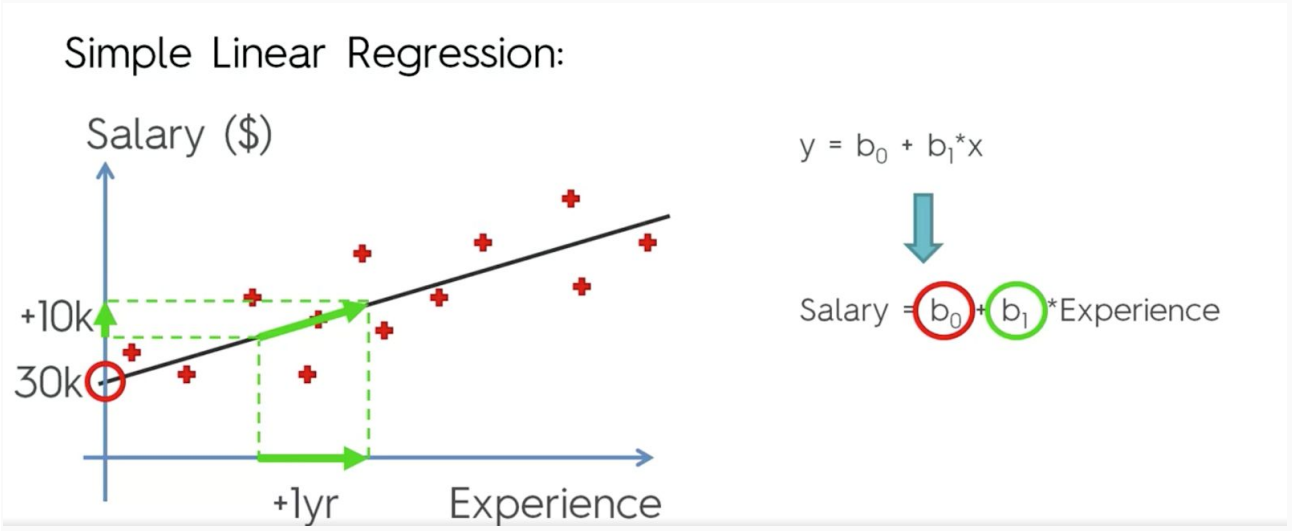
A note about sample size. In Linear regression the sample size rule of thumb is that regression analysis requires at least 20 cases per independent variable in the analysis.

1) linearity - linear regression needs the relationship between the independent and dependent variables to be linear. It is also important to check for outliers since linear regression is sensitive to outlier effects. The linearity assumption can best be tested with scatter plots, the following two examples depict two cases, where no and little linearity is present.
2) multivariate normality - linear regression analysis requires all variables to be multivariate normal. This assumption can best be checked with a histogram and a fitted normal curve or a Q-Q-Plot. Normality can be checked with a goodness of fit test, e.g., the Kolmogorov-Smirnof test. When the data is not normally distributed a non-linear transformation, e.g., log-transformation might fix this issue, however it can introduce effects of multicollinearity.
3) multicollinearity - linear regression assumes that there is little or no multicollinearity in the data. Multicollinearity occurs when the independent variables are not independent from each other. A second important independence assumption is that the error of the mean has to be independent from the independent variables.
4) no auto-correlation - linear regression analysis requires that there is little or no autocorrelation in the data. Autocorrelation occurs when the residuals are not independent from each other. In other words when the value of y(x+1) is not independent from the value of y(x). This for instance typically occurs in stock prices, where the price is not independent from the previous price.
5) homescedastcity - linear regression analysis makes is homoscedasticity. The scatter plot is good way to check whether homoscedasticity (that is the error terms along the regression are equal) is given.

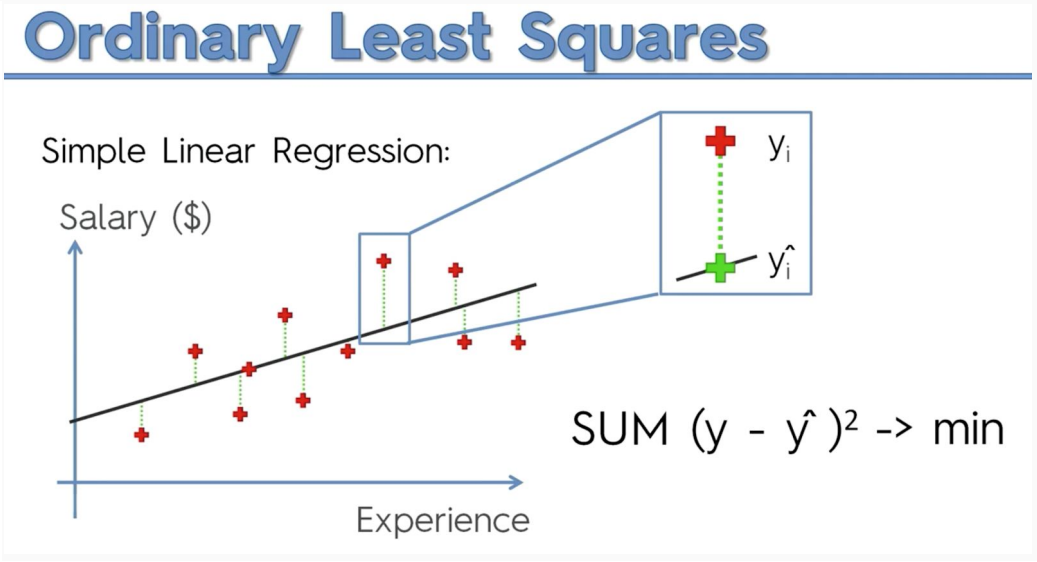
Źródło: http://www.statisticssolutions.com/assumptions-of-linear-regression/

1) Simple linear regression - finding the best fitting line





SUM shows the difference between the actual values and the model values



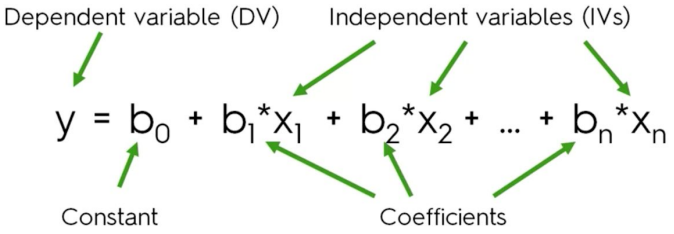
2) Multiple Linear Regression - like simple but with more variables

R&D Spend	Administration	Marketing Spend	State	Profit
165349.2	136897.8	471784.1	New York	192261.83
162597.7	151377.59	443898.53	California	191792.06
153441.51	101145.55	407934.54	Florida	191050.39
144372.41	118671.85	383199.62	New York	182901.99

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

Multiple Linear Regression



Dummy Variables

Profit	R&D Spend	Admin	Marketing	State	New York	California
192,261.83	165,349.20	136,897.80	471,784.10	New York	1	0
191,792.06	162,597.70	151,377.59	443,898.53	California	0	1
191,050.39	153,441.51	101,145.55	407,934.54	California	0	1
182,901.99	144,372.41	118,671.85	383,199.62	New York	1	0
166,187.94	142,107.34	91,391.77	366,168.42	California	0	1

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_3 + b_4 * D_1$$

3) Polynomial regression used when straight line does not do the job

Simple Linear Regression

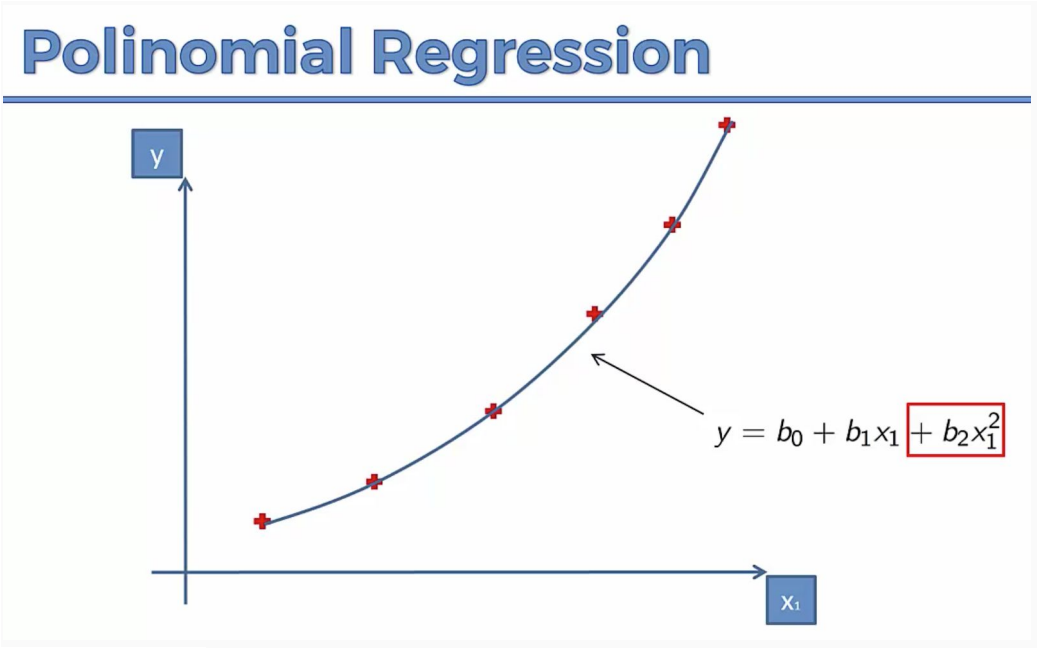
$$y = b_0 + b_1 x_1$$

Multiple Linear Regression

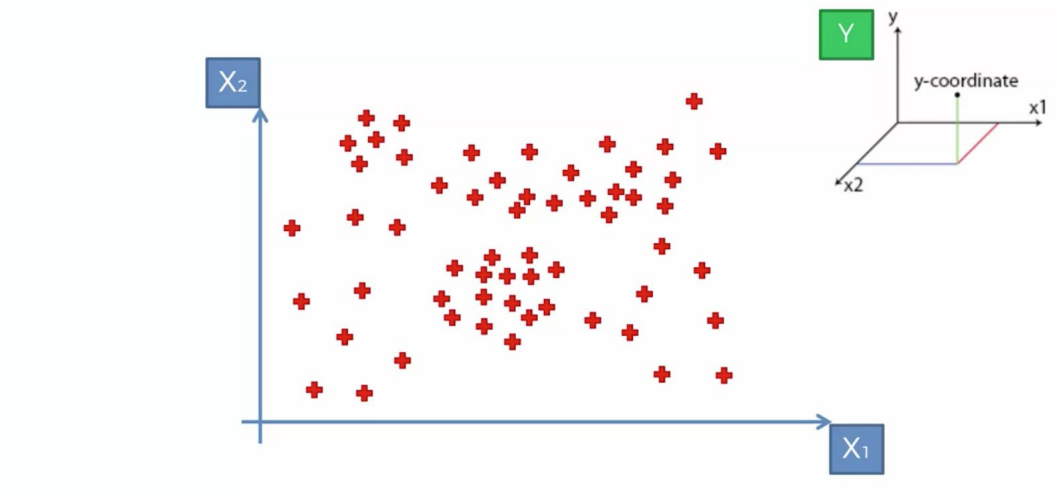
$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

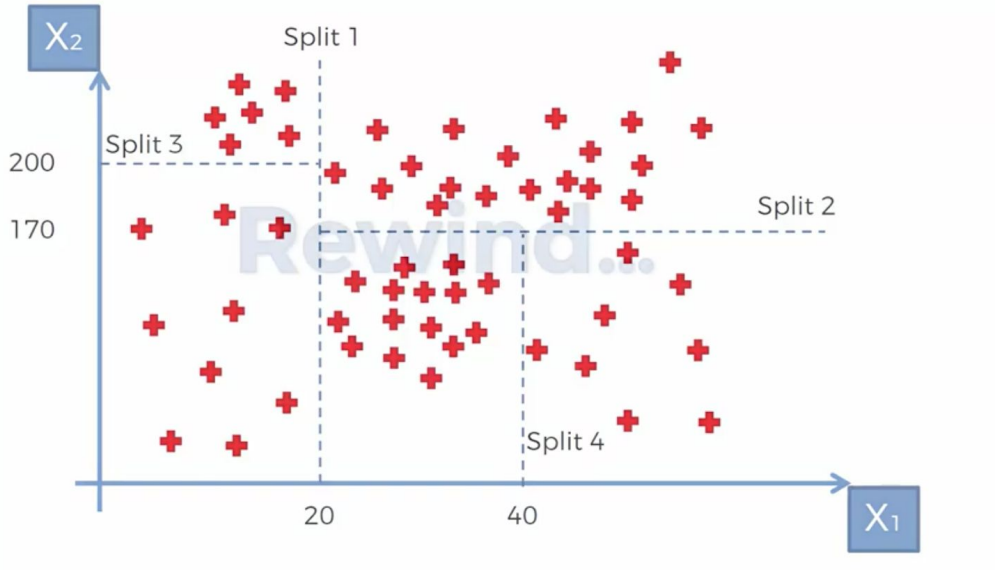
Polynomial Linear Regression

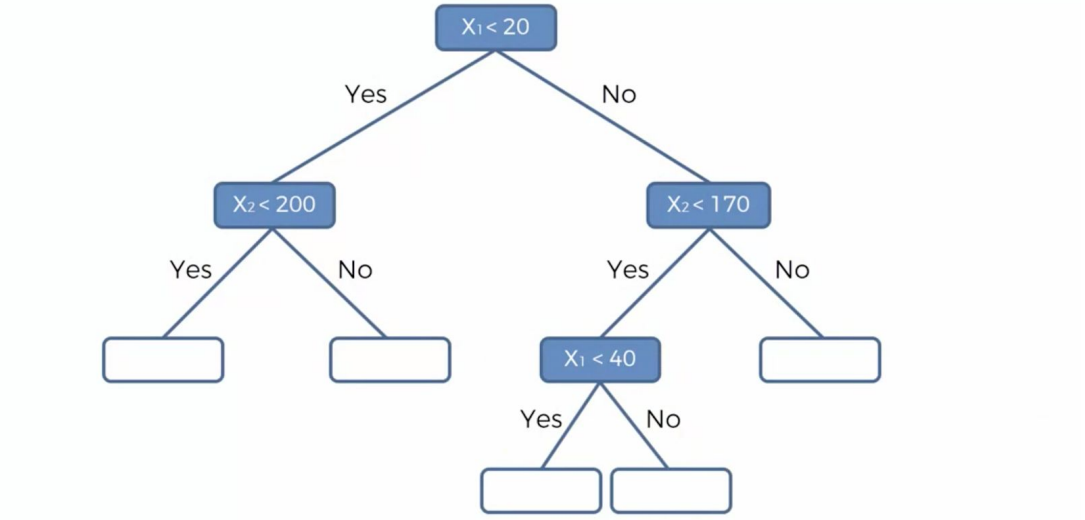
$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

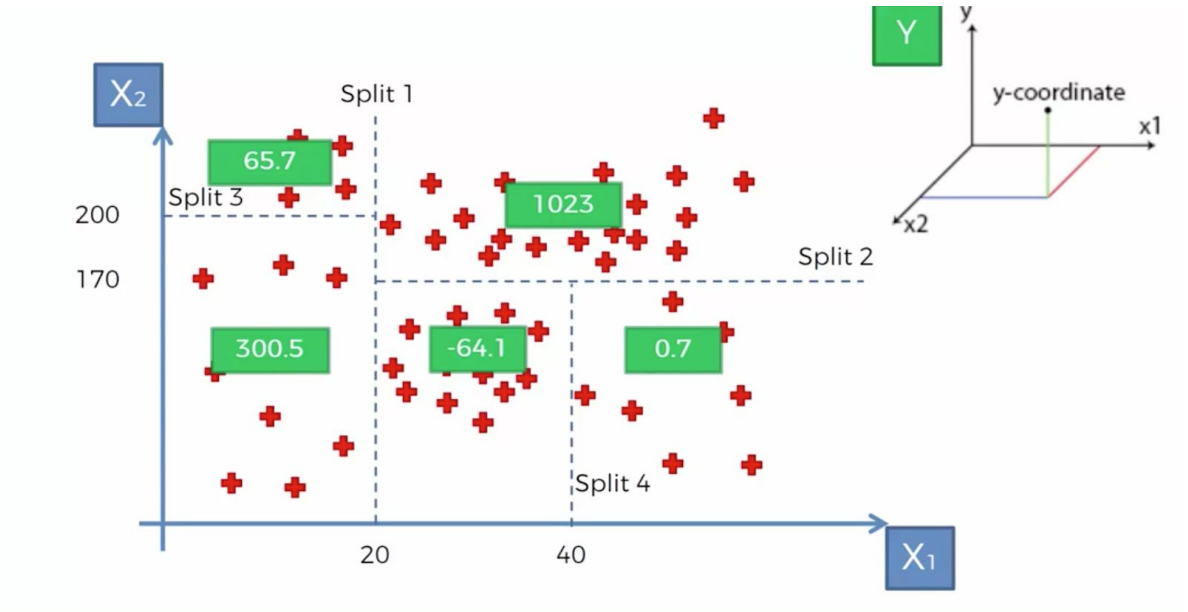


5) Decision Tree Regression

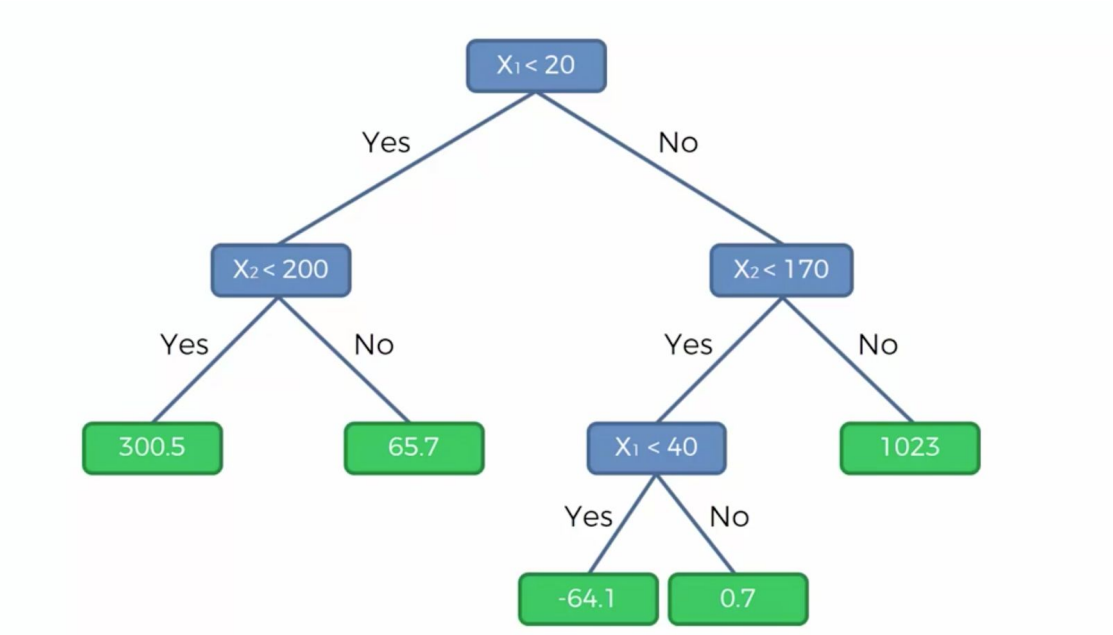



X2
Split 1
Split 3
200
170
Split 2
Split 4
20
40
X1

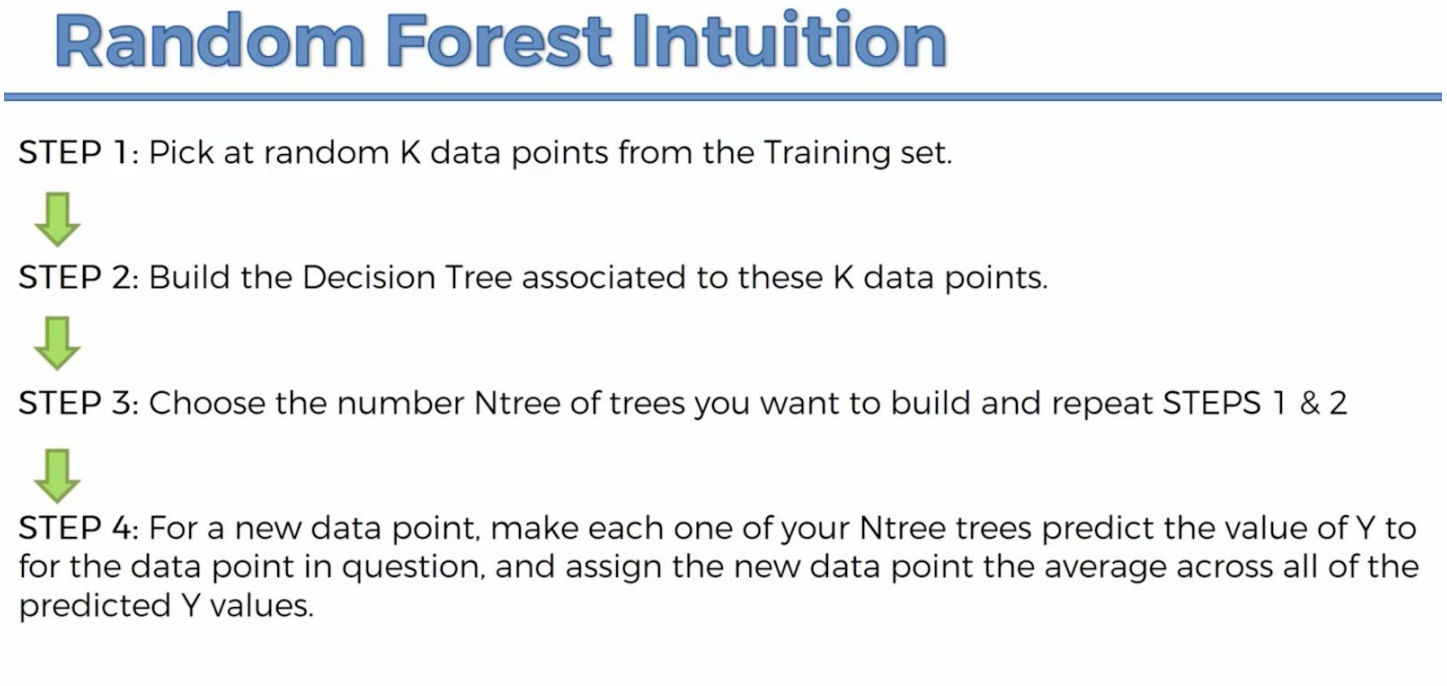

X1 < 20
Yes
No
X2 < 200
X2 < 170
Yes
No
Yes
No
X1 < 40
Yes
No


Y
y
y-coordinate
x1
x2
X2
Split 1
65.7
Split 3
200
170
1023
Split 2
300.5
-64.1
0.7
Split 4
20
40
X1

in green: average for y in a leaf

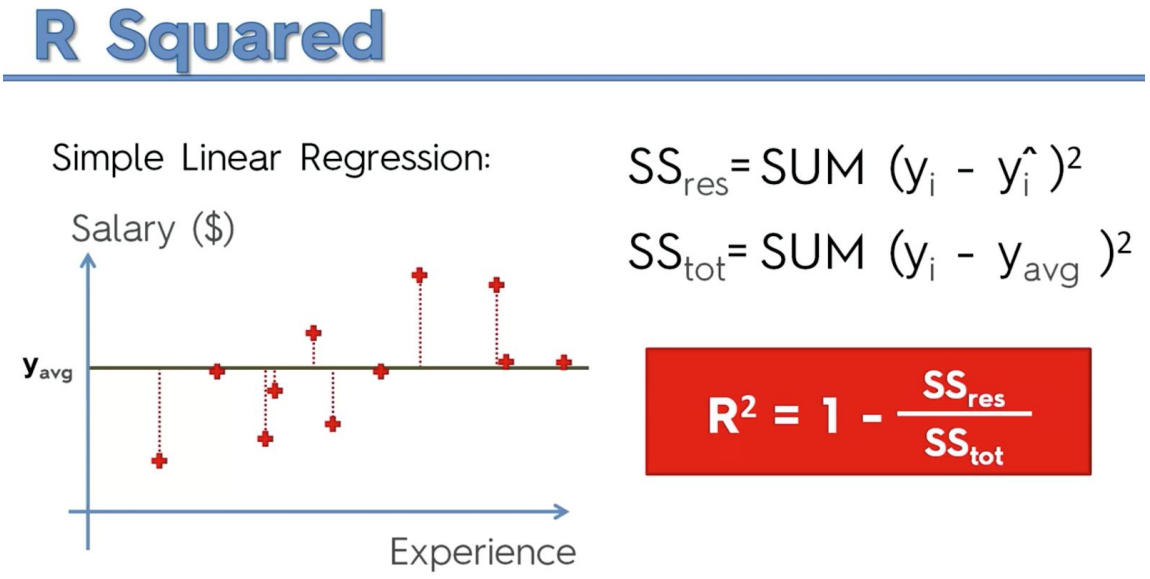


6) Random forest regression
type of **ensemble learning** - ensemble learning is when you take multiple algorithms or the same algorithm multiple times and you put them together to create something more powerful than the original



