UHASSELT
KNOWLEDGE IN ACTION

Maastricht University

Faculteit Wetenschappen
School voor Informatietechnologie

master in de informatica

Masterthesis

Interactieve exploratie en kwantitatieve evaluatie van anomaliedetectie in tijdreeksen

Wouter Pardon

Scriptie ingediend tot het behalen van de graad van master in de informatica

PROMOTOR :

Prof. dr. Stijn VANSUMMEREN

BEGELEIDER :

dr. Joris GILLIS

Dhr. Michiel JACOBS

2022
2023

De transnationale Universiteit Limburg is een uniek samenwerkingsverband van twee universiteiten in twee landen: de Universiteit Hasselt en Maastricht University.

UHASSELT
KNOWLEDGE IN ACTION

www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

Maastricht University

Faculteit Wetenschappen
School voor Informatietechnologie

master in de informatica

Masterthesis

Interactieve exploratie en kwantitatieve evaluatie van anomaliedetectie in tijdreeksen

Wouter Pardon

Scriptie ingediend tot het behalen van de graad van master in de informatica

PROMOTOR :
Prof. dr. Stijn VANSUMMEREN

BEGELEIDER :
dr. Joris GILLIS
Dhr. Michiel JACOBS

Dankwoord

Allereerst wil ik dr. Gillis en de heer Jacobs bedanken voor hun bijdrage. Ik heb veel interessante dingen geleerd en voornamelijk hoe anomaliedetectie in de praktijk bij TrendMiner wordt ingezet. Zonder hen zou deze thesis ook niet tot stand zijn gebracht. Het is fijn om te weten hoe grotere bedrijven onderzoek voeren en deze technologieën in de praktijk gebruiken. Als laatste wil ik ook prof. dr. Vansummeren bedanken voor zijn uitstekende begeleiding doorheen het jaar. Ik wil elk van deze personen bedanken voor de goede begeleiding en de nodige feedback om deze proef tot stand te brengen.

Samenvatting

In de laatste jaren is het onderzoek naar anomaliedetectie fors gestegen omdat er meer data ter beschikking zijn en ze door de technologie toegankelijker zijn om te verwerken. Anomaliedetectie is het detecteren van één of meerdere anomalieën. Een anomalie is een observatie of een reeks van observaties die sterk afwijken van het normaal.

Zulke observaties hebben twee mogelijke betekenissen: gewild of ongewild. In het eerste geval gaat men specifiek opzoek naar gewilde anomalieën om deze beter te begrijpen en eventueel acties uit te voeren. Bijvoorbeeld bij het meten van temperaturen doorheen de dag kan een plotselinge verandering in de temperatuur aantonen dat de sensor beschadigd is of er een storm op komst is.

In het andere geval worden ze uit de dataset verwijderd omdat ze geen verdere betekenis hebben of zelfs bias creëren voor de algoritmes die deze observaties gebruiken. Bijvoorbeeld bij het voorspellen van het weer voor de komende dagen zullen anomalieën, die geproduceerd zijn door beschadigde sensoren, de voorspellingen negatief beïnvloeden.

Door de stijgende populariteit van dit topic zijn er al heel wat algoritmes ontwikkeld die anomalieën binnen een tijdreeks kunnen detecteren. Een tijdreeks of timeseries is een reeks van waarden die gemeten zijn doorheen de tijd. Dit kan bijvoorbeeld de temperatuur zijn die elke dag gemeten wordt doorheen het hele jaar. Het is niet van zelfsprekend om elk algoritme te begrijpen. Daarom is het belangrijk dat er op een gebruiksvriendelijke visuele manier duidelijk gemaakt wordt hoe dergelijke algoritmes anomalieën detecteren. In deze thesis worden enkele algoritmes onderzocht alsook een implementatie die helpt om de gedetecteerde anomalieën, door elk van de algoritmes, beter te begrijpen.

Inhoudsopgave

1 Inleiding **6**

2 Achtergrond **8**
2.1 Tijdreeksen. 8
2.2 Kiezen van het correcte algoritme 8
2.2.1 Soorten tijdreeksen 8
2.2.2 Soorten anomalieën 10
2.2.3 Soorten algoritmes 11

3 Datasets **17**
3.1 Numenta Anomaly Benchmark (NAB) 17
3.2 Skoltech Anomaly Benchmark (SKAB) 18
3.3 GutenTAG 18
3.3.1 Yahoo-S5 22

4 Algoritmes **23**
4.1 Preprocessing 23
4.2 K-Means 25
4.2.1 Preprocessing van een tijdreeks 25
4.2.2 Clustering 26
4.2.3 Detecteren van anomalieën 27
4.3 ARIMA 29
4.3.1 Preprocessing van een tijdreeks 29
4.3.2 ARIMA-model 30
4.3.3 Detecteren van anomalieën 32
4.4 DWT-MLEAD 34
4.4.1 Frequentieanalyse in tijdreeksen 34
4.4.2 Nadelen van een Fourier-transformatie 35
4.4.3 Discrete wavelet transformatie (DWT) 37
4.4.4 Sliding windows 42
4.4.5 Maximum Likelihood Estimation 43
4.4.6 Anomaliedetectie 47
4.4.7 Padding 51

5 Resultaten **52**
5.1 Bepalen van de parameters 52
5.1.1 DWT-MLEAD 52
5.1.2 ARIMA 53
5.1.3 K-Means 53
5.2 Metrieken 55
5.2.1 TP, FP, FN, TN 55
5.2.2 Precision & recall 55
5.2.3 ROC 57

5.2.4 AUC 58
5.3 Overzicht van de resultaten 59
5.4 Conclusie van de resultaten 67

6 Exploratietool **68**
6.1 Metrieken voor individuele instanties 69
6.1.1 IBA - Index of Balanced Accuracy 69
6.1.2 F1 69
6.2 DWT-MLEAD 70
6.3 ARIMA 72
6.4 K-Means 75

7 Conclusies **77**

Hoofdstuk 1

Inleiding

Anomaliedetectie is het detecteren van observaties die sterk afwijken van alle andere. Dit speelt een cruciale rol in het identificeren van onregelmatigheden binnen complexe systemen, waardoor bedrijven vroegtijdig potentiële problemen en risico's kunnen detecteren. Voor grote bedrijven zoals TrendMiner vormt anomaliedetectie een essentieel onderdeel van hun activiteiten, aangezien ze worden geconfronteerd met grote hoeveelheden gegevens die voortdurend worden gegenereerd door diverse processen.

Bij dergelijke bedrijven is het onmogelijk om elk datapunt handmatig te bekijken en eventueel te markeren als anomalie. Daarnaast is het ook niet evident om elk soort anomalie met het blote oog te detecteren. Zelfs de kleinste afwijkingen kunnen ervoor zorgen dat heel het productieproces stilgelegd moet worden. Het vroegtijdig detecteren van anomalieën is van cruciaal belang. Indien een anomalie zich voordoet, kan men onmiddellijk tussenkomen waardoor de verloren kost in het productieproces drastisch verminderd wordt. Daarnaast is het ook belangrijk om te weten waarom een dergelijk algoritme een datapunt of sequentie van datapunten markeert als anomalie. Men zal meer inzicht krijgen in de types van anomalieën die door algoritmes worden gedetecteerd.

