Machine Learning Pipeline Automation

Amar Kisoensingh

15 maart 2021

Voorwoord

Samenvatting

Summary

Lijst van figuren

Lijst van tabellen

Inhoudsopgave

1 Inleiding

1.1 Het bedrijf: NGTI

NGTI is een mobile app development bureau gevestigd in Rotterdam, Nederland en maakt hoogwaardige mobiele applicaties voor native, hybrid en webgebruik.

1.1.1 Klanten van NGTI

1.1.2 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.

Slack

Interne communicatie gaat via Slack. Het programma faciliteren 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*.

Google Workspace

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

Zoom

Voorheen werdt Zoom alleen gebruikt om te videobellen met collega's en interviewees. In de tijd van het 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.

1.2 Opdracht

NGTI heeft voorzien dat ze haar applicaties 'slimmer' moet maken door machine learning in te zetten. Niet alleen zorgt dit voor een betere gebruikerservaring, maar geeft NGTI ook een voorsprong op haar concurrenten.

Om machine learning toe te passen is het raadzaam om een machine learning pipeline op te zetten. Een pipeline is een gestructureerde werkwijze om een model te trainen. Het opzetten van de pipeline en een competente model trainen is tijdrovend en vereist kennis in het domein. Vaak worden modellen getraind op een cloud computing platform zoals Azure, AWS of Google Cloud. Het probleem met platformen zoals deze is dat het ontzettend lastig is om te wisselen van platform en er is veel kennis vereist om een infrastructuur op te zetten.

De opdracht bestaat uit twee onderdelen:

1. onderzoek naar het automatiseren van het opzetten van een machine learning pipeline om het laagdrempelig en minder tijdrovend te maken.
2. onderzoek naar het maken van een platform agnostische oplossing

Hierbij zal een PoC gemaakt worden om aan te tonen of het haalbaar is. Een diepere duik in het probleem is te vinden in hoofdstuk 2.

1.3 Leeswijzer

2 Probleemanalyse

2.1 Probleemdefinitie

Zoals beschreven in paragraaf 1.2 is NGTI genoodzaakt om machine learning in te zetten om haar applicaties 'slimmer' te maken. Hier zijn een aantal redenen voor, namelijk:

1. gebruikerservaring verbeteren
2. voorsprong hebben op concurrenten

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

2.1.1 Expertise Machine Learning

Machine learning is geen triviaal onderwerp. Om een model te trainen is kennis nodig van verschillende domeinen: datamining, software engineering en statistieken. In een multidisciplinair team is het voor één teamlid niet nodig om alle domeinen te beheersen.

Doordat er voorkennis nodig is om een model te trainen, is het vaak te hoogdrempelig om te beginnen voor developers. De expertise is daar bovenop niet in een korte tijd te vergaren.

2.1.2 Opzetting pipeline

Er bestaan verschillende manieren om een model te trainen. Een machine learning pipeline opzetten is daar een van. In een pipeline wordt voor het trainen van het model de data voorbereid. Het opzetten van een pipeline kost tijd. Op zichzelf niet zo zeer veel tijd, maar als er veel wordt geëxperimenteerd met het trainen van modellen kan de tijd opstapelen. Ook zijn de stappen in een pipeline over het algemeen hetzelfde, ongeacht wat voor model je traint. Hierdoor worden taken vaak herhaalt tussen het opzetten van verschillende pipelines.

2.1.3 Vendor lock-in

Er bestaan een aantal diensten, zogenoemde Platform as a Service (PaaS), waarbij je een pipeline kan opzetten. 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 Doelstelling

De doelstelling is om een systeem te ontwikkelen waarbij developers met weinig tot geen kennis een model kunnen trainen, onderdelen van de pipeline geautomatiseerd zijn en platform agnostisch is.

Het trainen van een model is een iteratief proces omdat de data waarmee het model getraind is verouderd waardoor het model niet meer optimaal presteert. Om een nieuw model te trainen en consistent te blijven met hoe een model getraind wordt kan een pipeline opgezet worden. Een pipeline is dus herbruikbaar.

