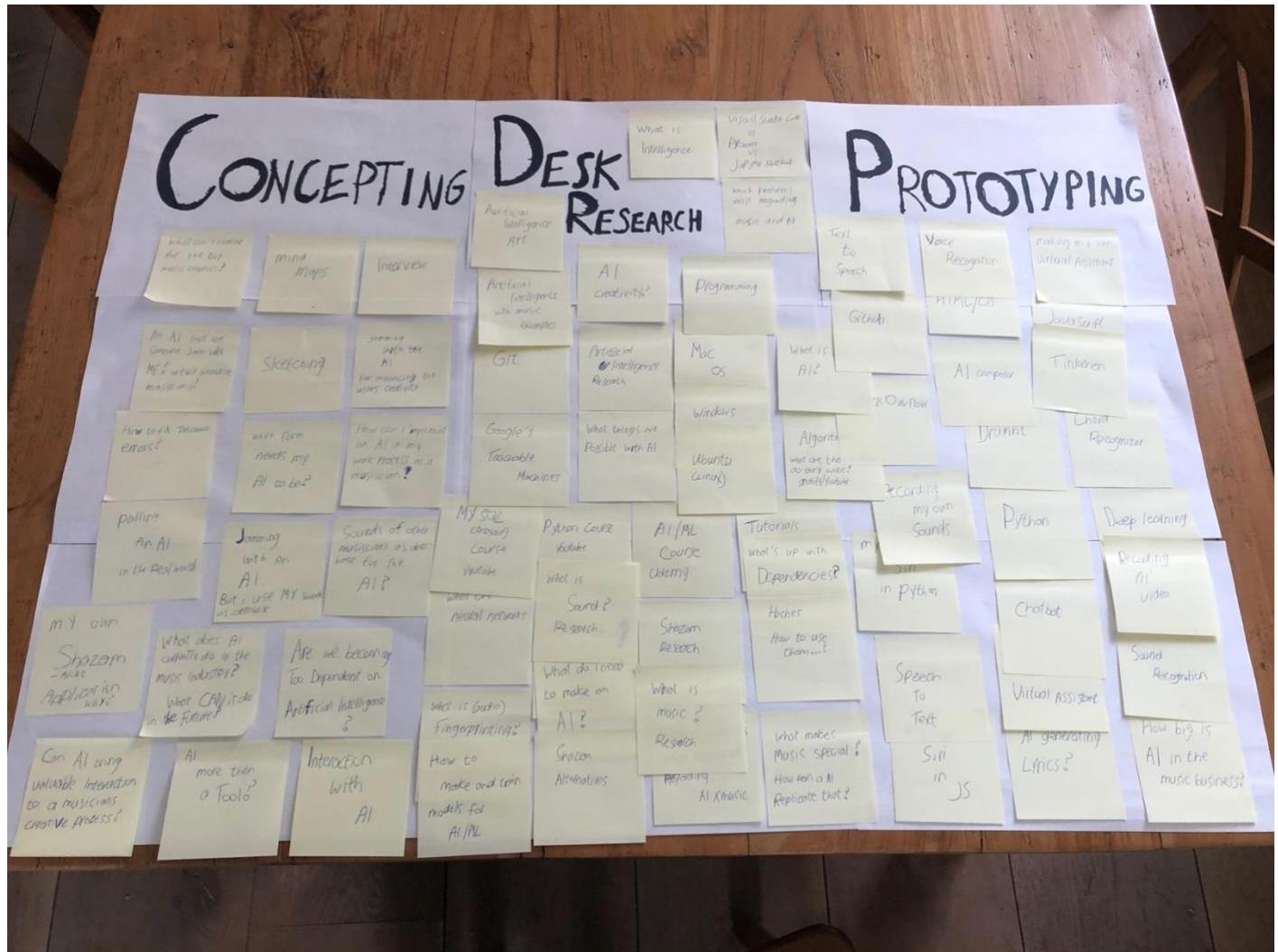


Procesboek



Max Walder
CMDJ3
2019-2020

Inhoudsopgave

Inleiding	3
Concepten & onderzoeksvragen	4
Deskresearch	7
Gebruikte Methoden	32
Prototypes	33
Reflectie	67
Bronnen	68

Inleiding

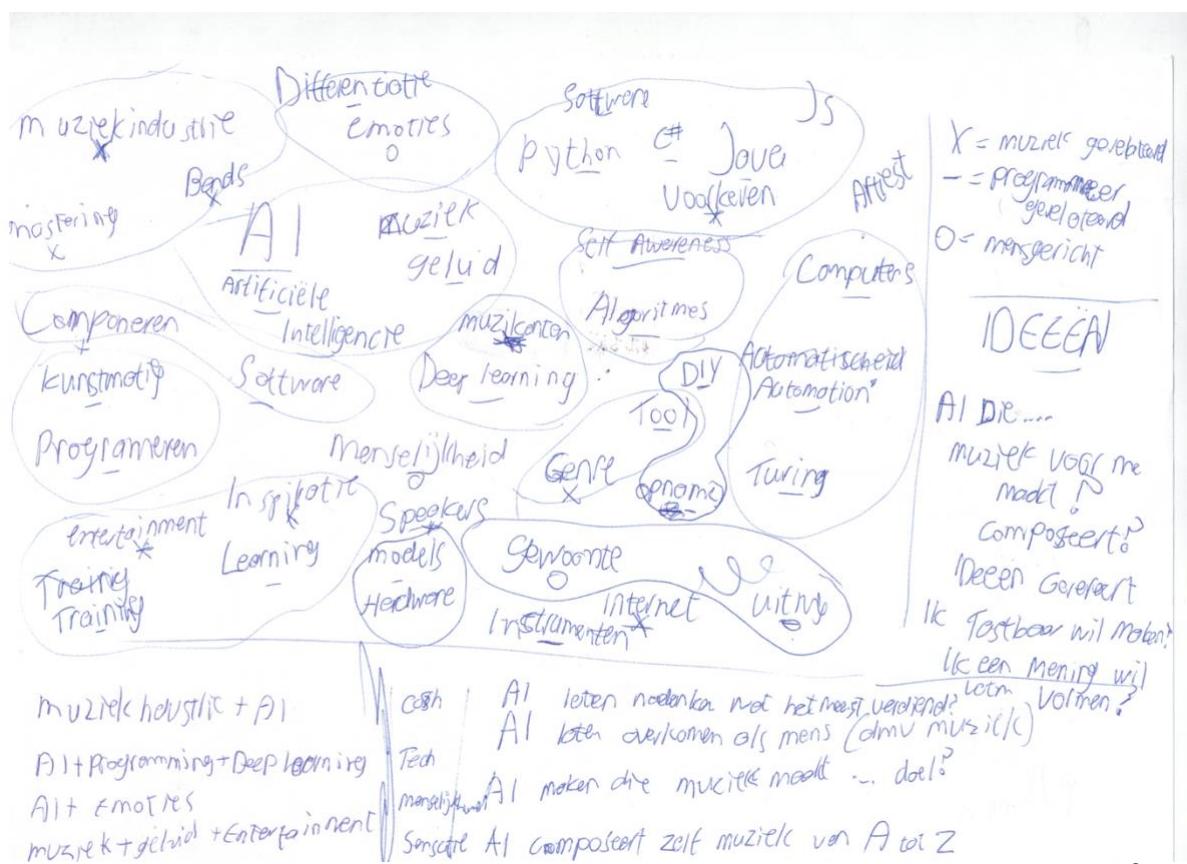
Dit document is mijn procesboek voor CMD jaar 3 semester 2, het lab. Ik had hier een hele lastige start, ik begon met mijn hoofdinteresses voor dit project op een rijtje te zetten. Ik hoopte door dit een beetje op een rijtje te zetten dat ik al wat inzichten zou krijgen in wat ik zou kunnen gaan doen voor dit labonderzoek.

Muziek: mijn hobby, brengt mensen samen, emotioneel, laat iemand ontsnappen aan de realiteit, leuk om naar te luisteren, zelfontplooiing, samenwerking.

Programmeren: iets dat ik leuk vind om te doen, dingen tot leven zien te komen vanuit het niets, een verhaal/item/iets laten zien aan de wereld via het internet (of ander medium), interactieve dingen bedenken/maken, kleine onderdeeltjes samen combineren totdat het één website is geworden, uitdaging.

Dit zijn twee dingen dat ik erg interessant vind. Voor mijn onderzoek ga ik iets van programmeren en muziek proberen te combineren om te kijken wat hier de nieuwe mogelijkheden mee kunnen zijn. Ik ben me erg gaan focussen op de muziekwereld, wat er qua technologie al is gemaakt, huidig op de markt is, en wat er in de toekomst zal gaan verschijnen.

Door deze twee dingen te combineren kwam ik al snel uit op de richting van muziek met Artificial Intelligence, maar dit is een heel erg breed concept. In dit document neem ik je mee in de stappen die ik heb gezet tijdens mijn afgelopen halfjaar.



Concepten & Onderzoeksvragen

Het begin was, net zoals bij ieder ander neem ik aan, erg zoeken naar wat ik nu precies wil gaan doen. Ik had een aantal concepten gemaakt waar ik misschien iets mee wilde gaan doen, ik heb hieronder de twee belangrijkste uitgeschreven.

Concept 1:

Ik ben zelf erg geïnteresseerd in muziek, ik zou daarom graag iets gaan onderzoeken waar dit in terugkomt. Ik heb zelf al wat onderzoek gedaan naar Spotify omdat ik het altijd heel erg knap vind dat Spotify bijvoorbeeld recommended lists kan maken om de gebruiker nieuwe muziek te laten ontdekken, of dat Spotify zelf bepaalde genres aan elk nummer geeft en hoe Spotify zelf playlists genereert. Het antwoord hierop is door een soort AI-robot dat werkt met algoritmes. In het geval van Spotify, een muzikant/label/band doet zijn of haar muziek uploaden naar Spotify en dan gaat Spotify hun magie uitoefenen op jouw files om je muziek bij de juiste andere muziek in te delen.

Dus in het kort wat er gebeurd is dat de AI van Spotify naar talloze nummers heeft geluisterd en hiervan geleerd heeft. Elk nieuw nummer kan hij een tag en een plaatsje geven aangezien hij hoort/ziet/weet waar de overeenkomsten met andere nummers zitten.

Mijn onderzoeksvraag is: Wat als we deze slimme AI-robot die werkt met algoritmes gebruiken om nieuwe muziek te maken. Deze AI heeft dus duizenden nummers in zich zitten en weet welke nummers de hits waren door de jaren heen. Deze robot weet welke muziek erg goed is ontvangen door bepaalde componenten die in dit nummer zitten. Is het mogelijk om een robot dit te laten analyseren om zelf een nummer te maken?

Als een robot een nummer gaat maken gebaseerd op duizenden nummer 1 hits komt er misschien iets erg leuks uit.

Maar waarom? Wat is hier het nut van? Stel het is mogelijk om een AI-Robot te ontwikkelen die nummer 1 hits kan maken, beïnvloedt dit de hele muzikale wereld. Een mens heeft goede en slechte dagen, een robot niet. Technisch gezien zou een robot hit achter hit kunnen maken, uiteindelijk heeft iedere muzikant zijn eigen hit-robot in de studio staan omdat deze betere muziek maakt en teksten schrijft als hem of haar.

En stel dit is mogelijk, wat gebeurd er dan met de overige creativiteit van de mens? Aangezien de robot al het denkwerk overneemt gaat een artiest/muzikant alleen nog maar optreden en de nummers inspelen die zijn gegenereerd door de AI.

Concept 2:

Wat gebeurt er als we robots een instrument geven. Is het mogelijk om robots zulke goede fijne bewegingen in te programmeren dat het mogelijk is om een robot/mechanische arm gitaar te laten spelen? Na wat kort onderzoek kwam ik erachter dat het mogelijk is om dit te

doen, maar dit is allemaal op basis van een stukje code, niet AI gegenereerd dat de robot dingen gaat spelen die hij zelf wil.

Maar waarom? Net zoals bij mijn vorige idee, ik vind het een interessant idee aangezien dit de muziekwereld compleet kan veranderen. Als mensen nu naar dj-muziek luisteren word er soms al gerefereerd naar ‘digitale muziek’, maar misschien is het wel mogelijk dat er over 10 jaar meerdere robot bands zijn.

Van deze twee hoofdideeën vond ik concept 1 het meest interessant in het begin van het lab. Ik was dus erg geïnteresseerd in AI en muziek. De focus bij dat concept lag nog heel erg op een AI-muziek te laten maken.

Uiteindelijk eindigde ik op:

How can the influence of AI improve the workflow of (DIY) music creators?

Vooral in het begin van dit lab ben ik heel erg veel deskresearch gaan uitoefenen op dit vlak, ik kwam er toen al snel achter dat ik vond dat dit concept erg gelimiteerd was, ik zou dan alleen onderzoek gaan doen naar wat voor soort applicaties, programma's er bestaan die wat doen met AI die zelf muziek maken. Dit vond ik te saai en weinig frictie hebben.

Ik keek nu bijvoorbeeld erg naar de rol van AI plugins in audiosoftware zoals Logic Pro X, Reaper en Audacity. Dit was natuurlijk nog allemaal deskresearch. Zo heb je bijvoorbeeld platformen zoals AIVA (software dat op basis van een paar parameters complete nummers genereert) en LANDR (software waar een band/muzikant zijn of haar track kan uploaden en hun AI doet de track masteren). Ik heb deze twee platformen ook zelf getest, hier kom ik later in detail nog op terug in dit document.

Na gesprekken met mijn begeleider ben ik er voor mezelf achter gekomen dat ik iets wil doen waar ik zelf als muzikant echt iets aan heb in het creatieproces, niet alleen als tool tijdens het opnemen bijvoorbeeld. Na deze gesprekken begon ik heel erg naar mezelf te kijken als muzikant, naar mezelf te kijken als iemand die graag muziek maakt als hobby.

Als ik naar mezelf kijk, hou ik heel erg van muziek maken, schrijven en opnemen. Maar echt het schrijfproces vind ik makkelijker om samen met andere te doen in plaats van alles in mijn eentje, zoals bijvoorbeeld in een band. Stel ik heb een supergaaf idee, maar het is midden in de nacht. In zo'n situatie kan ik niet ineens met andere gaan zitten om aan mijn idee te werken, maar een AI zou dat wel kunnen!

Als gevolg kwam ik uit op een nieuwe onderzoeksraag, namelijk:

How can AI improve the creative workflow of music creators?

Ik had zelf heel erg een klik met het idee om technologie in te gaan zetten op de creatieve aspecten van muziek maken. Hoe zou ik bijvoorbeeld een stukje software kunnen maken waar ik, of iemand anders iets aan heeft bij dit proces? Wat zijn de voordelen hiervan? De nadelen? Bestaan deze dingen al? Dit waren vragen die meteen bij mij naar boven kwamen.

Voor mij lag nu heel erg de focus op de interactie tussen mens en software. Als ik hier goed over nadenk, is creativiteit iets heel erg mens-eigen, dus hoe kan een stuk software dat hele mens-eigen proces beïnvloeden en hopelijk verbeteren?

Het leek mij heel erg interessant om uit te gaan zoeken wat een AI zou kunnen toevoegen aan het werkproces van een muzikant. Specifiek op de creatieve kant van zijn of haar proces.

Het eerste concrete idee dat ik had om uit te gaan werken is gaan jammen met een AI. Als ik muziek schrijf met een vriend bijvoorbeeld, kom ik met een stukje muziek, en daarna reageert de ander erop met feedback in de vorm van tips of een nieuw stukje muziek. Ik dacht, wat als ik hier de tweede persoon ga vervangen voor een Artificial Intelligence robot?

Ik had mijn focus geplaatst op het idee om een prototype te maken dat het mogelijk maakt om echt te kunnen jammen met een AI. Later in dit document kom ik hier ook nog uitgebreid op terug.

Als extra iteratie hierop kwam ik op het idee om niet een algemene AI te gebruiken hiervoor, maar bijvoorbeeld een AI met geluiden van mezelf, of een bepaalde muzikant. Hoe tof is het om bijvoorbeeld te kunnen gaan jammen met je favoriete muzikant? Dit idee wordt ook later nog toegelicht in dit document.

Op het einde van het lab had ik het probleem dat ik een beetje ben afgeweken van wat ik eigenlijk wilde gaan onderzoeken, alles ligt wel in lijn van wat ik wilde, maar ik heb hele andere soorten resultaten. Allemaal nuttige resultaten naar mijn mening, maar lastig om alles te plaatsen als een conclusie, als echt een rode draad in mijn onderzoek.

In de laatste weken van dit onderzoek kwam ik tot de conclusie dat ik vooral bezig ben geweest met het uitzoeken naar of ik, als muzikant waardevolle interacties kan krijgen met een AI in mijn creatieve proces.

Vandaar dat mijn uiteindelijke onderzoeksraag is geworden:

Can AI bring valuable interactions to a musicians creative process?

Deskresearch

Toen ik mijn eerste concept gevonden had, was ik meteen begonnen met allerlei soorten onderzoek, voornamelijk deskresearch. Aangezien het mijn eerste idee was om überhaupt iets te gaan doen met AI, koos ik ervoor om eerst hier goed onderzoek naar te gaan doen.

Artificial Intelligence Tutorial | AI Tutorial for Beginners | Artificial Intelligence | Simplilearn
<https://www.youtube.com/watch?v=FWOZmmIUqHg>

De Youtube link hierboven refereert naar een filmpje over AI dat erg leerzaam was. Veel van de komende informatie komt uit dit filmpje. Om het aan te vullen en completer te maken heb ik ook extra informatie gehaald uit andere bronnen.

Types of Artificial Intelligence:

Reactive Machines: Dit zijn eigenlijk de alledaagse machines die functioneren op het stukje code waarmee ze geprogrammeerd zijn. Bijvoorbeeld, jij komt thuis en de sensoren in je huis zien dat er een persoon is en schakelen het licht aan in je huis.

Limited Memory: Dit is bijvoorbeeld een zelfrijdende auto van Tesla, ze gebruiken huidige data en data van het verleden om beslissingen te maken. Bijvoorbeeld, de AI in je huis weet dat jij altijd rond 17.30 thuiskomt, dus zet om deze tijd het licht aan (een Smart System)

Theory of mind: Dit zijn robots die kunnen communiceren net alsof het een mens is, deze robots begrijpen bijvoorbeeld emoties van mensen. De wetenschap is nu bij deze stap, ze zijn hier vol mee bezig.

Self Awareness: Dit is echt de holy grail van AI-machines. Deze robots denken volledig voor zichzelf, zijn superslim en zijn ook bewust van wat ze doen. Dit is voor nu echt een toekomstbeeld van AI. Bijvoorbeeld, de robot zet het licht aan omdat hij het donker vindt en graag wilt zien wat hij (of zij?) doet.



Types of Artificial Intelligence



Achieving Artificial Intelligence:

Machine Learning: Deze machines leren van de data waar ze aan blootgesteld worden. Ze maken van deze data algoritmes die patronen ontdekken en gebruiken dit weer voor een bepaald doeleinde.

Deep Learning: Dit is het wat bekendstaat voor ‘neurale netwerken’. Het doel is om een machine het menselijk brein te leren nabootsen. Deep learning stelt computers dus in staat om nieuwe dingen te leren van grote hoeveelheden data, waarbij het niet uitmaakt of die data bestaat uit getallen, tekst, geluid of beeld.

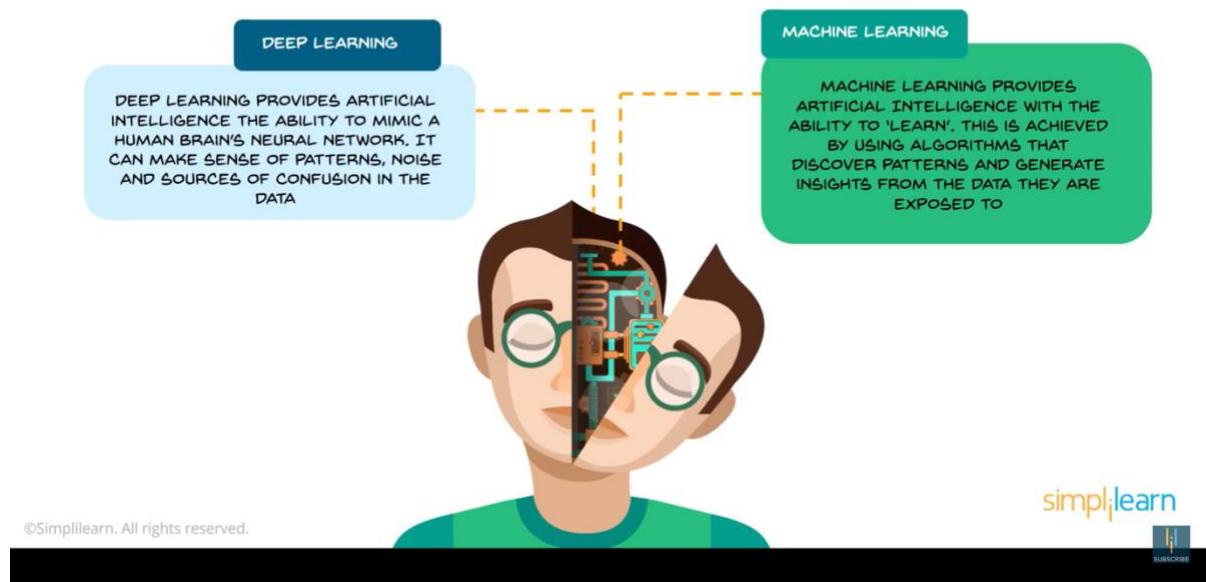
Maar hoe werkt dit? Er zijn 3 hoofdlagen in een neurale netwerk?

1: een input layer. Hier krijgt de AI-machine bijvoorbeeld allemaal verschillende foto's te zien.

2. verborgen layers: In de verborgen layers gaat de machine proberen verbanden te leggen en dingen te herkennen. Hoe meer verborgen layers je hebt, hoe accurater de resultaten zijn van de deeplearning.

3. output layer: Hier krijg je te zien wat iets is. De machine zegt bijvoorbeeld, ja deze foto is een portret of nee dit is geen portret. Het ligt dus erg aan de context van wat de machine aan het leren is. Wat precies wordt weergegeven in de output layer.

Achieving Artificial Intelligence



De meest voorkomende taal om een AI in te maken is in Python. Het kan uiteraard in allerlei andere talen.

In deze tutorial geven ze ook een relatief kort voorbeeld van een self learning AI, aangezien ik zelf nog geen ervaring heb met Python is dit erg lastig om dit uit te leggen.

