Health Marketing

Beschrijving projectopdracht

Begrippenlijst

Health Marketing – Het gebruik van markt-analyse en -beinvloedings technieken ter verbetering van de gezondheid van bevolkingsgroepen. Dit is dus niet hetzelfde als reclame maken voor medicijnen of placebo's.

Regressie-analyse – Een wiskundige methode om samenhang te ontdekken in reeksen cijfers betreffende waarnemingen, en op basis hiervan voorspellingen te doen.

Outlier (uitschieter) – Een waarneming die sterk van de andere, vergelijkbare waarnemingen afwijkt. Dit hoeft geen foute waarneming te zijn.

Lineaire regressie – Een te achterhalen verband tussen een aantal variabelen van de vorm

$$X\,\beta = y \qquad (1)$$

met X een gegeven matrix waarvan de rijen kwantitatieve waarnemingen van mogelijk samenhangende gebeurtenissen voorstellen, y een gegeven constante vector en β een onbekende vector van coefficienten, allen over het grondlichaam R, dus bestaande uit reele getallen.

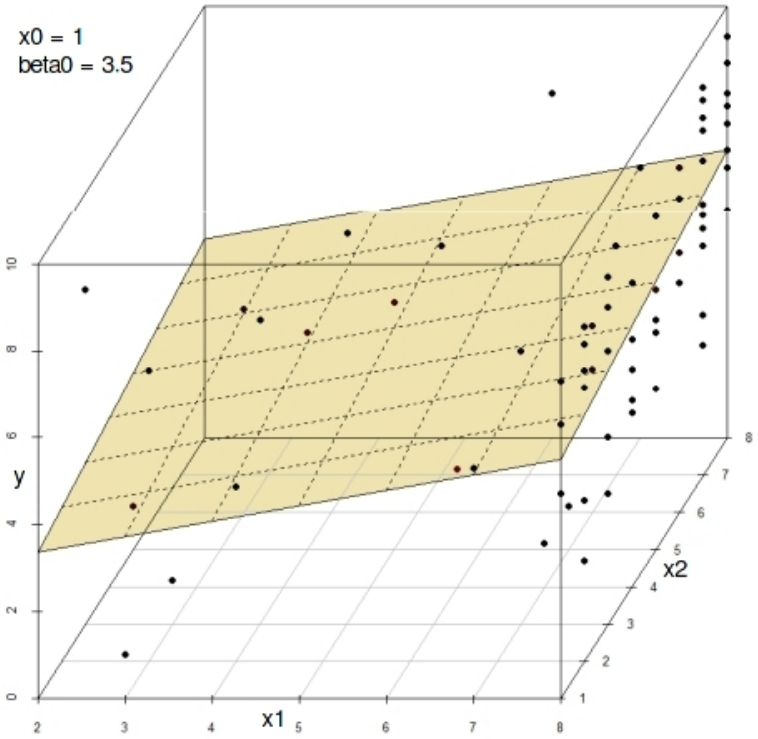
Indien het aantal rijen, dus waarnemingen, groter is dan het aantal coefficienten in β, dan is geen β meer te vinden die aan alle waarnemingen voldoet, tenzij de vergelijkingen die met (1) worden gedefinieerd afhankelijk zijn. Deze vergelijkingen zijn in praktische gevallen vrijwel altijd strijdig. Echter valt er dan nog wel een β' te berekenen waarvoor (1) zo goed mogelijk "klopt". Dit houdt in dat $X\,\beta' = y'$ met $|y' - y|$ minimaal.

Na enig rekenwerk blijkt dat in dit geval

$$\beta' = (X^T X)^{-1} X^T y \qquad (2)$$

De zinsnede "na enig rekenwerk" is een traditionele manier om in een wetenschappelijk artikel aan te geven dat de auteurs een berekening best moeilijk vinden, maar dat je alleen bij het clubje ingewijden hoort als je òf het grapje kent òf de berekening kunt maken. Engelstalige variant: "The proof is left as an exercise to the reader". Het bewijs van (2) speelt bij dit project geen rol, de toepassing ervan wel.