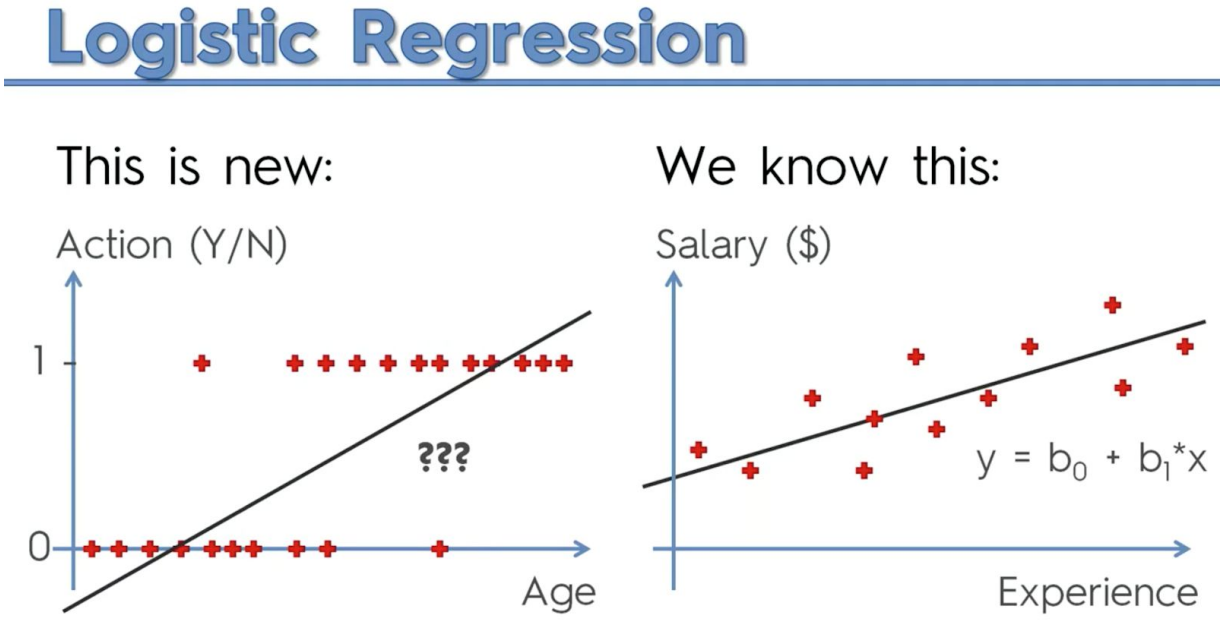
Evaluating Regression models performance may be done with R Squared
Have a look at Ordinary least square first.
SSres - sum of squared residuals it is the sum of the squares of residuals (deviations predicted from actual empirical values of data). It is a measure of the discrepancy between the data and an estimation model. A small RSS indicates a tight fit of the model to the data. It is used as an optimality criterion in parameter selection and model selection.
SStot - the total sum of squares

*there will always be a total sum of squared, so what you're trying to do in the regression is to fit the line to minimize the sum of squares of residuals (SSres), make it as small ap.
The average line shown on image is a kind of horizontal trendline, what you're trying to do by fitting the slope line and minimizing the SSres is you're trying to fit the best line, and R squared is telling you how good is your line, compared to an average line.
Thus, in ideal situation, R^2 = 1 (whhen SSres = 0), anyway the closer to 1, the better.
it may happen that R^2 is negative, when your line fits the data worse than the average line.
In case of more independent variables being added, we use **Adjusted R-Squared.**

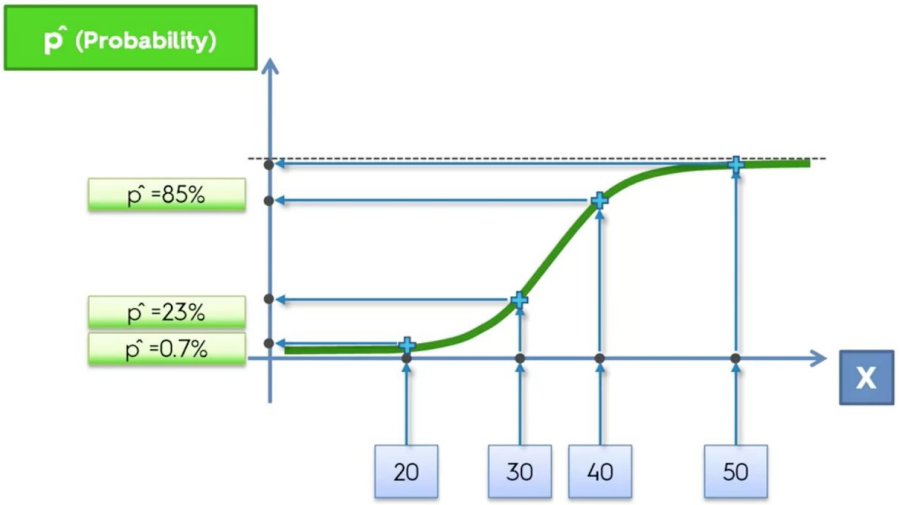


Classification

Logistic Regression
Similar to Simple Linear Regression



Logistic Regression



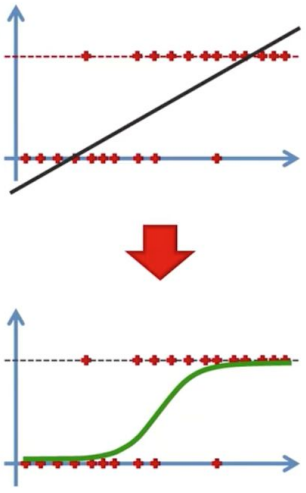
Logistic Regression

$$y = b_0 + b_1 * x$$

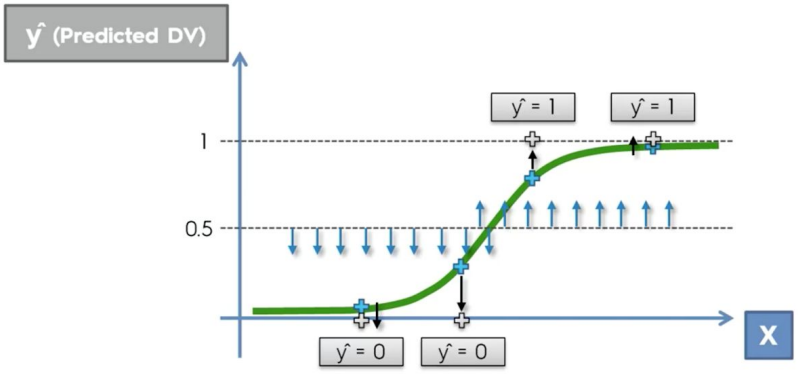
Sigmoid Function

$$p = \frac{1}{1 + e^{-y}}$$

$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 * x$$



Logistic Regression



Fin.

K-Nearest Neighbour

STEP 1: Choose the number K of neighbors

STEP 2: Take the K nearest neighbors of the new data point, according to the Euclidean distance

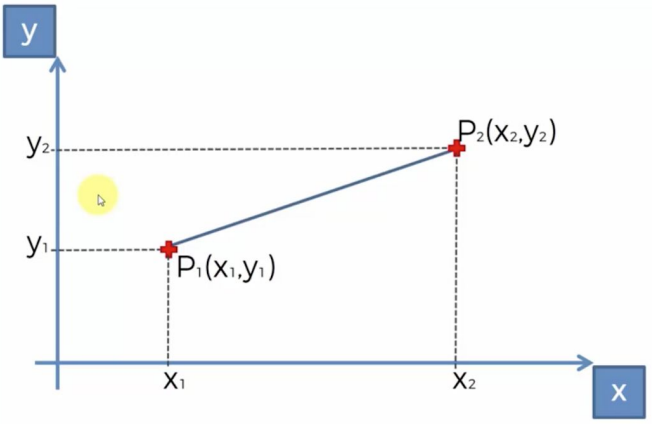
STEP 3: Among these K neighbors, count the number of data points in each category

STEP 4: Assign the new data point to the category where you counted the most neighbors

Your Model is Ready

STEP 1: Choose the number K of neighbors: K = 5

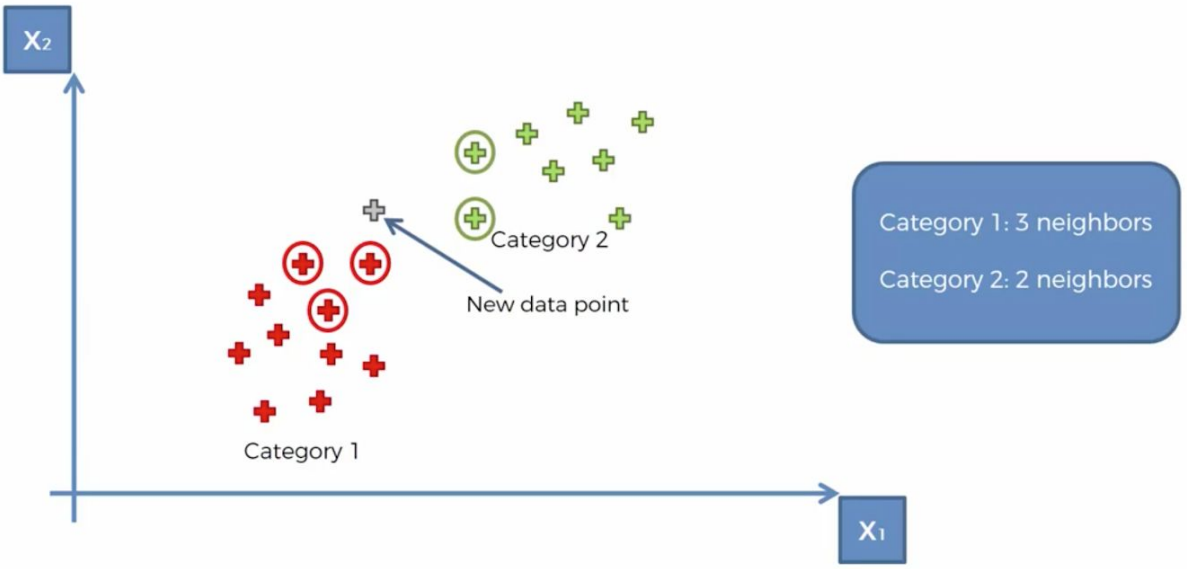
Euclidean Distance



Euclidean Distance between P_1 and $P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

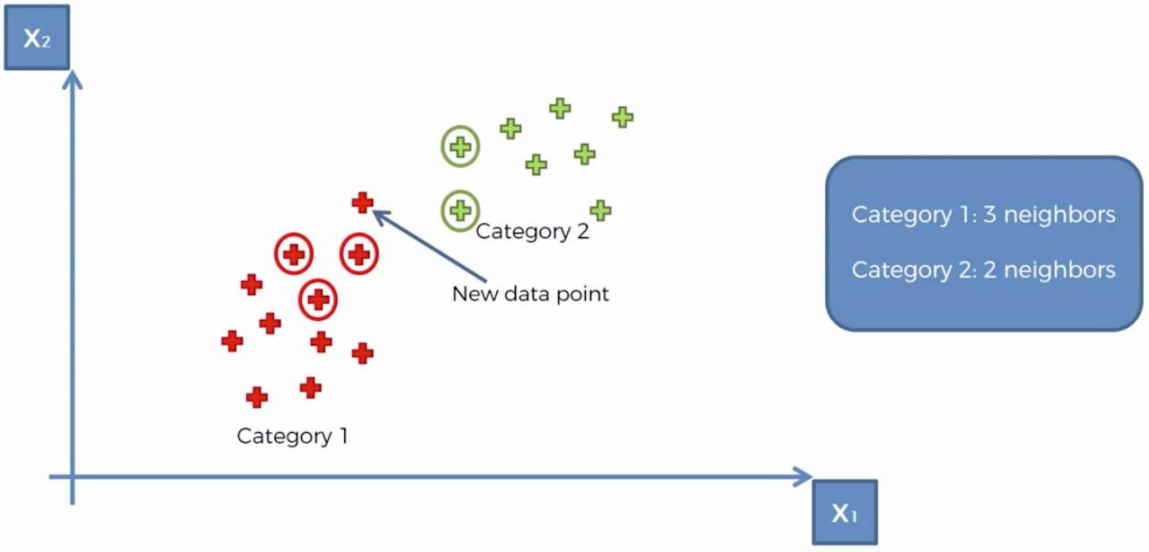
K-NN algorithm

STEP 3: Among these K neighbors, count the number of data points in each category



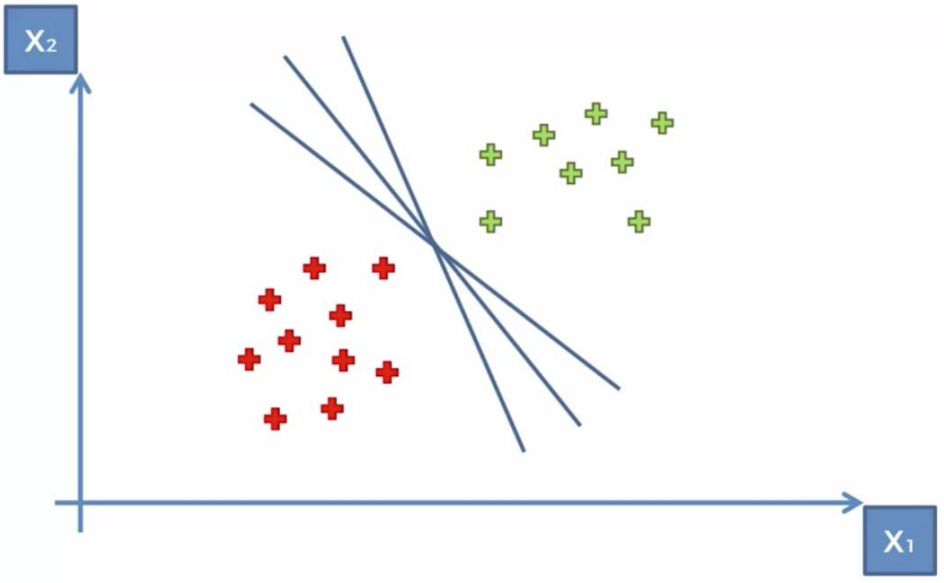
K-NN algorithm

STEP 4: Assign the new data point to the category where you counted the most neighbors

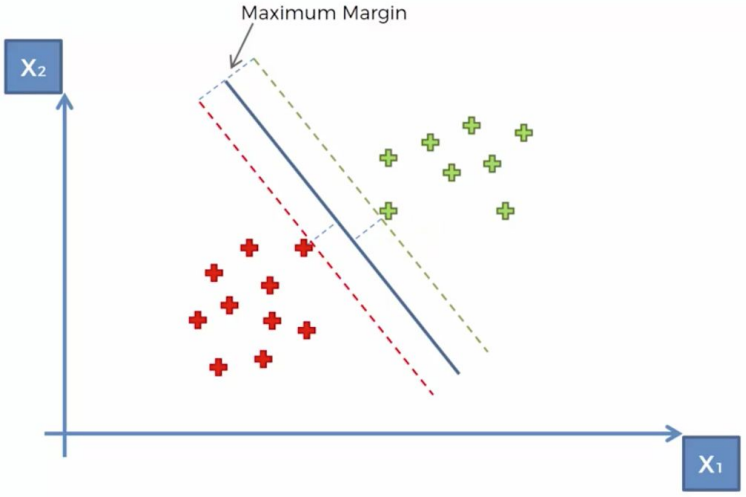


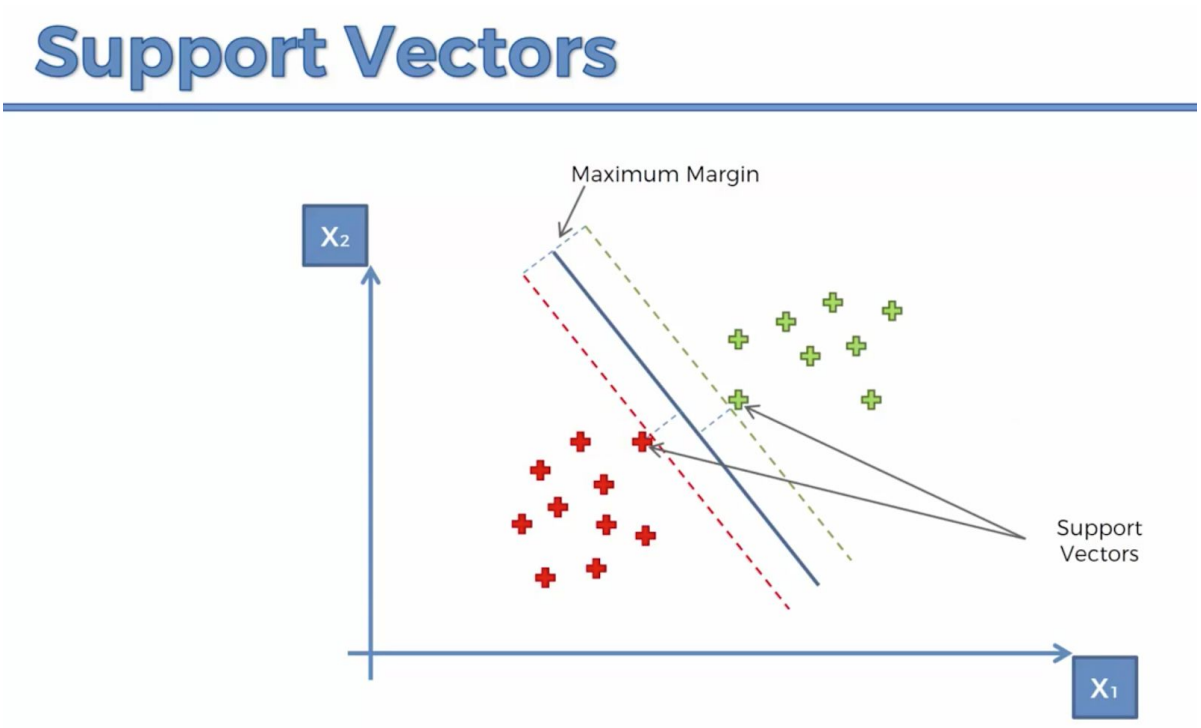
Support Vector Machine - find the best line - decision boundary - to separate space into classes

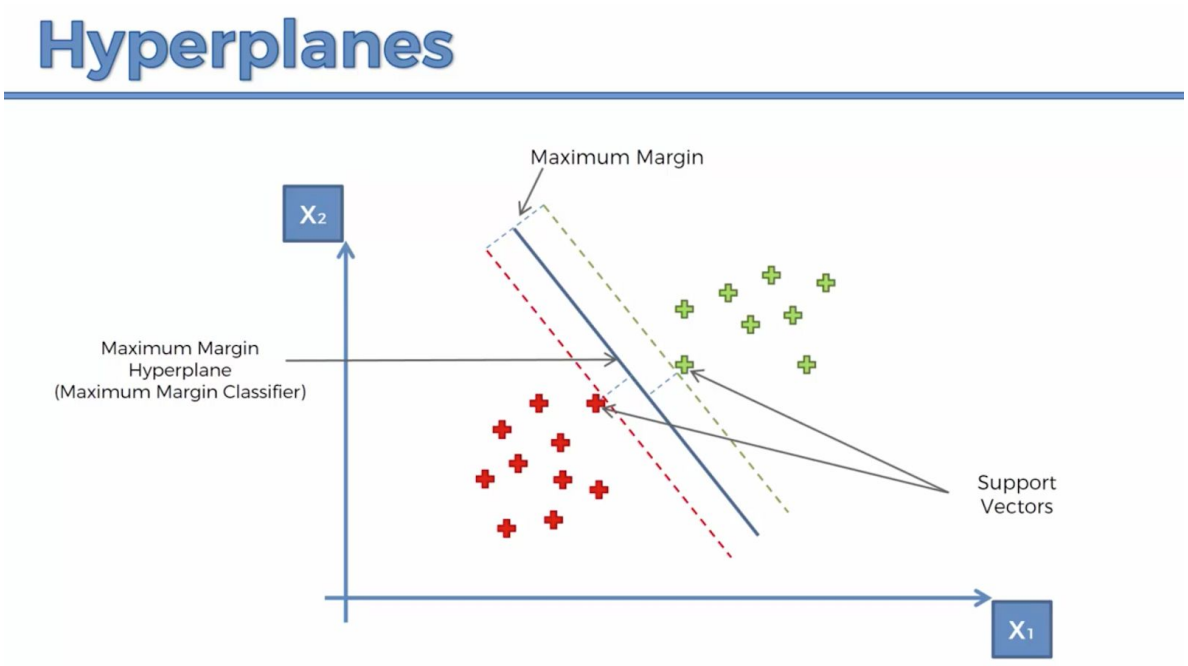
SVM



Maximum Margin



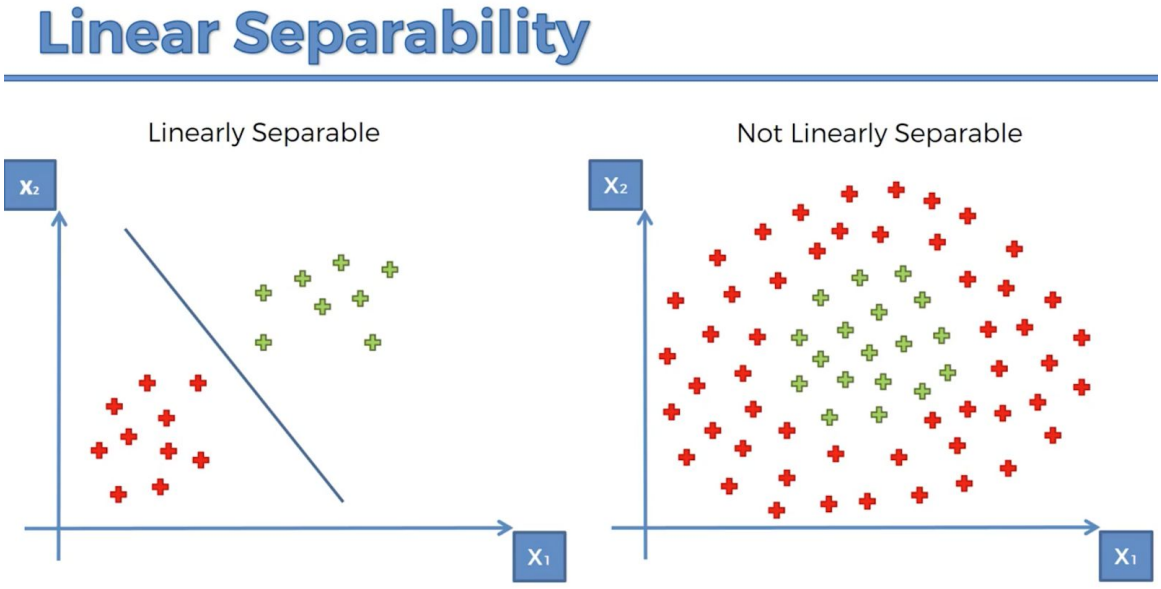




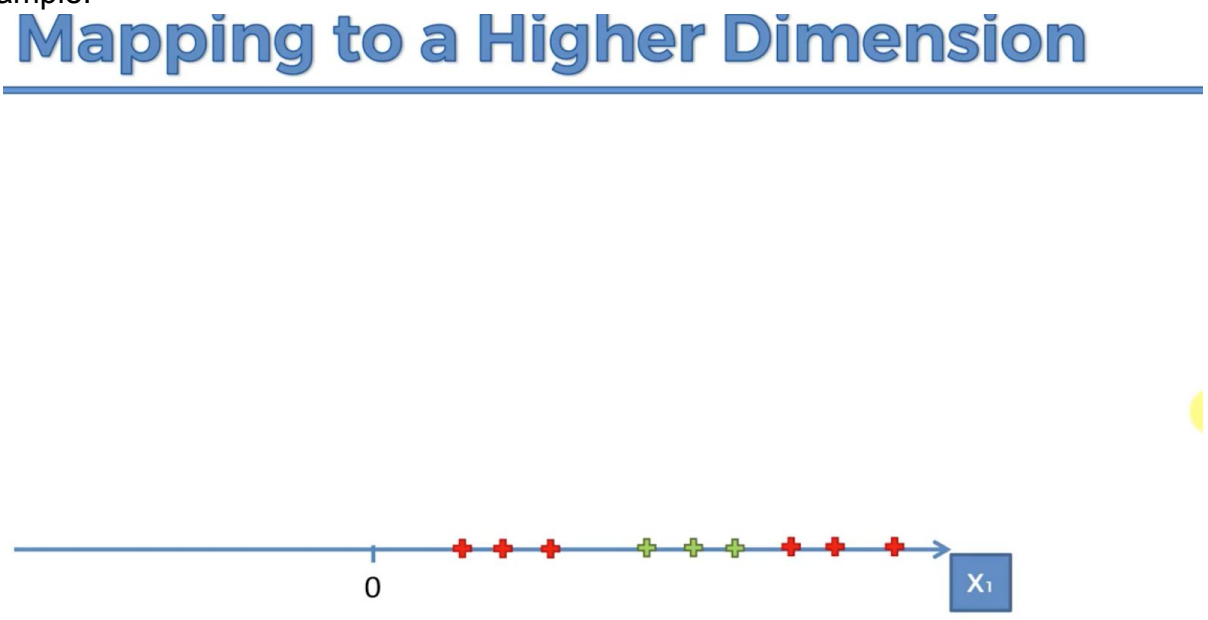
Why is SVM special and different?
The support vectors are grounding for placing a maximum margin and notice that these points are the closests to each other. Meaning if we had two sets one representing oranges(red +) and the other apples (green +), the most orange orange and most apple apple are the farthest from the line, whereas the chosen support vectors may actually be similar to each other like yellow apple and weird green orange. You may say that SVM is therefore risky type of algorithm as for analysis it uses cases that are very close to boundaries, and that's what makes it different and at times SVM performs much better than other algorithms.