Om een eigen AI te maken wordt aangeraden om de taal Python te gebruiken, in deze tutorial vond ik wat handige basisinformatie over de taal:

<https://www.youtube.com/watch?v=Y8Tko2YC5hA>

Ik ging vervolgens voor mezelf wat meer onderzoek doen naar de eigenschappen binnen het thema Artificial Intelligence;

Algoritmes:

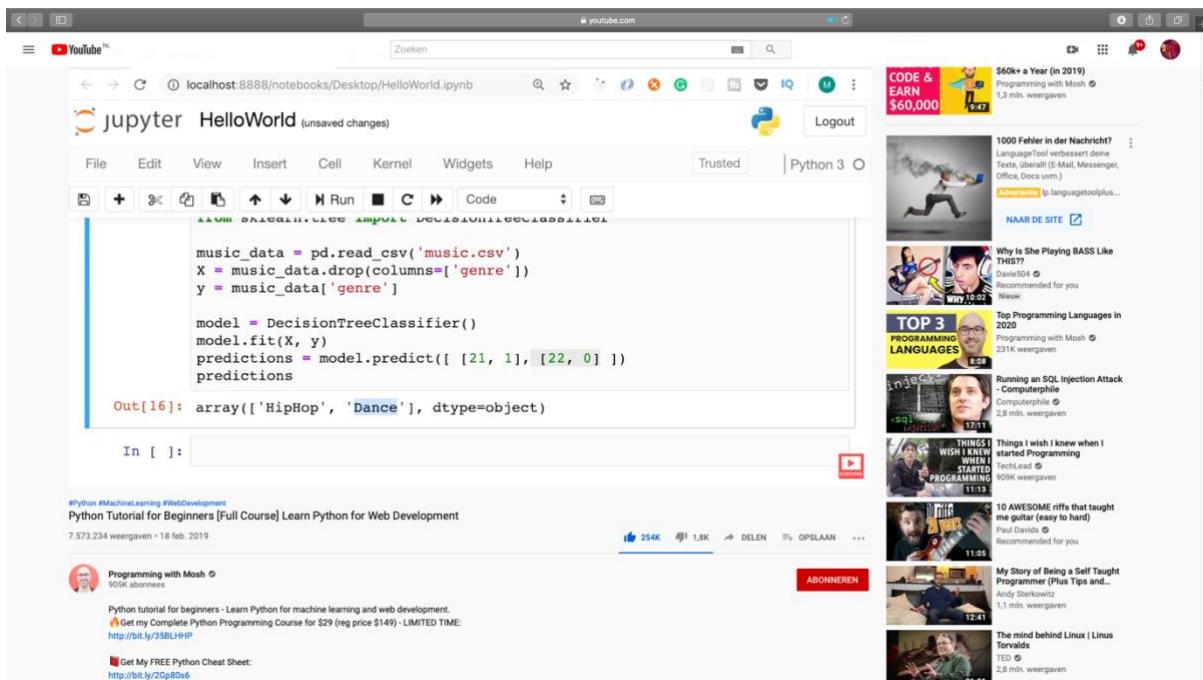
Algoritmes spelen een steeds grotere rol in bijvoorbeeld ons mediagebruik. Zo bepaald Facebook met algoritmes wat je op je tijdlijn ziet en kan YouTube jouw nieuwe filmpjes aanbevelen. Algoritmes bieden veel gemak, maar kunnen er bijvoorbeeld ook voor zorgen dat je in een filter bubbel terechtkomt.

Python Tutorial for beginners:

Python Tutorial for Beginners [Full Course] Learn Python for Web Development
https://www.youtube.com/watch?v=_uQrJ0TkZlc&t=125s

Dit is een volledige cursus van 6 uur op YouTube. Deze video is van het YouTube kanaal ‘Programming with Mosh’. Ik volg deze man zelf al langer voor HTML,CSS,JS en Angular tutorials. Hij is ook heel erg bekend van zijn volledige cursussen op Udemy. Dit is dus een 6-urige cursus helemaal gratis!

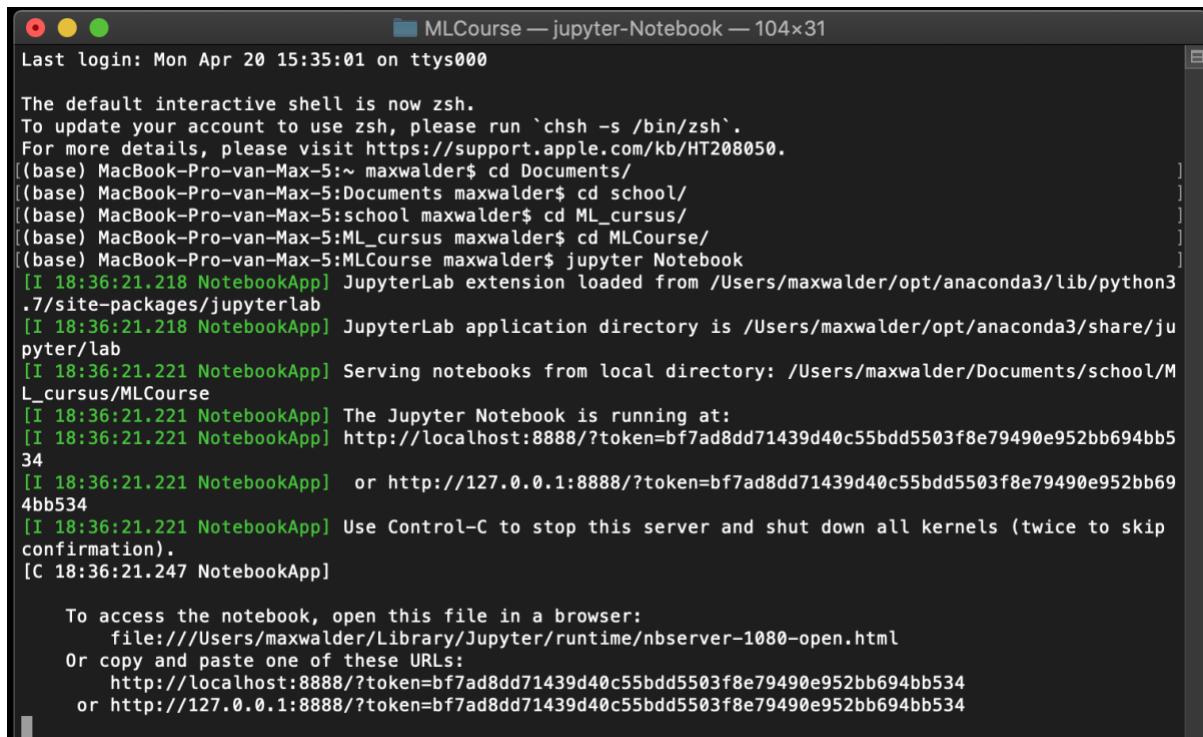
Zelf heb ik heel de 6 uur afgekeken en al veel van de basics over Python geleerd.



Voordat ik aan deze tutorial begon had ik nog geen ervaring met Python, nu heb ik veel geleerd over de beste editors, hoe ik deze kan instellen voor Python, de syntax, en de belangrijkste preciesies van werken met Python

De meest gebruikte editors zijn PyCharm en Jupyter Notebook. Aangezien je een eigen environment nodig hebt om python in te laden (aangezien het soms best zwaar is wat je wilt laten zien), kan Python niet altijd even makkelijk in andere browsers zoals Visual Studio. PyCharm is een editor echt specifiek ontwikkeld voor Python en is verder redelijk vergelijkbaar met bijvoorbeeld Visual Studio.

Waar in de tutorial vooral mee gewerkt werd was Jupyter Notebook. Als je deze applicatie hebt geïnstalleerd dan start je het op door de volgende code in te voeren in de command prompt/terminal: cd (naar de juiste map, scheelt wat tijd), Jupyter notebook.



```
MLCourse — jupyter-Notebook — 104x31
Last login: Mon Apr 20 15:35:01 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
[(base) MacBook-Pro-van-Max-5:~ maxwalder$ cd Documents/
[(base) MacBook-Pro-van-Max-5:Documents maxwalder$ cd school/
[(base) MacBook-Pro-van-Max-5:school maxwalder$ cd ML_cursus/
[(base) MacBook-Pro-van-Max-5:ML_cursus maxwalder$ cd MLCourse/
[(base) MacBook-Pro-van-Max-5:MLCourse maxwalder$ jupyter Notebook
[I 18:36:21.218 NotebookApp] JupyterLab extension loaded from /Users/maxwalder/opt/anaconda3/lib/python3
.7/site-packages/jupyterlab
[I 18:36:21.218 NotebookApp] JupyterLab application directory is /Users/maxwalder/opt/anaconda3/share/ju
pyter/lab
[I 18:36:21.221 NotebookApp] Serving notebooks from local directory: /Users/maxwalder/Documents/school/M
L_cursus/MLCourse
[I 18:36:21.221 NotebookApp] The Jupyter Notebook is running at:
[I 18:36:21.221 NotebookApp] http://localhost:8888/?token=bf7ad8dd71439d40c55bdd5503f8e79490e952bb694bb5
34
[I 18:36:21.221 NotebookApp] or http://127.0.0.1:8888/?token=bf7ad8dd71439d40c55bdd5503f8e79490e952bb694bb5
4bb534
[I 18:36:21.221 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip
confirmation).
[C 18:36:21.247 NotebookApp]

To access the notebook, open this file in a browser:
  file:///Users/maxwalder/Library/Jupyter/runtime/nbserver-1080-open.html
Or copy and paste one of these URLs:
  http://localhost:8888/?token=bf7ad8dd71439d40c55bdd5503f8e79490e952bb694bb534
  or http://127.0.0.1:8888/?token=bf7ad8dd71439d40c55bdd5503f8e79490e952bb694bb534
```

Als deze command is ingevoerd dan wordt er automatisch een localhost gestart en word Jupyter Notebook geopend in je browser, via hier kan je code schrijven en word het meteen in de browser uitgevoerd.

The screenshot shows a Jupyter Notebook interface running on localhost:8888/tree. At the top, there are tabs for 'Machine Learning in Python (Data Science and Deep L...', 'LANDR', 'python logo - Google Zoeken', and 'Home Page - Select or create a notebook'. Below the tabs, the word 'jupyter' is displayed. A navigation bar with 'Files', 'Running', and 'Clusters' buttons is visible. A message 'Select items to perform actions on them.' is shown above a file list. The file list includes various Jupyter notebooks and other files like 'emails' and 'ml-100k'. Columns for 'Name', 'Last Modified', and 'File size' are present. Buttons for 'Upload', 'New', and 'Logout' are at the top right.

Dit is wat er automatisch wordt geopend als je Jupyter opstart in de juiste map. Je ziet alle files die zich in de map bevinden, via hier kan je dubbelklikken op de file om het meteen te openen. Het werkt heel erg snel en gemakkelijk al zeg ik zelf.

The screenshot shows a Jupyter Notebook session titled 'HelloWorld - Jupyter Notebook'. The browser tab is 'localhost:8888/notebooks/Documents/school/Al_Mosh/HelloWorld.ipynb#'. The notebook interface has a toolbar with File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Kernel buttons. Below the toolbar, there are buttons for Run, Cell, Kernel, and Help. The main area shows code cells and their outputs:

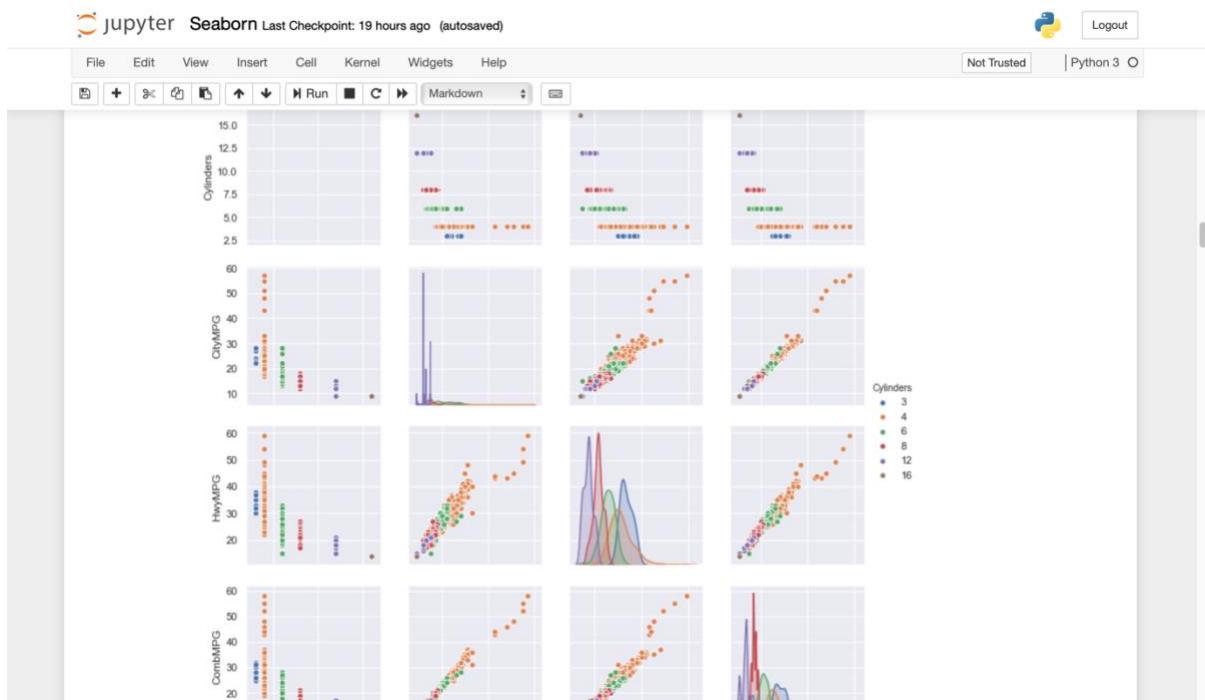
```

In [1]: import pandas as pd
In [4]: import pandas as pd
df = pd.read_csv('vgsales.csv')
df.shape
Out[4]: (16598, 11)

In [3]: df.describe()
Out[3]:
      Rank      Year   NA_Sales   EU_Sales   JP_Sales  Other_Sales  Global_Sales
count  16598.000000  16327.000000  16598.000000  16598.000000  16598.000000  16598.000000
mean   8300.605254  2006.406443  0.284667  0.146652  0.077782  0.048063  0.537441
std    4791.853933  5.828981  0.816683  0.505351  0.309291  0.188588  1.555028
min    1.000000  1980.000000  0.000000  0.000000  0.000000  0.000000  0.010000
25%   4151.250000  2003.000000  0.000000  0.000000  0.000000  0.000000  0.060000
50%   8300.500000  2007.000000  0.080000  0.020000  0.000000  0.010000  0.170000
75%  12449.750000  2010.000000  0.240000  0.110000  0.040000  0.040000  0.470000
max   16600.000000  2020.000000  41.490000  29.020000  10.220000  10.570000  82.740000

```

Zoals je ziet krijg je heel gemakkelijk data inzichtelijk te zien in de browser, dit werkt ook heel erg goed voor bijvoorbeeld allerlei soorten grafieken.



Doormiddel van SeaBorn kan je bijvoorbeeld al heel makkelijk al deze soorten grafieken laten weergeven op basis van je data.

Je kan in de beschrijving ook een cheatsheet downloaden, de cheatsheet is te bekijken op: <https://programmingwithmosh.com/wp-content/uploads/2019/02/Python-Cheat-Sheet.pdf>

In deze cheatsheet kan je heel veel informatie heel snel vinden over de taal Python.

Handige links:

PsyCharm: gratis te verkrijgen via (<https://www.jetbrains.com/pycharm/>) dit is een tekst editor speciaal voor Python. Dit programma is specifiek ontwikkeld voor het programmeren in Python.

<https://www.anaconda.com/distribution/> : Dit is een soort tekst editor speciaal voor machine learning en AI. Via Anaconda open je een soort tekst editor in een browser via een localhost en vanuit daar kan je je code typen en krijg je meteen daaronder te zien wat het doet. Aangezien je soms met hele grote bestanden werkt, is het niet handig werken vanuit een standaard tekst editor. Heel handig maar echt specifiek bedoeld voor het programmeren van machine learning en AI.

Pypi.org: community based platform waar programmeurs hun eigen gemaakte Python packages kunnen uploaden om te delen met andere Python programmeurs. Als je bijvoorbeeld een bepaalde functie wilt maken in Pyhton is het niet raar dat andere programmeurs al precies hetzelfde hebben gemaakt, grote kans dat je veel stukken code van deze website kan downloaden in de vorm van een package. Het fijne hieraan is dat ook bij bijna alles comments staan met de uitleg van de code.

Kaggle.com: Via deze website kan je bijna 300.000 datasets en (data)notebooks vinden voor een open source doeleinde. Een handig voorbeeld voor een deep learning systeem is dat er bijvoorbeeld een dataset is met 500 plaatjes met circa 1100 gemarkeerde gezichten in deze foto's. Als je een systeem wilt laten leren kan dat erg handig met deze dataset.

14 uur Machine Learning, Data Science and deep learning in Python Udemy cursus

Om meer te leren over het maken van een AI heb ik een cursus aangeschaft, namelijk:

Machine Learning, Data Science and Deep Learning With Python

Description: Complete hands-on machine learning tutorial with data science, Tensorflow, artificial intelligence and neural networks.

<https://www.udemy.com/course/data-science-and-machine-learning-with-python-hands-on/lecture/15090172?start=45#overview>

Dit is een Udemy cursus bestaand uit 14uur beeldmateriaal over dit onderwerp. Ik ben nu nog met de cursus bezig maar heb er al veel van geleerd.

Deze cursus focust zich heel erg op ML, Data Science, Deep Learning, TensorFlow, AI en Neural Networks. Hij geeft zijn lessen ook in Jupyter Notebook.

The screenshot shows the Udemy course page for 'Machine Learning, Data Science and Deep Learning with Python'. The course title is prominently displayed at the top. Below it, a brief description reads: 'Complete hands-on machine learning tutorial with data science, Tensorflow, artificial intelligence, and neural networks'. A thumbnail image features a man in a cap and a background of glowing neurons. A progress bar indicates completion: 'Preview van deze cursus bekijken' (Preview of this course). A note says: 'Je hebt deze cursus aangeschaft op 25 maart 2020' (You bought this course on March 25, 2020). At the bottom, there are two large buttons: 'Ga naar cursus' (Go to course) and 'Deze cursus delen' (Share this course).

Wat je leert

- ✓ Build artificial neural networks with Tensorflow and Keras
- ✓ Classify images, data, and sentiments using deep learning
- ✓ Make predictions using linear regression,
- ✓ Data Visualization with Matplotlib and

Ga naar cursus

Deze cursus delen

30-dagen-geld-terug-garantie

Ik ben dus veel bezig met het leren van nieuwe talen en preciepes die nodig zijn om deze talen te begrijpen. Voor Machine Learning heb je gewoon veel kennis nodig buiten het programmeren, zoals wiskunde bijvoorbeeld. Zelf kom ik van de VMBO en MBO af, dus ik miste hier al wat basiskennis van.

Artificial Intelligence in de muziekindustrie

Zoals ik heb verteld in de Concepten & Onderzoeken kopje, ik heb een paar platformen getest waar gebruik wordt gemaakt van Artificial Intelligence in de muziekindustrie.

Dit zijn voornamelijk AIVA en LANDR geweest, ik heb deze platformen getest en dit vervolgens ook op SoundCloud geplaatst.

AIVA (Artificial Intelligence Visual Artist)

Via AIVA kan iemand nummers laten genereren op basis van een aantal parameters en genres. Ik stond er echt versteld van hoe goed sommige ‘nummers’ klonken.

Created in February 2016, AIVA specializes in [Classical](#) and [Symphonic music](#) composition.^{[1][2]} It became the world’s first virtual composer to be recognized by a music society ([SACEM](#)).^{[3][4]} By reading a large collection of existing works of classical music (written by human composers such as [Bach](#), [Beethoven](#), [Mozart](#)) AIVA is capable of detecting regularities in music and on this base composing on its own.^{[5][6]} The algorithm AIVA is based on [deep learning](#) and [reinforcement learning](#) architectures.^[7] Since January 2019, the company offers a commercial product, Music Engine, capable of generating short (up to 3 minutes) compositions in various styles (rock, pop, jazz, fantasy, shanty, tango, 20th century cinematic, modern cinematic, and Chinese).

AIVA was presented at [TED](#)^[8] by Pierre Barreau.^[9]

Maar hoe werkt AIVA? Als je naar de website van AIVA gaat en een gratis account aanmaakt heb je de opties voor de beschikbare pakketten die te krijgen zijn voor AIVA.

The screenshot shows the AIVA website's pricing section. It displays the current plan as 'Free' and offers to 'Subscribe to any plan to access more features & benefits.' Below this is a detailed comparison table for different subscription plans:

	Free, Forever €0 / month	Standard Monthly €19 / month	Pro Monthly €59 / month	Standard Annually €14 / month *	Pro Annually €39 / month *
Billing cycle	-	Billed Monthly	Billed Monthly	* Billed Yearly (Save €60) <small>1</small>	* Billed Yearly (Save €240) <small>1</small>
Copyright ownership	Owned by AIVA	Owned by AIVA	Owned by YOU	Owned by AIVA	Owned by YOU
Usage type	Non Commercial	Non Commercial	Commercial	Non Commercial	Commercial
Composition Creation	∞	∞	∞	∞	∞
High quality virtual instruments	All	All	All	All	All
Downloads per month	3	15	∞	∞	∞
Downloads for Pop, Rock, Jazz	✗	✓	✓	✓	✓
Downloads for other presets	✓	✓	✓	✓	✓
Create folders	✗	✓	✓	✓	✓
Download folders	✗	✗	✗	✓	✓
Download chords	✗	✗	✓	✗	✓
Lossless 16 bit WAV file	✗	✗	✓	✗	✓
Stored Influences	10	250	250	250	250

At the bottom, there are buttons for 'Current Plan', 'UPGRADE', and five additional 'UPGRADE' buttons for each column. Below the table is a 'Frequently Asked Questions' section.

Als je dit gedaan hebt is het allemaal eigenlijk heel erg simpel om zelf een nummer te genereren met AIVA.

The screenshot shows the 'COMPOSITIONS' tab on the AIVA website. It lists ten generated tracks, each with a play icon, title, parameters, duration, creation date, and three-dot options menu:

TITLE	PARAMETERS	DURATION	CREATION DATE
New Composition #0	Rock, E Minor, Heavy Metal Ensemble, 140 BPM...	0:31	Apr 20, 2020
New Composition #1	Fantasy, D# Minor, Dragonborn Ensemble, 90 ...	1:52	Apr 20, 2020
New Composition #3	Jazz, G Major, Jazz Ensemble, 80 BPM, 4/4	2:31	Apr 20, 2020
New Composition #4	Fantasy, F Minor, Dragonborn Ensemble, 110 ...	1:43	Apr 20, 2020
New Composition #5	Fantasy, D# Minor, Small Tavern Band, 100 BP...	3:29	Apr 20, 2020
New Composition #6	Electronic, D Major, Dat Funk Ensemble, 130 B...	3:22	Apr 20, 2020
New Composition #7	Modern Cinematic, F Minor, Mallets & Orchest...	2:20	Apr 20, 2020
New Composition #8	Sea Shanties, E Minor, Symphonic Orchestra, ...	2:13	Apr 20, 2020
New Composition #9	Fantasy, D# Minor, Small Tavern Band, 80 BP...	3:20	Apr 20, 2020
New Composition #10	Modern Cinematic, G Minor, Epic Orchestra, 8...	3:16	Apr 20, 2020

At the bottom, there are buttons for 'New Folder' and 'Create Track'.

Op deze homepage, My Tracks zie je al je gemaakte tracks en kan je ook nieuwe genereren door op de groene knop te drukken.

The screenshot shows the 'COMPOSITIONS' tab on the AIVA website. It displays a grid of preset styles for generating new tracks:

	PRESET STYLE	UPLOAD AN INFLUENCE
	Modern Cinematic	
	20th Century Cinematic	
	Sea Shanty	
	Jazz	
	Fantasy	
	Pop	
	Rock	
	Chinese	
	Tango	
	Electronic	

Below the grid, there are 'Preview' buttons for each preset and a link to 'Not sure what to select? Read our suggestions'.

Vervolgens kom je op deze pagina uit waar je een genre kan kiezen voor je track.

The screenshot shows the AIVA web interface. At the top, there are tabs for 'COMPOSITIONS', 'INFLUENCES', and 'SHARED WITH ME'. Below this is a navigation bar with 'Create a track' and 'My Tracks' (which has a red notification dot). The main area is titled 'COMPOSITIONS' and displays a table with columns: TITLE, PARAMETERS, DURATION, and CREATION DATE. Two compositions are listed: 'New Composition #9' and 'New Composition #10'. Below the table is a 'New Folder' button and a 'Create Track' button. A modal window is open in the center, titled 'SELECTED PRESET: FANTASY'. It contains several dropdown menus and checkboxes. The 'KEY SIGNATURE' dropdown shows 'Auto'. The 'TIME SIGNATURE' dropdown shows 'Auto'. The 'PACING' dropdown shows 'Auto'. The 'INSTRUMENTATION' dropdown shows 'Auto' and lists 'Dragonborn Ensemble', 'Symphonic Orchestra', 'Solo Strings', and 'Strings & Woodwinds'. There are also checkboxes for 'Include Percussions (only for ensembles that...' and 'Include Tempo Variations'. At the bottom of the modal are 'Cancel' and 'Create' buttons. The bottom of the page shows a navigation menu with links like 'Piano Roll', 'Billing', 'Updates', 'Community', 'Tutorials', and 'FAQ', along with a search bar and a 'Switch to Piano Roll' button.

