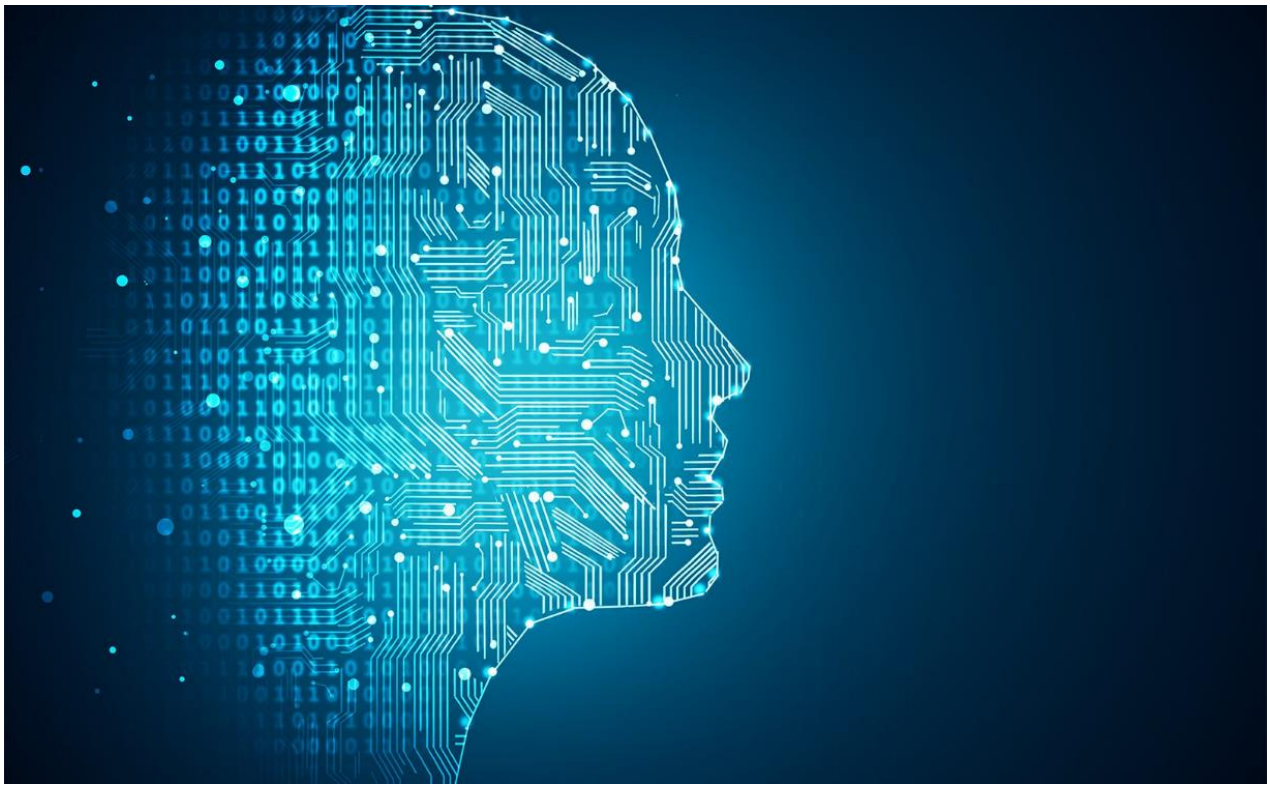
Adviesrapport DBB-project



Door: Guy Veenhof, Koen van Heertum en Ruben van Raaij

Docent: Joost Vanstreels

Datum: 10/26/2020

Inhoudsopgave

Overzicht stakeholders en doelstelling

Stakeholders en daarbij wat die stakeholders qua input hebben/ wat ze met die informatie doen.

In het process komen deze mensen aanbod:

Machinist, de machinist onderzoekt op een globale manier wat en waar het probleem zich bevindt.

Meldkamer, de meldkamer krijgt de globale probleem informatie van de machinist. Zij sturen een aannemer naar locatie om kritische informatie te krijgen. Deze vult wat gegevens aan in het systeem zodat de meldkamer een geschatte reparatietijd kan krijgen. Die informatie geven ze door aan de reisplanner.

