Machine Learning Pipeline Automation

Amar Kisoensingh

11 mei 2021

Voorwoord

Samenvatting

Summary

Lijst van figuren

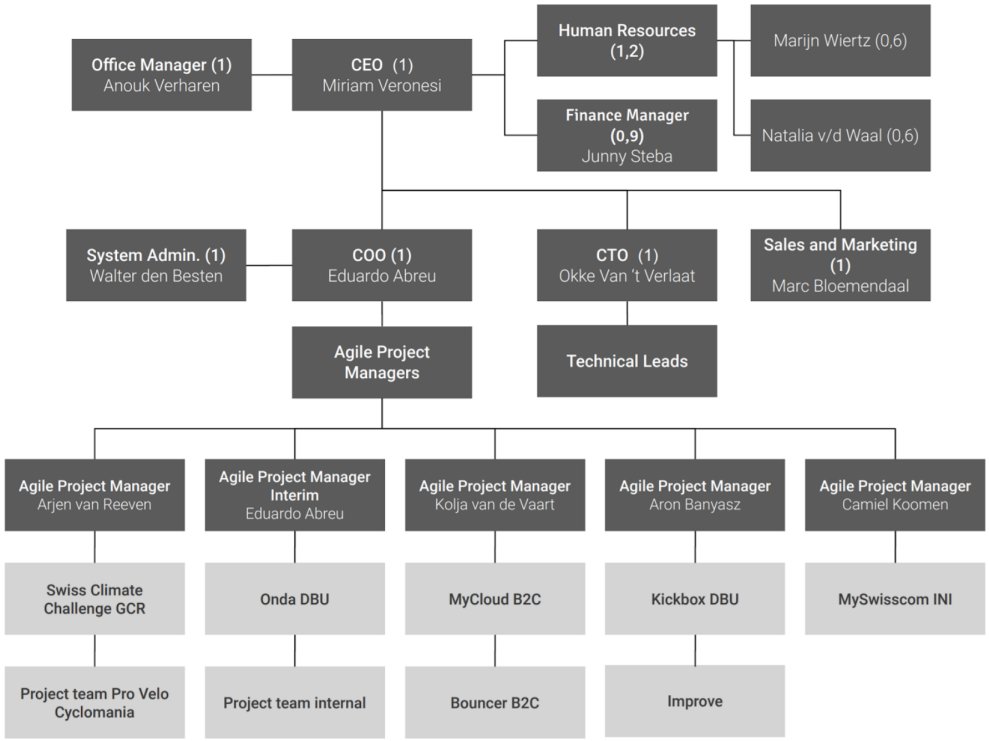
Lijst van tabellen

Inhoudsopgave

1 Inleiding

NGTI is een software ontwikkelbedrijf dat gevestigd is in Rotterdam. Met het starten van nieuwe projecten begint NGTI met de probleemstelling, mockups, wireframes en prototyping. Vervolgens wordt een applicatie (app) voor mobiel en/of webgebruik ontwikkeld en wordt support geleverd voor bijvoorbeeld updates of het oplossen van bugs [**ngti-services**]. Naast het maken van een apps op maat biedt NGTI ook andere diensten, zoals een apps framework of white label apps [**ngti-solutions**].

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 (TODO: en nog een reden).



Figuur 1.1: Organogram van NGTI op 29-03-2021 [**ngti-organogram**].

NGTI is een dochterbedrijf van Swisscom [**swisscom-other-division**]. 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.