Kernel SVM

SVM works with linearly separable data



This may be done by mapping to higher dimension or with so called Kernel trick
easy example:



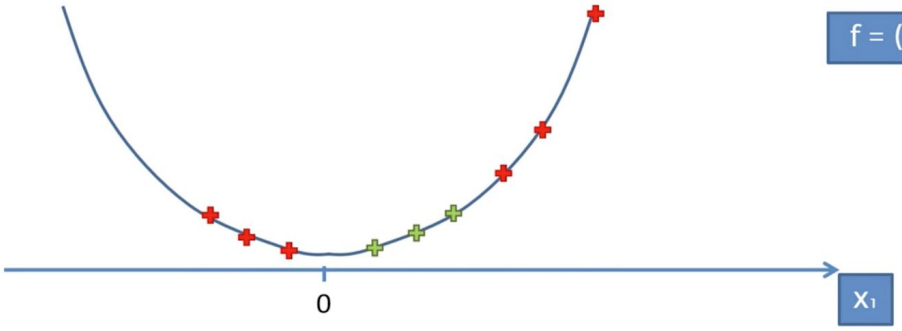
Mapping to a Higher Dimension

f = x - 5

Mapping to a Higher Dimension

f = x - 5

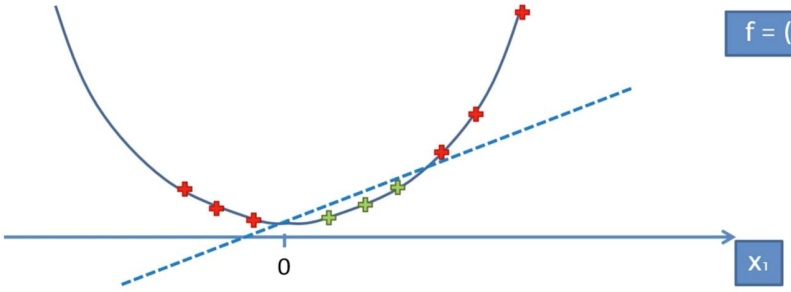
f = (x – 5)^2



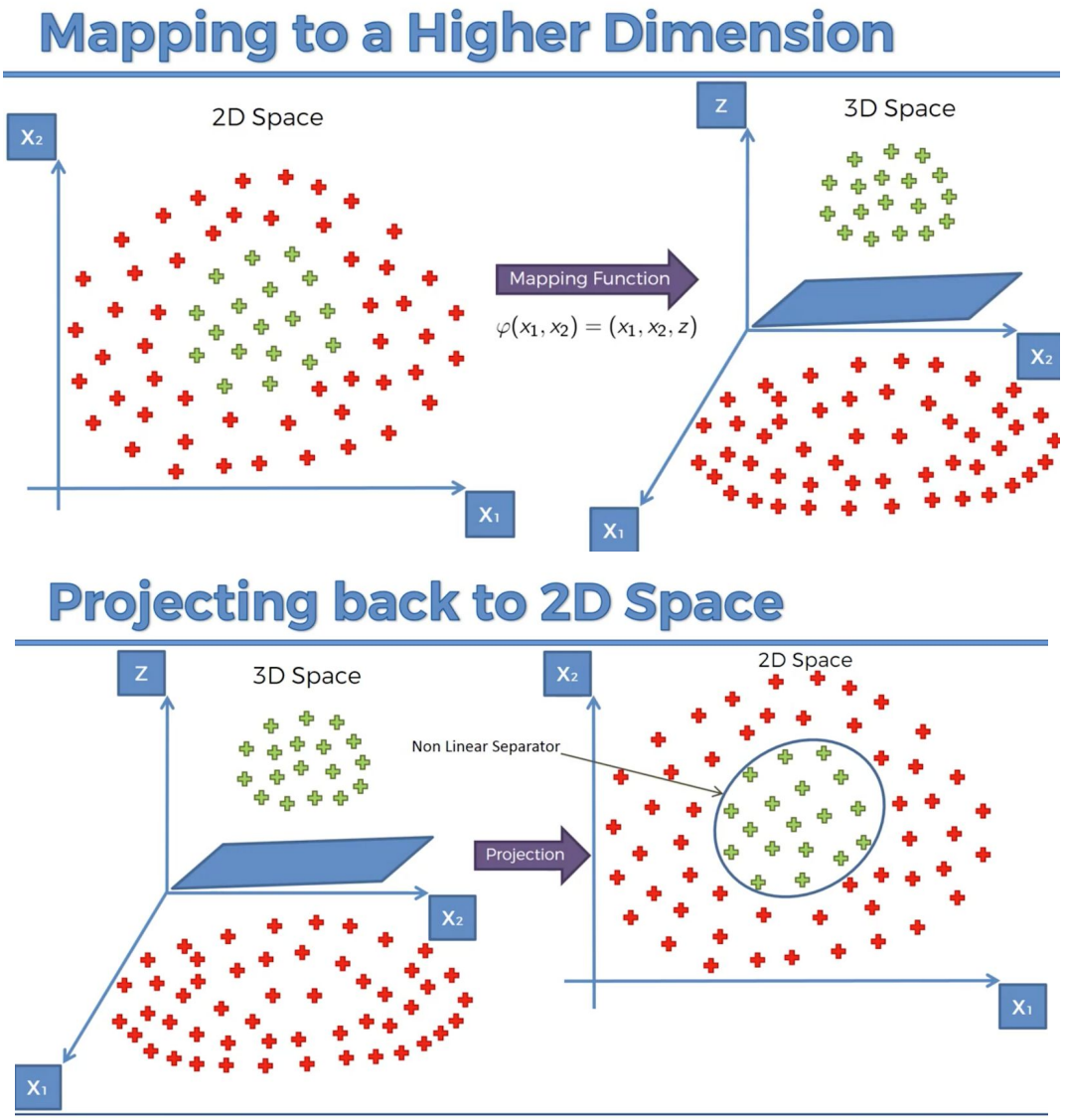
Mapping to a Higher Dimension

f = x - 5

f = (x – 5)^2



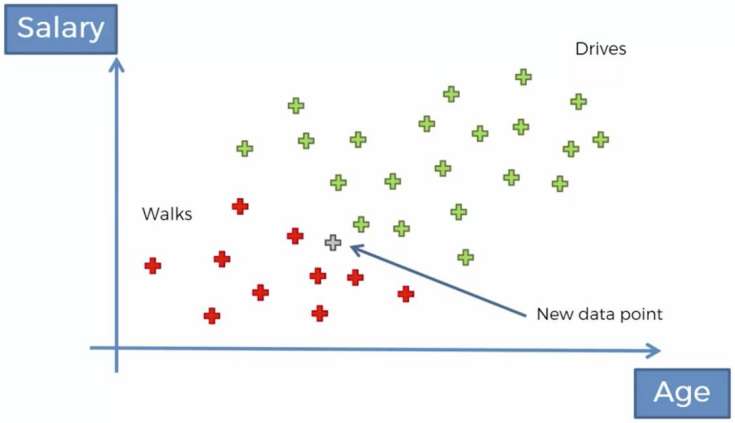
Other example



issue with that is it may be highly compute-intensive, so the other solution is Kernel trick
Więcej info: http://mlkernels.readthedocs.io/en/latest/kernels.html

Naive Bayes

Naïve Bayes



X - features, here age, salary

Step 1

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

$$P(Walks|X) = \frac{P(X|Walks) * P(Walks)}{P(X)}$$

#2 Marginal Likelihood

Step 2

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

$$P(Drives|X) = \frac{P(X|Drives) * P(Drives)}{P(X)}$$

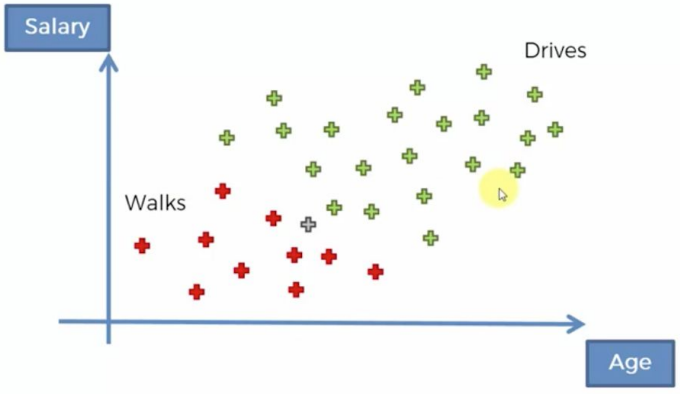
#2 Marginal Likelihood

Step 3

$$P(Walks|X)\ v.s.\ P(Drives|X)$$

example:

Naïve Bayes: Step 1

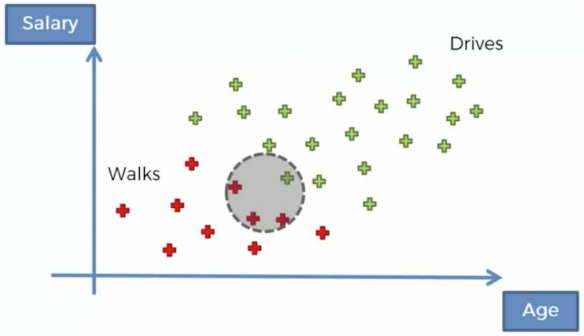


#1. P(Walks)

$$P(Walks) = \frac{Number\ of\ Walkers}{Total\ Observations}$$

$$P(Walks) = \frac{10}{30}$$

Naïve Bayes: Step 1



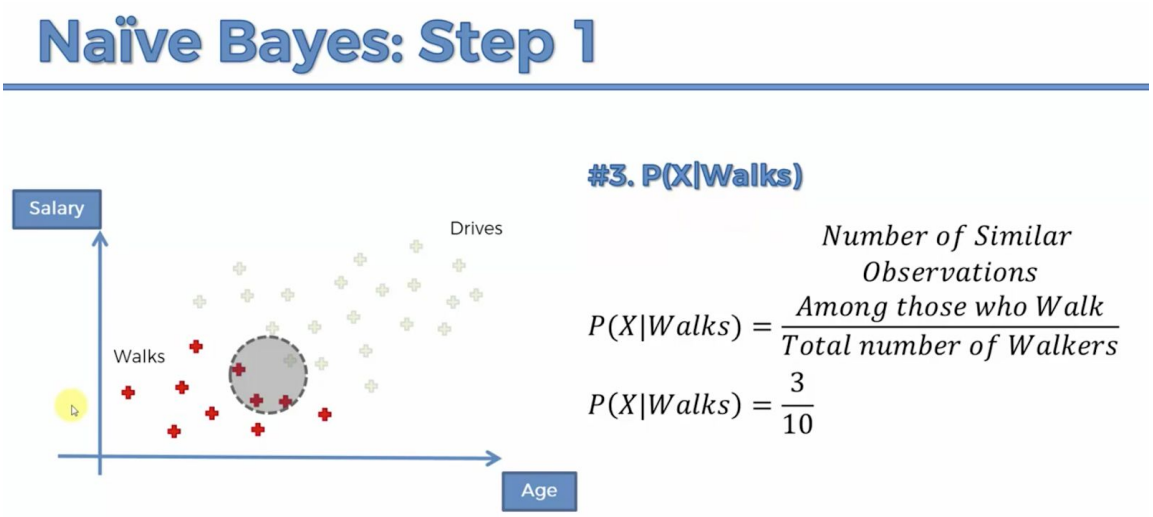
#2. P(X)

$$P(X) = \frac{Number\ of\ Similar\ Observations}{Total\ Observations}$$

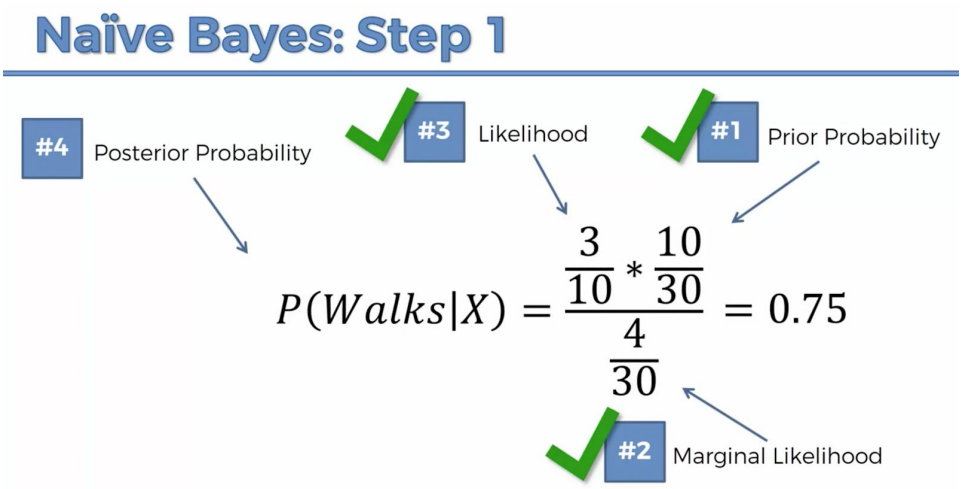
$$P(X) = \frac{4}{30}$$

you choose the radius yourself for the algorithm. Any observations in the circle are said to be similar to the new data point in terms of features (here similar salary and age)

P(X) tells us what is the likelihood that any new observation added to this data will fall inside the circle, more formally - what is the likelihood that randomly chosen data point from data set given that person walks exhibits features similar to the newly point added point



P(X|Walks) tells us what is the probability that randomly selected datapoint form a data set will be from the circle, given that person walks



Naïve Bayes: Step 2

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

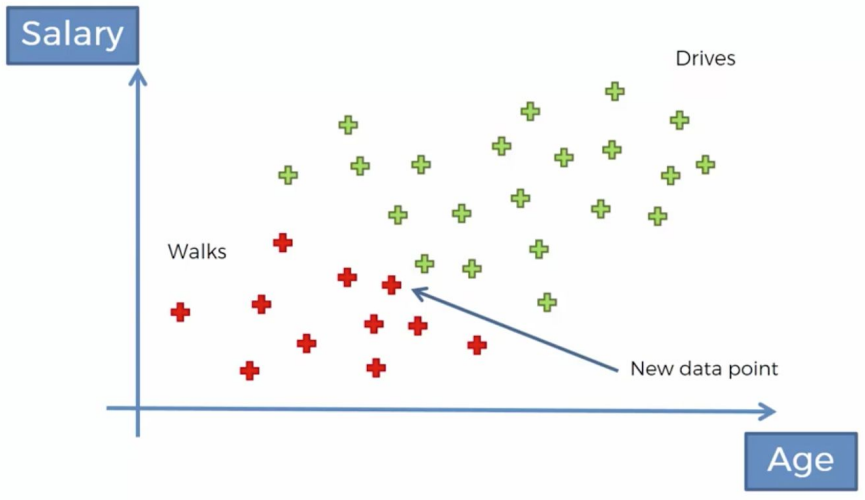
$$P(Drives|X) = \frac{\frac{1}{20} * \frac{20}{30}}{\frac{4}{30}} = 0.25$$

#2 Marginal Likelihood

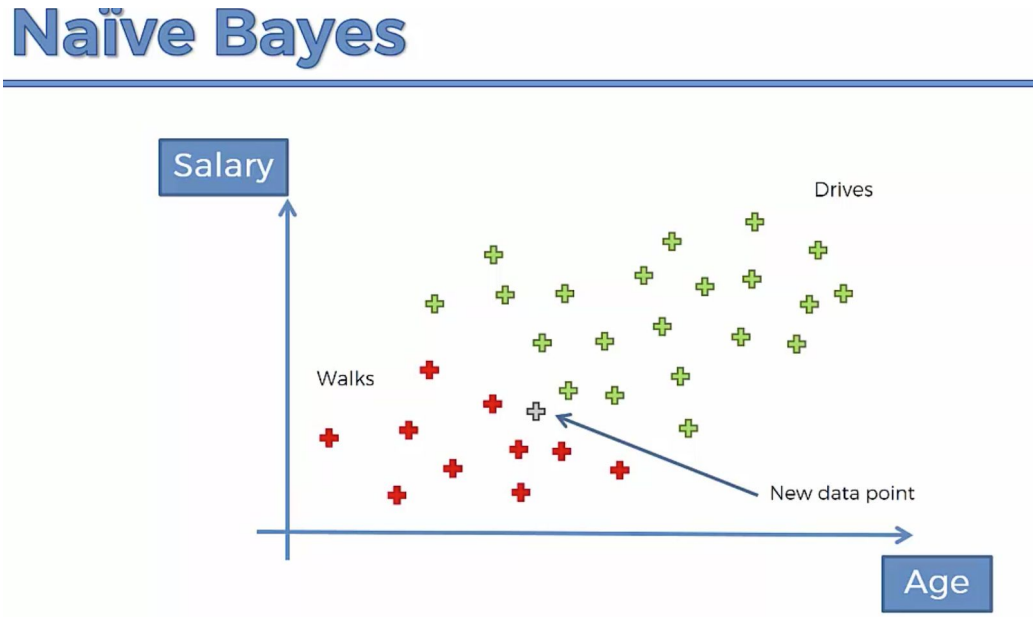
Step 3

$$0.75\ v.s.\ 0.25$$

Naïve Bayes



Why Naive?
Bayes Theorem requires some independence assumptions, and so does the algorithm, which does not have to be correct.



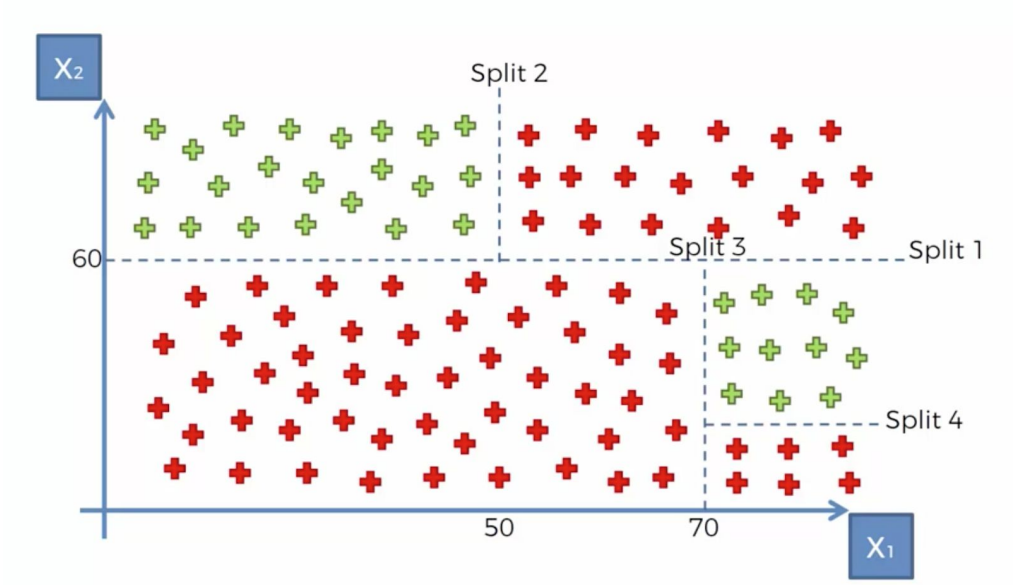
here, the bayes theorem requires that age and salary are independent, but if you think about it there's probably some dependence in getting older, gaining experience and earning more money. That's why it's called naive, as the variables are not completely independent but it's still applied and still performs well

Decision trees

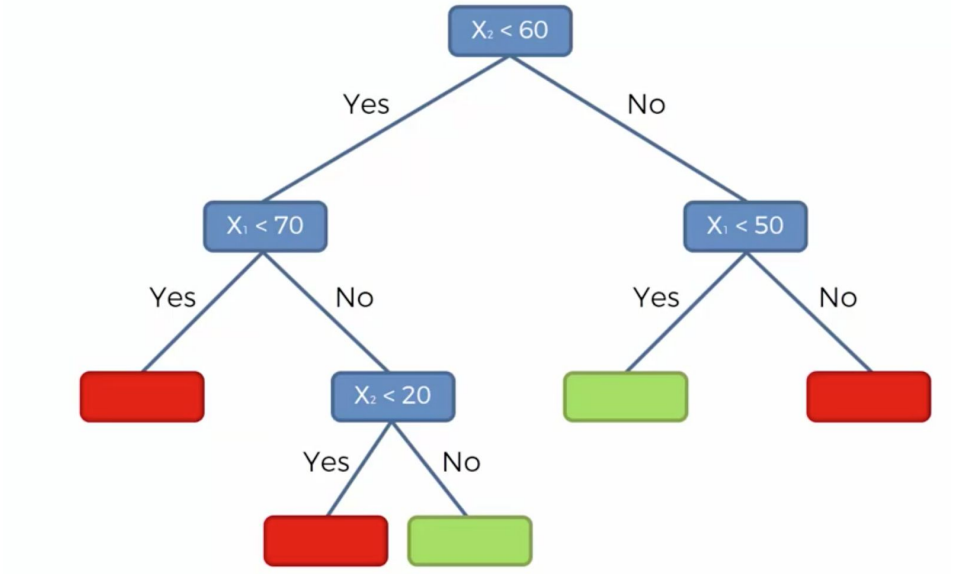
Firsty, the difference between classification trees and regression trees:

classification trees	regression trees
help you classify the data	help you predict outcomes that is the real number
Classification trees, as the name implies are used to separate the dataset into classes belonging to the response variable. Usually the response variable has two classes: Yes or No (1 or 0). If the target variable has *more* than 2 categories, then a variant of the algorithm, called C4.5, is used.	Regression trees are needed when the response variable is numeric or continuous. For example, the predicted price of a consumer good. Thus regression trees are applicable for *prediction* type of problems as opposed to *classification*.
Keep in mind that in either case, the predictors or independent variables may be categorical or numeric. It is the target variable that determines the type of decision tree needed.	
In a standard classification tree, the idea is to split the dataset based on homogeneity of data. Lets say for example we have two variables: age and weight that predict if a person is going to sign up for a gym membership or not. In our training data if it showed that 90% of the people who are older than 40 signed up, we split the	In a standard classification tree, the idea is to split the dataset based on homogeneity of data. Lets say for example we have two variables: age and weight that predict if a person is going to sign up for a gym membership or not. In our training data if it showed that 90% of the people who are older than 40 signed up, we split the

data here and age becomes a top node in the tree. We can almost say that this split has made the data "90% pure". Rigorous measures of impurity, based on computing proportion of the data that belong to a class, such as entropy or Gini index are used to quantify the homogeneity in Classification trees.	data here and age becomes a top node in the tree. We can almost say that this split has made the data "90% pure". Rigorous measures of impurity, based on computing proportion of the data that belong to a class, such as entropy or Gini index are used to quantify the homogeneity in Classification trees.
źródło: http://www.simafore.com/blog/bid/62482/2-main-differences-between-classification-and-regression-trees	



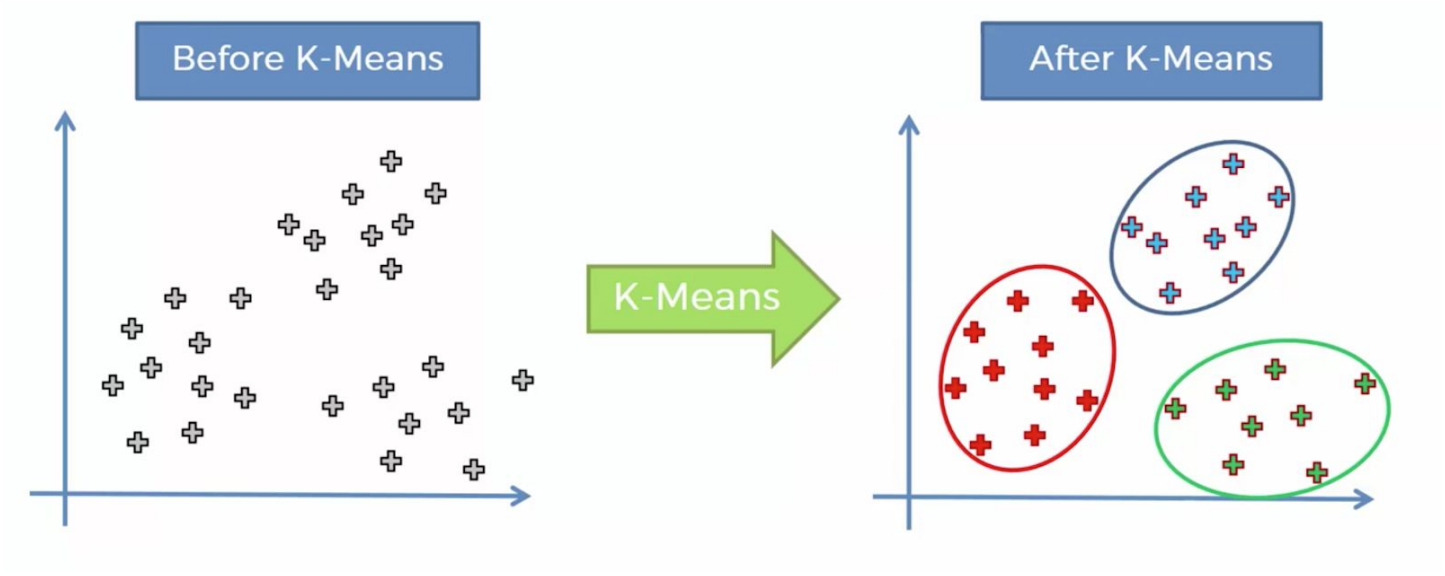
The splits are done in such way to maximize one category in a leaf (in huge simplification).

