

Machine Learning Pipeline Automation

Amar Kisoensingh

1 juni 2021

Voorwoord

Samenvatting

Summary

Afkortingen

API application programming interface 6, 24, 25

PaaS Platform as a Service 14, 15

Lijst van figuren

1.1	Organogram van NGTI op 29-03-2021 [3].	11
1.2	Screenshot van de Swiss Climate Challenge app [5].	12
1.3	Screenshot van de My Swisscom App [7]	12
4.1	Machine learning in de context van andere domeinen	21
4.2	Lifecycle van een model volgens Hapke en Nelson [13, p. 4].	22
4.3	Gamma hyperparameter van een SVC algoritme met waarde 0.1	25
4.4	Gamma hyperparameter van een SVG algoritme met waarde 100	25
4.5	Model uitgerold met behulp van een API.	26
4.6	Vorm van expliciete feedback gebruikt door YouTube.	27
4.7	Machine learning pipeline met een algoritme exploratie tussenstap.	28
4.8	Machine learning pipeline waarbij stappen geautomatiseerd kunnen worden voor het gemak van developers.	29

Lijst van tabellen

3.1	Onderzoeksmethode deelvraag 1	18
3.2	Onderzoeksmethode deelvraag 2	18
3.3	Onderzoeksmethode deelvraag 3	18
3.4	Onderzoeksmethode hoofdvraag	19
5.1	Knock-out criteria voor frameworks dat cloud computing platformen beheerd. . .	31
5.2	Criteria voor frameworks dat cloud computing platformen beheerd.	31
1	Scope deelvraag 1	42
2	Scope deelvraag 2	43
3	Scope deelvraag 3	43
4	Scope hoofdvraag	44

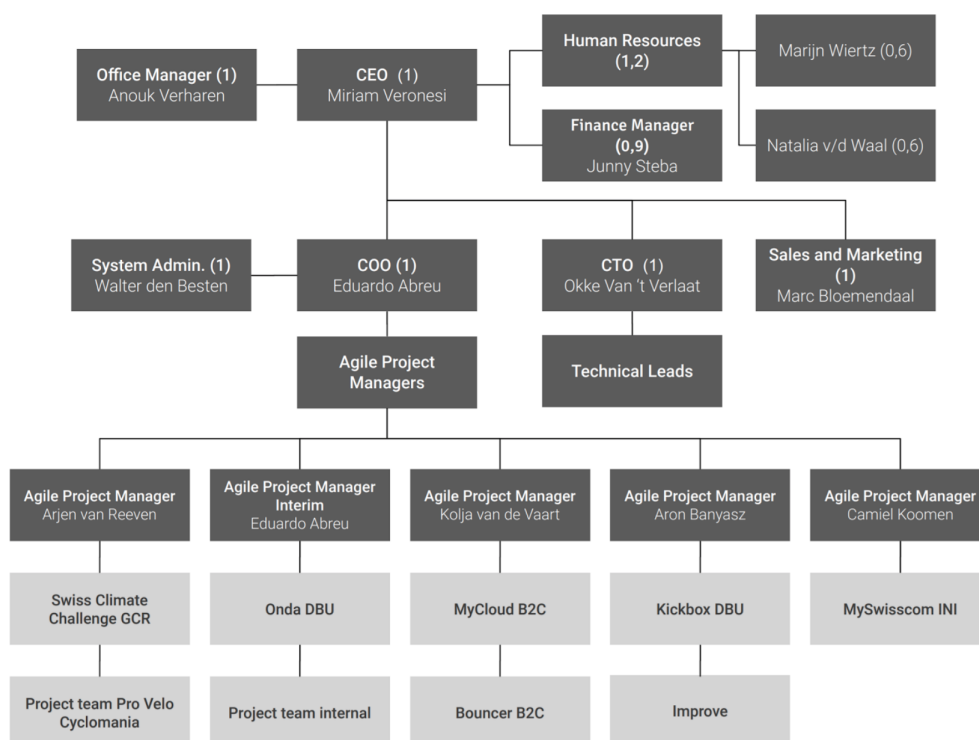
Inhoudsopgave

1	Inleiding	10
1.1	Projecten van NGTI	11
1.2	Tools die worden gebruikt	12
1.3	Aanleiding opdracht	13
1.4	Leeswijzer	13
2	Probleemanalyse	14
2.1	Obstakels voor NGTI	15
2.2	Vooronderzoek naar bestaande oplossingen	16
2.3	Doelstelling	16
2.4	Hoofd- en deelvragen	16
3	Onderzoeksmethoden en scope	17
4	Stappen in een machine learning pipeline	20
4.1	Wat is machine learning?	21
4.2	De stappen in een machine learning pipeline	22
4.3	Conclusie	27
4.4	Advies	27
5	Frameworks dat platformen beheert	30
5.1	Wat is een Infrastructure as Code?	31
5.2	Frameworks vergeleken met elkaar	31
5.3	Experiment met een framework	31
5.4	Conclusie	31
5.5	Advies	31
6	Architecturale ontwerp van de oplossing	32
6.1	Het C4 model	33
6.2	Sequence diagrams	33
7	Proof of concept	34
7.1	Scope definiëren	35
7.2	Mock up van de proof of concept	35
7.3	Vorbereiding van de proof of concept	35
7.4	Conclusie	35
7.5	Advies	35
8	Discussie	36
8.1	37
9	Reflectie	38
	Bibliografie	39

1 Inleiding

NGTI is een software ontwikkelbedrijf dat gevestigd is in Rotterdam. Opgericht in 2012 maakt NGTI applicaties (apps) voor mobiel en/of webgebruik. Naast het ontwikkelen werkt NGTI aan het hele traject om een app heen, namelijk de probleemstelling, mockups, wireframes en prototyping. Daarnaast levert NGTI ook support en lost bugs op nadat een app live is gegaan [1]. Ook maakt NGTI white label apps en frameworks [2].

Het bedrijf heeft, zoals de meeste bedrijven, een organogram (Figuur 1.1). In de praktijk is dit echter niet terug te vinden en wordt de structuur gezien als "plat". Collega's kunnen elkaar laagdrempelig benaderen waardoor niemand een onbekende is en verschillende disciplines makkelijk met elkaar samen kunnen werken.



Figuur 1.1: Organogram van NGTI op 29-03-2021 [3].

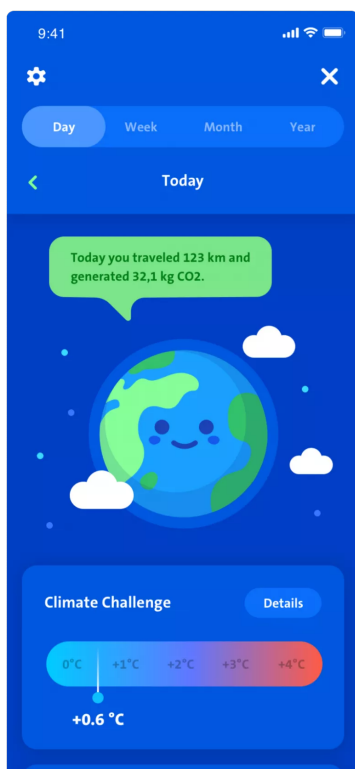
NGTI is een dochterbedrijf van Swisscom [4]. Sinds maart 2021 is het bekend gemaakt dat Swisscom van plan is om een afdeling, Swisscom DevOps Center, te fuseren met NGTI. Omdat de fusie onzekerheid met zich meebrengt voor de structuur en manier hoe NGTI werkt, zal de situatie vóór de fusie aangehouden worden gedurende het afstuderen.