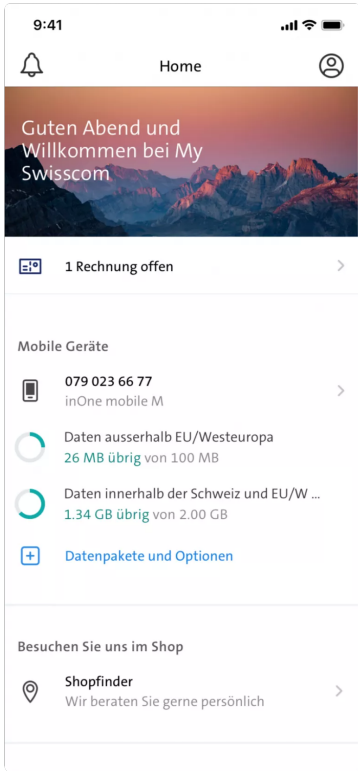
1.1 Projecten

NGTI heeft een vrij breed portfolio met apps voor verschillende doeleinden. Een van deze apps is de Climate Challenge App [**ngti-swisscom-climate-challenge**]. Met deze app kunnen gebruikers hun C02-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 [**swisscom-climate-challenge-integration**]. 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 [**ngti-my-swisscom-app**]. 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 [**ngti-swisscom-climate-challenge**].



Figuur 1.3: Screenshot van de My Swisscom App [**ngti-my-swisscom-app**]

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 verstu-

ren. 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 machine learning 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. Bij projecten waar ML wordt toegepast is in samenwerking met een bedrijf dat de modellen traint. 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.

2.1.1 Expertise Machine Learning

Machine learning is ingewikkeld onderwerp. Om een model te trainen is kennis nodig van verschillende domeinen: data mining, software engineering en statistieken. In een multi-disciplinair team is het voor één teamlid niet nodig om alle domeinen te beheersen.

Doordat er voorkennis nodig is om een model te trainen en een pipeline goed op te zetten, is het vaak te hoogdrempelig voor developers om een start te maken met machine learning. De expertise is daarnaast niet in een korte tijd te vergaren.

2.1.2 Opzetten pipeline

Bovenop de complexiteit van machine learning zelf bestaan er verschillende manieren om een model te trainen. Een machine learning pipeline opzetten is daar één van. Een pipeline is een workflow die 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 tijdwinst 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 Bestaande oplossingen om pipelines op te zetten

ML pipelines gebruiken binnen een cloud platform is geen nieuwe techniek. Dit is al enige tijd mogelijk bij platform-as-a-service (PaaS) bedrijven. Azure en Google cloud zijn PaaS bedrijven waarbij bijvoorbeeld een server of database gehuurd kan worden. Daarnaast bestaan er bedrijven dat zich specialiseren in ML pipelines en dat als enige service bieden. Het enige nadeel is dat vendor lock-in inherent is. In deelparagraaf 2.2.1 en deelparagraaf 2.2.2 wordt uitgelegd wat vendor lock-in is en waarom de service van deze bedrijven niet geschikt it.

Gedurende het vooronderzoek zijn frameworks dat cloud computing platformen zoals Azure en Google Cloud kan beheren. Het is mogelijk om programmatisch te communiceren met een platform. Dit geeft de mogelijkheid om een ML pipeline op te zetten in het gewenste systeem. De uitleg en bevindingen van deze frameworks is te lezen in deelparagraaf 2.2.3.

2.2.1 Machine learning pipeline specifieke service

Bedrijven zoals Algorithmia [**algorithmia-website**] en Valohai [**valohai-website**] bieden alleen diensten om pipelines op te zetten. Ze zorgen voor het databehoud dat door de gebruiker wordt geüpload en het trainen van het model. Verder kan er toezicht gehouden worden op de kosten, beschikbaarheid en prestatie van het model. Valohai heeft ook documentatie over hoe haar service werkt en een aantal blog posts die bij het ontwerpen van het systeem relevant zouden kunnen zijn.

2.2.2 Cloud computing platformen

De drie grote cloud computing platformen Amazon, Azure en Google hebben meer te bieden dan alleen een pipeline opzetten, zoals het hosten van een website, database of virtuele server. De cloud computing platformen hebben hetzelfde probleem als de machine learning pipeline specifieke services; vendor lock-in is onvermijdelijk. Wat wel een mogelijkheid zou kunnen zijn is dat de andere services van de cloud computing platformen gebruikt kunnen worden als onderdeel van het systeem.