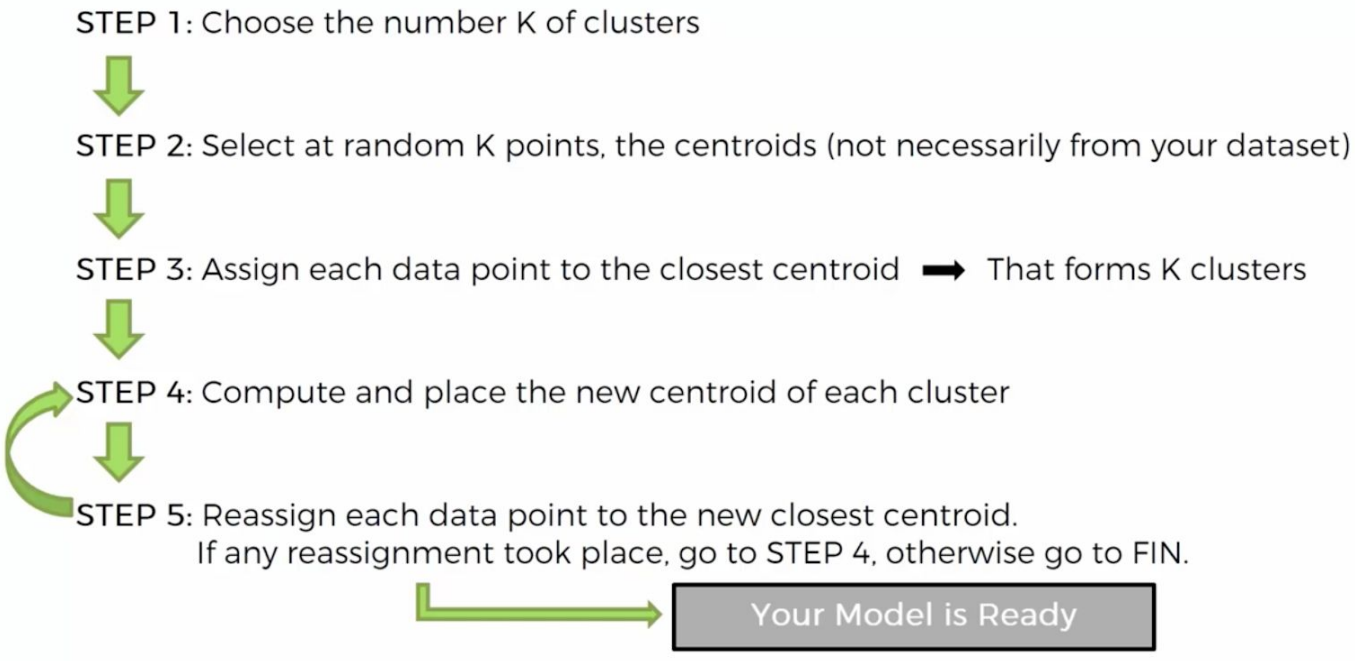


If the decision tree seems to be very complicated, instead of making additional splits, you may actually stop at any point and use the probabilities. Here, we could stop after split 2, saying anything above split 1 is according to split 2, and anything below split 1 is just red.

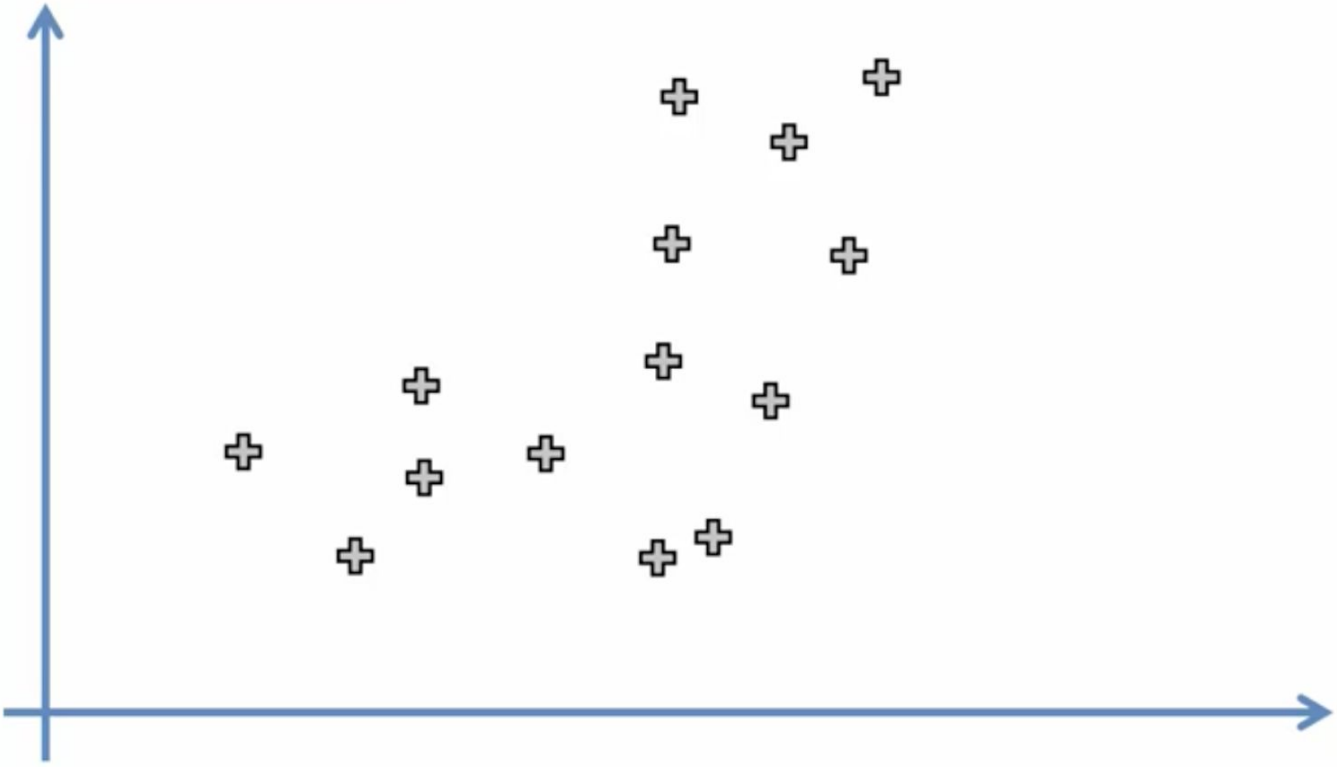
Clustering

K-Means

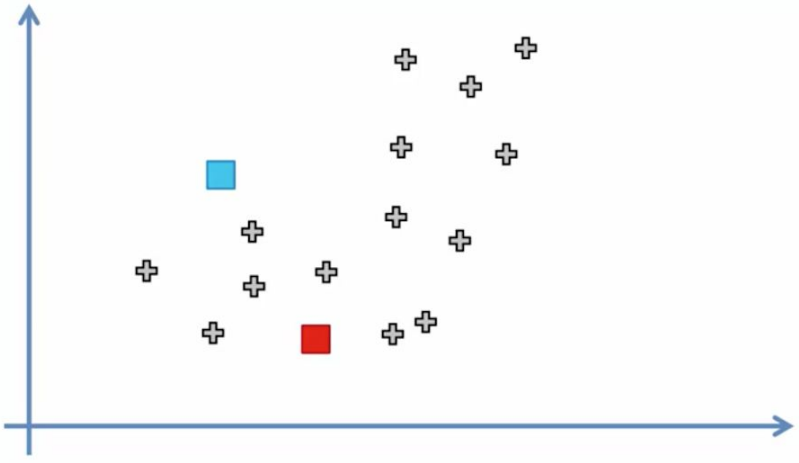




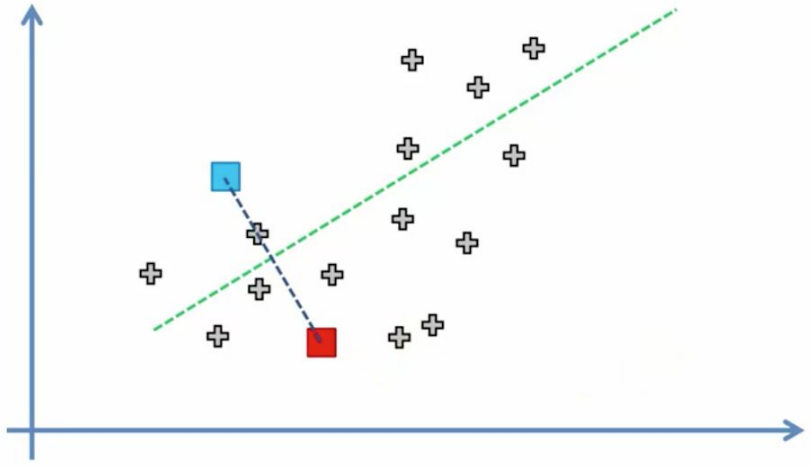
STEP 1: Choose the number K of clusters: K = 2



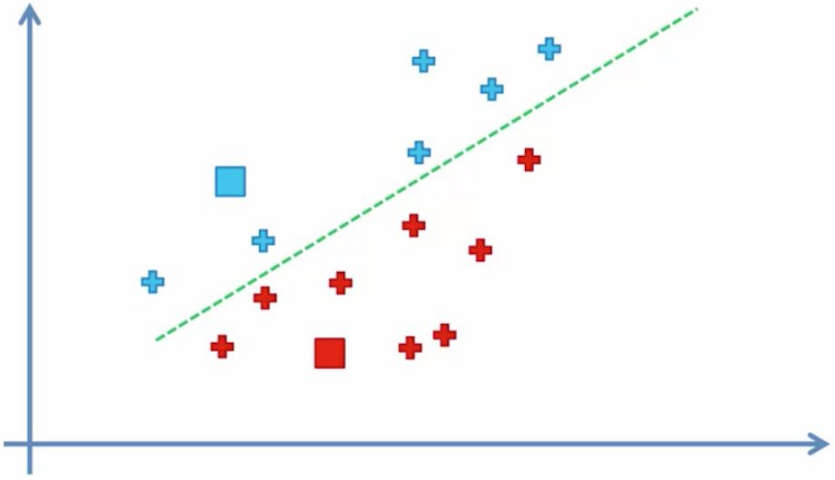
STEP 2: Select at random K points, the centroids (not necessarily from your dataset)



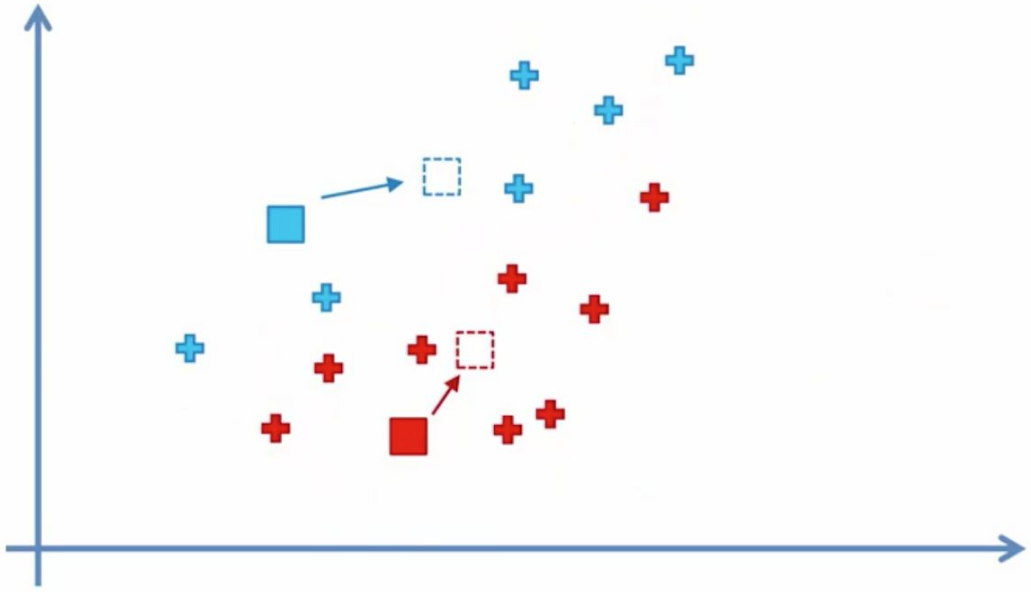
STEP 3: Assign each data point to the closest centroid ➞ That forms K clusters



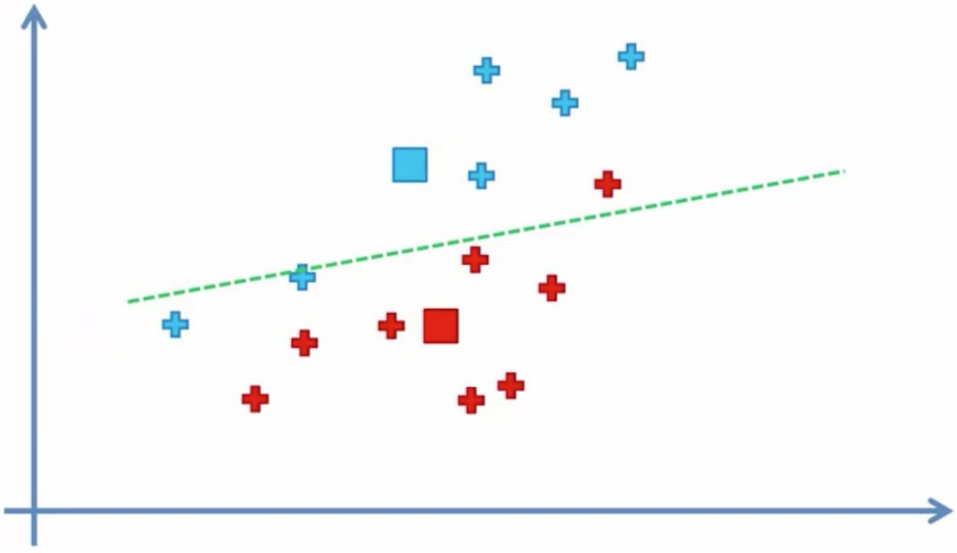
STEP 3: Assign each data point to the closest centroid ➞ That forms K clusters



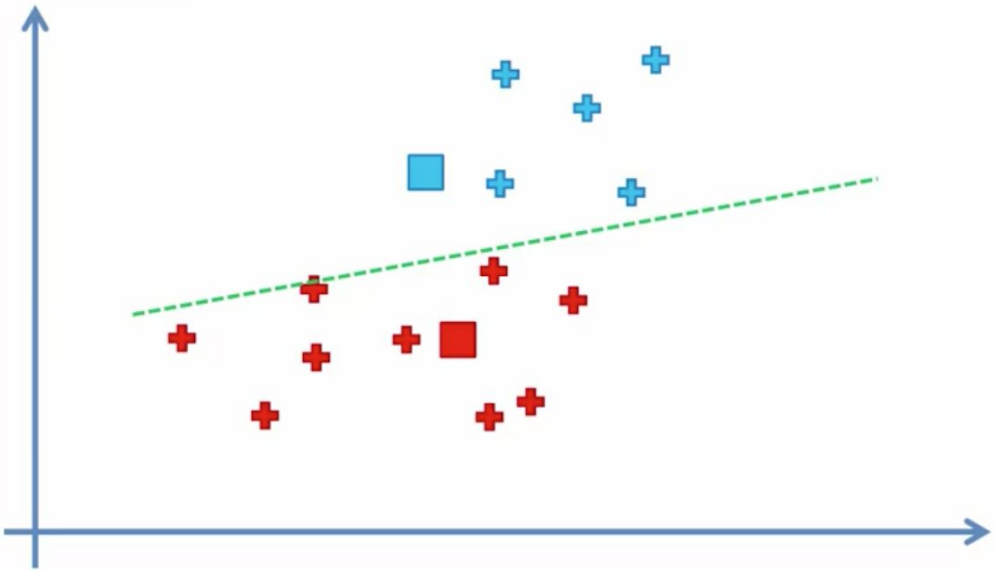
STEP 4: Compute and place the new centroid of each cluster



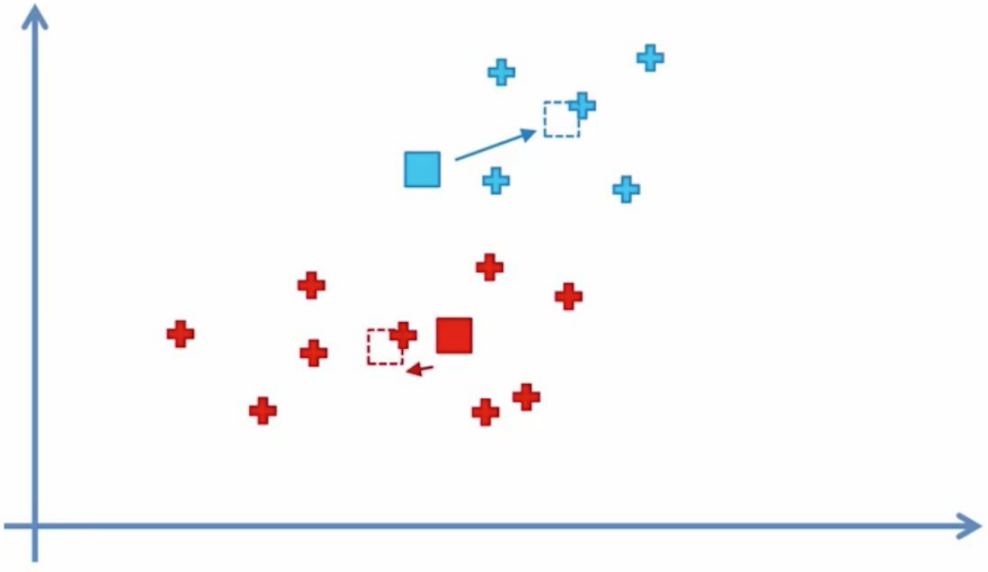
STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



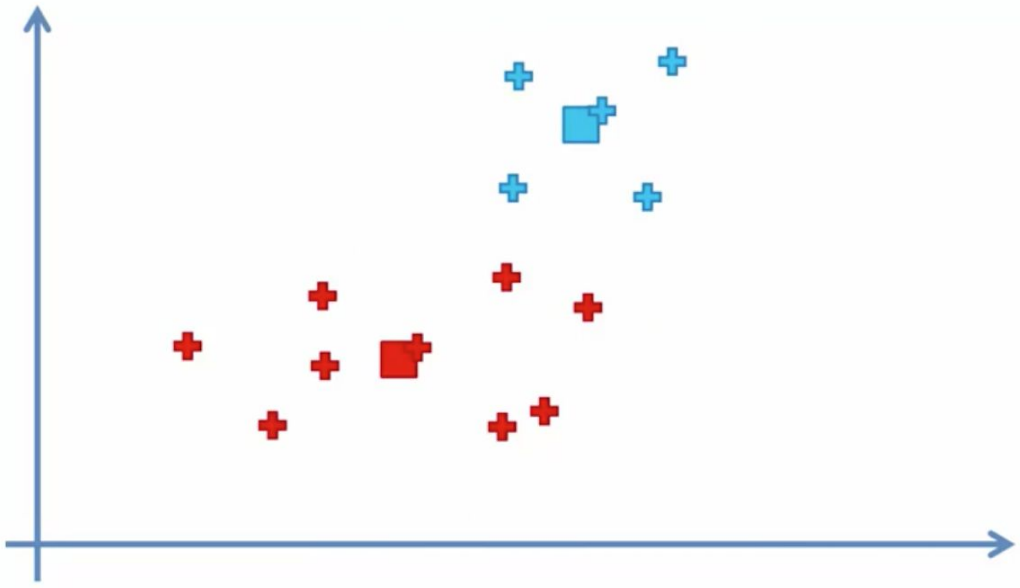
STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



STEP 4: Compute and place the new centroid of each cluster

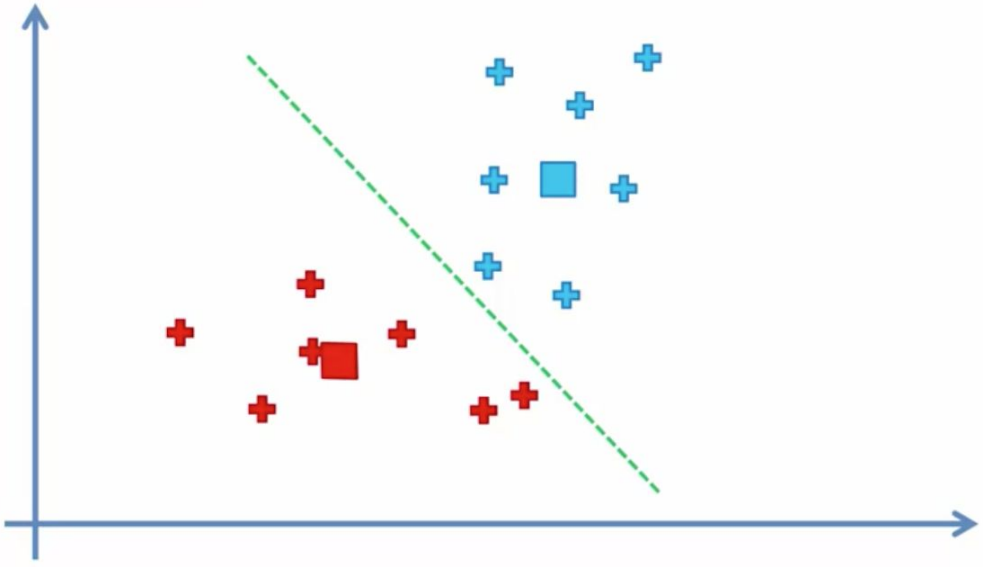


STEP 4: Compute and place the new centroid of each cluster

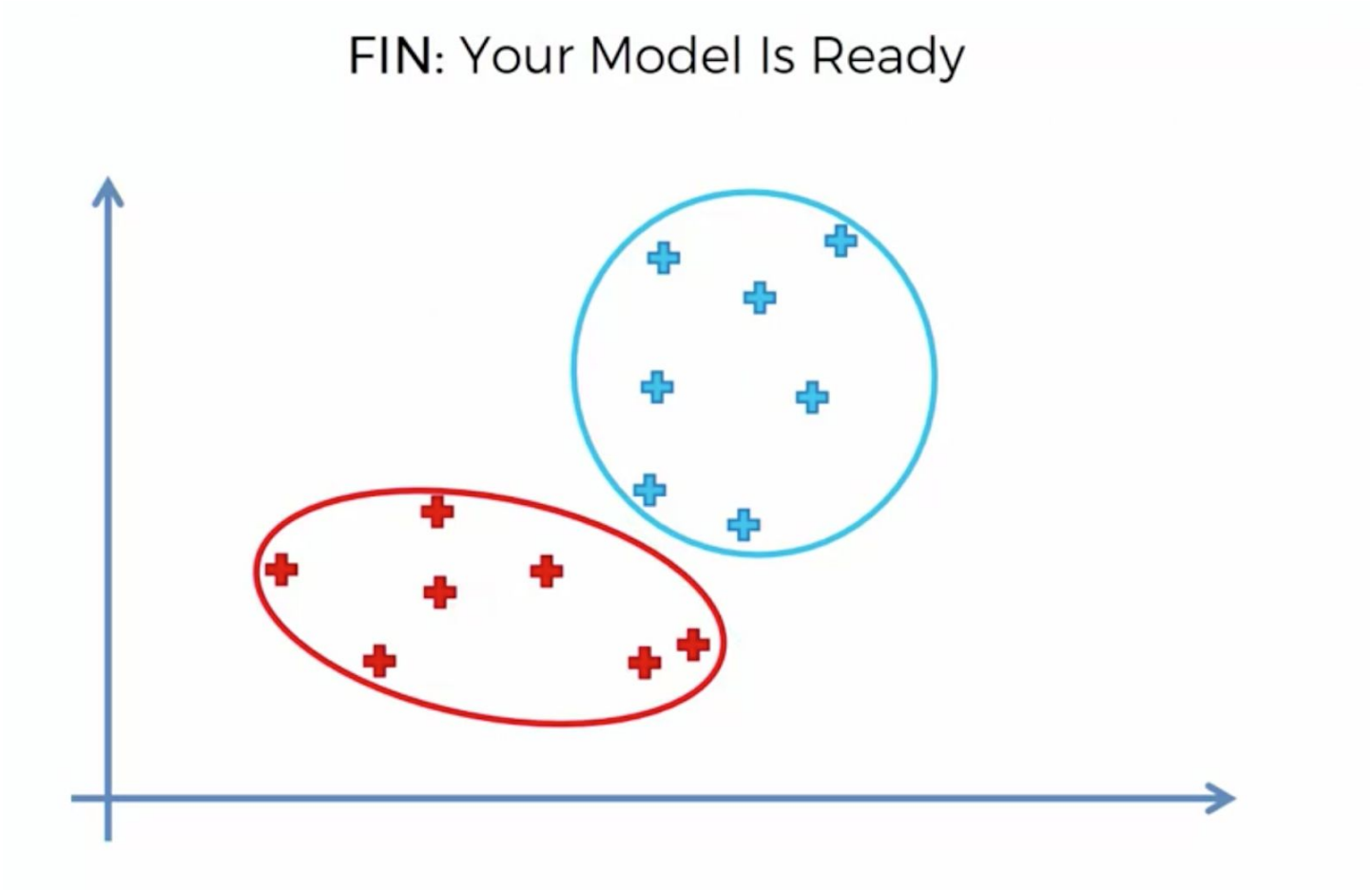


and so on until

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



Ran



Random initialization trap and K-means++ : https://en.wikipedia.org/wiki/K-means%2B%2B
How many K clusters to choose at the beginning may be found with usage of WCSS (within-cluster sum of squares)

Hierarchical Clustering

There are two types - agglomerative and divisive

Agglomerative HC

STEP 1: Make each data point a single-point cluster ➡ That forms N clusters

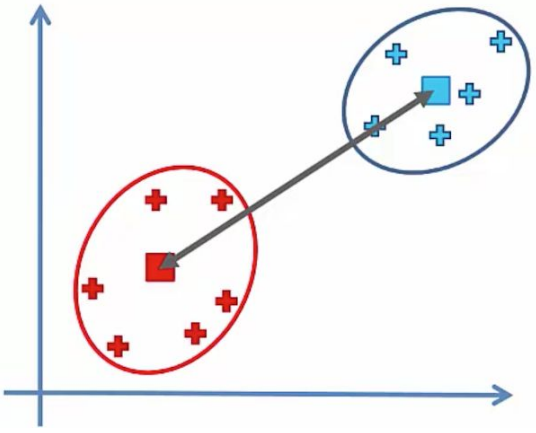
STEP 2: Take the two closest data points and make them one cluster ➡ That forms N-1 clusters

STEP 3: Take the two closest clusters and make them one cluster ➡ That forms N - 2 clusters

STEP 4: Repeat STEP 3 until there is only one cluster

FIN

Distance Between Clusters



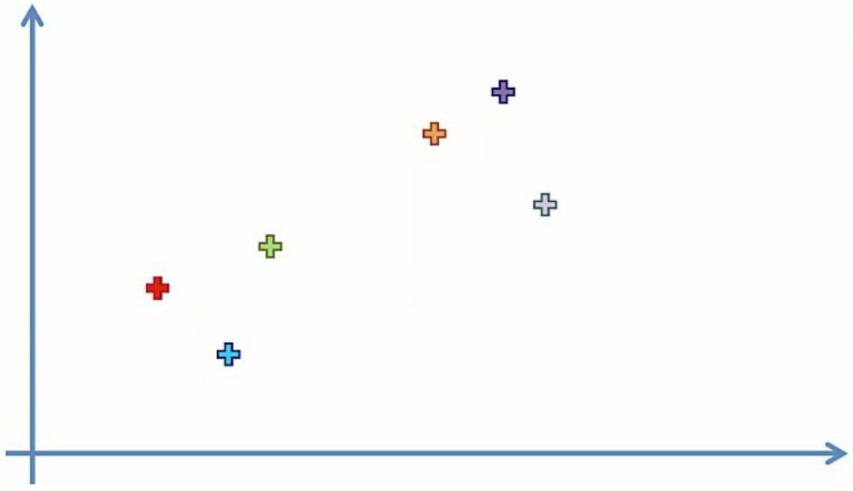
Distance Between Two Clusters:

- Option 1: Closest Points
- Option 2: Furthest Points
- Option 3: Average Distance
- Option 4: Distance Between Centroids

It’s very important to define what do you mean by ‘closest clusters’ that is what distance are you tkaing.