1.1 Projecten van NGTI

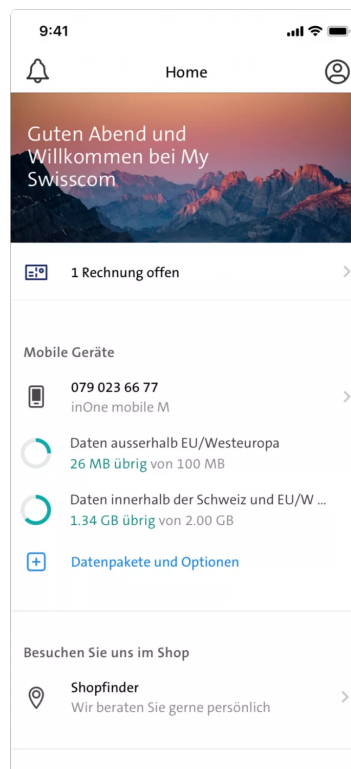
NGTI heeft een vrij breed portfolio met apps voor verschillende doeleinden. Een van deze apps is de Climate Challenge App [5]. Met deze app kunnen gebruikers hun CO₂-voetafdruk en impact in kaart brengen. Er wordt bijgehouden hoeveel kilometer de gebruiker reist en met welk vervoersmiddel. De app is onderdeel van twee bestaande nieuwsapps, Blick en Bluewin [6]. Het

doel is om de gebruiker aan te sporen om groener te reizen. Een screenshot van de app is te zien in Figuur 1.2.

Een andere oplossing is de My Swisscom App [7]. Dit is een native app voor Android en iOS waarbij Swisscom-klanten hun contract kunnen bestellen, wijzigen of beëindigen. In de app kunnen klanten ook de dataverbruik zien en instellingen voor abonnementen wijzigen. Een screenshot van de app is te zien in Figuur 1.3.



Figuur 1.2: Screenshot van de Swiss Climate Challenge app [5].



Figuur 1.3: Screenshot van de My Swisscom App [7]

1.2 Tools die worden gebruikt

Om productief te zijn, gebruikt NGTI een aantal tools en programma's om producten te maken en te communiceren met zowel collega's als klanten. De meest gebruikte en belangrijkste zijn Slack, Google Workspace, Zoom en Microsoft Teams.

1.2.1 Slack

Interne communicatie gaat via Slack. Het programma faciliteert collega's om elkaar met een lage instap te benaderen en berichten die voor het hele bedrijf relevant zijn te versturen. Ook zijn er 'channels' beschikbaar over specifieke onderwerpen, zoals: *#dev*, *#ios* en *#test-automation*.

1.2.2 Google Workspace

Met Google Workspace kunnen bestanden en documenten gemaakt, opgeslagen en gedeeld worden. Dit is mogelijk via een browser waardoor werknemers geen software hoeven te installeren. NGTI gebruikt het ook om collaboratief en parallel te werken aan hetzelfde document.

1.2.3 Microsoft Teams en Zoom

Voorheen werd Zoom alleen gebruikt om te videobellen met collega's en geïnterviewden. In de tijd van de pandemie is Zoom echter een belangrijke speler geworden om effectief samen te werken. Meetings zoals introducties van nieuwe collega's of demo's van producten worden online gehouden.

1.3 Aanleiding opdracht

NGTI wilt de gebruikerservaring van haar apps verbeteren en een voorsprong hebben op haar concurrenten. Dit kan NGTI op een aantal manieren doen waarvan apps "slimmer" maken er een van is. Het slimmer maken houdt voor NGTI in dat de app bijvoorbeeld beter kan anticiperen wat de gebruiker wilt en nodig heeft op een gegeven moment. Dit kan onder andere door het toepassen van machine learning (ML). De opdracht kan verdeelt worden in twee onderdelen:

- Onderzoek naar hoe een pipeline opgezet kan worden op een cloud computing platform door middel van een framework
- Onderzoek naar het maken van een platform-agnostische oplossing

Daarnaast zal een proof-of-concept (PoC) gemaakt worden om aan te tonen of het haalbaar is in de praktijk. Een diepere duik in het probleem en het definiëren van de onderdelen is te vinden in hoofdstuk 2.

1.4 Leeswijzer

TODO: Schrijf leeswijzer voor elk hoofdstuk

2 Probleemanalyse

Zoals beschreven in paragraaf 1.3 wilt NGTI ML toepassen om haar apps slimmer te maken. Hier zijn een aantal redenen voor, onder andere om de gebruikerservaring te verbeteren en om een voorsprong te hebben op concurrenten.

Het 'slimmer' maken van applicaties kan op verschillende manieren, maar met machine-learning kan een platform gebouwd worden waarmee elke richting op gegaan kan worden. Om machine-learning te implementeren in haar applicaties loopt NGTI tegen een aantal obstakels aan, namelijk: expertise vereist in het ML domein, benodigde tijd om een pipeline op te zetten en vendor lock-in.

2.1 Obstakels voor NGTI

NGTI loopt tegen de voorgenoemde obstakels aan omdat NGTI weinig ervaring heeft met ML. In samenwerking met een extern bedrijf worden modellen getraind die vervolgens gebruikt worden in apps van NGTI. Om zelf modellen te trainen heeft NGTI kennis over ML, ML pipelines en tijd nodig. Bovendien wilt NGTI niet bij één cloud computing platform haar infrastructuur opzetten maar gemakkelijk kunnen schakelen tussen platformen. De obstakels worden in de volgende koppen verduidelijkt.

2.1.1 Expertise Machine Learning

ML is ingewikkeld en diepgaand onderwerp. Om een model te trainen is kennis nodig van verschillende domeinen: data mining, software engineering en statistieken. Doordat er voorkennis nodig is om een model te trainen en een pipeline goed op te zetten, is het vaak te drempelig voor developers om een start te maken met ML. De expertise is daarnaast niet in een korte tijd te vergaren.

2.1.2 Opzetten pipeline

Bovenop de complexiteit van ML zelf bestaan er verschillende manieren om een model te trainen. Het opzetten van ML pipelines is daar een van. Een pipeline is een workflow dat bestaat uit een aantal stappen die doorgelopen worden om een model te trainen. In elke stap worden acties uitgevoerd, zoals het verwijderen van onbruikbare data of de prestatie van modellen vergelijken en een rapport met uitslagen genereren. Het opzetten van zo een pipeline én de actie(s) in de stappen definiëren kost tijd en vereist specifieke kennis. Daarnaast zijn de stappen en acties vaak hetzelfde voor verschillende pipelines. Het automatiseren en hergebruiken van stappen en acties tussen pipelines zou tot onder andere tijdswinst en het verminderen van herhaling van code kunnen leiden.

