HO
GENT

Faculteit Bedrijf en Organisatie

Vergelijkende studie van voorspellingsmodellen voor tijdreeksen

Emiel Declercq

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Johan Decorte
Co-promotor:
Stijn Lievens

Instelling:

Academiejaar: 2020-2021

Eerste examenperiode

Faculteit Bedrijf en Organisatie

Vergelijkende studie van voorspellingsmodellen voor tijdreeksen

Emiel Declercq

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Johan Decorte
Co-promotor:
Stijn Lievens

Instelling:

Academiejaar: 2020-2021

Eerste examenperiode

Woord vooraf

Deze bachelorproef werd geschreven als eindwerk voor het afronden van de opleiding Toegepaste Informatica aan de Hogeschool Gent met specialisatie e-business.

Daarnaast wil ik ook nog ir. Johan Decorte bedanken voor de vlotte begeleiding van deze bachelorproef en Stijn Lievens voor het opnemen van het co-promotorschap.

Mijn ouders wil ik ook hartelijk bedanken voor alle steun, niet alleen tijdens deze scriptie maar gedurende mijn volledige studietraject. Zeker tijdens deze lockdown kan ik mij voorstellen dat dit op sommige momenten heel wat stress heeft veroorzaakt. Tenslotte verdient ook mijn zus, Edith Declercq een woordje van dank voor al het naleeswerk tot in de late uurtjes en de resem van aangereikte tips bij het schrijven van een paper.

Samenvatting

In deze bachelorproef zullen verschillende voorspellingstechnieken voor verschillende types tijdreeksen geanalyseerd worden. Er zal ook getest worden op verschillende types tijdreeksen. Het onderscheid tussen deze types wordt gemaakt op basis van 2 criteria namelijk seizoensgebondenheid en het aantal onafhankelijke variabelen. Deze zullen dan onderverdeeld worden in een al dan niet seizoensgebondenheid en een enkele of meerdere onafhankelijke variabelen. In totaal zullen er dus voorspellingen gemaakt worden voor 4 verschillende types tijdreeksen namelijk.

- Tijdreeksen met enkel de tijd als onafhankelijke variabele en 1 afhankelijke variabele zonder seizoensgebonden verband
- Tijdreeksen met enkel de tijd als onafhankelijke variabele en 1 afhankelijke variabele met een seizoensgebonden verband
- Tijdreeksen met 2 onafhankelijke variabelen waarvan 1 de tijd en 1 afhankelijke variabele zonder een seizoensgebonden verband
- Tijdreeksen met 2 onafhankelijke variabelen waarvan 1 de tijd en 1 afhankelijke variabele met een seizoensgebonden verband

De eerste voorspellingstechniek die onder de loep zal genomen is polynomiale regressie. De tweede techniek die zal toegepast worden is een ARIMA/VARMAX model. Tenslotte zal een recurrent neuraal netwerk van het type LSTM (Long Term Short Memory) op dezelfde data toegepast worden. Voor elk van deze 4 types tijdreeksen zal dan bepaald worden welk van de 3 technieken het beste resultaat zal opleveren.

Inhoudsopgave

Lijst van figuren

1. Inleiding

De hoeveelheid data die men kan verwerven in het digitale tijdperk waarin we de dag van vandaag leven is gigantisch zeker met de opkomst van het internet der dingen beter gekend onder de noemer *Internet of Things*. Deze data kan enorm uiteenlopend zijn, zo wordt onder andere de buitentemperatuur bijgehouden, maar ook de waarde van de aandelen van een bedrijf op de beurs, het aantal bezette plaatsen in een parking, het aantal mensen dat positief getest heeft op het coronavirus, ...

Zo kan het lijstje nog een hele tijd aangevuld worden, maar de net opgenoemde gegevens zijn niet enkel voorbeelden van data. Deze zaken variëren ook nog eens doorheen de tijd. Om dit in contrast te stellen met een ander voorbeeld zou een naam of een postcode van je geboorteplaats niet variëren en dus niet tijdsgebonden zijn. Een sequentie van data die tijdsgebonden is wordt benoemd als een tijdreeks ofwel een *time series*.

Een voorbeeld hiervan zou het aantal dagelijks gewandelde kilometers van het afgelopen jaar zijn. Doordat dit een tijdreeks is zouden we het aantal dagelijks gewandelde kilometers van het volgende jaar kunnen voorspellen met behulp van bepaalde modellen. Daaruit zouden we dan bijvoorbeeld kunnen vaststellen dat er in de winter minder gewandeld zal worden. Het nut hiervan zou dan kunnen zijn dat men inziet dat men te weinig lichaamsbeweging zal hebben en daar dan zal op inspelen door een indoorsport te beoefenen zodat men toch nog voldoende lichaamsbeweging heeft. Bij dit voorbeeld is het verband zeer simpel en kan men zich afvragen waarvoor het opstellen van een model nuttig zou zijn aangezien dit vrij makkelijk af te leiden valt met het blote oog. Maar soms zal het verband een pak complexer zijn dan een seizoen waardoor het moeilijk te vatten valt voor het menselijk brein maar praktischer is om te identificeren door middel van een wiskundig model.

De modellen die onderzocht zullen worden zijn polynomiale regressie, ARIMA/VARMAX modellen en LSTM modellen. Daarnaast zal ook nog rekening gehouden met het aantal invoerparameters. Zo zullen modellen die gebruik maken van 1 invoerparameter benoemd worden als univariate modellen, modellen die gebruik maken van meerdere invoerparameters daarentegen zullen benoemd worden als multivariate modellen. Deze 2 types modellen kunnen dan nog eens onderverdeeld worden in seizoensgebonden data en niet-seizoensgebonden modellen. Seizoen wijst hier echter niet enkel op regelmatige afwijkingen binnen de tijdspanne van 1 jaar maar op cyclische regelmatige afwijkingen doorheen de volledige tijdreeks. Dit betekent dus dat een cyclus ook wekelijks herhaald zou kunnen worden. Naast de cyclische trend zullen er eventueel ook nog andere afwijkingen zijn, zo zou het gemiddelde van elke cyclus op lange termijn kunnen stijgen. Om dit te neutraliseren zou de elke waarde verminderd kunnen worden met een bepaalde stijgende factor die na het voorspellende

1.1 Probleemstelling

Voorspelde tijdreeksen kunnen zeer belangrijke data zijn om de richting waarin bepaalde beslissingen genomen moeten worden aan te geven. Een brandend actueel voorbeeld hiervan zou een voorspellingsmodel van de coronacijfers zijn wanneer er zo'n model beschikbaar is met een betrouwbaarheid van 100% wat zou er in principe geen discutie meer mogen zijn over de ernst van de situatie en het treffen van de correcte maatregelen zou een pak praktischer zijn. Dit is echter vrij onwaarschijnlijk aangezien er in dit geval tal van factoren een invloed hebben op die cijfers waarvan er velen zeer moeilijk te registeren vallen. Gelukkig zijn er ook problemen waarvan de factoren makkelijker registreerbaar zijn zoals het verminderen van de ijsoppervlakten aan de polen. Zo zal onderzocht worden welk model de beste voorspellingen maakt voor univariate niet-seizoensgebonden tijdreeksen, univariate seizoensgebonden tijdreeksen, multivariate niet-seizoensgebonden tijdreeksen en multivariate seizoensgebonden tijdreeksen. De conclusies en de broncode van dit onderzoek zal kan kunnen dienen als basis voor het opstellen van voorspellingsmodellen van andere tijdreeksen.

1.2 Onderzoeksvraag

Vergelijkende studie van voorspellingsmodellen voor tijdreeksen

1.3 Onderzoeksdoelstelling

De doelstelling van deze bachelorproef is om te achterhalen welk model van de volgende drie:

- Lineaire of polynoomregressie
- Autoregressie
- LSTM

de beste resultaten halen voor data waarbij een of meerdere onafhankelijke variabelen beschikbaar zijn en/of een seizoensgebonden verband aanwezig is. Dit onderzoek zal ook kunnen dienen als basis voor het opstellen van een voorspellingsmodel van andere tijdreeksen.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk **??**, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Stand van zaken

In dit hoofdstuk zullen de 3 gebruikte voorspellingstechnieken toegelicht worden.

2.1 Regressieanalyse

Regressieanalyse is een voorspellende modelleringstechniek die de relatie tussen een afhankelijke en onafhankelijke variabele onderzoekt. Deze techniek kan ook toegepast worden indien er meerdere onafhankelijke variabelen beschikbaar zijn. Dit betekent dus dat er een relatie gezocht wordt tussen de afhankelijke en de onafhankelijke variabelen om de best passende lijn of regressievergelijking op te stellen. Op basis van deze regressievergelijking waarbij de onafhankelijke variabele(n) als invoerwaarde gebruikt worden, zal de afhankelijke variabele voorspeld worden.
In het geval van tijdreeksen zal het tijdsverloop altijd de onafhankelijke variabele zijn bij univariate modellen en een van de invoervariabelen bij multivariate modellen. Er zijn verschillende manieren om aan regressie te doen de voornaamste technieken zijn: lineaire regressie, polynomiale regressie en logaritmische regressie. Deze zullen dan ook verder toegelicht worden.

2.1.1 Lineaire Regressie

De meest eenvoudige vorm van regressie is lineaire regressie. Dit type regressie wordt vaak gebruikt en wordt best toegepast op continue data (**Pant2019**). Dit is data die gelijkmatig stijgt of daalt en zal grafisch weergegeven worden als een rechte. Bij realistische

voorbeelden zullen deze waarden vrijwel nooit op een rechte lijn liggen maar zullen er altijd enkele uitschieters zijn. Indien we dit grafisch zouden weergeven krijgen we een rechte te zien die de gegeven waarden optimaal zal benaderen.

Een voorbeeld van zo'n lineaire data is weergegeven op de tabel 2.1. Op basis van de cijfers valt dit niet echt makkelijk waar te nemen maar op figuur 2.1 is er duidelijk stijgende tendens zichtbaar in de prijs naargelang de oppervlakte van de leefruimte groter is.