Indien β' uit (2) berekend is, kunnen nieuwe waarnemingen (rijen van matrix X) worden toegevoegd, waarmee nieuwe componenten van y kunnen worden voorspeld. Als er inderdaad sprake is van samenhang tussen de elementen van X en het "resultaat" y dan zijn die nieuwe componenten van y zinnige voorspellingen, gegeven de nieuwe waarnemingen.



Indien genoemde samenhang bestaat, geldt over het algemeen: hoe meer waarnemingen zijn gebruikt ter berekening van *β'*, des te beter kunnen bij nieuwe waarnemingen de nieuwe componenten van *y* worden voorspelt. Echter als zo'n samenhang niet bestaat, is zo'n voorspelling uiteraard niet mogelijk, hoeveel waarnemingen ook worden gedaan.

Een rode vlag die op deze situatie duidt is de ontdekking dat het minimum van $|y' - y|$ een relatief groot getal is, dat bovendien sterk fluctueert als willekeurige regels van (1) worden weggelaten. Er valt dan zogezegd aan de waarnemingen geen touw vast te knopen, m.a.w. ze lijken willekeurig en zijn dat waarschijnlijk ook.

Een bekend voorbeeld hiervan is het bij tijd en wijle irrationele gedrag van beleggers. Het ontdekken van "trends" hierin, met als doel de beurskoersen te voorspellen, heeft soms veel weg van koffiedikkijkerij, gebruikmakend van aan de intuitie appelerende termen als "steun" en "weerstand". Deze termen ontmoeten bij onervaren beleggers weining weerstand maar geven helaas ook niet veel steun.

Een andere situatie is dat er misleidende waarnemingen in de dataset zijn opgenomen. Soms zijn er outliers die dit doen vermoeden, maar ook waarnemingen in het "normale" bereik kunnen de zaak scheef trekken ("bias").

Een veel gemaakte fout is dat samenhang wordt geinterpreteerd als een causale (oorzaak en gevolg) relatie. Twee variabelen *a* en *b* kunnen sterk samenhangen zonder dat de onderliggende verschijnselen *A* en *B* een causale relatie hebben. Veel voorkomend geval is dat er een derde verschijnsel *C* is dat *A* en *B* veroorzaakt.

Grijs haar wordt niet veroorzaakt door rimpels en rimpels ook niet door grijs haar. Beiden worden meestal veroorzaakt door

veroudering.

Een lineaire verband tussen variabelen is met behulp van (2) eenvoudig te achterhalen. Echter zijn veel in de niet-wiskundige werkelijkheid bestaande verbanden niet-lineair. Het komt er dan vaak opaan één of meer van de oorspronkelijke variabelen te vervangen door variabelen die hiervan een functie zijn en die onderling wel een lineair verband hebben. Zo bestaat tussen de snelheid v en de hoeveelheid bewegingsenergie E van een auto geen lineair verband, maar als v vervangen wordt door $\sqrt{w}$ (substitutie) bestaat tussen w en E wel een lineair verband.

Om op het idee te komen van een dergelijke substitutie is het van belang kennis te hebben over het toepassingsgebied (context) van de betreffende regressie. Dat geldt ook voor deze project-opdracht. Met dat doel is zijn een klein aantal wetenschappelijke artikelen over die context toegevoegd.

Web scraping – Het zonder menselijke tussenkomst door een computerprogramma verzamelen van informatie vanaf het World Wide Web. De informatie kan in statische of dynamisch gegenereerde vorm op de webpagina aanwezig zijn. Daarnaast valt onderscheid te maken tussen web scraping met goedvinden van de eigenaar van de betreffende site en web scraping tegen de wens of het recht van die eigenaar.

Statische webdata – Gegevens op een webpagina die in hun uiteindelijke vorm door de server worden gegenereerd en verzonden. Zulke data is meestal eenvoudig herkennen indien men in een browser de “page source” (broncode van de pagina) bekijkt. Statische webdata is eenvoudig te scrapen, bijvoorbeeld met de Pyton library BeautifulSoup.