Het systeem zou bijvoorbeeld een server kunnen aanmaken, een model trainen, het resultaat downloaden en vervolgens de server verwijderen. Om dit zonder de grafisch interface te doen kan er gebruik worden gemaakt van frameworks die interacteert met het platform.

2.2.3 Frameworks om cloud computing platformen te beheren

Een framework dat cloud computing platformen kan beheren is Terraform [**terraform**]. Met Terraform is het mogelijk om een plan te schrijven waarin bijvoorbeeld staat welke type server nodig is. Bij het uitvoeren van het plan spreekt Terraform een cloud computing platform naar keuze aan om de server op te starten. Terraform kan vervolgens controleren of de server draait en de server afsluiten wanneer het niet meer nodig is.

Kubeflow [**kubeflow**] is een ander framework waarmee een pipeline kan worden opgezet. Het verschil met Terraform is dat Terraform flexibeler is met wat er aangemaakt kan

worden op een cloud computing platform. Kubeflow kan alleen een pipeline uitrollen. Een andere framework zoals Kubeflow is Apache Beam [**apache-beam**].

2.3 Doelstelling

Om haar doel, de gebruikerservaring 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, de stappen en acties automatiseren. Daarnaast moet het systeem ook platform-agnostisch zijn zodat het systeem niet aan één platform gebonden 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 [**research-methods**] 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. [**quantitative-vs-qualitative**].

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 [**quantitative-vs-qualitative**].

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 11

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 11

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 11

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
Dataverzamelingsmethode(n)	Literatuuronderzoek
Scope	Tabel 11

Tabel 3.4: Onderzoeksmethode hoofdvraag

4 Stappen in een Machine Learning pipeline

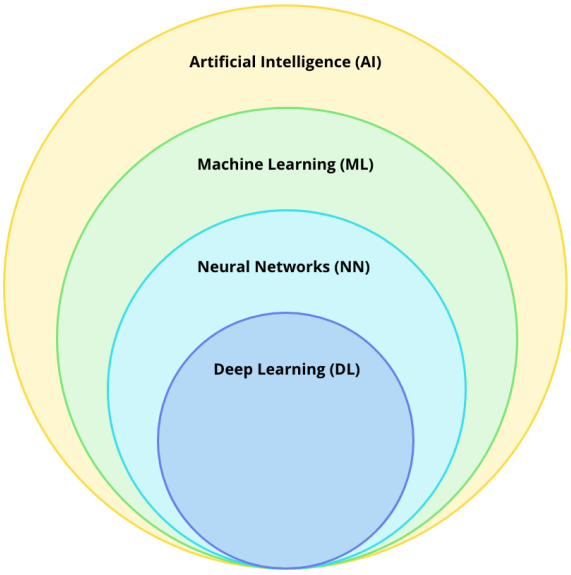
Een ML pipeline is zoals beknopt beschreven in deelparagraaf 2.1.2 een collectie van stappen dat wordt doorlopen om een model te trainen. Elke stap bevat een aantal acties dat wordt uitgevoerd, zoals onbruikbare data weghalen of de prestatie analyseren. De stappen en acties wordt in paragraaf 4.3 uitgelegd. 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 [**introduction-to-machine-learning**].

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 [**ml-think-about-ml-brownlee**].



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 [**ml-neural-network-nicholson**].

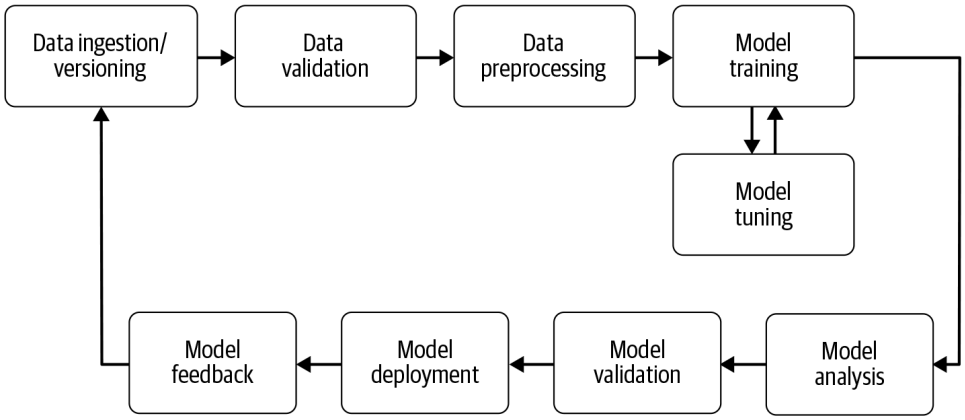
In het ML domein bestaan talloze algoritmes om voorspellingen te maken. In de [BIJLAGE