2.3 Bestaand oplossingen

Er bestaan een aantal oplossingen voor vrijwel alle problemen dat NGTI ondervindt. Elk oplossing is een PaaS van een derde partij waarbij vendor lock-in inherent is. Dit maakt ze ongeschikt maar betekent echter niet dat ze nutteloos zijn. Er kan namelijk gekeken worden hoe een pipeline wordt opgezet en daar vervolgens (gedeeltelijk) het systeem op baseren.

De oplossingen kunnen gecategoriseerd worden in twee groepen:

1. Machine learning pipeline specifieke services
2. Cloud computing platformen

2.3.1 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 documentatie een aantal blog posts die bij het ontwerpen van het systeem relevant zouden kunnen zijn.

2.3.2 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.

2.4 Hoofd- en deelvragen

Uitgaand van de drie focuspunten in paragraaf 2.1 kan de hoofdvraag als volgt worden geformuleerd:

Hoe kan een machine learning pipeline worden geautomatiseerd onafhankelijk van de onderliggende cloud computing platform of lokale framework?

De hoofdvraag kan worden onderbouwd met vier deelvragen. Om te beginnen is het verstanding om te onderzoeken hoe een machine learning model wordt getraind:

Welke stappen moeten worden ondernomen om een machine learning-model te trainen?

Vervolgens kan er op de deelvraag voortborduurt worden om het opzetten van een pipeline in kaart te brengen:

Hoe wordt een machine learning-pipeline opgezet?

Verder zijn er verschillende cloud computing platformen en lokale frameworks waarmee machine learning modellen getraind kunnen worden. Doordat het systeem platform agnostisch moet zijn is het van belang om verschillende platformen en frameworks te onderzoeken:

Wat zijn de verschillende en overeenkomsten tussen cloud computing platforms en lokale frameworks waarmee machine learning-modellen kunnen worden getraind?

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, waarin een machine learning pipeline kan worden opgezet en die platform-onafhankelijk is, eruit?

3 Onderzoeksmethoden

D1: Welke stappen moeten worden ondernomen om een machine learning-model te trainen?	
Doel	Onderzoek naar wat machine learning is en hoe een model getraind kan worden. Een visualisatie van hoe een model getraind wordt, een PoC waarbij een model getraind wordt en een conclusie en advies met vooral waarop gelet moet worden bij het trainen.
Methode	Kwalitatief
Validatie	Er is uitleg gegeven wat machine learning is en hoe een model getraind kan worden op een simpele manier waarbij de visualisatie de uitleg compleet maakt. Met het conclusie en advies kan iemand die vrijwel geen kennis heeft aan de slag met het trainen van een model.

Tabel 3.1: Deelvraag 1

D2: Hoe wordt een machine learning pipeline opgezet?	
Doel	Onderzoek naar wat een machine learning pipeline is, waarom het gebruikt wordt en welke onderdelen van de pipeline geautomatiseerd kunnen worden. PoC waarbij een pipeline met het model van de vorige deelvraag wordt opgezet. Conclusie en advies.
Methode	Kwalitatief
Validatie	Het is duidelijk wat een pipeline is en waarom het handig/noodzakelijk is om te gebruiken. In de conclusie en advies is het onder andere duidelijk of onderdelen geautomatiseerd kunnen worden.

Tabel 3.2: Deelvraag 2

D3: Wat zijn de verschillen en overeenkomsten tussen cloud computing platforms en lokale frameworks waarmee machine learning-modellen kunnen worden getraind?	
Doel	De grote cloud computing platformen en lokale frameworks in kaart brengen. De features, voor- en nadelen worden met elkaar vergeleken. Waar van toepassing wordt een kosten berekening gedaan. Conclusie en advies bevat onder andere welke platform en/of framework het beste is om mee te beginnen.
Methode	Kwalitatief
Validatie	Een overzichtelijke vergelijking tussen de cloud computing platformen en lokale frameworks is gemaakt. Uit de conclusie en advies moet blijken welke platform en/of framework het beste is om mee te beginnen.

Tabel 3.3: Deelvraag 3

D4: Hoe ziet de architecturale blauwdruk van een applicatie, waarin een machine learning pipeline kan worden opgezet en die platform-onafhankelijk is, eruit?	
Doel	Er wordt research gedaan naar hoe een pipeline geautomatiseerd opgezet kan worden en naar verschillende tools/frameworks. Technische designs worden gemaakt ter validatie en als blauwdruk.
Methode	Kwalitatief
Validatie	Uit de research en designs is het duidelijk hoe een systeem gebouwd kan worden.