Hieronder staat de basisformule voor een lineaire regressie.

$$y_i = \beta_0 + \beta_1 X_i + \varepsilon_i \tag{2.1}$$

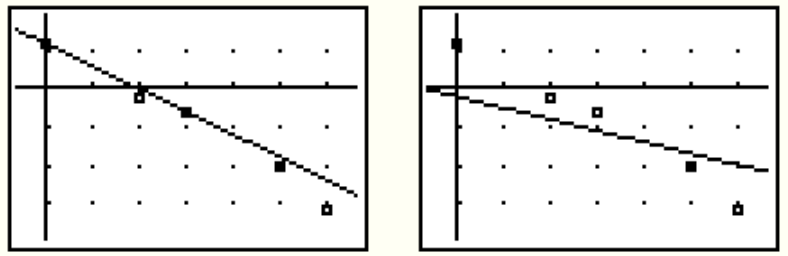
Het eerste deel van deze formule meerbepaald de eerste en de tweede term van de optelling, zal de lineaire component weergeven terwijl de resterende derde term rekening zal houden met de willekeurige fout.
De lineaire component is in principe een eerstegraadsfunctie die de basis zal vormen voor de formule. De y_i zal de afhankelijke variabele weergeven, dit zal dus de te voorspellen waarden weergeven. β_0 zal het intercept weergeven ofwel de waarde van de afhankelijke variabele wanneer de onafhankelijke variabele nul is. Of eenvoudigweg de waarde van de functie waarbij de curve de y-as snijdt. β_1 wordt benoemd als de richtingscoëfficient, deze zal bepalen hoe sterk de rechte stijgt of daalt. Een hogere richtingscoëfficient zal tot een steilere curve leiden terwijl een richtingscoëfficient die 0 benadert vlakker zal zijn, wanneer deze waarde effectief 0 is zal de afhankelijke waarde constant zijn de onafhankelijke variabele die weergegeven wordt door $\mathrm{X_i}$ heeft hier dus geen invloed op aangezien deze geneutraliseerd wordt door de 0. Indien de richtingscoëfficient negatief is zal deze curve ook dalen in plaats van stijgen.

De Random Error component dient om alle ruis te vatten. Ruis zijn eigenlijk willekeurige afwijkingen die een verband zouden beïnvloeden waardoor een curve van reëele data vrijwel nooit perfect lineair zal zijn. Wanneer deze modellen in de praktijk toegepast worden wordt hier echter vrijwel geen rekening mee gehouden. Dit aangezien verondersteld wordt dat deze afwijkingen elkaar zullen neutraliseren en zo zou de gemiddelde afwijking teweeggebracht door ruis in de curve 0 zijn. Dus ε_i zal een factor zijn die de ruis weergeeft in de formule van lineaire regressie maar valt te verwaarlozen in deze context.

Wanneer we de onafhankelijke variabelen invoeren in de formule zouden we dus de afhankelijke variabelen moeten kunnen voorspellen. Om deze formule in te vullen moeten echter waarden ingevoerd worden voor β_0 en β_1. De meest optimale waarden voor β_0 en β_1 zullen dus waarden zijn die er voor zorgen dat de curve zo nauw mogelijk aansluit bij de data zelf. Dit zal dan ook betekenen dat de opgetelde afstand tussen de datapunten en de curve die onze lineaire regressie weergeeft zo klein mogelijk zal zijn. Deze methode wordt ook wel benoemd als *the sum of the least squares*. Om dit grafisch even voor te stellen valt op te merken dat in figuur 2.1 de eerste curve nauwer aansluit bij de data dan de tweede en we kunnen ook wel vaststellen dat de som van de afstanden tussen de datapunten en de curve bij de eerste figuur lager ligt.

Hierdoor kan dus gesteld worden dat door β_0 en β_1 te variëren en te kijken welke parametercombinatie voor de curve zal zorgen waarvan de afstand van de datapunten en die curve

Figuur 2.1: Grafische weergave sum of least squares (Brown, 2020)



het laagst is, bepaald kan worden welke waarden voor β_0 en β_1 optimaal zijn.

De accuraatheid van een curve kan dus weergegeven aan de hand van formule 2.2.

$$Error = \sum (actual\ output - predicted\ output)^2 \tag{2.2}$$

Nu rijst de vraag hoe de parameters zodanig kunnen evolueren zodat ze de kleinst mogelijke foutmarge benaderen. Dit wordt gerealiseerd aan de hand van gradient descent een uitgebreide uitleg van gradient descent komt aan bod bij het bespreken van de LSTM techniek maar het komt er op neer dat gradient descent er zal voor zorgen dat de optimale parameters zo nauwkeurig mogelijk bepaald zullen kunnen met een zo laag gebruik van middelen.

Tabel 2.1: Voorbeelddata voor lineaire regressie (**Pant2019**)

Living area (feet^2)	Price (1000$)
2104	400
1600	330
2400	369
1416	232
3000	540

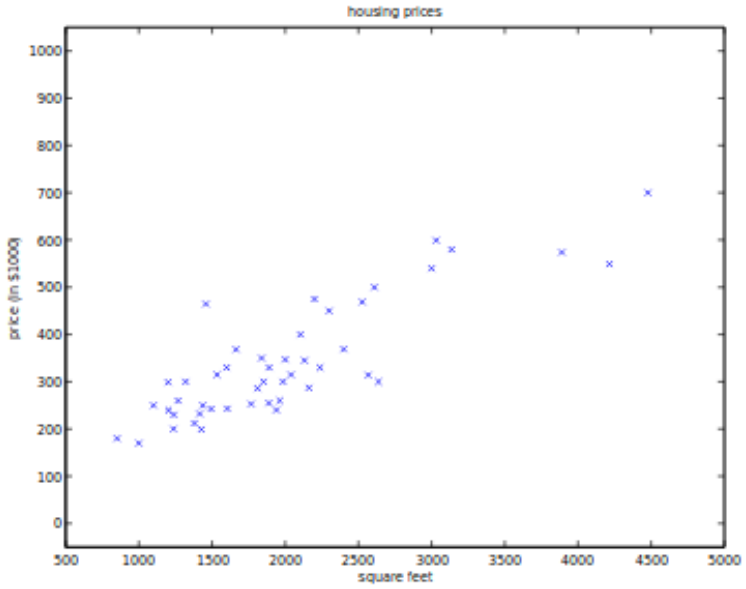
2.1.2 Polynomiale Regressie

Soms zal de foutmarge bij data hoog blijven bij lineaire regressie ook al is er een verband zichtbaar, dit verband zal dan waarschijnlijk niet te vatten vallen onder een gewone rechte. In zo'n geval kan polynomiale regressie hulp bieden, dit zal een complexere functie aan de data proberen fitten. Formule 2.3 is dan formule van een 2^{de} graadsfunctie

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2^2 \tag{2.3}$$

Dit zorgt ervoor dat de algemene formule herleid wordt naar de formule 2.4 indien we een polynomiale vergelijken wensen toe te passen bij de regressieanalyse.

Figuur 2.2: Grafische weergaven voorbeelddata voor lineaire regressie (**Pant2019**)



$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + ... + \beta_m x^m + residual\ error \tag{2.4}$$

Ook bij polynomiale regressie wordt gebruik gemaakt van gradient descent om de minimale kost te berekenen aangezien dit in principe een uitbreiding is op lineaire regressie.

Dus polynomiale regressie zal de beste gemiddelde inschatting kunnen maken van de relatie tussen de afhankelijke en onafhankelijke variabelen. De curve die gefit moet worden zal zeer complexere vormen dan een simpele rechte kunnen aannemen en meer dan enkel lineaire verbanden kunnen deduceren uit de data. Het grootste nadeel van polynomiale regressie is dat enkele uitschieters de resultaten sterk kunnen beïnvloeden terwijl er bij lineaire regressie meer validatietechnieken beschikbaar zijn voor het achterhalen van uitschieters (**Pant2019**).

2.1.3 Logistische Regressie

Logistische regressie is gelijkaardig aan lineaire regressie maar in tegenstelling tot lineaire regressie zal dit voornamelijk gebruikt worden voor classificatieproblemen, waarbij dus geen continue waarden dienen voorspeld te worden maar nominale waarden. Onder zijn meest simpele vorm zal dit type algoritme nuttig zijn bij het voorspellen van binaire waarden bijvoorbeeld indien men al dan niet aan een aandoening zal lijden. Dit wordt benoemd als binaire logistische regressie. Wanneer er 3 of meer categoriën die dienen voorspeld te worden waarvan de ene niet duidelijk hoger of lager is dan de andere zoals bijvoorbeeld type huisdier wordt dit benoemd als multinomial logistic regression. Indien er wel een logische ordening in zit zoals een aantal sterren op een filmbeoordeling wordt dit benoemd als ordinale logistische regressie. Aangezien in deze paper enkel continue

waarden voorspeld zullen worden, zal hier niet dieper op in gegaan worden (Swaminathan, 2018).

2.1.4 Meervoudige lineaire regressie

Indien er meerdere onafhankelijke variabelen beschikbaar zijn die een lineaire invloed zouden hebben op de afhankelijke variabele kunnen deze ook in rekening gebracht worden. Dit model zal dan opgesteld worden aan de hand van meervoudige lineaire regressie ofwel multiple linear regression. Formule 2.5 geeft weer hoe de onafhankelijke waarden bij meervoudige lineaire regressie berekend zullen worden.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_n x_{in} + \varepsilon_i \tag{2.5}$$

Net zoals bij enkelvoudige lineaire regressie zal y_i de afhankelijke variabele weergeven en β_0 het intercept. Het verschil zit hem in de termen die volgen want aangezien er rekening gehouden wordt met meerdere onafhankelijke variabelen, zullen ook deze verwerkt worden in de formule door middel van een waarde te voorzien voor elke onafhankelijke variabele weergegeven door X_i met bijhorend intercept weergegeven door β. De n in de formule geeft dan het aantal onafhankelijke variabelen weer. De ε_i zal ook in dit geval een factor zijn die rekening zal houden met de meetfouten (Kenton, 2020).

2.1.5 Toepassing

Deze regressietechnieken zullen dienen als basis voor andere meer uitgebreide technieken zoals ARIMA maar de verbanden die ze weergeven zijn te eenvoudig om effectief te gebruiken voor het opbouwen van modellen om aan extrapolatie te doen. (Sinha, 2019) Daarom zullen de klassieke regressiemethodes niet toegepast worden bij deze bachelorproef.