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. Een ML pipeline kan dit op een gestructureerde en reproduceerbare manier doen. In het ML domein bestaat geen consensus 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 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). In totaal zijn er, zonder de feedback loop stap, acht stappen dat elke keer doorlopen moeten worden om een model te trainen.



Figuur 4.2: Lifecycle van een model volgens Hapke en Nelson [**building-machine-learning-pipelines-oreilly**].

4.2.1 Voor- en nadelen van een machine learning pipeline

Hapke en Nelson [**building-machine-learning-pipelines-oreilly**] benoemen een aantal voordelen bij het gebruik van een pipeline voor het trainen van modellen:

- Voorkomen van bugs

  De stap waarbij de data wordt voorbereid is gebonden aan het trainen van een model. Zonder een pipeline zou het kunnen voorkomen dat een model is getraind en achteraf het proces om de data voor te bereiden is aangepast. Volgens Hapke en Nelson kan dit zonder een geautomatiseerde workflows zorgen voor bugs [**building-machine-learning-pip**

- Behulpzame broodkruimels

  Bij het doorlopen van de pipeline worden zaken bijgehouden zoals hyperparameters, gebruikte datasets en de modelstatistieken. Ook is te zien welk model momenteel is uitgerold. Mocht er wat fout gaan tijdens het trainen of uitrollen, is ook informatie terug te zien om het probleem te verhelpen.

- Standaardisatie binnen het team

  Binnen een team zal er één correcte manier zijn om een model te trainen; met de stappen in een pipeline. Dit zorgt ervoor dat er consistentie is tussen teamleden als een model wordt getraind en is het makkelijker voor nieuwe teamleden om te starten.

Een nadeel van het gebruik van een ML pipeline is dat het opzetten tijdrovend is en er kennis is vereist voor het opzetten en onderhouden. Dit is een nadeel dat NGTI zelf voorziet.

4.3 De stappen met een praktisch voorbeeld

Om te begrijpen wat er in de stappen gebeurt wordt elke stap kort uitgelegd met daarnaast een praktisch voorbeeld. Het praktisch voorbeeld is een ML pipeline dat geschreven is in Python en lokaal gedraaid kan worden. Niet alle stappen kunnen worden vertaalt naar de lokale ML pipeline. Dit zal worden aangegeven bij de stappen in kwestie.

4.3.1 Praktisch voorbeeld

Elk ML model wordt vergezeld door een ML probleem. Het ML probleem dat wordt opgelost door middel van de lokale ML pipeline is een bekend probleem binnen het ML domein, namelijk het identificeren van de iris familie. Hiervoor wordt de iris dataset gebruikt [**iris-dataset**]. Met deze dataset kan een ML model identificeren tot welke familie een bloem hoort op basis van de afmetingen van de kelk- en bloemblad. De dataset waarmee het model wordt getraind bestaat uit 150 voorbeelden van drie iris soorten: Setosa, Virginica en Versicolor. Elk voorbeeld heeft de lengte en breedte van de kelk- en bloemblad met de bijbehorende soort iris. In Tabel 4.1 is een voorbeeld van de dataset te zien. Het model zal een van de drie soorten kunnen herkennen op basis van de lengte en breedte van een gegeven kelk- en bloemblad.