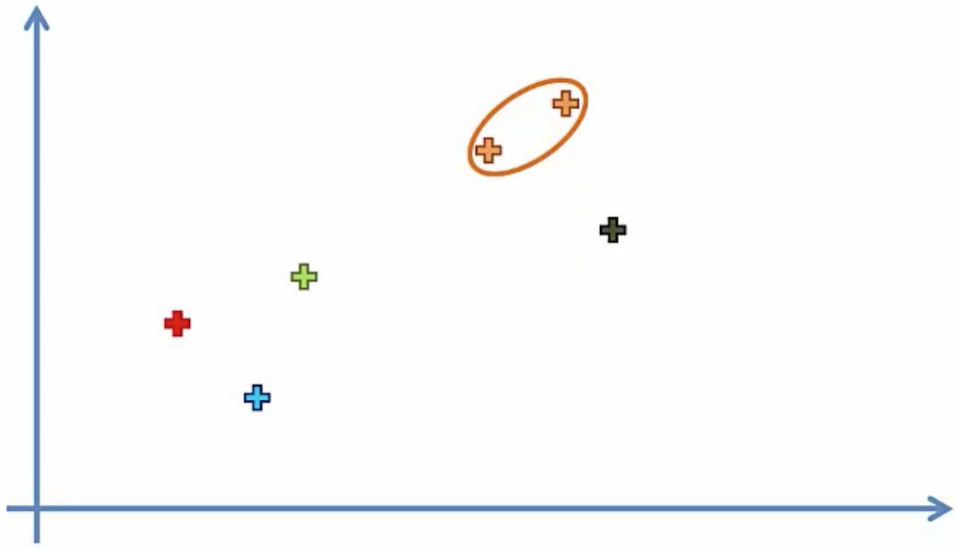
Consider the following dataset of N = 6 data points

STEP 1: Make each data point a single-point cluster ➡ That forms 6 clusters



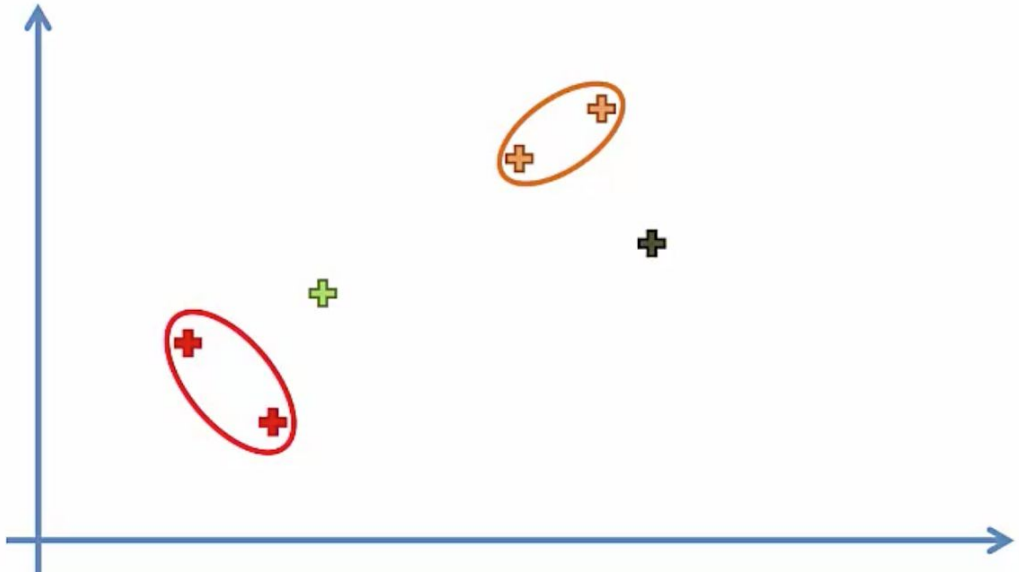
STEP 2: Take the two closest data points and make them one cluster

➞ That forms 5 clusters

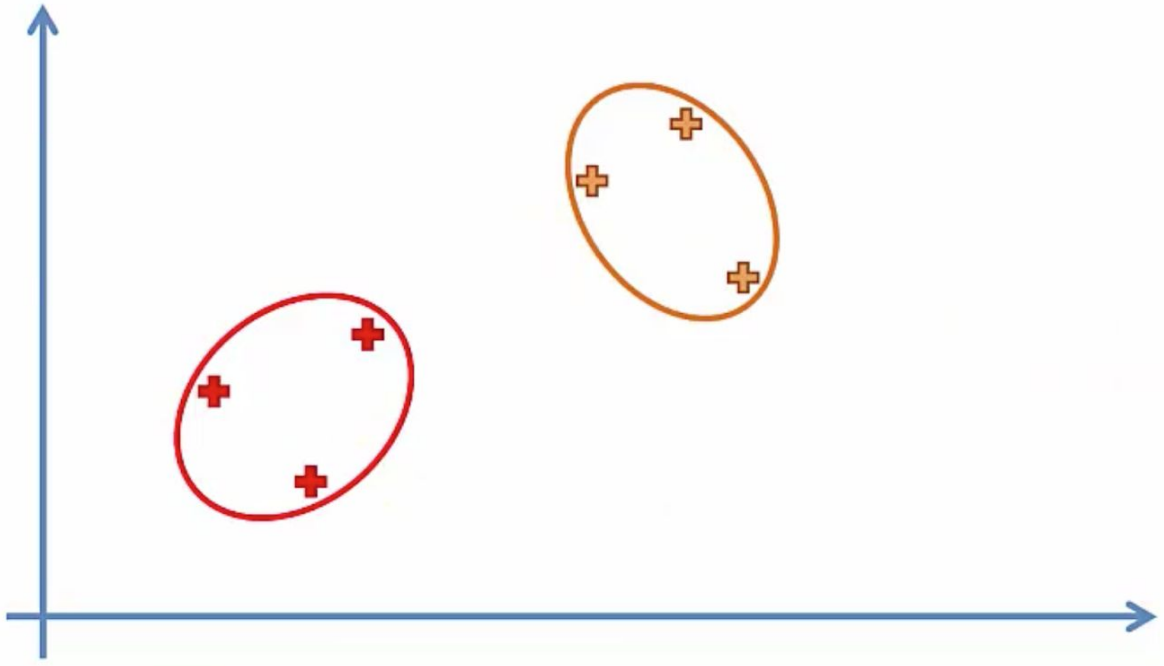


STEP 3: Take the two closest clusters and make them one cluster

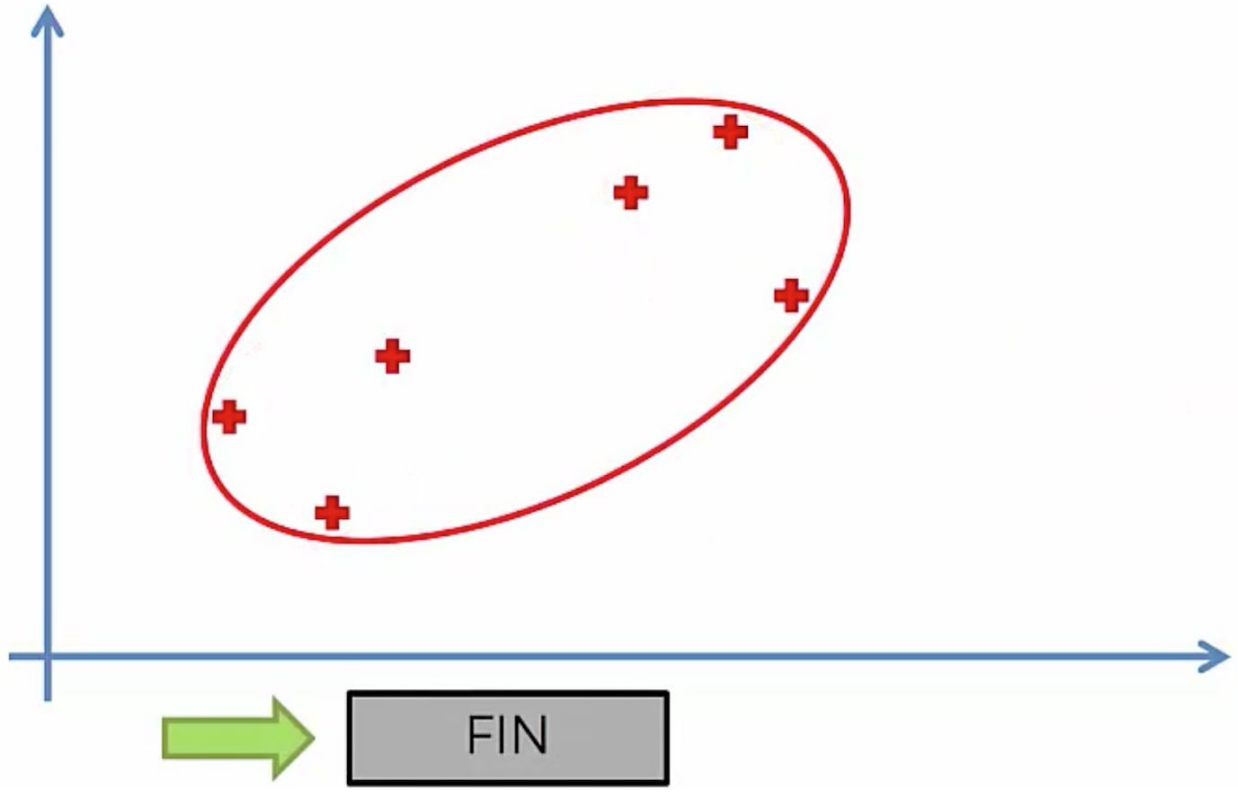
➞ That forms 4 clusters



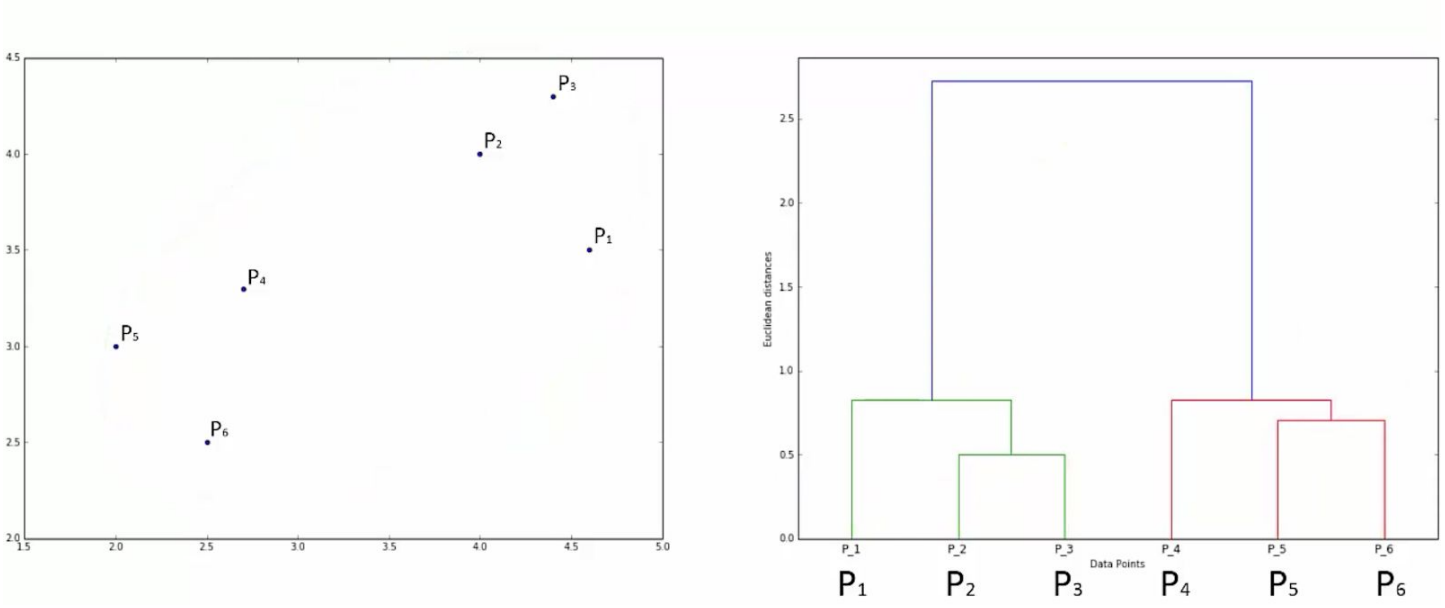
STEP 4: Repeat STEP 3 until there is only one cluster



STEP 4: Repeat STEP 3 until there is only one cluster



The memory of hierarchical clustering is saved in dendograms, where each bar represents a cluster created, and the height of such bar is equal to dissimilarity of points in the cluster, that is distnace between them.
Proper number of clusters may be found with setting a threshold in dendagram



Więcej info:
https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice - how to choose algorithms
http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/ - how are they grouped - learning style / similarity of form or function

23. Artificial neural networks.

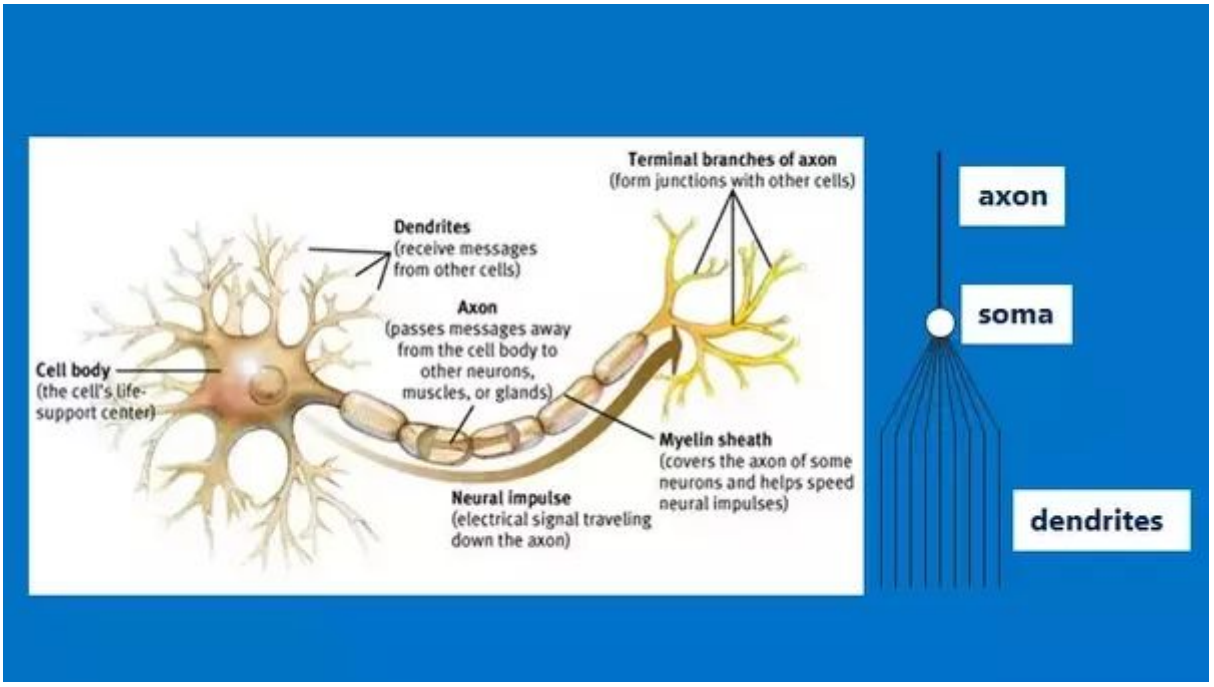
What's deep learning?
Deep learning (also known as **deep structured learning** or **hierarchical learning**) is the application to learning tasks of artificial neural networks (ANNs) that contain more than one hidden layers. Deep learning is part of a broader family of machine learning methods based on learning data representations, as opposed to task specific algorithms. Learning can be supervised, partially supervised or unsupervised.

Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation and bioinformatics where they produced results comparable to and in some cases superior to human experts

It's basically good in finding patterns.

The basic premises of deep learning start from the brain. It wants to emulate in some sense exactly what the human brain does. Human brain comprises of neural networks.

Here is a schematic of a neuron, and a pictorial description of it. The units of a neuron are axon, soma and dandrites. There are more, but lets stick to these basic ones for now.

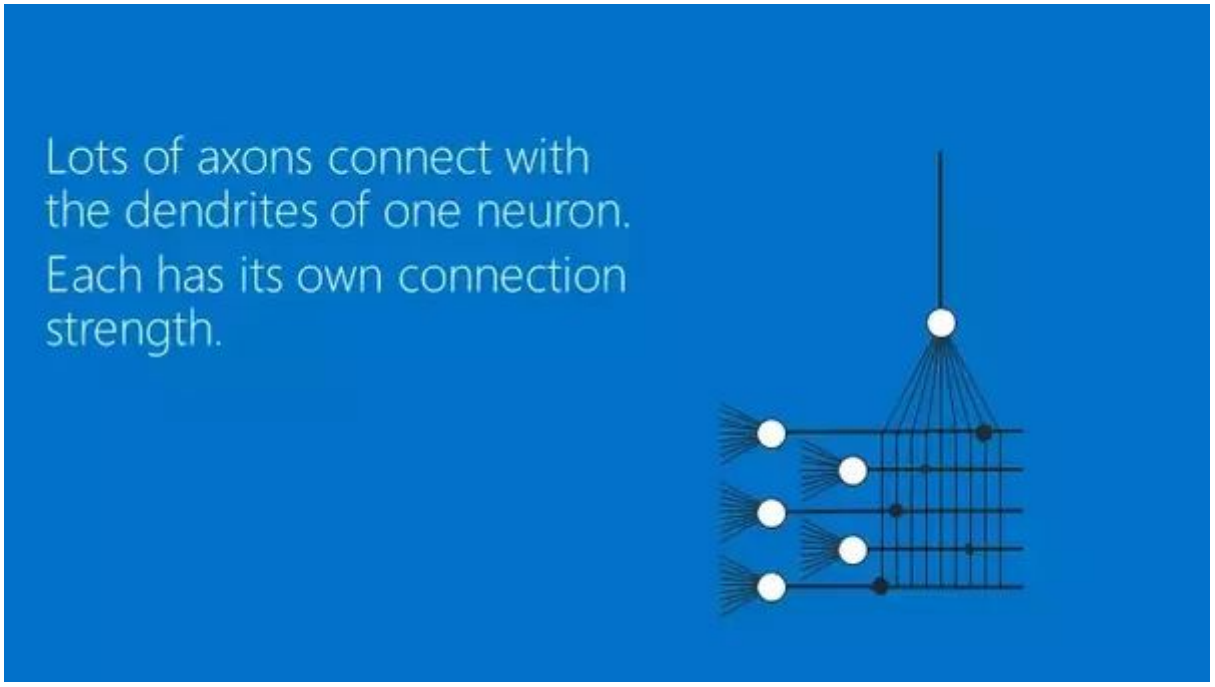


The soma collect information from the dendrites and passes it over to an axon.

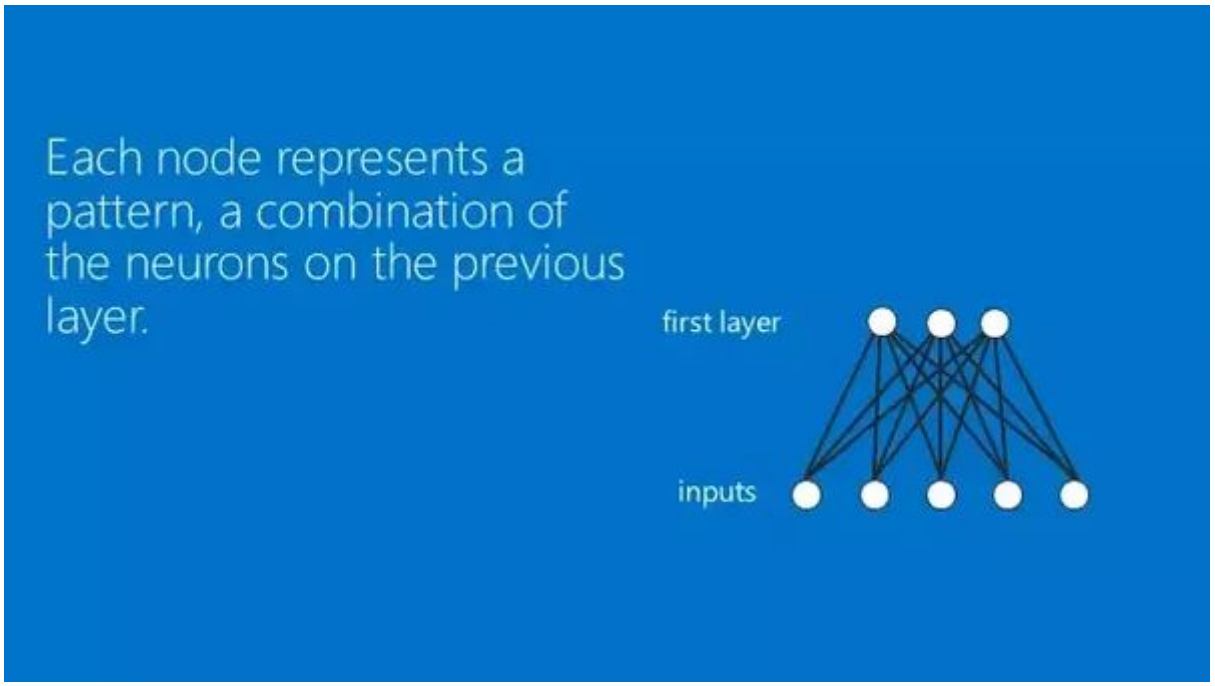
The dendrites can connect to the axons of next layer neurons with a process called synapse. This shows how synapse happens. Its more like an electrical activation.

To think of this pictorially, one can draw a schematic like the one shown in the below.

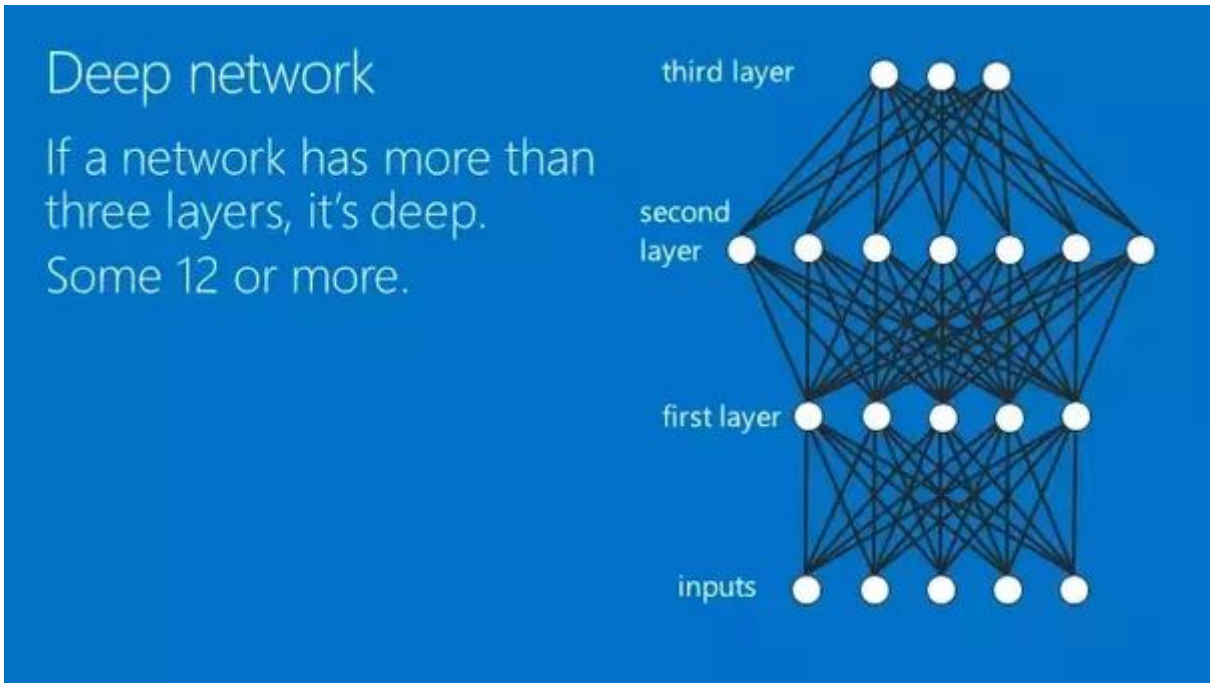
There can be several neurons connected by the process of synapse, and they can form multiple layers.