In deze masterproef worden er enkele unsupervised anomaliedetectie algoritmes onderzocht. Unsupervised algoritmes gebruiken, tijdens het trainen, als input een ongelabelde tijdreeks. Men hoeft niet elk type van anomalie te kennen om het model te trainen. Dit biedt enorme voordelen wanneer grote hoeveelheden data dienen verwerkt te worden en men niet elk soort anomalie kent. Elk van deze algoritmes detecteren verschillende soorten anomalieën binnen een tijdreeks. Anomalieën worden onderverdeeld in twee grote groepen namelijk puntanomalieën en sequentieanomalieën. Bij een puntanomalie wijkt er één punt af van de andere datapunten (observaties) in de nabije omgeving (lokaal) of van alle andere punten in de tijdreeks (globaal). Een sequentieanomalie is een reeks van datapunten die in hun geheel sterk afwijken van de andere datapunten.

Deze algoritmes worden geïmplementeerd in een exploratietool, die meer uitleg geeft waarom een algoritme datapunten markeert als anomalie en de andere niet. In deze tool is het mogelijk om een eigen gekozen univariate tijdreeks dataset te uploaden en te analyseren. Bij TrendMiner is het niet alleen belangrijk om te weten welke observaties anomalieën zijn, maar ook waarom deze anomalieën zijn. Dankzij deze informatie kunnen ze de klanten duidelijk informeren waar het is misgelopen in het productieproces en waarom.

De geïmplementeerde algoritmes worden getest op verschillende tijdreeksen uit verschillende benchmark datasets, om te controleren hoe goed deze algoritmes presteren op ongekende anomalieën. Om anomalieën te detecteren moet eerst het soort tijdreeks gekend zijn. Er zijn namelijk verschillende soorten tijdreeksen. Een tijdreeks met slechts één variabele is een univariate tijdreeks. Den voorbeeld is een tijdreeks bestaande uit temperaturen die gemeten zijn in het afgelopen jaar. Als een tijdreeks meerdere variabelen heeft spreekt men over een multivariate tijdreeks. Een accelerometer die veranderingen op de X-, Y- en Z-as meet, is hiervan een voorbeeld

Zoals eerder vermeld kan elk algoritme andere soorten anomalieën detecteren. Het is belangrijk te weten hoe anomaliedetectie algoritmes onderverdeelt zijn en hoe ze functioneren. De geïmplementeerde algoritmes moeten getest worden op verschillende tijdreeksen. Het is belangrijk om zorgvuldig de juiste tijdreeksen te kiezen om testresultaten te genereren, aangezien er meerdere soorten anomalieën bestaan. Daarnaast kan men niet eender welke test gebruiken om deze resultaten te genereren. Het is namelijk zo dat tijdreeksen voornamelijk normale data bevatten en weinig anomalieën, het is dus belangrijk om de juiste metrieken te kiezen.

Omdat het niet altijd duidelijk is waarom algoritmes datapunten markeren als anomalie, is er nood aan een exploratietool. Deze tool geeft visueel weer waarom datapunten gemarkeerd worden als anomalie of net niet. Elk algoritme wordt zorgvuldig bestudeerd om deze observaties te kunnen visualiseren.

Hoofdstuk 2

Achtergrond

2.1 Tijdreeksen.

Tijdreeksen bestaan uit een collectie van observaties die geordend zijn op tijd (Temporale ordening). In tegenstelling tot traditionele datasets (tabulaire gegevens) is dus de volgorde van deze observaties van belang. Als de volgorde van waardes binnen een tijdreeks verandert, wordt er een nieuwe dataset gemaakt met een volledig andere betekenis omdat deze niet meer op tijd is geordend. De betekenis van de data gaat hierdoor verloren.

Tijdreeksanalyse is het bestuderen van tijdreeksen om nuttige informatie te extraheren. In deze thesis ligt de nadruk op het detecteren van anomalieën binnen tijdreeksen. Een anomalie is een observatie die sterk afwijkt van alle andere observaties. De keuze van een anomaliedetectie algoritme hangt af van verschillende elementen [1].

2.2 Kiezen van het correcte algoritme

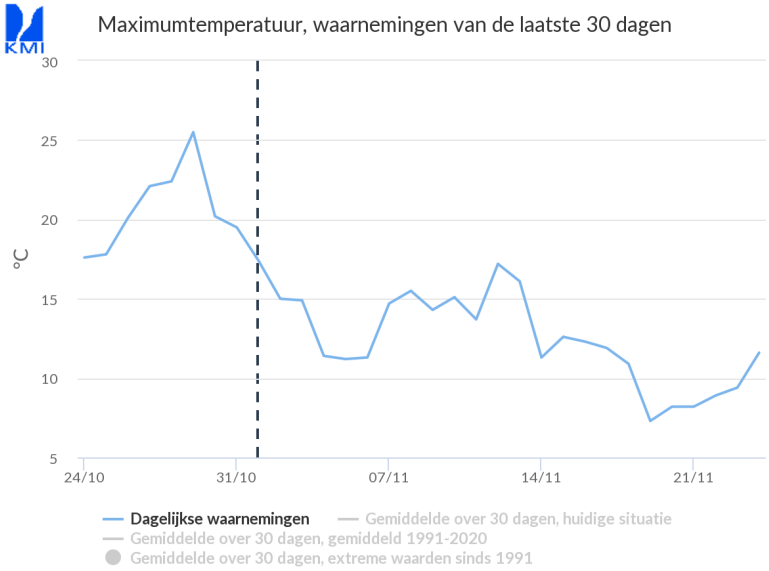
Het eerste element dat het algoritme definieert is het soort inputdata (tijdreeks). Dit kan univariate of multivariate zijn. Een ander belangrijk element is het soort gedetecteerde anomalie zoals: puntanomalie, sequentieanomalie of een gehele tijdreeks. Zowel de puntanomalieën als de sequentieanomalieën kunnen ook univariate of multivariate zijn. Het laatste element is de aard van het algoritme zelf. Algoritmes kunnen dienen om univariate anomalieën of multivariate anomalieën te detecteren. In de volgende paragrafen gaan we dieper in op deze elementen.

2.2.1 Soorten tijdreeksen

Het anomaliedetectie algoritme is afhankelijk van het soort tijdreeksdata. Zoals eerder vermeld kan een tijdreeks univariate of multivariate zijn. Deze onderscheiding is belangrijk bij het kiezen van het correcte algoritme.

Univariate tijdreeksen

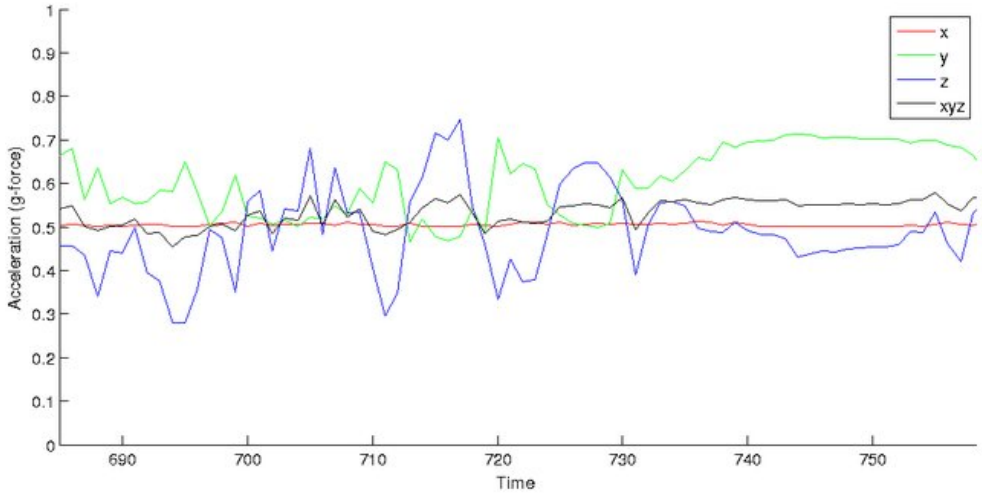
Univariate tijdreeksen bestaan uit een reeks observaties met slechts één variabele. Bijvoorbeeld het meten van de temperatuur op een bepaalde locatie. Deze observatie heeft slechts één variabele, namelijk de temperatuur van die locatie zoals die wordt weergegeven op figuur 2.1 [1].



