MAKE IT WORK

Project 1: gezondheidszorg

Projectverslag Daniël Schene

Vooraf

Het huidige verslag dient enkel ter ondersteuning van de bijbehorende notebooks en comments in de code - daarin staat namelijk grotendeels al beschreven welke stappen er worden ondernomen en waarom.

Per onderdeel van het project zal ik kort reflecteren op het proces, de gedachten achter bepaalde keuzes, en over eventuele verbeterpunten voor een volgende project.

Exploratory Data Analysis / Verkennen van de dataset

Voor de EDA heb ik grotendeels vastgehouden aan het ‘standaard’ proces, waarbij ik wel moet vermelden dat mij dit bij uitstek een onderdeel lijkt waarvoor je in de loop van de tijd, en naarmate de ervaring zich opstapelt, een steeds duidelijkere eigen routine vormt. Het was daarom nog een beetje tasten in het duister in het begin, maar ik heb mijn stappen zo duidelijk mogelijk proberen te beschrijven in het notebook.

Het was vooral zoeken naar een goede volgorde; het proberen te scheiden van data ‘schoonmaken’ en vervolgens verkennen leek mij in eerste instantie een goed streven, maar inmiddels ben ik er wel achter dat deze onderdelen eigenlijk continu overlappen, elkaar aanvullen en afwisselen. Hierdoor is het notebook misschien hier en daar wat chaotisch, juist ook omdat het vaak ook zoeken was naar een logische volgende stap. Voor het volgende project heb ik mij voorgenomen om wat meer structuur aan te brengen in de EDA; door alle stappen direct van commentaar te voorzien (ook eigen creaties zijn achteraf soms lastig te snappen) en door vooraf een lijstje te maken met welke onderdelen er absoluut in moeten, en welke onderdelen optioneel zijn (en waar ik dus in eerste instantie minder aandacht aan hoef te begeven).

Data pipeline / feature engineering / data verwerken

Het opzetten van de pipeline was vooral een kwestie van het goed vooraf bepalen van de gewenste stappen, en deze vervolgens implementeren. Waar ik dit in eerste instantie met csv bestanden deed, heb ik uiteindelijk een goede pipeline weten te maken die leest/schrijft in de database waarmee verbinding wordt gemaakt. Dit maakt het net even realistischer en komt al dichter in de buurt bij hoe het ‘in het echt’ zou kunnen gaan.

Regressie / bouw van het model

Het trainen van het model op verschillende subsets van de data heb ik geprobeerd om zo ‘dynamisch’ mogelijk te doen. In het geval van dit project hield dat in dat ik een programma wou maken wat voldeed aan de volgende punten:

- Het programma moest eenvoudig meerdere regressiemodellen kunnen trainen op verschillende subsets van de data en deze opslaan.

- Het programma moest zelf uit de verschillende getrainde modellen kiezen welke versie hiervan het beste scoorde, en deze, inclusief alle variabelen van het model, exporteren als dictionary zodat de ‘interface’ deze kon ophalen.
- De modellen moesten naast de standaard evaluatie metrics zoals de r-squared en msqe ook de ranges waarbinnen gewerkt wordt bevatten, evenals de uiteindelijke coëfficiënten van de variabelen (en eenheden waarmee gewerkt wordt). Op die manier krijgen we voor het model een dictionary die alle informatie bevat voor de interface om mee te kunnen werken.

Er zijn twee punten waarover ik achteraf nog niet tevreden ben. Het eerste punt is het werken in steeds hetzelfde pandas dataframe (veel ‘inplace’ gedaan), terwijl ik idealiter per mutatie of wijziging een nieuwe versie van het dataframe zou willen gebruiken. Dit maakt terugkijken ook een stuk makkelijker, omdat er een soort versiegeschiedenis ontstaat van de aanpassingen die er gemaakt zijn. Daarnaast behoud je ten alle tijden je originele dataframe als een soort back-up, in plaats van dat deze telkens allerlei wijzigingen ondergaat.

Het tweede verbeterpunt heeft te maken met het vaststellen van de bandbreedtes waarbinnen de variabelen mogen vallen. Om eerlijk te zijn heb ik dit voor het huidige project enigszins willekeurig gekozen. Ik heb vooral naar de dataset gekeken, en naar de waardes die er per variabel in de data voorkwamen. Aan de hand hiervan heb ik per variabel de grenzen gekozen. Hierbij heb ik het ontwerp vooral zo willen maken dat er, mocht er op een gegeven moment een hele nieuwe dataset komen, er in ieder geval wordt gefilterd op waardes die ver buiten de ranges van de eerste dataset vallen.

Interface / eindproduct

De laatste stap, het maken van een interface, heb ik (bewust) minder prioriteit gegeven dan de andere onderdelen. Voor dit project heb ik mijzelf aan het begin als doel gesteld om in ieder geval alles van het MVP goed te snappen, en in dat opzicht vond ik een solide en werkend basisprogramma belangrijker dan een mooie interface. Hierdoor heb ik eerst alles in Python proberen te doorlopen en implementeren, zoals het verkrijgen van input van de gebruikers, het evalueren hiervan, eventuele berichten aan de gebruiker, etc. Door dit eerst allemaal werkend te krijgen in Python heb ik na het testen hiervan besloten om voor nu niet verder in de wereld van de GUIs te duiken. Deels omdat ik hier geen tijd en/of prioriteit voor had, maar ook omdat de meer front-end-gerichte tools mij momenteel minder enthousiasmeren dan de achterliggende werking, en omdat ook het vaardiger worden in Python op dit moment de juiste bezigheid lijkt voor mij t.o.v. het leren van meer interface-gerichte tools of talen.

Qua code heb ik ook hier zoveel mogelijk geprobeerd om de boel dynamisch en flexibel te ontwerpen. Ik begon met het vragen van alle input in verschillende if-else statements, maar kwam er al snel achter dat het bij het steeds herhalen van dezelfde stap (of variatie daarop) beter is om te kijken of dit op te lossen is met een functie. Het programma moest in dat opzicht ook voldoen aan een aantal eisen:

- Het programma moet kunnen werken met modellen die op verschillende subsets van de data zijn getraind. Als bijvoorbeeld blijkt dat het beste model enkel ‘smoking’ en

‘exercise’ als parameters bevat, moet het programma niet ook de andere variabelen verwachten en/of opvragen. Op deze manier blijft het model efficiënter werken en werkt het enkel met de relevante parameters.

- Het opvragen van de gebruikersgegevens moet als volgt werken:
    - Er wordt geïtereerd over alle variabelen waarmee het model getraind is
    - Alle input wordt geëvalueerd op ‘juistheid’, de gebruiker kan maximaal twee keer een ongeldige waarde invoeren.
    - De bandbreedtes waarbinnen de input mag vallen worden dynamisch opgehaald en hoeven dus niet apart te worden ingesteld.
    - De gebruiker krijgt duidelijke instructies van het programma

Waar het in mijn optiek nog niet helemaal goed is gegaan is met het zorgen voor werkende shell scripts om zo in één keer de modellen te updaten/te trainen en apart de app op te starten. Het is sowieso niet ideaal om het programma te moeten openen (via het juiste pad) in de CLI/terminal. Uiteraard is dit niet de manier waarop een eindgebruiker dit programma zou gebruiken. Om de redenen die ik hierboven heb genoemd is dit in het project echter van ondergeschikt belang geweest. Voor het volgende project wil ik hier meer aandacht aan besteden als dit kan.