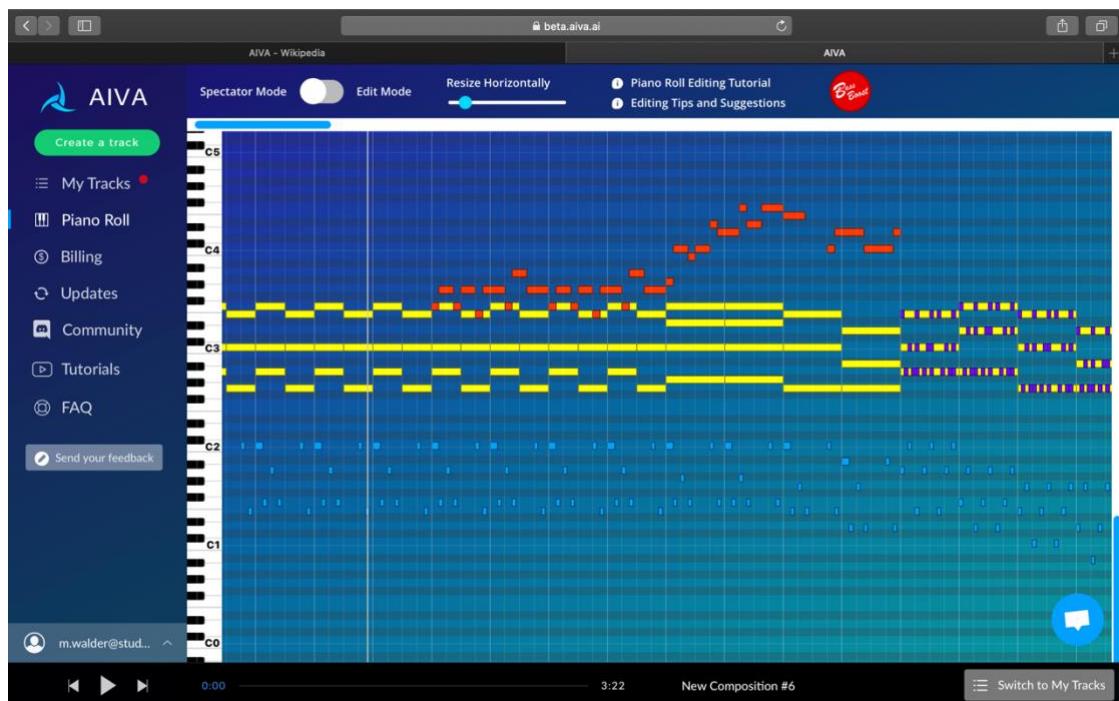
Bij deze volgende stap kan je de track een beetje personaliseren door een aantal parameters aan te geven. Standaard is het bijna allemaal auto, maar je hebt bij elke parameter veel keuzes, zelfs bij alle genres

This screenshot shows the same AIVA interface after the composition has been created. The 'COMPOSITIONS' table now includes the newly generated tracks. The table columns are: TITLE, PARAMETERS, DURATION, and CREATION DATE. The new entries are:

TITLE	PARAMETERS	DURATION	CREATION DATE
New Composition #1	Fantasy, D# Minor, Dragonborn Ensemble, 90 BPM, 4/4	1:52	Apr 20, 2020
New Composition #3	Jazz, G Major, Jazz Ensemble, 80 BPM, 4/4	2:31	Apr 20, 2020
New Composition #4	Fantasy, F Minor, Dragonborn Ensemble, 110 BPM, 4/4	1:43	Apr 20, 2020
New Composition #5	Fantasy, D# Minor, Small Tavern Band, 100 BPM, 4/4	3:29	Apr 20, 2020
New Composition #6	Electronic, D Major, Dat Funk Ensemble, 130 BPM, 4/4	3:22	Apr 20, 2020
New Composition #7	Modern Cinematic, F Minor, Mallets & Orchestra, 90 BPM, 4/4	2:20	Apr 20, 2020

The bottom of the page shows the same navigation menu and search bar as the previous screenshot.

Vervolgens kom je weer terug bij de homepage en zie je dat er een nieuwe track bij is gekomen en aan het genereren is. Het genereren van een track duurt meestal tussen een paar seconde en een minuut.



Hier ziet u de spectator mode van de ‘Piano Roll’ je ziet hier eigenlijk welke noten/akkoorden er zijn gebruikt bij het genereren van deze track.



Als je toch niet helemaal tevreden bent over je tracks heb je nog mogelijkheden om het aan te passen.

Ik heb zelf even met dit programma gespeeld en mijn beste nummers opgeslagen en op SoundCloud gezet.

U vindt ze hier:

<https://soundcloud.com/user-583891932/sets/aiva>

Door onderzoek te hebben gedaan naar AIVA heb ik nieuwe inzichten opgedaan over hoe bijvoorbeeld AI ingezet kan worden in het genereren van muziek.

Ik stond er echt versteld van hoe goed sommige tracks klonken die AIVA voor mij maakte. Dit proces opende echt mijn ogen op wat sommige mogelijkheden zijn en hoe groot deze markt eigenlijk is.

LANDR

Een ander voorbeeld is bijvoorbeeld het platform LANDR. Via deze service kan je doormiddel van hun AI-technologie jouw muziektrack laten masteren. Je hoeft dus als (DIY) muzikant je eigen opgenomen tracks niet zelf te masteren. Op basis van een aantal kenmerken geef je aan bij LANDR wat voor een soort nummer het is en wat het moet worden. Vervolgens doet LANDR de rest. LANDR biedt ook opties voor het distribueren van de muziek als het af is.

LANDR is an online, cloud-based, automated [mastering](#) service developed by [MixGenius](#) in Montreal, QC. It allows for the upload of audio tracks in various formats which are then instantly [digitally mastered](#) using [artificial intelligence](#) algorithms. Free and paid services are offered. In July 2017, LANDR added digital music distribution services, which allow users to push their work to popular streaming platforms like [Spotify](#), [Deezer](#), [Apple Music](#) and others.

Het werkt heel erg makkelijk om je track te uploaden, te masteren en vervolgens door te zetten naar de streamingdiensten zoals Spotify en Apple Music. Om snel een kleine impressie te laten zien heb ik een stukje gefilmd van hoe het eruit ziet.

U vindt de filmpje als u klikt op de volgende link: <https://youtu.be/C6EwysRVqUQ>

Dit is een nummer dat ik een tijd terug met mijn band heb gemaakt en nooit iets mee heb gedaan, er zit ook niet echt een master op, vandaar dat je best wat verschil hoort tussen de original en mastered geluid.

Ik kon moeilijk andere voorbeelden vinden waar duidelijk werd vergeleken tussen twee masters bijvoorbeeld, iets wat ik wel interessant vond is het volgende filmpje:

<https://www.youtube.com/watch?v=U7feYhEPukw>

Bij dit filmpje heeft een persoon een track gemaakt, deze vervolgens laten masteren door een professionele mastering engineer en door LANDR. Hierna geeft hij een vergelijking van beide masters en verteld hierover.



ARTISTS WORKING WITH AI

David Bowie – The Verbasizer

David Bowie, een muzikant die jarenlang aan de top stond gebruikte een vorm van AI in zijn creatieve schrijfproces. Een vriend van Bowie had voor hem een programma geschreven genaamd ‘The Verbasizer’. Bowie kan verschillende zinnen invoeren en drukt vervolgens op de ‘random’ button. Hierna is de output dezelfde zinnen in een totaal andere volgorde.

In een interview uit de docentaire ‘Inspirations’ van Michael Apted uit 1997 en een BBC documentaire ‘Cracked Actor’ praat de zanger over het programma. Zo zegt Bowie dat hij dit programma heel erg veel heeft gebruikt bij zijn album ‘Outside’ uit 1995 en op het vervolgalbum ‘Earthling’ 1997. De zanger zegt: ‘If you put three or four disassociated ideas together and create awkward relationships with them. The unconscious relationships that come from those pairings can be really startling sometimes’. Later in dit interview zegt hij hoe hij het programma gebruikt: ‘I take articles out of newspapers, poems that I’ve written, pieces out of other people’s books and put them all together in this warehouse, this container of information and hit the ‘random’ button’. Vervolgens krijgt hij dus hele aparte zinnen, structuren en soms woorden terug van het programma, hij zei in een ander interview over dit programma, ik hoef soms maar vier woorden te vinden die niks met elkaar te maken hebben maar mij toch een soort visie geven voor een thema, door dit apparaat kom ik zelf op ideeën voor nummers wat ik zelf nooit zomaar zou kunnen doen.

Hierop geeft hij het voorbeeld van het blad dat hij voor zich heeft: ‘The Top kills himself, that sounds like a boss, isn’t it? And suddenly I get like a vision about a boss in the 30’s

throwing himself out of the window, you know because of the great depression. That might be enough to set me off to write a song about that'.

Bowie heeft dus talloze teksten geschreven met behulp van zijn apparaat. Soms gebruikt hij volledige zinnen (of hij zet de structuur alleen goed) en soms gebruikt hij een paar woorden wat hij gebruikt als inspiratie. Het programma was voor hem echt een tool om betere teksten te laten schrijven. In zijn eigen woorden:

'It's almost like a technological dream in it's own way, it creates images from the dream state without having the boredom to go to sleep all night, or to get it out of your head. It will give me access to areas I wouldn't otherwise think about during the day.'

Uiteraard is dit niet echt iets wat wij nu als AI zien, maar het geeft wel duidelijk aan dat ook toen al, technologie werd gebruikt in het creëren van muziek door professionals.

https://www.youtube.com/watch?time_continue=16&v=x3IKLMgFaDA&feature=emb_title

<https://www.youtube.com/watch?v=6nlW4EbxD8>

'Hello World' - SKYGGEE

"Hello World" is the first multi-artist music album composed with Artificial Intelligence. Its goal is to show that AI can be used to create new, compelling music, and not mere lab demos. The album is intended to become a landmark album in the history of technology and music creation.

SKYGGEE is the architect of *Hello World*. Inspired by the tale of Hans Christian Andersen, the french composer Benoit Carré imagined this avatar as his own shadow revealed by his musical works with the AI tools of Flow-Machines.

Musician in residence in 2016, Benoit Carré began his exploration by composing a song in the style of The Beatles. This is how 'Daddy's Car' started buzzing on the web in September 2016.

'Ballad Of The Shadow' is the first song of SKYGGEE a.k.a. Benoit Carré & Flow-Machines. This is a song with a voice as flexible as a reflection in a mirror, it will take various forms through the album. SKYGGEE then opened the Flow-Machines studio to other artists. He wanted his experience to be collective and iconoclastic.

Stromae, Kiesza, Kyrie Kristmanson, C.Duncan, Camille Bertault, Mederic Collignon, NZCA Lines, ... had the desire and the curiosity to meet with Flow-Machines.

They composed, wrote, sang with SKYGG, always keeping their artistic freedom, with the challenge to create amazing songs. Ash Workman (Metronomy, Christine and The Queens) produced the album in collaboration with SKYGG and Michael Lovett.

Link naar de officiële videoclip van het nummer ‘Magic Man’:

https://www.youtube.com/watch?time_continue=23&v=MG7YvFqD4zc&feature=emb_title

spotify link van het hele album:

<https://open.spotify.com/album/0cGWC9bhEJA4l7jAaV7cqR?si=LpB-I07DQ7-qk8P-i0PSiw>

Shazam

Als hoofdidee wilde ik een AI maken die ook music recognition kan uitvoeren. Ik moest meteen denken aan Shazam en ging hier onderzoek naar doen.



Het idee waar ik dus naartoe aan het werken ben is een applicatie die mij een klein stukje muziek hoort spelen, dit verwerkt, en hierna een reactie teruggeeft wat een stukje muziek kan zijn dat gespeeld kan worden na het stukje dat ik heb gespeeld. Het doel is dus om tips te geven, mogelijkheden als het ware wat na het stukje komt dat de gebruiker net gespeeld heeft. Op deze manier kan je samen met je AI-muziek maken, samen jammen tot je een vet nummer gemaakt hebt.

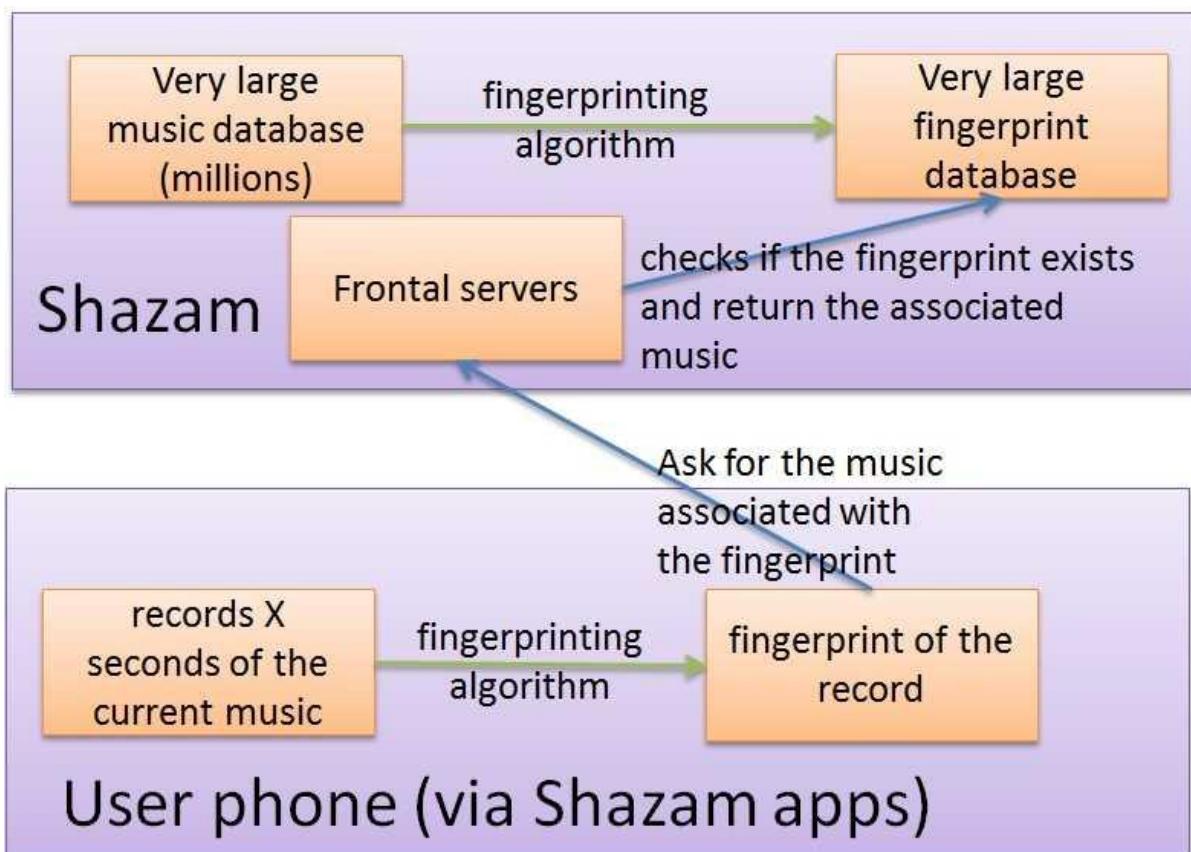
Maar waarom Shazam? Als ik mijn AI ga uittekenen, dan bestaat het uit twee delen.

1. Muziek opvangen, dit doorsturen naar een database
2. Een nieuw stukje muziek terugkrijgen (wat dus na het ingespeelde stuk kan komen), en dit teruggeven als een output.

Het programma Shazam, werkt praktisch hetzelfde als wat ik wil doen, in plaats van dat Shazam de naam, artiest en alle info teruggeeft, wil ik weten wat voor muziek ik kan spelen.

Voor deze reden ben ik onderzoek gaan doen naar Shazam en ging ik proberen om zelf een Shazam achtige programma te schrijven.

Maar hoe werkt Shazam nu eigenlijk precies? Ik vond onderstaand plaatje erg nuttig bij het begrijpen van hoe het werkt.



Shazam maakt dus een kleine fingerprint (ook wel audioprint genoemd) van gemiddeld 5 seconden, stuurt dit naar een server, door naar een database, en vervolgens geeft het programma als reactie terug wat de info is van het gescande nummer.

Maar in mijn geval, is het iets anders, stel ik krijg alles werkend hoe ik het wil, dan speel ik iets in van bijvoorbeeld ongeveer 5 seconde, het programma maakt hier dan een fingerprint/audioprint van, stuurt dit naar de server, door naar de database (precies hetzelfde als Shazam doet), hierna wordt dus gezocht naar een identiek stukje audio aan die 5 seconde uit de fingerprint/audioprint. Als dit gevonden is, ben ik benieuwd naar het riffje/akkoord/etc wat direct na het audioprint komt die ik heb doorgestuurd naar de database.

AUDIO Database

De database haalt
constant data op uit
de grote database

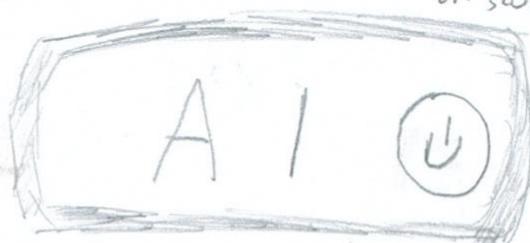
Database

Hier word gekennt of
de Audiopoint al bestaat
in de Audio Database

2 ↑ ↓ 3

Als het herkend is, pakt d'A1
een stukje audio dat DIRECT
na het ingestuurde stukje komt
en stuurt het terug maar dan

A1



Audio word opgevanger
als Audio fingerprint
naar A1

1

↑



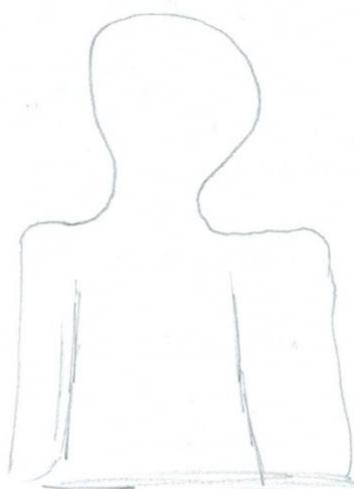
3

↓



5

Dit nieuwe stukje
audio word afgespeel



Ik ben van mening dat wat ik wil creëren, heel erg veel op Shazam lijkt, tot op de laatste stap,
de soort feedback. In plaats van de informatie van een nummer, wil ik een nieuwe fingerprint
terugkrijgen van de eerste paar seconde direct na de ingestuurde fingerprint.

Via Github had ik een paar Shazam-achtige projecten gevonden, maar ik kreeg dit dus niet aan de praat. Ik kan de code lezen, ik snap grote deels wat er gebeurt, maar ik krijg de stukken code niet werkend op mijn Macbook.

<https://ourcodeworld.com/articles/read/973/creating-your-own-shazam-identify-songs-with-python-through-audio-fingerprinting-in-ubuntu-18-04>

Dit project is gemaakt voor Ubuntu, een Linux operating system. Ik hoopte dat ik met Mac dit gewoon aan de praat kon krijgen maar voor nu werkte dat niet.

Ik ga mijn komende tijd hieraan besteden zodat ik het hopelijk wel werkend krijg, als dit gelukt is hoop ik de output te kunnen aanpassen naar iets anders.

Als dit gelukt is, dan heb ik als het goed is iets gemaakt dat redelijk op het idee lijkt dat ik zou willen creëren. Als dit gelukt is, moet ik het zien te plaatsen in een apparaat en hopend werkelijk krijgen zoals ik dit wil.

Hoe ik precies mijn software ga krijgen in een stukje hardware is nieuw voor mij. Ik heb zelf gelukkig wat contacten waar ik hier meer informatie over zou kunnen vragen, maar voor nu ligt daar nog niet mijn focus op.

Uiteindelijk vond ik dat iemand de script grotendeels heeft omgezet naar een Windows variant, ik ging hier uiteindelijk mee worstelen en dit heb ik gedocumenteerd staat onder Prototypes.

Ubuntu Operating System

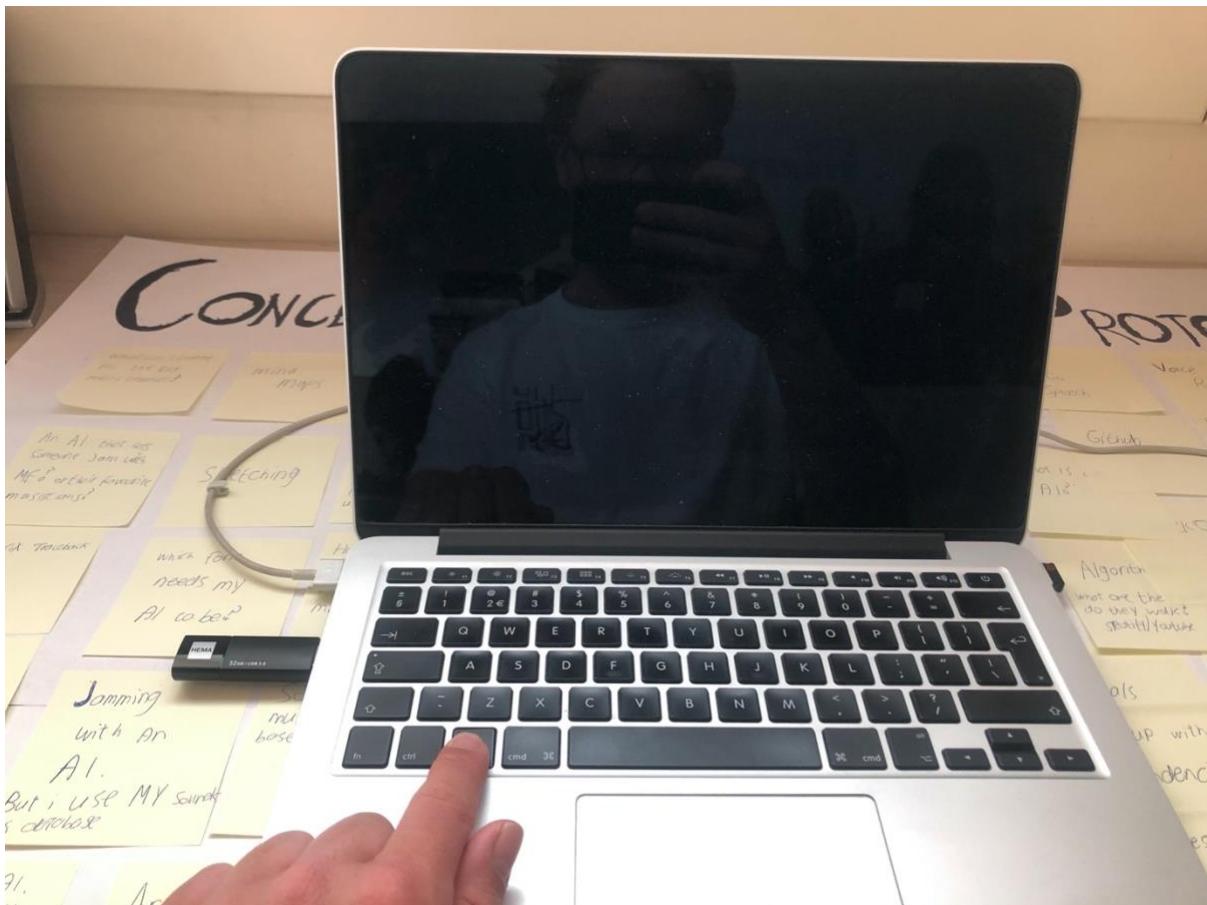
Omdat ik vooral ben bezig geweest met programmeren liep ik tegen een aantal problemen aan, namelijk dat ik niet alle projecten kon gebruiken op mijn laptop.