Tabel 3.4: Deelvraag 4

H: Hoe kan een machine learning pipeline worden geautomatiseerd onafhankelijk van de onderliggende cloud computing platform of lokale framework?	
Doel	Uit de requirements wordt de scope bepaald. Er worden designs gemaakt om de interface te testen. Een PoC wordt gebouwd.
Methode	Kwalitatief
Validatie	De PoC is werkzaam en voldoet aan de requirements.

Tabel 3.5: Hoofdvraag

4 Requirementsanalyse

4.1 Stakeholders

4.2 Requirements

5 Hoe werkt Machine Learning?

Machine learning is onderdeel van het domein computer science. Binnen machine learning worden algoritmische modellen gebouwd om statistische problemen op te lossen. Dit wordt gedaan met behulp van een dataset dat verzameld is uit de echt wereld of gefabriceerd door mensen. Het model kan met behulp van de dataset een redelijk nauwkeurige voorspelling maken [**the-hundred-page-machine-learning-book**].

Hoe het niet AI is -¿ https://machinelearningmastery.com/think-machine-learning/

5.1 Hoe leert een machine learning model?

Een machine learning model leert met behulp van een dataset om een voorspelling te maken. Een dataset kan duizenden feature vectors (voorbeelden) bevatten, waarbij elke vector een of meerdere features (attribuut) heeft. In een voorbeeld waarbij een model moet kunnen voorspellen of een persoon kanker heeft kan een feature vector één persoon zijn en de features het gewicht, leeftijd, lengte en of de persoon kanker heeft [**google-ml-terminology**].

Het model gaat langs alle feature vectors en probeert correlaties te vinden tussen de features. Op basis hiervan kan het model vervolgens een voorspelling doet met een feature vector dat het model nooit heeft gezien.

Er zijn een aantal manieren om een algoritme te laten trainen, gegroepeerd in vier vormen.

5.1.1 Supervised learning

Met supervised learning wordt een model getraind door middel van een **gelabeld** dataset. Dit betekent dat een feature vector gepaard gaat met de gewenste label.

Feature	**Label**
woorden, afzender, tijd van versturen	spam/geen spam

Tabel 5.1: Voorbeeld gelabeld dataset

Het model kan leren dat woorden zoals 'gratis' in e-mails die tussen 01:00 en 03:00 worden verstuurd vaak als spam gelabeld zijn. Het model kan concluderen dat als een e-mail met dezelfde features als input word gegeven, dat het een spam e-mail is [**google-ml-terminology**].

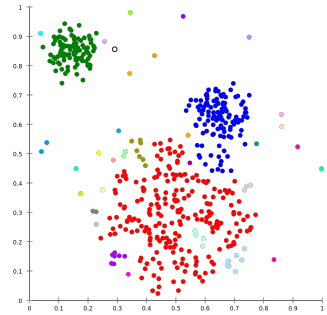
5.1.2 Unsupervised learning

Unsupervised learning lijkt grotendeels op supervised learning. Het verschil is dat de feature vectors in de dataset niet gepaard zijn met een label.

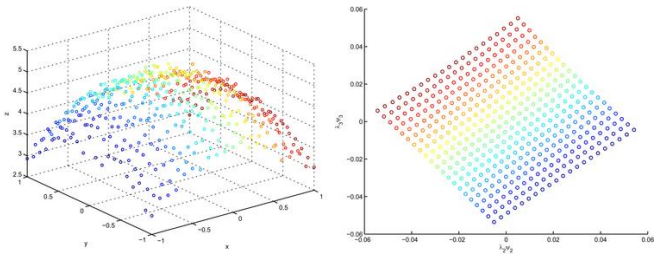
Feature	**Label**
woorden, afzender, tijd van versturen	?

Tabel 5.2: Voorbeeld niet gelabeld dataset

Het doel bij unsupervised learning is om de data op een bepaalde manier te transformeren om er iets waardevols uit te halen. Een voorbeeld hiervan is het *clusteren* van data. Hierbij wordt soortgelijke data gegroepeerd (Figuur 5.1). Een andere voorbeeld is *dimensionality reduction* (Figuur 5.2). Feature worden weggelaten om twee features tegen over elkaar te zetten om bijvoorbeeld uitschieters te vinden [**the-hundred-page-machine-learning-book**].



