[Date]

Sem Plasmeijer

[Company name]

[Company address]

[Document title]

[Document subtitle]

Inhoudsopgave

[Hoe kunnen AI & ML worden onderhouden na implementatie en productie? 2](#_Toc131756839)

[Planning 2](#_Toc131756840)

[Data Preparation 2](#_Toc131756841)

[Model Engineering 2](#_Toc131756842)

[Model Evaluation 2](#_Toc131756843)

[Deployment 2](#_Toc131756844)

[Monitoring & Maintenance 3](#_Toc131756845)

[Model exports 3](#_Toc131756846)

[Hoe kunnen AI & ML verantwoordelijk worden gebruikt? 4](#_Toc131756847)

[Hoe verantwoord je data gebruik en data transformaties gemaakt door een algorithme? 4](#_Toc131756848)

[Systemen en keuzes 4](#_Toc131756849)

[Hoe kunnen AI & ML worden geïmplementeerd binnen een Javaproject? 5](#_Toc131756850)

# Samenvatting

# Inleiding

# Hoe kunnen AI & ML worden onderhouden na implementatie en productie?

Meeste machine learning models volgen een standaard stappenplan voordat deze in een productie omgeving worden gezet. De standaard zes stappen beschreven door datacamp, zijn:

1. Planning
2. Data Preparation
3. Model Engineering
4. Model Evaluation
5. Model Deployment
6. Monitoring and Maintance

(Datacamp, 2022)

### Planning

Onder planning valt het oppakken van domeinkennis en het verbeteren van de probleemstelling. Hierbij kan gedacht worden aan vroege hypothese stelling zetten over wat een netwerk mogelijk zou terug kunnen geven. De toepasbaarheid van het probleem verbeteren, en het zoeken naar databronnen en de hoeveelheid van deze databronnen.

### Data Preparation

Data prep is het vergaren van data, deze labelen, opschonen en onderhouden. Onder onderhouden valt het verwijderen en toevoegen van nieuwe features aan de data, maar ook de opslag van data en hoe deze gebruikt kan worden het model.

### Model Engineering

Model engineering is het onderdeel waarbij keuzes worden gemaakt voor welke vormen van machine learning modellen er gebruikt gaan worden, welke metrics hiervoor gebruikt gaan worden, en het in elkaar zetten van dit model.

### Model Evaluation

Vanuit de vierde stap wordt bepaald of het model klaar is voor productie, ruim 90% van offline getrainde modellen worden niet naar productie gebracht en worden herstart vanuit de data preparation stadium. Naast het verifiëren van de correctheid van een model wordt hier ook gekeken naar regelgeving en ethische risico’s die dit model van het algoritme mogelijk ondergaat.

### Deployment

Vergelijkbaar met andere services, ML deployment kan gedaan worden op lokale services, cloud VMs of op package niveau naar andere programma’s. Vanuit hier kan het worden aangeroepen via API-calls of direct vanuit de browser als dashboard.

In dit stadium van het project worden er ook al vergelijking gemaakt met andere modellen rondom het gebruik van RAM-allocatie, opslag en processing power doormiddel van A/B testen.

### Monitoring & Maintenance

Modellen die uiteindelijk volledig uitgerold zijn worden automatisch onderhouden vergelijkbaar met standaard algoritmes in het logging systeem. Wanneer een model abnormaal gedraagt, errors oplevert of klachten van klanten ontvangt zal het development team deze automatisch ontvangen of deze worden gelogd in het systeem. Vanuit hier worden modellen niet vaak volledige opnieuw het lifecycle in gebracht, ingevallen van software of architectuur updates zal het model opnieuw naar het data preparation stadium worden gebracht.

Voor dit onderzoek zal de focus liggen rondom de laatste twee stappen in de lifecycle van een ML model. Er zal gekeken worden naar hoe verschillende modellen productie ingebracht kunnen worden. Hoe deze op verschillende niveaus gehost kunnen worden, en hoe deze gemonitord kunnen worden op verschillende niveaus.

## Model exports

(Pykes, 2022) In een artikel rondom hoe verschillende ML libraries hun modellen opslaan komen een aantal verschillende technieken terug die door alle packages op hun eigen manier gebruiken. De 3 hoofdmanieren die universeel bruikbaar waren:

* Opslag van ML model parameters door middel van JSON
* Pipeline opslag door middel van PMML
* Model opslag door middel van HDF5

Het verschil ligt in het totale plaatje van de opslag. JSON-opslag heeft twee voordelen over de andere twee oplossingen. Eerst lightweight en niet ML model specifiek, doordat alleen de parameters worden opgeslagen is het snel her implementeerbaar in andere talen zolang de pipeline herbouwd kan worden. Dit leidt ook tot het tweede voordeel, je hebt volledig beheer van wat er naar de opslag met je model gebeurt. (Pykes, 2022)

PMML: Predictive Model Markup Language (Wikipedia, 2023) is een XML-taal bedoeld voor de overdacht van ML modellen. In tegenstelling tot JSON-opslag is PMML bedoeld voor de overdacht van volledige modellen. PMML is ontstaan in 1998 en wordt nog onderhouden door de Data MIning Group, door zijn lange bestaan is PMML robuust en de afkomst vanuit XML maakt het een universele oplossing voor de overdracht van modellen.