2.1.3 Vendor lock-in

Er bestaan een aantal diensten, zogenoemde Platform as a Service (PaaS), waarbij je een pipeline kan opzetten en acties kan definiëren. Een van de problemen met een PaaS is vendor lock-in. Dit betekent dat, als er eenmaal een pipeline is opgezet, de overdraagbaarheid van de pipeline naar een andere PaaS vrijwel onmogelijk is. Ook zijn de opties en mogelijkheden om uit te breiden in de toekomst gelimiteerd.

2.2 Vooronderzoek naar bestaande oplossingen

Het gebruik van ML pipelines is geen nieuwe techniek en is mogelijk bij PaaS en bedrijven die zich specialiseren in ML pipelines. Het gemakt met zulke services is dat de gebruiker gelijk kan beginnen met het trainen van ML modellen en zich niet zorgen hoeft te maken over infrastructurele details. Echter is er sprake van vendor lock-in en kunnen services van deze bedrijven niet gebruikt worden.

Gedurende de vooronderzoek zijn "Infrastructure as Code" frameworks naar voren gekomen. Dit zijn frameworks waarmee, middels code, een infrastructure binnen een Platform as a Service (PaaS) opgezet kan worden. Dit kan gebruikt worden als onderdeel van de PoC en wordt in een van de deelvragen verder onderzocht.

2.3 Doelstelling

Om haar doel, de gebruikerservaring van apps verbeteren en een voorsprong hebben op concurrent, te bereiken is NGTI van plan ML pipelines te gebruiken om ML toe te passen. De gewenste oplossing is een systeem waarbij developers met weinig tot geen kennis een model kunnen trainen. Het systeem moet de infrastructurele taken voor zich nemen, zoals het opzetten van een pipeline en de stappen en acties automatiseren. Daarnaast moet een ML pipeline pijnloos doorlopen kunnen worden op verschillende platformen zodat het systeem platform-agnostisch is.

2.4 Hoofd- en deelvragen

Uitgaand van de drie obstakels kan de hoofdvraag als volgt worden geformuleerd:

In welke mate kan een machine learning pipeline worden geautomatiseerd onafhankelijk van het onderliggende cloud computing platform?

De hoofdvraag kan worden onderbouwd met vier deelvragen. Om te beginnen is het verstandig om te weten welke stappen er in een machine learning pipeline zit:

Waar bestaat een machine learning pipeline uit?

Daarnaast is een framework nodig dat, door middel van code, verschillende cloud computing platformen kan beheren:

Hoe kan een framework verschillende cloud computing platformen beheren om een machine learning pipeline op te zetten?

Ten slotte wordt een PoC gemaakt om te laten zien of het probleem oplosbaar is. Hiervoor is een doordachte voorbereiden onmisbaar:

Hoe ziet de architecturale blauwdruk van een applicatie, waarmee een platform-onafhankelijk machine learning pipeline opgezet kan worden, eruit?

3 Onderzoeksmethoden en scope

Om elke hoofd- en deelvraag te beantwoorden, wordt er bij elk gebruik gemaakt van een onderzoeksmethode. Volgens Scribbr [8] zijn er twee onderzoeksmethoden: kwantitatief en kwalitatief. Bij een kwantitatief onderzoeksmethode wordt data verzameld waarmee bijvoorbeeld grafieken of tabellen gemaakt kunnen worden. De focus bij een kwalitatief onderzoeksmethode ligt bij het verzamelen van verschillende interpretaties en opvattingen. Hierop kan optioneel een eigen interpretaties op gemaakt worden. [9].

Onder kwantitatief en kwalitatief vallen verschillende dataverzamelingsmethoden. Deze beschrijft simpelweg de manier hoe data wordt verzameld. Dit kan bijvoorbeeld met een enquête, literatuuronderzoek op websites en in boeken of een onderzoek over een lange periode [9].

Elke hoofd- en deelvraag is gekoppeld aan een onderzoeksmethoden. Vervolgens is beschreven welk(e) dataverzamelingsmethode(n) wordt gebruikt met een korte toelichting. Daarnaast wordt op een hoog niveau de scope bepaald.

D1: Waar bestaat een machine learning pipeline uit?	
Methode(s)	Kwalitatief
Dataverzamelings methode(n)	Literatuuronderzoek, fundamenteel onderzoek, toegepast onderzoek
Scope	Bijlage 9

Tabel 3.1: Onderzoeksmethode deelvraag 1

D2: Hoe kan een framework verschillende cloud computing platformen beheren om een machine learning pipeline op te zetten?	
Methode	Kwalitatief
Dataverzamelings methode(n)	Literatuuronderzoek, vergelijkend onderzoek
Scope	Tabel 9

Tabel 3.2: Onderzoeksmethode deelvraag 2

D3: Hoe ziet de architecturale blauwdruk van een applicatie, waarmee een platform-onafhankelijk machine learning pipeline opgezet kan worden, eruit?	
Methode	Kwalitatief
Dataverzamelings methode(n)	Literatuuronderzoek
Scope	Tabel 9

Tabel 3.3: Onderzoeksmethode deelvraag 3

H: In welke mate kan een machine learning pipeline worden geautomatiseerd onafhankelijk van de onderliggende cloud computing platform?	
Methode	Kwalitatief
Dataverzamelings methode(n)	Literatuuronderzoek
Scope	Tabel 9

Tabel 3.4: Onderzoeksmethode hoofdvraag

4 Stappen in een machine learning pipeline

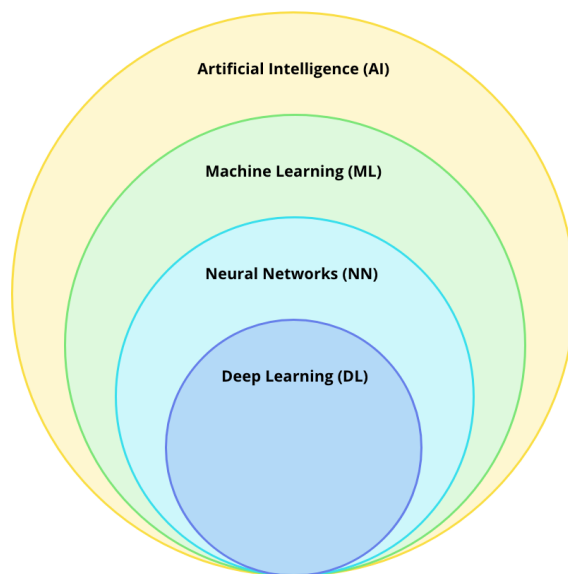
Een ML pipeline is zoals beknopt beschreven in deelparagraaf 2.1.2 een collectie van stappen dat wordt doorlopen om een model te trainen. Elke stap bevat een aantal acties dat wordt uitgevoerd, zoals onbruikbare data weghalen of de prestatie analyseren. De stappen en acties worden in paragraaf 4.2 uitgelegd. Een van de stappen in een pipeline is het trainen van het model. Om een idee te krijgen van ML en modellen trainen wordt dit kort uitgelegd.