One can form multiple nodes, and multiple layers. There is really no theoretical limits to how many layers can be formed.



A neural network is convoluted, when they are mixed up and sending many edges in the subsequent layers, and it is called deep, when there are at least ~ 3 or more layers of them.

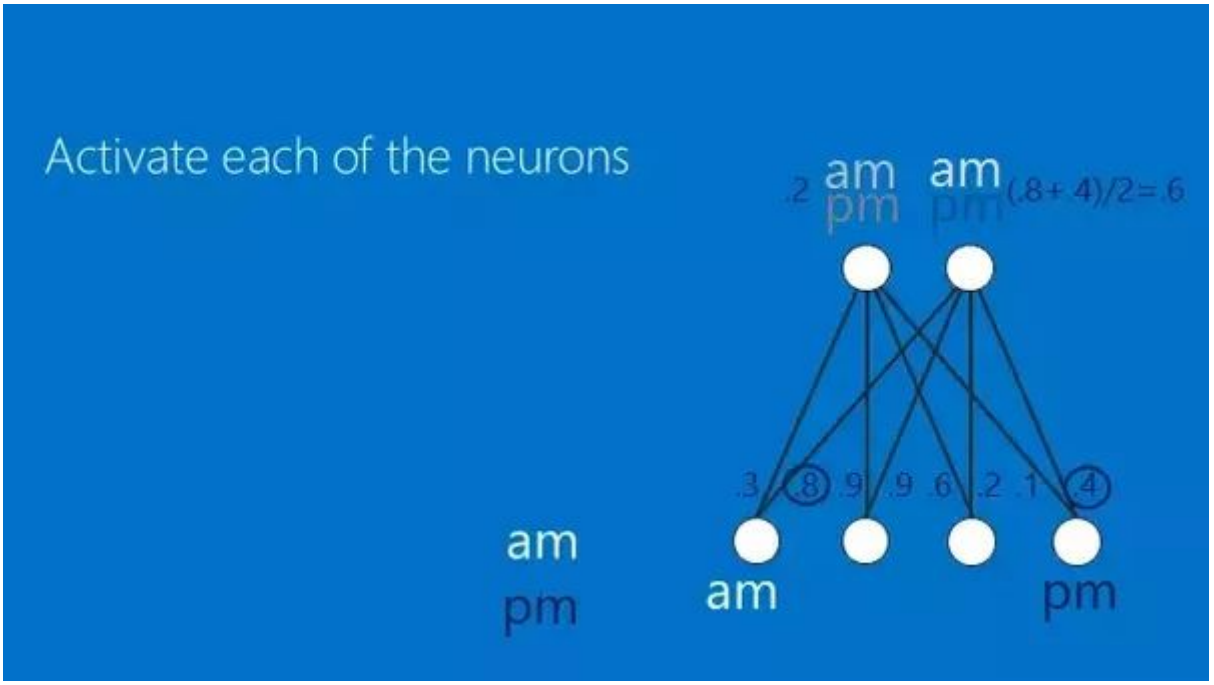


It can be seriously deep. People have created up to ~ 128 layers.

Now here is how the deep learning process looks like.

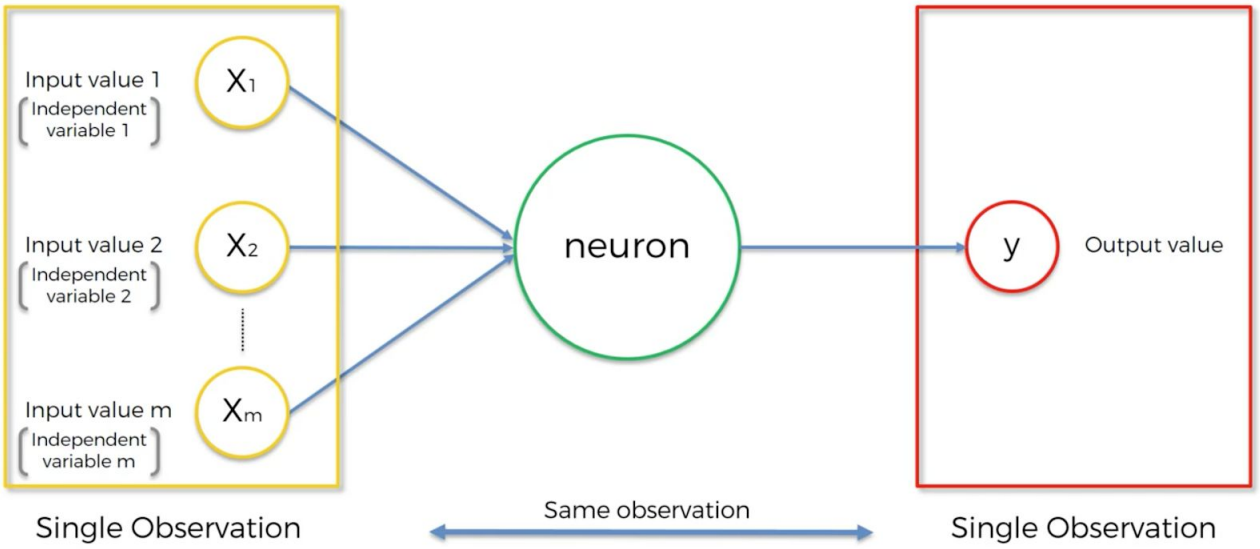
Deep learning wants to take input, and pass it to subsequent layers, and so on until a desirable output is met. With iterative process, the weights of the branches is changed, so as to match the output with the desirable output.

For example, if you want to figure out when a restaurant is open in evening vs morning, you can create this network, and evolve the weights according the the formula: adjustment = error x gradient x delta.



- neuron

The Neuron

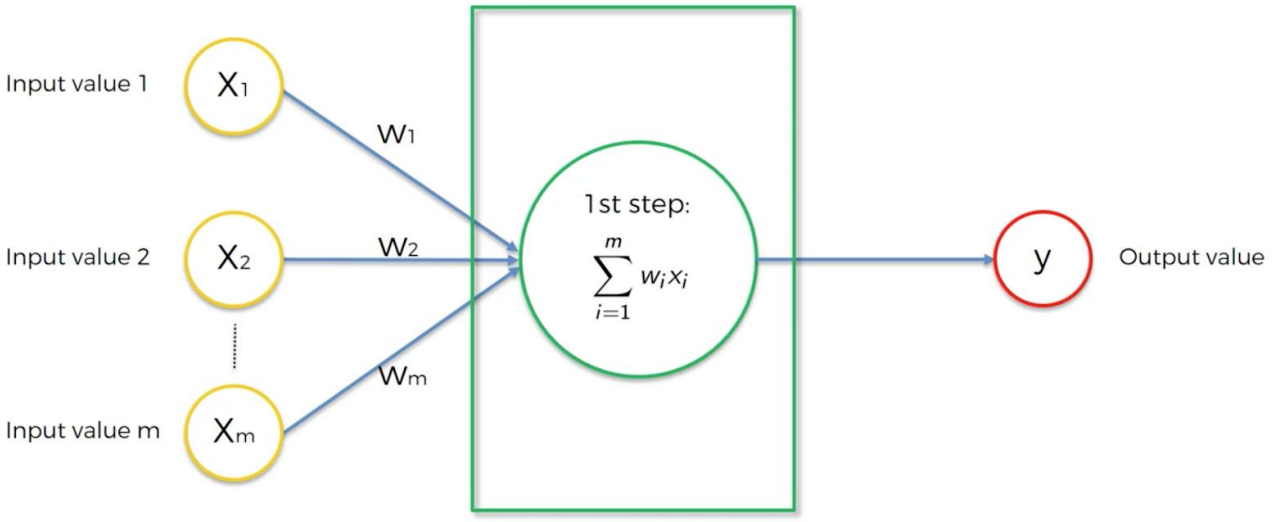


- the input values need to be standardized or normalized
    - Data standardization is the process of transforming data from disparate sources and systems into a consistent format. Standardizing data is a critical step in a data quality process because it makes it easier to identify errors, outliers and other issues within your data sets. It also makes your data easier to analyze and ensures that it is reliable.
        - more: https://www.ringlead.com/4-steps-data-standardization/#.WVjJR8bURZI
    - In ANN and other data mining approaches we need to normalize the inputs, otherwise the network will be ill-conditioned. In essence, normalization is done to have the same range of values for each of the inputs to the ANN model. This can guarantee stable convergence of weight and biases
- output value can be:
    - continoous
    - binary
    - categorical (in this care there'll be several output values)

What happens in neuron

1) sums all weighted input values

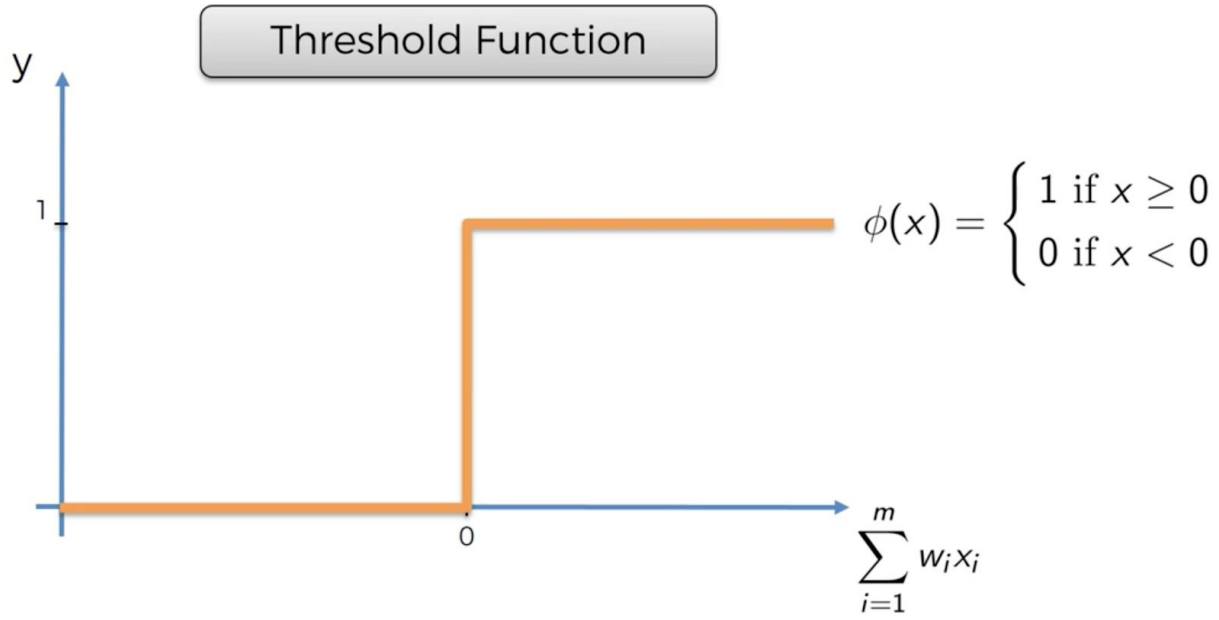
The Neuron



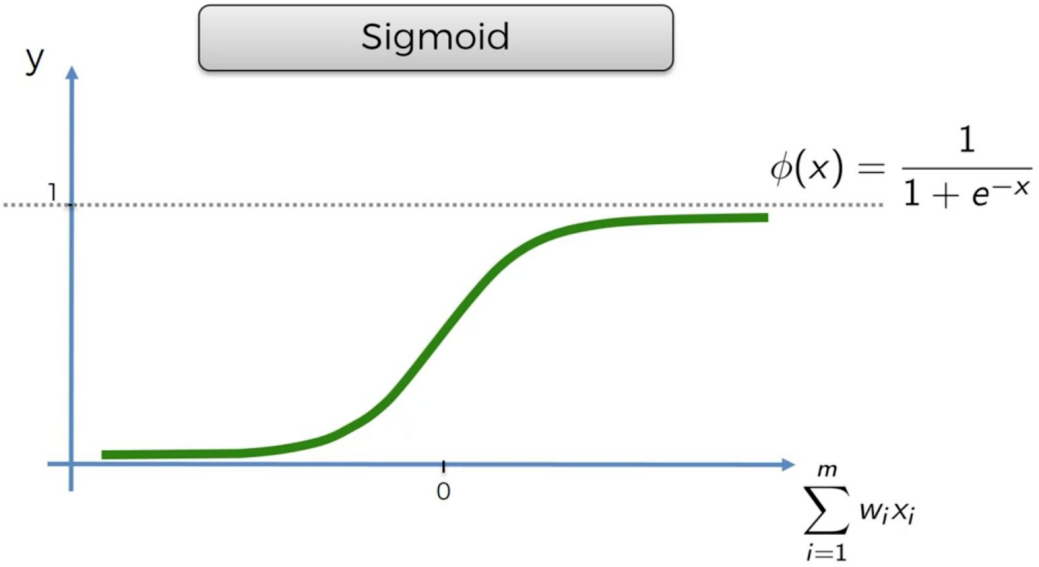
2) Applies activation function

- activation function

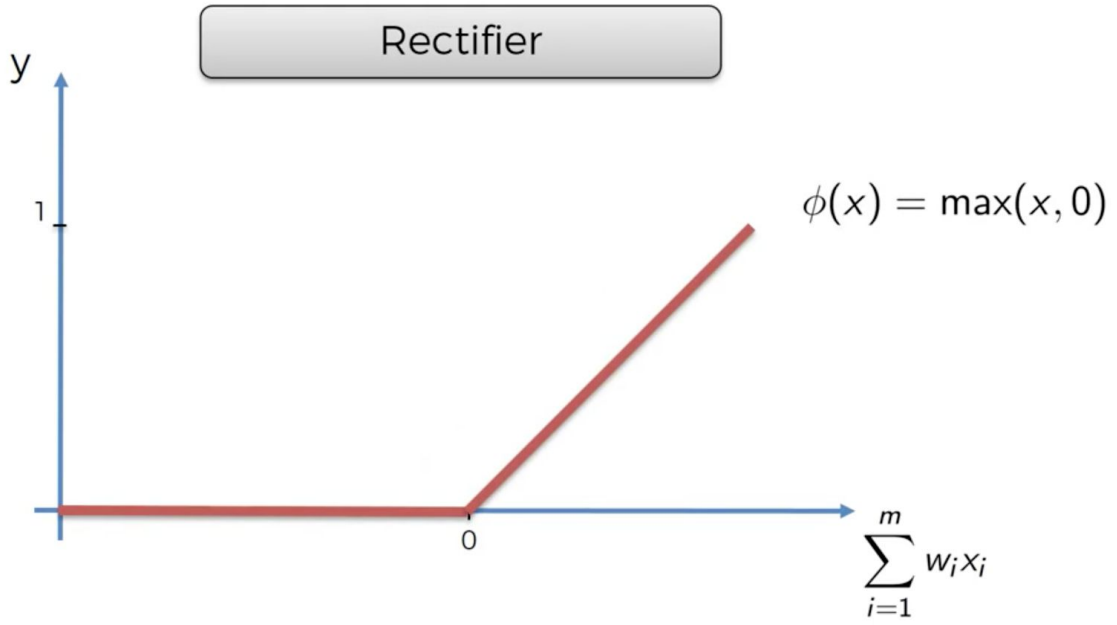
  4 popular activation functions are: (value = sum(wi,xi))
  - threshold function - if value < 0 , function passes 0, if value is >= 0, it passes 1 - either yes or no



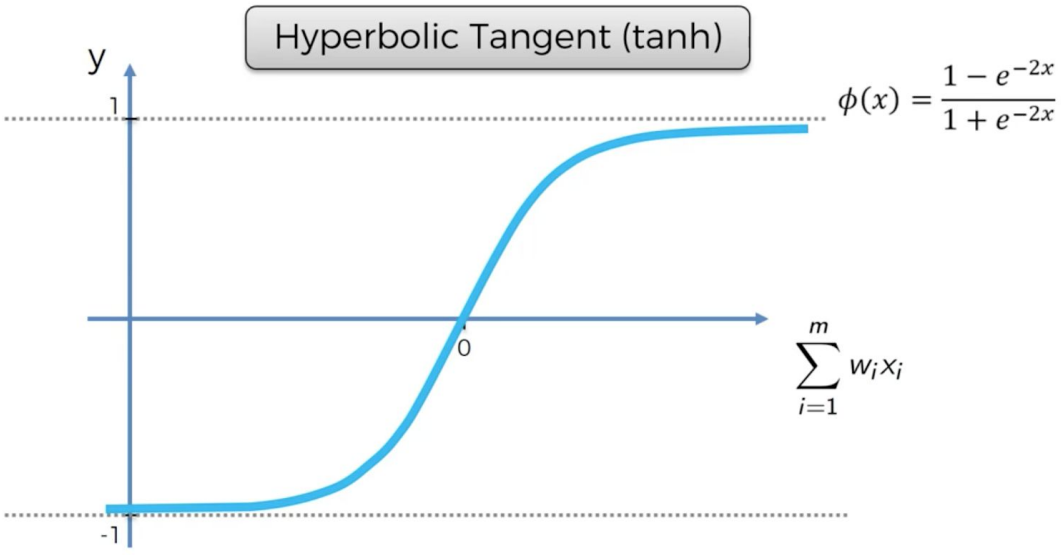
  - sigmoid function - it's smooth gradual progression

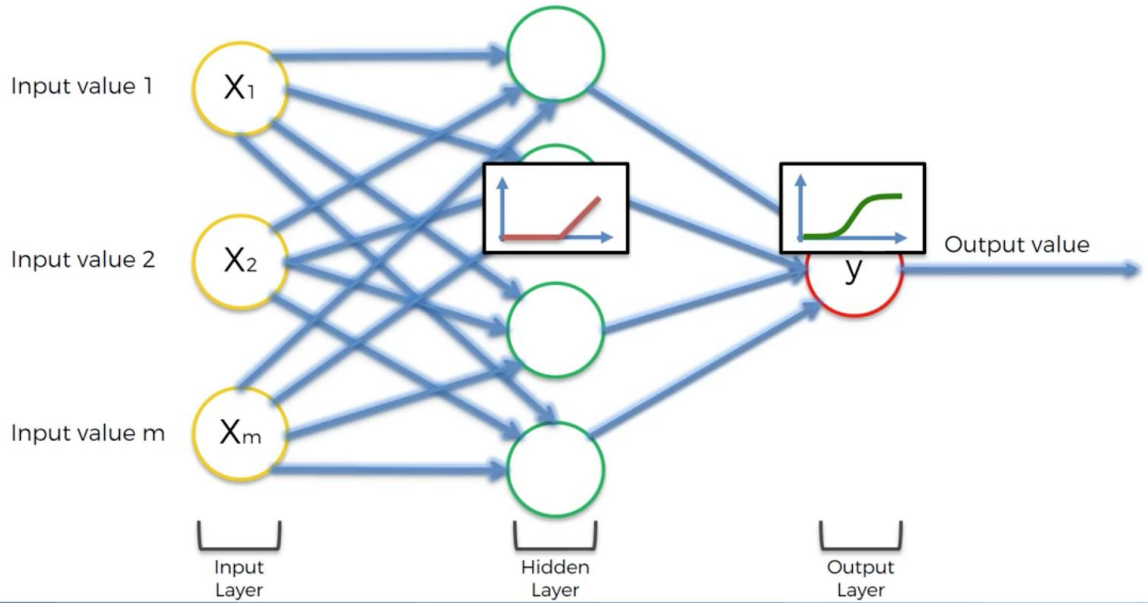


- rectifier - one of the most popular function for ANN, all the way to 0 passes 0, and then it gradually progresses as the input value increases



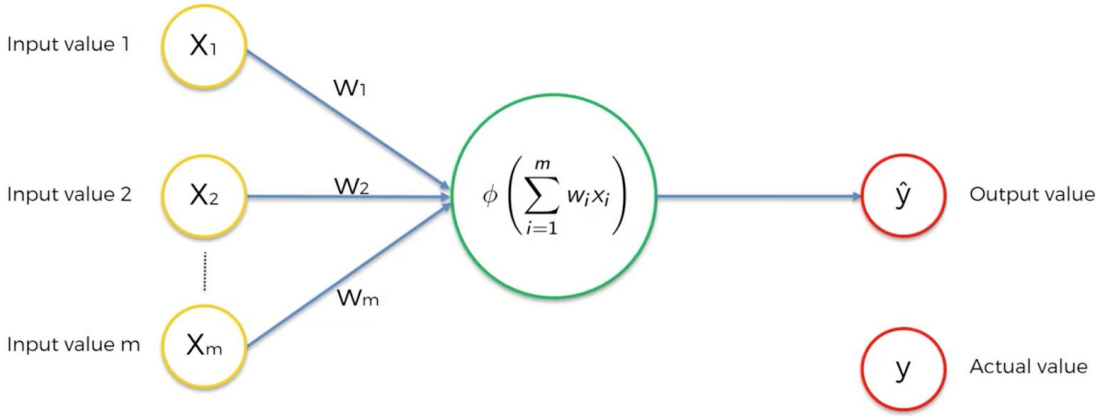
- hyperbolic tagent - similar to sigmoid, but here it goes below 0





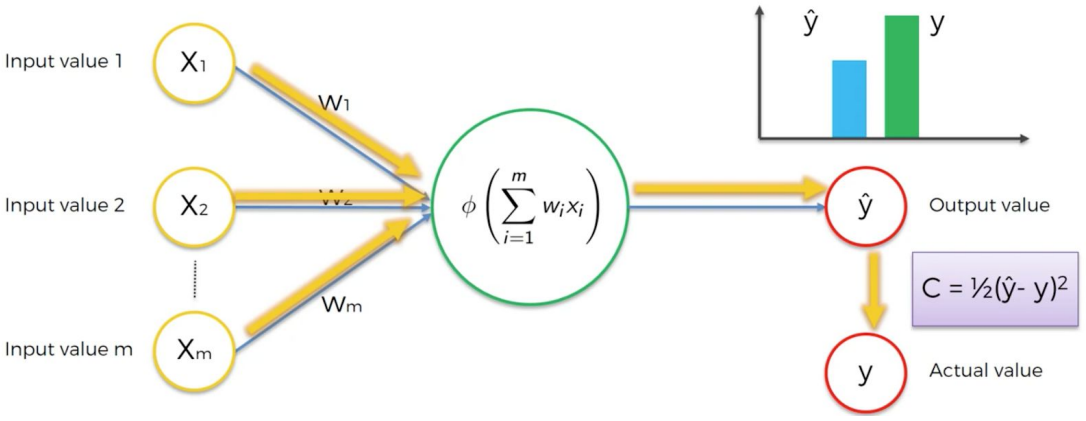
- how do NN learn
  - you create a NN and provide the input, it’s gonna learn by itself

How do Neural Networks learn?



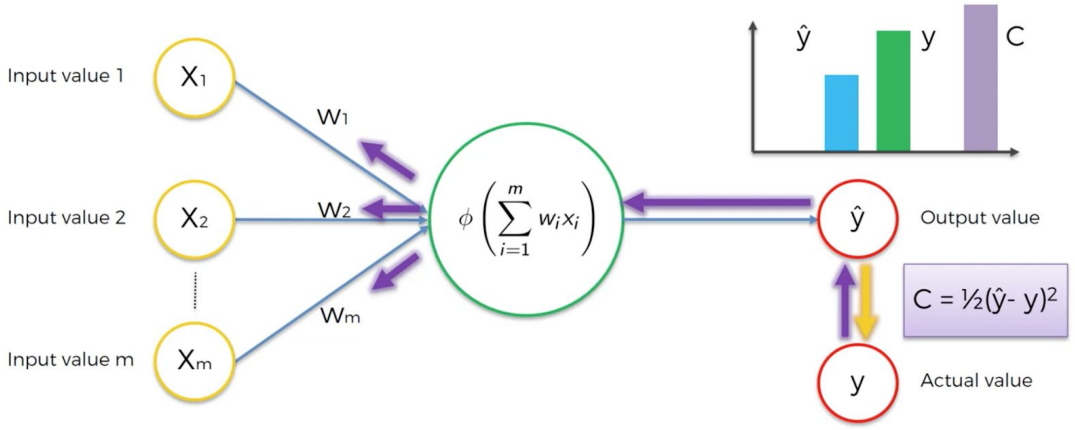
You compare the output and the actual value, calculate the cost function C to get an error of prediction - goal is to obviously minimise cost function
(There are different types of cost functions, the one I present is the most popular)

How do Neural Networks learn?