Zelf heb ik een Apple Macbook Pro maar veel projecten die ik tegenkwam zijn voor Ubuntu. Ubuntu is een open source, gratis operating system dat weer een variant van Linux is.

Ik heb in het verleden Windows op mijn Macbook gehad d.m.v. bootcamp. Wat dit inhoudt is dat je een deel van je harde schijf vrijmaakt en toewijst aan iets anders. In dit geval, een 2^e operating system zoals Windows of Linux. Mijn Macbook zit nu erg vol, dus dit was voor mij niet te doen. Er zijn wel andere opties, en dat heb ik dan ook heel erg uitgebreid geprobeerd voor elkaar te krijgen.



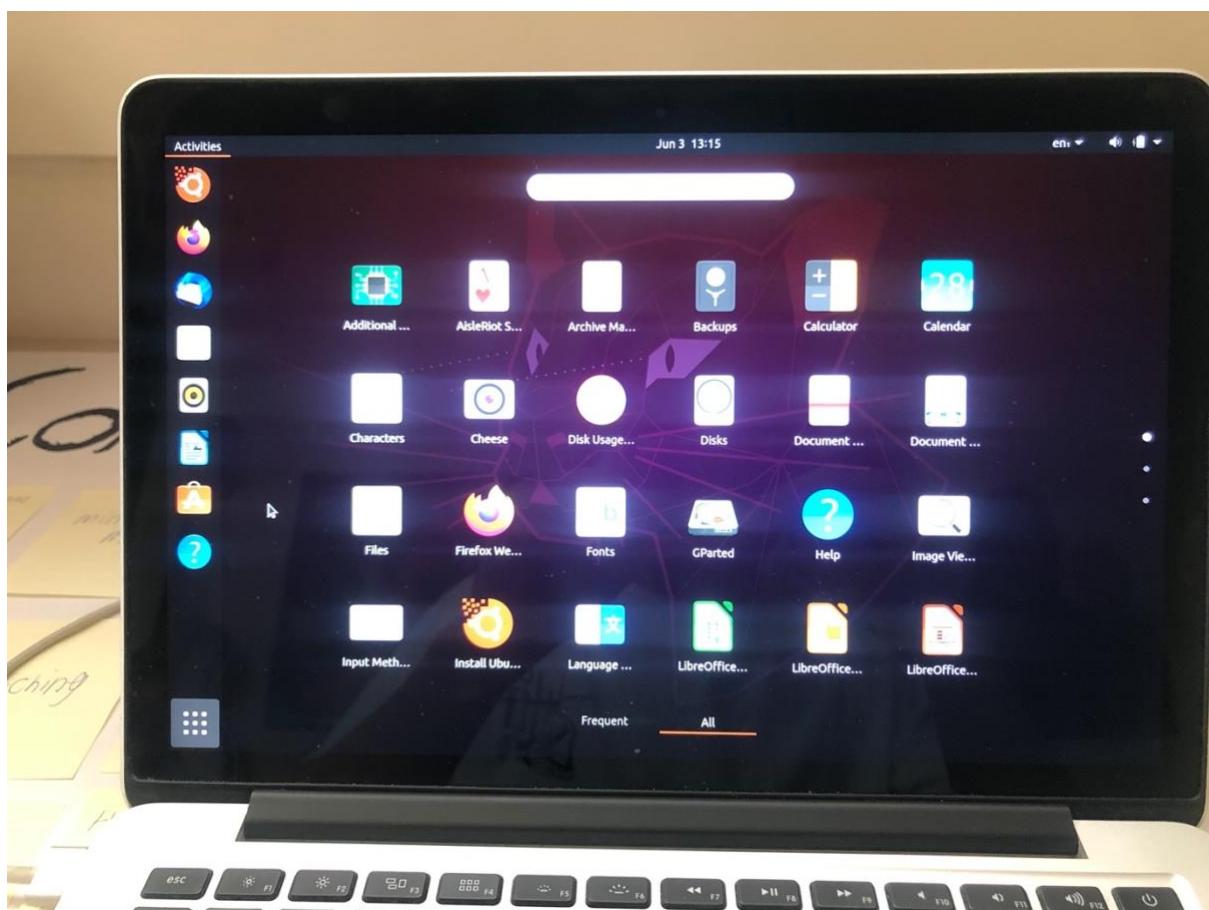
Namelijk door bootable USB-sticks te gebruiken. Ik heb hier een 8gb USB 2.0 stick en 32gb USB 3.0 stick voor gebruikt. (De 32gb heb ik later pas aangeschaft speciaal hiervoor).



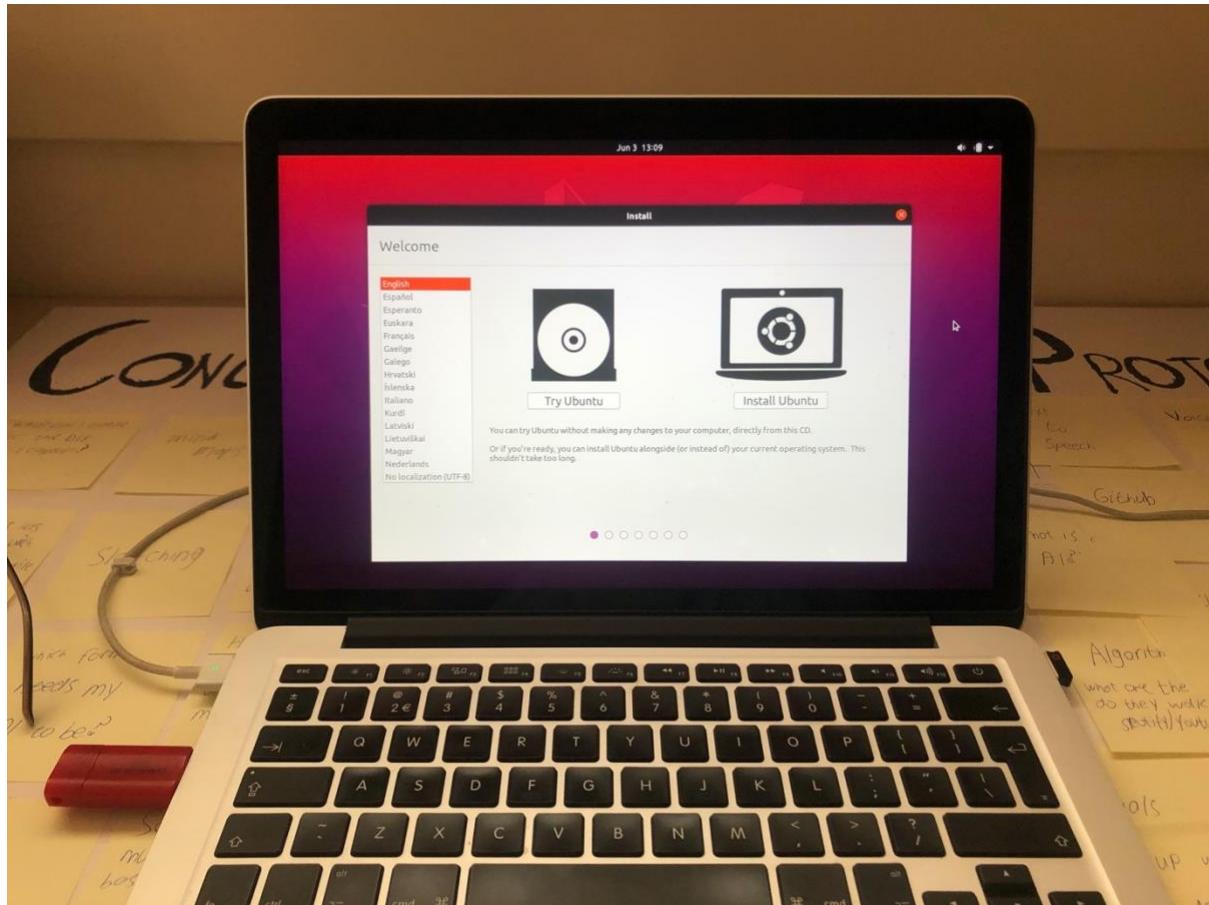
Als je apparaat uitstaat met een bootable usb erin, dan moet je de op de aan knop drukken en meteen de ALT-toets indrukken. Als je dit doet krijg je hierna de optie om een opstartschrift te kiezen, die van de Mac zelf, of in dit geval Ubuntu. Ik heb hier voor Ubuntu gekozen.



Als eerste wordt het natuurlijk opgestart, dit duurt al heel erg lang. Op een standaard bootable usb duurt dit wel al een heel stuk korter. Op de persistent usb duurt dit echt 10 minuten.



Persoonlijk vind ik dat als het eenmaal opgestart is, Ubuntu er erg mooi uitziet. Veel dingen lijken erg op Mac OS. De bovenstaande afbeeldingen zijn van de Bootable Persistant usb, hier staat dus Ubuntu volledig op geïnstalleerd. Ik kan die usb in elke willekeurige pc of laptop stoppen en het apparaat opstarten op Ubuntu.



De rode usb is alleen bedoeld om het voor de 1e keer op te kunnen starten, bijvoorbeeld om het op die manier te installeren, of om het simpelweg gewoon even te testen. Alles dat je doet tijdens de 'Try Ubuntu' optie gaat verloren als je de laptop uitzet, hier kwam ik op de pijnlijke manier achter. Ik had heel wat dingen al geprobeerd en begon al code te typen, en toen ik de dag erna mijn laptop aanzette op Ubuntu was alles weg.

Het eerste wat ik hierover wil vertellen is dat je duidelijk moet weten wat voor een type usb je ervoor gebruikt, een 2.0 is aanzienlijk trager als een 3.0. Ik had op beide sticks Ubuntu gezet en zag echt veel verschil, ook al was de rode 2.0 stick niet mega traag, de 3.0 stick was echt mega snel.

Hierna heb ik een persistente storage bootable usb gemaakt (op de zwarte usb). Dit houdt in dat als bijvoorbeeld de het opstartbestand van Ubuntu 3 gigabyte is, je de rest van je opslag op de stick kan gebruiken als harde schijf. Je hebt dan letterlijk heel Ubuntu + opslag op 1 usb stick staan.

Om dit te gaan proberen te maken heb ik deze tutorial gevolgd.

<https://www.howtogeek.com/howto/14912/create-a-persistent-bootable-ubuntu-usb-flash-drive/>

Het probleem is dat de code om dit te doen alleen werkt in de terminal van een Ubuntu machine. Ik probeerde dit eerst te doen met dezelfde stick als waar ik Ubuntu mee kon opstarten, maar dit eindigde in een hele hoop problemen. Ik heb een vriend die programmeur is en werkt met Ubuntu, hij kwam langs om dit te fixen, uiteindelijk had ik een Ubuntu Bootable USB stick met Persistente storage!

Het enige nadeel.... Dit is super, super en super traag. Het is natuurlijk wel logisch dat als ik alles op een 32gb usb zet dat het niet mega snel is, maar dit sloeg alles. Ik kon hier totaal niet mee werken want alleen een mapje openen kon al minuten duren, dit is dus niet hoe het hoort te werken.

Toevallig dat rond deze tijd mijn vader een nieuwe Windows laptop moest aanschaffen voor zijn werk, en ik heb deze geclaimd voor programmeren. Ondanks dat Ubuntu het dus niet is geworden uiteindelijk, heb ik nu wel de optie om een Windows laptop te gebruiken.



Gebruikte Methoden

Om op het punt te komen waar ik nu ben gekomen heb ik de volgende methoden gebruikt.

Desk Research – Dit is de eerste methode die ik heb gebruikt, dit deed ik door bijvoorbeeld YouTube tutorials te kijken en bijvoorbeeld artikelen opzoeken en bestuderen. Grotendeels van mijn tijd heb ik besteed aan het researchen van AI, programmeren en sound recognition. Ik heb ook veel tijd gespendeerd aan het kijken hoe AI nu wordt ingezet in, in het verleden is ingezet en hoe het in de toekomst ingezet kan gaan worden. Ik wilde veel informatie gaan halen uit mijn prototypes, deze zou ik nooit kunnen maken door dit onderzoek.

Mind Mapping – Om in het begin van dit project op ideeën te komen, heb ik veel mind maps gemaakt. Het gaf me meer inzicht in mijn eigen denk proces omdat ik het visueel kon maken.

Interview – Doormiddel van teams had ik met wat mensen gesproken over mijn project, ideeën en wat ik eruit wilde proberen te halen. Ik probeerde deze gesprekken te houden met mensen die ook muzikant zijn zodat ik waardevolle feedback zou terugkrijgen.

Tinkeren – Dit is voor mij een beetje een speciale. Ik probeerde informatie op te zoeken doormiddel van deskresearch, als dit interessant was ging er ermee spelen – tinkeren. Als dit echt potentie had ging ik beginnen aan een prototype. Dit werkt voor mij veel fijner als alleen in theorie denken op papier bijvoorbeeld.

Prototyping – Dit is uiteindelijk de methode waar ik de meeste tijd in heb zitten. Ik vond het erg waardevol om bepaalde ideeën uit te werken tot een prototype zodat ik vervolgens kon testen of iets klopte of niet. Het was ook erg interessant om door prototypes te maken op andere inzichten uit te komen als ik eigenlijk had verwacht.

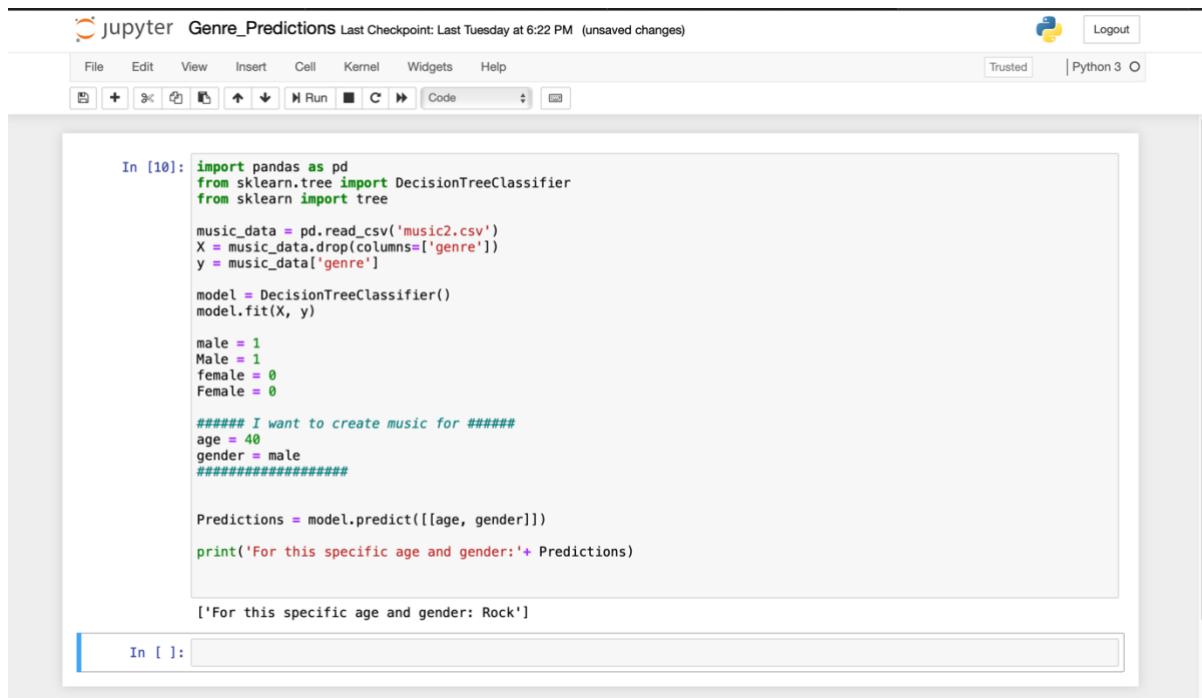
Prototypes:

Hoe kan ik software inzetten in de keuze van muziekgenres?

Onderstaande prototype is wat ik gebruikte in het begin van het lab als Artefact. Ik wilde een AI-programma hebben die een muzikant kan helpen met het schrijven van nieuwe muziek. Nu weet ik na mijn onderzoek gedaan te hebben, dat dit te gecompliceerd is als klein prototype.

Als prototype heb ik een klein programma geschreven dat op basis van leeftijd en geslacht een aanbeveling geeft voor type genre. Een muzikant (vooral handig voor digitale muziek) kan bijvoorbeeld invoeren, ‘ik wil een nummer maken voor mannen van circa 25 jaar oud’. Het programma geeft hierop een aanbeveling van wat voor soort muziek deze doelgroep houdt.

Hoe ziet het eruit?



The screenshot shows a Jupyter Notebook interface. The title bar says "jupyter Genre_Predictions Last Checkpoint: Last Tuesday at 6:22 PM (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and various cell type icons. A status bar at the bottom right shows "Trusted" and "Python 3".

In the main area, cell In [10] contains the following Python code:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

music_data = pd.read_csv('music2.csv')
X = music_data.drop(columns=['genre'])
y = music_data['genre']

model = DecisionTreeClassifier()
model.fit(X, y)

male = 1
Male = 1
female = 0
Female = 0

##### I want to create music for #####
age = 40
gender = male
#####
#####

Predictions = model.predict([[age, gender]])

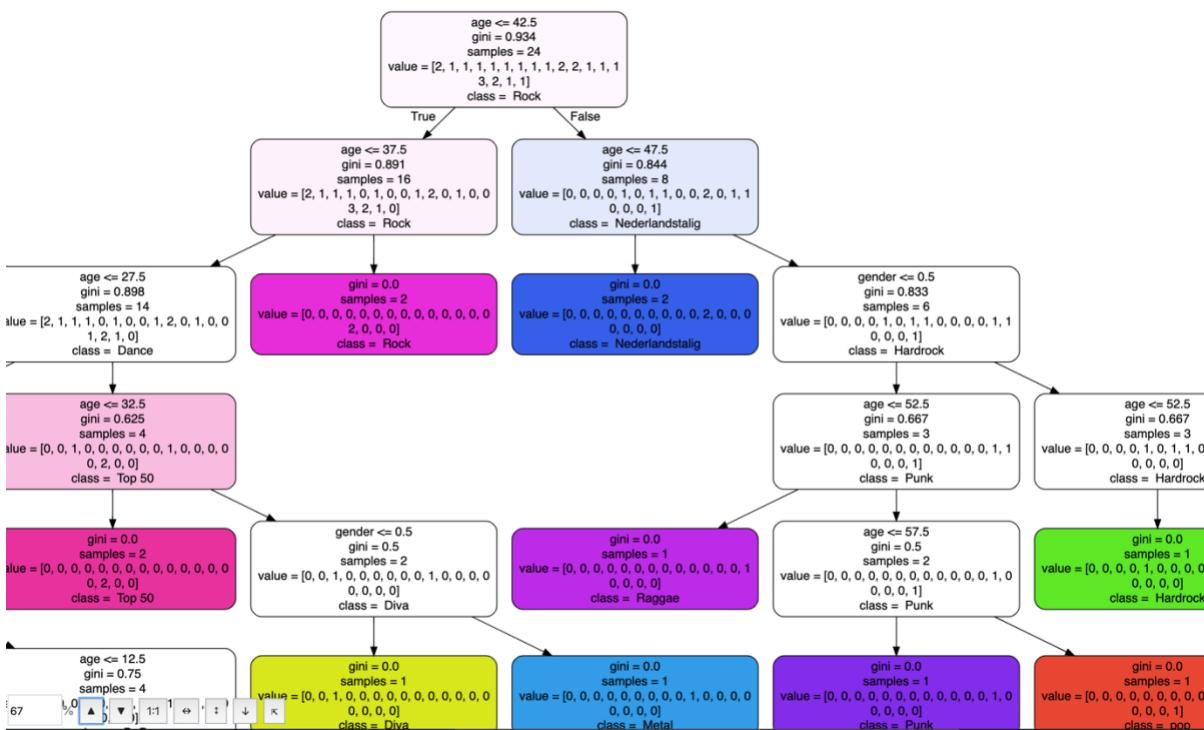
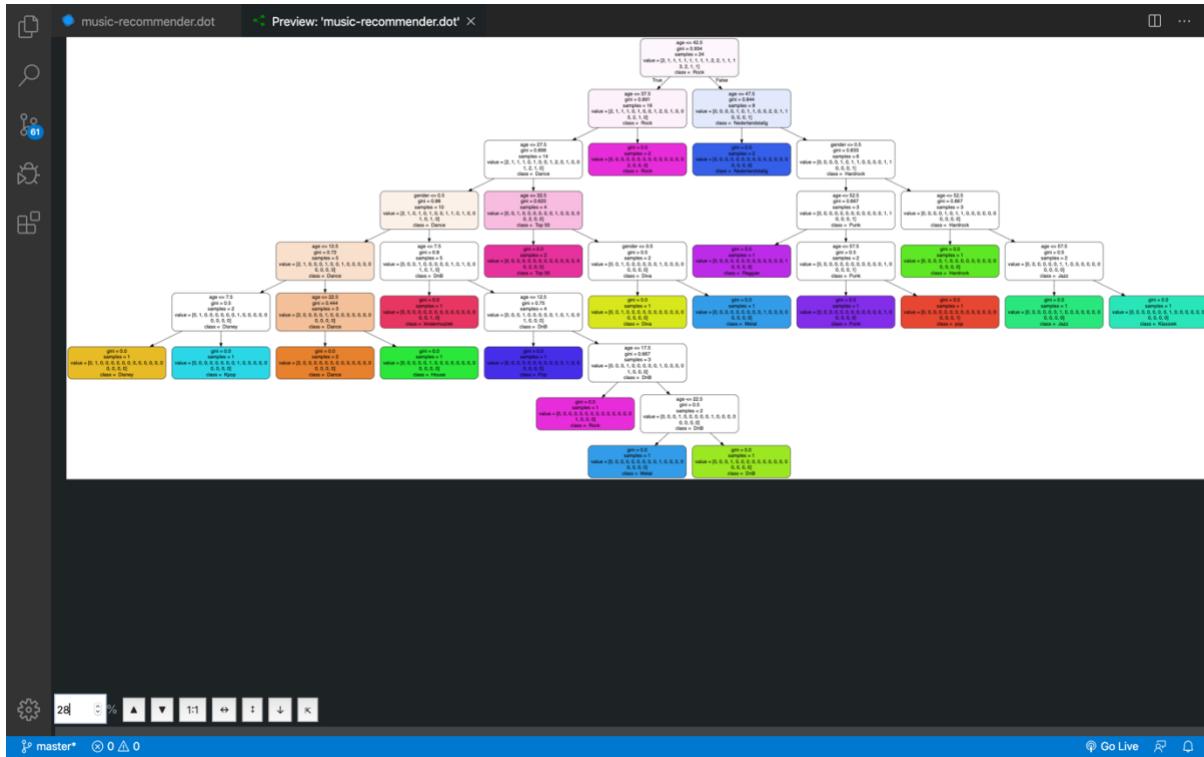
print('For this specific age and gender:' + Predictions)
```

The output cell below shows the result:

```
['For this specific age and gender: Rock']
```

In de screenshot heb ik als leeftijd 40 genomen en heb geslacht als man. Ik gebruik hiervoor Jupyter Notebook, dit is een van de meest standaard editors als je met Python werkt aangezien je meteen ziet wat er gebeurt, deze editor wordt al gerund in de terminal en wat u hier ziet draait via mijn localhost.

Ik kan hier ook de statistieken visueel maken via een extension in Visual Studio. Je kan hier zien hoeveel procent het welk genre zou aanraden. Aangezien de database niet erg groot is met data erin is het visuele model niet optimaal als hoe het zou kunnen zijn.



Voor dit prototype ben ik dus bezig geweest met kansberekeningen. Toen ik verder ging met dit project kwam ik er al snel achter dat dit prototype weinig meerwaarde heeft. Ik ben hier dus ook niet verder mee gegaan.

Het was een leuk project om mee te werken, vooral aangezien ik nog nooit dit soort dingen heb gedaan, maar uiteindelijk is dit totaal niet iets waar ik iets mee wil gaan doen.

Jam with an AI, in different forms

Wat wil ik gaan onderzoeken?

Hoe kan ik een AI maken waar ik mee muziek kan schrijven, samen mee kan jammen.

Hoe wil ik dit gaan realiseren?

Door te gaan onderzoeken hoe dit in elkaar zit en dit vervolgens te gaan maken in de taal Python.

Waarom is dit relevant?

Vaak als er muziek wordt geschreven voor een band, dan komt 1 persoon met een stukje muziek en iemand anders vult dit aan (dit is een van de mogelijkheden). Dit proces gaat heen en weer totdat er een nummer is ontstaan, maar wat als je hier de tweede persoon weghaalt?

Wat als ik iets inspeel en de AI reageert hierop met een nieuw stukje. Op deze manier kan ik samen met de AI-muziek schrijven beetje bij beetje. Ik gebruik de AI dan als tool om nieuwe creativiteit uit op te doen.

De theorie hierachter:

De theorie zoals ik denk dat het werkt, op basis van het onderzoek dat ik tot nu gedaan heb is als volgt:

Ik speel een stukje in met mijn gitaar, bijvoorbeeld het E-akkoord. De AI heeft een goed aantal nummers in zijn database die goed geanalyseerd zijn. Op het moment dat de AI het E-akkoord hoort, gaat hij kijken in zijn database van, waar in welke nummers komt dit akkoord ook voor? Als de AI daar een antwoord op heeft, kijkt hij naar wat voor een akkoord komt dan na het E-akkoord in dit nummer?

Het resultaat, de output van de AI is dat het reageert met een ander akkoord dat hij heeft bedacht op basis van de nummers uit zijn database.

Hoe kan ik een AI in een stukje hardware stoppen die mijn bandleden zouden kunnen vervangen voor het schrijven van een nummer?

Wat als ik een device voor me heb staan die hoort wat er gebeurt en hier vervolgens op reageert.

Ik speel bijvoorbeeld een akkoord of een riffje op mijn gitaar, en het AI-device reageert hierop met iets wat erna zou kunnen komen. Op deze manier kan je muziek schrijven met een AI.

Ik vind vooral de interactie interessant tussen gebruiker en de AI. Wat zijn nu de meeste basisdingen die ik nodig heb om de interactie te creëren tussen AI en de gebruiker?

Voor de interactie:

- Geluid
- Frequentie
- Toon
- Volume

Voor de gebruiker:

- Instrument
- Idee van waar hij mee bezig is
- Het apparaat

Voor de AI

- Database van nummers + kennis hiervan
- Het kunnen vertalen van wat hij hoort, en hier op kunnen reageren

Het meest essentiële proces wat er gebeurt tussen de AI en de gebruiker om dus te kunnen jammen met de AI, is het feit dat de AI moet kunnen horen wat er gebeurt en hoe hij dit kan verwerken en een antwoord kan geven.

Op het moment dat de AI niet goed weet wat hij opvangt, kan hij ook nooit iets teruggeven.

Design:

Ik denk dat het design moet gaan lijken op een bluetooth speaker of iets in de trant van een Google Home. Het apparaat moet compact zijn en niet te groot. Hoe groter het apparaat wordt hoe minder makkelijk het is om snel mee te nemen en te gebruiken.

Persoonlijk denk ik dat de kracht ligt in de portability en het moet makkelijk kunnen werken. Als je echt voor een kast van een apparaat moet gaan zitten dan is je interactie met het apparaat ook heel anders.

Mogelijke vormen:

- Speaker
- Google home achtig
- Blokje
- Ketting
- Horloge
- Via een laptop
- Tv
- Telefoon
- Afstandsbediening
- Kastje aan je gitaar/instrument



Ik heb een aantal van deze items thuis gezocht en heb een kleine test uitgevoerd. Ik probeerde het product na te bootsen met constant andere vormen voor de AI. U vindt de video hier:

<https://youtu.be/8YjzwhauXDRKBQ>

Na aanleiding van een interview en de reacties van mensen die het filmpje hebben gezien:

Kan ik zeggen dat de vorm ‘speaker’ het normaalste wordt ervaren onder de ondervraagden. Dit is voornamelijk omdat het makkelijk is. Puur omdat dit een hele simpele vorm is, geeft dit gemak in de ogen van de ondervraagden. Qua interactie geeft dit een ‘relaxed’ gevoel, als je kijkt naar een laptop dan wordt het gauw weer als werken gezien om muziek op te gaan nemen. Omdat het een tool moet zijn om te helpen met inspiratie opdoen, is het noodzakelijk dat het dit omarmt en iets mee doet, als je iets pakt dat erg serieus wordt gevonden, dit gevoel helpt niet echt mee om een situatie te creëren waar de gebruiker inspiratie uit kan halen.

Het is voor mij een heel waardevol inzicht om te weten welke vorm, door de ondervraagden, het beste is voor dit soort product. Ik ben zelf erg geïnteresseerd in de interactie met deze AI. Ik heb het gevoel dat de vorm daar een van de belangrijkste factor in is.



Met of geen telefoon app voor de instellingen?

Hoe connect ik het device aan een database?

↳ Hoe krijgt het internet?

↳ via mobiele app? laptop/Desktop
met/zonder knopjes? ken het offline? WiFi/LG verbruik

Hoe reageert het op verschillende genres instrumenten?

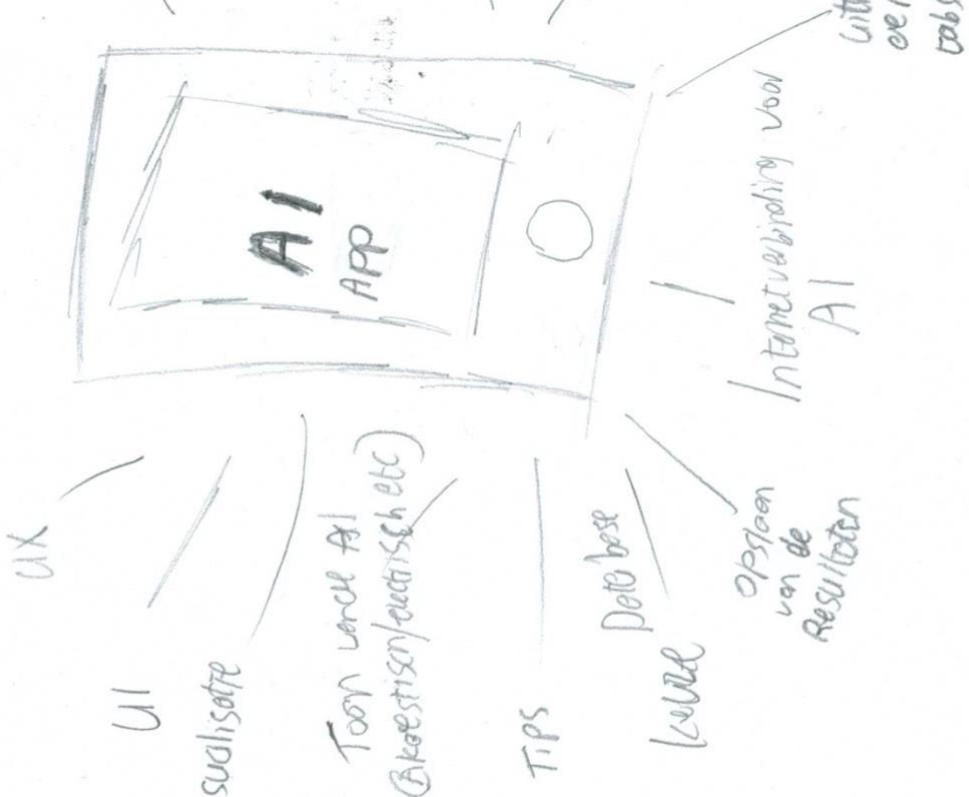
+ punten aan concept -

- cilindrisch (speaker) vorm
- portable
- compact design
- 'logische' interactie
- 'Relaxed' sfeer als Random de speaker
- internet.
- Interface

- laptop app
- (mirrored) portable
- meer kracht
- meer mogelijkheden inter
- UX
- minder relaxed sfeer
- voelt 'serieuze' aan
- 'slechtere' interactie

Ik had verder ook nog een aantal dingen genoteerd over mijn onderzoek, vooral wat vragen waar ik nog antwoorden op moet gaan zoeken.

Voorstellen van app bij de AI



Ik zat ook nog met het idee om er een mobiele app voor bij te maken om het apparaat te kunnen instellen. Als je alle mogelijkheden moet verwerken in bijvoorbeeld een speaker, dan wordt het misschien veel minder mooi als dat je gewoon een app hebt op je telefoon/laptop waar je je AI aan moet linken. Hierboven ziet u hier een woordweb van.

Shazam applicatie

Zoals eerder aangegeven bij mijn research over Shazam, ik wilde een soort Shazam maken, maar dan net ietsje anders. Ik ben al heel erg lang hiermee bezig, ik heb het geprobeerd op Mac, Windows en Ubuntu.

Oorspronkelijk was dit project een Ubuntu project:

<https://ourcodeworld.com/articles/read/973/creating-your-own-shazam-identify-songs-with-python-through-audio-fingerprinting-in-ubuntu-18-04>

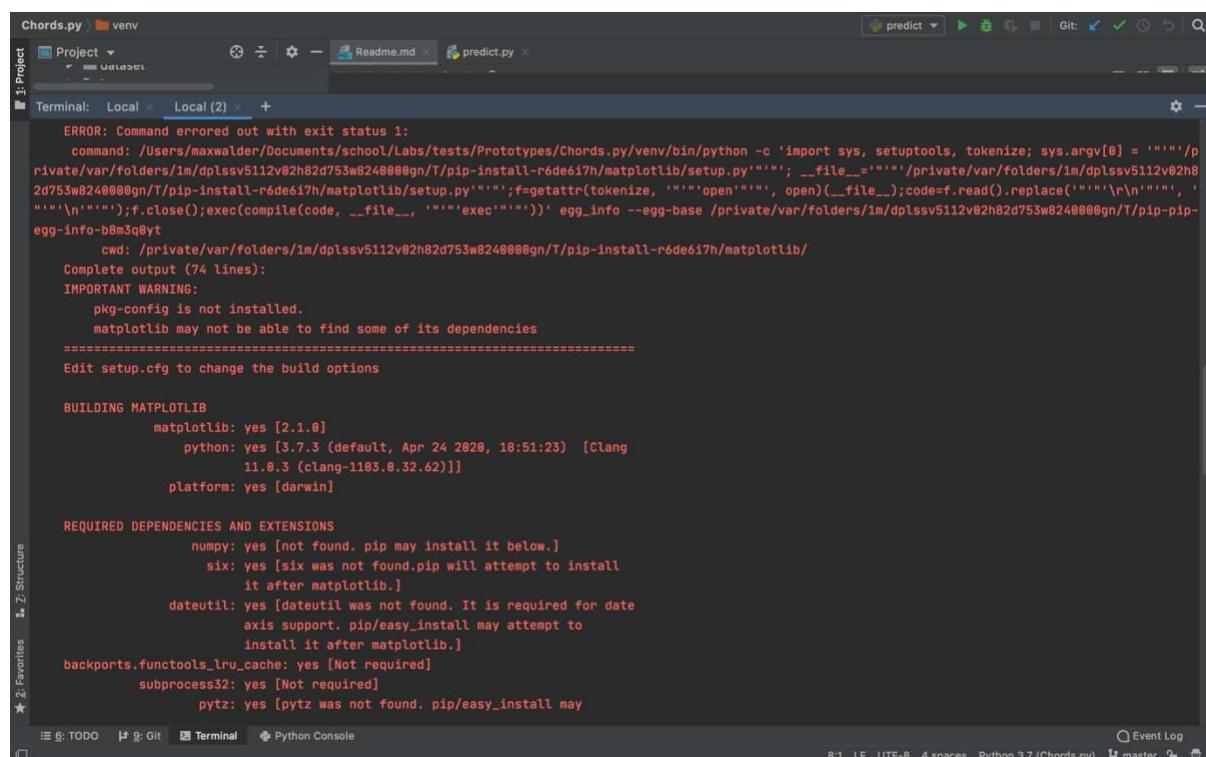
Het ziet er allemaal heel erg simpel uit, maar natuurlijk is dat niet het geval. Om te beginnen is dit dus een Ubuntu project waar gebruik wordt gemaakt van Python 2.7. Dit is een hele oude versie van Python.

Chords Recognition

Bij het zoeken naar andere mogelijkheden voor music recognition kwam ik dit project tegen op github: <https://github.com/amrondondp/Chords.py>

Dit is een university project van een student uit 2017. Het project is eigenlijk heel simpel, de applicatie kan gitaraakkoorden herkennen.

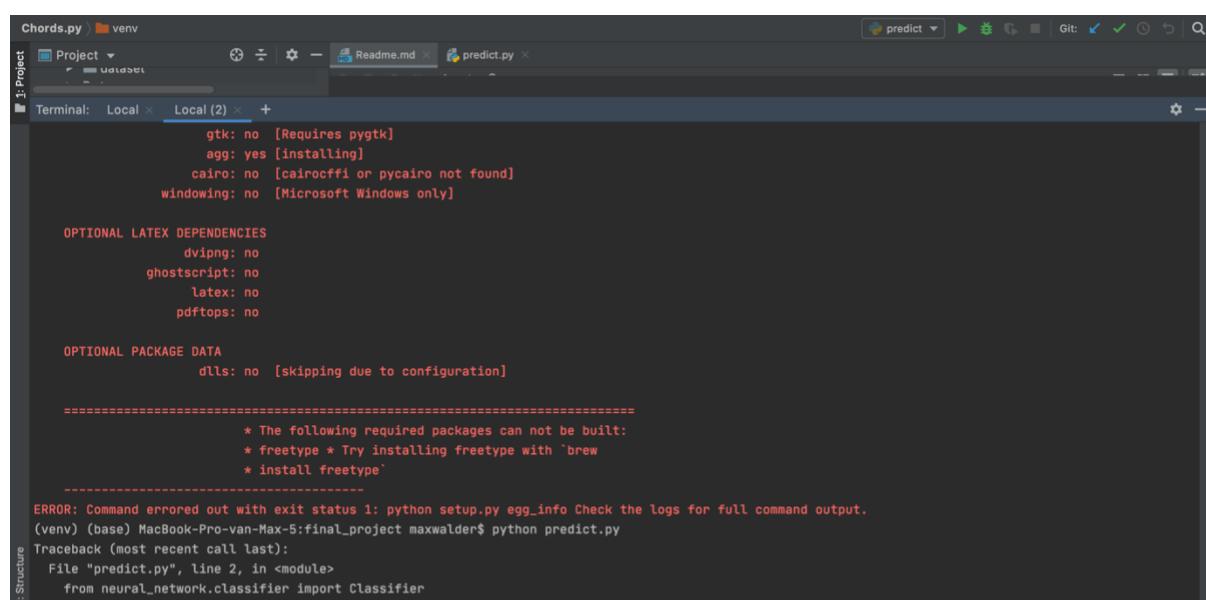
Jammer genoeg kreeg ik dit beslist niet werkend.



```
Chords.py  venv predict README.md predict.py
Project  Local  Local (2) + Terminal: Local (2) + ERROR: Command errored out with exit status 1:
command: /Users/maxwalder/Documents/school/Labs/tests/Prototypes/Chords.py/venv/bin/python -c 'import sys, setuptools, tokenize; sys.argv[0] = """/private/var/folders/im/dplssv5112v02h82d753w8240000gn/T/pip-install-r6de617h/matplotlib/setup.py"""; __file__="""'/private/var/folders/im/dplssv5112v02h82d753w8240000gn/T/pip-install-r6de617h/matplotlib/setup.py""";f=getattr(tokenize, """'open'""", open)(__file__);code=f.read().replace("""'\r\n'""", """'\n'""");f.close();exec(compile(code, __file__, """'exec'"""))' egg_info --egg-base /private/var/folders/im/dplssv5112v02h82d753w8240000gn/T/pip-pip-egg-info-b8m3q@vt cwd: /private/var/folders/im/dplssv5112v02h82d753w8240000gn/T/pip-install-r6de617h/matplotlib/ Complete output (74 lines):
=====
IMPORTANT WARNING:
  pkg-config is not installed.
  matplotlib may not be able to find some of its dependencies
=====
Edit setup.cfg to change the build options

BUILDING MATPLOTLIB
  matplotlib: yes [2.1.0]
    python: yes [3.7.3 (default, Apr 24 2020, 18:51:23)  [Clang
      11.0.3 (clang-1103.0.32.62)]]
    platform: yes [darwin]

REQUIRED DEPENDENCIES AND EXTENSIONS
  numpy: yes [not found. pip may install it below.]
  six: yes [six was not found. pip will attempt to install
    it after matplotlib.]
  dateutil: yes [dateutil was not found. It is required for date
    axis support. pip/easy_install may attempt to
    install it after matplotlib.]
  backports.functools_lru_cache: yes [Not required]
  subprocess32: yes [Not required]
  pytz: yes [pytz was not found. pip/easy_install may
    install it below.]
```



```
Chords.py  venv predict README.md predict.py
Project  Local  Local (2) + Terminal: Local (2) + gtk: no  [Requires pygtk]
  agg: yes  [installing]
  cairo: no  [cairocffi or pycairo not found]
  windowing: no  [Microsoft Windows only]

OPTIONAL LATEX DEPENDENCIES
  dvipng: no
  ghostscript: no
  latex: no
  pdftops: no

OPTIONAL PACKAGE DATA
  dils: no  [skipping due to configuration]
=====
* The following required packages can not be built:
  * freetype * Try installing freetype with `brew
    * install freetype`
```

=====
ERROR: Command errored out with exit status 1: python setup.py egg_info Check the logs for full command output.
(venv) (base) MacBook-Pro-van-Max-5:final_project maxwalder\$ python predict.py
Traceback (most recent call last):
 File "predict.py", line 2, in <module>
 from neural_network.classifier import Classifier

Dit project is gemaakt in Ubuntu, ik hoopte dat ik het hierdoor alsnog werkend kreeg aangezien Mac relatief dichtbij Ubuntu (Linux) ligt normaal gesproken.

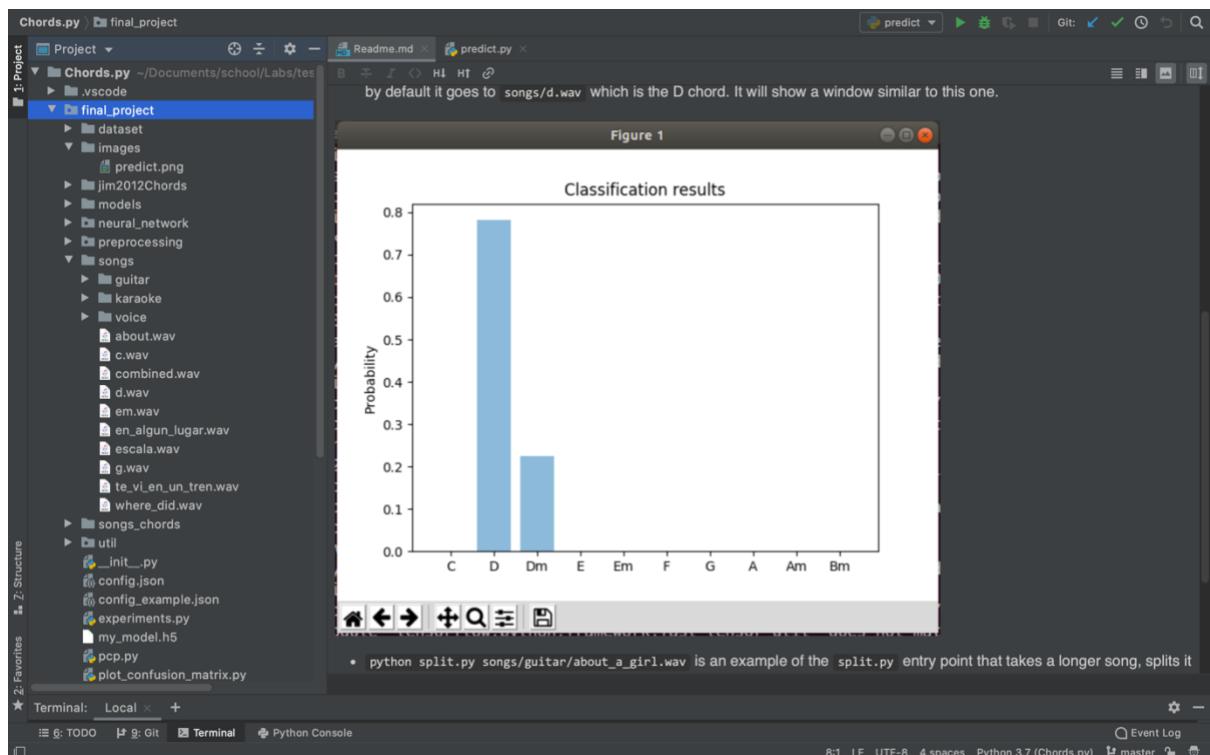
```
Chord Recognition in Python

NOTE this was tested using the following setup:

os: Ubuntu 18.04

python --version
Python 3.6.9
```

De packages die gebruikt zijn voor dit project werken dus niet allemaal voor Mac, ik heb op dit moment ook te weinig ervaring om te weten hoe ik hier oplossingen voor kan vinden. Ik kan de keus maken om hier heel veel tijd aan te besteden, maar dat vind ik voor nu niet de beste oplossing.



Het project zelf was veelbelovend, maar ik krijg dit nu gewoon niet werkend.

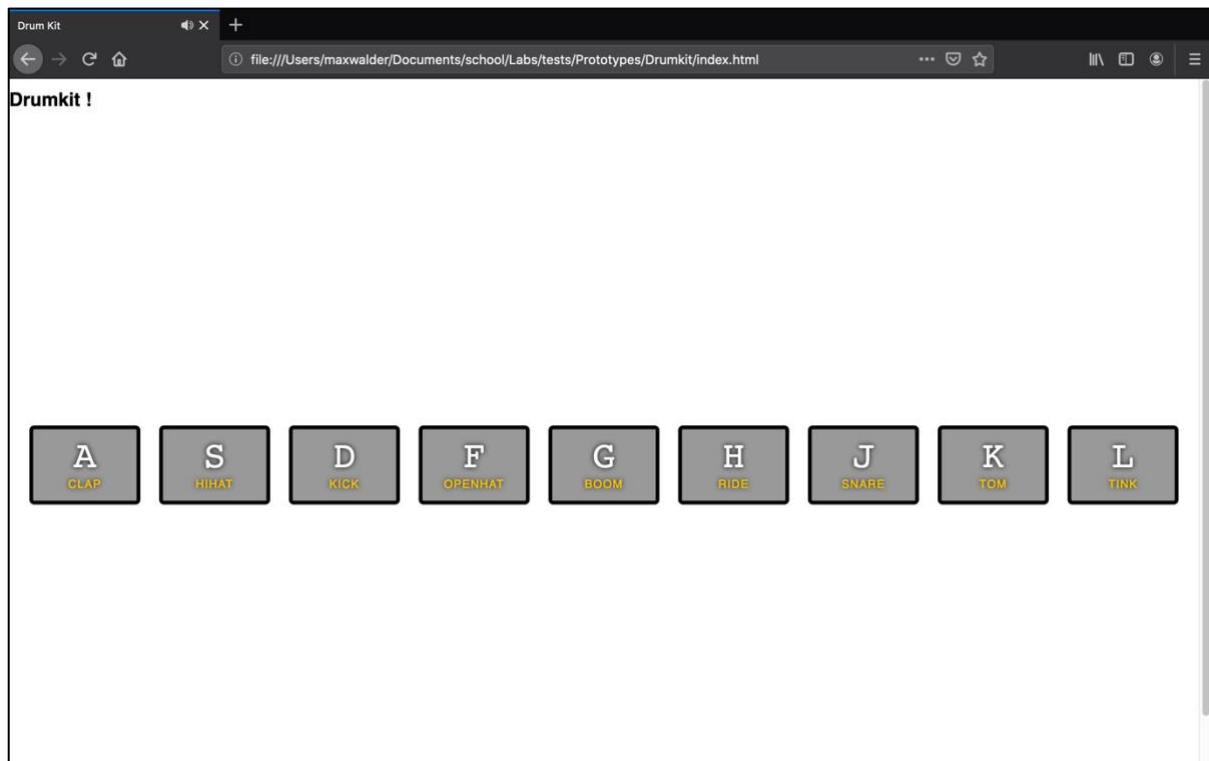
Vandaar dat ik besluit om dit project als failure te beschouwen en door te gaan naar een volgend prototype.

Ik heb de afgelopen tijd meer gespeeld met verschillende soorten talen, tekst editors en operating systems.

Drumkit

Aangezien heel veel van mijn projecten bleken te gaan falen ben ik gaan experimenteren met andere talen en typen projecten ging ik voor mijn gevoel echt even terug naar de basis van mijn project. Hoe kan ik muziek met programmeren gaan combineren, dus even het hele Artificial Intelligence plaatje wegdenken.

Ik kwam vervolgens op het idee om geluid te kunnen maken met het toetsenbord. Na wat deskresearch besloot ik om dit te gaan doen in JavaScript.



Wat ik hier gemaakt heb is een drum machine. Zodra je op een van de bovenstaande toetsen drukt, speelt er een geluidje. Op het moment dat je een toets indrukt veranderd ook de kleur van de box die eromheen zit naar geel. Ondanks dat design hier niet echt mijn hoofdfocus was, heb ik wel snel geprobeerd er nog iets moois van te maken.

Ik heb hier een filmpje van gemaakt om te demonstreren hoe de applicatie eruitziet:

```

10  <h1>Drumkit !</h1>
11  <div class="keys">
12      <div data-key="65" class="key">
13          <kbd>A</kbd>
14          <span class="sound">clap</span>
15      </div>
16      <div data-key="83" class="key">
17          <kbd>S</kbd>
18          <span class="sound">hihat</span>
19      </div>
20      <div data-key="68" class="key">
21          <kbd>D</kbd>
22          <span class="sound">kick</span>
23      </div>
24      <div data-key="70" class="key">
25          <kbd>F</kbd>
26          <span class="sound">openhat</span>
27      </div>
28      <div data-key="71" class="key">
29          <kbd>G</kbd>
30          <span class="sound">boom</span>
31      </div>
32      <div data-key="72" class="key">
33          <kbd>H</kbd>
34          <span class="sound">ride</span>
35      </div>
36      <div data-key="74" class="key">
37          <kbd>J</kbd>
38          <span class="sound">snare</span>
39      </div>
40      <div data-key="75" class="key">
41          <kbd>K</kbd>
42          <span class="sound">tom</span>
43      </div>
44      <div data-key="76" class="key">
45          <kbd>L</kbd>
46          <span class="sound">tink</span>
47      </div>
48  </div>

```

Ln 1, Col 1 Spaces: 2 UTF-8 LF HTML Prettier

De code is eigenlijk erg simpel, elke toets op het toetsenbord heeft een bepaalde code, de A-toets bijvoorbeeld is 65. Ik heb voor alle toetsen die ik wilde gebruiken ze in een div gezet, namelijk keys, individueel zitten ze ook in de div genaamd key. Dit heb ik zo gedaan om in CSS extra dingen ermee te kunnen doen, als een geheel, en individueel.