Kelkblad - lengte	Kelkblad - breedte	Bloemblad - lengte	Bloemblad - breedte	Soort
5.1	3.5	1.4	0.2	Iris-setosa

Tabel 4.1: Voorbeeld van de iris dataset

Het doel is dus om de stappen in de ML pipeline te beproeven en niet om te experimenteren met een onopgelost probleem.

4.3.2 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 ko-

men, zoals lokaal, een storage bucket 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 dataset zijn allebei 20% van de originele dataset [**building-machine-learning-pipelines-oreilly**].

Een usecase 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 handig om versiebeheer toe te passen. Zo is goed te zien welke dataset welk model produceert. Een versie geven aan een dataset gebeurt voordat de dataset wordt ingeladen [**building-machine-learning-pipelines-oreilly**]. Versiebeheer voor datasets kan bijvoorbeeld met DVC [**dvc**] of Pachyderm [**pachyderm**].

De iris dataset wordt in Figuur 4.3 geïmporteerd door middel van code. Normaal gesproken zou, voordat deze code uitgevoerd wordt, een versie gegeven worden aan de dataset. Omdat de schaal van dit voorbeeld klein is en om de voorbeeld reproduceerbaar te houden is dit niet gedaan. De dataset wordt aangemaakt met de namen voor de kolommen op regel 5 net zoals het voorbeeld in Tabel 4.1. Vervolgens is de dataset onder de variabel naam $dataset$ beschikbaar voor de komende stappen.

```
1 # URL to the dataset
2 url =
  "https://gist.githubusercontent.com/UNRULYEON/7765e8bc37f928fc91aea3
  d109c337f5/raw/890f6e5c11f740150d57c578cf06d5bf9f35000d/iris.csv"
3
4 # Names of the columns
5 names = ["sepal-length", "sepal-width", "petal-length", "petal-
  width", "class", ]
6
7 # Create dataset
8 dataset = read_csv(url, names=names)
```

Figuur 4.3: Dataset importeren

4.3.3 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 [**building-machine-learning-pipelines-oreilly**]. Tijdens de validatie stap wordt gecontroleerd op het volgende:

- Afwijkingen in de dataset
- Wijzigingen in de structuur
- Algemene statistieken in vergelijkingen met voorgaand datasets [**building-machine-learning-pipel**

De controle kan worden gedaan met behulp van gegenereerde statistieken. Uit de statistieken kan blijken dat de dataset meer van een datapunt heeft dan andere. Dit kan ervoor zorgen dat het model beter presteert op een specifieke datapunt ten opzichte van andere datapunten.

De code voor het genereren van de statistieken hangt af van hoe de dataset eruit ziet en wat er gegenereerd moet worden. In Figuur 4.4 is te zien hoe statistieken worden gegenereerd uit de iris dataset.

```
# Statistical summary
print(dataset.shape)
print(dataset.head(5))
print(dataset.describe())
print(dataset.groupby('class').size())

# Visual summary
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False,
sharey=False)
pyplot.show()

dataset.hist()
pyplot.show()

scatter_matrix(dataset)
pyplot.show()
```

Figuur 4.4: Statistieken genereren

	sepal-length	sepal-width	petal-length	petal-width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Figuur 4.5: Statistieken - Eerste 5 regels van de iris dataset

