

# Verslag Distributed Databases

Dylan Cluyse, Laura Renders, Liam Goethals

14 december 2022

# Inhoudsopgave

# Inleiding

Tijdens het opleidingsonderdeel 'Distributed Databases' maakten wij kennis met Apache Spark. Spark biedt data-analyse gericht grootschalige gegevensverwerking. Deze technologie wordt aangeboden voor Java, Python, R en Scala. Voor dit opleidingsonderdeel werd Java gekozen als programmeertaal. Spark bevat enkele zijtakken dat zich richt op andere aspecten binnen data-verwerking, één daarvan is MLLib. MLLib is een pakket gericht op machine learning met Spark. Om meer kennis te vergaren kregen wij de groepsopdracht om via het platform Kaggle deel te nemen aan machine learning (ML) gerichte competities.

In dit verslag nemen wij u mee in de wereld van ML met gebruik van Spark. Wij willen u met volle plezier enkele concrete casussen tonen die gebruik maken van de verschillende regressie- en classificatiemethoden.

Allereerst willen wij de tijdsduur van een taxi-rit in downtown New York berekenen met regressie. Vervolgens detecteren wij kredietkaartfraude met behulp van binaire classificatie. Als derde oefening willen wij op basis van tekstinhoud achterhalen of een Tweet gerelateerd is aan een ramp. Tot slot geven wij u onze bevindingen mee van het MLLib pakket. Hier leggen we de aanpak van Spark en SKLearn, een pakket dat we in het opleidingsonderdeel Machine Learning zagen, parallel tegenover elkaar.

# 1. Tijd van een taxi-verplaatsing voorspellen met regressie.

## Probleemstelling

Als eerste opdracht deden wij mee aan de "New York City Taxi Trip Duration"competition. Dit als late inzending. Bij deze competitie krijgen wij twee datasets: een training- en een testset. Er wordt gevraagd om de totale tijdsduur van een taxirit te berekenen op basis van de gekregen data. Volgende kolommen werden gegeven: het ID van de verkoper, het aantal passagiers in een taxi, het longitude en latitude van de plaats waar de passagier(s) werden opgehaald, de longitude en latitude van de plaats waar de passagiers werden gedropt en de pick-up en drop-off tijd. De data bevat echter foute datatypes en outliers. Deze moet eerst worden weggefilterd.

Bron: <https://www.kaggle.com/competitions/nyc-taxi-trip-duration>

## Aanpak

Onze data is nog te ruw en daarom maken wij een clean-functie aan. De pick-up en drop-off datetime wordt meegegeven als datetime-object. Dit moeten we veranderen naar een bruikbaar formaat voor het regressiemodel. Eerst gaan wij de datetime kolommen opsplitsen in twee kolommen: 'hour' en 'day'. We willen numerieke waarden, dus we behouden enkel de dag van de week (bijvoorbeeld 1) en het uur van de dag (bijvoorbeeld 12). Vervolgens gaan wij de outliers uit onze dataset halen. Voor de outliers willen wij kijken naar de rijen die een veel te hoge longitude of latitude hebben. De outliers kunnen wij bepalen door te kijken naar rijen die buiten het bereik van (3 x standaardafwijking) vallen. Als laatst verwijderen we alle rijen die geen passagiers meenemen.

Vervolgens gaan wij twee kolommen toevoegen: distance en speed. In

de dataset krijgen wij de coördinaten mee (longitude en latitude) van het vertrek- en aankomstpunt. Deze bieden weinig waarde aan ons model. Met de afgelegde afstand heeft het model meer grip om de afgelegde tijd te kunnen berekenen. Hiervoor gebruiken wij de haversine-formule zoals hieronder afgebeeld. Met deze formule berekenen we de kortst mogelijke afstand tussen twee punten op het oppervlak van een bolvormig object. De 'great-circle distance' berekenen we met behulp van een user-defined function. Hierbij geven we vier double-variabelen mee met een double als return-waarde. De vier input-variabelen zijn: de longitude van het vertrekpunt, de latitude van het vertrekpunt, de longitude van het aankomstpunt en de latitude van het aankomstpunt. Wiskundige tussenstappen, zoals de vierkantswortel of het omzetten naar een radiaan, doen we met de Math-library van Java.