2.2 ARIMA

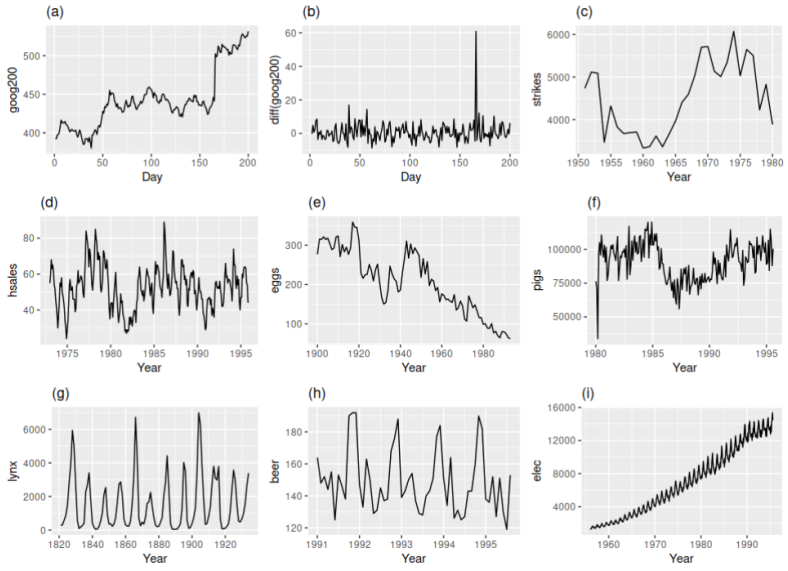
ARIMA is een modeleringstechniek die voorspellingen maakt op basis va, stationaire en gedifferentiëerde data. Daarom worden deze begrippen eerst toegelicht alvorens de techniek zelf te bespreken.

2.2.1 Stationariteit

Een stationaire tijdreeks is een tijdreeks waarvan de eigenschappen niet afhankelijk zijn van het tijdstip waarop de reeks wordt geobserveerd (Rob J Hyndman, 2018). Indien er een duidelijke trend dus een algemene stijging of daling aanwezig is zal de tijdreeks niet

stationair zijn. Ook wanneer er een seizoensverband aanwezig is die lijkt op een sinus of cosinusfunctie wordt de data als niet stationair beschouwd. Wanneer er echter een cyclisch verband aanwezig is kan er een speciaal type ARIMA model gebruikt worden dat rekening zal houden met deze seizoenscomponent genaamd SARIMA deze zal zodadelijk toegelicht worden. In figuur 2.3 zijn enkele voorbeeldtijdreeksen zichtbaar dit toont aan dat het niet zo evident is om te bepalen welke tijdreeks stationair is.

Figuur 2.3: Voorbeeldtijdreeksen (Rob J Hyndman, 2018)



De cijfers in de figuur omschrijven de volgende data:

(a) De aandeelkoersen van Google voor 200 opeenvolgende dagen
(b) De dagelijkse veranderingen in de aandeelkoersen van Google voor 200 opeenvolgende dagen
(c) Het jaarlijkse aantal stakingen in de Verenigde Staten
(d) Maandelijkse verkoop van nieuwe gezinswoningen in de Verenigde Staten
(e) Jaarlijkse prijs van een dozijn eieren in de Verenigde Staten (rekening houdend met inflatie)
(f) Maandelijks aantal varkens die geslacht worden in Victoria, Australië
(g) Jaarlijks aantal lynxen die opgesloten worden in het McKenzie River district van noordwest Canada
(h) Maandelijkse Australische bierproductie
(i) Maandelijkse Australische elektriciteitsproductie

2.2.2 Seizoenseffect

Bij de figuren (d), (h) en (i) is er een duidelijk seizoenseffect zichtbaar. Er vallen duidelijke trends waar te nemen op figuren (a), (c), (e), (f) en (i). Ook in figuur (g) lijkt er een duidelijk seizoenseffect te zijn, maar deze veranderingen zijn aperiodisch dus op lange termijn valt dit niet te voorspellen aldus is deze tijdreeks stationair.

2.2.3 Differentiatie

Aan de hand van deze grafieken kan ook een ander begrip toegelicht worden dat een belangrijk gegeven is bij het opstellen van een ARIMA model namelijk differentiatie. Zo wordt op figuur (a) de aandeelkoers van google weergegeven voor 200 opeenvolgende dagen, maar hier is een duidelijk stijgende trend zichtbaar. Hierdoor kan de data niet op deze manier geïnterpreteerd worden door een ARIMA model. Maar we kunnen de data op een andere manier voorstellen namelijk door de koerswissellingen zoals weergegeven wordt op figuur (b) waardoor er geen trend meer aanwezig zal zijn en nog steeds geen zichtbare seizoensgebondenheid aanwezig is.

Random walk differentiatie

Bij random walk differencing wordt de data gedifferntiëerd door het verschil te nemen tussen een tijdstap en de tijdstap die ervoor komt. Dit zal de evolutie van de tijdreeks weergeven en aangeven hoeveel een waarde precies verschilt van de tijdstap ervoor. Deze differentiatie kan uitgedrukt worden als

$$y'_t = y_t - y_{t-1}. \tag{2.6}$$

De gedifferentiëerde reeks zal 1 waarde minder hebben dan de originele serie omdat er geen waarde voor de eerste y'_1 komt om deze te differentiëren.

Wanneer de gedifferentiëerde reeks ruis is kan het model uitgedrukt worden als

$$y_t - y_{t-1} = \varepsilon_t, \tag{2.7}$$

Hierbij zal $epsilon_t$ de ruis weergeven. Door deze formule te hervormen bekomen we het ”random walk model".

Deze modellen worden vaak gebruikt voor niet-stationaire data, voornamelijk bij financiële en economische toepassingen.

De voorspellingen van dit model zijn gelijk aan dit de laatste observatie.

Tweedegraadsdifferentiatie

In sommige gevallen zal gedifferencieerde data nog steeds niet stationair zijn dan is het eventueel mogelijk om de gedifferentiëerde nogmaals te differentiëren. Dit kan als volgt geformuleerd worden.

$$\begin{aligned} y_t^n &= y'_t - y'_{t-1} \\ &= (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \\ &= y_t - 2y_{t-1} - y_{t-1} \end{aligned} \tag{2.8}$$

Op deze manier kan een verandering in de veranderingen gedetecteerd worden. Uiteraard zou men nog een stap verder kunnen gaan maar dit blijkt in praktijk zo goed als nooit nodig te zijn.

Seizoensdifferentiatie

Een laatste manier om data te differentiëren is seizoensdifferentiatie. Hierbij zal telkens het verschil genomen worden van een tijdstap en diezelfde tijdstap van de vorige seizoenscyclus. Dit kan als volgt weergegeven worden,

$$y'_t = y_t - y_{t-m}. \tag{2.9}$$

Hier zal m slaan op het aantal observaties in een seizoen. Indien deze data, die seizoensgedifferenciëerd is, ruis blijkt te zijn dan zou een geschat model voor de originele data kunnen geformuleerd worden als

$$y_t - y_{t-m} = \varepsilon_t, \tag{2.10}$$

Om seizoensdifferentiaties en gewone differentiaties uit elkaar te houden worden gewone differentiaties ook benoemd als eerste differentiaties.

Het is mogelijk dat deze verschillende soorten differentiaties gecombineerd moeten worden om tot een stationaire dataset te komen. Een dataset waarbij het waarschijnlijk nodig zal zijn om eerste differentiaties en seizoensdifferentiaties gecombineerd moeten worden is die van de maandelijkse Australische elektriciteitsproductie zichtbaar op figuur 2.3 (i). Aangezien er een seizoenscyclus en een stijgende trend aanwezig is.

2.2.4 Toeliching ARIMA

ARIMA is een acroniem gevormd door AutoRegressive Integrated Moving Average wat vrij vertaald kan worden als het autoregressie geïntegreerde voortschrijdend gemiddelde. Deze benaming geeft ook onmiddellijk de sleutelaspecten van dit model weer.

Zo staat autoregressief voor het modelleren van een volgende stap in een sequentie als een lineaire functie op basis van de voorgaande waarden in deze sequentie.
Daarnaast is ARIMA ook een geïntegreerd model wat inhoudt dat niet de cumulatieve waarden gebruikt worden om voorspellingen te maken maar er gekeken wordt naar het

verschil tussen die cumulatieve waarden.
Ten slotte wordt ook gekeken naar het voortschrijdend gemiddelde in plaats van naar de ruwe data zelf dit zal ervoor zorgen dat extreme waarden veroorzaakt door anomaliteiten geneutraliseerd worden. Hierdoor zal de globale trend beter waarneembaar zijn.

ARIMA modellen zijn theoretisch gezien, de meest algemene klasse van modellen om een tijdreeks te voorspellen die stationair/stationary gemaakt kan worden.

Elk van deze componenten is geparameteriseerd en kan dus aangepast worden deze parameters worden benoemd als (p, d & q)

p: Het aantal verlate observaties die worden opgenomen in het model. Deze worden in het Engels benoemd als lag observations of lag order.
d: Zal bepalen hoeveel keer ruwe observaties van elkaar afgetrokken worden dit kan ook benoemd worden als de mate van differentiatie of the degree of differencing in het Engels.
q: Deze parameter zal het aantal tijdstappen waarvan het voortschrijdend gemiddelde genomen wordt aangeven. Deze wordt in het Engels benoemd als the order of moving average.

Een lineair regressiemodel wordt geconstrueerd met data waarbij er een graad van differentiatie is zodat deze stationair is. Dit houdt in dat trend en seizoensstructuren die het regressiemodel op een negatieve manier beïnvloeden verwijderd worden.

Bij elke parameter kan ook de waarde 0 opgegeven worden zodat dit aspect horende bij die parameter buiten beschouwing gelaten wordt en dit model effectief gebruikt kan worden als gelijk welke combinatie zoals bijvoorbeeld een autoregressief en een geïntegreerd model al dan niet gebruikmakend van het voortschrijdend gemiddelde (Brownlee, 2018a).

d zal dus de differentiatiegraad weergeven wanneer we hier rekening mee houden kunnen we stellen dat de formule voor ARIMA er als volgt uit zal zien (Lau, 2020):

$$\hat{y_t} = \mu + \phi_1 y_{t-1} + ... + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} - ... - \theta_q \varepsilon_{t-q} \tag{2.11}$$

Om de werking van ARIMA te schetsen zullen kort enkele basisparametercombinaties toegelicht worden.

ARIMA(1,0,0)

Dit zal een autoregressief model van de eerste orde zijn ofwel een first-order autoregressive model. Deze parametercombinatie zal gebruikt worden als de reeks stationair en autogecorreleerd is. Dit zou dus voorspeld kunnen worden door de eigen waarde te nemen en deze op te tellen met een constante. De voorspellingsvergelijking zal er als volgt uit zien:

$$\hat{y_t} = \mu + \phi_1 y_{t-1} \tag{2.12}$$

Dit stelt voor dat er regressie uitgevoerd en kan benoemd worden als een ARIMA(1,0,0) model. Wanneer het gemiddelde van y gelijk is aan nul, zal de constante term achterwege gelaten worden. Indien de hellingscoëfficient ϕ_1 positief maar minder dan één is zal het model de waarde van de volgende periode voorspellen als ϕ_1 keer zo ver van het gemiddelde dan de waarde van deze periode.