Aannemer, de aannemer krijgt globaal de informatie van het probleem door, zoals het probleemgebied en waarschijnlijke oorzaak. Vervolgens gaat deze zelf op weg naar het probleemgebied om daar zelf een conclusie te trekken en het probleem op te lossen.

Reisplanner, de reisplanners krijgen de geschatte storing/reparatietijd door en visualiseert dit in de reisapp, om reizigers informatie te geven over de vertragingen.

Reizigers, de reizigers zijn de getroffenen. Zij zullen een omweg moeten krijgen van de reisplanners als het blijkt dat de treinen te lang vertraagd zijn.

Treinplanner, de treinplanner zorgt ervoor dat de treinen een omweg kunnen nemen zodra er een probleem/ongeval gebeurd is. Ze krijgen de geschatte eindtijd van het probleem via de reisplanners. Hun taak is de goede doorstroming van de treinen zodat de reizigers niet te veel last hebben van het probleem.

Doelstellingen opschrijven van het project en wat de doelstellingen zijn voor ProRail.

De doelstelling van ons project is het verkrijgen van een betrouwbare reparatietijd met zo min mogelijk features/inputs/kritische informatie, om reizigers zelf in te laten schatten of ze een omweg nemen of wachten op de volgende trein.

Gewenste situatie

https://ibb.co/LJNfG6B (Nieuwe proces)

https://ibb.co/Y3yBGtc (Oude proces)

In het oude proces is te zien dat de (pessimistische) prognose van de aannemer de reparatietijd bepaald. In het nieuwe proces is daarentegen een tweede reparatietijd berekend, namelijk van het systeem. Dit systeem gaat dan naar de vorige meldingen kijken. Ondanks de geschiedenis van vele meldingen heeft het systeem toch een lage betrouwbaarheid. Dus is het voor de meldkamer toch van noodzaak om een overweging te maken tussen deze 2 reparatietijden, want het is altijd mogelijk dat de aannemer vele onverwachte problemen kan krijgen, door bijvoorbeeld bereikbaarheid of arbeidsziektes. Deze kan het systeem niet inschatten.

Requirements oplossing

1. Een machine learning model waarmee we kunnen voorspellen hoe lang een bepaald probleem gaat duren voordat het opgelost is.

2. Een applicatie die de data op een duidelijke manier kan weergeven aan de reisplanners van NS, zodat deze de reizigers kunnen updaten hoe lang ze verwachten dat een bepaald probleem gaande blijft.
3. De code moet kunnen runnen en ingeleverd worden in een Jupyter Notebook.
4. De computer moet de python, sklearn, pandas en numpy packages geinstalleerd hebben.

