



# **PXL-DIGITAL**

PXL - IT

## **Data Advanced Deel 3: MACHINE LEARNING**

**Lector(en)**

Heidi Tans

Sam van Rijn

Julie Vranken

# 0. ML: Doel

De student:

- kan de werkwijze van ML uitleggen
- kent het verschil tussen supervised en unsupervised ML
- kent het verschil tussen classificatie, regressie en clustering
- kan op basis van een probleem identificeren welk type ML erop toegepast kan worden
- kan binnen classificatie decision trees en naive bayes algoritmes implementeren
- kan binnen regressie linear regression algoritmes implementeren
- kan ML algoritmes toepassen op classificatie en regressie problemen
- kan evaluatiemetriecken toepassen op een classificatie en regressie probleem
- kan uitleggen waarom de dataset opgesplitst wordt in een training en testset
- kan een dataset voorbereiden voor ML op toe te passen

Extra items (FYI)

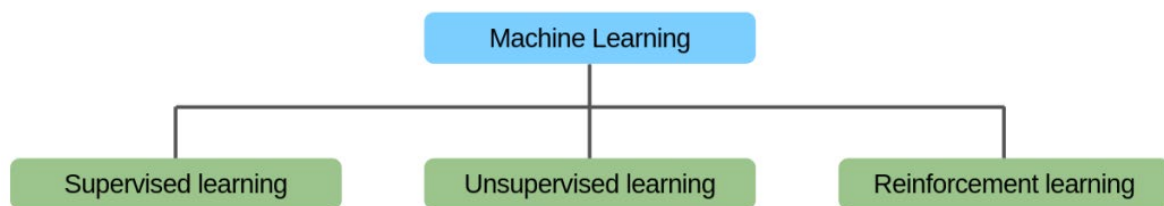
- SVM algoritme
- Entropie / information gain
- Implementatie van clustering en recommendations

# 1. Inleiding

Machine Learning is zo'n term die je tegenwoordig overal hoort. Wat is het nu precies? Moet je een wiskundige/statistische achtergrond hebben om aan ML te doen?

Machine Learning (ML) leert machines zelf taken uit te voeren. Zo simpel is het. De complexiteit komt met de details. ML komt in principe neer op het vermogen om zich aan te passen aan nieuwe situaties. Iedere situatie brengt informatie voort en ML wordt gebruikt om hier patronen in te ontdekken en te gebruiken. De machine stelt door middel van algoritmes "zelf" bepaalde regels op, om bepaalde input te koppelen aan bepaalde output. ML is hierdoor een onmisbaar onderdeel van het onderzoeksgebied van kunstmatige intelligentie, omdat ML ervoor zorgt dat een machine kan evolueren.

Er zijn 3 soorten waarop een machine kan leren:



In deze introductie nemen we 2 soorten Machine Learning onder de loep: Supervised en Unsupervised. Van elke soort bestuderen we onderstaande categorieën:

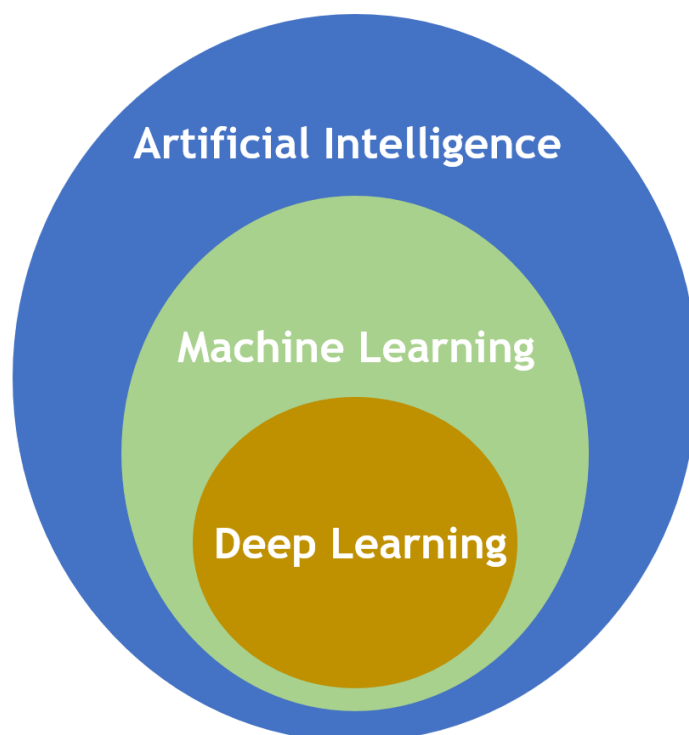
- Supervised Machine Learning
  - Classificatie
  - Regressie
- Unsupervised Machine Learning
  - Clustering

Verder gaan we in op Recommendations: algoritmes die relevante aanbevelingen doen aan gebruikers.

## 2. AI, ML & Deep Learning

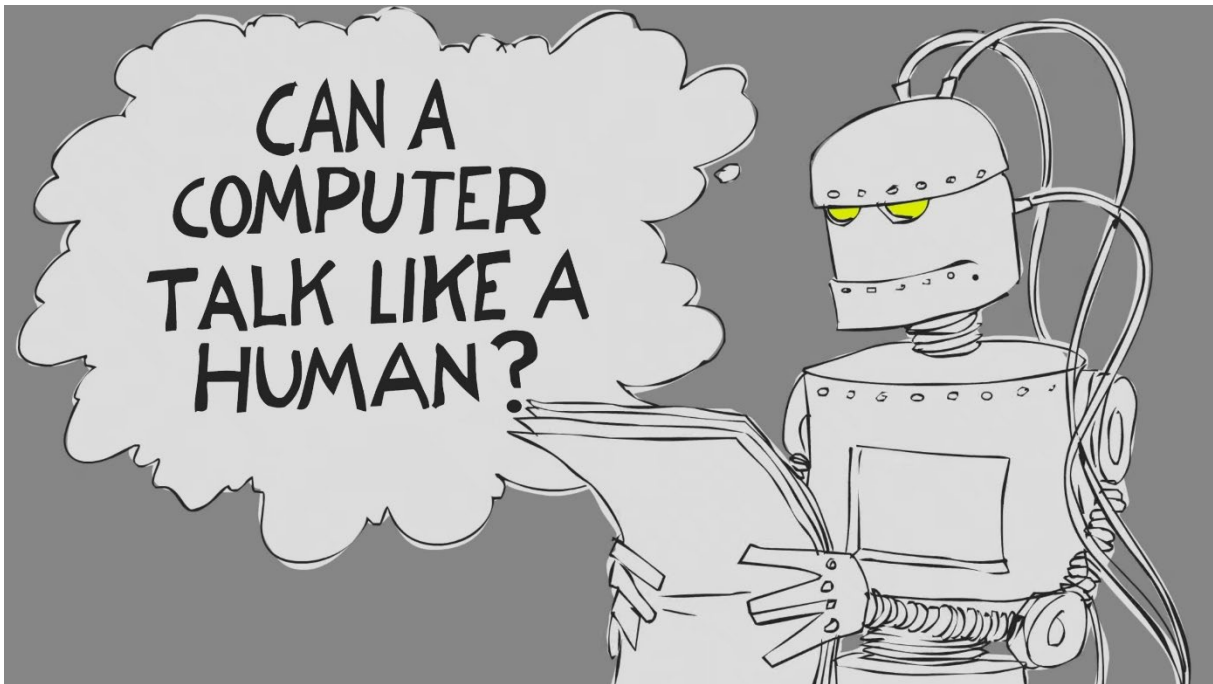
Vooraleer we het domein van Machine Learning verder betreden, is het belangrijk om duidelijk te definiëren wat we bedoelen als we het over deze term hebben. Artificial Intelligence (AI), Machine Learning (ML) en Deep Learning (DL) zijn namelijk “buzzwords” die je vaak samen hoort. Maar hoe verhouden deze drie begrippen zich nu ten opzichte van elkaar?

- **Artificial Intelligence (AI):** elke techniek die computers in staat stelt om menselijke intelligentie na te bootsen
- **Machine Learning (ML):** een subset van AI die machines in staat stelt om taken te verbeteren met “ervaring” zonder expliciet geprogrammeerd te zijn
- **Deep Learning (DL):** een subset van ML dat gebruik maakt van het concept van neurale netwerken om complexe problemen op te lossen



### 3. Historiek

In 1950 omzeilde Alan Turing het traditionele debat over de definitie van intelligentie door een praktische test voor computerintelligentie te introduceren. Deze test staat eenvoudigweg bekend als de Turing Test. Hierin wordt het vermogen getest van een machine om intelligent gedrag te vertonen dat gelijkwaardig is aan -of niet te onderscheiden is van- dat van een mens.



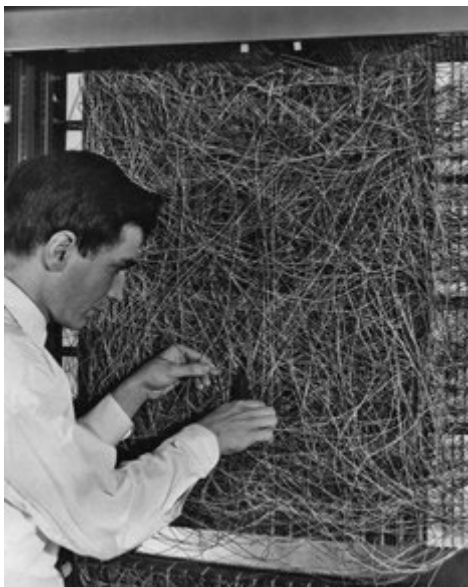
Arthur Samuel, een werknemer van IBM, is degene die de term *machine learning* introduceerde (rond 1952). Hij werd als werknemer van zijn bedrijf gezien als een autoriteit en pionier op het gebied van gaming en kunstmatige intelligentie. Hij gebruikte hier een schaakspel voor, dat steeds “slimmer” werd al naar gelang het meer speelde. Het spel onthield winnende zetten en gebruikte deze in zijn eigen partijen.

Dit was het begin van de uitwerking van het concept van *machine learning* in het algemeen, en van de schaakcomputer in het bijzonder: het wordt dan ook regelmatig aangemerkt als zijnde de grootvader van Deep Blue, de IBM-computer die in 1997 schaakkampioen Garri Kasparov versloeg.