Figuur 2.1: Univariate tijdreeks: maximumtemperatuur gemeten door het KMI [2].

Multivariate tijdreeksen

In tegenstelling tot univariate tijdreeksen hebben multivariate tijdreeksen meerdere variabelen die veranderen over tijd. Bijvoorbeeld een accelerometer die veranderingen op de X-, Y- en Z-as gaat meten zoals aangegeven op figuur 2.2. Dit soort tijdreeksen hebben andere algoritmes nodig om de data te analyseren. Elke variabele kan ook apart door een univariate algoritme worden gehaald, maar dan wordt de correlatie tussen de verschillende variabelen niet meegerekend. Het is ook mogelijk om *dimensionality* reductie toe te passen om slechts één variabele over te houden. Hier verliezen we dan ook weer belangrijke informatie van andere variabelen [1].

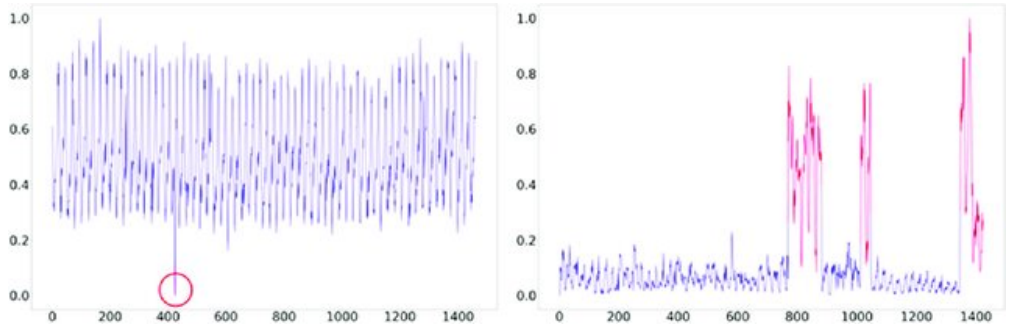


Figuur 2.2: Multivariate tijdreeks: data afkomstig van een accelerometer [3].

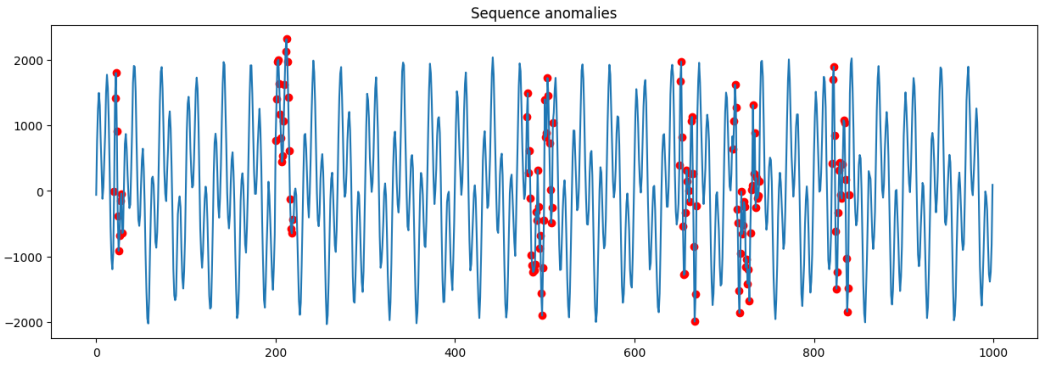
2.2.2 Soorten anomalieën

Een ander element om anomaliedetectie algoritme onder te verdelen is het soort anomalie dat gedetecteerd wordt. Een puntanomalie is één enkel punt dat fors afwijkt van alle andere punten (globaal) of van omringende punten (lokaal). Dit soort anomalietype kan ook weer zowel univariate zijn als multivariate.

Sequentieanomalie is een reeks van anomalieën die in hun geheel een anomalie vormen. Het verschil met een puntanomalie is dat elk individueel punt in een sequentie geen anomalie is maar eerder de sequentie in zijn geheel een anomalie is. Figuur 2.3 toont het verschil tussen puntanomalieën en globale sequentieanomalieën. Sequentieanomalieën kunnen dan ook weer lokaal zijn zoals in figuur 2.4.



Figuur 2.3: De linker figuur geeft een puntanomalie weer terwijl de rechter figuur een globale sequentieanomalie weergeeft. [4]



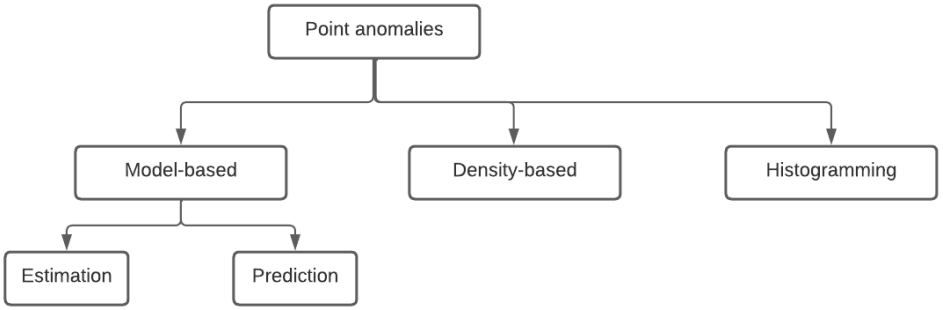
Figuur 2.4: Lokale sequentieanomalieën

Het laatste soort anomalieën is een volledige variabele die afwijkt van andere variabelen. Dit kan enkel gebeuren bij een multivariate tijdreeks omdat er één variabele is die in zijn geheel teveel afwijkt van andere variabelen. Bijvoorbeeld meerdere sensoren die een temperatuur gaan meten over verschillende locaties binnen één gemeente. Elke sensor vormt een variabele en als één van de variabele teveel afwijkt van alle andere variabelen dan kan men concluderen dat de metingen van deze sensor niet correct zijn. Zowel puntanomalieën als sequentieanomalieën kunnen univariate of multivariate zijn.

2.2.3 Soorten algoritmes

Anomaliedetectie algoritmes zijn ontworpen om specifieke soorten van afwijkingen te detecteren. Sommige algoritmes kunnen enkel toegepast worden voor het opsporen van anomalieën bij univariate tijdreeksen en andere bij multivariate tijdreeksen. Zoals eerder vermeld worden anomalieën onderverdeeld in twee grote groepen, namelijk puntanomalieën en sequentieanomalieën.

Puntanomalie algoritmes - univariate



Figuur 2.5: Types puntanomaliedetectie algoritmes voor univariate data [1]

De eerste manier om een anomalie te detecteren is om het huidige punt van het verwachtte punt af te trekken en deze waarde te gaan vergelijken met een vooraf berekende *threshold*. Algoritmes van deze vorm zijn op twee soorten modellen gebaseerd zoals weergegeven in tabel 2.1. Figuur 2.5 geeft een overzicht van methodes voor puntanomalieën.