4.1 Wat is machine learning?

ML houdt in dat een computer een taak kan uitvoeren zonder ervoor expliciet geprogrammeerd te zijn. Dit wordt gedaan door een ML model te laten leren van een gegeven dataset. Vervolgens kan er een voorspelling worden gemaakt [10, p. 1-3].

De domeinen deep learning (DL), neural networks (NN) en artificial intelligence (AI) komen vaak voor als het over ML gaat. Zoals weergegeven in Figuur 4.1 is te zien dat ML een subset is van AI, NN een subset van ML en als laatste DL dat een subset is van NN.

Bij AI wordt niet alleen ML toegepast, maar ook concepten zoals beredeneren, plannen, vooruitdenken, onthouden en terug refereren. Een voorbeeld hiervan is dat een ML model kan voorspellen wat het volgende woord in een zin kan zijn, maar een AI kan beredeneren waarom de zin gebouwd is zoals het is en hoe het binnen de context van de alinea past [11].



Figuur 4.1: Machine learning in de context van andere domeinen

Een NN bestaat uit een collectie van nodes dat gemodelleerd is naar de hersenen. Een NN heeft minimaal 3 lagen: een input laag, een verborgen laag en een output laag. Elke laag bevat neuronen dat data als input kan krijgen en data als output aan de volgende laag meegeeft. DL is een NN dat meerdere verborgen lagen bevat [12].

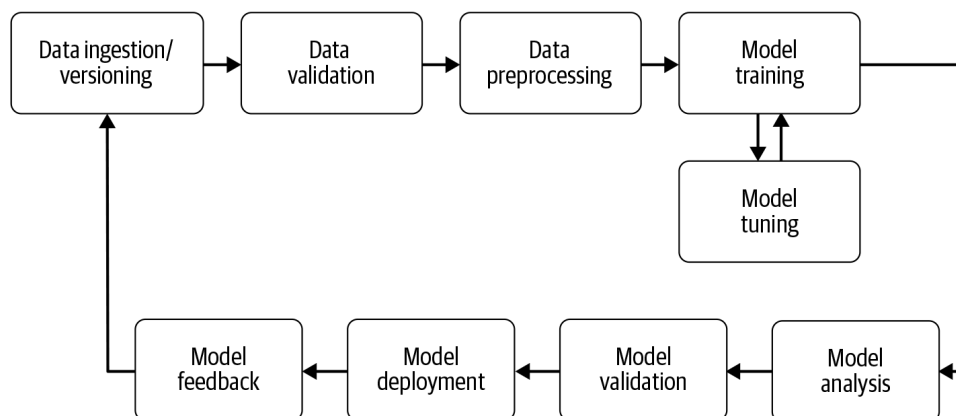
In het ML domein bestaan talloze algoritmes om voorspellingen te maken. In de [BIJLAGE LINK] is een mind-map te vinden van de algoritmes die gedurende de scriptie naar voren zijn gekomen. Het grotendeels kan gegroepeerd worden in vier stijlen: supervised, unsupervised,

semi-supervised en reinforcement learning. In de [BIJLAGE LINK] is meer informatie over de stijlen te vinden.

Het trainen van een model is een proces waarbij vooraf data wordt opgeschoond en achteraf de prestatie van het model wordt gevalideerd. Normaliter wordt dit gedaan door specifieke code te schrijven voor deze taken. Een valkuil is dat de code niet bij elke developer werkt en schaalbaar niet in alle gevallen naar een productie omgeving. Een manier om dit wel te behalen is om te werken met ML pipelines. Zoals kort uitgelegd in deelparagraaf 2.1.2 bestaat een ML pipeline uit stappen en acties. Er is echter geen consensus binnen het ML domein over wat de juiste stappen en acties in een pipeline zijn. Voor de scriptie is gekozen voor een pipeline van Hapke en Nelson uit het boek "Building Machine Learning Pipelines". Het boek is gemaakt en gepubliceerd door O'Reilly; een bekend en gecrediteerd bedrijf dat boeken maakt binnen het software engineering domein.

4.2 De stappen in een machine learning pipeline

Een ML pipeline begint met het opnemen van data en eindigt met het ontvangen van feedback om de prestatie van het model te verbeteren. De pipeline bevat een aantal stappen zoals data voorbereiden, het model trainen en het uitrollen van het model (Figuur 4.2).



Figuur 4.2: Lifecycle van een model volgens Hapke en Nelson [13, p. 4].

In totaal zijn er, zonder de feedback loop stap, acht stappen dat elke keer doorlopen moeten worden om een model te trainen. In de volgende subkoppen zullen de stappen worden doorlopen met een korte uitleg over wat er gebeurt in een stap.

4.2.1 Stap 1: Data opname en versiebeheer (Data ingestion/versioning)

De eerste stap in de pipeline is het opnemen van data. Met deze data zal het model getraind, gevalideerd en getest worden. De dataset kan van een of meerdere bronnen komen, zoals lokaal, een online opslag locatie of van een database. Zodra de data is ingeladen, moet het verdeeld worden tussen een train, validatie en test dataset. Normaal gebeurt dit met een split ratio van 6:2:2. De train dataset is 60% en de validatie en test datasets zijn allebei 20% van de originele dataset [13, p. 27-37].

Een use case van een pipeline is dat een nieuw model getraind kan worden door een geüpdatet dataset te gebruiken. Dit wordt gedaan door de voorgaande dataset te gebruiken waarbij nieuwe data is toegevoegd. Door het gebruik van verschillende datasets is het verstandig om versiebeheer toe te passen. Zo is goed te zien welk dataset welk model produceert. Een versie geven aan een dataset gebeurt voordat de dataset wordt ingeladen [13, p. 39-40]. Versiebeheer voor datasets kan bijvoorbeeld met DVC [14] of Pachyderm [15].

4.2.2 Stap 2: Data validatie (Data validation)

Nu de dataset verdeeld is, een versie heeft en op een bereikbare plek is, kan de data gevalideerd worden. Deze stap is vooral belangrijk om te voorkomen dat een model wordt getraind dat niet nuttig is aangezien het trainen veel tijd in beslag kan nemen. Een bekende uitdrukking is "garbage in = garbage out". Dit betekent dat als de dataset niet goed is, het model ook niet goed zal presteren [13, p. 43]. Tijdens de validatie stap wordt gecontroleerd op het volgende:

- Afwijkingen in de dataset
- Wijzigingen in de structuur
- Algemene statistieken in vergelijkingen met voorgaand datasets [13, p. 44]

Bij het controleren van afwijkingen in de dataset wordt gekeken naar waarden die opmerkelijk zijn. Afwijkende waarden liggen te ver van het gemiddelde en kan een verkeerd beeld schetsen bij het trainen van het model. Deze uitschieters kunnen simpelweg uit de dataset gefilterd worden.