Naast een kolom 'distance' voegen wij ook een kolom 'speed' toe. Deze zal de snelheid in miles per hour bijhouden. Hiervoor gebruiken wij opnieuw een user-defined-function met twee doubles als inputparameters en één double als outputparameter.

Na het maken van de kolommen tonen wij, in de terminal, de gemiddelde snelheid en gemiddelde afstand per dag en per uur. Vervolgens starten we met een pipeline te maken. Voor we de features assembleren naar één kolom moeten wij eerst de categorische waarde "store and forward flag" gaan omzetten naar een numerieke waarde. Een vlag met waarde 'true' zal dan waarde 1 krijgen en omgekeerd. Dit doen we met een StringIndexer object. Vervolgens gaan wij de features omzetten naar één feature-kolom. Dit doen we met een VectorAssembler-object. Hierbij geven we de vendor id, aantal passagiers, het uur, de dag, de vlag en de afstand mee als features. De features-kolom zal een kolom zijn met alle features in één vector. Het probleem is dat alle features verschillende numerieke waarden en bereiken heeft. Dit zal een effect hebben op de uitkomst van ons model. Wij moeten onze feature-kolom op een gelijke schaal gaan brengen. Dit doen we door te werken met een MinMaxScaler-object. Een alternatief voor deze opdracht was een StandardScaler. Als laatste object gaan we ons regressiemodel meegeven. Voor dit regressieprobleem hebben wij gekozen voor een lineair model, een randomforestmodel, een gradientboostmodel en een generalised lineair model.

Bij alle modellen, op het lineaire model na, moeten we de parameters finetunen. Als we dit niet doen zal ons model niet optimaal benut worden en tegenvallende resultaten teruggeven. Dit probleem pakken wij aan door te werken met een ParamGridBuilder. Hiermee kunnen wij meerdere waarden voor verschillende parameters meegeven. Iedere combinatie zal aan bod komen. Hoe meer parameters, hoe langer de uitvoertijd van de applicatie.

## Evaluatie

Bij dit regressieprobleem voeren wij twee algemene testen uit: de correlatiematrix en de vier metrieken. Deze metrieken zijn de Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) en de correlatiewaarde ( $R^2$ ).

## 2. Credit Card fraude achterhalen met classificatie.

Bron: <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>

### Aanpak

We hebben hier twee klassen: correcte tweets en fraudulente tweets. We moeten een aanpak vinden voor een binaire classificatiemodel.

### Evaluatie

Bij het evalueren van het classificatiemodel ging onze voorkeur uit naar de confusionmatrix. Zo hebben wij een zicht op hoe goed ons model de klassen kan voorspellen. Bij de logistische regressie merken wij een goed evenwicht op tussen valse positieven en valse negatieven. De classificatiemodellen met Random Forest en Decision Trees voorspellen daarentegen weinig valse negatieven, maar wel uitbundig veel valse positieven. Als extra hebben wij ook de nauwkeurigheid van het model berekent.

### 3. De tekstinhoud van tweets detecteren op .

#### Aanpak

Gebruikte regressiemodellen: \* Lineaire regressie \* Random forest regressie

De resultaten van de metrieken waren zwak. Onze presumptie lag bij de volatiele waarden. Sommige waarden waren tussen het bereik van 0 - 10. De ... feature daarentegen had een range van 0 - 1000. Dit hebben we opgelost door een MinMaxScaler toe te voegen aan de pipeline.

#### Evaluatie



## 4. Verschillen highlighten met andere machine learning pakketten

Sklearn

...

## Conclusie