Tabel 2.1: Modelgebaseerde algoritmes in univariate tijdreeks. Met $k \geq 1$ and $k_1, k_2 \geq 0$ zodat $k_1 + k_2 > 0$ en waarbij $\hat{x}_t$ de verwachte waarde is. [1]

Soort model	Gebruikte data	Verwachte waarde	Puntanomalieën
Estimation models	$\{x_{t-k_1}, ..., x_t, ..., x_{t+k_2}\}$	$\hat{x}_t$	$\lvert x_t - \hat{x}_t \rvert > r$
Prediction models	$\{x_{t-k_1}, ..., x_t - 1\}$	$\hat{x}_t$	

Het berekenen van het verwachtte punt $\hat{x}_t$ en threshold r hangt af van welk model er gebruikt wordt. Bij estimation modellen worden zowel observaties gebruikt die in het verleden van t liggen als observaties die in de toekomst van t liggen om het punt $\hat{x_t}$ en de threshold te berekenen. Op basis van deze gegevens kan dan de afstand berekend worden tussen het punt x_t en het verwachtte punt $\hat{x}_t$. Als deze afstand groter is dan r, dan kan er geconcludeerd worden dat het punt x_t een anomalie is.

In tegenstelling tot estimation gebaseerde modellen gaan prediction modellen enkel data uit het verleden gebruiken. Algoritmes onder deze vorm worden gebruikt wanneer de toekomstige data ongekend is. Een minpunt bij deze algoritmes is dat ze periodiek moeten worden bijgewerkt om accurate voorspellingen te blijven maken. Sommige modellen worden achterliggend automatisch geüpdatet waardoor het niet nodig is om ze elke keer volledig te moeten her-trainen [1].

Voorgaande modelgebaseerde algoritmes detecteren anomalieën op basis van de afstand tussen het punt en het verwachtte punt en een voorberekende threshold. Andere algoritmes zoals histogramming en density berekenen anomalieën op een andere manier.

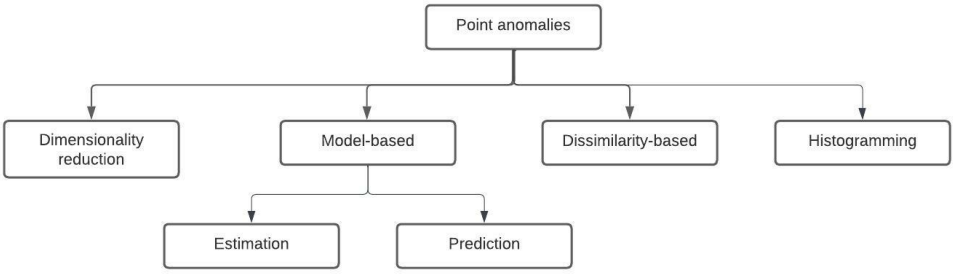
Densitygebaseerde algoritmes berekenen hoeveel y-waardes (met y gelijk aan de waarde van het punt en x de tijd waarop de waarde werd gemeten), er in de nabije omgeving van de waarde van een bepaald punt x liggen met een bepaalde afstand R. Als het aantal buren van x kleiner is dan r dan is het punt x een anomalie. Waarbij r een indicatie geeft van wat het minimum aantal buren voor x zou moeten zijn.

$$|\{x \epsilon X | d(x, x_t) \leq R\}| < r$$

Histogramming algoritmes maken een histogram van de dataset. Om een histogram te maken wordt de data opgedeeld in een verzameling van buckets. Daarna wordt elke bucket vergeleken met dezelfde bucket maar dan genomen van de verwachtte (normale) tijdreeks dataset. Als deze twee buckets te ver uit elkaar liggen kan er geconcludeerd worden dat de bucket anomalieën bevat.

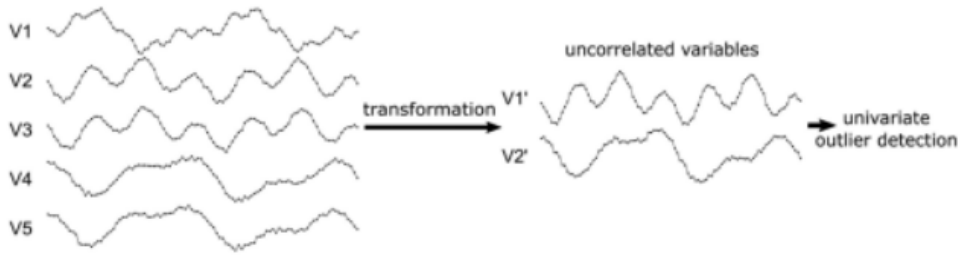
Puntanomalie algoritmes - multivariate

Puntanomalieën in multivariate tijdreeksen kunnen opgespoord worden door op elke variabele een univariate puntanomalie algoritme toe te passen. Het nadeel hiervan is dat belangrijke relaties tussen verschillende variabelen niet worden verwerkt waardoor cruciale informatie verloren gaat. Figuur 2.6 geeft een overzicht welke andere technieken hiervoor beschikbaar zijn.



Figuur 2.6: Types puntanomaliedetectie algoritmes voor multivariate tijdreeksen [1]

De eerste techniek om multivariate anomalieën te detecteren is het aantal dimensies verminderen door een PCA (*Principal Component Analysis* [5]) algoritme toe te passen op de gehele dataset. Dit geeft als resultaat één of meerdere nieuwe variabelen die geen correlaties met elkaar bevatten. Op deze variabelen is het dan mogelijk om univariate anomaliedetectie algoritmes toe te passen. Figuur 2.7 geeft weer wat het resultaat kan zijn na het uitvoeren van een PCA algoritme op een multivariate tijdreeks.



Figuur 2.7: Resultaat na het uitvoeren van een PCA algoritme op een multivariate tijdreeks [1]

Net zoals bij univariate technieken is het mogelijk om modelgebaseerde algoritmes te gebruiken bij multivariate tijdreeksen. Deze zijn ook gebaseerd op de afstandsvergelijking van:

$$|x_t - \hat{x}_t| > r$$

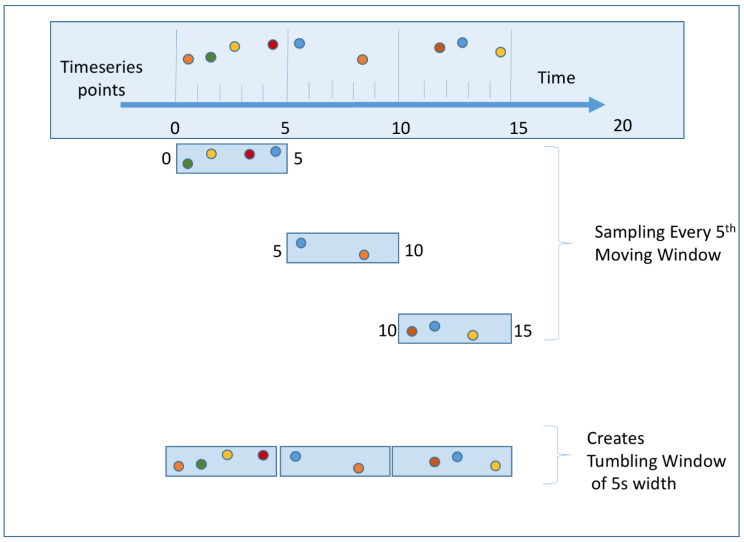
Waarbij $\hat{x}_t$ een k-dimensioneelpunt is. Net zoals bij univariate gebruiken multivariate estimationgebaseerde technieken de gehele tijdreeks (datapunten uit het verleden en toekomst) om anomalieën te detecteren terwijl predictiongebaseerde modellen alleen data uit het verleden gebruiken om anomalieën te zoeken. Deze technieken zijn gebaseerd op dezelfde als univariate algoritmes maar dan veralgemeend voor multivariate tijdreeksen.