Het kan voorkomen dat bij een nieuw dataset de type van waarden zijn gewijzigd. Een *int* kan bijvoorbeeld veranderd zijn in een *string* of *boolean*. Er is dan sprake van een wijziging in de structuur van de dataset. Dit is problematisch omdat er een vertaaltap gemaakt moet worden naar iets bruikbaar. Als dit niet mogelijk is moeten de waarden uitgefilterd worden wat de prestatie van het model negatief kan beïnvloeden.

De algemene statistieken is een hulpmiddel om te controleren op afwijkingen en wijzigingen. Vaak kan de controle in een oogopslag gedaan worden.

4.2.3 Stap 3: Data voorbereiden (Data preprocessing)

Het voorbereiden van de dataset is een stap dat de prestatie van het model verbetert en het proces van het trainen versnelt. Deze stap kan verdeeld worden in twee sub-stappen: het opschonen en het optimaliseren van de dataset.

Bij het opschonen worden bijvoorbeeld duplicaten of waarden die onbruikbaar zijn uit de dataset weggehaald. Onbruikbare waarden zijn waarden die simpelweg niet kloppen of verkeerd zijn ingevoerd. Hierbij kan bijvoorbeeld gedacht worden aan een medewerker die de interactietijd met een klant moet bijhouden, maar is vergeten om de eindtijd te noteren. Voor het algoritme zal het dan lijken alsof de medewerker een klant tot sluitingstijd heeft geholpen.

Datasets kunnen geoptimaliseerd worden om twee redenen: algoritmes werken sneller met waarden die dicht bij 0 liggen en algoritmes kunnen niet met elke waarde in de dataset omgaan. Om de efficiëntie van het algoritme te verbeteren kunnen een aantal technieken gebruikt worden:

- Schalen

Bij het schalen van data worden de waarden getransformeerd waarden dat ligt tussen een schaal, zoals tussen 0 en 100 of 0 en 1. Bij het schalen wordt het **bereik** getransformeerd [16].

- Normaliseren

Als een dataset wordt gestandaardiseerd, worden de waarden getransformeerd in een standaard normale verdeling waarbij het gemiddelde 0 en de afwijking 1 is. Hierbij wordt de **vorm** van de dataset getransformeerd [16]. Het normaliseren is belangrijk als het algoritme waarden vergelijkt met verschillende eenheden [**feature-scaling-standardization**]. In sommige gevallen is normalisering een vereiste bij een aantal algoritmes [17].

- Categorisch codering

Een groot aantal algoritmes kan alleen omgaan met numerieke waarden. Het kan voorkomen data datasets categorische waarden bevatten. Dit zijn waarden dat een label voorstellen, zoals *first*, *second* of *third*. Om deze waarden te gebruiken moeten ze worden gecodeerd als een numerieke waarde. De label kan als de waarde 1 of 2 gecodeerd worden. Deze techniek heet integer encoding en kan gebruikt worden als de volgorde van de codering klopt. Een andere techniek om waarden te coderen heet hot encoding. Deze techniek wordt gebruikt waarbij de labels geen relatie met elkaar hebben en maakt gebruik van meerdere waarden om een label te beschrijven:

$$\begin{bmatrix} \text{Animal} & \text{C1} & \text{C2} & \text{C3} \\ \text{cat} & 1 & 0 & 0 \\ \text{dog} & 0 & 1 & 0 \\ \text{mouse} & 0 & 0 & 1 \end{bmatrix}$$

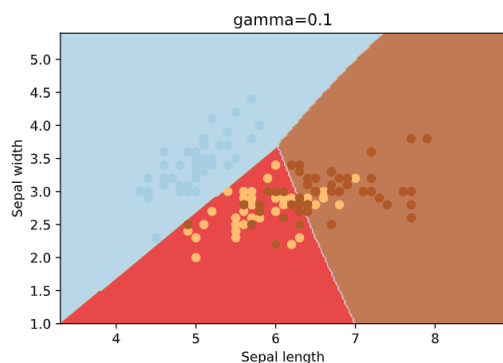
Dit is een goed moment om het getransformeerde dataset op te slaan als "tussen-dataset" zodat deze stap niet herhaalt hoeft te worden. Het voorbereiden van grote datasets kan een grote hoeveelheid tijd in beslag nemen.

4.2.4 Stap 4 en 5: Model trainen en tunen (Model training and Model tuning)

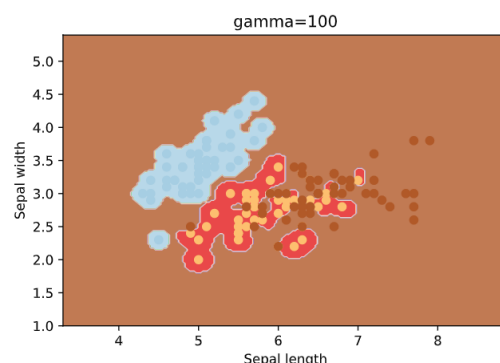
Na het valideren en voorbereiden van de dataset kan het model getraind worden. Het trainen gaat door middel van een ML algoritme. Een algoritme is vergelijkbaar met een conventionele algoritme in software engineering zoals bijvoorbeeld binary search, merge sort en depth first search. Het algoritme slaat na het trainen met de dataset regels, nummers en algoritme specifieke data structuren op in de vorm van een model. Het algoritme kan worden gezien als een specifiek programma waarmee, gegeven een input, voorspellingen mee gedaan kan worden [18].

Elk algoritme heeft variabelen om er voor te zorgen dat het algoritme beter aansluit op de dataset. Deze variabelen heten *hyperparameters*. Het algoritme heeft standaard hyperparameters die niet altijd de optimale prestatie behalen. Op een heuristische wijze kunnen de optimale hyperparameters gevonden worden [19]. Dit proces heet *tunen*.

In Figuur 4.3 en Figuur 4.4 is te zien hoe de *gamma* hyperparameter van een SVC algoritme invloed heeft op het model. Dit algoritme classificeert op basis van gegeven waarden. De waarden worden geplotted en zal binnen een van de drie kleuren vallen. Het model met als hyperparameter waarde 100 zal vaker de classificatie met de bruine kleur als voorspelling geven dan als de waarde 0.1 zou zijn.



Figuur 4.3: Gamma hyperparameter van een SVC algoritme met waarde 0.1



Figuur 4.4: Gamma hyperparameter van een SVG algoritme met waarde 100

4.2.5 Stap 6 en 7: Model analyse en validatie (Model analysis and Model validation)

Na het trainen en tunen kan analyse en validatie plaats vinden. Bij het analyseren wordt de prestatie van het model vergeleken met voorgaande modellen. Om dit te doen kan, net als in deelparagraaf 4.2.2, statistieken gegenereerd worden van het model. De soort statistiek hangt af van wat voor soort algoritme is gebruikt. Een classificatie algoritme heeft bijvoorbeeld andere statistieken dan een regressie algoritme. Op basis van de uitkomst kan de dataset of hyperparameters aangepast worden om de accuraatheid te verhogen.