Indien men een autoregressief model van de 2^{de} orde wil voorstellen zal er bij formule (2.12) rechts een y_{t-2} term toegevoegd worden. Wanneer men een model van nog hogere orde wenst voor te stellen dient men analoog termen toe te voegen. Afhankelijk van de tekens en de grootte van de coëfficienten zal een ARIMA(2,0,0) model een systeem onderschrijven waarvan de gemiddelde omkering sinusoïdaal oscillerend zal zijn, wat te vergelijken valt met de beweging van massa aan een veer die onderhevig is aan willekeurige uitwijkingen.

ARIMA(0,1,0)

Deze vorm van ARIMA past de meest simpele vorm van differentiatie toe namelijk "random walk", hierbij zal de eerste differentiatie die zonet besproken is in de sectie differentiatie toegepast worden op de data. Door middel van deze differentiatie kan niet-stationaire data omgevormd kunnen worden naar stationaire data indien dit nog steeds niet het geval is kan deze data in een hogere graad gedifferenciëerd worden. Dit kan weergegeven onder de vorm van deze formule:

$$\hat{y}_t = \mu + y_{t-1} \tag{2.13}$$

Hier zal de constante term μ de gemiddelde verandering van periode tot periode weergeven dit kan ook benoemd worden als de "long term drift" in y. Aangezien het enkel een niet-seizoensgebonden verschil en en een constante term omvat wordt het geclassificeerd als een "ARIMA(0,1,0) model met constante". Een random walk zonder drift model zou een ARIMA(0,1,0) model zonder constante zijn.

ARIMA(1,1,0)

Deze ARIMA configuratie staat voor een gedifferenciëerd autoregressief model van de eerste graad. Indien de fouten van een random walk model autogecorreleerd zijn is het mogelijk dat dit verholpen kan worden door een vertraging ofwel lag toe te voegen aan de afhankelijke variabele van de voorspellingsvergelijking. Zo zou dus regressie uitgevoerd worden op het eerste verschil van y en zichzelf, vertraagd met 1 tijdstap. Hierbij zal de vergelijking er als volgt uitzien:

$$\hat{y}_t = \mu + y_{t-1} + \phi_1(y_{t-1} - y_{t-2}) \tag{2.14}$$

ARIMA(0,1,1)

Hier wordt de laatste parameter van ARIMA voor het eerst aangesproken de q. Hierdoor zal er nu gewerkt worden met een voortschrijdend gemiddelde van de data. Dit zal er voor zorgen dat ruis iets meer geneutraliseerd wordt.

$$\hat{y_t} = \mu + y_{t-1} - \theta_1 e_{t-1} \tag{2.15}$$

De constante μ kan eventueel achterwege gelaten worden, deze zal er voor zorgen dat de voorspelling op lange termijn eerder hellend zal zijn hierbij zal het ook mogelijk zijn om een negatieve coëfficient toe te laten. Deze constante zorgt er ook voor dat het ARIMA model een SES model wordt. Indien hier niet voor gekozen wordt zal de voorspelling op lange termijn vlak zijn. Zonder constante wordt dit type model benoemd als simple exponential smoothening indien er een constante gebruikt wordt hier growth aan toegevoegd, wat wijst op de stijgende trend op lange termijn waarvan de helling gelijk is aan μ.

ARIMA(0,2,1) of ARIMA(0,2,2)

Deze configuraties zonder constante worden ook benoemd als lineaire exponentiële afvlakking. Hier zal de 2^{de} afgeleide genomen worden omwille van de waarde voor d.

$$\hat{y_t} = 2y_{t-1} - y_{t-2} - \theta_1 e_{t-1} - \theta_2 e_{t-2} \tag{2.16}$$

De θ_1 en θ_2 zullen staan voor de MA(1) en MA(2) coëfficienten. Dit zal overeenkomen met Holt's model en in speciale gevallen met Brown's model. het zal gebruik maken van exponentiële gewogen voortschrijdende gemiddeldes om het lokale niveau en lokale trend (local level and local trend) te schatten. Voorspellingen op lange termijn zullen naar een rechte lijn toe convergeren wiens helling zal afhangen van de gemiddelde trend naar het einde van de reeks toe.

ARIMA(1,1,2)

Dit wordt benoemd als damped-trend linear exponential smoothing en wordt weergegeven door de formule:

$$\hat{y_t} = y_{t-1} + \phi_1(y_{t-1} - y_{t-2}) - \theta_1 e_{t-1} - \theta_2 e_{t-2} \tag{2.17}$$

Dit model zal de lokale trend aan het einde van de reeks extrapoleren maar zal afvlakken naar het einde toe.

Bij het opstellen van een ARIMA mode wordt aangeraden om een van de parameters p of

q niet groter te laten worden dan 1 aangezien er een hoge waarschijnlijkheid is dat dit zal leiden tot overfitting of andere fouten.

Een manier om de corelaties en een eventueel vertraagd effect te identificeren tussen de waarden rekening houdend met trend, seasonality en het residu is een ACF ofwel een auto-corelation function (**Salvi2020**).

Een aanvulling daarop is een PACF ofwel een partial auto-correlation function deze zal eventuele correlaties wanneer de effecten na eerdere vertragingen verwijderd worden (door middel van ACF) weergeven.

Een goede vuistregel bij het opstellen van een ARIMA model is het verhogen van de p indien er een positieve correlatie is dit houdt in dat de afhankelijke variabele zal stijgen als de onafhankelijke variabele stijgt. Wanneer er een negatieve corelatie is en de onafhankelijke variabele stijgt zal de afhankelijke variabele dalen en in dit geval is het beter om de q te verhogen.

Een stappenplan om ARIMA toe te passen op een dataset zal er als volgt uit zien:

1. Plot de data en identificeer ongewone waarden
2. Transformeer de data om de variantie te stabiliseren indien nodig
3. Als de data niet stationair is neem eerste differentiaties van de data totdat deze stationair is.
4. Onderzoek de ACF/PACF en kijk of een ARIMA(p,d,0) of ARIMA(0,d,q) model gebruikt moet worden
5. Test de modellen uit en gebruik AICc (een scoringstechniek) om het model te verbeteren.
6. Bekijk de residuen van het gekozen model door ze te plotten samen met de ACF en een portmanteau test van de residuen te doen. Indien deze er niet uitzien als ruis is het aangeraden een nieuw model te proberen
7. Wanneer de residuen er uitzien als ruis kunnen voorspellingen gemaakt worden

2.2.5 Varianten op ARIMA

SARIMA

In sommige gevallen blijkt er een seizoensgebondenheid aanwezig te zijn in de data. Deze seizoensgebondenheid zal zorgen voor constante fluctuaties. Deze manier van beïnvloeden is iets waar bij een gewoon ARIMA model geen rekening mee wordt gehouden. Daarom is er een variant op het ARIMA model genaamd SARIMA waarbij de toegevoegde S zal staan voor de seizoensgebondenheid. Dit zal in rekening gebracht worden door extra parameters toe te voegen. Zo zullen de parameters bij ARIMA (p,d,q) zijn, terwijl de parameters bij SARIMA $(p,d,q)(P,D,Q)m$ zijn, waarbij m zal staan voor het aantal tijdsstappen binnen een seizoenscyclus. Zo zal bij een jaarlijkse cyclus m gelijk zijn aan 12 als de tijdspanne tussen de observaties 1 maand bedraagt. Daarnaast zal de p staan voor autoregressieve seizoensorde ofwel de seasonal autoregressive order, d

voor de gedifferencïeerde seizoensorde ofwel de seasonal difference order en q voor de seizoensorde van het voortschrijdend gemiddelde ofwel de seasonal moving average order (Brownlee, 2018b).

VARMAX

Wanneer er meerdere tijdsreeksen voorspeld moeten worden waarbij er een verband mogelijk zou kunnen zijn tussen de verschillende tijdsreeksen kan de VARMAX techniek gebruikt worden. Hierbij zal de V staan voor vector wat duidt op het interpreteren en voorspellen van vectoren. Dit type model zal dan ook gebruikt worden bij het voorspellen van multivariate of meervoudige tijdreeksen AR wijst andermaal op autoregressief. MA slaat ook weer op het voortschrijdend gemiddelde. De X in VARMAX staat voor de invloed van exogene factoren op de factoren die voorspeld moeten worden.

2.3 Long Short Term Memory

Als tweede methode die gebruikt zal worden om tijdreeksen te voorspellen wordt gekozen voor een recurrent neuraal netwerk met gebruik van Long Short Term Memory modellen wat afgekort wordt als LSTM.

Een recurrent neuraal netwerk zal uitgebreid toegelicht worden in de volgende secties maar dit kan kort omschreven worden als een structuur die gebruik maakt van trainingsdata om te “leren” en op basis van dit leerproces voorspellingen kunnen maken van nieuwe data. Om dit met een simpel voorbeeld te schetsen zou een neuraal netwerk net zoals wij als mens leren dat er bij veel bewolking kans zal zijn op regen. Om dit te realiseren zullen alle parameters echter numeriek doorgegeven moeten worden aan het neuraal netwerk in dit geval zou een soort van bewolkingsgraad aangewezen zijn. Op basis daarvan zal het neuraal netwerk dan een kans op regen kunnen aangeven.

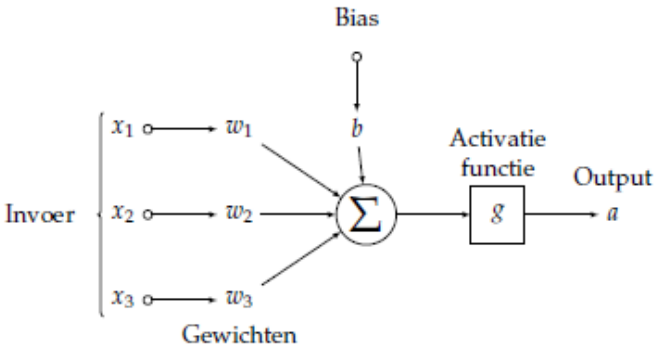
Een LSTM is een type recurrent neuraal netwerk dat gebruikt wordt bij sequentíele voorspellingsproblemen. Dit betekent dat de volgorde van de data een rol zal spelen en dat bij het voorspellen van een waarde in een reeks rekening zal gehouden worden met een aantal waarden die vlak voor de te voorspellen waarde komen.