Multivariate dissimilarity gebaseerde technieken werken zoals de univariate varianten op basis van het aantal omliggende buren. Als dit kleiner is dan een voorbepaalde threshold dan kan er geconcludeerd worden dat het punt een anomalie is.

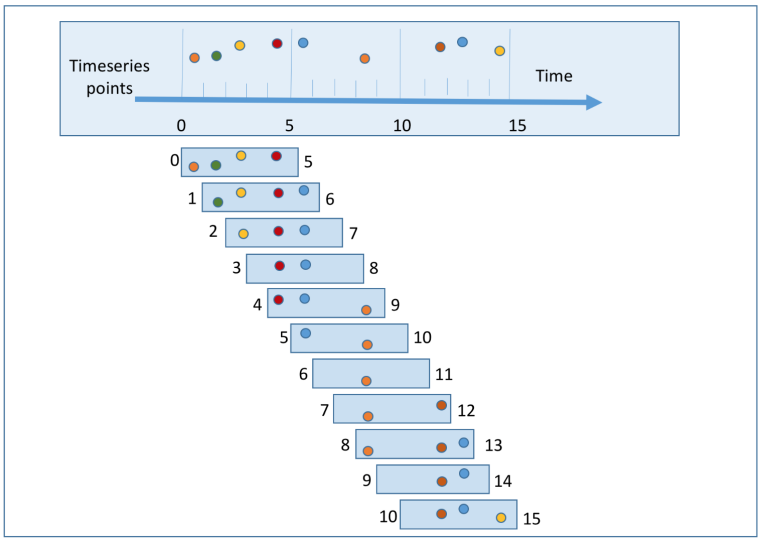
Hetzelfde geldt voor de histogramming technieken deze zijn ook nauw verwant met de univariate varianten alleen werken deze meestal niet met de ruwe data. Deze technieken voeren eerst een preprocessing stap uit op de data, zoals de tijdreeks omzetten in een graaf waarbij de nodes multivariate punten zijn en de edges de similarity waarden tussen twee punten. Men kan dan een *random walk* model gebruiken om de gelijkheid tussen nodes te berekenen. Op basis van deze gelijkheid kan men dan bepalen welke punten anomalieën zijn en welke niet [1].

Sequentieanomalie algoritmes - univariate

Zoals eerder vermeld zijn sequentieanomalieën een reeks van datapunten die in hun geheel een anomalie zijn. Het detecteren van dergelijke sequenties is complexer dan het detecteren van puntanomalieën. Eén van de redenen is dat een sequentie eender welke lengte kan hebben. Methoden kunnen dus een vaste of een variabele grootte van window hebben. Een window is een sequentie van datapunten die bestudeerd worden. Dit window "schuift"(slide) over heel de data om voor elke sequentie te bepalen of deze anomalie is of niet. Als een window een anomalie blijkt te zijn, dan zullen alle datapunten binnen dit window gelabeld worden als anomalie. Een window is anomalie wanneer deze te sterk afwijkt van alle andere windows. Het sliden of schuiven van een window, waarbij er telkens de grootte van een window wordt opgeschoven, over heel de dataset wordt weergegeven in figuur 2.8. Een andere mogelijkheid is dat het window per datapunt opschuift zoals weergegeven in figuur 2.9 [1].



Figuur 2.8: Sliding window view van een tijdreeks waarbij er per window een heel window wordt opgeschoven [6].



Figuur 2.9: Sliding window view van een tijdreeks waarbij het window telkens één datapunt opschuift [6].

Een voor hand liggende techniek, om te berekenen of een window anomalie is, is om elke sequentie met elke andere sequentie te vergelijken:

$$\forall S \, \epsilon \, A, \, min_{D' \epsilon A, D \cap D' = 0} \left(d(D, D') \right) \, > \, min_{S' \epsilon A, S \cap S' = 0} (d(S, S'))$$

A is een verzameling van alle sequenties (windows) verkregen van de tijdreeks X door sliding window toe te passen van een willekeurige grootte. D is de sequentie die wordt vergeleken, door middel van de functie d, met alle andere sequenties D', waarbij D en D' elementen zijn van A. S is ook een sequentie van A die wordt vergeleken met alle andere sequenties S' die afkomstig zijn uit A. Om te bepalen of D een anomalie vormt wordt tussen S en S' vergeleken met de afstand van D en D', door middel van de functie d. Functie d berekent de euclidische afstand tussen D en D'. Dit betekend, voor alle combinaties D, D' de afstand tussen alle combinaties van S, S', berekend wordt om te bepalen of een window D' anomalie is. Als lengte van de sequenties, D en D', nemen we drie, dan kan de euclidische afstand tussen D en D' als volgt berekend worden, waarbij F, G, H de variabelen zijn van sequentie D en F', G', H' de variabelen zijn van sequentie D':

$$d(D, D') = \sqrt{(F - F')^2 + (G - G')^2 + (H - H')^2}$$

Het is belangrijk om te vermelden dat D en D', in dit voorbeeld, geen enkel gemeenschappelijk punt hebben net zoals S en S' (met andere woorden deze overlappen elkaar niet). Omdat elke sequentie met elke andere sequentie vergeleken wordt bekomen we een tijdscomplexiteit van $O(n^2)$ waarbij n het aantal sequenties zijn [1].

Bij de voorgaande techniek is er geen notie van normaalheid van de data. Er kan dus niet met honderd procent zekerheid gezegd worden dat deze sequentie een anomalie is of niet. Dissimilarity gebaseerde technieken berekenen de gelijkheid tussen de geobserveerde sequentie S en de verwachte (normale) sequentie $\hat{S}$ (waarbij functie s een gelijkheidsfunctie is en r een threshold) [1].

$$s(S, \hat{S}) \, > \, r$$

Om te bepalen wat "normale"data zijn kan men ofwel gebruik maken van dezelfde tijdreeks waarop men analyse uitvoert of van een andere tijdreeks die gelijkend is aan de huidige (geobserveerde) tijdreeks maar dan zonder anomalieën. Wanneer men gebruik maakt van dezelfde tijdreeks kan men met behulp van clustering technieken de data eerst clusteren. Bij clusters die geen anomalieën bevatten gaan de punten zeer geconcentreerd bij elkaar liggen. Op deze clusters kan men manueel detecteren of de cluster normale data bevat. Deze data kan dan vergeleken worden met de geobserveerde sequentie. Een andere manier is kijken naar de vorige geobserveerde sequentie. Probleem hierbij is dat wanneer de data bij elke sequentie een ander gemiddelde en/of een andere variantie heeft, we deze dus niet met elkaar kunnen vergelijken.

Predictiongebaseerde modellen bij univariate sequentie-technieken gebruiken data uit het verleden om datapunten in de toekomst te voorspellen. Elke reeks van voorspellingen is dan een window of sequentie. Om te bepalen of deze sequentie abnormaal is wordt het verschil berekend tussen de voorspelde sequentie en de toekomstige sequentie. Wanneer deze afstand groter is dan een threshold wordt deze sequentie gelabeld als anomalie.

