# Inleiding Project 3 MIW

Voor project 3 moeten we kortgezegd een app maken die voor medisch personeel en patienten inzicht biedt in de factoren die de gezondheid van patienten beinvloeden en adviezen geeft aan de patienten voor verbetering van de gezondheid. Deze adviezen worden gemaakt op basis van data. Het advies aan de patient verloopt via de arts, de arts is dus de directe gebruiker van de app.

# Onderverdeling Werkzaamheden

## Repository opzetten

Voor een geordende ontwerpproces moet ook een geordende codebase opgezet worden. Er is al een repository op github aangemaakt MakeAIWork3. Daarbinnen wordt een aparte map aangemaakt voor de healthapp.

De repository van de helathapp wordt ingedeeld in de volgende onderdelen.

* Webscraping
* Machine learning/ statistiek.
* App interface.

De codebase wordt gestart met cookiecutter. Deze structuur is gericht op het ontwikkelen van een Machine Learning model.

De code wordt ontwikkelt met een aparte virtuele Python environment. Als basis gebruik Ik de environment die aangemaakt wordt met create\_virtual\_env.sh

## Webscraping

De gezondheidsdata waarop de voorspellingen gemaakt worden staan op een webpagina. We hebben toegang tot deze webpagina opgegeven gekregen, de gegevens moeten nog wel van de pagina afgehaald worden.

1 ) Lezen van de gegevens van de webpagina.

2) Omzetten data voor verdere verwerking.

Er zijn twee opties:

* Experimenten met Beautiful Soup. Met request worden gegevens van een website opgevraagd. Met Beautiful Soup kan aansluitend html code naar een formaat vertaald worden die geschikter is voor verdere verwerking. Om gegevens van een webpagina af te halen heb je naast beautifulsoup nog extra bibliotheken nodig, zoals request.
* Scrapy. Alternatief voor Beautiful Soup is Scrapy. Deze bibliotheek biedt meer functionaliteit om met data van webpagina’s om te gaan.

Extra: Selenium is een bibliotheek voor interactie via webbrowsers. Daarmee kun je ook buttons automatiseren.

Bij testen door andere studenten bleek scrapy ingewikkeld. Binnen de app is in eerste instantie gebruik gemaakt van beautifulsoup.

## Opslag Data, Database

De gegevens van de website worden omgezet geschikt voor verdere verwerking. Na het scrapen van de data van de webpagina moeten bewerkingen uitgevoerd kunnen worden zoals herschikken, selecteren berekenen statistische waarden. Om dit te faciliteren ligt het voor de hand de data te converteren naar een pandas dataframe. Echter in de praktijk komen de gegevens vaak in een database terecht voordat deze verder verwerkt worden. De gegevens van de webpagina worden eerst in een database geplaatst. Aansluitend worden de gegevens uit de database gehaald en omgezet naar en pandas dataframe.

We gaan voor één of beide systemen uitproberen. Voor beide systemen doen we het volgende

* data omzetten van pandas dataframe naar database.
* Toevoegen van nieuwe data aan de database
* Queries maken om data te selecteren

## Machine Learning/ Statistiek

De gegevens uit de database moeten uiteindelijk vertaald worden naar een voorspelling van de gezondheid van een patiënt en een advies hoe deze te verbeteren. Met een ML model kan dit gerealiseerd worden.

De gegevens kunnen uit de database gehaald worden. De data kan verrijkt worden met behulp van statistiek/machine learning.

Eerst moet je beschrijvende statistiek doen op de data. Op deze manier krijg je inzicht in de data, betrouwbaarheid, spreiding. Vervolgens kun je gegevens naast de gegevens van de patient leggen.

* Vertaling van gegevens naar samengestelde gegevens. Bijvoorbeeld uit de lengte
* Keuze model. Gezien de gegevens lijken er twee modellen in aanmerking te komen. Regressiemodel, K-means. Kunnen meerdere modellen gecombineerd worden?

Op basis van metrics wordt bepaald welk model het beste is.

Uit de analyse kan blijken dat bepaalde factoren niet van belang zijn.

## App interface gebruiker

De gegevens uit het model moeten vertaald worden voor de gebruikers. De volgende vragen moet de app beantwoorden:

* Wat wil de gebruiker

Wat is de

* Bevragen machine learning model.
* De app moet uiteindelijk gebruiker kan een app interface maken die
* Aantrekkelijke visualisatie gegevens.

Django is een aantrekkelijk framework.

## Programmeer proces, bron-code- unit testen

Uiteindelijk wordt een app gemaakt waarin alle onderdelen samen komen. De code wordt object georiënteerd opgesteld. De bron code wordt in Python opgesteld.

De app kan in een app worden opgesteld.

Onderdeel van een geordend programmeer proces is de unit test. Voor een nuttige inzet van unit testen is het volgende gebeurd:

* Identificeren essentiële onderdelen app.
* Wat moet er getest worden?
* Stel de essentiele onderdelen.

## App in Docker

In voorafgaande projecten maakten is de code ontwikkeld op een laptop. Op deze manier kan snel code ontwikkeld worden. Een belangrijk nadeel van deze aanpak is dat de code op een andere computer niet direct werkt. Een Docker omgeving kan een

* Bepalen instellingen Docker omgeving
* Installatie script. Met dit script worden
  + Pakketten, bibliotheken geinstalleerd,
  + Omgevingsvariabelen ingesteld/vastgesteld.