```

47  </div>
48  </div>
49
50  <audio data-key="65" src="sounds/clap.wav"></audio>
51  <audio data-key="83" src="sounds/hihat.wav"></audio>
52  <audio data-key="68" src="sounds/kick.wav"></audio>
53  <audio data-key="70" src="sounds/openhat.wav"></audio>
54  <audio data-key="71" src="sounds/boom.wav"></audio>
55  <audio data-key="72" src="sounds/ride.wav"></audio>
56  <audio data-key="74" src="sounds/snare.wav"></audio>
57  <audio data-key="75" src="sounds/tom.wav"></audio>
58  <audio data-key="76" src="sounds/tink.wav"></audio>
59
60  <script>
61      function removeTransition(e) {
62          if (e.propertyName !== 'transform') return;
63          e.target.classList.remove('playing');
64      }
65
66      function playSound(e) {
67          const audio = document.querySelector(`audio[data-key="${e.keyCode}"]`);
68          const key = document.querySelector(`div[data-key="${e.keyCode}"]`);
69          if (!audio) return;
70
71          key.classList.add('playing');
72          audio.currentTime = 0;
73          audio.play();
74      }
75
76      const keys = Array.from(document.querySelectorAll('.key'));
77      keys.forEach(key => key.addEventListener('transitionend', removeTransition));
78      window.addEventListener('keydown', playSound);
79  </script>
80
81
82  </body>
83  </html>
84

```

Ln 1, Col 1 Spaces: 2 UTF-8 LF HTML Prettier

Onder de div keys heb ik aangegeven welk geluidsfragment bij welke key hoort. Daaronder had ik meteen mijn Javascript geschreven. Deze prototype heeft alleen een HTML en CSS-bestand, voor die paar regels Javascript vond ik het niet nodig om een .js bestand voor aan te maken.

Het belangrijkste wat er gebeurt in dit deel van de code is dat er daadwerkelijk een geluid wordt afgespeeld op het moment dat de toets in word gedrukt. Dit kan je zien in de laatste zin waar ik specifiek heb gekozen voor ‘keydown’.

Om te weten welke key-codes er bij welke toets horen had ik een handige tabel gevonden op een website, namelijk: <https://css-tricks.com/snippets/javascript/javascript-keycodes/>

Op het onderstaande plaatje zie je een deel van de tabel.

Key	Code	Key	Code	Key	Code
backspace	8	e	69	numpad 8	104
tab	9	f	70	numpad 9	105
enter	13	g	71	multiply	106
shift	16	h	72	add	107
ctrl	17	i	73	subtract	109
alt	18	j	74	decimal point	110
pause/break	19	k	75	divide	111
caps lock	20	l	76	f1	112
escape	27	m	77	f2	113
(space)	32	n	78	f3	114

Resultaten:

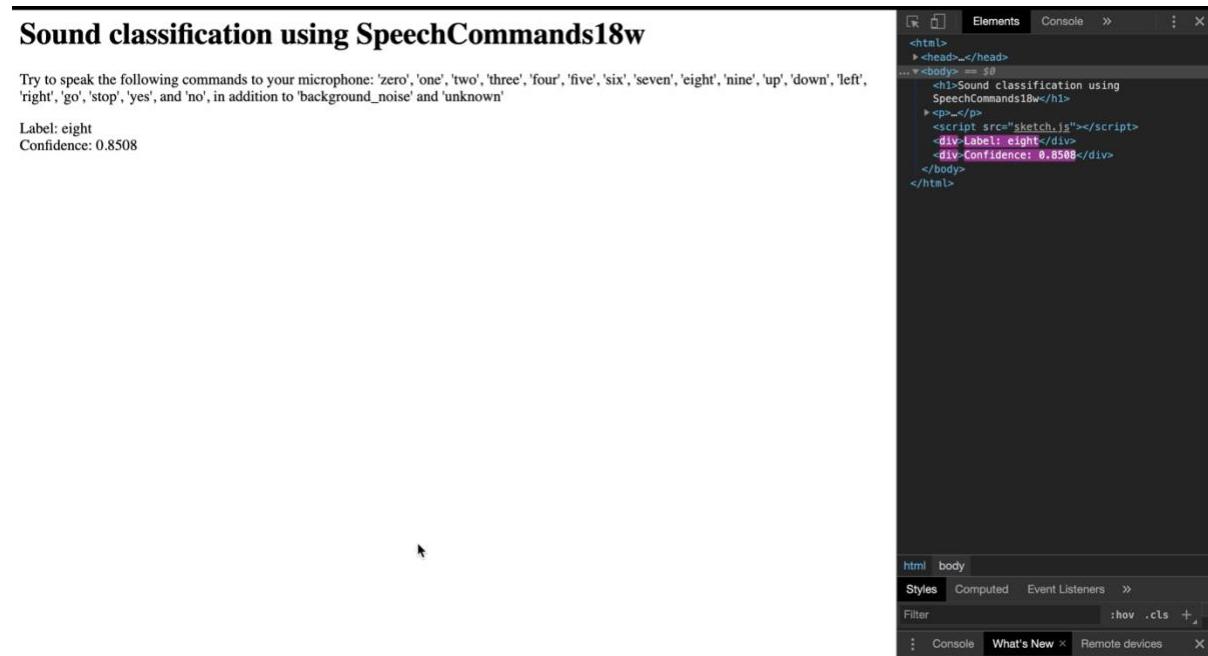
Ik wist toen ik hieraan begon dat het al een beetje ver weg was van mijn onderzoeksraag. Toch wilde ik hieraan gaan beginnen, deze prototype kostte me niet heel erg lang om te maken, maar het was wel een begin van meer. Ik pakte constant te grote projecten waar ik uiteindelijk niets mee kon. Het inicht dat ik heruit heb opgedaan is dat ik gewoon klein moet beginnen want anders dan lukt het niet. Er zijn zoveel projecten die volledig, of half outdated zijn. Door te gaan werken aan dat soort projecten als een relatief nieuw persoon met

bijvoorbeeld Python, kost heel erg veel tijd en moeite. Ik kreeg nu door ervaring een duidelijker beeld voor hoe ik dingen wel of niet moet gaan aanpakken.

Sound Classification

Nadat ik de drumkit had gemaakt ging ik spelen met speech recognition in Javascript. Ik probeerde voorheen in Python dus grote dingen te doen zonder echt te kijken naar de basics. Ik wilde nu in Javascript dit proberen te doen doormiddel van dit kleine prototype.

Dit is een project van ML5, ik heb voor dit prototype dus niet zelf veel geprogrammeerd. Het ziet er als volgt uit:



Op het moment dat ik een woord roep, dan komt er achter het woord ‘Label’ het woord te staan dat ik zeg. Eronder bij ‘Confidence’ word aangegeven hoe zeker dat het is dat het klopt. Dit is altijd een getal tussen de 0 en 1.

Ik heb hier een snelle test van gemaakt, u kunt dit filmpje zien op:
<https://www.youtube.com/watch?v=Yepj1XtKPVo&feature=youtu.nl>

De code ziet er als volgt uit:

The screenshot shows the Visual Studio Code interface with two tabs open: 'index.html' and 'sketch.js'. The 'index.html' tab contains the following code:

```
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sound classification</title>
    <script src="http://localhost:8080/ml5.js" type="text/javascript"></script>
  </head>
  <body>
    <h1>Sound classification using SpeechCommands18w</h1>
    <p>Try to speak the following commands to your microphone: 'zero', 'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine'</p>
    <script src="sketch.js"></script>
  </body>
</html>
```

The 'sketch.js' tab contains the following code:

```
// Initialize a sound classifier method with SpeechCommands18w model. A callback needs to be passed.
let classifier;
// Options for the SpeechCommands18w model, the default probabilityThreshold is 0
const options = { probabilityThreshold: 0.7 };
// Two variable to hold the label and confidence of the result
let label;
let confidence;

async function setup() {
  classifier = await ml5.soundClassifier('SpeechCommands18w', options);
  // Create 'label' and 'confidence' div to hold results
  label = document.createElement('DIV');
  label.textContent = 'label ...';
  confidence = document.createElement('DIV');
  confidence.textContent = 'Confidence ...';

  document.body.appendChild(label);
  document.body.appendChild(confidence);
  // Classify the sound from microphone in real time
  classifier.classify(gotResult);
}

setup();

// A function to run when we get any errors and the results
function gotResult(error, results) {
  // Display error in the console
  if (error) {
    console.error(error);
  }
  // The results are in an array ordered by confidence.
  console.log(results);
  // Show the first label and confidence
  label.textContent = 'Label: ' + results[0].label;
  confidence.textContent = 'Confidence: ' + results[0].confidence.toFixed(4);
}
```

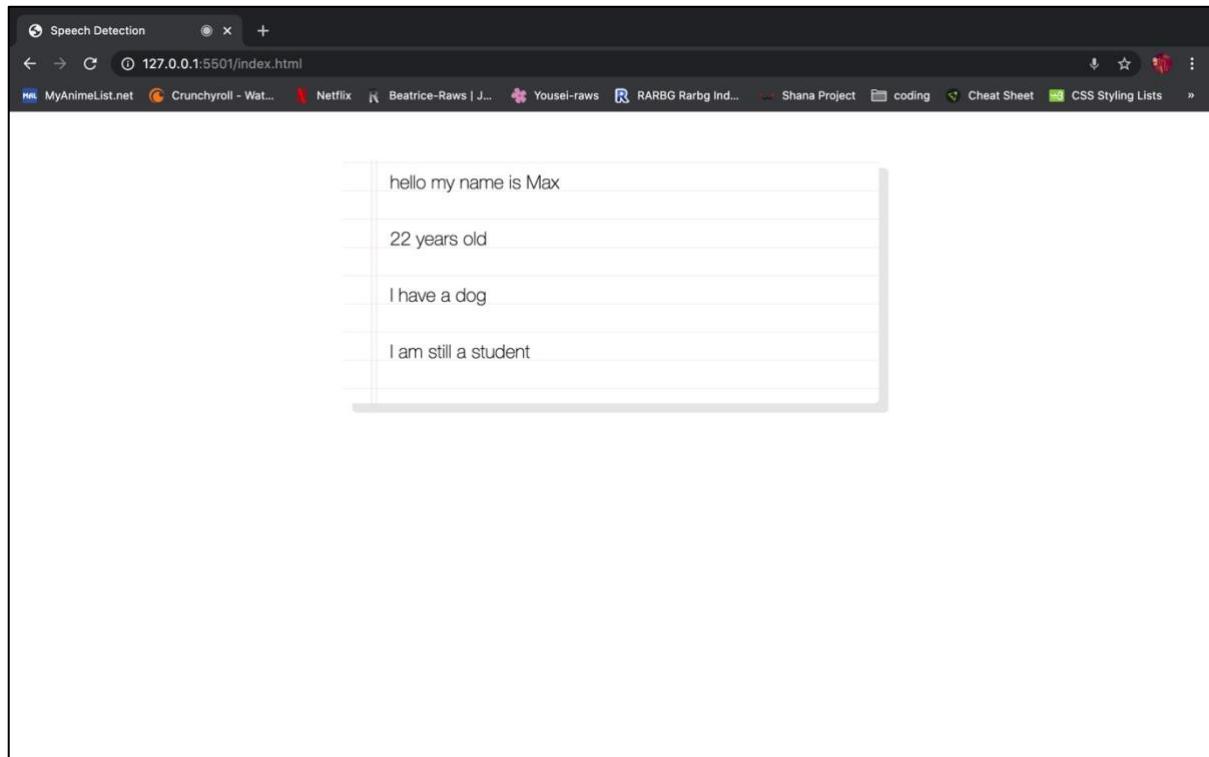
Resultaten:

Deze code vind ik niet mega interessant omdat dit niet helemaal van mij is, maar namelijk van ML5. Ik probeerde hier allerlei dingen aan aan te passen, maar dat ging niet zo soepel. Ik kon niet makkelijk dingen aanpassen aangezien dit allemaal gelinkt zit aan ML5.

Dit project was wel voor mij het begin van werken met Sound Classification

Speech Recognition

Ik wilde dus in Python gaan proberen om een AI te maken, een belangrijke interactie daarvoor is dus communiceren met de AI. Ik kreeg dit niet gemakkelijk voor elkaar en besloot om precies dit te gaan doen in een andere taal, Javascript.



Speechrecognition is een ingebouwde functie in JavaScript. Door een youtube tutorial te volgen had ik dit in elkaar gezet, letterlijk wat er gebeurd is als volgt:

Ik open de code met een live server, vervolgens doe ik praten richting mijn ingebouwde microfoon van mijn Macbook. In de browser zie je nu op het scherm staan wat ik net heb gezegd. Hij hoort dus wat ik zeg, en noteert dit vervolgens.

```
index.html
<div class="words" contenteditable>
</div>
<script>
window.SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;

const recognition = new SpeechRecognition();
recognition.interimResults = true;
recognition.lang = 'en-US';

let p = document.createElement('p');
const words = document.querySelector('.words');
words.appendChild(p);

recognition.addEventListener('result', e => {
  const transcript = Array.from(e.results)
    .map(result => result[0])
    .map(result => result.transcript)
    .join('');

  const poopScript = transcript.replace(/poop|pooh|shit|dump|gi/, '▲');
  p.textContent = poopScript;

  if (e.results[0].isFinal) {
    p = document.createElement('p');
    words.appendChild(p);
  }
});

recognition.addEventListener('end', recognition.start);

recognition.start();
</script>
<style>
  html {
    font-size: 10px;
  }
</style>
```

Bij deze code is eigenlijk alweer alleen het Javascript stukje erg interessant. Doormiddel van de SpeechRecognition() command te gebruiken roep je de functie aan. De rest van de code is eigenlijk alleen van toepassing dat alles word genoteerd wat er gezegd word.

Ik heb aan deze code weinig echt veranderd. Ik wilde zelf weten hoe precies SpeechRecognition werkte en of ik het werkend kon krijgen, aangezien ik met Python vaak vastliep

Resultaten:

Dit was een hele snelle kleine test. Ik heb dus gewoon een tutorial gevuld, maar het is wel werkend, uiteindelijk is SpeechRecognition relatief makkelijk werkend te krijgen, je zou dit ook heel makkelijk kunnen verwerken in andere applicaties of programma's. Misschien als Python niks wordt dat ik hier verder mee kan gaan werken voor een AI project.

Speech Recognition 2

Met de technieken die ik heb geleerd in de vorige prototype ben ik met een nieuwe file begonnen en heb ik iets compleet nieuws van gemaakt. In Javascript heb ik een applicatie gemaakt waar ik kan praten tegen mijn browser en de browser praat terug.

Ik wilde hier eigenlijk proberen uit te zoeken hoe Speech Recognition werkt en hoe ik dit werkend kan krijgen. Dit is dus letterlijk een iteratie op het vorige prototype.



Als je de code opent zie je alleen een button met 'Talk' erin. Zodra hierop geklikt wordt start je de applicatie, je start de recognition functie. Op het moment dat je iets zegt wordt dat onder de talk button weergegeven. In bovenstaande afbeelding had ik dus als laatste 'How are you doing' gezegd tegen de applicatie. De applicatie doet hier ook echt op reageren doormiddel van spraak.

De code is eigenlijk relatief klein. Er wordt dus teruggepraat, maar jammer genoeg zit hier nog niks van een AI achter. Alle reacties die die geeft zijn op basis van ingevoerde argumenten die ik in mijn code heb staan.

Dit is een klein programmatje dat echt werkt, uiteindelijk is het alleen een manier om met de applicatie of 'AI' te kunnen interacteren en je krijgt reactie terug.

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows a folder named "VOICE_RECOGNITION_AI" containing "app.js", "index.html", and "style.css".
- WORKBENCH**: Shows tabs for "index.html", "app.js", and "style.css".
- CODE EDITOR**: Displays the content of "app.js". The code is a JavaScript script for a voice recognition application. It includes arrays for greetings, weather, music, and restaurant responses, and logic to handle user input and speech synthesis.
- STATUS BAR**: Shows "Ln 10, Col 1" and "JavaScript".

```
const btn = document.querySelector('.talk');
const content = document.querySelector('.content');

const greetings = ['Im good how about you?', 'Doing good', 'leave me alone', 'the weather is good and'];
const weather = ['Weather is fine', 'Look through a window', 'We\'re dutch, so it cant be good'];
const music = ['i like Red Hot Chili Peppers', 'music that makes me cry', 'yea yea yea yea'];
const restaurant = ['You can\'t even go due to corona anyway', 'anything thats fast', 'Macdonalds soun'];
const corona = ['Corona is taking longer then expected'];

const SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;
const recognition = new SpeechRecognition();

recognition.onstart = function() {
    console.log("Voice is activated, you can talk now")
};

recognition.onresult = function(event) {
    const current = event.resultIndex;
    const transcript = event.results[current][0].transcript;
    content.textContent = transcript;
    readOutLoud(transcript);
};

btn.addEventListener('click', () => {
    recognition.start();
});

function readOutLoud(message){
    const speech = new SpeechSynthesisUtterance();

    speech.text = 'I Couldnt hear your bad accent';

    if(message.includes('how are you')){
        const finalText = greetings[Math.floor(Math.random() * greetings.length)];
        speech.text = finalText;
    }else if(message.includes('weather')){
        const finalText = weather[Math.floor(Math.random() * weather.length)];
        speech.text = finalText;
    }else if(message.includes('restaurant')){
        const finalText = restaurant[Math.floor(Math.random() * restaurant.length)];
        speech.text = finalText;
    }
}
```

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows a folder named "VOICE_RECOGNITION_AI" containing "app.js", "index.html", and "style.css".
- WORKBENCH**: Shows tabs for "index.html", "app.js", and "style.css".
- CODE EDITOR**: Displays the content of "app.js". The code is identical to the previous screenshot, but the status bar indicates "Ln 56, Col 1" and "JavaScript".
- STATUS BAR**: Shows "Ln 56, Col 1" and "JavaScript".