Met de validatie van een model moet er gekeken worden met een genuanceerd perspectief. Validatie gaat namelijk over hoe eerlijk een model is als er gekeken wordt naar een bepaalde groep. Een groep kan gedefinieerd worden als geslacht, etniciteit, locatie of leeftijd. Het kan namelijk voorkomen dat de dataset voornamelijk bestaat uit bijvoorbeeld vrouwen. Dit *kan* ervoor zorgen dat het model minder accurate voorspellingen maakt voor mannen [10, p. 109-110]. Een voorbeeld waar het gebruik van ML grote negatieve gevolgen had was de toeslagenaffaire in 2020. De Nederlandse overheid maakte gebruik van een systeem dat aangaf of een burger mogelijk bijstandsfraude had gepleegd. Het systeem maakte gebruik van ML om fraude aan te geven. Jaren na het gebruik bleek dat er sprake was van etnische profilering. De dataset bestond voornamelijk uit immigranten uit Islamitische landen. Het systeem heeft hierdoor een *bias* [20].

4.2.6 Stap 8: Model uitrollen (Model deployment)

Na het analyseren en valideren kan het model uitgerold worden. Het uitrollen kan op twee manieren: inbakken in een app of serveren met een application programming interface (API).

Met het inbakken in een app wordt bedoeld dat het model meegenomen wordt als de app gebouwd

wordt. Voordelen van deze methode is dat het model offline bruikbaar is en de rekenkracht van het apparaat gebruikt kan worden om te voorspellen of classificeren. Een nadeel is dat het model niet makkelijk geüpdatet kan worden.

Een model serveren door middel van een API. De app dat het model wilt gebruiken zal een request moeten doen met input. De API geeft de input door aan het model. Het model geeft een predictie terug dat de API doorstuurt naar de applicatie. Het updaten van het model gaat gemakkelijker dan het model inbakken met een app. Een nadeel is de eis dat de app een internetverbinding moet hebben om het model te gebruiken [10, p. 130].

In Figuur 4.5 is te zien hoe een model geserveerd wordt via een API. Een endpoint genaamd "*classify*" is gedefinieerd waarop een *POST* request gedaan kan worden. De endpoint haalt de input uit de request en geeft het door aan het model. De model maakt vervolgens een predictie dat uiteindelijk naar de app teruggestuurd wordt.

```
1 @app.route("/classify", methods=["POST"])
2 def classify():
3     # Get input from request
4     json = request.get_json()
5     data = json["data"]
6
7     # Get prediction from model
8     classification = model.predict([data])
9
10    # Return prediction to app
11    return jsonify({ "family": classification[0] })
```

Figuur 4.5: Model uitgerold met behulp van een API.

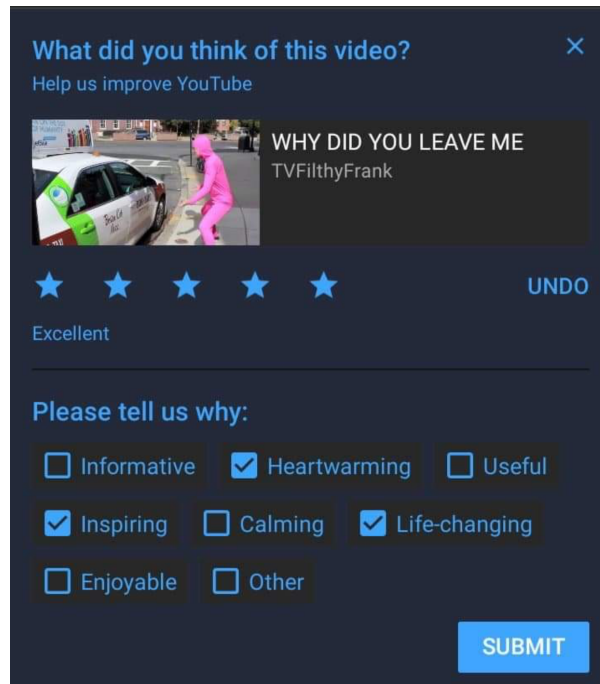
4.2.7 Stap 9: Feedback loop (Model feedback)

Om het model continue te verbeteren kan feedback verzameld worden. De feedback geeft een beeld van de effectiviteit van het model in een productie omgeving. Feedback kan verdeeld worden in twee soorten: impliciet en expliciet [10, p. 264].

Het verzamelen van impliciete feedback wordt gedaan zonder dat de gebruiker zich daar bewust van is. Dit wordt gedaan door bij te houden of een voorspelling geschikt is. Een voorbeeld van deze methode is het bijhouden of een gebruiker een video kijkt dat door het algoritme voorgesteld is.

In tegenstelling tot impliciet wordt bij expliciete feedback gevraagd aan de gebruiker of een voorspelling gepast is. Een voorbeeld van deze vorm in de praktijk is hoe YouTube expliciete feedback vraagt (Figuur 4.6). Naast het aangeven of de voorspelling gepast was, kan de gebruiker ook aangeven waarom de video gepast was.

Het verzamelen van feedback is niet noodzakelijk maar helpt met het verbeteren van het model. Deze stap gaat samen met een nieuwe dataset naar de eerste stap in de lifecycle (Figuur 4.2).



What did you think of this video?

Help us improve YouTube

WHY DID YOU LEAVE ME
TVFilthyFrank

★ ★ ★ ★ ★ UNDO

Excellent

Please tell us why:

☐ Informative ☒ Heartwarming ☐ Useful

☒ Inspiring ☐ Calming ☒ Life-changing

☐ Enjoyable ☐ Other

SUBMIT

Figuur 4.6: Vorm van expliciete feedback gebruikt door YouTube.

Met de nieuwe dataset en de feedback kan een nieuw model getraind worden dat beter presteert dan de voorganger.

4.3 Conclusie

In dit hoofdstuk is er onderzoek gedaan naar de antwoord op de deelvraag: **D1: Waar bestaat een machine learning pipeline uit?** Het onderzoek is uitgevoerd met behulp van zowel digitale bronnen als een boek.

Een ML pipeline bestaat uit verschillende stappen die op hun beurt bestaan uit acties. De stappen zijn gericht op het waarborgen van de kwaliteit van de dataset en model, het voorbereiden van de dataset en het trainen van het model. Daarnaast kan het model uitgerold worden in een productie omgeving en kan er feedback verzameld worden over de prestatie van het model volgens gebruikers.

De acties in de stappen moeten bij het eerste gebruik van de pipeline gedefinieerd worden. De acties kan bijvoorbeeld het filteren van onbruikbare data, een model trainen of statistieken genereren zijn. Als dit vast staat, kan de pipeline hergebruikt worden om nieuwe modellen te trainen met nieuwe datasets. Op deze manier is een ML pipeline een gestructureerde en reproduceerbare proces.