Dynamisch gegenereerde webdata – Gegevens waarvan de uiteindelijke vorm door de client (browser) wordt gegenereerd, vaak door middel van programma-code in de talen JavaScript of WebAssembly. Ook dynamisch gegeneerde webdata is te scrapen. Hiervoor gebruikt men een headless browsing. De basis voor generatie van zulke gegevens is wel van de server afkomstig en kan bijvoorbeeld de vorm hebben van een algoritme of van model-parameters.

Model-parameters – Getallen die het gedrag van een wiskundig model bepalen.

Wiskundig model – Een verzameling wiskundige relaties die zich gedragen analoog aan een natuurlijk verschijnsel. Een wiskundig model kan “tot leven worden gebracht” in de vorm van een computer-simulatie.

Headless browsing – Dit is niet het eindeloos kattenfilmpjes kijken op YouTube, maar het met behulp van een ander computerprogramma aansturen van een browser. Die browser wordt dan opgestart in de z.g.n. “headless mode”, d.w.z. zonder user interface. Een Python library die browsers op die manier kan aansturen is Selenium.

Web service – Een gegevensverwerkende server-toepassing die niet voor rechtstreeks gebruik door een browser bedoeld is, maar z’n gegevens wel via een URL beschikbaar maakt of van een client verkrijgt.

Web app – Een client die bedoeld is voor samenwerking met een web-services.

Micro services – Een verzameling web services die tevens client bij elkaar zijn en gezamelijk een

flexibele maar niet persee efficiente of veilige applicatie vormen. De toegenomen flexibiliteit maakt micro services populair en de beveiliging wordt geleidelijk aan beter, hoewel er, zeker bij overheidsinstanties, nog flinke gaten in zitten. Gelukkig is niet algemeen bekend waar (security through obscurity). Een groep bij elkaar horende micro services doet zich aan een browser client voor als een eenheid.

URL (Uniform Resource Locator) – Een eenduidige, meestal voor menselijke interpretatie geschikte identificatie van een Internet adres, vaak in de vorm van een woord dat een bedrijfsnaam, begrip of activiteit aanduidt. Veel nieuwere URL's zijn neologismen, nieuwe woorden die hun eerste betekenis krijgen door de activiteit waar de betreffende URL toegang toe verleent. Begrippen zoals "googlen", "youtuber" en "appen" zijn afgeleid uit zulke URL's. URL's worden door z.g.n nameservers omgezet in IP adressen. Dit zijn getallenreeksen die door routers worden gebruikt om informatie de juiste weg te laten volgen. In URL's bestaat een levendige handel.

REST (Representational State Transfer – Een manier door uitbreiding van een URL met aaneengeschakelde keywords, elk voorafgegaan door "/" informatie binnen de aan die URL gekoppelde website eenduidig toegankelijk te maken.

Voorbeeld: *https://www.bol.com/nl/nl/m/zakendoen-met-bolcom*

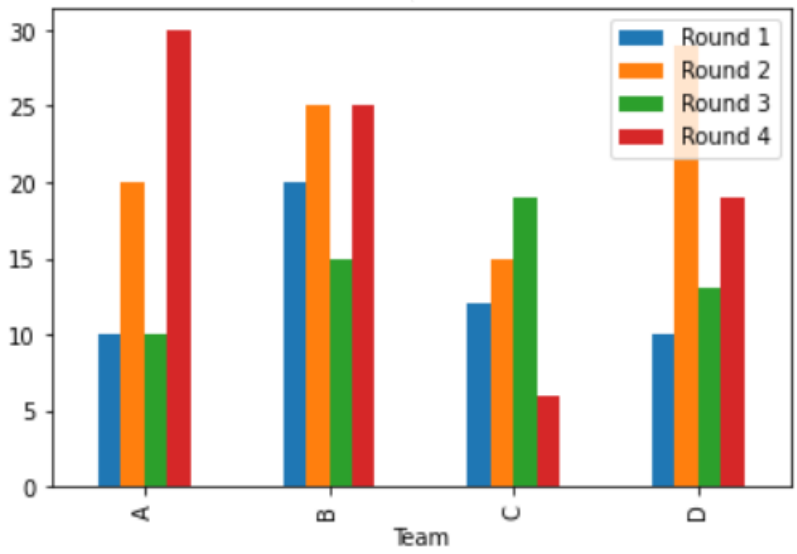
De benaming Representational State Transfer suggereert een diepzinnig onderliggend concept. Over wat dat concept precies is, bestaan echter zeer uiteenlopende opvattingen. Bij het lezen hierover is het goed de uitspraak van filosoof Ludwig Wittgenstein met een geel plakbriefje op de rand van je beeldscherm te plakken: "Wat gezegd kan worden, kan duidelijk worden gezegd. Wat niet gezegd kan worden, daarover kan men beter zwijgen."