```
const transcript = event.results[current][0].transcript;
content.textContent = transcript;
readOutLoud(transcript);
};

btn.addEventListener('click', () => {
    recognition.start();
});

function readOutLoud(message){
    const speech = new SpeechSynthesisUtterance();

    speech.text = 'I Couldnt hear your bad accent';

    if(message.includes('how are you')){
        const finalText = greetings[Math.floor(Math.random() * greetings.length)];
        speech.text = finalText;
    }else if(message.includes('weather')){
        const finalText = weather[Math.floor(Math.random() * weather.length)];
        speech.text = finalText;
    }else if(message.includes('restaurant')){
        const finalText = restaurant[Math.floor(Math.random() * restaurant.length)];
        speech.text = finalText;
    }else if(message.includes('music')){
        const finalText = music[Math.floor(Math.random() * music.length)];
        speech.text = finalText;
    }else if(message.includes('corona')){
        const finalText = corona[Math.floor(Math.random() * corona.length)];
        speech.text = finalText;
    }

    speech.volume = 1;
    speech.rate = 0.8;
    speech.pitch = 1.25;
    window.speechSynthesis.speak(speech)
}
```

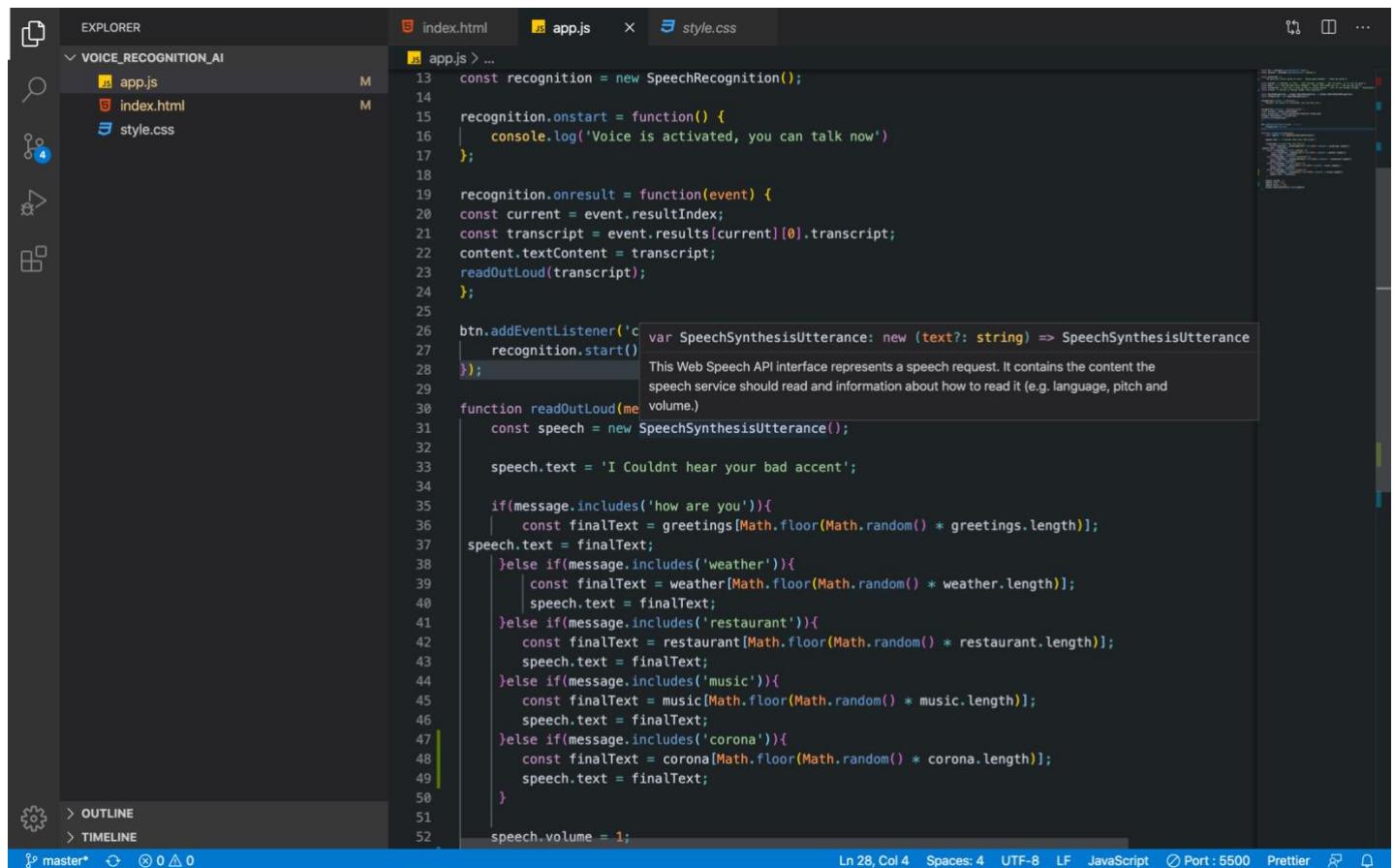
Uit de bovenste screenshot, de const greetings, weather, music, restaurant en corona zijn de antwoorden die mogelijk zijn op de vragen die ik de applicatie bijvoorbeeld stel. In de tweede screenshot zie je in de readOutLoud functie de if else statements. Deze statements zijn eigenlijk alle mogelijke opties. Hoe meer if else statements ik maak, hoe meer deze applicatie kan. Ik zeg bij elke statement eigenlijk:

```
If(message.includes('nieuw woord/zin')){
```

Const finalText=.....(nieuwe const met antwoorden zoals bijv greetings). De volgende stap is (Math.floor(Math.random() * (nieuwe const met antwoorden zoals bijv greetings).length).

Eigenlijk wat hier gewoon gebeurt, zodra dat een bepaald woord/zin wordt herkend, geef dan een random antwoord uit de aangegeven lijst met antwoorden. De math.floor optie zorgt dat je niet een rare waarde terug krijgt, maar echt wat er is aangegeven in de lijst. Vervolgens door de math.random pakt hij een willekeurig antwoord uit deze lijst.

Het terugpraten word geregeld in de readOutLoud functie, dit komt door het gebruik van de code: const speech = new SpeechSynthesisUtterance(). Hierdoor is het mogelijk dat het gekozen antwoord echt wordt omgezet naar spraak.



The screenshot shows a code editor interface with the following details:

- Explorer View:** Shows a project structure for "VOICE_RECOGNITION_AI" containing "app.js", "index.html", and "style.css".
- Code Editor:** The active file is "app.js". The code implements a speech recognition API and a speech synthesis API. It includes logic to handle various user messages like "how are you", "weather", "restaurant", "music", and "corona" by selecting a random response from a predefined list ("greetings", "weather", "restaurant", "music", "corona") and reading it out loud at a volume of 1.
- Toolbars and Status Bar:** The status bar at the bottom shows "Ln 28, Col 4" and "JavaScript". Other status indicators include "master*", "Spaces: 4", "UTF-8", "LF", "Port : 5500", and "Prettier".

```
const recognition = new SpeechRecognition();
recognition.onstart = function() {
  console.log('Voice is activated, you can talk now');
};

recognition.onresult = function(event) {
  const current = event.resultIndex;
  const transcript = event.results[current][0].transcript;
  content.textContent = transcript;
  readOutLoud(transcript);
};

btn.addEventListener('click', () => {
  recognition.start();
});

function readOutLoud(message) {
  const speech = new SpeechSynthesisUtterance();

  speech.text = 'I Couldnt hear your bad accent';

  if(message.includes('how are you')){
    const finalText = greetings[Math.floor(Math.random() * greetings.length)];
    speech.text = finalText;
  }else if(message.includes('weather')){
    const finalText = weather[Math.floor(Math.random() * weather.length)];
    speech.text = finalText;
  }else if(message.includes('restaurant')){
    const finalText = restaurant[Math.floor(Math.random() * restaurant.length)];
    speech.text = finalText;
  }else if(message.includes('music')){
    const finalText = music[Math.floor(Math.random() * music.length)];
    speech.text = finalText;
  }else if(message.includes('corona')){
    const finalText = corona[Math.floor(Math.random() * corona.length)];
    speech.text = finalText;
  }

  speech.volume = 1;
}
```

Resultaten:

Deze prototype werkt volledig naar wat de bedoeling was, het is alleen heel erg inaccuraat (er wordt geen gebruik gemaakt van API's bijvoorbeeld). Omdat er ook een random spraak uit komt, is de kwaliteit daarvan ook niet erg mooi. Het is een hele erge saaie robotachtige stem waar je de feedback van terugkrijgt. Voor mij is het nu duidelijk hoe dit werkt en waarom.

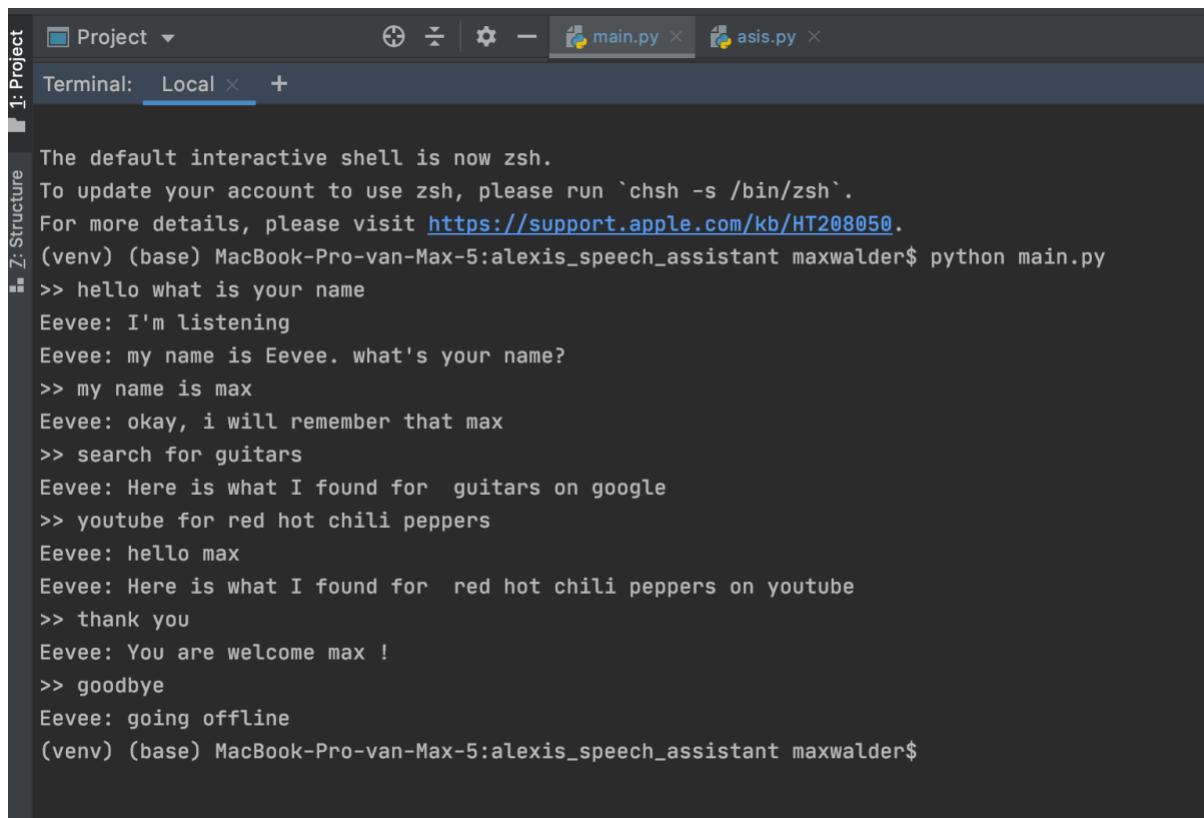
Ik zou het heel gaaf vinden om als volgende stap, dit uit te gaan breiden naar iets veel breder. Het moet dan een groter project gaan worden met meerdere functies en ik wil dit gaan proberen in Python.

Speech Recognition 3 (final)

Als 3e Speech Recognition prototype heb ik de basis gebruikt van prototype 1 en 2, maar dit keer geschreven in Python. Dit was een tutorial waar werd uitgelegd hoe je dit moest maken, maar eerst kreeg ik dit niet voor elkaar ivm dependencies.

Nu ik wat meer kennis had opgedaan was het me nu wel gelukt om dit te maken. Het was voor mij dus erg nuttig om de vorige twee prototypes te maken. Zonder dat was ik nooit doorgegaan met deze richting en had ik dit prototype nooit af kunnen maken.

Het verschil nu is dat de applicatie wordt gestart in de terminal van je tekst editor, in mijn geval, PyCharm. Bij de vorige varianten van dit prototype gebeurde alles in de browser. De feedback die je krijgt is veel beter en uitgebreider. Ook zijn er op deze manier in Python meerdere dingen mogelijk.



The screenshot shows a PyCharm interface with a terminal window open. The terminal output is as follows:

```
The default interactive shell is now zsh.  
To update your account to use zsh, please run `chsh -s /bin/zsh`.  
For more details, please visit https://support.apple.com/kb/HT208050.  
(venv) (base) MacBook-Pro-van-Max-5:alexis_speech_assistant maxwalder$ python main.py  
>> hello what is your name  
Eevee: I'm listening  
Eevee: my name is Eevee. what's your name?  
>> my name is max  
Eevee: okay, i will remember that max  
>> search for guitars  
Eevee: Here is what I found for guitars on google  
>> youtube for red hot chili peppers  
Eevee: hello max  
Eevee: Here is what I found for red hot chili peppers on youtube  
>> thank you  
Eevee: You are welcome max !  
>> goodbye  
Eevee: going offline  
(venv) (base) MacBook-Pro-van-Max-5:alexis_speech_assistant maxwalder$
```

Dit is hoe het eruit ziet in de terminal, alles dat ik zeg staat achter ‘>>’. De ‘AI’ heet Eevee. Je ziet dat onder elke vraag die ik heb gesteld, er Eevee: staat. Dit houdt dus in dat dat de reactie is die ik terugkrijg van de applicatie. Alles wat daar staat, word ook verteld doormiddel van een mp3 file die speciaal hiervoor wordt aangemaakt, afgespeeld en vervolgens verwijderd word.

Het leuke aan deze Prototype is dat ik ook echt commands kan geven. Zo is het met dit prototype voor nu mogelijk om:

- Je kan vragen hoe het gaat met de AI
- Je kan vragen wat haar naam is
- Screenshots maken
- Google search uit te voeren, er wordt een nieuw tabblad aangemaakt en gezocht naar het woord/zin die je hebt gezegd.
- YouTube search, net zoals bij Google search wordt hier een nieuw tabblad geopend, dit keer naar Youtube en er wordt meteen gezocht naar de zoekopdracht.
- Google Maps search, deze spreekt al voor zich.
- Tijd opvragen, als je vraagt naar de tijd dan wordt deze verteld.

The screenshot shows the PyCharm IDE interface with the 'Eevee_speech_assistant' project open. The 'main.py' file is the active editor. The code is written in Python and defines a class 'person' with methods for setting a name and checking for specific terms in voice data. It also includes imports for speech_recognition, playsound, gTTS, random, time, ctime, webbrowser, yfinance, os, and pyautogui. The code uses the Recognizer() class from speech_recognition to record audio from a microphone. The PyCharm interface includes a Project sidebar, a Structure view, and a Terminal tab at the bottom.

```
import speech_recognition as sr
import playsound
from gtts import gTTS
import random
import time
from time import ctime
import webbrowser
import yfinance as yf
import os
import pyautogui

class person:
    name = ''

    def setName(self, name):
        self.name = name

    def there_exists(self, terms):
        for term in terms:
            if term in voice_data:
                return True

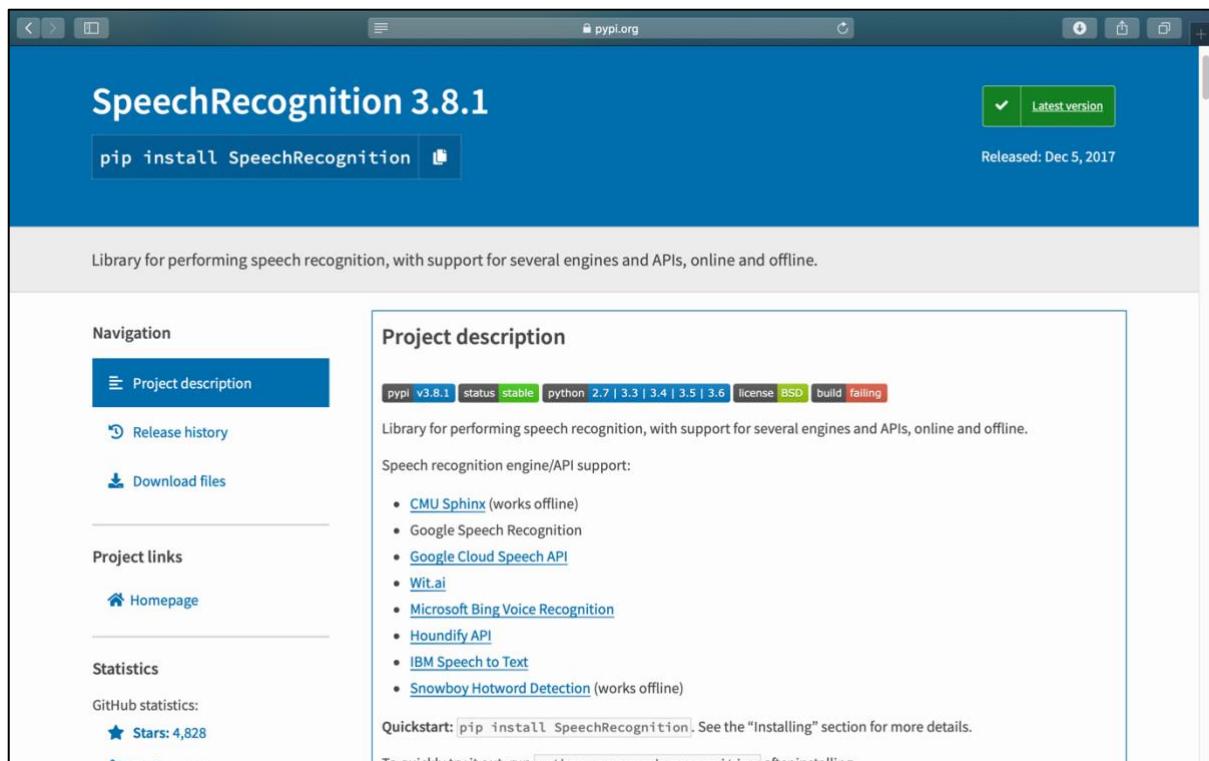
r = sr.Recognizer()

def record_audio(ask=False):
    with sr.Microphone() as source:
```

In het begin had ik ook heel erg last met het runnen van deze code, de grootste reden, de dependecies. Ik heb deze applicatie uiteindelijk werkend gekregen door al mijn nieuwe kennis toe te passen. Het grootste probleem was de 1^e dependencie uit de lijst,

Import speech_recognition as sr. Om dit te laten werken moet je het eerst installeren met de volgende command in je terminal: pip install speechrecognition.

Eigenlijk heel erg simpel, maar je moet wel zelf nog uit gaan zoeken bij welke virtuele omgeving dit bijvoorbeeld wel en niet werkt en met welke versie van Python. Meerdere van deze dependencies vereisen weer iets andere versies bijvoorbeeld, het is heel goed opletten wat wel en niet werkt in welke combinaties. Uiteindelijk kwam ik erachter hoe ik effectief mijn fouten kon gaan oplossen.



Bijvoorbeeld zoals bij deze; SpeechRecognition. Je ziet bij Python staan: 2.7|3.3|3.4|3.5|3.6. Python is momenteel al bij 3.8.3. Deze SpeechRecognition package is dus nog niet up-to-date. Bovenaan bij de omschrijving staat het volgende al aangegeven:

‘Library for performing speech recognition, with support for several engines and APIs, online and offline.’

Het gaat voor ons vooral over de Google Speech Recognition. De hele herkenning gaat uiteindelijk over de API van Google. Dat is de reden dat alles nu veel meer accuraat is als voorheen.

Met deze package/library heb je ook meteen een hele hoop opties, zo kan je meteen gebruik maken van CMU Sphinx, zodat het offline werkt bijvoorbeeld.

Het is dus heel erg handig dat door 1 package te installeren, je toegang hebt tot een hele hoop functies. Waarschijnlijk is dat dus ook een reden dat het nog niet een update heeft gehad tot de huidige Python versie. Al deze functies moeten ook updates krijgen zodat ze werkend zijn op de nieuwere Python.

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** Eevee_speech_assistant
- File:** main.py
- Code Content:** The code defines three functions: record_audio, speak, and respond. The record_audio function records audio from a microphone, uses Google's speech-to-text API to recognize it, and returns the lowercase text. The speak function converts text to speech using gTTS and plays it back. The respond function checks if the input contains 'hey', 'hi', or 'hello' and returns a random greeting.

```
def record_audio(ask=False):
    with sr.Microphone() as source:
        if ask:
            speak(ask)
        audio = r.listen(source)
        voice_data = ''
        try:
            voice_data = r.recognize_google(audio)
        except sr.UnknownValueError:
            speak('I did not get that')
        except sr.RequestError:
            speak('Sorry, the service is down')
    print(f">>>> {voice_data.lower()}")
    return voice_data.lower()

def speak(audio_string):
    tts = gTTS(text=audio_string, lang='en')
    r = random.randint(1,20000000)
    audio_file = 'audio' + str(r) + '.mp3'
    tts.save(audio_file)
    playsound.playsound(audio_file)
    print(f"Eevee: {audio_string}")
    os.remove(audio_file)

def respond(voice_data):
    if there_exists(['hey', 'hi', 'hello']):
        greetings = ["hey, how can I help you", person_obj.name, "hey, what's up", person_obj.name, "I'm listening"]
        greet = greetings[random.randint(0, len(greetings)-1)]
```

Bij bovenstaande screenshot ziet hoe de verbinding precies werkt, door de package PyAudio (dit was eerst ook een probleem) maakt het mogelijk om de microfoon van mijn laptop te gebruiken in dit project. Ik roep deze dus aan met de command: `with Microphone() as source`.

- PyAudio 0.2.11+ (required only if you need to use microphone input, [Microphone](#))

Bovenstaande tekst staat tussen het rijtje met Requirements, zonder PyAudio zou dit hele project al niet werken, en dat is iets waar ik dus eerst ook last van had.

Onder het stuk van de microfoon gaat het over gTTS. Dit is de hele dependencie dat zorgt dat er gepraat wordt. Door het script wordt er al bepaald wat er gezegd gaat worden, hier word vervolgens een mp3 van gemaakt, dit kan je zien bij: `audio_file = 'audio' + str(r) + 'mp3'` Daaronder staat: `tts.save(audio_file)` hier word dus zoals verwacht, het bestand opgeslagen.

Vervolgens word het door de dependencie playsound afgespeeld en daarna weer verwijderd doormiddel van: os.remove(audio_file). Dus elke keer dat ik iets vraag en antwoord krijg wordt er een nieuw file aangemaakt, afgespeeld en vervolgens weer verwijderd.

Ik vind dit dus al iets dat erg mooi is aan de kracht van Python. Dit moet vast ook wel kunnen in Javascript en andere talen, maar het werkt gewoon top op deze manier. Ook kan ik nu dus Python aansturen om bepaalde taken te vervullen zoals het openen van een zoekcriteria in Google.

```

    speak(f'I'm very well, thanks for asking {person_obj.name}')

if there_exists(["what's the time", "tell me the time", "what time is it"]):
    time = ctime().split(" ")[3].split(":")[0:2]
    if time[0] == "00":
        hours = '12'
    else:
        hours = time[0]
    minutes = time[1]
    time = f'{hours} {minutes}'
    speak(time)

if there_exists(["search for"]) and 'youtube' not in voice_data:
    search_term = voice_data.split("for")[-1]
    url = f"https://google.com/search?q={search_term}"
    webbrowser.get().open(url)
    speak(f'Here is what I found for {search_term} on google')

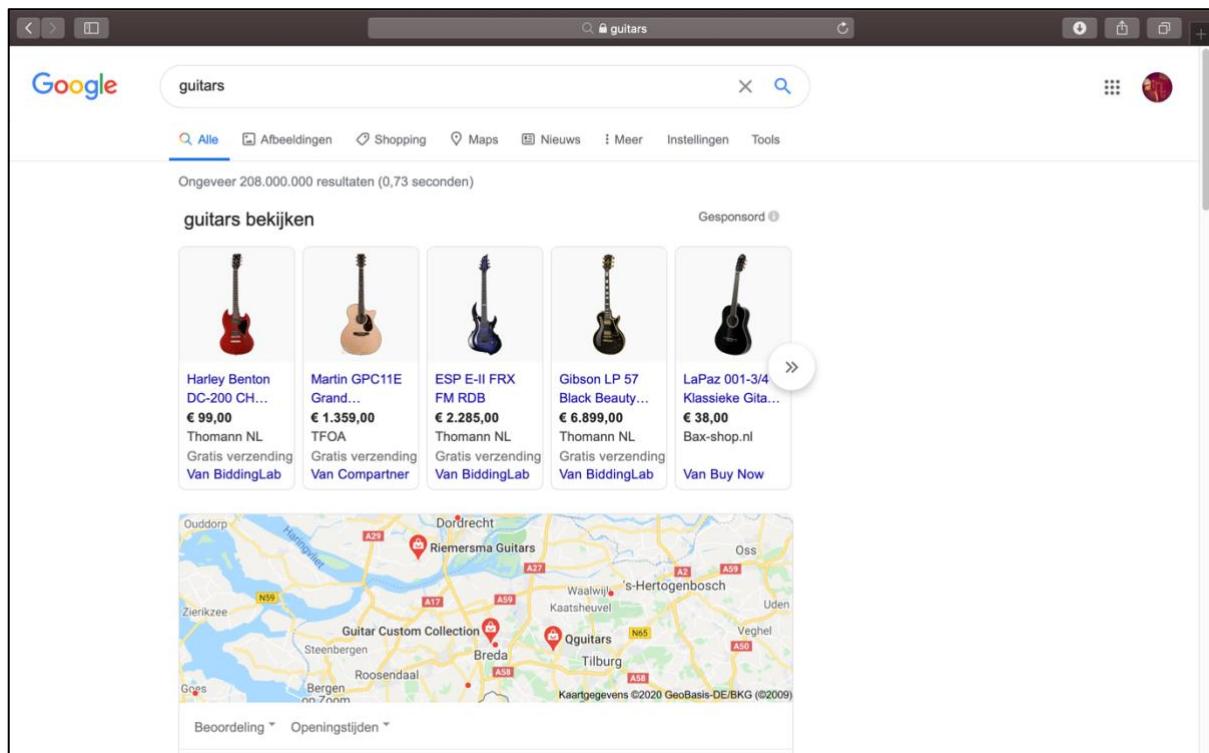
if there_exists(["youtube for"]):
    search_term = voice_data.split("for")[-1]
    url = f"https://www.youtube.com/results?search_query={search_term}"
    webbrowser.get().open(url)
    speak(f'Here is what I found for {search_term} on youtube')

if there_exists(["price of"]):
    search_term = voice_data.lower().split(" of ")[-1].strip()
    stocks = {
        "apple": "AAPL",
        "microsoft": "MSFT",
        "facebook": "FB"
    }

```

Bij bovenstaand screenshot ziet u hoe ik de commands heb gemaakt. Het zijn, net zoals bij het vorige project, een hoop if/else statements. Het ziet er hier in Python wel iets anders uit als bij de Javascript variant.

Het interessante hier vind ik zijn bijvoorbeeld de Google searches. Ik zeg bijvoorbeeld, search for guitars. Vervolgens herkent de applicatie ‘search for’ en haalt deze twee woorden weg en plaatst de rest van de zin als zoekopdracht in de browser.