ML model

Fases CRISP-DM

TODO: Uitleg over hoe we het hebben opgesteld in de jupyter notebooks

Business understanding: Door middel van het data-woordenboek hebben we gekeken welke data een tijd inhoudt en features die een goed verband hebben met de reparatietijd. Het liefst zo min mogelijk, zodat de gebruiker gemakkelijker en sneller de reparatietijd kan krijgen.
Data understanding: door de verschillende kolommen uit de data te halen kwamen we al snel achter dat met: 'stm_aanntpl_tijd' en 'stm_fh_ddt' de reparatietijd kolom aangemaakt kon worden. De types moesten nog wel aangepast worden. We keken naar hoe ver de getallen van het gemiddelde van de reparatietijd liggen door naar de RMSE score te kijken, die stond namelijk op bijna 300. We wisten dat we nog wat aan data cleaning moesten doen.
Data Preparation: We begonnen met de dubbele waardes eruit te halen met behulp van de kolom meldingsnummer. De gekozen features zijn: Prioriteit en oorzaakcode. Daarvan hebben we alle NaN's weggehaald en zo nog genoeg meldingen overgehouden. Ook hebben we de outliers eruit gehaald, eerst door simpelweg de sterke outliers eruit te halen. Vervolgens hebben we, na enige gesprekken met de productowner, alle lage reparatietijden ook uit de dataset gehaald. Alles onder de 4 minuten kan namelijk gezien worden als een melding i.p.v. een storing. In ons geval zijn alle waardes dus boven de 4 minuten en onder de 480 minuten. Het baseline model geeft ons nu een verbeterde RSME-waarde van 78 i.p.v. 300.
Modeling: We zijn als allereerst begonnen met het maken van een baseline model. "De reparatietijd is altijd gelijk aan de gemiddelde reparatietijd". Zonder data preparation hadden we een RMSE-score van 300. Met de data preparation wordt dat dus 78. Dat kan toch iets beter. Met het juiste model kan deze RMSE-score verlaagd worden. Linear regressie werd voor ons het eerst geteste model. Maar die viel slecht tegen met een RSME-score van 77. Een lastiger model was het DecisionTree (Classifier) model. Dit model kwam helaas ook niet ver. We kregen advies om over te gaan naar Random Forest Regressor. Dit daarentegen was wel een verbetering van 74. We gingen daarop verder werken. Al kwamen we er later achter dat het niet (of simpelweg bijna niet) mogelijk was om de betrouwbaarheid ervan te berekenen. We wisten wel dat Regressor modellen ons een goeie waarde gaven. Uiteindelijk kwamen we tot de conclusie dat DecisionTree Regressor het beste was, ondanks het een 0.4 hogere RMSE-score had. Daarmee konden we met wat gepriegel de betrouwbaarheid berekenen en samen met de reparatietijd visualiseren op de applicatie

Proof of concept

Voor de gebruiker is het veranderen van code in jupyter notebook niet handig. Daarom is er een applicatie gemaakt om de gebruiker het makkelijk te maken. Deze applicatie vraagt alleen de inputs, om de reparatietijd te berekenen. De inputs zijn: oorzaakcode en prioriteit. Wanneer beide zijn

ingevoert, verschijnt er een reparatietijd en een betrouwbaarheid. Deze gegevens kunnen niet simpelweg met een knop opgeslagen worden, maar als je de meldingsnummer handmatig invoert dan kan je een screenshot van de applicatie maken en doorsturen naar de desbetreffende persoon. Zo kan de volgende persoon ook weten dat het om bijvoorbeeld melding 788 gaat. Het meldingsnummer is niet noodzakelijk om het systeem te laten runnen, maar meer om specificaties op te vangen van de melding zelf.

Validatie van oplossing

Deze validatieset geeft de volgende inputs en het systeem de volgende outputs:

(Getest op 6:47PM 5-11-2020)

Melding 1 (160488): 2.0(Prioriteit) & 148.0(Oorzaakcode) geeft 31 min met bh: 1.5

Melding 2 (192813): 2.0(Prioriteit) & 218.0(Oorzaakcode) geeft 48 min met bh: 4.5

Melding 3 (868221): 1.0(Prioriteit) & 130.0(Oorzaakcode) geeft 89 min met bh: 101.6

Meldingen 1 en 2 spreken voor zichzelf, ze hebben een lage betrouwbaarheid wat goed teken is. Daarnaast geeft te betrouwbaarheid ook maar 2, wat betekent dat het geen noodgeval is, om er zo snel mogelijk aan te werken. De reden daarvoor kan bijvoorbeeld doordat het een klein ongeval is, niet storend of op een niet veel gereden punt.

Melding 3 daarentegen geeft een hoge geschatte reparatietijd door met een slechte betrouwbaarheid, met waarschijnlijk de reden dat het een druk punt is en een groot probleem is, wat erg lang kan duren. Daarnaast kan er veel verschillende meldingen zijn bij prioriteit 1, zoals meldingen die een kwartier duren of zelfs een weken kunnen duren. Dat verslechtert de betrouwbaarheid.

Nawoord

Wij hebben als team heel veel geleerd van dit project. Zijn tegen problemen aangelopen en hebben ze goed verbeterd om te leren van de fouten die wij eerder hebben gemaakt. Hierdoor konden wij een project neerzetten waar wij zelf trots op kunnen zijn. Bedankt voor de mogelijkheid om in dit project samen te mogen werken met Pieter en Oscar van ProRail. Joost bedankt voor alle hulp door de weken heen! Tot volgend blok!