Na Samuel volgden nog veel innovaties die allen verder gingen op het vinden van algoritmes die zichzelf leren aan de hand van de verzamelde data, en die op basis hiervan in staat zijn tot zelfstandige besluitvorming.

Een hoogtepunt hiervan vond plaats in 1958, het jaar waarin Frank Rosenblatt de Perceptron ontwierp. Dit was het eerste kunstmatige neurale netwerk - een kopie van het menselijke neurale netwerk - wat de basis vormde voor meer menselijke intelligentie in computers. Aan de hand van stimuli (input) werd een analyse uitgevoerd, en vervolgens omgezet in een reactie (output).



Ook noemenswaardig is het project van studenten van Stanford University in 1979, die tot de ontwikkeling van de zogenaamde 'Stanford Cart' leidde: een bewegende robot die in staat was om zelfstandig door een kamer te bewegen en onderweg obstakels te ontwijken. Een verre voorvader van zelfrijdende auto's en robotstofzuigers, dus.



## 4. Machine Learning problemen herkennen

Door een aantal voorbeelden te overlopen, krijg je geleidelijk aan intuïtief voeling wat ML is en waar het kan gebruikt worden. Bij problemen die zich in de toekomst voordoen, denk je na dit deel misschien ook even aan ML om jou te helpen met het vinden van een oplossing. ML blijkt de drijvende kracht te zijn achter heel wat applicaties uit het dagelijkse leven die wij eigenlijk “vanzelfsprekend” noemen.

### Een typisch, klassiek ML voorbeeld

Je bent een “alien” en kijkt naar de bevolking ergens op de wereld en je weet dat er 2 soorten mensen op de wereld zijn: zeg man en vrouw. Maar je kent de karakteristieken om te differentiëren tussen man en vrouw niet.

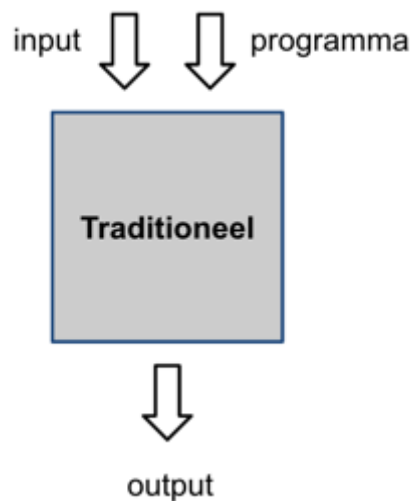
Maar je kunt wel zien dat mensen die groter zijn dan bvb 1.8 tot de groep van de mannen behoren. De andere groep zijn dan de vrouwen.



*Je zou dit probleem kunnen oplossen op volgende “klassieke” manier:*

```
if lengte > 1.8:  
    geslacht = "man"  
else:  
    geslacht = "vrouw"
```

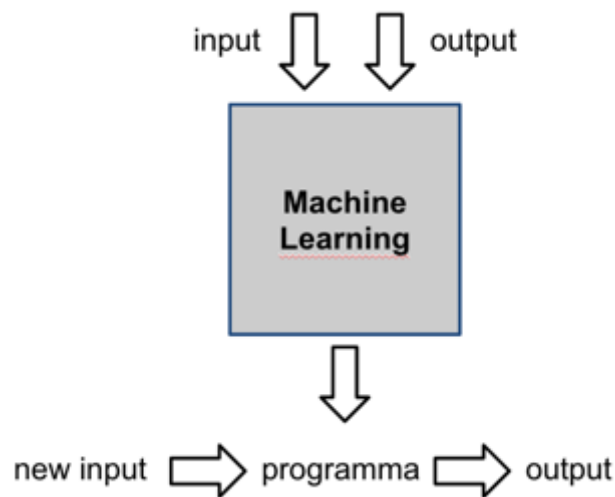
Je steekt data én een programma in een computer en je verkrijgt output; dit is een benadering van het probleem op basis van regels; op basis van programmatielogica.



***ML benadert dit probleem op een andere manier:***

Die alien kidnapt jou en jij kan per persoon aanduiden of het een man of een vrouw is. Nadat die alien geobserveerd heeft hoe jij een 100- of 1000-tal mensen classificeert als man of vrouw (= dit noemen we de training data set); leert hij “intuïtief” hoe te differentiëren tussen geslachten.

Deze ML benadering is vrij gelijklopend met “hoe” mensen leren; hoe het menselijk brein werkt. Hoe leerde je vroeger als kind? Je herkende patronen. Je werd voor langere tijd blootgesteld aan bepaalde fenomenen en je leerde door ervaringen.



**“Definitie ML”:** ML is het proces waarbij een computerprogramma of systeem in staat is te leren hoe een taak uit te voeren en dit door ervaring. Ervaring voor een computer is data!

Wat is het verschil met een Rule based aanpak?

**Rule based:** je past een aantal statische regels toe op de huidige context (tijdstip, dag, ...).

**ML:** Ook in de ML benadering pas je een aantal regels toe maar het verschil is dat deze regels (automatisch) geüpdatet worden op basis van nieuwe data.

**Benadering mbv ML:** Deze benadering bestaat uit 3 stappen:

1. Verzamel een grote set data ivm mensen (10 of zelfs 100 verschillende variabelen zouden mensen kunnen typeren) en blijf deze ook verzamelen op een continue manier.
2. Gebruik een algoritme dat “zelfstandig” het verband vindt tussen deze variabelen en het geslacht van een mens
3. Update dit verband voortdurend mbv nieuwe data

## Voorbeeld

- Netflix kan hun service personaliseren naar elke individuele gebruiker toe want zij lijken te weten waar wij in geïnteresseerd zijn.
- Google Maps: we kunnen er niet meer zonder; deze service maakt je dagelijks leven zoveel makkelijker.
- Spelling checker
- e-mail programma: bepaalde e-mails worden als “spam” geclassificeerd. Hoe komt dit?
- De zelfrijdende auto die geheel autonoom van punt A naar B kan, onderweg rekening houdend met alle variabelen - zoals over verkeer, omstandigheden, obstakels, verkeersregels, etc.



- Bio-informatica waarmee onder meer eiwitfuncties voorspeld kunnen worden
- Natuurlijke taalverwerking zodat bvb de functie en betekenis van woorden in een zin bepaald kunnen worden en effectievere vertaal- en schrijfprogramma's ontworpen kunnen worden.

- De slimme thermostaat: Deze slimme thermostaat voor in huis leert van de gebruiker die de thermostaat bedient. Wanneer deze de thermostaat bijvoorbeeld iedere maandag, woensdag en vrijdag rond een uur of zes 's avonds op 21 graden zet, dan leert de thermostaat zich dit schema aan. Hierdoor zorgt hij ervoor dat het iedere maandag, woensdag en vrijdag rond zes uur 's avonds 21 graden is; zonder tussenkomst van de gebruiker.



- Gezichts- en stemherkenning die aan de hand van visuele of audio-input bepaalde kenmerken kan opslaan en dit matchen aan bijpassende output. Een gezichtherkenningssysteem is hier een uitstekend voorbeeld van, waarbij input van talloze bewakingscamera's uitgelezen wordt om gelijkende matches te vinden met een vooraf opgegeven profiel.
- Data ivm “user clicks and views”. Deze worden gebruikt om aanbevelingen te doen over dingen die een gebruiker zou kunnen willen of nodig heeft.

Deze voorbeelden lijken een soort intelligentie van “hun eigen” hebben. Maar het is ML dat hen brengt tot deze zogeheten intelligentie.

Voor ML heb je dus veel data nodig. Gelukkig wordt er tegenwoordig constant data verzameld vandaar dat ML een heel breed toepassingsgebied heeft.

Dus... de volgende keer als zich een probleem voordoet waarbij je de computer een taak dient te leren en je hebt veel data, stel jezelf ook eens de vraag of ML dit probleem misschien kan oplossen ipv rechttoe rechtaan programmeertechnieken toe te passen.

***Wanneer gebruik je ML dan? Enkele vuistregels:***

- Wanneer het moeilijk is om het probleem te gieten in regels omwille van het grote aantal variabelen die de outcome beïnvloeden en de moeilijkheid om de relatie tussen de variabelen te snappen
- Een voorwaarde om ML benadering toe te passen is dat je beschikt over een grote set historische data
- De patronen of relaties tussen de data zijn dynamisch (blijvend veranderend onder andere omstandigheden)

Let op: ML is vaak aantrekkelijk om toe te passen maar het mag zeker niet zomaar vanaf nu de default keuze worden.

## 5. Het ML proces

Op basis van het voorgaande, kunnen we beslissen of we bij een specifiek probleem voor de ML aanpak gaan of niet. Wanneer we ons probleem met ML gaan oplossen volgen we een gestructureerd plan om tot de oplossing te komen

- 4.1 Bepaal welk type ML probleem we hebben
- 4.2 Gebruik de data die je hebt
- 4.3 Pas een standaard algoritme toe op jouw data
- 4.4 Evalueer de performantie van het gekozen algoritme

Wanneer je deze 4 stappen volgt, kan je elk ML probleem oplossen.

### a. Type ML – problemen

Het type ML - probleem is onder te verdelen in 2 grote categorieën:

**Supervised Machine Learning en Unsupervised Machine Learning.**

Bij **Supervised Machine Learning** hebben we training data voorhanden waar zowel **input als output** aanwezig is. Bijvoorbeeld we willen een persoon categoriseren als “male” of “female”. De data die we gebruiken om een label te plakken op deze persoon bevat een aantal personen en hun karakteristieken maar **ook het juiste label** (juiste outcome: male of female) is aanwezig.

In meer wiskundige bewoordingen:

- De inputvariabelen  $(x_1, x_2, \dots, x_n)$  alsook de outcome variabele  $y$  is aanwezig in de training data set
- Het algoritme is getraind om de functie  $f$  te bepalen die het verband tussen beide geeft nl  $y = f(x_1, x_2, \dots, x_n)$
- Mbv deze functie voorspellen we op basis van een nieuwe input  $(x_1, x_2, \dots, x_n)$  de outcome  $y$
- Verder wordt de functie  $f$  continu bijgestuurd

Bij **Unsupervised Machine Learning** beschikt het algoritme over training data die enkel voorzien is van input (geen output).