Een pretentieloze uitleg luidt: Een webserver houdt geen state bij (weet niet waar hij gebleven is), omdat dat dat, bij het te woord staan van veel clients "tegelijkertijd", traditioneel als inefficient wordt beschouwd, vooral als die clients cookies weigeren. Door de state (toestand) in de URL te representeren wordt deze last afgeschoven op de client. Daardoor kunnen zeer veel clients te woord worden gestaan met beperkte servercapaciteit.

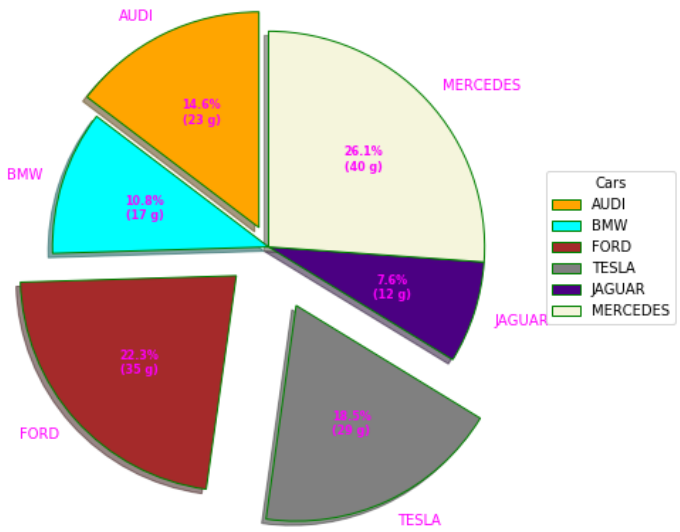
Ondanks de zinloze dikdoenerij rondom deze term is REST zeer nuttig, al was het alleen maar omdat je toegangspaden tot gegevens in je browser kunt bookmarken en de backbutton op de browser het gewoon doet, wat lang niet bij alle browser-applicaties het geval is.

Cookie – Een klein stukje informatie dat een webserver opslaat op de client (in de browser). Het kan gaan om bijv. inlog-gegevens, maar ook om tracking cookies, die bedrijven in staat stellen gebruikers te stalken en bijvoorbeeld na eenmalige aanschaf van een zak paprikachips via het web wekenlang in elk on-line tijdschrift dat de betreffende gebruiker leest, advertenties van paprikachips in te voegen.

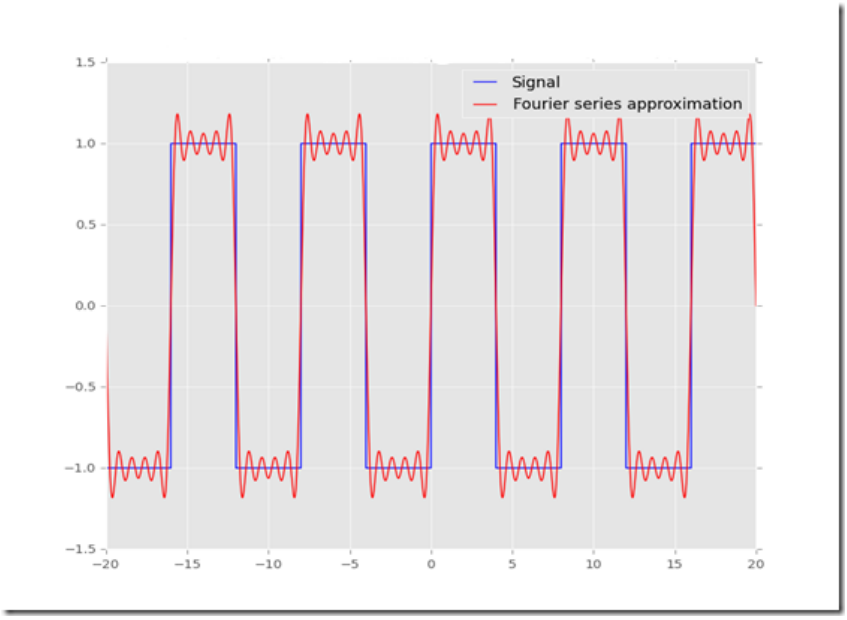
Bar chart (Staafdiagram) – Een plaatje zoals onderstaand, geliefd bij statistici, dat met de Python library MatPlotLib kan worden getekend:



Pie chart (Taart diagram) – Een plaatje zoals onderstaand, geliefd bij managers, dat met de Python library MatPlotLib kan worden getekend:



Line chart (Lijn-grafiek of kortweg grafiek) – Een plaatje zoals onderstaand, geliefd bij technici, dat met de Python library MatPlotLib kan worden getekend:



MDR (Medical Devices Regulation) – Een door de EMA opgesteld pakket van regels met betrekking tot medische en paramedische apparatuur en hiermee samenhangende software. De MDR is van groot belang als aansporing tot professionalisering van het ontwikkelen van software op het terrein van de gezondheidszorg. Dit is een terrein waarop het falen van programmatuur niet met een disclaimer of met schouderophalen kan worden afgedaan. De hoogte van de lat wordt met de MDR duidelijker. Hoe er beheerst en reproduceerbaar overheen te springen blijft vooralsnog zeer problematisch. De oplossing ligt vooral in “ontworpen kwaliteit”. Kwaliteitsborings-procedures zetten hier toe aan, toch ligt hier vooral een gedeelde verantwoordelijkheid voor de ontwerpers zelf. Het dragen van deze gedeelde verantwoordelijkheid vereist, naast technische kennis en persoonlijke ethiek, systematische intervisie en een open bedrijfscultuur.

EMA (European Medicines Agency) – Een europees regelgevend orgaan op het terrein van medische wetenschap, gezondheidszorg, medicatie en medisch handelen.

FDA (Food and Drugs Administration) – De amerikaanse tegenhanger van de EMA.

Bare bones – Kaal, zonder speciale voorzieningen of een mooi “jasje”.

Dedicated – Toegesneden op een speciale taak.

POC (Proof Of Concept) – Een systeem (bijvoorbeeld computerprogramma) dat bedoeld is om de haalbaarheid en effectiviteit van een bepaald idee aan te tonen. Uitdaging hierbij is het POC overzichtelijk en flexibel te houden (dit laatste is nodig omdat vaak verschillende varianten moeten worden uitgeprobeerd), zonder de tijdsinvestering die bij een productie-klaar systeem hoort wat betreft user interfacing, foutafhandeling en certificerings-aspecten. Van belang is vooral, het POC goed doordacht en kernachtig op te zetten, indachtig de wijsheid: “Niets is definitiever dan een tijdelijke oplossing”. Toch is het maken van een POC zeer de moeite waard. User interfacing, foutafhandeling en certificering zijn vaak samen goed voor 95% van de ontwikkelkosten.

View tree – Een hierarchische verzameling op het scherm zichtbare vensters en bedieningsorganen zoals knopjes, schuifjes, edit-windows en tree views. Omdat dit deel van de applicatie zichtbaar is, wordt het vaak als eerste gemaakt en daarna wordt de werkzame code eraan gehangen (event handlers).

Deze manier van coderen werkt in het begin makkelijk en redelijk snel, maar leidt tot een instabiele structuur voor elke applicatie die niet puur uit een Front End bestaat, en waarvoor dus niet geldt WYSIAYG (What You See Is All You Get).