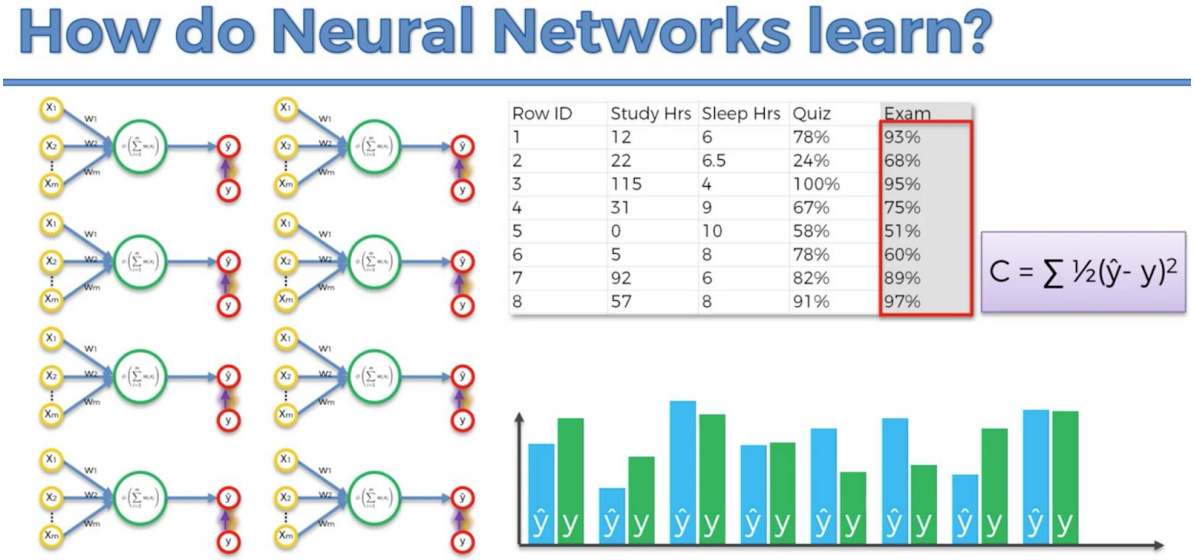
Then you go back and modify the weights

How do Neural Networks learn?

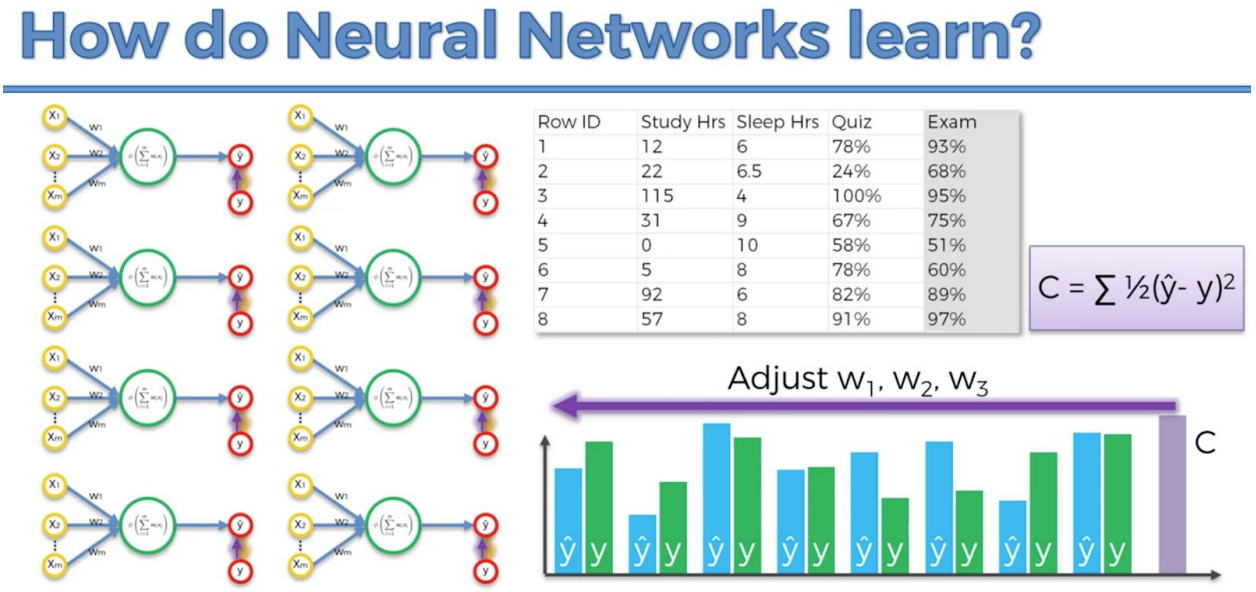


And so on to get the smallest cost function.

Example with many rows of data



While updating the weights, you update them for all perceptrons as they share it

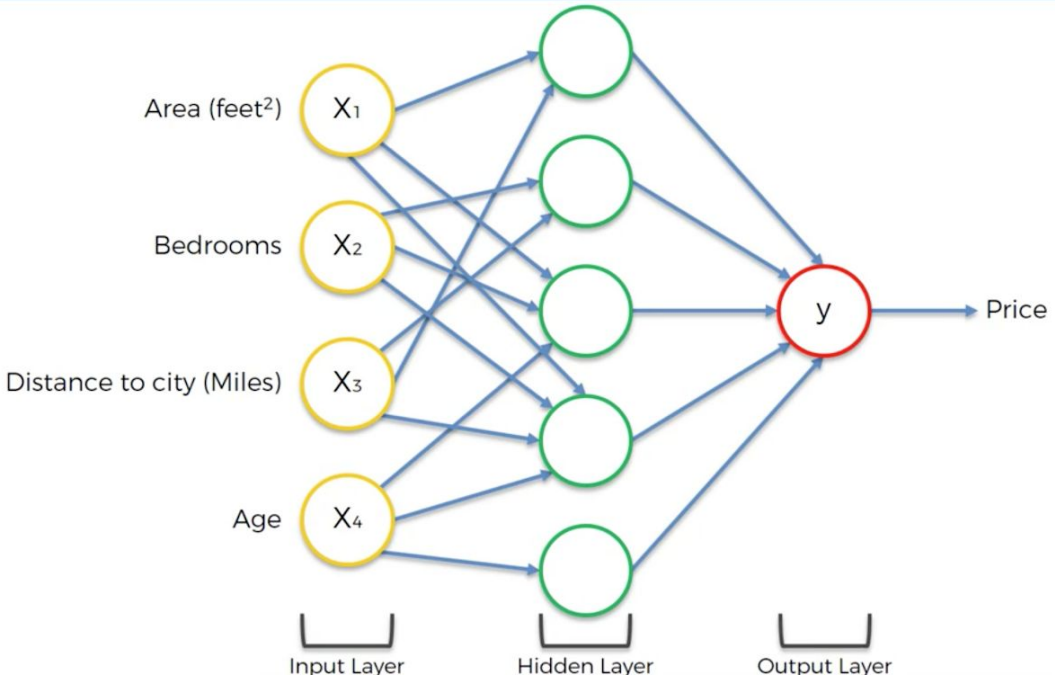


The process of coming back and modifying weights is called backpropagation

- how do NN work

Once the network has been trained with enough learning examples, it reaches a point where you can present it with an entirely new set of inputs it's never seen before and see how it respond.
Say we have already trained neural network and want to choose predict a house price.
We assume 4 input variables - area, no bedrooms, distance to city, age of building.
Each neuron in hidden layer kind of ‘focuses’ on specific inputs. One neuron cannot predict the price for the building but together, each having a different view on inputs may do.

How Do Neural Networks Work?



- gradient descent - how can we minimize cost function
  -

Usage of neural networks
On the basis of this example, you can probably see lots of different applications for neural networks that involve recognizing patterns and making simple decisions about them. In airplanes, you might use a neural network as a basic autopilot, with input units reading signals from the various cockpit instruments and output units modifying the plane's controls appropriately to keep it safely on course. Inside a factory, you could use a neural network for quality control. Let's say you're producing clothes washing detergent in some giant, convoluted chemical process. You could measure the final detergent in various ways (its color, acidity, thickness, or whatever), feed those measurements into your neural network as inputs, and then have the network decide whether to accept or reject the batch.
There are lots of applications for neural networks in security, too. Suppose you're running a bank with many thousands of credit-card transactions passing through your computer system every single minute. You need a quick automated way of identifying any transactions that might be fraudulent—and that's something for which a neural network is perfectly suited. Your inputs would be things like 1) Is the cardholder actually present? 2) Has a valid PIN number been used? 3) Have five or more transactions been presented with this card in the last 10 minutes? 4) Is the card being used in a different country from which it's registered? —and so on. With enough clues, a neural network can flag up any transactions that look suspicious, allowing a human operator to investigate them more closely. In a very similar way, a bank could use a neural network to help it decide whether to give loans to people on the basis of their past credit history, current earnings, and employment record.

24. Architecture of distributed and parallel systems, methods of parallel and distributed processing.

Various software and hardware architectures are used for distributed and parallel computing.
At a lower level, it is necessary to interconnect multiple CPUs with some sort of network. At a higher level, it is necessary to interconnect processes running on those CPUs with some sort of communication system.

Architecture:
- shared memory between all CPUs (e.g. modern com wputer) (usually this is called parallel system)
    * difficult to create hardware
    * not scalable
    * high cost (design, manufacturing complexity)
    * short lifetime: the model is not easily extensible
- distributed memory multicomputer (e.g. multiple computers connected over network)
    * processor can only access its own (local) memory, interprocess communication is achieved by sending messages between computers (e.g. using MPI library)
    * communication between computers can be realised in many different ways (e.g. computer may directly send messages to all other computers, use fast connection to send messages to some peers faster, ring topology - only can send messages to computers next to it, grid, cluster, etc.)
    * communication is expensive

Methods:
- divide and conquer
- multi-agents systems

25. Grids and clusters. Exploitation and development problems.

Grid vs Cluster

Both - two or more interconnected computers used to solve a problem (by sharing resources)
Cluster - The same hardware and OS. Tightly coupled systems (located in one place)
Grid - Different hardware and OS. Decentralized (nodes may be distributed over a LAN/WAN)

Exploitation

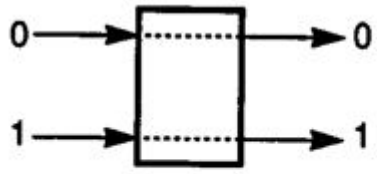
(wszystkie złożone obliczeniowo zadania, na egzaminie można podać np:)

- Predictive Modeling and Simulations - weather forecasting, flood warning
- Engineering Design and Automation - computational aerodynamics, image processing, AI
- Energy Resources Exploration - reservoir modeling, seismic exploration
- Medical, Military and Basic Research - medical imaging, nuclear weapon design, quantum mechanics problems
- Visualization - computer-generated graphics, films and animations

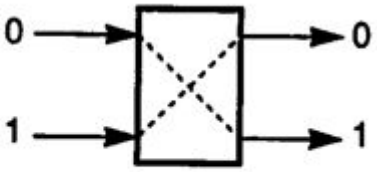
Development problems

- Security - authorization, authentication, reliable access and data integrity need to be provided.
- Trusted results - prevention from producing false, misleading, or erroneous results by participants - solved by check if 2 random nodes produced the same result for the same task.
- Network error handling - solved by redundant paths to nodes (if one is broken, second is used)
- CPU-scavenging - uses desktop computer instruction cycles that would otherwise be wasted at night, during lunch

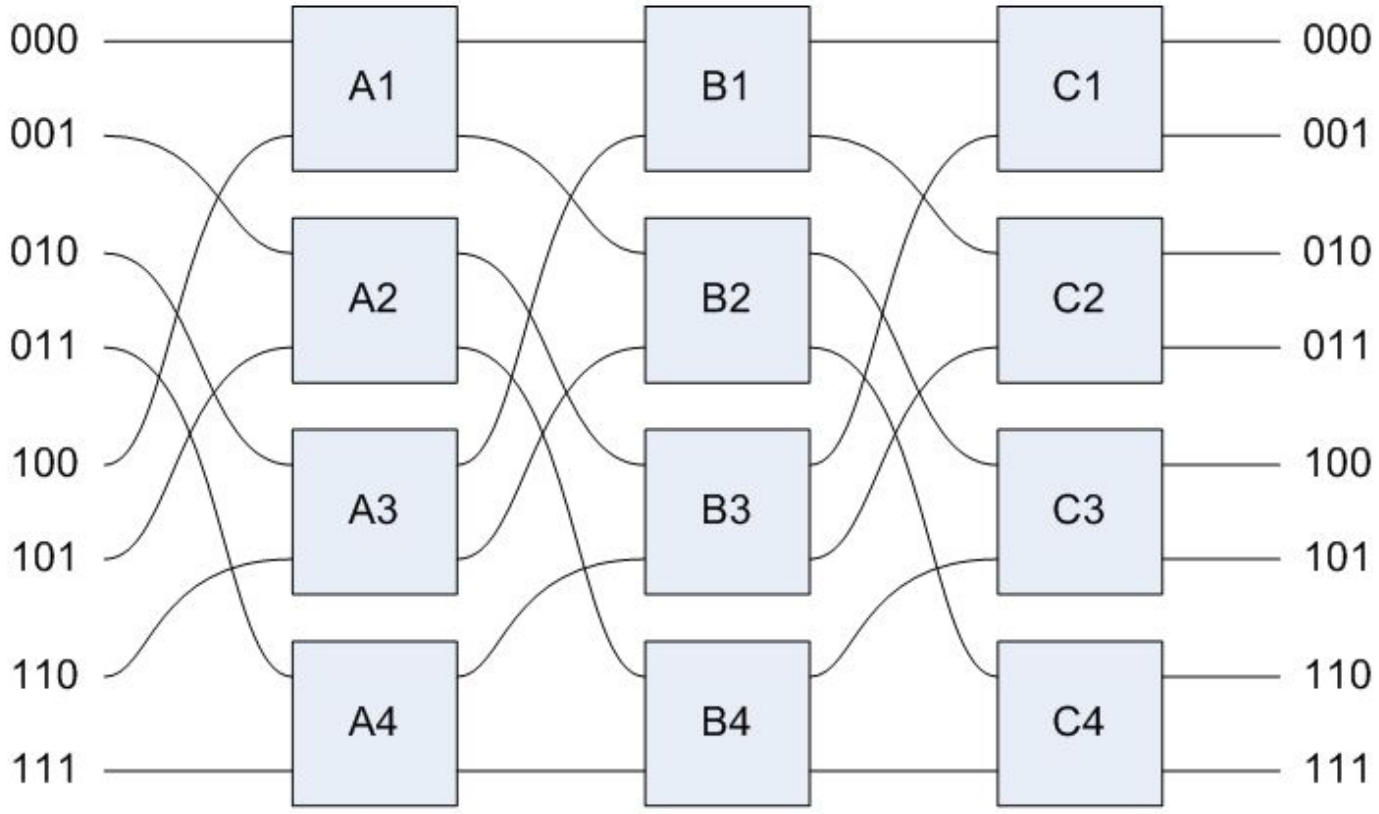
26. Static and dynamic interconnection networks, typical topologies, different routing strategies. K



(a) Straight



(b) Crossover



W zasadzie powyższe obrazki mówią wszystko. Chodzi o sposoby łączenia terminali z lewej strony z terminalami z prawej strony obrazka drugiego. Połączenia mogą być oczywiście różne, zazwyczaj zakłada się switche takie, jak pokazano na obrazku pierwszym czyli dwustopniowe: ustawione równolegle albo skrzyżowane.
Generalnie static networks to takie, w których ustala się pewną topologię na cały czas trwania komunikacji a dynamic to takie, gdzie każda wysyłka każdej informacji wiąże się z ustawianiem na nowo.

27. Automatic program parallelisation, dependencies in sequential programs, identification of parallelism,

Automatic program parallelisation - process of converting sequential code into multi-threaded code (automatically, provided by compiler during compilation). Done on the base of complex program analysis (i.a. dependence analysis). The autoparallelization is focused on loops (take most of the execution time, parallelizable)

Dependencies in sequential programs (compiler perspective) - statement S2 depends on S1 if S1 must be executed before S2.

Control dependency - condition guard - statement S2 is executed only if " S1" statement is false

```
S1      if x > 2 goto L1
S2      y := 3
S3  L1: z := y + 1
```

Data dependency - data in a given statement is dependent on the other statements

Flow/True dependency - S1 modifies a resource that S2 reads and S1 precedes S2 in execution

```
S1      x := 10
S2      y := x + c
```

Antidependence - S2 modifies a resource that S1 reads and S1 precedes S2 in execution

```
S1      x := y + c
S2      y := 10
```

Output dependence - S1 and S2 modify the same resource and S1 precedes S2 in execution

```
S1      x := 10
S2      x := 20
```

Input dependence - S1 and S2 read the same resource and S1 precedes S2 in execution

```
S1      y := x + 3
S2      z := x + 5
```

Identification of parallelism

- Dependencies detection between loop iteration (if not dependent - loop may parallelized).
- Compiler may try to transform a loop to the form which is parallelizable.
- Evaluation of parallelization (does the program run faster - if not - do not parallelize)

28. Evaluations of parallel systems: performance metrics, scalability of parallel systems, Amdhal, Gustafson and other laws.

https://www.ii.pwr.edu.pl/~kwiatkow/pardist2/lec3.pdf

The **parallel runtime** (Tpar) is the time from the moment when computation starts to the moment when the last processor finished its execution.

Performance metrics:

1) **Speedup** (S): the speedup is defined as a ratio of the time needed to solve a problem on a single processor (Tseq) to the time required to solve the problem on parallel system with p processors (Tpar)
2) **Efficiency**: the efficiency of a parallel program is defined as a ratio of speedup to number of processors

The **scalability of parallel system** is a measure of its capacity to increase speedup in proportion to the number of processors. The parallel system is scalable if it maintains efficiency at a fixed value by simultaneously increasing the number of processors and the size of the problem. The scalability reflects a parallel system's ability to utilize increasing processing resources effectively.

Amdhal's law is often used in parallel computing to predict the theoretical speedup when using multiple processors. It states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time that mode can be used (for example: it is possible that only a part of an application can be parallelized).

Gustafson's law says that increase of problem size for large machines can retain scalability with respect to the number of processors. Gustafson's objection is that massively parallel machines allow computations previously unfeasible since they enable computations on very large data sets in fixed amount of time. In other words, a parallel platform does more than speeding up the execution of a code: it enables dealing with larger problems.

There are also:

1) **Lee's law** (S <= p / log_2 p) - it is a generalization of Amdhal's law
2) **Minsky's law** (S <= log_2 p)

Both of them **give an upper bound for speedup** (p - number of processors).

29. Rule-based knowledge representations.

rule-based systems are used as a way to store and manipulate knowledge to interpret information in a useful way. They are often used in artificial intelligence applications and research.

Normally, the term 'rule-based system' is applied to systems involving human-crafted or curated rule sets. Rule-based systems constructed using automatic rule inference, such as rule-based machine learning, are normally excluded from this system type.

A classic example of a rule-based system is the domain-specific expert system that uses rules to make deductions or choices. For example, an expert system might help a doctor choose the correct diagnosis based on a cluster of symptoms, or select tactical moves to play a game.

A typical rule-based system has four basic components:[1]

- A list of rules or **rule base**, which is a specific type of knowledge base.
- An inference engine or semantic reasoner, which infers information or takes action based on the interaction of input and the rule base. The interpreter executes a production system program by performing the following match-resolve-act cycle:[*citation needed*]
  - Match: In this first phase, the left-hand sides of all productions are matched against the contents of working memory. As a result a conflict set is obtained, which consists of instantiations of all satisfied productions. An instantiation of a production is an ordered list of working memory elements that satisfies the left-hand side of the production.
  - Conflict-Resolution: In this second phase, one of the production instantiations in the conflict set is chosen for execution. If no productions are satisfied, the interpreter halts.
  - Act: In this third phase, the actions of the production selected in the conflict-resolution phase are executed. These actions may change the contents of working memory. At the end of this phase, execution returns to the first phase.
- Temporary working memory.
- A user interface or other connection to the outside world through which input and output signals are received and sent.

One of the most popular approaches to knowledge representation is to use production rules, sometimes called IF-THEN rules. They can take various forms.e.g.

IF condition THEN action
IF premise THEN conclusion
IF proposition p1 and proposition p2 are true
  THEN proposition p3 is true

the IF part is called the antecedent (condition) and the THEN part is called consequent (conclusion/action). the antecedent can have multiple parts joined by keywords such as AND, OR or a combination of both.

The antecedent of a rule incorporates two parts: an object and its value, which are connected by an operator, these are operators such as is, are, is not, or mathematical operators like ‘<’ ‘>’

IF ‘age of customer’ < 18

THEN ‘signature of parent’ is required

Members of expert system development team: Domain expert, programmer, knowledge engineer, project manager and the end-user.

Some of the benefits of IF-THEN rules are that they are modular, each defining a relatively small and, at least in principle, independent piece of knowledge. New rules may be added and old ones deleted usually independently of other rules.

A rule based system will contain global rules and facts about the knowledge domain covered. During a particular run of the system a database of local knowledge may also be established, relating to the particular case in hand.

If several rules could all fire at once the inference engine must have a mechanism for "conflict resolution". This may be achieved, for example, by having some predefined order, perhaps on the basis of the strength of the conclusion, or alternatively on the basis of frequency of rule usage.

Forward and Backward chaining through the rules may be used. The two systems each have their advantages and disadvantages and in fact answer different types of question. For example, in Mycin a forward chaining system might answer the question "what do these symptoms suggest?" whereas a backward chaining system might answer the question "does this patient suffer from a pelvic abscess?" In general, rules and goals may need to be constructed differently for forward and backward chaining systems.

well-defined problems: math. problems, chess

ill-defined problems: how to get a job, how to bring economical decisions etc...

30. Knowledge based systems - inference mechanisms.

In the field of Artificial Intelligence, **inference engine** is a component of the system that applies logical rules to the knowledge base to deduce new information. The first inference engines were components of expert systems. The typical expert system consisted of a knowledge base and an inference engine. The knowledge base stored facts about the world. The inference engine applies logical rules to the knowledge base and deduced new knowledge. This process would iterate as each new fact in the knowledge base could trigger additional rules in the inference engine. Inference engines work primarily in one of two modes either special rule or facts: forward chaining and backward chaining. Forward chaining starts with the known facts and asserts new facts. Backward chaining starts with goals, and works backward to determine what facts must be asserted so that the goals can be achieved.

- BACKWARD CHAINING

Backward chaining starts with a list of goals (or a hypothesis) and works backwards from the consequent to the antecedent to see if there is data available that will support any of these consequents. An inference engine using backward chaining would search the inference rules until it finds one which has a consequent (**Then** clause) that matches a desired goal. If the antecedent (**If** clause) of that rule is not known to be true, then it is added to the list of goals (in order for one's goal to be confirmed one must also provide data that confirms this new rule).

For example, suppose a new pet, Fritz, is delivered in an opaque box along with two facts about Fritz:

- Fritz croaks
- Fritz eats flies

The goal is to decide whether Fritz is green, based on a rule base containing the following four rules:

1. **If** X croaks and X eats flies – **Then** X is a frog
2. **If** X chirps and X sings – **Then** X is a canary
3. **If** X is a frog – **Then** X is green
4. **If** X is a canary – **Then** X is yellow

With backward reasoning, an inference engine can determine whether Fritz is green in four steps. To start, the query is phrased as a goal assertion that is to be proved: "Fritz is green".

1. Fritz is substituted for X in rule #3 to see if its consequent matches the goal, so rule #3 becomes:

If Fritz is a frog – **Then** Fritz is green

Since the consequent matches the goal ("Fritz is green"),the rules engine now needs to see if the antecedent ("If Fritz is a frog") can be proved. The antecedent therefore becomes the new goal:

Fritz is a frog

2. Again substituting Fritz for X, rule #1 becomes:

If Fritz croaks and Fritz eats flies – **Then** Fritz is a frog

Since the consequent matches the current goal ("Fritz is a frog"), the inference engine now needs to see if the antecedent ("If Fritz croaks and eats flies") can be proved. The antecedent therefore becomes the new goal:

Fritz croaks and Fritz eats flies

3. Since this goal is a conjunction of two statements, the inference engine breaks it into two sub-goals, both of which must be proved:

Fritz croaks

Fritz eats flies

4. To prove both of these sub-goals, the inference engine sees that both of these sub-goals were given as initial facts. Therefore, the conjunction is true:

Fritz croaks and Fritz eats flies

therefore the antecedent of rule #1 is true and the consequent must be true:

Fritz is a frog

therefore the antecedent of rule #3 is true and the consequent must be true:

Fritz is green

- FORWARD CHAINING

Forward chaining starts with the available data and uses inference rules to extract more data (from an end user, for example) until a goal is reached. An inference engine using forward chaining searches the inference rules until it finds one where the antecedent (**If** clause) is known to be true. When such a rule is found, the engine can conclude, or infer, the consequent (**Then** clause), resulting in the addition of new information to its data.

Inference engines will iterate through this process until a goal is reached.

For example, suppose that the goal is to conclude the color of a pet named Fritz, given that he croaks and eats flies, and that the rule base contains the following four rules:

1. **If** X croaks and X eats flies - **Then** X is a frog
2. **If** X chirps and X sings - **Then** X is a canary
3. **If** X is a frog - **Then** X is green
4. **If** X is a canary - **Then** X is yellow

Let us illustrate forward chaining by following the pattern of a computer as it evaluates the rules. Assume the following facts:

- Fritz croaks
- Fritz eats flies

With forward reasoning, the inference engine can derive that Fritz is green in a series of steps:

1. Since the base facts indicate that "Fritz croaks" and "Fritz eats flies", the antecedent of rule #1 is satisfied by substituting Fritz for X, and the inference engine concludes:

`Fritz is a frog`

2. The antecedent of rule #3 is then satisfied by substituting Fritz for X, and the inference engine concludes:

`Fritz is green`

One of the advantages of forward-chaining over backward-chaining is that the reception of new data can trigger new inferences, which makes the engine better suited to dynamic situations in which conditions are likely to change.