### Voorbeeld

Netflix die jou een nieuwe serie aanraadt. Hoe gaat dit in zijn werk? Netflix plaatst jou in een groepje (cluster: vooraf niet gedefinieerd) met andere gebruikers die ongeveer dezelfde series als jij bekeken. Deze gebruikers hebben die nieuwe serie ook bekeken en positief beoordeeld.

In meer wiskundige bewoordingen:

- Enkel de inputvariabele  $(x_1, x_2, \dots, x_n)$  is aanwezig in de training data set (geen outcome  $y$ )
- Het algoritme brengt patronen, structuren, groepen, ... naar voren die op voorhand niet opgenomen zijn in de dataset.



## **b. Data**

Wanneer je voor ML gekozen hebt, heb je een grote hoeveelheid historische data nodig. Deze data kan numeriek van aard zijn maar tevens beschikbaar zijn in ongestructureerde tekst, video's, foto's, ... Om deze data te gebruiken in een ML context is het nodig deze data om te zetten naar betekenisvolle numerieke data. Bvb een foto: je kan de hoogte en breedte van de foto gebruiken en een numerieke waarde die de kleurintensiteit van elke pixel in de foto weergeeft. Het bepalen van betekenisvolle kenmerken (en die omzetten in numerieke waarden) is cruciaal voor het oplossen van het probleem.

## **c. ML - algoritme**

Tot slot gebruik je een algoritme om in de verzamelde data (historische data) patronen te ontdekken; patronen in de vorm van regels.

Deze regels tonen de verbanden tussen variabelen aan. Alle regels samen die het algoritme gevonden heeft, wordt een model genoemd. In het voorbeeld om e-mail te classificeren als spam of niet, representeert het model de relatie die er is tussen bepaalde tekst in een e-mail en het al dan niet spam zijn. Een model kan een wiskundige vergelijking zijn.

Welke algoritme er gekozen wordt, hangt af van het type probleem:

In tegenstelling tot wat doorgaans gedacht wordt, is het uitvoeren van het algoritme niet het meest bepalende deel in het oplossen van ML probleem aangezien heel wat programmeertalen “build in packages” hebben waarin deze algoritmes voorzien zijn. Er bestaan vuistregels voor het beslissen wanneer welk algoritme toe te passen. Je kan ook via “plug and play” meerdere bestaande algoritmes uitvoeren op jouw data. De setup van het probleem, het klaarmaken van de data en het juist presenteren van data en output, vergt vaak het meeste energie.

## d. Evaluatie

Het evalueren van een ML - algoritme is een essentieel onderdeel van elk project. Hoe kunnen we het “succes” van een algoritme berekenen? Hoe weten we wanneer we ons model voldoende hebben getraind en we tevreden kunnen zijn over het resultaat? We moeten immers een notie hebben van hoe goed het algoritme presteert op ongeziene gegevens.

Hiervoor zijn er “performance metrics” ontworpen. Zo zijn er tientallen verschillende statistieken ter beschikking voor de diverse ML - problemen.

Hieronder lijsten we een aantal van de meest populaire evaluatie metrieken op, die in de praktische voorbeelden van hoofdstuk 6 uitvoerig aan bod komen.

Voor classificatie (zie hoofdstuk 6.1):

- Accuracy
  - Nauwkeurigheid van ons model
  - Hoeveel voorspellingen heeft ons model correct gedaan?
  - $\text{Number of correct predictions} / \text{Total number of predictions}$
  - $(TP + TN) / (TP + FP + TN + TP)$
- Recall
  - Welk deel van de werkelijk positieven zijn correct voorspeld?
  - $TP / (TP + FN)$
- Precision
  - Welk deel van de positieve voorspellingen waren correct?
  - $TP / (TP + FP)$
- F1
  - $(\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$

Deze 4 bovenstaande metrieken worden gebruikt om de accuraatheid van een model te meten en worden berekend aan de hand van de true positives (TP), true negatives (TN), false positives (FP) en false negatives (FN). Deze laatsten worden samengevat in onderstaande tabel, die ook wel de “confusion matrix” genoemd wordt.

		PREDICTED LABEL	
		Negative	Positive
TRUE LABEL	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

Voor regressie (zie hoofdstuk 6.2)

- MSE (Mean Squared Error)
  - Deze metriek berekent het gemiddelde van de gekwadeerde errors (dit is het verschil tussen de geschatte en de actuele waarden).

In werkelijkheid willen we modellen bouwen die niet te vaak grotere fouten genereren. In een ideale wereld bouwen we een 100% accuraat model... in de praktijk is dit echter niet haalbaar maar proberen we modellen te bouwen waarbij de fouten die er gemaakt worden, zo klein mogelijk zijn. We willen dus een statistiek die grotere fouten harder bestraft dan kleinere. Dit kan met behulp van het kwadraat van de fouten. Dat zal ervoor zorgen dat een grotere fout een veel meer uitgesproken effect zal hebben.

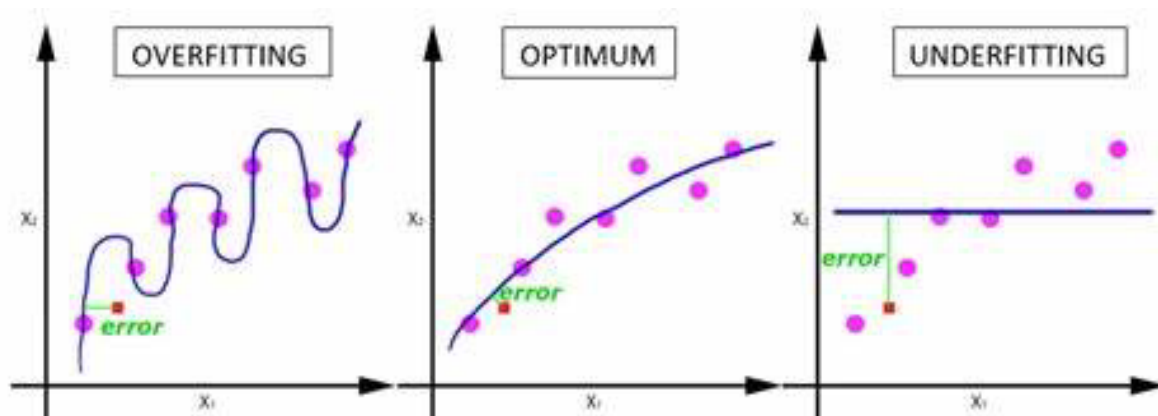
Deze metriecken komen nog aan bod in de jupyter notebooks.

## Overfitting - Underfitting

Het gebruik van evaluatiestatistieken is van cruciaal belang om ervoor te zorgen dat het model correct en optimaal werkt.

“Overfitting” en “Underfitting” zijn de twee grootste oorzaken voor slechte prestaties van machine learning-algoritmen.

- **Overfitting:** Treedt op wanneer het model goed presteert voor een bepaalde set gegevens (bekende gegevens) maar niet in staat is om aanvullende gegevens (onbekende gegevens) of toekomstige waarnemingen betrouwbaar te voorspellen
- **Underfitting:** Treedt op wanneer het model de onderliggende structuur van de gegevens niet adequaat kan vastleggen



## 6. ML problemen uitgewerkt

In volgende paragrafen gaan we eerst dieper in op Supervised Machine Learning. Hier wordt zowel Classificatie (een observatie wordt toegekend aan één of andere categorie = categorische variabele) als Regressie (een continue waarde wordt voorspeld = numerieke variabele) onder de loep genomen.

Bij Unsupervised Machine Learning behandelen we Clustering (items worden gegroepeerd op basis van gelijkenissen).

We sluiten af met een blik op Recommendations.

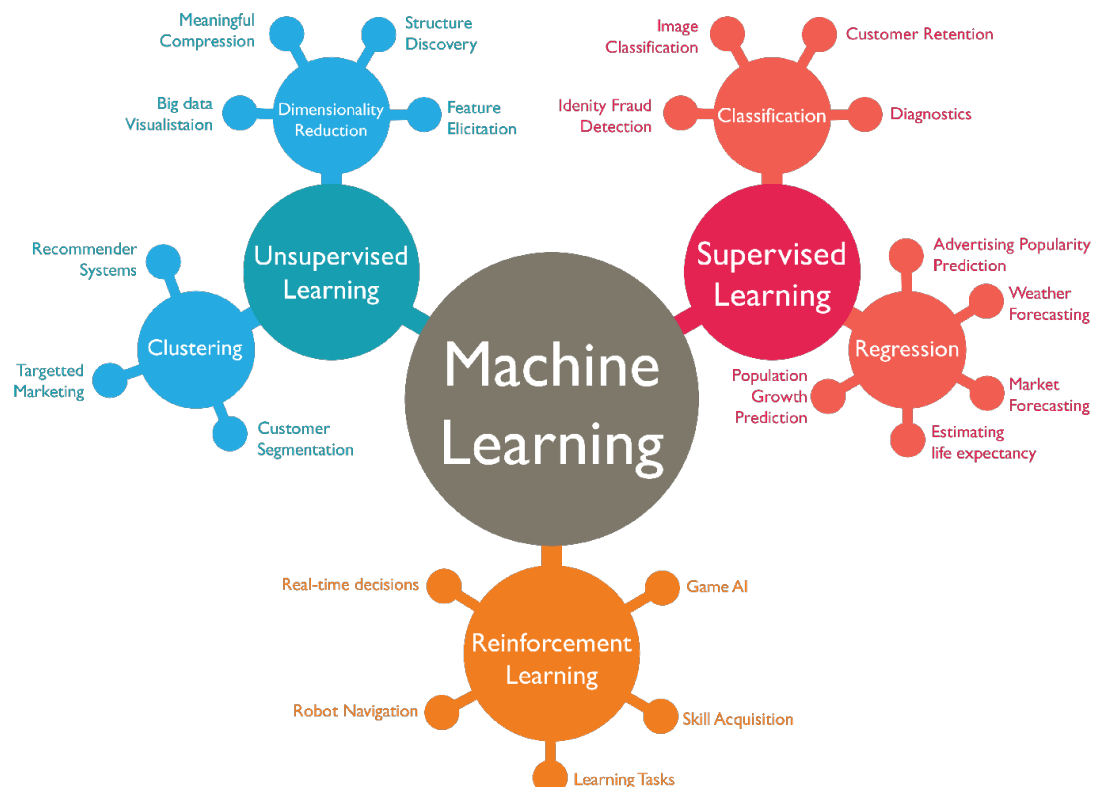
6.1 Supervised Machine Learning: Classificatie

6.2 Supervised Machine Learning: Regressie

6.3 Unsupervised Machine Learning: Clustering

6.4 Recommendations

Hieronder een visueel overzicht van de verschillende takken binnen Machine Learning. Merk op dat we enkel de bovengenoemde topics zullen behandelen.