	sepal-length	sepal-width	petal-length	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Figuur 4.6: Statistieken - De iris dataset beschreven

```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

Figuur 4.7: Statistieken - Iris dataset gegroepeerd

4.3.4 Stap 3: Data voorbereiden (Data preprocessing)

Niet alle data types kan in een model [source]. Data moet worden opgeschoond en gewijzigd worden zodat een model efficiënt ermee kan werken [source].

Opschoon(technieken/manieren):

- Schalen

  Uitleg schalen

- Normaliseren

  Uitleg normaliseren

- Hot encoding

  Uitleg Hot encoding

Sommige frameworks zoals TFX hebben functies die je kan gebruiken om dit te doen. Voordeel van het gebruik van die functies is:

- Efficiënt verwerken van data

  Uitleg hoe

- Training-serving skew

  Uitleg training-serving skew – Met een pipeline kan je training-serving skew voorkomen. Training-serving skew is dat de functies/APIs die gebruikt zijn om de data voor te bereiden om het model te trainen anders kunnen zijn als er een predictie wordt gedaan.

uitleg hoe het in het iris voorbeeld gedaan zou worden

4.3.5 Stap 4 en 5: Model trainen en tunen (Model training - Model tuning)

Een ML algoritme is net zoals een conventionele algoritme in software engineering. Het algoritme slaat na het trainen met een dataset de regels, nummers en algoritme specifieke data structuren op in de vorm van een model. Het model is als het ware een programma waarmee, gegeven een input, voorspellingen mee gedaan kan worden [https://machinelearningmastery.c
between-algorithm-and-model-in-machine-learning/].

Er zijn talloze algoritmes [algoritmes mindmap]. Algoritmes kunnen specialiseren in specifieke predicties, zoals image recognition of voorspellen van woorden. Het is handig om vooraf te testen welke het beste werkt voor het probleem. Het zou ideaal zijn als dit onderdeel is van de pipeline.

Over het algemeen gebeurt er hetzelfde, ongeacht welk algoritme is gebruikt [p84]:

- Laad de train en validatie dataset
- Definieer het model
- Train het model
- Sla het model op

[Hyperparameters uitleg]

[Uitleg hyperparameters SVC] [https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html]

4.3.6 Stap 6 en 7: Model analyse en validatie (Model analysis - Model validation)

Ethische verantwoording van de model. Waarom het belangrijk is om het model te analyseren. Hoe behandeld het model voorspellingen van:

- mensen met verschillende variabelen zoals geslacht, locatie of leeftijd.
- image recognition met verschillende licht condities.

Meetinstrumenten voor algoritme groepen.

Voor voorbeeld kan classification metrics (confusion matrix) gebruikt worden.

4.3.7 Stap 8: Model uitrollen (Model deployment)

Het opgeslagen model kan simpelweg geladen worden in een webserver zoals Flask. Dit schaalt echter niet en het is lastig om bij te houden met welke versie van het model een voorspelling wordt gedaan.

Tensorflow Serving maakt dit makkelijker.

4.3.8 Stap 9: Feedback loop (Model feedback)

Impliciet en expliciet feedback

- Impliciet

  Een gebruiker kijkt een gesuggereerde film of koopt een gesuggereerde
- Expliciet

  De applicatie vraagt aan de gebruiker of de suggestie goed is

s

4.4 Machine learning versimpelen

4.5 Conclusie

4.6 Advies

5 Frameworks dat platformen beheert

Gewenste situatie

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.

6 Architecturale ontwerp van de oplossing

6.1 Literatuur

6.2 Design

6.3 Conclusie

7 Oplossing

7.1 Scope definiëren

7.2 Research techstack

7.3 Wireframe

7.4 Mockup

7.5 POC

7.6 Conclusie

8 Conclusie

9 Aanbeveling

10 Discussie

11 Reflectie

Bibliografie

Bijlagen

Scope hoofd- en deelvragen

Scope deelvraag 1

D1: Uit welke stappen bestaat een machine learning pipeline?	