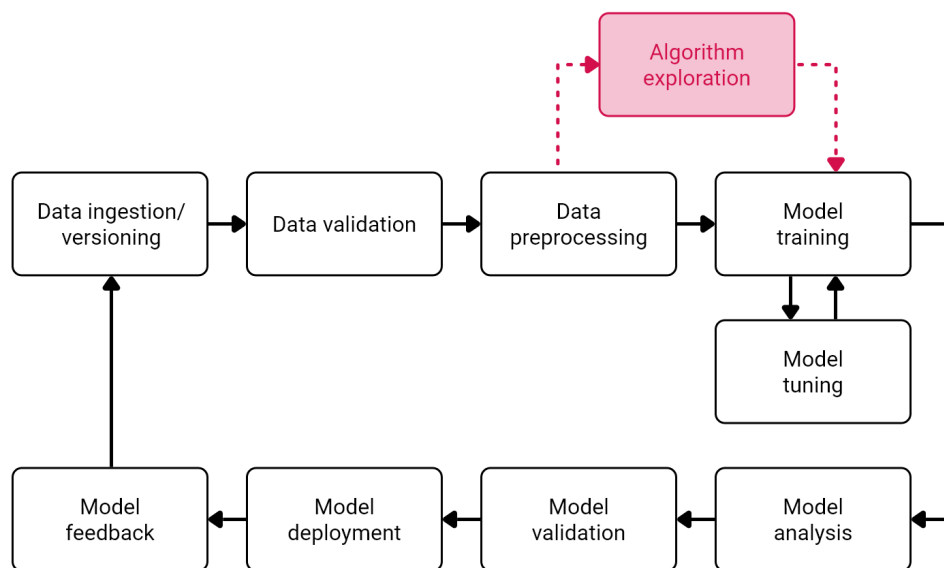
4.4 Advies

Het in gebruik nemen van een ML pipeline werkwijze is initieel ingewikkeld maar lonend op lange termijn. Het opzetten van de pipeline en de acties definiëren kost tijd en kennis. Echter als dit

eenmaal gedaan is, kan eenvoudig een verbeterd model worden getraind door een nieuwe dataset aan te leveren. Een verbeterd model is vrijwel gegarandeerd omdat stappen in de pipeline de dataset en het model analyseren en valideren.

4.4.1 Machine learning pipeline verbeteringen

Zoals eerder aangegeven in paragraaf 4.1 bestaat er niet één ML pipeline dat correct is. Een ML pipeline is een werkwijze waarbij de stappen anders geïnterpreteerd kan worden. Hierdoor is er geen regelmaat tussen verschillende pipelines. Dit is ook niet haalbaar aangezien stappen gemaakt kunnen worden dat specifiek is voor het algoritme of workflow van het bedrijf. De stappen in de pipeline van Hapke en Nelson (Figuur 4.2) is een valide basis maar kan uitgebreid worden om de ervaring van developers te verbeteren. In stap 4 en 5 (deelparagraaf 4.2.4) wordt een model getraind waarbij het beste algoritme om het ML probleem op te lossen al bekend is. De verwachting is dat de keuze/onderzoek vóór het starten van de pipeline is gedaan. Tussen stap 3 en 4 kan echter een stap tussen zitten om te experimenteren met een combinatie van verschillende algoritmes en hyperparameters. In Figuur 4.7 is te zien hoe zo een stap zou passen in de lifecycle. De stap is met een stippellijn aangegeven omdat de stap de eerste keer meegenomen kan worden als de pipeline wordt doorlopen. Na de eerste keer is de beste algoritme en hyperparameters al bekend en kan deze stap overgeslagen worden.



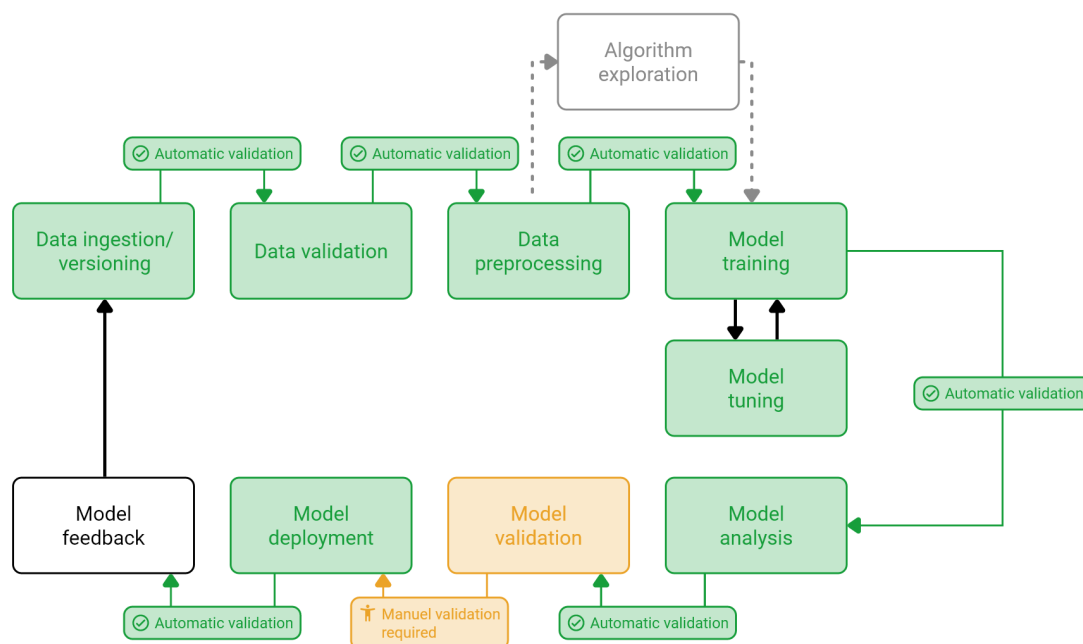
Figuur 4.7: Machine learning pipeline met een algoritme exploratie tussenstap.

In de "Algorithm exploration" stap kunnen verschillende algoritmes dat hetzelfde doel bereiken en een variatie van hyperparameters voor elk algoritme getest worden tegen het dataset. Uit deze tests kan een prestatiescore gegenereerd worden om vervolgens te kiezen voor het algoritme dat het geschiktst lijkt. Na de keuze kan de pipeline verder gaan met de volgende stappen.

4.4.2 Machine learning versimpelen

Een van de focuspunten is om te onderzoeken in hoeverre ML te vergemakkelijken is voor developers. Met de huidige pipeline is er vrij veel kennis vereist over ML om ermee te werken.

Toch kan een groot deel van de stappen geautomatiseerd worden. Tussen elke stap kan een validatie plaatsvinden met een aantal randvoorwaarden waaraan voldaan moeten worden om door te gaan naar de volgende stap. In Figuur 4.8 is te zien hoe tussen elke stap een validatie plaats vindt. Om bijvoorbeeld van stap 6 (Model analysis) naar stap 7 (Model validation) te gaan, moet het model even goed of zelfs beter presteren dan de voorganger. Hoe veel beter het model moet kunnen presteren is aan een developer. Mocht het zo zijn dat het model niet voldoet, kan de pipeline stopgezet worden.



Figuur 4.8: Machine learning pipeline waarbij stappen geautomatiseerd kunnen worden voor het gemak van developers.