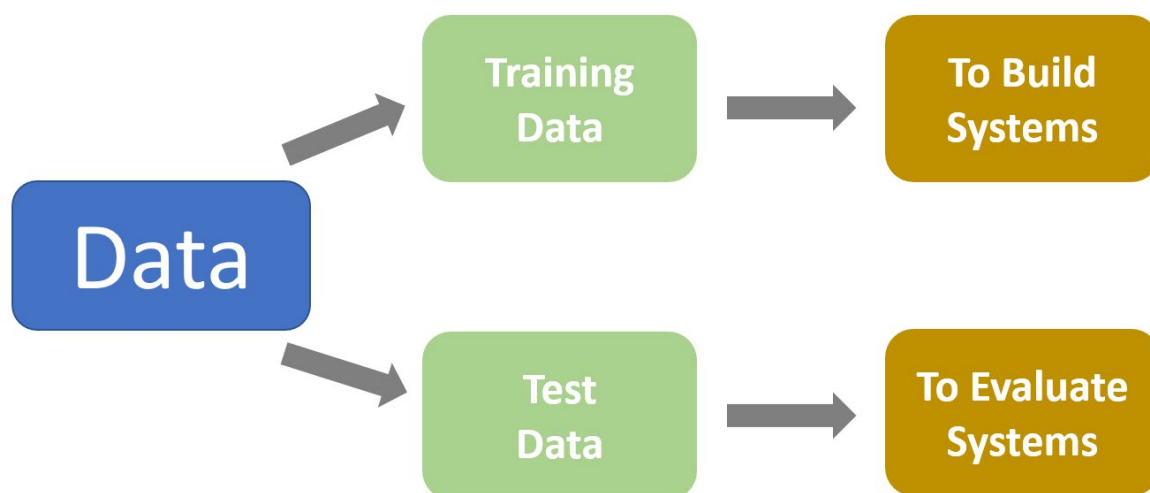
## 6.1 Supervised ML: Classificatie

Alle classificatie problemen hebben een aantal gemeenschappelijke karakteristieken:

- **Doel:** Een object (= problem instance, nieuwe observatie) toekennen aan één of andere categorie (=classificeren)
- **De categorieën (labels)** waaraan we een object toekennen, zijn op voorhand gekend. We onderscheiden verschillende soorten classification:
  - binary classification: outcome kan slechts uit 2 categorieën bestaan bvb Male of Female - Spam of geen Spam - Reclame of niet
  - multi-class classification: outcome kan uit meerdere categorieën bestaan bvb Setosa, Versicolor of Virginica (Iris data set)
  - multi-label classification: per object (instance) worden meerdere labels voorspeld bvb (True female) of (True Male) of (False Female) of (False Male)
  - multi-output classification: multi-class + multi-label samen bvb (zondag, januari)
- **Classifiek** is een algoritme dat de indeling uitvoert en een label geeft aan het object. Beschouw de classifier voorlopig als een black box. Wat binnen in die black box gebeurt, kunnen we ons voorstellen als wiskundige of statistische algoritmes (regels of vergelijkingen).

Voorbeelden van classifiers: Naive Bayes Algoritme, Support Vector Machine Algoritme, Decision Tree algoritme, Logistic Regression algoritme, Linear Discriminant Analysis algoritme, K-nearest neighbor, Random Forests, Logistic Regression, ...

- **Features** (eigenschappen - kolommen in onze dataset): We moeten van onze data features verzamelen die representatief zijn voor het probleem dat je wil oplossen. Belangrijk is dat deze features numeriek van aard zijn. Zelfs als je tekst of foto's gebruikt als data voor je ML probleem, moet je deze gegevens omzetten naar een numerieke grootte.
- **Training:** De classifier gebruikt (een deel van de) data (training data) waarvan de categorie reeds gekend is (Supervised ML). Van deze training data leert onze classifier hoe nieuwe data te classificeren. Dit noemt men "het model trainen". Het tracht regels en patronen af te leiden van de training data. De training data kunnen we zien als een tuple van features én label.
- **Testing:** test het model gebruik makend van test data. Soms bestaat deze testdata uit een deel van de originele dataset die niet gebruikt is in de training fase. Indien er niet genoeg data voorhanden is, wordt nieuwe data gebruikt als testdata. In deze fase classificeren we de testdata in één van de categorieën. De efficiëntie van onze classifier hangt af van de juistheid van het classificeren van objecten die niet in de training data zitten.



In onderstaande secties bespreken we 3 bestaande classificatiealgoritmes. Naast een toelichting van de theoretische en/of wiskundige achtergrond zullen we elk van deze algoritmes ook uitwerken op een aantal voorbeelden in een jupyter notebook.

6.1.1 Decision Tree algoritme

6.1.2 Naive Bayes algoritme

6.1.3 Support Vector Machine algoritme

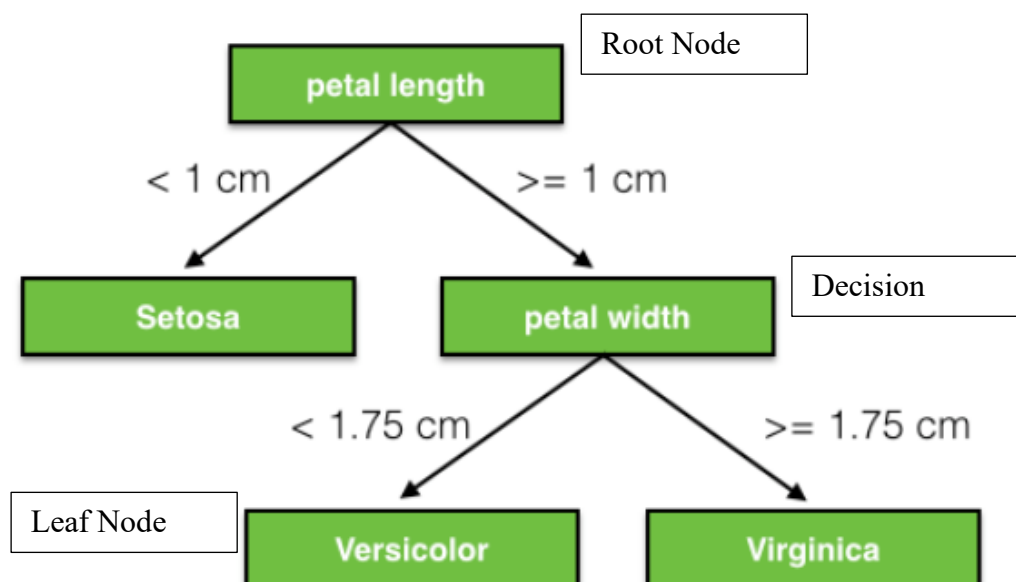


### 6.1.1 Decision Tree algoritme

Decision trees zijn één van de eenvoudigste en meest populaire algoritmes die worden gebruikt voor het voorspellen van categorische gegevens. Het is als een stroomdiagram, maar met een structuur van een boom.

De interne knooppunten (decision nodes/internal nodes) van de boom vertegenwoordigen een test of een vraag over een kenmerk (feature). Elke vertakking is de mogelijke uitkomst van de gestelde vraag. Het knooppunt waar de boom begint staat bekend als de 'Root Node' en het eindknooppunt (leaf nodes/terminal nodes) duidt op een klasselabel.

De pijlen in de boom duiden dus een beslissing aan, waarbij ieder blad één van de uitkomsten (categorisch) is.



## **Laten we even inzoomen op hoe het algoritme "Decision Tree" achterliggend werkt**

Indien we ons toespitsen op de basis, zijn beslisboomalgoritmes niets anders dan een reeks if-else uitspraken die worden gebruikt om een resultaat te voorspellen op basis van een dataset. Deze flowchart-achtige structuur helpt ons bij het nemen van beslissingen.

Het idee achter een decision tree is om een dataset op te delen in kleinere datasets op basis van de beschrijvende features totdat we een set bereiken die klein genoeg is en gegevenspunten bevat die onder één label samengebracht kunnen worden.

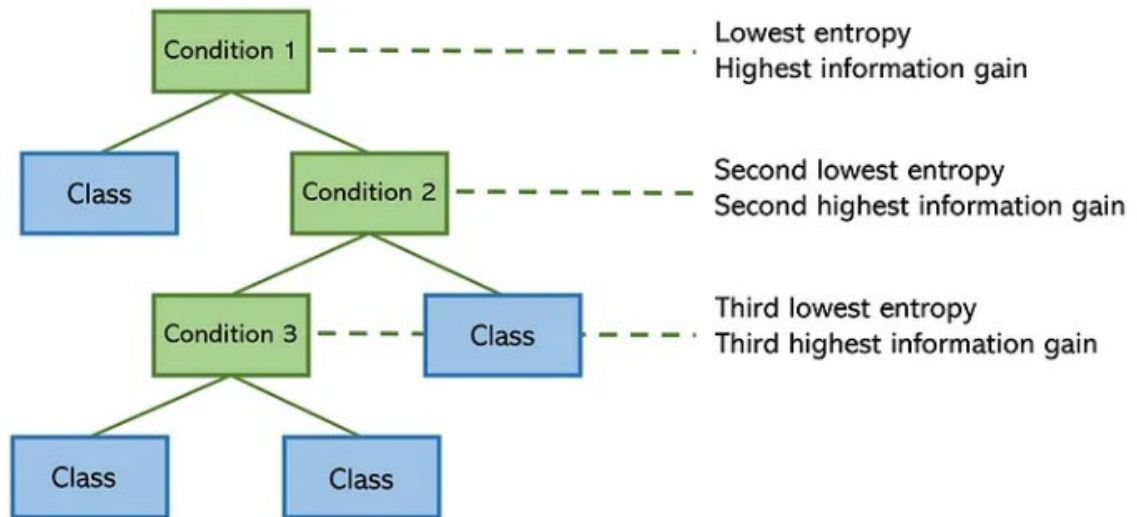
Entropie en Information Gain zijn 2 belangrijke statistieken die worden gebruikt bij het bepalen van de relevantie van de besluitvorming bij het samenstellen van een beslissingsboom.