Figuur 5.1: Voorbeeld van *clustering*



Figuur 5.2: Voorbeeld van *dimensionality reduction*

5.1.3 Semi-supervised learning

Semi-supervised learning is een combinatie van supervised learning en unsupervised learning waarbij het doel dezelfde is als supervised learning. Het is gebruikelijk dat het merendeel van de dataset niet gelabelde data is [**the-hundred-page-machine-learning-book**].

5.1.4 Reinforcement learning

Reinforcement learning wordt ook wel 'survival learning' genoemd. Een analogie hiervoor is een organisme dat moet overleven in verschillende situaties door de juiste actie te verrichten. Acties kunnen een positief of een negatief effect hebben. Acties dat langdurig een positieve effect hebben moeten versterkt worden zodat het model zo veel mogelijk positieve acties verricht [**bayesian-reasoning-and-machine-learning-book**].

5.2 Stappenplan voor het trainen van een model

Het trainen van een model neemt een klein gedeelte in beslag van het hele proces. Vooraf moet het probleem worden gedefinieerd waarna vervolgens de data wordt verzameld. Het is belangrijk om de data grondig te verkennen want als het model traint met onbruikbare data, is het model ook onbruikbaar. Dit heet ook wel garbage in = garbage out. Na het verkennen kan de data opgeschoond worden. Dit wordt gedaan met de garbage in = garbage out regel in gedachte; hoe beter de data, hoe beter het model zal presteren. Feature engineering is het tegenovergestelde van het opschonen van de dataset. Waarbij data wordt weggehaald, wordt er nieuwe data geïntroduceerd. Dit wordt gedaan met de bestaande data en zorgt ervoor dat belangrijke elementen van de dataset worden aangestipt voor het model. Voor het trainen van het model moet de dataset gesplitst worden in een training dataset en test dataset. Daarna kan het model getraind worden met behulp van het afstemmen van model- en hyperparameters en cross validation. Model- en

hyperparameters zijn 'instellingen' voor het model en met cross validation kunnen verschillende instellingen worden getest met alleen de training dataset. Als laatste kan een model gekozen worden dat het beste presteert [**data-science-primer**].

Deze stappen kunnen verschillen. Kan voorkomen dat je meer stappen moet doen of andere. Sommige modellen vereisen specifieke handelingen, deze stappen zijn voor de meest gebruikte.

5.2.1 Data verkennen

Bij het verkennen wordt gekeken welke features kunnen worden gebruikt. Het kan handig zijn om de data te plotten in bijvoorbeeld een histogram, een staafdiagram of een boxplot. [**data-science-primer**] Het plotten kan handig zijn voor het vinden van:

- uitschieters
- meetfouten
- features die gescheiden het model negatief kunnen beïnvloeden, maar samen positief

Vervolgens kunnen er correlaties tussen variabelen gevonden worden. Variabelen in een dataset kunnen gerelateerd aan elkaar zijn op een aantal manieren. Brownlee [**ml-correlation-brownlee**] geeft een aantal voorbeelden:

- Een variabele kan de waarde van een andere variabele veroorzaken of hiervan afhankelijk zijn
- Een variabele kan lichtelijk geassocieerd zijn met een ander variabele
- Twee variabele kan afhankelijk zijn van een derde onbekende variabele

Een correlatie kan positief wat betekent dat de variabelen in dezelfde richting bewegen, bijvoorbeeld het gewicht en de lengte van een persoon. Het kan ook negatief zijn. Dit betekent dat als één variabele de ene kant op gaat, de andere variabele de tegengestelde richting op gaat. Een voorbeeld hiervan is het aantal tentamens en recreative tijd. Bij een neutrale correlatie betekent het dat de variabelen niks met elkaar te maken hebben [**ml-correlation-brownlee**].