Frequentiegebaseerde technieken gebruiken zoals predictiongebaseerde technieken, data uit het verleden om te voorspellen of een huidige sequentie abnormaal is of niet. Deze techniek is ook gebaseerd op de frequentietechniek bij puntanomalieën. Men berekent hoe vaak de geobserveerde sequentie S voorkomt in de tot nu toe bekomen dataset. Net zoals hier bestaat ook weer de moeilijkheid om na te gaan wanneer twee sequenties gelijk zijn met elkaar. Informatiegebaseerde technieken zijn sterk verwant met de frequentiegebaseerde algoritmes. Hoe vaker een sequentie voorkomt hoe meer informatie er bestaat over de geobserveerde sequentie en dus waarschijnlijk een normale sequentie is.

Sequentieanomalie algoritmes - multivariate

Zoals eerder vermeld kan men in eerste instantie zoeken naar variabelen die onafhankelijk zijn van elkaar en daarop univariate technieken toepassen. Nadeel is, dat er geen rekening wordt gehouden met de temporale informatie van de variabelen. Een manier om zulke variabelen te detecteren is een dimensionality reduction algoritme toe te passen zoals eerder vermeld. De variabelen kunnen daarna individueel geanalyseerd worden door een univariate algoritme. Net zoals bij univariate zijn er hier ook estimation-, prediction- en dissimilaritygebaseerde technieken die opgebouwd zijn uit de univariate variant.

Tijdreeksanomalie

De laatste soort anomalie is een tijdreeksanomalie. Een tijdreeksanomalie is een volledige tijdreeks variabele die abnormaal is. Het is mogelijk om deze technieken ook in een streaming context te gebruiken. De technieken die hier gebruikt worden zijn gebaseerd op de voorgaande technieken maar dan veralgemeend om gehele variabelen te detecteren. Dus kan men hier ook gebruik maken van dissimilaritygebaseerde technieken waarbij pair-wise het verschil wordt berekend tussen de verschillende tijdreeksen. De wijze waarop dit kan gebeuren is het clusteren van hele variabelen. Wanneer de afstand tussen een variabele en de clustercentroid groter is dan een threshold r, dan kan men concluderen dat deze variabele een anomalie is. [1].

Soorten learning

Al de voorgaande beschreven technieken kunnen nog onderverdeeld worden in supervised, semi-supervised of unsupervised *learning* technieken. Supervised algoritmes hebben een volledig gelabelde dataset nodig als input. Alle soorten anomalieën (lokaal/globaal, punt/sequentie,...) moeten vooraf gekend en gelabeld zijn. Unsupervised learning is het tegenovergestelde van supervised learning. Het is niet noodzakelijk vooraf te weten wat de normale data is en wat de anomalieën zijn. Geen enkel datapunt moet gelabeld zijn als input bij het unsupervised model. Semi-supervised modellen verwachten dat de normale data gelabeld is of enkel de anomalieën.

Focus in deze thesis

Deze thesis focust zich op univariate unsupervised technieken, omdat bedrijven zoals TrendMiner grote hoeveelheden data verwerken waardoor het niet altijd evident is om aan (semi-) supervised technieken te voldoen. Supervised algoritmes hebben als train input een volledig gelabelde dataset met alle mogelijk anomalieën die zich kunnen afspelen. Dit is binnen deze context niet altijd eenvoudig.

Hoofdstuk 3

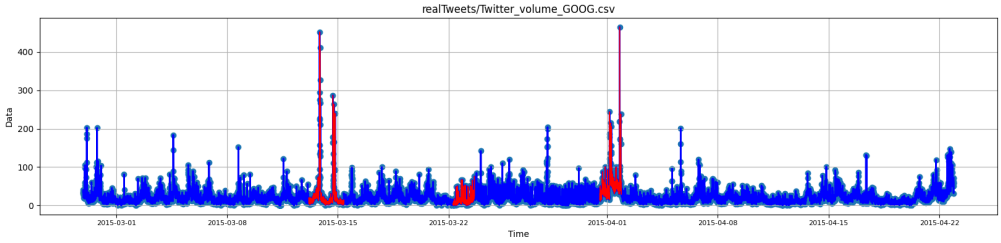
Datasets

Het kiezen van de juiste tijdreeksdataset is belangrijk om algoritmes te testen. Zoals aangegeven in het vorige hoofdstuk is niet elk algoritme geschikt voor elke tijdreeks. Voor univariate technieken kan men enkel univariate tijdreeksen gebruiken of multivariate datasets waarbij elke variabele apart door een univariate algoritme gehaald wordt. Daarnaast is het belangrijk om rekening te houden met het soort anomalie. Een ruime variatie van anomalieën is belangrijk om zulke algoritmes te evalueren. In deze thesis is een tijdreeksdataset een bestand dat bestaat uit één univariate tijdreeks. Dit bestand is een CSV-bestand met als key-value pair het tijdstip en de waarde van de meting.

3.1 Numenta Anomaly Benchmark (NAB)

De Numenta Anomaly Benchmark tijdreeksdataset bestaat uit 50 real-world en artificiële tijdreeksen zowel gelabeld als ongelabeld. Ze bevat zowel gevarieerde sequentieanomalieën als puntanomalieën wat ze dus geschikt maakt om beide technieken te testen. NAB heeft in elke tijdreeks anomalieën tenzij anders aangegeven.

De real-world tijdreeksen zijn gemaakt van onder andere het aantal Twitter *mentions* van grote bedrijven zoals Google en IBM. Deze worden om de vijf minuten gemeten gedurende een bepaalde periode. De andere datasets bestaan uit tijdreeksen die samengesteld zijn uit het CPU-gebruik van AWS-servers en het real time autoverkeer in Minnesota, gemeten door verschillende sensoren (snelheid, occupancy en sensoren die de tijd meten tussen de verplaatsingen). NAB bevat scripts om het algoritme te evalueren samen met notebooks om ze te visualiseren [7]. Figuur 3.1 geeft een Twitter tijdreeks als voorbeeld weer.



Figuur 3.1: Tijdreeks gemeten door het aantal Twitter-mentions van Google.

3.2 Skoltech Anomaly Benchmark (SKAB)

SKAB (v0.9) bevat 34 gelabelde en ongelabelde tijdreeksen zowel met puntanomalieën als sequentieanomalieën. In tegenstelling tot de NAB-dataset wordt hier elk punt gelabeld met extra informatie over de anomalie. Deze extra informatie verduidelijkt of het een changepoint voor een sequentie is of niet. Een changepoint is het punt waarop een sequentie een ander gemiddelde, variantie of trend heeft. Elk bestand is één experiment en bevat één anomalie. Elk van deze datasets is een multivariate tijdreeks waarvan de data afkomstig is van verschillende sensoren in een testomgeving. Voorbeelden van de sensoren zijn: Accelerometer, drukmeter, voltagemeter, etc. [8]