Wanneer we te maken hebben met grote hoeveelheden gegevens, geeft elk punt in de dataset ons verschillende informatie over het uiteindelijke resultaat. Dat betekent dat er altijd enige onzuiverheid in onze gegevens zit. Deze onzuiverheid kan worden gekwantificeerd door de entropie van de gegevens te berekenen.

Aan de andere kant geeft elk datapunt verschillende informatie over de uiteindelijke uitkomst. Information Gain geeft aan hoeveel informatie een bepaalde variabele/functie ons geeft over het uiteindelijke resultaat.

Entropy en Information Gain zijn twee hoofdconcepten die achterliggend worden gebruikt bij het construeren van een beslissingsboom enerzijds om de knooppunten te bepalen, anderzijds om de beste manier te achterhalen om de data op te splitsen in kleinere datasets.

In een beslissingsboom zullen de takken dus op die manier gekozen worden dat dit leidt naar een zo snel en efficiënt mogelijke uitkomst (ook wel de target label genoemd).



## 6.1.2 Naive Bayes algoritme

Naive bayes is een probabilistisch algoritme gebaseerd op voorwaardelijke kansen en de “regel van bayes”. We zullen de werking van dit algoritme toelichten aan de hand van een aantal voorbeelden.

### Voorbeeld: Weather Prediction

Hieronder vind je een training dataset van het weer en de bijhorende doelvariabele ‘Play’ (die de speelmogelijkheden suggereert). Nu moeten we classificeren of men speelt of niet op basis van de weersomstandigheden.

Stap 1: Converteer de dataset in een frequentie tabel

Stap 2: Creëer een waarschijnlijkheidstabel door het berekenen van de verschillende kansen (bv. de totale kans op ‘Play’ is 0.64, de totale kans op ‘Overcast’ is 0.29).

Stap 3: Gebruik de regel van Bayes voor het berekenen van de voorwaardelijke kansen voor iedere klasse. De klasse met de hoogste voorwaardelijke kans is de uitkomst van de predictie.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
All	5	9
	=5/14	=9/14
	0.36	0.64

*Probleemstelling: De spelers zullen spelen als het weer 'sunny' is. Is dit statement correct?*

We kunnen dit oplossen door middel van de voorgestelde methode van voorwaardelijke kansen. Hiervoor berekenen we  $P(\text{Yes} \mid \text{Sunny})$  en  $P(\text{No} \mid \text{Sunny})$  en kijken welke de hoogste kans heeft

$$P(\text{Yes} \mid \text{Sunny}) = P(\text{Sunny} \mid \text{Yes}) * P(\text{Yes}) / P(\text{Sunny})$$

Hier hebben we:

- $P(\text{Sunny} \mid \text{Yes}) = 3/9 = 0.33$
- $P(\text{Sunny}) = 5/14 = 0.36$
- $P(\text{Yes}) = 9/14 = 0.64$

En dus

- $P(\text{Yes} \mid \text{Sunny}) = 0.33 * 0.64 / 0.36 = 0.60$ , hetgeen dus de hogere kans heeft.

## Voorbeeld: Sentiment Analysis

Hoe iemand zich voelt (positief of negatief) kan gemeten worden met een techniek die we Sentiment analysis noemen. Deze techniek bestaat al lang maar is de afgelopen jaren erg populair geworden omdat iedereen te pas en te onpas op social media zijn gevoel, bedenkingen, ... post. Dus is er onnoemelijk veel data voorhanden. De data bestaat vooral uit tweets, meldingen, emoji's, ... Deze data is vrij ongestructureerd maar wel publiekelijk toegankelijk voor iedereen.

Wat doet sentiment analysis: We hebben een commentaar, tweet, .. en we moeten deze classificeren als zijnde ofwel positief ofwel negatief. Features (variabelen, dus onze kolommen) om de data te representeren zijn hier niet zo evident: we hebben ongestructureerde data en voor een classificatieprobleem hebben we nood aan numerieke maatstaven.

Vooraleer aan het effectief classificeren van tweets te kunnen beginnen (op basis van de “regel van Bayes”), moet de ongestructureerde data eerst omgezet worden naar numerieke data. Een mogelijkheid om teksten om te zetten naar numerieke data is dmv “**term frequency representation**”: Dit is een erg populaire manier om tekst om te zetten naar numerieke waarden om zo te kunnen gebruiken in ML algoritmes. We gaan hier eerst even dieper op in.

## Hoe werkt “term frequency representation”?

Men maakt een lijst van ALLE woorden die in ALLE commentaren of teksten of ... kunnen voorkomen.

(woord<sub>1</sub>, woord<sub>2</sub>, ... , ... , ... , ... , ... , ... , ... , ... , ... , ... , woord<sub>N</sub>)

(hallo, dit, zijn, alle, woorden, die, kunnen, voorkomen, in, een, tekst, en, ook, is)

Hiermee kunnen we om het even welke tekst voorstellen door een tupel van getallen. Elk getal stelt het aantal keer voor dat het specifieke woord in het geheel van de training data voorkomt. Dit noemt men de woordfrequentie.

Bv: *(hallo, dit, is, een, tekst, en, een, tekst)*  
*(hallo, dit, zijn, alle, woorden, die, kunnen, voorkomen, in, een, tekst, en, ook, is)*

De testzin (*hallo, dit, is, een, tekst, en, een, tekst*) wordt dan voorgesteld door volgend N-tupel (14-tupel in ons voorbeeld):

$$(1, 1, 0, 0, 0, 0, 0, 0, 0, 2, 2, 1, 0, 1)$$

Hiermee zien we dat “hallo” 1 keer in de zin voorkomt en “tekst” 2 keer in de zin voorkomt. Wanneer een woord niet voorkomt in de zin is de frequentie uiteraard 0. Merk op: Aangezien elk document anders van lengte is, is het mogelijk dat éénzelfde term in lange documenten veel vaker voorkomt dan kortere. Zo wordt de term frequentie vaak gedeeld door de documentlengte (ook wel het totale aantal termen in het document) als een manier om te normaliseren. Voor voorgaand voorbeeld zou dit worden:

 $(1/8, 1/8, 0, 0, 0, 0, 0, 0, 0, 0, 2/8, 2/8, 1/8, 0, 1/8)$

Nu de “ongestructureerde” tweets omgezet zijn in numerieke waarden mbv term frequency representaiton, kan Naive Bayes zijn werk doen om de tweets als positief / negatief te classificeren.

Dit noemen we de “**training fase**”: hier maakt het algoritme (**Naive Bayes** in ons geval) gebruik van commentaren die al gelabeld zijn als zijnde positief of negatief (de training data set) om nieuwe tweets te classificeren.

Income	Outcome
tekst 1 $\xrightarrow{\text{term freq repr}}$ (1,0,3,1,...,0)	<i>Positief</i>
tekst 2 $\xrightarrow{\text{term freq repr}}$ (1,1,2,1,...,1)	<i>Positief</i>
tekst 3 $\xrightarrow{\text{term freq repr}}$ (2,0,1,0,...,0)	<i>Negatief</i>
tekst 4 $\xrightarrow{\text{term freq repr}}$ (0,0,1,1,...,2)	<i>Positief</i>
tekst 5 $\xrightarrow{\text{term freq repr}}$ (1,1,1,1,...,0)	<i>Negatief</i>
tekst 6 $\xrightarrow{\text{term freq repr}}$ (1,2,0,0,...,0)	<i>Positief</i>
...	
tekst 1000 $\xrightarrow{\text{term freq repr}}$ (1,0,2,2,...,3)	<i>Positief</i>

Mbv de training data berekent het algoritme de kans op een positieve tekst en wel als volgt:

$$P_o = \frac{\# \text{ positieve comments in de training data}}{\text{totaal \# comments in de training data}}$$

en owv de complementregel weten we dan dat de kans op een negatieve tekst gelijk is aan  $1 - P_o$ .

Neem voor de verdere uitrekening  $P_o = 55\%$  en  $1 - P_o = 45\%$



Teksten zijn samengesteld uit zinnen, die op hun beurt uit woorden bestaan. Van deze individuele woorden wordt vervolgens de positiviteitskans berekend.

We berekenen bvb de positiviteitskans van het woord “blij”:

$$P_{blij} = P(Pos\ tekst|blij) = \frac{\text{Som van alle woordfreq van blij in pos teksten}}{\text{Som van alle woordfreq van blij in hele training data}}$$

Dit is dus een maatstaf die ons aangeeft van al de keren dat “blij” voorkomt in een tekst, hoe vaak het voorkomt in een positieve tekst.

Analoog kunnen we nu de negativiteitskans van hetzelfde woord berekenen als zijnde:  $1 - P_{blij}$

Dit doen we nu voor elk woord dat voorkomt in de training dataset; wat ons bvb volgende tabel oplevert:

	<i><b>Positiviteitskans</b></i>
<b>blij</b>	0.92
<b>geweldig</b>	0.95
<b>pxl</b>	0.81
<b>data</b>	0.99
<b>saai</b>	0.1
<b>moeilijk</b>	0.33
...	

Hiermee zijn we aanbeland in de **testfase**. We passen voorgaande toe op testdata om nieuwe teksten te classificeren als positief of negatief.

We nemen een nieuwe tekst bestaande uit een aantal woorden en het algoritme gaat een positiviteits- en negativiteitsscore voor deze tekst berekenen op basis van de waarden uit de training data.