De relatie tussen twee variabelen heet de covariantie. Met een formule kan de covariantie worden berekent:

$$cov(X, Y) = (sum(x - mean(X)) * (y - mean(Y))) * 1/(n - 1)$$

Waarbij het resultaat het volgende kan zijn:

$$[[385.33297729, 389.7545618][389.7545618, 500.38006058]]$$

Omdat dit moeilijk te interpreteren kan zijn, werden er coëfficiënten bedacht. Twee daarvan zijn de *Pearson correlation coefficient* en de *Spearman correlation coefficient*. De achterliggende formule verschillen, maar het resultaat moet de covariantie in context bren-

gen. Het resultaat uit een coëfficiënt formule is een getal tussen de 1 en -1, waarbij 1 een positieve correlatie is en een -1 een negatieve correlatie. Er is geen correlatie als het resultaat een 0 is. Vaak is er een correlatie als er een waarde boven $0,5$ en onder $-0,5$ voorkomt.

5.2.2 Data opschonen

Voor het opschonen van data is er geen beste manier omdat elk dataset verschilt van elkaar. Toch kunnen een aantal vuistregels aangehouden worden om na deze stap een betere dataset te hebben. Als eerst kan er gekeken worden welke data weg kan. Dit kan duplicaten zijn of data dat niet relevant is. Na het weggooien moet structurele fouten worden gecorrigeerd als ze voorkomen. Dit zijn fouten zoals typefouten, woorden die de ene keer wel hoofdletter zijn maar de andere keer niet, fout gelabelde data, enz.

Een optionele stap is om uitschieters weg te gooien. Het kan voorkomen dat uitschieters het model negatief beïnvloeden. Als dat het geval is, kan de uitschieter uit de dataset weggehaald worden.

Als laatste stap is het belangrijk om te kijken of er data mist. Hiervoor zijn er twee manieren op dit te verhelpen: het weglaten van data of infereren. Helaas zijn beide manieren sub-optimaal omdat er bij het weglaten kostbare data wordt weggegooid en het infereren is natuurlijk niet accuraat [**data-science-primer**].

5.2.3 Feature engineering

Bij feature engineering worden er nieuwe input features gemaakt die bestaan uit bestaande features. Het nut hiervan is dat er informatie aangestipt kan worden voor het model wat relevant is. Een manier hiervan is om 'interactie features' te maken. Hierbij worden twee features samengevoegd tot een derde feature. Een voorbeeld van het maken van een interactie feature:

> Een model moet kunnen voorspellen of een huis op de huizenmarkt het goed gaat doen. De dataset bevat een aantal features waaronder *num_schools* en *median_school*. *num_schools* is het aantal scholen binnen een radius van 5km van het huis en *median_school* het gemiddelde van alle scholen in de radius.
>
> Er kan beargumenteerd worden dat een huis beter verkoopt als er **veel scholen in de buurt zijn** die ook **een hoge score hebben**. Er kan een nieuwe feature gemaakt worden genaamd *school_score* dat wordt berekent door *num_schools* $*$ *median_school* te doen. *school_score* is de interactie feature.

Een tweede manier om features te maken is om bestaande features samen te voegen. Dit is handig om te doen als de features los van elkaar niet vaak voorkomen in de dataset én samenhang hebben met elkaar.

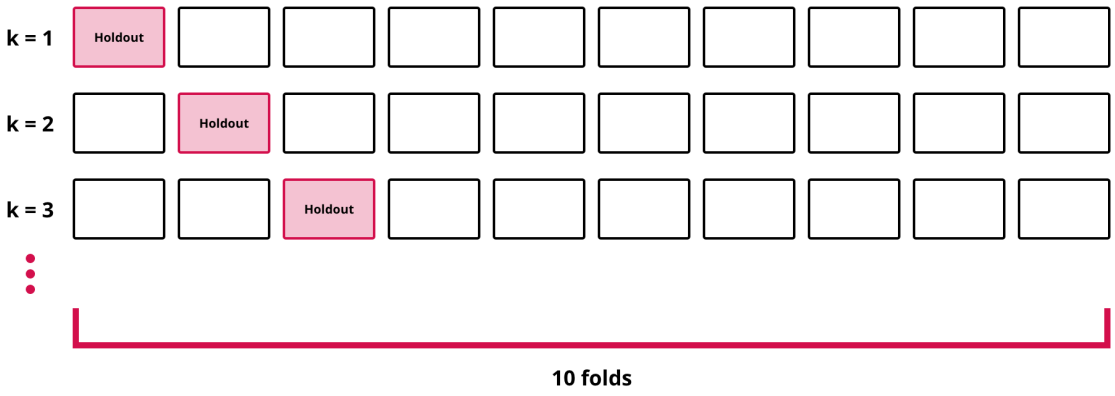
5.2.4 Een model trainen

Voordat het model getraind kan worden, moet de dataset gesplitst worden in een training dataset en een test dataset. Het is gebruikelijk om de dataset te splitsen in 80% training en 20% test of 50% training en 50% test; hiervoor is er geen vaste regel en hangt af van de