2.3.1 Theoretische toelichting

Een neuraal netwerk valt makkelijkst te vergelijken met een menselijk brein. Zo zullen wij bijvoorbeeld het cijfer 1 herkennen aan zijn vormen. Door de verschillende pixels, gebruikt voor het weergeven van ons cijfer 1 als inputwaarden voor een neuraal netwerk te gebruiken en de outputwaarden gelijk te stellen met cijfers 0 tot en met 9 kan het netwerk aangeleerd worden deze cijfers te herkennen.

Dit gebeurt niet vanzelf maar door een combinatie van eenvoudige rekeneenheden die

Figuur 2.4: Voorstelling van een neuron die invoer ontvangt van 3 andere neuronen (Lievens, 2018)



verbonden zijn en waartussen er verbindingen zijn die beschreven worden met een reëel getal dat de sterkte van een verbinding aangeeft (Lievens, 2018). Zo'n combinatie van eenvoudige rekeneenheden die ook benoemd worden als neuronen wordt net zoals bij het menselijk brein een neuraal netwerk genoemd dit wordt weergegeven op figuur 2.4.

Zo bevinden zich tussen de input en de outputlaag verschillende lagen die benoemd worden als verborgen lagen. In deze lagen bevinden zich knooppunten (neuronen) waarin waarden worden berekend die een interpretatie geven van de inputwaarden waarmee ze verbonden zijn. Wanneer we dit toepassen op ons voorbeeld zou in een knooppunt bijvoorbeeld kunnen bekeken worden of er in het midden van de te interpreteren pixels een lijn staat. Wanneer dit het geval zou zijn zou er in dit neuron een grote waarde weergegeven worden en zal deze een sterk signaal sturen. Wanneer we voor dit voorbeeld met 1 verborgen laag zouden werken zou het neuron dat een sterk signaal zendt bij het herkennen van een centrale verticale lijn een belangrijke waarde zijn voor het herkennen van een 1 en een 4. De sterke van dit signaal zal dus een grote invloed hebben op de kans dat het resultaat een 1 of een 4 is. Wanneer we hier dieper op ingaan kunnen we aan elk neuron een formule toewijzen die er zo uitziet:

$$n = w_1 i_1 + w_2 i_2 \tag{2.18}$$

W staat voor de gewichten, i staat voor de inputwaarde. Stel dat bij dit neuron de bovenste 3 pixels genomen worden en we voor de eenvoud werken met een 3x3 afbeelding, dan zal de formule er als volgt uitzien:

$$n = w_1 i_1 + w_2 i_2 + w_3 i_3 \tag{2.19}$$

n zal dan staan voor gewogen gemiddelde, door hier een bias aan toe te voegen b net zoals bij lineaire regressie en dit als invoer te gebruiken voor een activatiefunctie g zal de sterkte van het uitvoersignaal bekomen worden.

Indien een neuron effectief geactiveerd wordt zal bepaald worden door de volgende

formule.

$$a = g(n+b) \tag{2.20}$$

Een bias b zal bij een neuraal netwerk zorgt voor een correcte neutrale toestand binnen een neuraal netwerk wanneer alle features nul zijn net zoals bij lineaire regressie.

Door het gebruik van een activatiefunctie g zal dit netwerk ook meer beginnen lijken op de werking van neuronen zoals ze voorkomen in de natuur.

Een activatiefunctie zal gebruikt worden om voor non-lineariteit te zorgen binnen het model. Wanneer we de activatiefunctie zouden laten wegvallen zou het volledige neurale netwerk kunnen herleid worden tot één lineaire functie waardoor er enkel weer lineaire verbanden gelegd kunnen worden. Het toevoegen van een activatiefunctie zal er voor zorgen dat de lineariteit gebroken wordt en er binnenin het netwerk nieuwe, abstracte features gegenereerd kunnen worden die kunnen resulteren in een betere interpretatie van de invoerdata.

Toepassing

Bij dit voorbeeld zijn we nog steeds op zoek naar een 1 en kennen een hoge waarde toe aan donkere kleuren opdat het model de afbeelding zou kunnen interpreteren. Dit zou betekenen dat i_1 en i_3 een lage waarde moeten hebben en i_2 een hoge waarde. In dit geval zullen de gewichten voor w_1 en w_3 gelijkgesteld worden aan -1 en het gewicht voor w_2 aan 1. Daardoor zal n een hoge waarde krijgen als de centrale pixel zwart is en de buitenste pixels wit. Dit wordt de propagation function genoemd. Om deze waarden om te zetten naar een waarde die interpretabel is doorheen het volledige model wordt het resultaat van de propagation function nog eens gebruikt als invoerwaarde voor een activatiefunctie. In dit geval zou de hyperbolic tangent (tanh) een goede keuze zijn, maar dit hangt af van de toepassing. Bij dit voorbeeld zou dat signaal dan naar de outputlaag gestuurd worden die op zijn beurt ook weer diezelfde formule toepast maar dan niet met de invoerwaarden maar met de resultaten uit de verborgen laag. Hieruit zal dan een score zou komen die aangeeft hoe waarschijnlijk het is dat de waarden die de pixels weergeven uit de invoerlaag het cijfer vormen dat toegekend is aan dit neuron in de uitvoerlaag.

Deze toepassing waarbij een cijfer wordt herkend is een voorbeeld van een toepassing van voorwaards gerichte neurale netwerken omdat de data slechts in 1 richting verloopt, een resultaat wordt niet beïnvloed door voorgaande inputwaarden. Bij recurrente neurale netwerken is dit wel het geval. Hierbij wordt rekening gehouden met de beoordelingen van voorgaande inputwaarden een voorbeeld hiervan is het beoordelen van zinnen de volgorde van de ingevoerde woorden zal een rol spelen. Zo zal ‘eet ik‘ geïnterpreteerd kunnen worden als een vraag aan de hand van de woordanalyse. Terwijl ’ik eet’ zal beoordeeld worden als een statement.

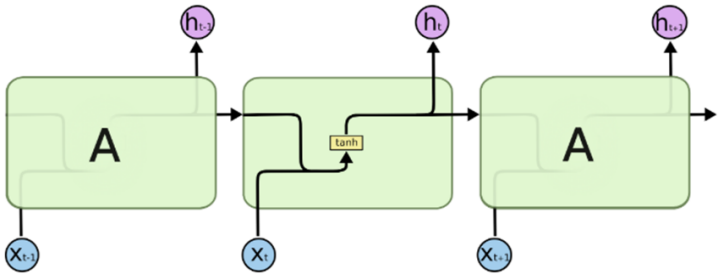
Dit zal louter gebeuren op basis van de woordvolgorde waarbij connecties tussen verschillende cellen teruglopend kunnen zijn onder elkaar en de vorige inputwaarden dus een

invloed zullen hebben op de beoordeling van de volgende inputwaarden (Lievens, 2018).

Dit is dus een gelijkaardige werking aan voorwaarts gerichte neurale netwerken maar de signalen verlopen niet allemaal in dezelfde richting binnen het netwerk. Zo kan er een signaal van de 2de verborgen laag teruggestuurd worden naar de 1ste verborgen laag. Op die manier hebben de waarden van vorige iteraties een impact op de volgende resultaten en dit is wat we willen bereiken.

Een lus van eenzelfde stuk uit het neuraal netwerk die informatie van vorige iteraties doorgeeft aan zichzelf dit wordt weergegeven op Figuur 2.5.

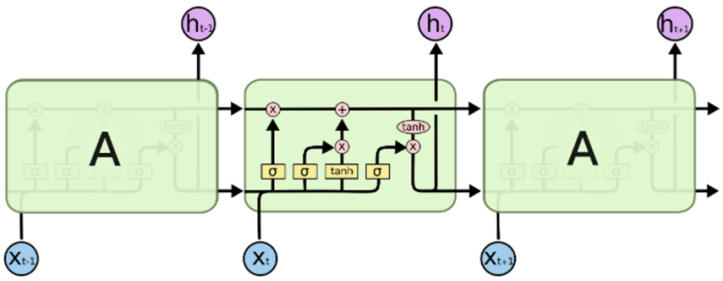
Figuur 2.5: Cel in een neuraal netwerk dat gegevens van een andere cel gebruikt om een nieuw signaal uit te sturen (Olah, 2015)



Om het verloop van een tijdreeks te voorspellen moet ook rekening gehouden worden met het tijdsverloop. Zo zal het verloop van de vorige dagen een invloed hebben op het verloop van de volgende dagen. Dus voor deze toepassing zullen recurrente neurale netwerken gebruikt worden aangezien voorwaarts gerichte neurale netwerken geen rekening houden met de voorgaande inputwaarden.

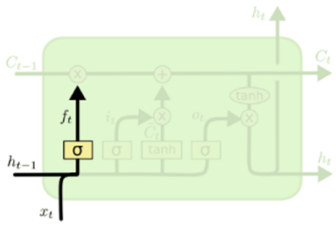
Het probleem bij gewone recurrente neurale netwerken is dat deze moeilijk patronen herkennen op lange termijn. Hiervoor biedt de long short term memory variant van recurrente neurale netwerken een oplossing. Dit zorgt ervoor dat enkel de data die bijgehouden moet worden uit vorige iteraties een impact zal hebben op nieuwere resultaten. De werking van een LSTM wordt voorgesteld op figuur 2.6.

Figuur 2.6: Gedetaileerdere figuur van een cel uit een neuraal netwerk dat informatie interpreteert uit een andere cel van het neurale netwerk (Olah, 2015)



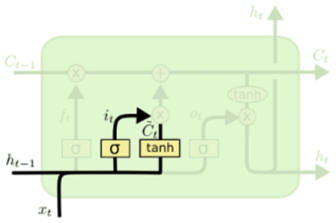
Dit aan de hand van 4 stappen:

Figuur 2.7: Grafische weergave van de forget layer binnen het neuron (Olah, 2015)



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figuur 2.8: Grafische weergave van de input gate layer binnen het neuron (Olah, 2015)



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

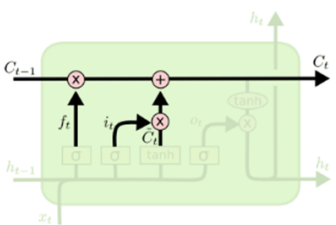
Stap 1 Eerst wordt bepaald welke info weggegooid mag worden in de “forget layer”. Deze zal een waarde tussen 0 en 1 uitvoeren waarbij een 0 zal betekenen dat deze waarde volledig genegeerd mag worden voor elke component van de cell state en waarbij een 1 zal betekenen dat alles behouden zal moeten worden.