We berekenen dit voor bvb de zin “data is geweldig”

$$PosScore = P_{data} * P_{geweldig} * P_O = 0.99 * 0.95 * 0.55 = 0.51$$

$$NegScore = (1 - P_{data}) * (1 - P_{geweldig}) * (1 - P_O) = 0.01 * 0.05 * 0.45 = 0.000225$$

We vergelijken deze 2 scores en geven het label met de hoogste score aan de nieuwe zin.

Merk hierbij op dat niet betekenisvolle woorden (zoals is, het, en, een, ...) genegeerd worden.

### **Waarom wordt dit NAIVE Bayes genoemd?**

Wanneer we de algemene positiviteitsscore berekenen, gebruiken we **onafhankelijk** de positiviteitsscores van de verschillende woorden in het comment. We houden geen rekening met het feit hoe de positiviteitsscore beïnvloed wordt wanneer er paren van woorden optreden. Het Naive Bayes algoritme veronderstelt dat elke variabele **onafhankelijk** bijdraagt tot de positiviteitsscore.

Dit is best een grote veronderstelling waarbij je zou kunnen denken dat het Naive Bayes algoritme niet erg krachtig is. Dit is niet waar: Naive Bayes is uitermate aangewezen wanneer je

- weinig training data voorhanden hebt
- wanneer je zelf weinig verdere info hebt of kennis in het domein

## Voorbeeld: Spam Filter

Het naive bayes algoritme wordt onder andere ook gebruikt voor het filteren van e-mails in data-analyse. Als je een e-mail account hebt, zijn we er zeker van dat je al gezien hebt dat e-mails gecategoriseerd worden in verschillende buckets en automatisch gemarkeerd worden als spam, promoties, belangrijk, enz. Is het niet prachtig om te zien dat machines zo slim zijn en het werk voor je doen?

Het concept van een spam filtering is dus eenvoudig: detecteer spam-e-mails van authentieke (niet-spam / ham) e-mails. Om dit te doen, zou het doel zijn om een meting te krijgen van hoe 'spammy' een inkomende e-mail is. De regel van Bayes komt hier om de hoek kijken.

Met deze regel willen we de waarschijnlijkheid dat een e-mails spam is, vinden. Dit op basis van de woorden die voorkomen in de e-mail. We doen dit door de waarschijnlijk van elk woord in de e-mail te berekenen en op die manier een algemene 'spam' metriek te construeren.

### 6.1.3 Support Vector Machine Algoritme

Een support Vector Machine (SVM) is een supervised learning algoritme dat de data verdeelt in groepen met zogeheten hyperplanes door gebruik te maken van support vectors. We zullen de werking van dit algoritme toelichten aan de hand van een voorbeeld.

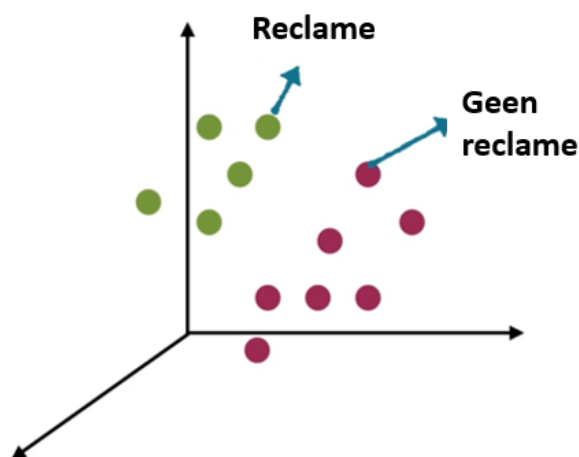
#### Voorbeeld: add detection

Men wil bepalen of een afbeelding (problem instance) een reclame advertentie is of niet. De features zijn tupels van numerieke waarden van die beelden bvb hoogte, breedte, url, tekst op pagina, tekst bij afbeelding, ...



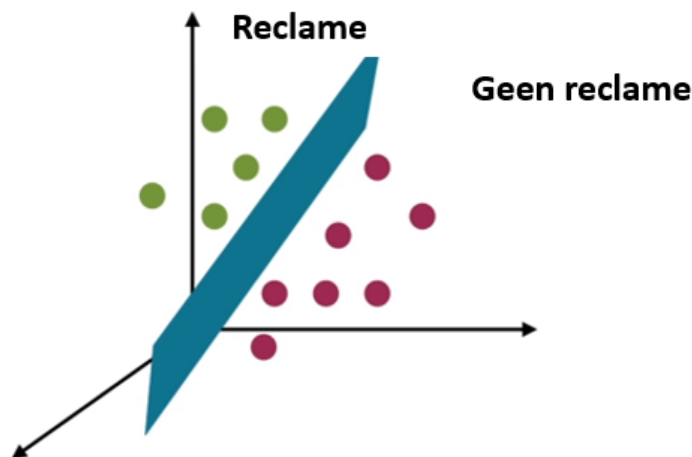
De training data bestaat uit een grote data set van afbeeldingen die reeds het label wel of geen advertentie ontvangen hebben. Aan al deze afbeeldingen is een tuple van N getallen geassocieerd.

Deze punten kunnen voorgesteld worden in een N - dimensionale figuur. Bij veel ML - problemen wordt er gebruik gemaakt van N - dimensionale figuren. In de training fase zetten we alle figuren van onze training data set uit in een N - dimensionaal assenstelsel en we markeren de reclame figuren bvb als groen en de niet - reclame figuren als rood.

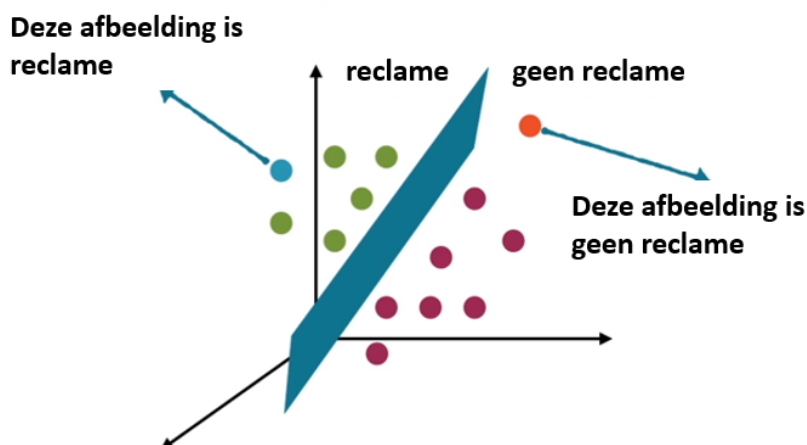


Hoe werkt het Support Vector Machine algoritme?

SVM classifiers zoeken dat vlak dat de datapunten het beste van mekaar scheidt. In ons geval gaat het SVM algoritme op basis van de punten van de training data een **grensvlak** zoeken tussen de 2 groepen afbeeldingen.



Eens deze grens gevonden, kunnen we verder gaan naar de test - fase waar we nieuwe afbeeldingen willen gaan classificeren als reclame of geen reclame. We zetten de nieuwe figuur uit op het 'assenstelsel' en kijken gewoon aan welke kant van de scheiding dit nieuwe punt valt.



SVM kunnen we **enkel** gebruiken voor **binaire** classificatie (dus in 2 categorieën), dit in tegenstelling tot Naïve Bayes waar we kunnen indelen in om het even welk aantal categorieën.

Jupyter notebook: [Machine\\_Learning\\_1\\_Classification\\_Iris.ipynb](#)

Jupyter notebook: [Machine\\_Learning\\_2\\_Classification\\_Titanic.ipynb](#)

## 6.2 Supervised ML: Regressie

Met behulp van regressie trachten we een continue variabele (= output) te voorspellen. Hier verschilt regressie van classificatie waar de output categorisch is. Verder weten we dat deze output afhangt van / beïnvloed wordt door sommige andere variabelen (input).

Dit is wat regressie doet: regressie zoekt de functie die het verband tussen de afhankelijke variabele en onafhankelijke variabelen zo goed mogelijk benadert. Is dit verband lineair, kwadratisch, exponentieel, ... ?

Lineaire regressie is veruit de meest gebruikte en gekende regressie techniek.

Net zoals bij classificatie heeft regressie ook nood aan training data. Deze training data is een grote dataset waarbij zowel input als de output (= de waarde die voorspeld moet worden) aanwezig is (dus supervised Machine Learning).

## Theorie achter eenvoudige lineaire regressie

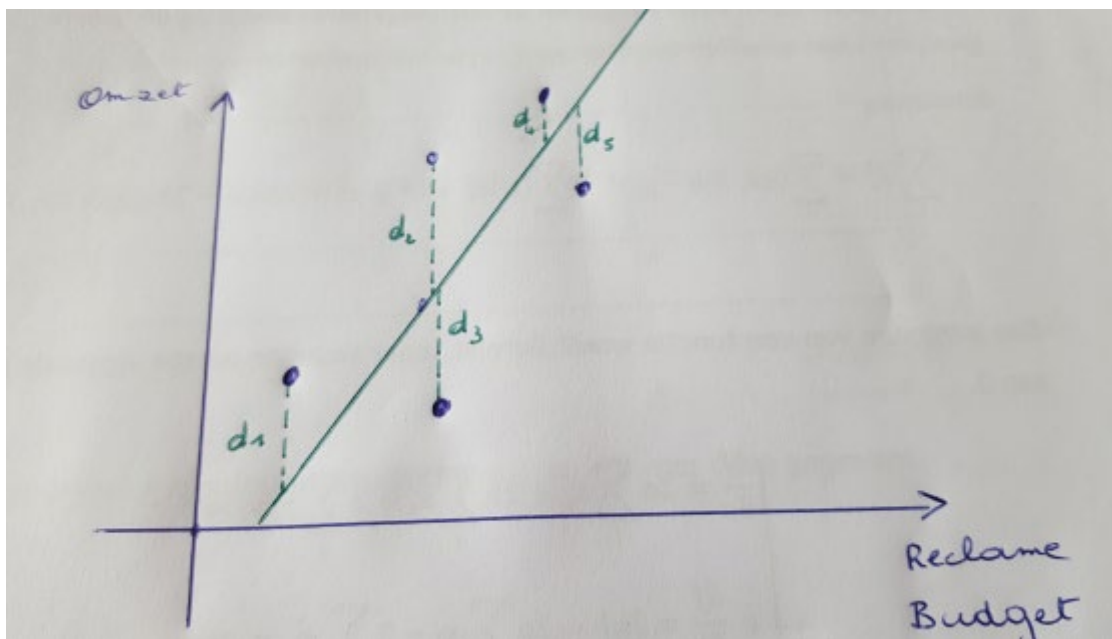
Een belangrijk aspect bij het op de markt brengen van een product is de hoeveelheid geld die aan reclame wordt besteed. Doel is natuurlijk dat reclame maken ook de verkoop van je product stimuleert.