```

113     stock = yf.Ticker(stock)
114     price = stock.info["regularMarketPrice"]
115
116     speak(f'price of {search_term} is {price} {stock.info["currency"]} {person_obj.name}')
117 except:
118     speak('oops, something went wrong')
119
120 if 'find location' in voice_data:
121     location = record_audio('What is the location?')
122     url = 'https://google.nl/maps/place/' + location + '/'
123     webbrowser.get().open(url)
124     speak('Here is the location for ' + location)
125
126 if there_exists(['capture', 'my screen', 'screenshot']):
127     myScreenshot = pyautogui.screenshot()
128     myScreenshot.save('/Users/maxwelder/Documents/school/Labs/screen.png')
129     speak('Successfully saved your screenshot')
130
131 if there_exists(['thanks', 'thank you']):
132     speak("You are welcome " + person_obj.name)
133
134 if there_exists(['exit', 'quit', 'goodbye']):
135     speak("going offline")
136     exit()
137
138 time.sleep(1)
139
140 person_obj = person()
141 while True:
142     voice_data = record_audio()
143     respond(voice_data)
144
145     respond() if 'find location' in voice_data

```

Bij bovenstaande screenshot zien nog een paar commands die ik bedacht had, ik kan deze applicatie zo groot maken als ik wil. Doordat deze applicatie doormiddel van de Speech Recognition import gebruik maakt van een Google API is het heel erg accuraat.

Ik heb ook snel een test filmpje gemaakt, deze is te zien op:

Resultaten:

Ik wilde dus graag in Python gaan werken met dit vlak, proberen interacties te krijgen met een ‘AI’. In mijn opzicht is dit nu gelukt, ik kan hier hele gesprekken mee voeren als ik wil en ik kan vragen of de ‘AI’ taken gaat uitvoeren voor mij, dit is nu dus volkomen mogelijk.

Ik liep eerst tegen talloze errors aan bij het openen van deze applicatie, dit kwam uiteindelijk door niet goed door te hebben wat de verschillen zijn in de versies van bijvoorbeeld Python. De tekst editor maakte uiteindelijk niet mega veel uit, maar zoals PyCharm is veel duidelijker als het gaat om een Interpreter (Python versie) kiezen gaat bijvoorbeeld.

Het is heel erg naar dat Python naar 3.8 is gegaan en daardoor de meeste dependencies/libraries/packages niet werken. Overal waar ik zoek naar informatie vind ik, dat het voor python 3 is. Maar als je net een versie te vroeg of te ver zit, werkt een of meerdere van deze dependencies niet meer.

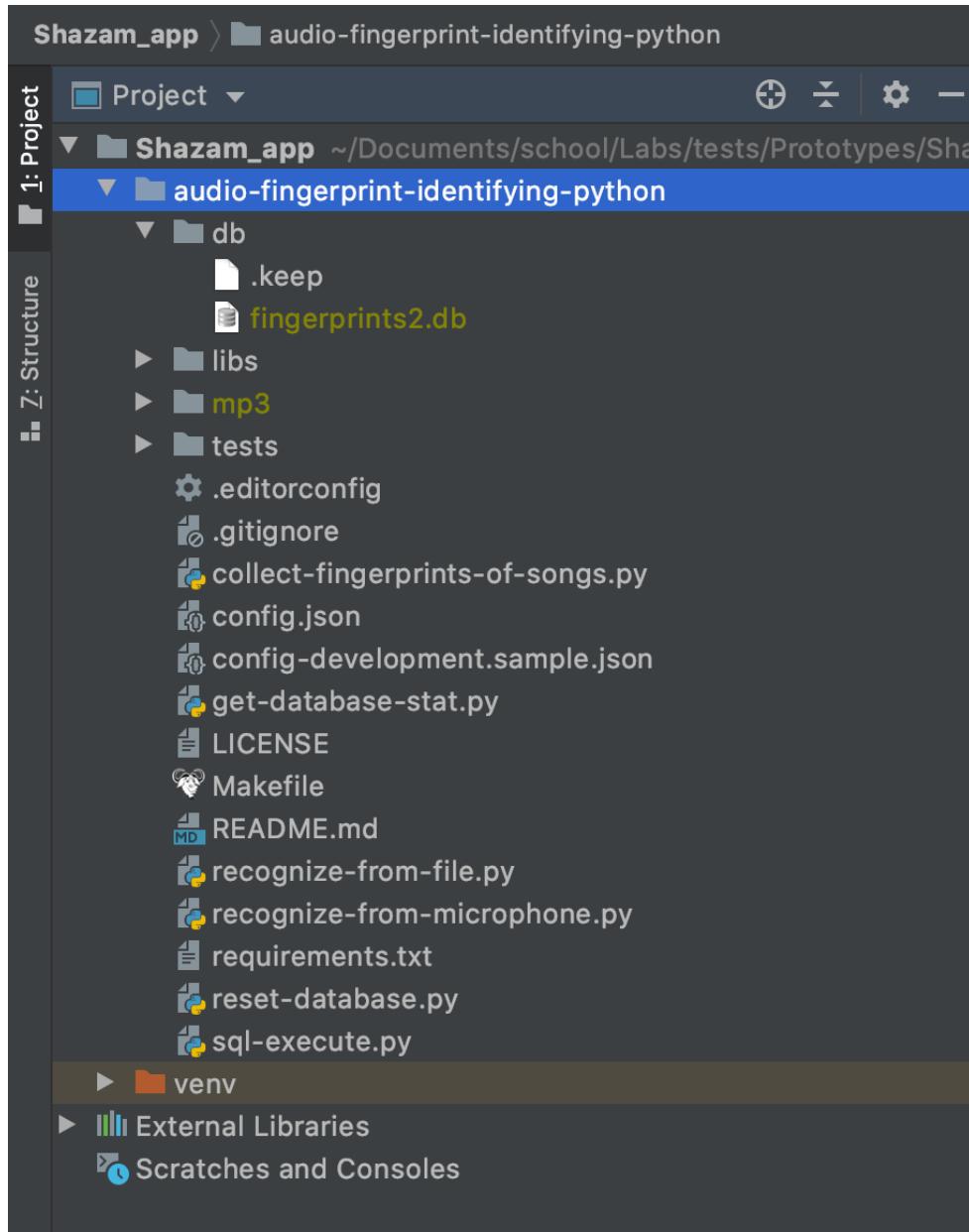
Ik ben voor mezelf nu eigenlijk tot de conclusie gekomen dat ik erg veel heb geleerd over Speech Recognition en hoe ik hiermee kan interacteren in mijn eigen software. Ondanks dat dit nog niet echt een ‘AI’ is, is dit inzicht iets dat zeker van toepassing is op AI.

Ik ben hierdoor anders naar de interactie met deze software gaan kijken. Ik kan nu veel meer vertellen en doen aan de interacties die plaatsvinden als iemand gaat interacteren met een AI. Ik hoop door deze informatie en kennis in te zetten, ik in een later stadium een AI kan maken waar ik mee zou kunnen jammen. Het doel daarvan is natuurlijk om een muzikant te helpen bij het creatieve proces, maar ik ben door mijn onderzoek erachter gekomen dat het de ervaring verbeterd door er mee te kunnen interacteren.

Shazam Applicatie

Na blijven proberen was het me dan toch gelukt. Ik heb mijn eigen Shazam applicatie werkend!

Dit project was een project van Github dat ik echt beslist niet werkend kreeg. Ik had eerst ook een Windows variant gevonden, dus ik ging het proberen op een Windows laptop, maar dit lukte ook niet. Maanden alles laten staan en uiteindelijk met nieuwe kennis het opnieuw geprobeerd, en nu, nu lukte het wel.



Dit project werkt met een paar hoofd python bestanden.

- Get-database-stat.py
- Reset-database.py
- Collect-fingerprinst-of-songs.py
- Recognize-from-microphone.py

Als ik Get-database-stat.py run, dan krijg ik te zien welke nummers er momenteel in de database zitten.

```
(venv) (base) MacBook-Pro-van-Max-5:audio-fingerprint-identifying-python maxwalder$ python get-database-stat.py
sqlite - connection opened

* total: 13 song(s) (867132 fingerprint(s))
** id=13 02 Snow ((Hey Oh)).mp3: 295318 hashes
** id=11 05 Suck My Kiss.mp3: 199644 hashes
** id=9 09 Basket Case.mp3: 138787 hashes
** id=8 unleashed Powers.mp3: 96845 hashes
** id=5 Dammit.mp3: 93857 hashes
** id=3 GDAMC.mp3: 12341 hashes
** id=12 GDAMC_solo.mp3: 11739 hashes
** id=10 Suck_My_Kiss_acoustic.mp3: 8421 hashes
** id=6 A_minor_scale.mp3: 3353 hashes
** id=2 G_minor_scale.mp3: 2751 hashes
** id=4 testo2.mp3: 2331 hashes
** id=7 testo.mp3: 1709 hashes
** id=1 A.mp3: 36 hashes

* duplications: 0 song(s)

* colissions: 848347 hash(es)

done
sqlite - connection has been closed
★ (venv) (base) MacBook-Pro-van-Max-5:audio-fingerprint-identifying-python maxwalder$
```

Wat er gebeurd is, je ziet dat er 13 nummers op het moment van de screenshot in de database zitten. De hashes zijn eigenlijk de fingerprints die zijn aangemaakt voor elk nummer. Op het moment dat ik de app aan ga zetten, dan worden de fingerprints van de nieuwe recording vergeleken met de fingerprints in deze database. Als het helemaal overeenkomt, dan heb je een match.

De remove-databse.py file doet precies wat je denkt dat het doet, het leegt de database.

Collect-fingerprints-from-songs.py file zorgt dat er een algoritme over alle nummers gaat in de database en er worden hashes gecreëerd.

```

python_itspoma
Project Terminal: Local
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208850.
(venv) (base) MacBook-Pro-van-Max-5:audio-fingerprint-identifying-python maxwalder$ python collect-fingerprints-of-songs.py
sqlite - connection opened
* id=1 channels=2: A.mp3
  already exists (36 hashes), skip
* id=2 channels=2: G_minor_scale.mp3
  already exists (2751 hashes), skip
* id=3 channels=2: GDAmC.mp3
  already exists (12341 hashes), skip
* id=4 channels=2: testo2.mp3
  already exists (2331 hashes), skip
* id=10 channels=2: Dammit.mp3
  new song, going to analyze..
  fingerprinting channel 1/2
/Users/maxwalder/Documents/school/Labs/tests/python_itspoma/audio-fingerprint-identifying-python/libs/fingerprint.py:91: RuntimeWarning: divide by zero
encountered in log10
arr2D = 10 * np.log10(arr2D) # calculates the base 10 logarithm for all elements of arr2D
local_maxima: 3899 of frequency & time pairs
finished channel 1/2, got 54481 hashes
fingerprinting channel 2/2
local_maxima: 3977 of frequency & time pairs
finished channel 2/2, got 55573 hashes
storing 93857 hashes in db
* id=5 channels=2: A_minor_scale.mp3
  already exists (3353 hashes), skip
* id=6 channels=2: testo.mp3
  already exists (1709 hashes), skip
* id=7 channels=2: unleashed Powers.mp3
  already exists (96845 hashes), skip
[ 0: Git  1: Python Console  2: Terminal  3: TODO]
Plugin Error: Plugin "Requirements" is incompatible (until build 193.SNAPSHOT < PC-201.7223.92). (6 minutes ago)
Event Log
37:1 LF UTF-8 EditorConfig Python 2.7 (python_itspoma) master

```

Bij bovenstaand plaatje zie je in het rood dat deze nummers worden geskipt, dat komt omdat deze al in de database zitten. In het groen zie je het proces voor een nieuwe song.

Dit duur van dit proces hangt ervan af hoeveel nummers je in de database wilt plaatsen en hoelang de nummers zijn. Hoe langer het nummer, hoe meer hashes.

Hierna ben je eigenlijk klaar, je kan nu beginnen met het herkennen van muziek!

Ik laat dit starten door de command: `python recognize-from-microphone -s 5`.

De `-s` staat voor seonden, en daarna de 5. Dit betekent dus luister voor 5 seonden. Dit kan zo lang als je wilt, maar eigenlijk is 5 seonden precies genoeg om een accuraat resultaat te krijgen. Als je minder lang doet is het niet accuraat.

Tijdens het luisteren ziet het er zo uit. Hoe meer hashes het vind, hoe meer hashtags je ziet staan.

```

sqlite - connection opened
* started recording..
 00510 #
 00613 #
 00633 #
 00566 #
 00299
 00354 #
 00423 #
 00327
 00051
 00268
 00033
 00281
 00296
 00680 ##
 00498 #
 00263
 00272
 00486 #
 00422 #
 00295
 00292
 00645 #
 00340 #
 00030
 00230
 00447 #
 00557 #
 00677 ##
 00391 #

```

Uiteindelijk na de 5 seconden is dit wat je ziet (mits het herkend is).

```
* recording has been stopped
* recorded 204800 samples
  fingerprinting channel 1/2
  local_maxima: 65 of frequency & time pairs
  ** found 57 hash matches (step 805/805)
  finished channel 1/2, got 57 hashes
  fingerprinting channel 2/2
  local_maxima: 66 of frequency & time pairs
  ** found 57 hash matches (step 819/819)
  finished channel 2/2, got 114 hashes

  ** totally found 114 hash matches
=> song: 09 Basket Case.mp3 (id=9)
    offset: 91 (4 secs)
    confidence: 42
```

Je ziet nu dus dat het werkte, het nummer is gevonden.



Als ik gitaarmuziek opneem, en dit vervolgens in mijn database plaats, dan werkt het ook echt als ik er met mijn gitaar voor ga zitten.

```

python_itspoma) audio-fingerprint-identifying-python > collect-fingerprints-of-songs.py
Project Terminal: Local
00196
00132
00117
01105 #####
02050 ######
01369 #####
00877 ##
00767 ##
00523 #
00876 ##
* recording has been stopped
* recorded 409600 samples
fingerprinting channel 1/2
local_maxima: 171 of frequency & time pairs
** found 84 hash matches (step 1000/2259)
** found 84 hash matches (step 1000/2259)
** found 27 hash matches (step 259/2259)
finished channel 1/2, got 195 hashes
fingerprinting channel 2/2
local_maxima: 171 of frequency & time pairs
** found 84 hash matches (step 1000/2259)
** found 84 hash matches (step 1000/2259)
** found 27 hash matches (step 259/2259)
finished channel 2/2, got 390 hashes

** totally found 390 hash matches
=> song: GDAmC.mp3 (id=3)
  offset: 286 (13 secs)
  confidence: 18
SQLite - connection has been closed
(venv) (base) MacBook-Pro-van-Max-5:audio-fingerprint-identifying-python maxwalder$
```

Bij bovenstaande screenshot had ik een akkoord progressie ingespeeld en dit werd succesvol herkent.

Ik heb voor dit prototype een video gemaakt en op YouTube geplaatst, deze is te vinden op: <https://youtu.be/wLkwy5Moibc>

Resultaten:

Het eerste wat ik wil noemen is het feit dat ik het werkend kreeg. Het was dan wel niet precies met de output die ik wilde, maar dit werkt, en het werkt goed.

Het is voor mij als muzikant ook erg interessant om te zien dat ik ermee kan interacteren met mijn gitaar. Voor nu zit de database gevuld met wat liedjes en mijn eigen geluiden, deze kleine database kan worden uitgebreid tot zo groot als maar kan.

Ik vind het ook een mooi feit dat het verschil in key kan horen. Ik had een G-mineur toonladder gespeeld en in A-mineur. Dit is exact hetzelfde riedeltje, het enige verschil is dat de A 1 noot hoger is. Dit wist de applicatie keer op keer goed te herkennen.

Door deze tests uit te voeren weet ik dat het goed mogelijk is om een AI te maken waar je kan jammen met je favoriete artiest. Als ik de database simpelweg vul met geluiden van die artiest, dan zou het al moeten kunnen werken.

Voor nu is er de tijd niet meer voor om de output te veranderen om er daadwerkelijk mee te gaan jammen. Maar door mijn ervaring die ik afgelopen periode heb opgedaan kan ik met zekerheid zeggen dat het mogelijk is.

Reflectie:

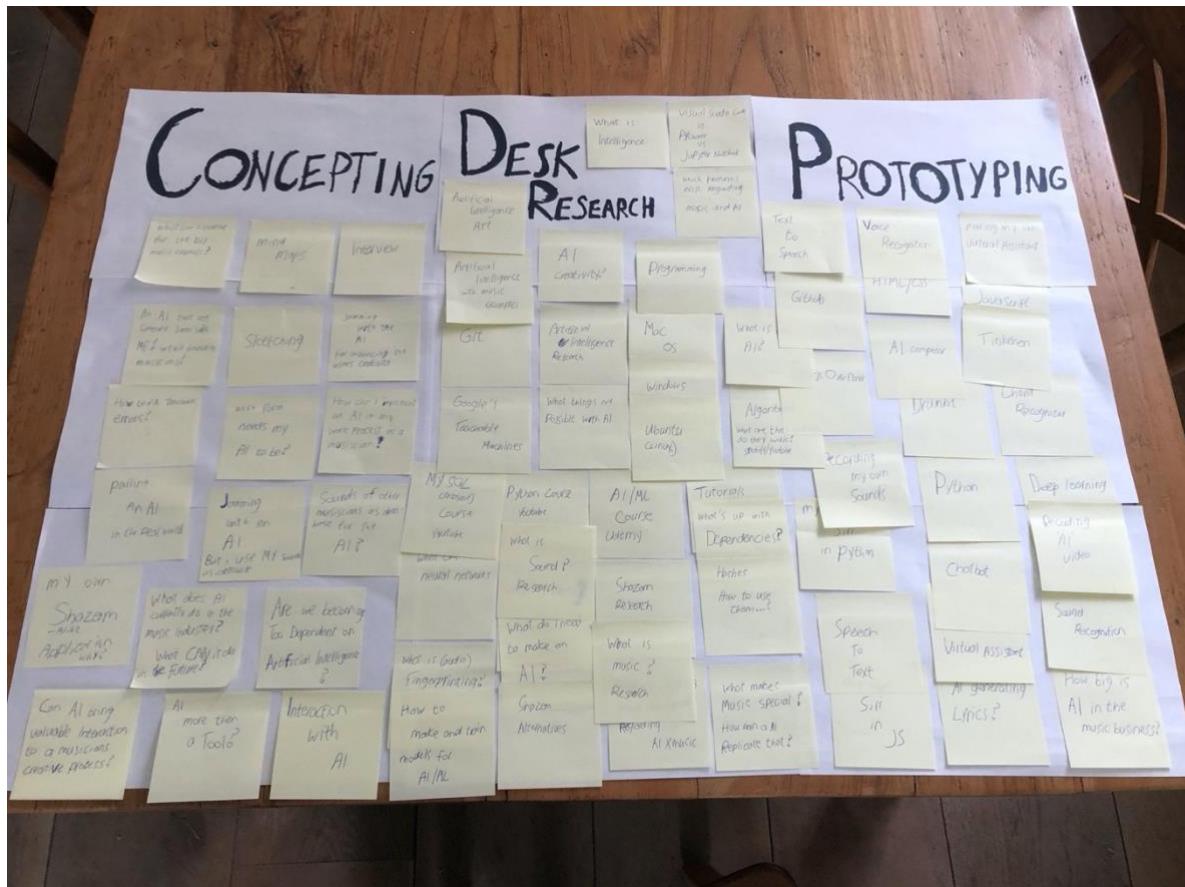
Ik heb heel erg veel geleerd uit dit half jaar. Ik weet nu bijvoorbeeld hoe een AI en Shazam werkt, waar AI goed voor is, waar AI niet goed voor, Python geleerd, inzicht gekregen in de verschillen tussen Max OS, Windows, Ubuntu en ik snap hoe je een virtual assistant maakt bijvoorbeeld.

Ik heb voor mijn eigen gevoel veel goede stappen gezet en ben goed te werk gegaan. Ik vond alleen dat door de corona situatie mijn grip op het hele lab weg was. Het is heel raar om alles thuis te moeten doen, normaal was ik gewoon bijna elke dag naar school gegaan, want dan kan ik me veel beter concentreren. Zelfde geldt voor de lessen, ik heb er weinig van opgestoken, op deze manier werkte het voor mij gewoon niet goed. Ik snap dat iedereen het beste heeft geprobeerd van te maken, maar voor mij werkte dit allemaal niet ideaal.

Ik ben erg trots dat ik mijn prototypes af heb gekregen want dat is iets waar ik zelf als programmeur gewoon erg veel waarde aan hechtte.

Ik vind ook dat mijn resultaten een goede basis zijn voor meer. Ik denk echt dat door mijn twee hoofd prototypes te combineren, ik een hele gave applicatie zou kunnen maken waar je echt mee zou kunnen jammen. Als dat af zou zijn, zou ik op nog andere en betere conclusies uit kunnen komen.

Ik kijk nu positief terug op het afgelopen halfjaar. Ondanks dat heel erg veel tegen zat, heb ik er veel van geleerd en heb ik een nieuwe kijk op de topics waar ik nu onderzoek naar heb gedaan.



Bronnen:

<https://www.mediawijssheid.nl/kunstmatigeintelligentie/>
<https://www.mediawijssheid.nl/filterbubbel/>
<https://www.salesforce.com/nl/products/einstein/ai-deep-dive/#>
<https://www.graydon.nl/blog/zo-past-u-kunstmatige-intelligentie-toe-uw-bedrijf>
https://www.youtube.com/watch?v=_uQrJ0TkZlc&t=16238s
www.google.nl
www.wikipedia.nl
<https://www.udemy.com/course/data-science-and-machine-learning-with-python-hands-on/>
<https://www.youtube.com/watch?v=FWOZmmIUqHg>
www.landr.com
www.aiva.ai
<https://github.com/itspoma/audio-fingerprint-identifying-python>
<http://coding-geek.com/how-shazam-works/>
<https://github.com/worldveil/dejavu>
<https://willdrevo.com/fingerprinting-and-audio-recognition-with-python/>
<https://www.toptal.com/algorithms/shazam-it-music-processing-fingerprinting-and-recognition>
www.stackoverflow.com
<https://youtu.be/8YjzwhauXDRKBQ>
<https://soundcloud.com/user-583891932/sets/aiva>
<https://github.com/yoyonel/audio-fingerprint-identifying-python>
https://www.tutorialspoint.com/artificial_intelligence_with_python/artificial_intelligence_with_python_primer_concepts.htm
<https://mindmajix.com/neural-network-in-artificial-intelligence>
https://www.tutorialspoint.com/artificial_intelligence_with_python/getting_started.htm
<https://youtu.be/C6EwysRVqUQ>
https://en.wikipedia.org/wiki/Société_des_auteurs,_compositeurs_et_éditeurs_de_musique
<https://en.m.wikipedia.org/wiki/Shazam>
<http://coding-geek.com/how-shazam-works/>
<https://pypi.org/project/SpeechRecognition/>
https://youtu.be/MbSrdd_fIw
<https://youtu.be/wLkwy5Moibc>
<https://github.com/maxiicano/Prototypes>