Stap 2 Voor de volgende stap zal in de “input gate layer” bepaald worden welke waarden geüpdatet moeten worden aan de hand van een sigmoïde funtie die hier zal resulteren in i_t. Daarna zal de vector geconstrueerd worden met de potentiële waarden $\tilde{C}_t$.

Stap 3 In deze stap zullen we de voorgaande celwaarde vernieuwen. Zo zullen door C_{t-1} te vermenigvuldigen met f_t de overbodige waarden vergeten worden. En door het product van i_t en C_t hierbij op te tellen zullen de achterhaalde waarden geüpdatet worden.

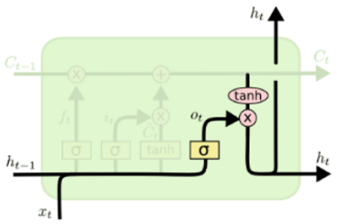
Stap 4 Als laatste zal de uitvoer bepaald worden. Dit is een gefilterde versie van de celtoestand. Die filtering gebeurt aan de hand van een sigmoïde functie. Het resultaat hiervan wordt dan vermenigvuldigd met de tanh van de celtoestand. Met deze formule

Figuur 2.9: Grafische weergave van het 2^{de} deel van de input gateway (Olah, 2015)



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figuur 2.10: Grafische weergave van het onderdeel van de cel waarin de input vermenigvuldigd wordt met de tanh van de celtoestand om de outputwaarde te bekomen (Olah, 2015)



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

verkrijgen we de uitvoerwaarde die de cyclus kan verderzetten.

2.3.2 Praktische toelichting

Deze praktische toelichting is gebaseerd op het artikel van Jason Brownlee (Brownlee, 2018c) en zal dienen als basis voor het opstellen van een LSTM netwerk.

Univariate LSTM modellen

Univariate LSTM models maken voorspellingen voor data waarbij er slechts 1 afhankelijke variabele is. Hierbij bestaan de problemen uit een reeks observaties. Op basis van vorige waarden in een reeks zal de volgende waarde in diezelfde reeks voorspeld worden. In deze subsectie zullen de voorgestelde modellen toegepast worden op eenstaps univariate tijdreeksen maar deze kunnen makkelijk aangepast worden om gebruikt te worden bij andere soorten tijdreeksen.

Datavoorbereiding Een tijdreeks zelf bestaat uit een sequentie van waarden zoals hieronder bijvoorbeeld.

Listing 2.1: Originele datasequentie

```
[10, 20, 30, 40, 50, 60, 70, 80, 90]
```

Om deze waarden te gebruiken voor het trainen van het neuraal netwerk zal deze tijdreeks omgevormd moeten worden naar samples. De dimensies van deze samples zullen bepalen hoeveel inputwaarden en outputwaarden zullen gebruikt worden bij het model. Aangezien we bij dit voorbeeld met eenstapsvoorspellingen werken zal elk sample over 1 outputwaarde beschikken. Bij dit voorbeeld worden 3 inputwaarden genomen. Dit zal als gevolg hebben dat de samples van de gegeven datasequentie er als volgt zullen uitzien.

Listing 2.2: Samples van de gegeven datasequentie

```
X,                  y
10, 20, 30          40
```

```
20, 30, 40          50
30, 40, 50          60
...
```

Vanilla LSTM

Een Vanilla LSTM is een LSTM model dat over één enkele verborgen laag beschikt en een uitvoerlaag die een voorspelde waarde zal aangeven. In deze verborgen laag zullen zich 50 nodes bevinden. De hoeveelheid te kiezen nodes is afhankelijk van de data. Zo zullen meer nodes de foutmarge verkleinen maar een kleiner aantal nodes zal meer gaan generalizeren, wat het eindresultaat ook zou kunnen verbeteren. Uiteindelijk moet hier een evenwicht in gevonden worden.

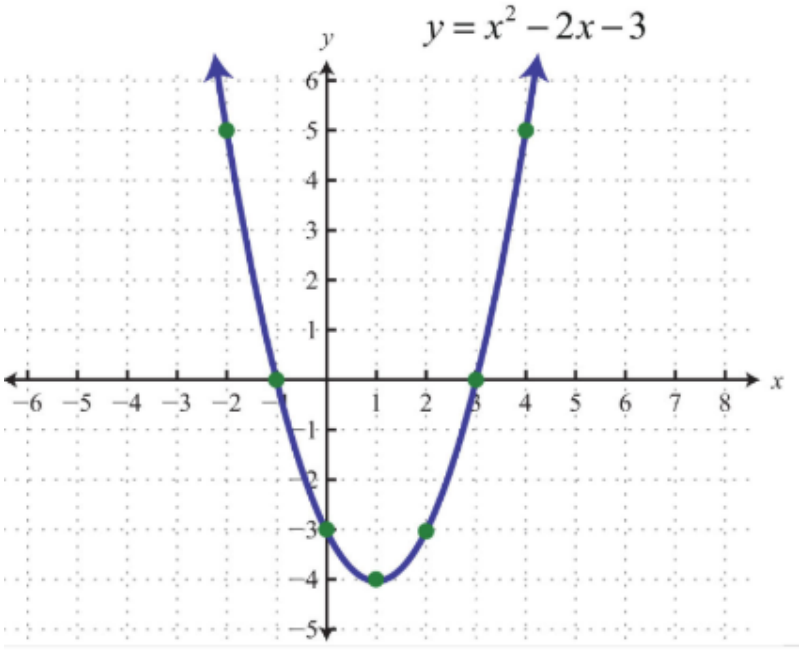
Om te bepalen of een signaal binnen een neuron effectief sterk genoeg zal zijn om te activeren en er zelf een te sturen zal de relu activatiefunctie gebruikt worden. Dit is de meest gebruikte activiatiefunctie in neurale netwerken (Liu, 2020) en heeft als voordeel dat het er weinig complexe wiskunde aan te pas komt bij de berekening waardoor het performant is. Daarnaast convergeert het ook sneller. Door de lineariteit zal de helling niet afvlakken waneer x vergroot. Tenslotte is deze functie ook ijl wat inhoudt dat er altijd iets van activatie is, specifiek bij de relu functie alle waarden onder 0 liggen. Ijlheid komt de efficiëntie van het model ten goede omdat hierdoor bepaalde neuronen enkel geactiveerd zullen worden bij specifieke prikkels. Zo zou een netwerk voor image recognition een bepaald neuron kunnen hebben dat zal activeren indien er een kattenoor waar te nemen valt, maar het zou niet wenselijk zijn moest dit neuron actief zijn indien er een afbeelding van een gebouw wordt getoond.

De gewichten van relaties tussen de neuronen in het neuraal netwerk zullen bepaald worden aan de hand van de Adam versie van stochastische gradient descent. Om stochastische gradient descent te definiëren zal eerst toegelicht moeten worden wat gradient descent precies inhoudt. Gradient staat voor de afgeleide van de verlies functie en descent betekent afdaling. De combinatie van deze 2 begrippen houdt in dat men zal afdalen tot het laagste punt bereikt wordt (Srinivasan, 2019).

We kunnen dit principe makkelijkst voorstellen aan de hand van een parabool. Deze wordt grafisch weergegeven op figuur 2.7. Het minimum van deze parabolische functie wordt bereikt wanneer x een waarde van 1 heeft. Als startpunt wordt een willekeurige waarde op de curve genomen waarbij dan nagegaan wordt in welke richting bewogen moet worden om dichter bij het minimum te komen. Wanneer het minimum gezocht wordt zou een stapgrootte gedefinieerd kunnen worden en dan telkens met even grote stappen afgedaald kunnen worden. Wanneer deze stapgrootte bepaald moet worden zou bij een kleine stapgrootte het minimum nauwkeuriger gedefinieerd zijn maar zou dit veel meer berekeningen vergen. Bij een grote stapgrootte zou het minimum snel berekend kunnen worden, maar het zou onnauwkeurig zijn.

Bij gradient descent wordt deze stapgrootte dynamisch bepaald op basis van het product van de hellingsgraad en de learning rate, een vooraf bepaalde waarde die moet voorkomen

Figuur 2.11: Parabolische functie



dat de stapgrootte te groot wordt. Wanneer de hellingsgraad laag zal zijn betekent dit dat het minimum wordt benaderd en dus kleinere stappen genomen zullen worden om dit zo nauwkeurig mogelijk te bepalen. De kans dat 0 effectief wordt bereikt is zeer klein, daarom stopt het algoritme wanneer een vooraf bepaalde minimale stapgrootte wordt bereikt. Daarnaast kan ook een maximum aantal stappen ingesteld worden waarbij het algoritme stopt.

Gradient descent kan duidelijk toegelicht worden aan de hand van deze 5 stappen, bij elke stap moet de waarde van de afgeleide opnieuw berekend worden:

1. Neem de afgeleide van de verliesfunctie voor elke parameter in deze functie
2. Kies willekeurige waarden voor deze parameters
3. Voer de parameterwaarden in in de afgeleiden (in praktijk zal de gradiënt numerisch bepaald worden)
4. Bereken de stapgroottes: Stapgrootte = helling * learning rate
5. Bereken de nieuwe parameters: Nieuwe parameter = oude parameter - stapgrootte

Herhaal stappen 3 tot 5 tot het minimum bereikt wordt.

Bij dit voorbeeld zou gewoon de afgeleide van deze parabool kunnen genomen worden om zo tot het minimum te komen, maar bij complexere functies is dit niet mogelijk en biedt gradient descent hiervoor een oplossing.

Bij gebruik van stochastic gradient descent wordt niet telkens alle data gebruikt wanneer de parameters herberekend worden, maar een willekeurige subset uit de dataset om het

aantal berekeningen te reduceren wat deze methode performanter maakt zonder al te veel nauwkeurigheid te verliezen (Starmer, 2019).

Dan rest enkel Adam nog verklaard te worden. Dit is een variant op de klassieke vorm van gradient descent waarbij de learning rate aangepast voor elk gewicht binnen het netwerk en doorheen het trainingsproces.

Deze aanpassingen worden door de auteurs van Adam omschreven als het combineren van 2 andere uitbreidingen op gradient descent namelijk:

Adaptive Gradient Algorithm (AdaGrad) die er voor zorgt dat een per-parameter learning rate die performantie zal verbeteren van de gradiënt.

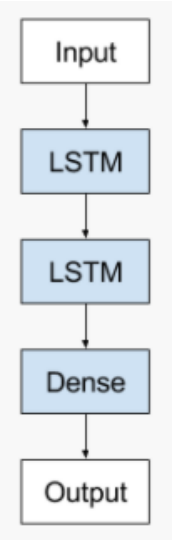
Root Mean Square Propagation (RMSProp) deze uitbreiding op gradient descent zorgt voor aanpassingen in de learning rates op basis van de schaal van de laatste gradiënten kortom hoe snel deze veranderen (Brownlee, 2017b).