3.3 GutenTAG

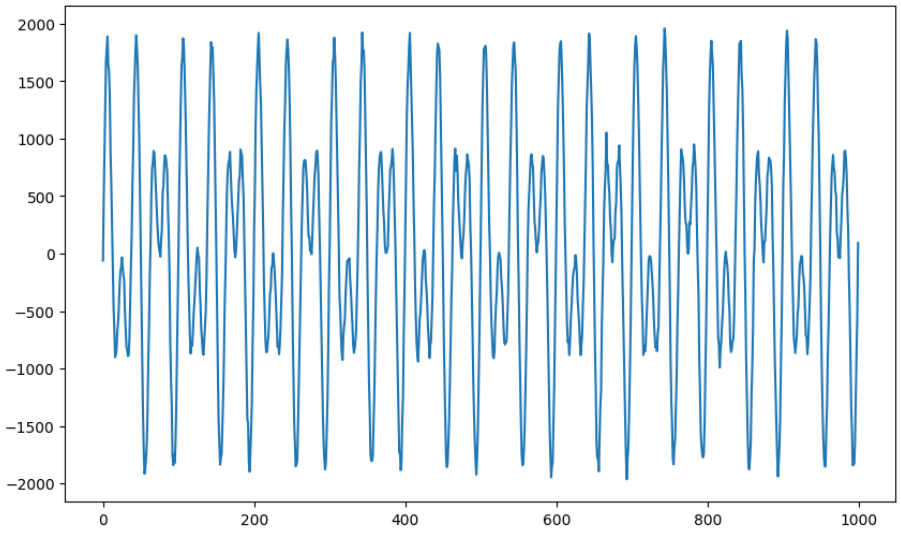
GutenTag is een tool om gevarieerde tijdreeksen te maken met verschillende soorten anomalieën. Een GutenTag tijdreeks is een uni- of multivariate tijdreeks met een basisoscillatie of combinatie van basisoscillaties. Een basisoscillatie is een wave die een tijdreeks simuleert. Enkele basisoscillaties zijn:

1. sine
2. cosine
3. square
4. random-walk
5. cylinder-bell-funnel
6. ECG
7. polynomial
8. random-mode-jump
9. formula
10. sawtooth
11. dirichlet
12. mls
13. custom-input

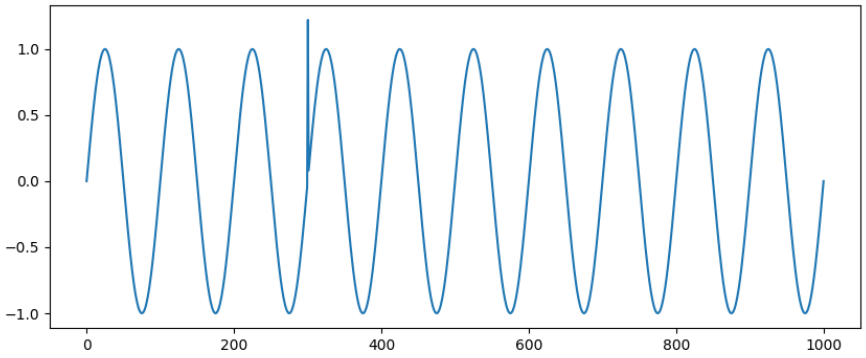
Figuur 3.2 is een voorbeeld van een tijdreeks gegenereerd door twee sinusoscillaties. Elk van deze wave functies hebben parameters om bijvoorbeeld de variantie en trend van een tijdreeks aan te passen.

Zoals eerder besproken kan een anomalie een puntanomalie (lokaal of globaal) of een sequentieanomalie zijn. Puntanomalieën worden hier beschreven als extremums die globaal of lokaal zijn. Figuur 3.3 is een voorbeeld van een extremumanomalie in GutenTAG [9].

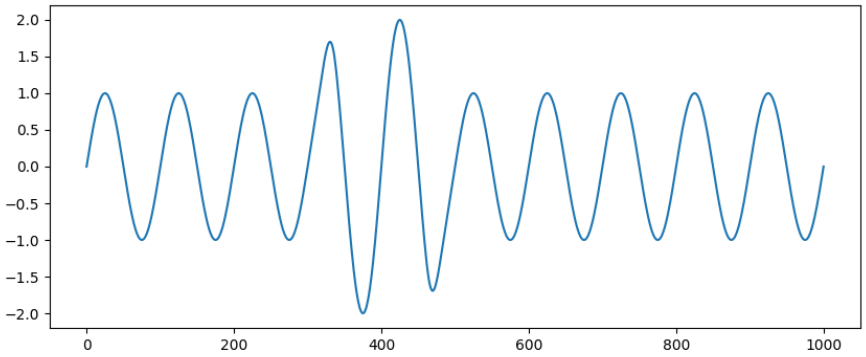
Met behulp van GutenTag is het mogelijk om verschillende soorten sequentieanomalieën te genereren zoals: amplitude, frequentie, gemiddelde, patroon, patroon-shift en trend. De eerste soort sequentieanomalie die besproken wordt is de amplitude deze rekt een sequentie van de tijdreeks verticaal uit naar boven en beneden. Figuur 3.4 is een voorbeeld van een amplitudeanomalie.



Figuur 3.2: Een tijdreeks gegenereerd door een sinusfunctie met een anomalie [9].



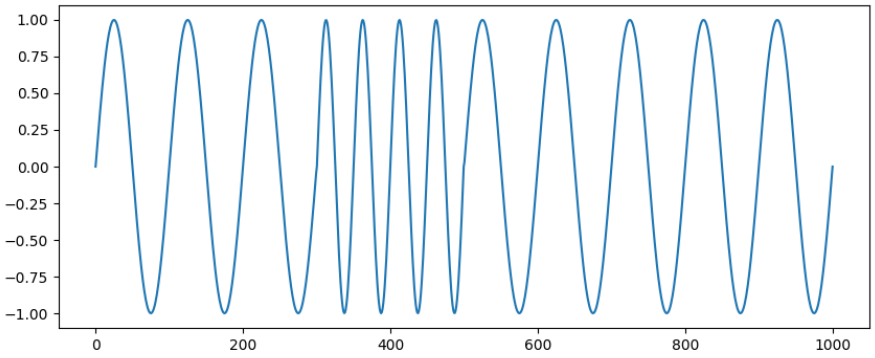
Figuur 3.3: Een globaal puntanomalie gegenereerd door GutenTag [9].



Figuur 3.4: Een amplitudeanomalie gegenereerd door GutenTag [9].

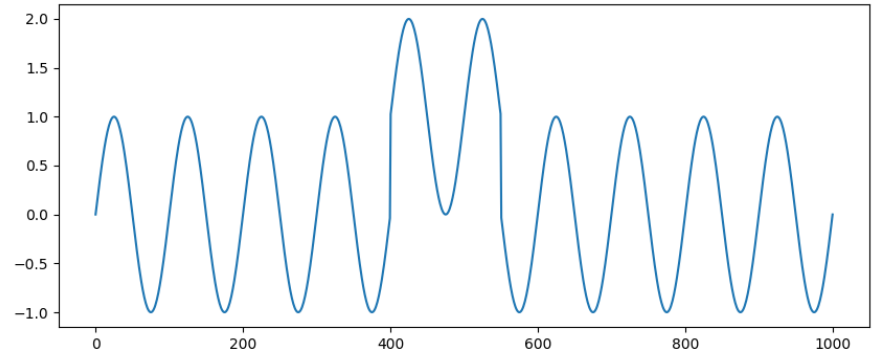
Als tweede is het mogelijk om de frequentie van een reeks datapunten aan te passen om een frequentieanomalie te genereren. Als men de frequentie van een reeks datapunten aanpast volgen datapunten sneller op elkaar waardoor golven horizontaal tegen elkaar aangeduwd worden.