Front end – Het voor de gebruiker zichtbare, meestal zeer veranderlijke deel van een applicatie, vaak draaiend in een browser of als web app op een mobiele telefoon. Veelal ongeschikt als stabiele basis voor een serieuze, duurzame applicatie.

Back end – Het voor de gebruiker onzichtbare deel van een applicatie, vaak draaiend op de server. Dit is de motor van de applicatie en meestal betrekkelijk stabiel vergeleken bij het front end. In toenemende mate draait een deel van die motor echter ook op de client, daarmee de server ontlastend (load balancing).

Full stack – Een applicatie die bestaat uit zowel één of meer front-ends (views) als een back-end, met binnen het back-end vaak een uitsplitsing in controllers (berekeningen en logica) en models (gegevensstructuren).

MVC (Model View Controller) – Een design pattern waarbij een applicatie wordt verdeeld in models, views en controllers zoals beschreven bij *full stack*.

Design Pattern (Ontwerp-patroon) – Een standaard principe-oplossing voor een standaard categorie problemen, vooral bij het ontwerpen van software. Design patterns verschaffen ontwikkelaars een gemeenschappelijke taal en een schatkist met ideeen en bewezen best practices. Ze kunnen zelden letterlijk worden toegepast maar omvatten gecodifeerde (eenduidig omschreven) ervaring van andere ontwerpers.

Design patterns hebben nooit absolute, dogmatische geldigheid maar zijn desondanks van grote waarde. Ze moeten er op den duur toe leiden dat ook bij software niet voortdurend het wiel wordt uitgevonden.

Doel

Het doel van deze opdracht is, practische ervaring op te doen met:

1. Verzamelen van gegevens vanaf een dynamisch gegenereerde webpagina.
2. Prepareren (converteren, filteren (“schoonmaken”), completeren, uniformeren) van die gegevens.
3. Transformeren van die gegevens naar een vorm die zich leent voor lineaire regressie, waarbij de levensverwachting in jaren wordt voorspeld uit aanleg en levensstijl.
4. Uitvoeren van zo’n regressie, eerst met “bare bones” NumPy, daarna met een dedicated regressor uit SciKitLearn.
5. Grafisch representeren van de resultaten met behulp van de Python library MatPlotLib. Hierbij kunnen vooral bar charts en pie charts van pas komen. Line charts liggen minder voor de hand maar kunnen een nuttige aanvulling zijn.
6. Opbouw van een POC uit modules die apart te testen zijn. Het gebruik van classes is hierbij een optie.
7. Centraal staat ontwikkeling van een representatief, overtuigend POC zonder de rompslomp die komt kijken bij het productierijp maken.

Meer dan de vorige project-opdrachten is deze opdracht tevens een ontwerp-opdracht. Het gaat erom verstandige afwegingen te maken tussen kosten en baten. Hoe kan de haalbaarheid van de requirements worden aangetoond zonder nodeloze kosten, maar ook zonder alles twee keer te doen, eerst slecht en daarna goed.

Van goede code wordt ook bij het maken van een POC al geprofiteerd. Je creeert voor jezelf een ordelijke werkomgeving, het kenmerk van een professional en bewezen preventief ten aanzien van uitglijders. Documenteer dus van meet af aan effectief en bondig in je code (Literate Programming) en schrijf je code systematisch en met een heldere, flexibele structuur en nauwkeurig naamgeving. Ga ervanuit dat de basiselementen van data-manipulatie en berekeningen uiteindelijk zonder veel wijzigingen in de structuur van de uiteindelijke productiecode belanden, zoals ballen in een kerstboom.

Data-presentatie is bij het POC secundair. Dit aspect moet wel goed genoeg zijn voor een oordeel over de haalbaarheid van de concepten achter het POC, maar het is bijna zeker dat stakeholders hier nog met flinke modificaties komen. Daarom dient de structuur van de applicatie te worden bepaald door de basiselementen van data-manipulatie en berekeningen (meestal back-end), niet door het user interface (front-end). Hier bestaat een populaire kreet voor: “Don’t hang your application into the view tree”.