En zal geoptimaliseerd worden op basis van de 'mse' verliesfunctie.

Listing 1: Datavoorbereiding dataset ijsexpansie

```
# univariate lstm voorbeeld
from numpy import array
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense

# Opslitsen van univariate sequentie in samples
def split_sequence(sequence, n_steps):
X, y = list(), list()
for i in range(len(sequence)):
# vinden van het einde van dit patroon
end_ix = i + n_steps
# nagaan of we ons na de sequentie bevinden
if end_ix > len(sequence)-1:
break
# verwerven van invoer en uitvoerdelen van het patroon
seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
X.append(seq_x)
y.append(seq_y)
return array(X), array(y)

# Bepalen van de sequentie
raw_seq = [10, 20, 30, 40, 50, 60, 70, 80, 90]

# bepalen van het aantal tijdsstappen
n_steps = 3

# onderverdelen in samples
X, y = split_sequence(raw_seq, n_steps)

# reshape from [samples, timesteps] into [samples, timesteps, features]
n_features = 1
```

Figuur 2.12: Een stacked LSTM (Brownlee, 2017c)




```
X = X.reshape((X.shape[0], X.shape[1], n_features))

# definieer model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(n_steps, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

# fit model
model.fit(X, y, epochs=200, verbose=0)

# demonstreer voorspelling
x_input = array([70, 80, 90])
x_input = x_input.reshape((1, n_steps, n_features))
yhat = model.predict(x_input, verbose=0)
print(yhat)
```

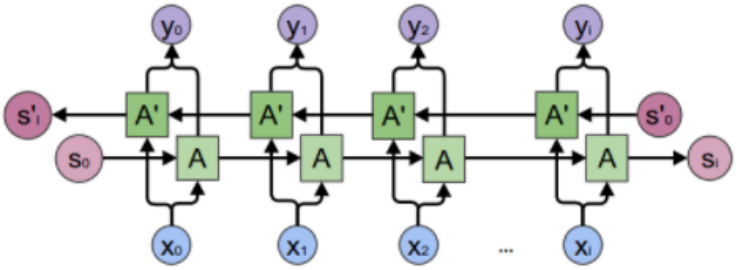
Stacked LSTM

Een stacked LSTM is een variant op het Vanilla LSTM waarbij er meerdere LSTM lagen na elkaar geplaatst worden. Het toevoegen van een extra laag zal voor een langere uitvoeringstijd zorgen. Het voordeel hiervan is dat hierdoor meerdere onderdelen makkelijker geregistreerd kunnen worden en in verband met elkaar gebracht kunnen worden. Het stapelen van LSTM lagen zal ervoor zorgen dat deze lagen ook onderling uitvoerwaarden zullen uitwisselen die een sequentie van tijdsstappen zal beschrijven en niet gewoon 1 enkele tijdstap (Brownlee, 2017c).

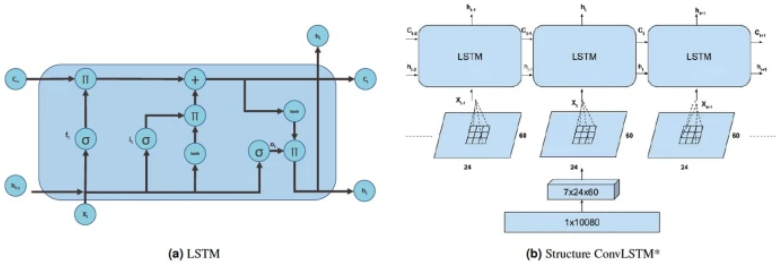
Bidirectionele LSTM

Bidirectionele LSTMs kunnen getrained worden gebruik makend van alle beschikbare invoerinformatie uit het verleden en de toekomst binnen een specifieke tijdspanne. Dit door middel van de toestandsneuronen op te splitsen van een normaal Recurrent Neuraal Netwerk in een deel dat verantwoordelijk is voor de positieve tijdsdirectie en een deel voor de negatieve tijdsdirectie. (Brownlee, 2017a)

Figuur 2.13: Weergave van een bidirectioneel LSTM model (**Colah2015**)



Figuur 2.14: **(a)** Gewone LSTM cell, **(b)** Weergave van een ConvLSTM (Syed Ashiqur Rahman, 2019)



Bidirectionele LSTMs worden voornamelijk gebruikt bij language processing omdat de definitie van een woord afgeleid kan worden uit de context. De begrippen die bepalend zijn voor de definitie kunnen zich zowel voor als achter het woord zelf bevinden dus moet de zin vanuit 2 richtingen geanalyseerd kunnen worden om de definitie te achterhalen.

CNN LSTM

Een convolutioneel neuraal netwerk ook wel een CNN genoemd wordt voornamelijk gebruikt bij het analyseren van tweedimensionale grafische data. Een CNN is ook sterk in het extraheren en aanleren van features van eendimensionele data. Een CNN model kan ook gecombineerd worden met een LSTM model om een CNN-LSTM te vormen. Hierbij wordt het CNN gebruikt om subreeksen te interpreteren die op zich dan weer zullen geïnterpreteerd worden als een reeks door het LSTM model.

ConvLSTM

ConvLSTM is ook een soort van CNN-LSTM waarbij het convolutionele deel rechtstreeks is ingebouwd binnen het LSTM gedeelte. Dus ook dit CNN-LSTM werd ontwikkeld voor het interpreteren van tweedimensionale grafische data maar kan aangepast worden voor gebruik bij univariabele tijdreeksvoorspellingen.

Multivariabele LSTM modellen

Bij multivariabele tijdreeksen zullen er in tegenstelling tot univariabele tijdsreeksen wel meerdere observaties zijn per tijdseenheid. Deze worden dan nog eens onderverdeeld in multiple input series en multiple parallel series.

Meerdere invoerreeksen

Bij de variant met meerdere invoerreeksen zullen er meerdere invoerwaarden zijn voor elk tijdstip. Hierbij zal slechts voor 1 van de invoerreeksen een voorspelling gemaakt worden.

Meerdere parallelle reeksen

Bij de LSTM variant waarbij er meerdere parallelle reeksen zijn zullen er meerdere invoerwaarden zijn voor elk tijdstip. Hier zal er voor elke reeks een waarde voorspeld worden.

Multi-Step LSTM modellen

Tot nu toe werden enkel maar one-step LSTM modellen toegelicht waarbij de invoerreeks slechts zal resulteren in 1 uitvoerwaarde. Bij multi-step LSTM modellen zal er meer dan 1 uitvoerwaarde voorspeld worden.

Vector output model

Bij dit type multi-step LSTM model zal de uitvoer onder de vorm van een vector weergegeven worden die de resultaten van de volgende tijdstappen zal weergeven en niet enkel van de volgende tijdstap.

Encoder-Decoder Model

Dit model is ontwikkeld om sequenties met variabele uitvoer te voorspellen (Brownlee, 2017a). Het is ook ontworpen om problemen waarbij er zowel invoer-als uitvoersequenties zijn te behandelen. Deze problemen worden ook wel benoemd als sequence-to-sequence of seq2seq-problemen. Deze techniek wordt voornamelijk gebruikt bij vertalingstoepassingen maar kan ook toegepast worden op tijdreeksen. Het model bestaat uit 2 hoofdonderdelen de encoder en de decoder.
De encoder is verantwoordelijk voor het lezen en interpreteren van de invoerreeks. De encoder zal dan een vector met een vaste lengte uitvoeren die een interpretatie zal geven van de invoerreeks. Normaal zal de encoder een Vanilla LSTM model zijn, maar er kan even goed een stacked, bidirectioneel of CNN model gebruikt worden.
De decoder zal de uitvoer van de encoder dan gebruiken als invoer. Die zal dan voor elke tijdsstap een uitvoerwaarde zal geven. Deze uitvoer zal dan nog eens geïnterpreteerd worden door een verborgen laag alvorens dit volledig model de multi-step uitvoerreeks zal teruggeven.

Multivariate Multi-Step LSTM Models LSTM modellen

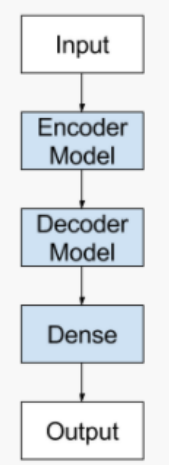
Multiple input Multi-Step Output

Wanneer men over een dataset beschikt met meerdere invoerparameters maar slechts 1 enkele parameter voor meerdere tijdsstappen dient te voorspellen spreekt men over multiple input multi-step output.

Multiple Parallel Input and Multi-Step Output

Wanneer de dataset waarvan een voorspelling gemaakt moet worden meerdere invoerparameters heeft maar er ook meerdere voorspeld moeten worden zal dit benoemd worden als een multiple parallel input and multi-step output LSTM.

Figuur 2.15: Encoder-Decoder Model (Brownlee, 2017a)



2.4 Prophet

Prophet is een procedure om tijdreeksen te voorspellen ontwikkeld door Facebook. („Forecasting at scale." 2020) Deze procedure baseert zich op een additief model waar niet-lineaire trends gefit worden aan jaarlijkse, wekelijkse en dagelijkse seizoensgebondenheid hierbij zal ook rekening gehouden worden met zogenaamde "holiday effects", fluctuaties in de data die veroorzaakt worden door vakantieperiodes. Deze techniek zal best werken met tijdreeksen die een sterk seizoensgebonden effect hebben en waarvan ook meerdere seizoenen aan historische data beschikbaar zal zijn. Daarnaast houdt Prophet ook rekening met ontbrekende data, verschuivingen in de trend en zal het ook rekening houden met uitschieters.
Het is eenvoudig om te gebruiken maar er zijn ook optimalisaties mogelijk indien men het model wil finetunen. Prophet is zowel beschikbaar in R als in Python

2.5 Validatietechnieken

Om de kwaliteit van de modellen te beoordelen zal er telkens een deel van de dataset achtergehouden worden die niet zal gebruikt worden bij het opstellen van het model. Deze achtergehouden dataset wordt de testset genoemd. In het geval van een tijdreeks zullen dit altijd de laatste waarden zijn. Het model zal dan trachten deze laatste waarden te voorspellen waarna de voorspelde waarden vergeleken zullen worden met de waarden van de testset. Op basis hiervan kunnen we achterhalen hoe groot de fout is bij elk model, het model met de laagste fout zal dan de meest accurate voorspelling maken. In deze sectie zullen enkele verschillende methodes om de fout te berekenen toegelicht worden (Rob J Hyndman, 2018).