Aan de hand van gegevens over vroegere verkopen en reclamebudgetten kan je met behulp van regressieanalyse het verband tussen deze twee gegevens nagaan.

**Regressie:** we zoeken een verband tussen een afhankelijke variabele en één of meerdere onafhankelijke variabelen

**Lineaire regressie:** het verband tussen de afhankelijke en onafhankelijke variabelen is lineair

**Eenvoudige lineaire regressie:** er is slechts één onafhankelijke variabele



De regressierechte heeft als vergelijking:  $y = ax + b$  met

$y$  = afhankelijke variabele

$x$  = onafhankelijke variabele

$a$  = helling van de regressierechte (richtingscoëfficiënt)

$b$  = snijpunt van de regressierechte met de  $y$  - as

Doel:  $a$  en  $b$  bepalen aan de hand van data; zodanig dat de rechte de gegevens zo goed mogelijk benadert. Het bepalen van  $a$  en  $b$  gebeurt hier via de **methode van de kleinste kwadraten**.

De lineaire regressierechte benadert de 'puntenwolk' nooit exact. Er wordt per gegeven een fout gemaakt. We trachten de totale fout te minimaliseren.

We berekenen  $\sum_i d_i$ , maar aangezien de ene (negatieve) afwijking de andere (positieve) kan opheffen trachten we  $\sum_i d_i^2$  te minimaliseren.

Berekening

$$\sum_i d_i^2 = \sum_i (ax_i + b - y_i)^2 = \sum_i (a^2 x_i^2 + b^2 + y_i^2 + 2ax_i b - 2ax_i y_i - 2by_i)$$

Een minimum van een functie wordt bereikt, daar waar de eerste afgeleide gelijk is aan 0.

$$\begin{cases} \frac{\delta f}{\delta a} = 2a \sum_i x_i^2 + 2b \sum_i x_i - 2 \sum_i x_i y_i = 0 \\ \frac{\delta f}{\delta b} = 2nb + 2a \sum_i x_i - 2 \sum_i y_i = 0 \end{cases}$$

Dit geeft volgend stelsel dat opgelost dient te worden:

$$\begin{cases} \sum_i x_i a + n b = \sum_i y_i \\ \sum_i x_i^2 a + \sum_i x_i b = \sum_i x_i y_i \end{cases}$$

Met als oplossing:  $a = \frac{\sum x_i y_i - n \bar{x} \bar{y}}{\sum x_i^2 - n \bar{x}^2}$  en  $b = \bar{y} - a \bar{x}$



### Voorbeeld

Een winkel in huishoudelijke artikelen voert een vijf maandelijks experiment uit om het effect van reclame op de omzet te bepalen. De resultaten vind je hieronder.

Maand	Uitgaven reclame x (in 10000)	Omzet y (in 10000)
1	1	1
2	2	1
3	3	2
4	4	2
5	5	4

Zoek een kleinste kwadratenbenadering  $y = ax + b$  voor deze gegevens.

$$\bar{x} = 3 \quad \bar{y} = 2 \quad n = 5 \quad \sum_{i=0}^5 x_i y_i = 1 * 1 + 2 * 1 + 3 * 2 + 4 * 2 + 5 * 4 = 37$$

$$a = \frac{37 - 5 * 3 * 2}{55 - 5 * 3^2} = 0.7 \quad b = 2 - 0.7 * 3 = -0.1 \quad \rightarrow \quad y = 0.7 * x - 0.1$$

We passen dit Linear Regression algoritme toe op de Boston Houses dataset om de huisprijs te voorspellen.

## Enkele andere voorbeelden van regressie:

- Wat zal het rendement zijn van een aandeel op een bepaalde datum? Dit rendement kan afhangen van meerdere variabelen (= onafhankelijke variabelen) bvb dag van de week, dag van de maand, het rendement van de afgelopen dagen, andere producten op de markt, ....
- Als de wachttijd vergroot, welk effect heeft dit op klanttevredenheid?
- Wat is de verwachte verkoop van een bepaald product in een bepaalde week?
- voorspellen van “vraag” waaruit “aanbod” bepaald wordt: Het is heel belangrijk voor elk bedrijf om de verkoop op een bepaald punt te kunnen voorspellen. Deze info wordt gebruikt voor het plannen van een stock, het bepalen van het aantal VTE's, het bepalen van de hoeveelheid basisstoffen die moeten besteld worden bij leveranciers, het berekenen hoeveel transport er nodig is, ... De verkoop in een bepaalde periode kan afhangen van de hoeveelheid verkoop in de vorige week, het budget gespendeerd aan marketing, is het vakantie of niet, ...
- gezichtsherkenning: Wanneer je naar een foto van een gezicht kijkt, herken je ogen, neus, mond, ... We willen ook een computer leren om een foto van een persoon te herkennen. Hoe leren we een computer waar de ogen, neus, mond van een persoon op een foto zijn? Wanneer we dit kunnen, kunnen er virtuele kleedkamers gemaakt worden waar je met jouw gezicht bvb een zonnebril kan passen en zien hoe de ene of de andere jou staat. We zoeken bvb de x - en y - waarden van het linkeroog, rechteroog, ... Deze posities hangen bvb af van de relatieve posities in de foto en de karakteristieken van de omliggende pixels van een oog in deze foto. De positie van elk punt kan gevonden worden door een aantal afzonderlijke regressieproblemen.

Jupyter Notebook: Machine\_Learning\_3\_Regression\_Boston.ipynb

## 6.3 Unsupervised ML: Clustering

Clustering is een manier om items te groeperen op basis van “bepaalde maatstaven van gelijkenissen”. We hebben een grote set van objecten en willen deze indelen in groepen zodat objecten die tot dezelfde groep behoren, gelijkenissen vertonen.

Het basisidee om dit te doen is als volgt:

- gebruikers die in dezelfde cluster zitten moeten gelijkenissen hebben (dus de intra cluster gelijkenissen worden gemaximaliseerd)
- gebruikers in verschillende clusters zijn verschillend t.o.v. elkaar (dus de inter cluster gelijkenissen worden geminimaliseerd).

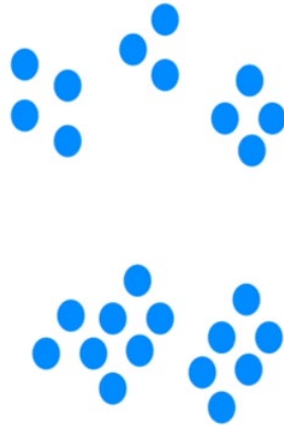
Belangrijk hierbij is dat de **groepen** waarin deze gebruikers ingedeeld worden, **op voorhand NIET gekend** zijn! Dit is anders dan in classificatie: hier begin je met een bepaalde set van gekende groepen waarin je de objecten plaatst. We gebruiken het clustering algoritme om inzicht te krijgen in welke groepen er kunnen bestaan. De op deze manier gevormde clusters bestuderen, levert vaak interessante informatie op wat betreft gelijkenissen binnen de clusters. Deze kennis kan dan gebruikt worden om gerichte acties te voorzien per cluster (promoties voorstellen, producten aanprijzen, ...).

**Voorbeeld:** je hebt een grote groep mensen die actief zijn op social media. Op basis van gemeenschappelijke kenmerken deelt het clusteringalgoritme deze gebruikers op in clusters (groepen) zodanig dat in één cluster de gebruikers “vergelijkbaar” zijn met elkaar. Door deze clusters te bestuderen, kan je later misschien afleiden dat elke cluster één of andere betekenisvolle groep vertegenwoordigt (demografisch, voorkeur van de gebruikers, leeftijd, ...). Wanneer een nieuwe gebruiker zich aanmeldt op social media, dan kan je classificatie gebruiken om deze user toe te kennen tot één of andere groep die ontstaan is door clustering.

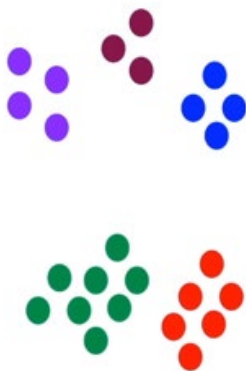
### **Werkwijze:**

Van alle mensen (los van clusters) worden features verzameld: leeftijd, woonplaats, frequentie in het gebruik van bepaalde topics, keywords die vaak gebruikt worden op social media, ...

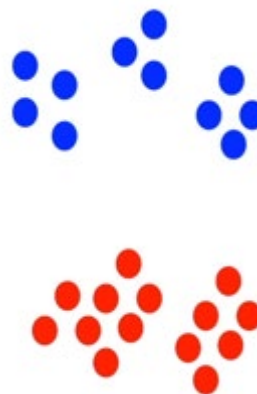
Op deze manier wordt aan elke persoon een  $N$  - tupel van cijfers gekoppeld. Dat  $N$  - tupel stelt een punt in de  $N$  - dimensionale ruimte voor. Het clustering algoritme gaat groepen van mensen zoeken die dicht bij mekaar liggen.



Er zijn meerdere manieren om de puntenwolk op te delen in clusters:



5 clusters



2 clusters

Elke cluster heeft sommige gemeenschappelijke kenmerken maar wij weten nog niet welke dit zijn.

Visueel is het duidelijk wat er gebeurt. In beide manieren zijn de clusters gebaseerd op de afstand tussen gebruikers: we minimaliseren de gemiddelde afstand tussen gebruikers in één en dezelfde cluster en maximaliseren de afstand tussen gebruikers die zich in verschillende clusters bevinden.