Scope	Binnen de scope: • 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 Buiten de scope: • Automatisering van stappen en acties • Een versimpeling van machine learning
Toelichting	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.

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	Binnen de scope: • 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 Buiten de scope: • 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	Binnen de scope: • Technische tekeningen Buiten de scope: -
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

Criteria met toelichting voor cloud computing platformen

Criteria	Toelichting
ML Pipeline aanmaken	Het platform moet het aanmaken van een machine learning pipeline kunnen ondersteunen.
Serverbeheer	Aanmaken, wijzigen en verwijderen van een server. Een server kunnen aanmaken bij het platform is praktisch omdat een server voor meerdere doeleinde gebruikt kan worden.
Database beheer	Aanmaken, wijzigen en verwijderen van een database. De PoC moet data ergens kunnen opslaan en vandaan halen; dit is mogelijk in een database.
Storage bucket beheer	Aanmaken, wijzigen en verwijderen van een storage bucket. In een storage bucket kan grote hoeveelheden data opgeslagen worden zoals bestanden en afbeeldingen. Dit kan handig zijn om bijvoorbeeld train en test data op te slaan.
Uptime 99.9%	Het platform moet een vorm van uptime kunnen garanderen waarbij het percentage zo dicht mogelijk bij 99.9 ligt. Hierdoor is de kans dat NGTI door een storing bij een platform niet productief kan zijn zo klein mogelijk.
Regionale beschikbaarheid	Waar data wordt opgeslagen en de servers draaien is vrij belangrijk. Het platform moet ten minste West-Europa ondersteunen in verband met de gegevensbescherming in de EU (GDPR). In een ideale situatie zou het platform ook specifiek Zwitserland ondersteunen aangezien de meeste klanten van NGTI vandaar komen.
Toegankelijkheid documentatie ML pipeline	In het geval dat de PoC wordt uitgebreid tot een applicatie is het belangrijk om onderhoud uit te voeren en eventueel nieuwe functies toe te voegen. Het is daarom van belang dat het platform documentatie heeft over ML pipelines.
APIs	Om het platform programmatisch aan te sturen zijn APIs dat het platform beschikbaar stelt onmisbaar. Op deze manier kunnen frameworks het platform beheren en kan eventueel een op maat gemaakte oplossing gebruikt worden.
Inhoud ML	Voor NGTI is het belangrijk dat het platform niet alleen ondersteuning heeft om modellen te trainen, maar meer mogelijkheden biedt zoals het schrijven van een eigen algoritme of out-of-the-box oplossingen.

Tabel 5: Criteria voor cloud computing platformen

Gedetailleerd overzicht cloud computing platformen

AWS

Criteria	Toelichting
ML Pipeline aanmaken	Met Amazon SageMaker kunnen pipelines gemaakt worden. Daarnaast heeft Amazon specifieke services om bijvoorbeeld vooroordelen te detecteren, statistieken te meten en data te verzamelen (overzicht).
Serverbeheer	Met Amazon EC2 kunnen servers worden opgestart, gewijzigd of verwijderd worden. Een overzicht wat Amazon EC2 allemaal kan
Database beheer	Amazon biedt een database service aan waarbij een database aangemaakt, gewijzigd of verwijderd kan worden. De databases zijn gesorteerd op type in dit overzicht.
Storage bucket beheer	Wat betreft opslag biedt Amazon twee soorten aan: Amazon S3 en Amazon Elastic Block Store. Het verschil is dat S3 data opslaat als een object ten opzicht van Elastic Block Storage, dat data op de conventionele manier opslaat.
Uptime 99.9%	De uptime staat beschreven in de Service Level Agreement (SLA) en is voor elke service anders. Over het algemeen biedt Amazon voor elke service een uptime van 99.9%: Amazon SageMaker, Amazon EC2 en Elastic Block Store, Amazon Relational Databases en Amazon S3.