2.5.1 Foutmaten

MAE

MAE staat voor mean absolute error. Deze zal berekend worden door het verschil te nemen tussen elke voorspelde tijdstap en de verwachte waarde voor deze tijdstap, hier de absolute waarde van te nemen en dan het gemiddelde te nemen van al deze absolute waarden. De formule voor de gemiddelde absolute fout wordt hieronder weergegeven.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{2.21}$$

Dit is de meest eenvoudige foutmaat en wordt vaak gebruikt al moet er echter mee rekening gehouden worden dat deze fout eenheidsgebonden is en kan deze niet gebruikt worden om tijdreeksen te vergelijken die gebruik maken van andere eenheden. Dit heeft wel als voordeel dat deze fout makkelijk te interpreteren valt. Zo zou bijvoorbeeld een fout van 1 bij een tijdreeks over temperatuur in graden Celsius betekenen dat de gemiddelde fout bij het voorspellen van de temperatuur 1 graad Celsius is bij dat model.

RMSE

RMSE staat voor root mean squared error. Deze wordt berekend door het verschil te nemen tussen tussen elke voorspelde tijdstap en de verwachte waarde bij deze tijdstap, dit te kwadrateren, het gemiddelde te nemen van deze kwadraten en dan de vierkantswortel te nemen van dit gemiddelde. Dit kan ook als volgt geformuleerd worden:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}} \tag{2.22}$$

Ook bij RMSE zal de eenheid gelijk blijven. Het verschil met MAE is echter dat een groot verschil tussen de voorspelde en effectieve waarde zwaarder zal doorwegen bij RMSE doordat deze eerst gekwadrateerd worden voordat het gemiddelde genomen wordt (Kampakis, 2020).

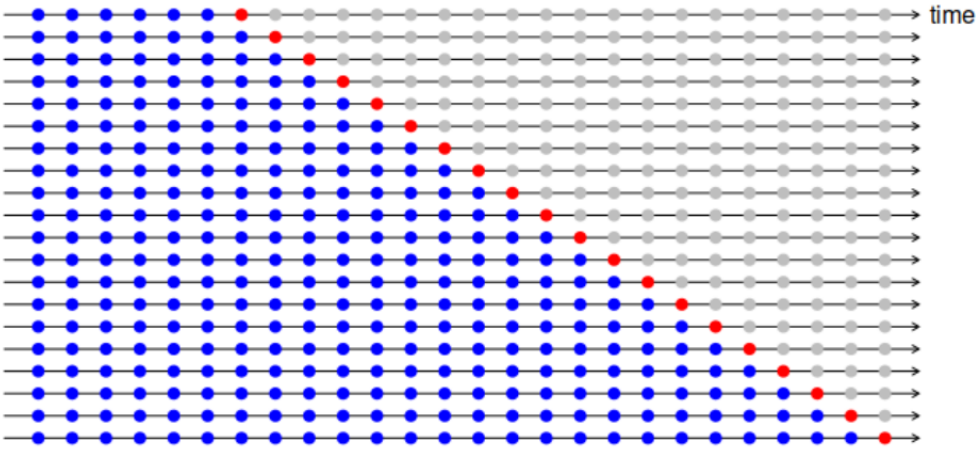
MAPE & RMSPE

MAPE staat voor mean absolute percentage error en RMSPE voor root mean square percentage error. Dit zijn procentuele fouten, dit type fout kan vergeleken worden tussen tijdreeksen met een verschillende eenheid. Dit zijn variaties op respectievelijk MAE en RMSE maar waarbij deler toegevoegd wordt bij het berekenen van het verschil tussen de voorspelde en de effectieve waarde. De formules van deze fouten worden hieronder weergegeven.

$$MAPE = \frac{\sum_{i=1}^{n} |\frac{y_i - \hat{y}_i}{y_i}|}{n} \tag{2.23}$$

$$RMSPE = \sqrt{\frac{\sum_{i=1}^{n} (\frac{y_i - \hat{y}_i}{y_i})^2}{n}} \tag{2.24}$$

Figuur 2.16: Grafische weergave van de structuur van een cross-validation tijdreeks (Rob J Hyndman, 2018) waarbij 1 tijdreeks zal opgedeeld worden in 20 train- en testreeksen



2.5.2 Cross-validation

Om een tijdreeks nog beter te benutten kan ook cross-validation gebruikt worden. Hierbij zal dezelfde tijdreeks meerdere malen gebruikt worden om het model te trainen, maar zal deze telkens uitgebreid worden zoals zichtbaar in figuur 2.17. De accuraatheid van het model zal dan bepaald worden door de het gemiddelde te nemen van de accuraatheid voor elke opgestelde testset. Er moet echter rekening mee gehouden worden dat deze nieuwe training en testsets altijd nieuwe observaties moeten bevatten en de observaties van de trainingsset altijd moeten voorkomen voor deze van de bijhorende testset (Shrivastava, 2020).

3. Methodologie

Listing 2: Datavoorbereiding dataset ijsexpansie

```
ice2 = pd.read_csv('./data/seaice2.csv')
# ice2
ice2_mean = ice2.mean()[1:-2]
# ice2_mean
ice2_mean.index = ice2_mean.index.values.astype(int)

plt.title('Yearly ice extent')
plt.scatter(ice2_mean.index,ice2_mean)
plt.xlabel('Years')
plt.ylabel('Extent')
plt.show()

# ice2['2018']
# pd.concat([ice2['2016'],ice2['2017'],ice2['2018'],ice2['2019']]).reset_index()[0]
# ice2[['2018']].append(ice2[['2019']])
ice2.rename(columns={'Unnamed: 0' : 'Month', 'Unnamed: 1' : 'Day'}, inplace = True)
ice2.drop([' ','1981-2010','Day','1978','2020'],axis=1,inplace=True)
values = ice2.values
i = 0
for row in values :
if type(row[0]) != str :
values[i][0] = month
else:
month = row[0]
i = i +1
# ice2.columns.values
ice2_clean = pd.DataFrame(values)
ice2_clean.columns = ice2.columns.values
```

```
29 # ice2_clean.head(5)
30 ice2_monthly_mean =
    ↪ ice2_clean.set_index('Month').astype(float).groupby('Month',sort=False).mean()
31 # ice2_monthly_mean
32 # ice2_monthly_mean.T.stack().index.get_level_values(0)
33 #
    ↪ ice2_monthly_mean.T.stack().reset_index(level=['Month']).drop(columns=['Month'])
34 ice2_monthly_mean_chron =
    ↪ ice2_monthly_mean.T.stack().reset_index(level=['Month']).drop(columns=['Month'])
35 # ice2.columns.size
36 plt.title('Monthly ice extent')
37 plt.plot(ice2_monthly_mean_chron.values)
38 plt.xticks(np.array(range(0,500,75)))
39 plt.xlabel('Cumulative month')
40 plt.ylabel('Extent')
41 plt.show()
42
43 # np.unique(ice2_monthly_mean_chron.index.values).size*12
44 print('from ' + ice2_monthly_mean_chron.index.values[0] + ' until ' +
    ↪ ice2_monthly_mean_chron.index.values[-1])
45 ice2_monthly_mean_chron =
    ↪ ice2_monthly_mean.T.stack().reset_index(level=['Month']).drop(columns=['Month'])
46 ice2_monthly_mean_chron.columns = ['ice_extent']
47 ice2_monthly_mean_chron
```

Bibliografie

Brown, S. (2020). Least Squares — the Gory Details. Verkregen van https://brownmath.com/stat/leastsq.htm

Brownlee, J. (2017a). Encoder-Decoder Long Short-Term Memory Networks. Verkregen 24 november 2020, van https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/

Brownlee, J. (2017b). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. Verkregen 24 november 2020, van https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Brownlee, J. (2017c). Stacked Long Short-Term Memory Networks. Verkregen van https://machinelearningmastery.com/stacked-long-short-term-memory-networks/#:~:text=A%20Stacked%20LSTM%20architecture%20can,for%20all%20input%20time%20steps.

Brownlee, J. (2018a). 11 Classical Time Series Forecasting Methods in Python (Cheat Sheet). Verkregen 24 november 2020, van https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/

Brownlee, J. (2018b). A Gentle Introduction to SARIMA for Time Series Forecasting in Python. Verkregen 24 november 2020, van https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/

Brownlee, J. (2018c). How to Develop LSTM Models for Time Series Forecasting. Verkregen 20 mei 2020, van https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Forecasting at scale. (2020). Verkregen van https://facebook.github.io/prophet/

Kampakis, S. (2020). Performance measures: RMSE and MAE. Verkregen van https://thedatascientist.com/performance-measures-rmse-mae/

Kenton, W. (2020). Multiple Linear Regression (MLR). Verkregen 24 november 2020, van https://www.investopedia.com/terms/m/mlr.asp

Lau, R. (2020). Introduction to ARIMA: nonseasonal models. Verkregen van https://people.duke.edu/~rnau/411arim.htm

Lievens, S. (2018). Artificiële Intelligentie, lesnota's HOGENT, "147–149".

Liu, D. (2020). A Practical Guide to ReLU. Verkregen 24 november 2020, van https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7

Olah, C. (2015). Understanding LSTM Networks. Verkregen 26 mei 2020, van https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Rob J Hyndman, G. A. (2018). Forecasting: Principles and Practice. Verkregen van https://otexts.com/fpp2/

Shrivastava, S. (2020). Cross Validation in Time Series. Verkregen van https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4

Sinha, P. (2019). LINEAR REGRESSION,LEAST-SQUARES REGRESSION,OUTLIERS AND INFLUENTIAL OBSERVATIONS,RESIDUALS,LURKING VARIABLES,EXTRAPOLATION. Verkregen van https://cafepharmablog.wordpress.com/2019/02/20/linear-regressionleast-squares - regressionoutliers - and - influential - observationsresidualslurking - variablesextrapolation/

Srinivasan, A. V. (2019). Stochastic Gradient Descent — Clearly Explained !! Verkregen 24 november 2020, van https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31

Starmer, J. (2019). Gradient Descent, Step-by-Step. Verkregen van https://www.youtube.com/watch?v=sDv4f4s2SB8&ab_channel=StatQuestwithJoshStarmer

Swaminathan, S. (2018). Logistic Regression — Detailed Overview. Verkregen van https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

Syed Ashiqur Rahman, D. A. A. (2019). Deep Learning using Convolutional LSTM estimates Biological Age from Physical Activity. Verkregen van https://www.nature.com/articles/s41598-019-46850-0