Inhoud

Gegeven is een website (op dit moment *https://wiztech.nl/mitw/mcr/data_generator.html*) met daarop

dynamisch gegenereerde gegevens betreffende levensstijl en levensverwachting.

Opdracht is een modulaire, waar nuttig objectgeorienteerde applicatie te schrijven die voldoet aan de 7 punten die opgesomd zijn bij de paragraaf *Doel*.

Tevens dient code te worden geschreven om de betreffende modules onafhankelijk te testen. Validatie van het geheel (in tegenstelling tot testen van de delen) vindt plaats door een docent, die de uit de regressie berekende parameters vergelijkt met het model dat aan de genoemde website ten grondslag ligt.

Besteed ook aandacht aan de wijze van presenteren. Hierbij geldt: Form Follows Function, de vorm staat ten dienste van datgene waar de applicatie voor bedoeld is, namelijk inzicht in de samenhang tussen leefwijze en levensverwachting. Diagrammen hoeven niet “gelikt” of imponerend te zijn, maar doeltreffendheid, eenduidigheid en leesbaarheid is wel van belang. Diagram-keuze en kleurgebruik dienen dit doel.

Werkwijze

Installeer de betrokken libraries.

Het gaat om SciKitLearn, Selenium en MatPlotLib, Google is your friend.

Experimenteer wat met voor jou onbekende libraries en met headless browsing. Doel is door te krijgen wat de mogelijheden zijn voor je je op het ontwerp stort.

Stel een projectplan op. De inhoud hiervan is beschreven bij het autonome voertuig project.

Maak een begin met de projectevaluatie. De inhoud hiervan is beschreven bij het autonome voertuig project.

Schrijf de broncode – Deel je code in in losse, apart testbare programma’s of modules voor tenminste webscraping, data preparatie, regressie analyse met en zonder SciKitLearn en grafische presentatie van de resultaten. Schrijf ook een verbindende “main module” en voor elke module een unit test.

Valideer de broncode – Voer de tests uit zoals beschreven in de testspecificaties en verwerk de resultaten in een puntsgewijs, kwantitatief georienteerd testrapport. Vermeldt eventuele substituties van regressie-kenmerken en de waarde van de geschatte parameters van het resulterende lineaire model (de β vector).

Voltooi de projectevaluatie – Zie het autonome voertuig project.

Middelen

- Zelfde als voorheen.

Producten

- Projectplan
- Broncode
- Testrapport
- Projectevaluatie
- Overige documenten die je zelf als relevant beschouwt

Toetsing

Formatieve toetsing vind plaats tijdens gesprekken in de les en indien nuttig tijdens een afrondend gesprek. Het gaat daarbij om het geven van feedback en adviezen door de docenten, niet om beoordeling.

Bijzondere aandacht wordt in dit project besteed aan de manier waarop je je broncode in onafhankelijk testbare onderdelen hebt verdeeld. Dit kunnen modules of classes zijn, of allebei.

Daarnaast komen juist in dit project metafunctionele aspecten aan de orde. Gezondheidszorg neemt hierbij een bijzondere plaats in. Ethische vragen rond privacy, betaalbaarheid, beschikking over eigen leven, maar ook gedifferentieerde verzekeringspremies zijn actueel.

Inventariseer voor jezelf hoe je tegen dit soort zaken aankijkt. Er is geen “voorgebakken” goed antwoord. Het is echter wel een maatschappelijke discussie waar je als ontwikkelaar mee te maken krijgt, actief of passief. Hoever vind je dat jouw verantwoordelijkheid daarin gaat? Vroeg of laat ben je behalve ontwikkelaar mogelijk ook consument. Hoe vind jij dat deze zaken geregeld moeten zijn en

waarom?

Een laatste aspect is de regelgeving. Denk van te voren eens na over testbaarheid en betrouwbaarheid. Zijn er zaken die je hier in een professionele omgeving anders wilt doen dan bij het maken van bijvoorbeeld een computer-game? Welke? Hoe zou dat in het ontwikkelproces tot uiting kunnen komen, o.a. wat betreft samenwerking met collega's?

--//--