Regionale beschikbaarheid	Amazon heeft in totaal zes fysieke datacenters in Europa (overzicht). Ook is het mogelijk om een specifiek regio te kiezen zoals het hier beschreven staat. Helaas is Zwitserland geen beschikbare regio.
Toegankelijkheid documentatie ML pipeline	Amazon heeft een uitgebreide services voor ML binnen AWS. Alle services zijn te vinden in dit (overzicht).
APIs	Voor al haar services stelt Amazon een API beschikbaar. Een overzicht voor de documentatie is hier te vinden. API documentatie voor ML en SageMaker is ook beschikbaar.
Inhoud ML	Amazon heeft een groot aantal ML specifieke services waar gebruik van gemaakt kan worden. Een paar voorbeelden daarvan is Amazon Translate, Amazon Healthlake en Amazon Forecast.

Tabel 6: AWS tegenover criteria op 06-04-2021

Azure

Criteria	Toelichting
ML Pipeline aanmaken	
Serverbeheer	
Database beheer	
Storage bucket beheer	
Uptime 99.9%	
Regionale beschikbaarheid	
Toegankelijkheid documentatie ML pipeline	
APIs	
Inhoud ML	

Tabel 7: Azure tegenover criteria op 06-04-2021

Google Cloud

Criteria	Toelichting
ML Pipeline aanmaken	
Serverbeheer	
Database beheer	
Storage bucket beheer	
Uptime 99.9%	
Regionale beschikbaarheid	
Toegankelijkheid documentatie ML pipeline	
APIs	
Inhoud ML	

Tabel 8: Google Cloud tegenover criteria op 06-04-2021

DigitalOcean

Criteria	Toelichting
ML Pipeline aanmaken	
Serverbeheer	
Database beheer	
Storage bucket beheer	
Uptime 99.9%	
Regionale beschikbaarheid	
Toegankelijkheid documentatie ML pipeline	
APIs	
Inhoud ML	

Tabel 9: DigitalOcean tegenover criteria op 06-04-2021

IBM Cloud

Criteria	Toelichting
ML Pipeline aanmaken	
Serverbeheer	
Database beheer	
Storage bucket beheer	
Uptime 99.9%	
Regionale beschikbaarheid	
Toegankelijkheid documentatie ML pipeline	
APIs	
Inhoud ML	

Tabel 10: IBM Cloud tegenover criteria op 06-04-2021

Alibaba

Criteria	Toelichting
ML Pipeline aanmaken	
Serverbeheer	
Database beheer	
Storage bucket beheer	
Uptime 99.9%	
Regionale beschikbaarheid	
Toegankelijkheid documentatie ML pipeline	
APIs	
Inhoud ML	

Tabel 11: Alibaba tegenover criteria op 06-04-2021

Oracle Cloud Infrastructure

Criteria	Toelichting
ML Pipeline aanmaken	
Serverbeheer	
Database beheer	
Storage bucket beheer	
Uptime 99.9%	
Regionale beschikbaarheid	
Toegankelijkheid documentatie ML pipeline	
APIs	
Inhoud ML	

Tabel 12: Oracle Cloud Infrastructure tegenover criteria op 06-04-2021

Kamatera Cloud

Criteria	Toelichting
ML Pipeline aanmaken	
Serverbeheer	
Database beheer	
Storage bucket beheer	
Uptime 99.9%	
Regionale beschikbaarheid	
Toegankelijkheid documentatie ML pipeline	
APIs	
Inhoud ML	

Tabel 13: Kamatera Cloud tegenover criteria op 06-04-2021

Cloudways

Criteria	Toelichting
ML Pipeline aanmaken	
Serverbeheer	
Database beheer	
Storage bucket beheer	
Uptime 99.9%	
Regionale beschikbaarheid	
Toegankelijkheid documentatie ML pipeline	
APIs	
Inhoud ML	

Tabel 14: Cloudways tegenover criteria op 06-04-2021

Vultr

Criteria	Toelichting
ML Pipeline aanmaken	
Serverbeheer	
Database beheer	
Storage bucket beheer	
Uptime 99.9%	
Regionale beschikbaarheid	
Toegankelijkheid documentatie ML pipeline	
APIs	
Inhoud ML	

Tabel 15: Vultr tegenover criteria op 06-04-2021

BigML Inc.

Criteria	Toelichting
ML Pipeline aanmaken	
Serverbeheer	
Database beheer	
Storage bucket beheer	
Uptime 99.9%	
Regionale beschikbaarheid	
Toegankelijkheid documentatie ML pipeline	
APIs	
Inhoud ML	

Tabel 16: BigML Inc. tegenover criteria op 06-04-2021

H2O.ai Inc.

Criteria	Toelichting
ML Pipeline aanmaken	
Serverbeheer	
Database beheer	
Storage bucket beheer	
Uptime 99.9%	
Regionale beschikbaarheid	
Toegankelijkheid documentatie ML pipeline	
APIs	
Inhoud ML	

Tabel 17: H2O.ai Inc. tegenover criteria op 06-04-2021