HDF5 of Hieracchical data format kan beschreven worden als een data of file systeem binnen een aparte file. Het is bedoeld voor de opslag van grote en complexe databronnen (Wasser, 2020). Vergelijkbaar met standaard file systemen op een OS, HDF5 slaat zijn data op in groepen waarin datasets kunnen worden geplaatst of meer groepen met meer datasets. De meeste data wordt opgeslagen in arrays, hierdoor werkt HFD5 goed met time series data, en is het efficiënter dan SQL datastructuren wanneer het aankomt op het terughalen van tijd gevoelige data.

# Hoe kunnen AI & ML verantwoordelijk worden gebruikt?

## Hoe verantwoord je data gebruik en data transformaties gemaakt door een algorithme?

Voor AI en modules die aanpassingen maken aan data moet er gekeken worden naar zowel Nederlandse en Europese wetgeving. GDPR definieert AI-systemen als programma’s die geen stappenplan volgen maar leren op basis van een data set of voor afgaande uitkomsten. Het gevolg van deze methode is een programma waarbij de ontwikkelaar weinig tot geen kennis heeft van hoe een AI tot een uitkomst komt. Doordat machines geen kennis hebben van fout, ethiek of een moraal waarnemen hebben deze systemen strengere wetgeving nodig. (EU, 2023)

### Systemen en keuzes

Een groot deel van GDPR en black box systemen gaat rondom het gebruik van deze systemen in automatische keuze systemen. Voor systemen waarbij de impact van het resultaat gevolgen kan hebben op wetgeving, en of gevolgen kunnen hebben op levensbeslissingen, wordt er extra regelgeving geplaatst.

Startend met menselijke input: een automatische uitslag moet niet alleen wordt goedgekeurd worden door een gebruiker maar de menselijke input moet betekenisvolle impact hebben op het uiteindelijke resultaat. (EU, 2023)

Toestemming: een automatische beslissing moet toestemming van alle partijen die gevolgen hebben met het resultaat. Binnen GDPR is toestemming niet alleen het klikken van een checkbox. Iedere stap en relevante informatie moet gedeeld worden met de gebruiker.

Helderheid: het beschrijven van hoe een algoritme in elkaar zit is lastig, en het delen van hoe deze in elkaar zit kan gevolgen hebben op de inkomsten van een bedrijf. Daarom is het begrijpelijk dat deze niet gedeeld kunnen worden. Maar het is op het bedrijf om te delen welke data punten gebruikt worden in het systeem en de hoofdfactors te delen van een beslissing.

Verificatie en Testing: Wanneer een beslissing is genomen moet er een aantal stappen kunnen worden ondernomen om het resultaat te verklaren en te testen. Startende met verificatie van het antwoord moet altijd mogelijk zijn door een menselijke gebruiker. Verder moet het resultaat van een systeem beschreven kunnen worden en niet alleen neerkomen op een antwoord. Tot slot moet de gebruikte dataset & resultaat grondig getest worden op discriminerende of foute beslissingen.

Als laatst wordt er een verwachting gesteld dat een “Risico” of “Data protection” assement wordt ondernomen om alle risico’s vast te stellen van een systeem en in kaart te brengen welke data punten beschermd moeten worden onder andere wetgevingen.

# Hoe kunnen AI & ML worden geïmplementeerd binnen een Javaproject?

(Alhamid, 2020) De meest gebruikte Java frameworks voor het implementeren van algoritmes zijn als volgt:

* Weka
* Apache MXNet
* Deeplearning4j
* Mallet
* Spark MLlib
* PyTorch
* MOA
* Tensorflow

Deze frameworks bieden allemaal iets anders op de volgende gebieden: Welke format van data exports accepteren ze, welke algoritmes bevatten ze, hoeveel ondersteuning hebben ze en tot slot bieden ze een python wrapper aan.

## Spring Boot & DJL

(Shapirov, 2020) DJL is een open-source framework voor het verbinden van Java API en ML libraries. De huidige documentatie beschrijft de abstractie laag voor de voor de volgende ML frameworks.

* Tensorflow
* Apache MXNet
* PyTorch

Het voordeel dat DJL brengt is de mogelijkheid om te verwisselen tussen verschillende ML frameworks zonder de codebase aanzienlijk te veranderen. Verder brengt het een voor geteste ML algoritme met zich mee waarbij. Vanwege zijn grote voordelen heeft Spring daarom een starter dependency aangemaakt waarbij, de ontwikkelaars geen persoonlijke beans moeten aanmaken voor dependency injects.

# Testing

Voor dit project wil ik de verschillende libraries over elkaar heen testen, kijken wat de impact is van de verschillende frameworks is op NFR’s en vervolgens deze in productie brengen. Voor data & data opslag ben ik van plan een dataset te creëren op mijn huidige project. De opslag van deze data zal of lokaal in een database worden opgevangen, of in een cloud storage.