dataset en het model. De dataset wordt gesplitst om overfitting te voorkomen. Een model dat overfit is presteert met een hoge accuraatheid op de training dataset, maar laag op een dataset dat het model nog nooit heeft gezien. Door een test dataset apart te houden kan er gevalideerd worden of het model goed presteert met data dat het model nog nooit heeft gezien [**data-science-primer**].

Tijdens het trainen en voorspellen werkt het model met model parameters en model hyperparameters. Model parameters zijn interne variabelen van het model die niet expliciet gespecificeerd zijn. Model hyperparameters worden daarin tegen wel handmatig gespecificeerd en kan bijvoorbeeld het aantal leersessies zijn [**ml-model-hyper-parameter-brownlee**]. Het vinden van de juiste waarde van model hyperparameters kan alleen met experimenteren worden achterhaald [**data-science-primer**].

Omdat de test dataset niet kan worden gebruikt tijdens het experimenteren moet er gebruik worden gemaakt van cross validation. Er zijn een aantal manieren om cross validation uit te voeren waarbij **10-fold cross-validation** een van de meest gebruikte is. Bij deze methode wordt de training dataset opgesplitst in 10 delen, ook wel folds genoemd, waarvan 1 een *holdout* fold is. Het model zal met de 9 folds trainen en wordt getest met de holdout fold. Dit zal 10 keer gedaan worden met elke keer een andere holdout fold zoals weergegeven in Figuur 5.3. Na elke train sessie wordt een score berekent dat wordt opgeteld bij de andere sessies. Hiervan wordt het gemiddelde berekend en het resultaat is de *cross-validated score*. Hoe hoger dit resultaat, hoe beter het model presteert [**data-science-primer**].



Figuur 5.3: Visualisatie van de 10-k cross-validation methode

Om de beste set van parameters te achterhalen kan 10-k cross-validation toegepast worden op elke set. De cross-validated scores kunnen met elkaar worden vergeleken om de beste model hyperparameter voor het model te achterhalen. Een voorbeeld van de tests op een set is te zien in Figuur 5.4.

Model Hyperparameters		CV
Penalty ratio	Penalty strength	CV-score
75/25	0.01	0.63
75/25	0.05	0.64
75/25	0.10	**0.67**
50/50	0.01	0.62
50/50	0.05	0.63
50/50	0.10	0.66

Figuur 5.4: Test matrix op elke model hyperparameter set

Nu de 'beste' model hyperparameters zijn gevonden, kan het model getest worden tegen de test dataset. Er zijn een aantal prestatiestatistieken waarmee gekeken kan worden hoe goed het model presteert. Voor regressie modellen is **Mean Squared Error (MSE)** of **Mean Absolute Error (MAE)** aanbevolen waarbij het model beter presteert als het getal lager is [**data-science-primer**]. Voor classificatie modellen is de **Area Under ROC Curve (AUROC)** aanbevolen waar het model juist beter presteert als het getal hoger is [**data-science-primer**].

5.3 Kennis vereist om een model te trainen

5.4 Wachttijden voorspellen

5.5 Conclusie

5.6 Advies

6 Hoe werkt een Machine Learning pipeline

6.1 Literatuur

6.2 POC

6.3 Conclusie

7 Cloud computing platformen en lokale frameworks

7.1 Literatuur

7.2 Conclusie

8 Architecturale ontwerp van de oplossing

8.1 Literatuur

8.2 Design

8.3 Conclusie

9 Oplossing

9.1 Scope definiëren

9.2 Research techstack

9.3 Wireframe

9.4 Mockup

9.5 POC

9.6 Conclusie

10 Conclusie

11 Aanbeveling

12 Discussie

13 Reflectie

Bibliografie

Bijlagen