Modus Ponens: It can be summarized as "*P* implies *Q* and *P* is asserted to be true, so therefore *Q* must be true." Backward and forward chaining are repeated applications of modus ponens

31. Incompleteness, inconsistency and uncertainty of knowledge.

- INCOMPLETENESS:

Incomplete information should preferably be ignored during problem solving unless the missing information is necessary to solve the problem. The PDO/EPB representation provides for unknown information to be ``hidden'' (and thereby ignored) by the use of multiple levels of detail. Areas of a boundary which are irrelevant to a problem can be described extremely coarsely - perhaps just as a ``wiggle''.
In the case where information is incomplete, but the missing quantities are required for the problem solution, the system must be able to hypothesise a constrained range of values for the unknown quantity. This ability was explained in the discussion of determination of fit in the path planning system. An unknown magnitude, whether a calculated ``synthetic'' magnitude or simply one that could not be measured, can be represented in the partial distance ordering as being completely unconstrained. The qualitative reasoning system can then make use of a repertoire of qualitative geometry techniques for constraining the value.

If we want to build a classifier from a set of incomplete records, we may do it in two ways. Either we build a complete data set, then we apply traditional learning methods; or we use a procedure which can internally manage missing data. We are more interested in the first strategy, for the following reasons: ! Data collectors may have knowledge of missing data mechanisms, which can be used while building the complete data set.
If we do not know the underlying dependency, which is usually the case, it may be very difficult to analyze this data. ! With a complete data set, every existing learning technique can be used. ! A common starting point allows comparisons of different learners.

An example of a classifier that can handle missing data is C4.5. This algorithm builds a decision tree which can be then used to classify new records, and it has shown to be very efficient if compared with other methods. The easiest way to obtain a complete data set from an incomplete one is to erase missing items. These conventional methods are used because of their simplicity, and are usually the default option in statistical packages. In this report we present two variations of this strategy: listwise deletion and pairwise deletion. If we do not want to lose data and perhaps information, we may try to guess missing items. This process is generally called imputation. In fact, usually missing values depend on other values, and if we find a correlation between two variables, we may use it to impute missing items.

Imputations may be deterministic or random (stochastic). In the first case, imputations are determined by the incomplete data table, and are the same if the method is applied again. This is useful if we want to compare the work of different researchers, and if we want to repeat simulations. In the second case, imputations are randomly drawn. We may use different kinds of information to impute missing data. We may analyze the behavior of all the other records on the missing variable (global imputation based on missing attribute). We may try to find a relationship between other variables and the missing one in all the other records, and use this relationship to impute the missing value (global imputation based on non-missing attributes). Or we may look for similar 4 records to see how they behave on the missing attribute (local imputation). If we are interested in guessing statistics, i.e., summary values of the sample, we may also use the EM algorithm [5], based on the maximization of

a likelihood function. This algorithm directly estimates statistics, without imputing values - even if it can be used also for imputation.

A well-known approach to incomplete supervised classification is tree-based classification. From an incomplete data table, a decision tree is built, which is also able to classify unseen records with missing values. For a short overview of tree-based classification techniques see [16] and [17]. In general, we may build a decision tree following this procedure: 1. Select a test which can partition records in disjoint sets. 2. Partition the current set of records (initially, the training set). 3. For every partition, repeat this procedure. The most famous algorithm of this kind is C4.5 [3], which has shown to be very effective [4]. It chooses the test which maximizes the gain ratio, a measure that expresses the improvement of classification power of the tree after the partition. This technique can also be used in presence of missing values. 1. The test is chosen using a measure similar to the gain ratio, adjusted by a factor that depends on the number of records complete on the tested attribute. 2. Every incomplete record is then assigned to all partitions, with a probability which depends on the size of the partition. 3. When an unseen record must be classified, but the testing attribute is unknown, all possible paths are examined, leading to more than one possible class.

- INCONSISTENCY:

Handling contradictory data is one of the most complex and important problems in reasoning under uncertainty. To handle inconsistent information one needs a logic that, unlike classical logic, allows contradictory yet non-trivial theories. Logics of this sort are called paraconsistent [da Costa, 1974]. There are many AI applications that are based, in one way or another, on some paraconsistent logic
A **paraconsistent logic** is a logical system that attempts to deal with contradictions in a discriminating way. Alternatively, paraconsistent logic is the subfield of logic that is concerned with studying and developing paraconsistent (or "inconsistency-tolerant") systems of logic.

to reason with inconsistent knowledge it is advised to use an existing framework or algorithms created for such tasks, like *DeLP, which* is an argumentative framework based on logic programming which is capable of dealing with possibly inconsistent knowledge bases (KB) codified as a set of horn-like clauses called DeLP programs. When presented with a query, DeLP performs a dialectical process in which all arguments in favor and against a conclusion are considered; arguments regarded as ultimately undefeated will be considered warranted.

Another possible solution is to clean the data of the inconsistencies, this most often require additional input from a domain expert in order to clarify the inconsistencies.

- UNCERTAINTY:

On the uncertainty issue, the most long-established way of dealing with it is probability theory. In the probabilistic approach, a logic sentence is labelled with a real number in the range 0 to 1, 0 meaning the sentence is completely false, and 1 meaning it is completely false. A label of 0.5 means there is equal chance that the sentence is true or false.

other possible ways are:

As a Certainty Factor: how certain are we that X is true or not true.

– CF(x)= 0: I have no knowledge if x is true or false
– CF(x)= -1: I believe x is not true
– CF(x)= 1: I believe x is true 3.

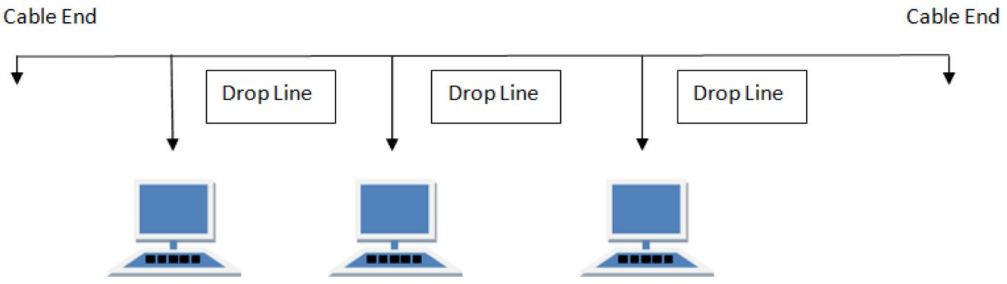
As Fuzzy Logic: to what degree x is in various sets.

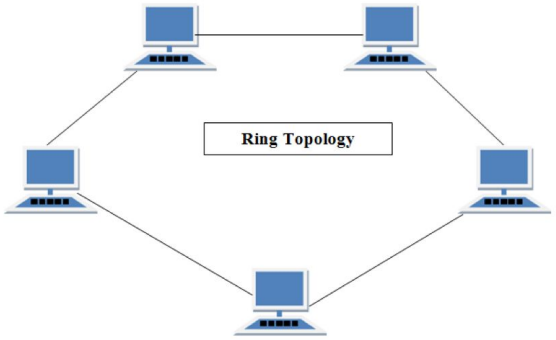
– a given cup of coffee can be member of cold-drink and hot-drink sets at the same time.

We can calculate the probability of an event with Bayes Theorem:

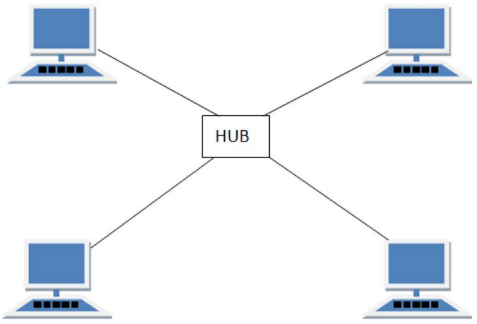
Given: p(y | x) = p(y & x) / p(x)
Then: p(y & x) = p(y | x) * p(x)
Substituing this into our prior formula:
p(x | y) = p(x & y) / p(y)
Gives: p(x | y) = p(y | x) * p(x) / p(y)

32. Topologies of Computer Network.

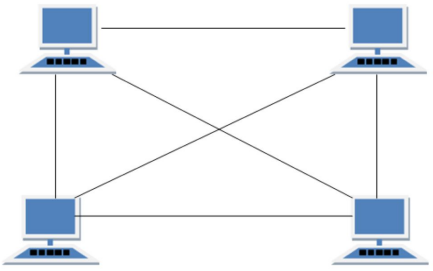
<table>
<tr><th colspan="3">BUS TOPOLOGY</th></tr>
<tr><td colspan="3"> 
</td></tr>
<tr><th>Features</th><th>Advantages</th><th>Disadvanatges</th></tr>
<tr><td>It transmits data only in one direction.
Every device is connected to a single cable</td><td>It is cost effective.
Cable required is least compared to other network topology.
Used in small networks.
It is easy to understand.
Easy to expand joining two cables together.</td><td>Cables fails then whole network fails.
If network traffic is heavy or nodes are more the performance of the network decreases.
Cable has a limited length.
It is slower than the ring topology.</td></tr>
</table>

RING TOPOLOGY		
		
Features	Advantages	Disadvanatges
1. A number of repeaters are used for Ring topology with large number of nodes, because if someone wants to send some data to the last node in the ring topology with 100 nodes, then the data will have to pass through 99 nodes to reach the 100th node. Hence to prevent data loss repeaters are used in the network. 2. The transmission is unidirectional, but it can be made bidirectional by having 2 connections between each Network Node, it is called **Dual Ring Topology**. 3. In Dual Ring Topology, two ring networks are formed, and data flow is in opposite direction in them. Also, if one ring fails, the second ring can act as a backup, to keep the network up. 4. Data is transferred in a sequential manner that is bit by bit. Data transmitted, has to pass through each node of the network, till the destination node.	1. Transmitting network is not affected by high traffic or by adding more nodes, as only the nodes having tokens can transmit data. 2. Cheap to install and expand	1. Troubleshooting is difficult in ring topology. 2. Adding or deleting the computers disturbs the network activity. 3. Failure of one computer disturbs the whole network.

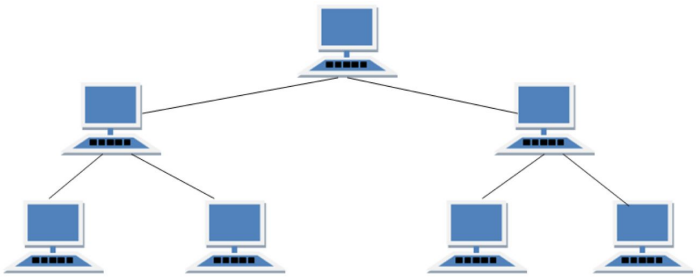
STAR TOPOLOGY

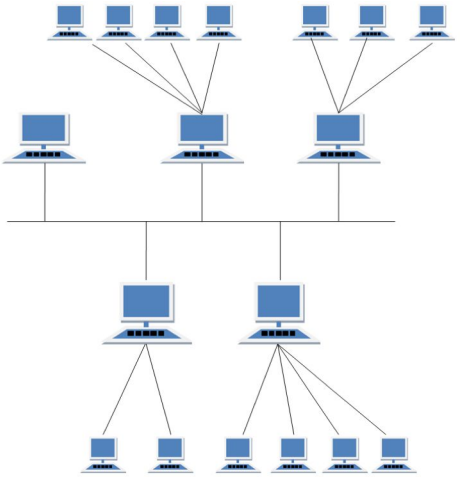


Features	Advantages	Disadvantages
1. Every node has its own dedicated connection to the hub. 2. Hub acts as a repeater for data flow. 3. Can be used with twisted pair, Optical Fibre or coaxial cable.	1. Fast performance with few nodes and low network traffic. 2. Hub can be upgraded easily. 3. Easy to troubleshoot. 4. Easy to setup and modify. 5. Only that node is affected which has failed, rest of the nodes can work smoothly.	1. Cost of installation is high. 2. Expensive to use. 3. If the hub fails then the whole network is stopped because all the nodes depend on the hub. 4. Performance is based on the hub that is it depends on its capacity

MESH TOPOLOGY		
		
Features	Advantages	Disadvanatges

1. TYPES: **Partial Mesh Topology :** In this topology some of the systems are connected in the same fashion as mesh topology but some devices are only connected to two or three devices. 2. **Full Mesh Topology :** Each and every nodes or devices are connected to each other. 1. Fully connected. 2. Robust. 3. Not flexible.	1. Each connection can carry its own data load. 2. It is robust. 3. Fault is diagnosed easily. 4. Provides security and privacy.	1. Installation and configuration is difficult. 2. Cabling cost is more. 3. Bulk wiring is required.

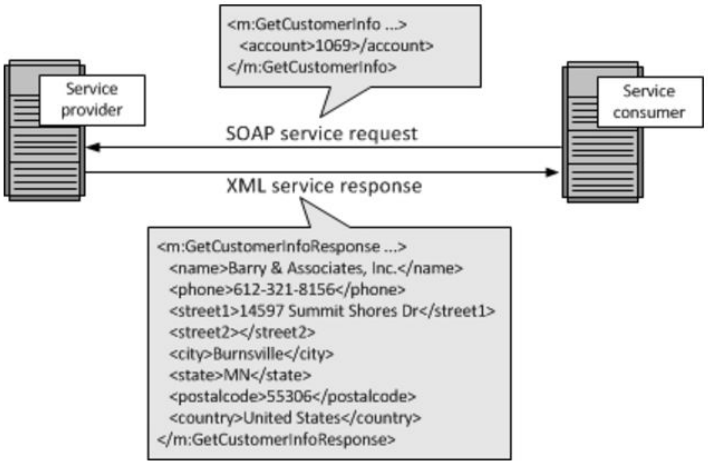
TREE TOPOLOGY		
		
Features	Advantages	Disadvanatges
1. Ideal if workstations are located in groups. 2. Used in Wide Area Network.	1. Extension of bus and star topologies. 2. Expansion of nodes is possible and easy. 3. Easily managed and maintained. 4. Error detection is easily done.	1. Heavily cabled. 2. Costly. 3. If more nodes are added maintenance is difficult. 4. Central hub fails, network fails.

HYBRID TOPOLOGY		
		
Features	Advantages	Disadvanatges
1. It is a combination of two or topologies 2. Inherits the advantages and disadvantages of the topologies included	1. Reliable as Error detecting and trouble shooting is easy. 2. Effective. 3. Scalable as size can be increased easily. 4. Flexible	1. Complex in design. 2. Costly.

33. Internet and Web services Architecture. Web and P2P systems. K

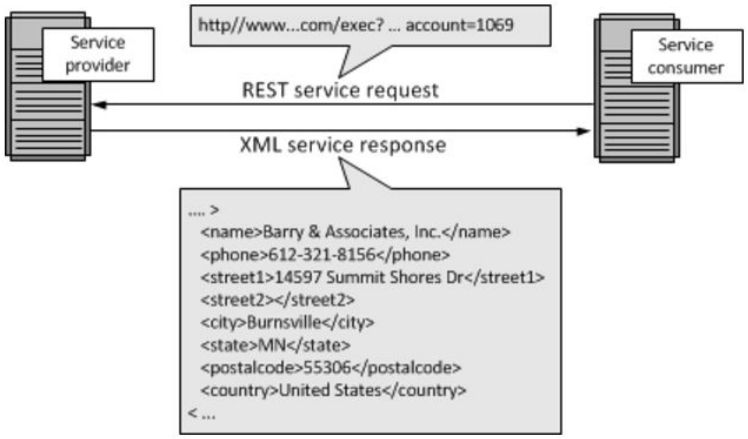
There are three main approaches to web services: SOAP, REST i JSON.

SOAP



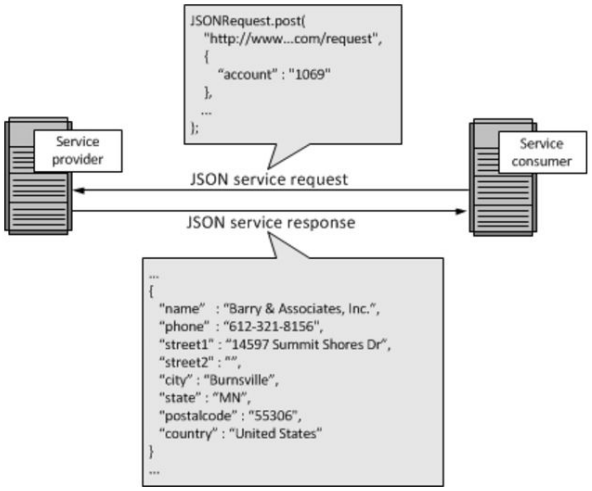
Soap uses XML to encode and HTTP to transfer the messages (but it's possible to other protocols). SOAP file includes <envelope> as a main tag, with <body> and optional <header> inside. All the obligatory data is always included in <body>. Additional <fault> may be added in case of defining an error.

REST



REST is far simpler and easier to use than SOAP. It uses common interface, stateless communication, resources, representations and hyper-media.

JSON



JSON is a lightweight format of data exchange. Text should be coded in UTF-8 which is treated as a standard. All the data are variables (JSON contains co executable code). In general, JSON takes less space than XML, but is harder to cherry-pick data from it.

P2P, as opposed to client-server is an architecture, gives all the hosts the same rights (so, there are no roles). Client-server is based on an idea that servers provide services while clients request those services.

34. Measurement, estimation and prediction of communication time in the Internet.

One ought to provide as shortest page load time (PLT) as possible. PLT is the time it takes between when a user carries out an action that causes a page to load to when the web page fully loads

Possible Performance Indicators
• Response Time: from the user request until the last download is complete
• Navigate Time: from the user request until the server sends navigate complete
• DownLoad Time: sum of all the begin-to-end downloads – greater than Response Time when there is interleaving
• Wait Time: when nothing is being done

• To impact and improve user centric performance, focus on 9 core metrics:

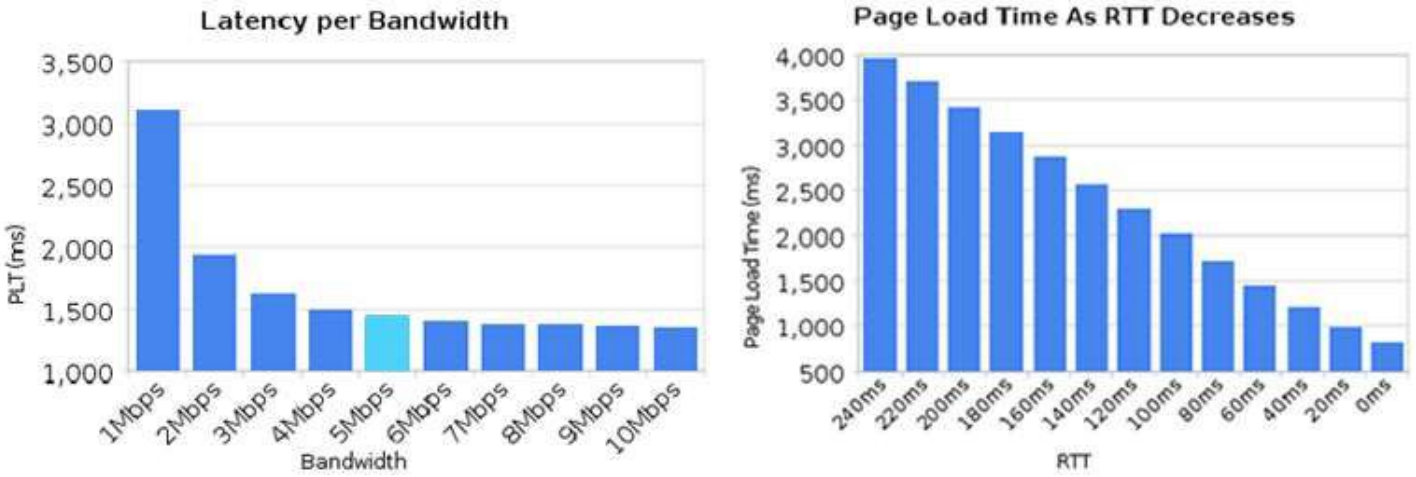
– Availability
– Outages
– Average Download Time - Geo Mean
– Time in Client Versus Time In Generation/Backend
– Variability - 85th and 95th percentiles
– Geographic Variability
– Hourly Variability (Load Handling)
– Third Party Quality
– Size/Element Count/Domains

Bandwidth

• Bandwidth is a widely-used term that usually refers to the data-carrying capacity of a network or data transmission medium. It indicates the maximum amount of data that can pass from one point to another in a unit of time.

Latency
• Network latency is the term used to indicate any kind of delay that happens in data communication over a network.
• Most of the time, a packet's round-trip time (RTT) is measured.
• Network latency can be tested through ping tests and traceroutes.

Latency: The new Web performance bottleneck



• Upgrading connection from 1Mbps to 2Mbps halves the PLT, but quickly thereafter we are into diminishing returns. In fact, upgrading from 5Mbps to 10Mbps results in a mere 5% improvement in page loading times! In other words, an average consumer in the US would not benefit much from upgrading their connection when it comes to browsing the web.
• However, the latency graph tells an entirely different story. For every 20ms improvement in latency, we have a linear improvement in page loading times. There are many good reasons for this: an average page is composed of many small resources, which require many connections, and TCP performance of each is closely tied to RTT.

If we want a faster browsing experience then reducing the round trip time (RTT) should be near the top of our list. Or, as Mike Belshe put it: more bandwidth doesn't matter (much).

Possible parameters of HTTP transactions:
• CONNECT – the time taken to form a connection by the browser with the server, i.e. it is the time spacing between the SYN packet sent by the client and the ACK-SYN packet received by the client.
• LEFT_BYTES – the time taken to receive data from the server, namely it is the time spacing between the first and last data packets received by the client.

RTT and HTTP transfer rate estimation

• The RTT can be estimated by CONNECT.

• The (average) transfer rate can be estimated by dividing a number of bytes transferred by the time taken to download them (LEFT_BYTES).

35. The Web Server model. Access and scheduling algorithms for HTTP requests in a Web Server. K

The primary function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP). Pages delivered are most frequently HTML documents, which may include images, style sheets and scripts in addition to text content.

Web server is a software that provides services for other programs (usually in a computer network). On the example of popular Apache: it supports a variety of features, many implemented as compiled modules which extend the core functionality. A sample of other features include Secure Sockets Layer and Transport Layer Security support (mod_ssl), a proxy module (mod_proxy), a URL rewriting module (mod_rewrite), custom log files (mod_log_config), and filtering support (mod_include and mod_ext_filter).
Traditionally, requests at a Web server are scheduled independently of their size. The requests are timeshared, with each request receiving a fair share of the Web server resources.

Common scheduling disciplines include the following:
-Random scheduling (RSS)
-First In, First Out (FIFO), also known as First Come First Served (FCFS)
-Last In, First Out (LIFO)
-Shortest seek first, also known as Shortest Seek / Service Time First (SSTF)
-Elevator algorithm, also known as SCAN (including its variants, C-SCAN, LOOK, and C-LOOK)
-N-Step-SCAN SCAN of N records at a time
-FSCAN, N-Step-SCAN where N equals queue size at start of the SCAN cycle
-Completely Fair Queuing (CFQ) on Linux
-Anticipatory scheduling
-Noop scheduler
-Deadline scheduler
-mClock scheduler
-Budget Fair Queueing (BFQ) scheduler.
-Kyber

36. Differences between IPv4 and Ipv6.

More information:
https://www.ibm.com/support/knowledgecenter/en/ssw_i5_54/rzai2/rzai2compipv4ipv6.htm

There are currently two versions of **Internet Protocol** (IP): IPv4 and IPv6 (upgrade of IPv4).
IP specifies the technical format of packets and the addressing scheme for computers to communicate over a network.
IPv4 uses a **32-bit** (4 bytes) address scheme allowing for a total of **2^32 different addresses** (over 4 billion addresses). Numeric addresses (**four 1 byte decimal numbers** separated by a dot). Example: **168.0.0.13**.
IPv6 addresses are **128-bit** (16 bytes) IP addresses written **in hexadecimal numbers separated by colons**. Example: **3ffe:1900:4545:3:200:f8ff:fe21:67cf**.

	IPv4	IPv6
address length	32-bit (4 bytes)	128-bit (16 bytes)
example	**168.0.0.13**	**3ffe:1900:4545:3:200:f8ff:fe21:67cf**
format	decimal	hexadecimal
capable of addresses	over 4 billion (2^32)	2^128 ≅ 3.4 * 10^38 (2^96 times more than IPv4)
address types	unicast (one-to-one packet transmission), multicast (to multiple hosts, optional but often implemented), broadcast (transmission of a packet to all hosts)	unicast, multicast (part of the base specification of IPv6), anycast (sending packet to any one host, doesn't matter which one)
mobility	slower fixed host to mobile host communication	unlike IPv4, mobile IPv6 avoids triangular routing and is therefore as efficient as native IPv6
security	not designed to be secured, IPsec added later	IPsec (Internet Protocol security) - authenticates and encrypts packets of data sent over the network

37. Multimedia technologies used in information systems.

A computer hardware/software system used for
-Acquiring and Storing
-Indexing and Searching
-Manipulating (editing and quality enhancement)
-Distributing
-Protecting
-large amount of visual information

Consist of Images, video, animations, and associated meta-data

Examples:
Web Media Search Engines
Mobile Multimedia Portals
Digital Libraries for Large Organizations
e-Services
Passenger information systems (trams, buses, subway, railway)
Public security (monitoring systems, rescue services, public order services)
City, campus, institution information systems
Systems of virtual cities, districts, municipalities
Systems providing specific information to the client

38. Processing and access to multimedia data.

39. Designing of multimedia interface of computer applications.

40. Methods, techniques and tools used for designing and construction of mobile systems.