Niet alle aspecten van de pipeline kunnen versimpelt worden. Een van de stappen dat niet kan is stap 7 (Model validation) in Figuur 4.8. Deze stap vereist input van een mens aangezien de bias van het model wordt beoordeelt. Wel kunnen rapporten over de bias gegenereerd worden zodat de workflow van developers gestroomlijnd wordt.

5 Frameworks dat platformen beheert

Om

5.1 Wat is een Infrastructure as Code?

5.2 Frameworks vergeleken met elkaar

Criteria	Toelichting
Programmatisch servers beheren	Het framework moet via een CLI of code een server kunnen opzetten om code uit te voeren.
Ondersteuning voor cloud computing platformen	Om platform-agnostisch te zijn moet het framework minstens twee cloud computing platformen ondersteunen waarop een ML model getraind kan worden.
Uitgebreide documentatie	

Tabel 5.1: Knock-out criteria voor frameworks dat cloud computing platformen beheerd.

Criteria	Toelichting
Ondersteuning voor lokale oplossingen	Naast het ondersteunen van cloud computing platformen moet het framework ook lokale oplossingen zoals Kubernetes of Docker ondersteunen zodat een developer lokaal het model kan trainen [https://www.npmjs.com/package/node-docker-api].

Tabel 5.2: Criteria voor frameworks dat cloud computing platformen beheerd.

5.3 Experiment met een framework

5.4 Conclusie

5.5 Advies

6 Architecturale ontwerp van de oplossing

6.1 Het C4 model

6.1.1 Level 1: Context

6.1.2 Level 2: Containers

6.1.3 Level 3: Components

6.1.4 Level 4: Code

6.2 Sequence diagrams

7 Proof of concept

7.1 Scope definiëren

7.1.1 User requirements verzamelen

7.1.2 Requirements vertalen naar een Kanban board

7.2 Mock up van de proof of concept

7.3 Voorbereiding van de proof of concept

7.3.1 Aanmaken van pipelines

7.3.2 Datasets uploaden

7.3.3 Configuratie definiëren

7.3.4 Pipeline starten

7.3.5 Model downloaden

7.4 Conclusie

7.5 Advies

8 Discussie

8.1

9 Reflectie

Bibliografie

-
- [1] 22 mrt 2021. URL: <https://www.ngti.nl/diensten/>.
 - [2] 29 mrt 2021. URL: <https://www.ngti.nl/oplossingen/>.
 - [3] NGTI B.V. „Organogram”.
 - [4] 29 mrt 2021. URL: <https://www.swisscom.ch/en/about/beteiligungen-swisscom-uebersicht.html>.
 - [5] 29 mrt 2021. URL: <https://www.ngti.nl/cases/swiss-climate-challenge/>.
 - [6] 29 mrt 2021. URL: <https://www.swissclimatechallenge.ch>.
 - [7] 29 mrt 2021. URL: <https://www.ngti.nl/cases/my-swisscom-app/>.
 - [8] 25 mrt 2021. URL: <https://www.scribbr.nl/scriptie-structuur/methodologie-in-je-scriptie/>.
 - [9] 25 mrt 2021. URL: <https://www.scribbr.nl/onderzoeksmethoden/kwalitatief-vs-kwantitatief-onderzoek/>.
 - [10] E. Alpaydin. *Introduction to Machine Learning*. 4de ed. 2020. ISBN: 9780262043793.
 - [11] 16 mrt 2021. URL: <https://machinelearningmastery.com/think-machine-learning/>.
 - [12] 16 mrt 2021. URL: <https://wiki.pathmind.com/neural-network>.
 - [13] H. Hapke en C. Nelson. *Building Machine Learning Pipelines*. 1ste ed. 2020. ISBN: 9781492053194.
 - [14] 6 apr 2021. URL: <https://dvc.org>.
 - [15] 6 apr 2021. URL: <https://www.pachyderm.com>.
 - [16] 27 mei 2021. URL: <https://www.kaggle.com/rtatman/data-cleaning-challenge-scale-and-normalize-data>.
 - [17] 27 mei 2021. URL: <https://www.kdnuggets.com/2020/04/data-transformation-standardization-normalization.html>.
 - [18] 28 mei 2021. URL: <https://machinelearningmastery.com/difference-between-algorithm-and-model-in-machine-learning>.
 - [19] 12 mrt 2021. URL: <https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/>.
 - [20] 30 mei 2021. URL: <https://fortune.com/2020/02/11/a-i-fairness-eye-on-a-i/>.

Bijlagen

Scope hoofd- en deelvragen

Scope deelvraag 1

D1: Uit welke stappen bestaat een machine learning pipeline?	
Scope	<p>Binnen de scope:</p> <ul style="list-style-type: none">• In kaart brengen uit welke stappen een pipeline bestaat• Acties die in een stap worden uitgevoerd• Of het mogelijk is om stappen te versimpelen / abstraheren voor developers• Of het mogelijk is om stappen en acties te automatiseren <p>Buiten de scope:</p> <ul style="list-style-type: none">• Automatisering van stappen en acties• Een versimpeling van machine learning
Toelichting	<p>Er wordt gekeken naar welke stappen er in een pipeline zitten. De theorie wordt vervolgens toegepast in een experiment. De nadruk ligt vooral of de mogelijkheid er is om stappen en acties te automatiseren en of machine learning versimpeld kan worden, niet dat er een uitwerking is.</p>

Tabel 1: Scope deelvraag 1

Scope deelvraag 2

D2: Hoe kan een framework verschillende cloud computing platformen beheren om een machine learning pipeline op te zetten?	
Scope	<p>Binnen de scope:</p> <ul style="list-style-type: none">• High-level uitleg over hoe een framework werkt• Inventarisatie met de "knock-out" methode• Criteria lijst voor frameworks• Opmerkelijke features die relevant zijn voor de PoC• Ervaring opdoen doormiddel van een pipeline te maken op twee cloud computing platformen <p>Buiten de scope:</p> <ul style="list-style-type: none">• Performance en snelheid
Toelichting	Frameworks dat cloud computing platformen beheert worden in kaart gebracht. Met knock-out criteria wordt de lijst verkort. Met een framework wordt een pipeline opgezet.

Tabel 2: Scope deelvraag 2

Scope deelvraag 3

D3: Hoe ziet de architecturale blauwdruk van een applicatie, waarmee een platform-onafhankelijk machine learning pipeline opgezet kan worden, eruit?	
Scope	<p>Binnen de scope:</p> <ul style="list-style-type: none">• Technische tekeningen <p>Buiten de scope: -</p>
Toelichting	De literatuuronderzoek slaat op of de technische tekeningen gemaakt zijn volgens een standaard zoals UML. Dit komt niet terug als theorie maar de bronnen worden wel vermeld.

Tabel 3: Scope deelvraag 3

Scope hoofdvraag

H: In welke mate kan een machine learning pipeline worden geautomatiseerd onafhankelijk van de onderliggende cloud computing platform?	
Scope	De scope wordt bepaald na de requirement analyse.
Toelichting	Onderzoek naar documentatie van gebruikte framework(s).

Tabel 4: Scope hoofdvraag