Afstand is een abstract begrip in deze  $N$  - dimensionale ruimte, maar in “real life”

kan deze afstand betekenisvol zijn vb. geografisch op dezelfde plaats wonen, voorkeur hebben voor dezelfde dingen, zelfde leeftijd hebben of ...

**Bestaande clustering Algoritmes:**

K - means clustering: dit werken we iets dieper uit op volgende pagina

K - median clustering

Hiërarchical clustering

Density-based clustering

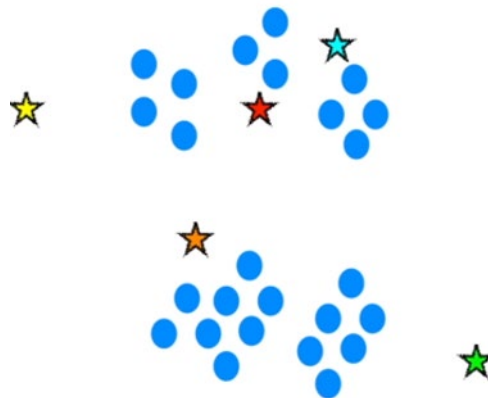
Distribution-based clustering

## K - Means algoritme voor het clusteren van documenten

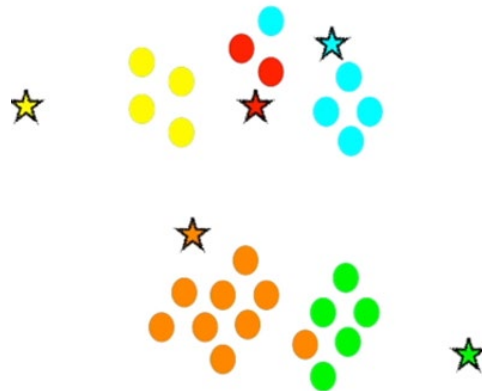
Gegeven een bepaalde set van documenten (artikels, reviews, tweets, boeken, webpagina's, ...) , groepeer (cluster) deze op basis van gelijkheid. Na de clustering ontdek je misschien dat er interessante thema's zijn binnen de clusters: zelfde auteur, zelfde uitgiftejaar, vergelijkbaar onderwerp of...

1. Dataset: al de documenten die je gebruikt om de clustering uit te voeren
2. Features: er zijn meerdere manieren om "tekst" om te zetten in numerieke waarden. Cfr. "term frequency representation". Elk document wordt op deze manier een N - tuple van cijfers. Dit punt in de N - dimensionale ruimte wordt uitgezet in een N - dimensionaal assenstelsel.
3. Het K - means algoritme deelt onze data in K groepen in waarbij K op voorhand bepaald wordt door de gebruiker.

**STAP 1:** We beginnen met het initialiseren van een set van middens (K means = K middens van de clusters die we willen vinden). Het aantal clusters wordt vastgelegd door de gebruiker (zeg 5). De initiële middens kunnen random zijn maar er zijn technieken om deze initiële waarden op een zodanige manier te bepalen dat het algoritme sneller tot convergeren overgaat (hier gaan we niet verder op in).



**STAP 2:** Nadat de initiële middens gekozen zijn, zal elk punt (dus elk document) toegewezen worden aan één van deze initiële middens. Op deze manier ontstaan de eerste clusters.



**STAP 3:** Eens deze clusters gemaakt zijn, zoeken we nieuwe middens (die onze initiële middens vervangen) door het midden van elke cluster te zoeken / bepalen.

We herhalen stap 2 en 3 totdat de nieuwe middens niet meer wijzigen. Dit noemen we convergentie. En op deze manier hebben we de  $K$  ( $=5$ ) clusters waarin we de documenten wilden verdelen.



## 6.4 Recommendations

Dit soort van ML probleem houdt zich bezig met “aanbevelingen” doen, de klant nieuwe noden laten ontdekken. Op basis van eerder gedrag van **jezelf** (opzoeken op internet, luisteren naar bepaalde liedjes, aankopen in een grootwarenhuis, ...) en/of **vergelijkbare gebruikers** worden items waar de gebruiker mogelijk nog in geïnteresseerd is, voorspeld zelfs nog voor hij/zij het zelf al beseft. Gepersonaliseerde “aanbevelingen” winnen aan belang. Ze hebben een (positieve) invloed op o.a. koopgedrag. Het kunnen voorspellen van gebruikerstevredenheid is hier nauw mee verbonden. Dit is een erg grote business geworden, zeker in de online wereld.

Iets anders waar Recommendations toe bijdraagt is “**trouw zijn aan**” een website: hoe langer je een bezoeker kan houden bij jouw site, hoe meer geneigd hij/zij is om te kopen, wat het uiteindelijke doel is van een webwinkel. Dus zoveel te persoonlijker je een website kan maken voor de klant, zoveel te meer geneigd de klant is om veel tijd te spenderen op deze site en dus meer te kopen.



### Voorbeeld:

- Bezoek eender welke “commerciële” website en je krijgt reclame over producten die je misschien wel leuk vindt en eventueel zal aankopen. Je zoekt online informatie op voor een nieuwe GSM → je krijgt reclame over telefoonhoesjes...
- Wanneer je Netflix of Spotify of... gebruikt, krijg je aanbevelingen voor programma's, films, liedjes die je misschien leuk vindt gebaseerd op hetgeen je reeds bekeken, geluisterd hebt.
- Online winkels hebben standaard zeer uitgebreide catalogi: Netflix (zeer uitgebreid aanbod aan films), Spotify (zeer ruim aanbod aan liedjes), Amazon ... maar gebruikers beperken zich vaak tot het bekijken van enkel de startpagina of een paar andere pagina's die maar een paar muisklikken verwijderd zijn. Gebruikers hebben dus sturing nodig in het vinden van wat ze zoeken of zelfs in wat ze interessant vinden. Soms kunnen aanbevelingen gebruikers leiden tot iets waarvan ze eigenlijk nog niet wisten dat ze het nodig hadden of leuk vinden.
- Vroeger waren webpagina's statisch (vergelijkbaar met een fysieke winkel). Reclame die verscheen, het uitzicht van een website, ... was gemaakt voor het “doorsnee” publiek; dus niet gepersonaliseerd. Dit is geëvolueerd naar pagina's waar (te pas maar soms ook te onpas) gepersonaliseerde boodschappen verschijnen.

Hoe gaat dit **praktisch** in zijn werk?

**Hoe kunnen we aanbevelingen doen op basis van browser geschiedenis?**

Gebruik het aantal keer dat een product bekeken is als een score voor dat product.

Bereken de score van alle producten op basis van de views door alle gebruikers.

Selecteer hieruit de top 10 producten voor een specifieke user.

**Wanneer je een product koopt, krijg je vaak meldingen in de aard van: andere bezoekers kochten ook dit... Hoe worden deze meldingen geselecteerd?**

We berekenen voor elk product een score. Hieruit zoeken we naar producten met een hoge score onder de gebruikers die reeds het product waar we ons op focussen, gekocht hebben. We zoeken dus naar een deelverzameling van producten met een hoge rating onder users die het specifieke product al gekocht hebben. Selecteer hieruit de top 10 producten

**Hoe vinden we de top 10 films die interessant zijn voor een gebruiker?**

We beschikken over een dataset met movie ratings en verder ook over het “movie profiel” van onze gebruiker.

We zoeken nu naar vergelijkbare users met onze gebruiker: users die dezelfde films wel / niet appreciëren.

Op basis van deze “neighbor” users wordt de top 10 meest interessante films voor onze gebruiker voorgesteld.

Het **gemeenschappelijk probleem** in voorgaande voorbeelden is dat we een **score** moeten berekenen voor alle producten gebaseerd op historische scores, gegevens, muisklikken, ... van alle gebruikers.

Dit wil zeggen dat de rating van een product voorspeld moet worden voor een bepaalde gebruiker zelfs als die gebruiker nog geen indicatie gegeven heeft dat hij/zij geïnteresseerd is in dit product.

Hiervoor gebruikt men “Collaborative Filtering”: deze algoritmes voorspellen scores van producten die een gebruiker nog niet gekocht of gezien heeft op basis van gedrag uit het verleden van andere vergelijkbare gebruikers.

**Definitie Collaborative Filtering:** een techniek, waarbij men gebruik maakt van historische gegevens, verzameld van andere gelijke gebruikers, met als doel het doen van aanbevelingen voor een nieuwe klant op basis van voorkeuren van andere klanten met een vergelijkbaar klantprofiel.

De **basisaannname van Collaborative Filtering** is zeer simpel: als je 2 gebruikers vindt met dezelfde mening over heel wat producten, dan zullen ze ook dezelfde mening hebben over andere producten.

Jupyter Notebook: `Machine_Learning_4_Recommandations_Movie.ipynb`

# 7 Bronnen

## PluralSight:

- Understanding Machine Learning (David Chapell)
- Understanding Machine Learning with Python (Jerry Kuratta)
- How to think about Machine Learning Algorithms (Swetha Kolalapudi)
- Play by Play: Machine Learning Exposed (Katharine Beaumont & James Weaver)
- Building Machine Learning models in Python with scikit-learn (Janani Ravi)
- Preparing data for Machine Learning (Janani Ravi)
- Reducing Dimensions in Data with scikit-learn (Janani Ravi)

## Websites:

- [http://www.cad.zju.edu.cn/home/zhx/csmath/lib/exe/fetch.php?media=2011:presentation\\_ml\\_by\\_ibrar.pdf](http://www.cad.zju.edu.cn/home/zhx/csmath/lib/exe/fetch.php?media=2011:presentation_ml_by_ibrar.pdf)
- <https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68>
- [https://www.python-course.eu/machine\\_learning.php](https://www.python-course.eu/machine_learning.php)
- <https://internetofthingsnederland.nl/wat-is-machine-learning/>
- <https://www.youtube.com/watch?v=GGWBMg0i9d4>
- <https://3bplus.nl/wat-is-machine-learning-een-introductie/>

**Seminarie:**

- Infofarm: An introduction to Machine Learning (Anaïs Ools, Glenn Cich en Lorenz Feyen)

Notebooks:     Machine\_Learning\_1\_Classification\_Iris  
                     Machine\_Learning\_3\_Regression\_Boston  
                     Machine\_Learning\_4\_Recommendations\_Movie