Figuur 3.5 is een voorbeeld van een tijdreeks met een frequentieanomalie.



Figuur 3.5: Anomalie met een aangepaste frequentie [9].

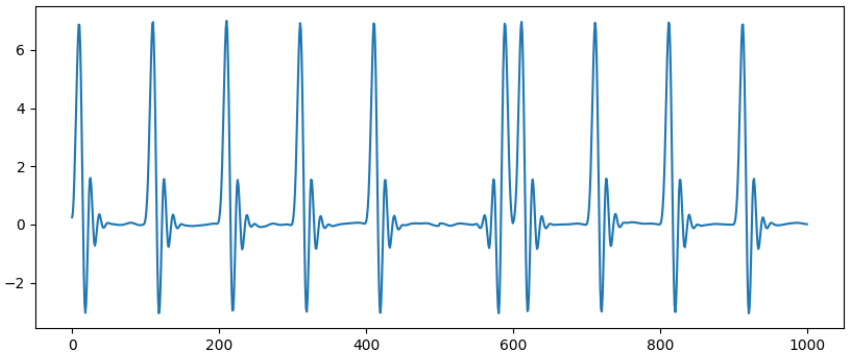
Bij een sequentieanomalie met een ander gemiddeld wordt een reeks van datapunten binnen een tijdreeks in zijn geheel verhoogd of verlaagd zodat men lokaal of globaal een hogere/lagere sequentie bekomt. Een voorbeeld hiervan wordt weergegeven op figuur 3.6.



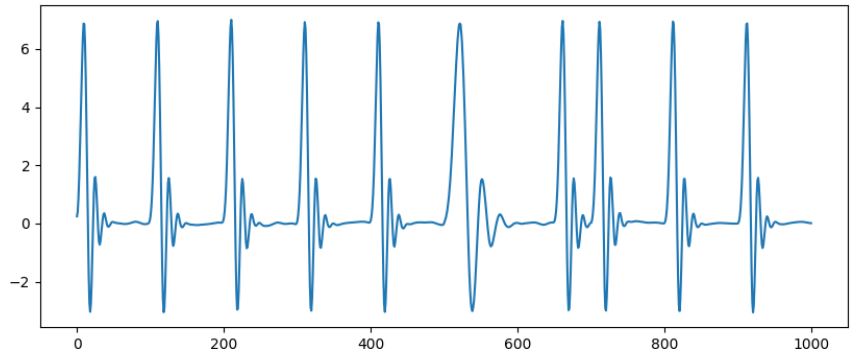
Figuur 3.6: Anomalie met een aangepast gemiddelde [9].

Een patroonanomalie verandert het patroon van de waves terwijl patroon-shift het patroon gaat veranderen door datapunten uit elkaar te trekken. Figuur 3.7a geeft een voorbeeld weer van een patroonanomalie en figuur 3.7b is een voorbeeld van een patroon-shiftanomalie.

Het is ook nog mogelijk om de trend van een tijdreeks aan te passen. Dit kan bijvoorbeeld veranderingen van sensoren door de dag gaan simuleren. Dit is daarom niet noodzakelijk een anomalie maar kan wel geanalyseerd worden als een sequentieanomalie zoals aangegeven in figuur 3.8

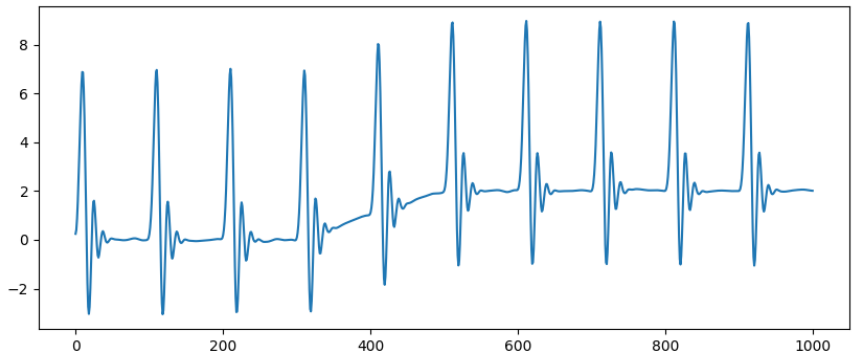


(a) Shift.



(b) patroon-shift.

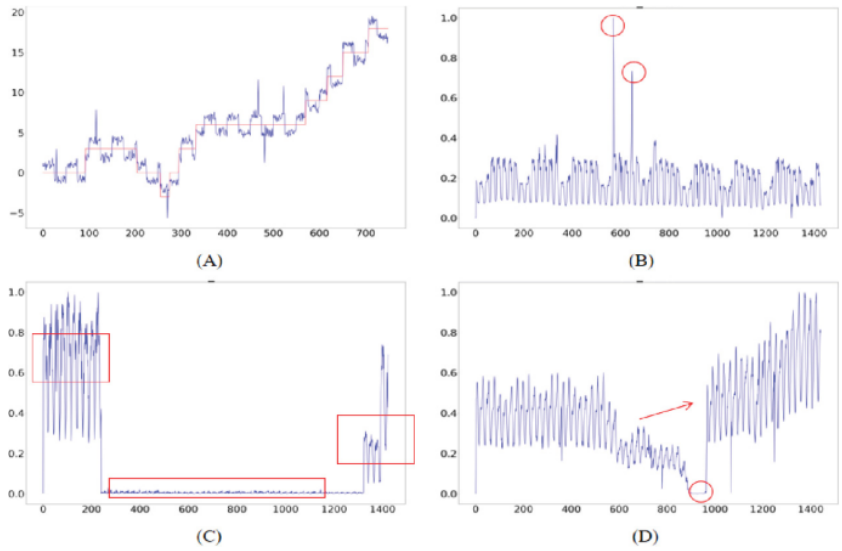
Figuur 3.7: Verschil tussen shift en patroon-shift shiftanomalie [9].



Figuur 3.8: Veranderen van de trend van een tijdreeks [9].

3.3.1 Yahoo-S5

De Yahoo-S5 benchmark dataset bestaat uit echte en synthetische tijdreeksdatasets. Deze tijdreeksen bevatten een ruime hoeveelheid aan verschillende soorten anomalieën. Elke aparte tijdreeks heeft verschillende soorten trending, noise en seasonality. De verzameling van synthetische datasets bestaat uit de metingen van de load van verschillende Yahoo-services. De Yahoo-S5 benchmark dataset is niet publiek beschikbaar en moet aangevraagd worden, daarnaast is de benchmark dataset enkel toegestaan voor niet-commercieel gebruik [10].



Figuur 3.9: Voorbeeld van enkele tijdreeksen afkomstig uit de Yahoo-benchmark dataset [11].

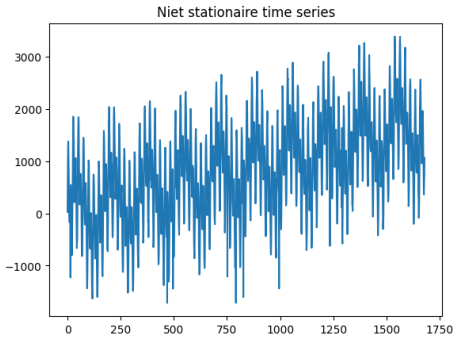
Hoofdstuk 4

Algoritmes

In dit hoofdstuk worden een aantal algoritmes besproken die verschillende soorten anomalieën detecteren. Er wordt vergeleken welk algoritme beter is in het detecteren van een soort anomalie dan de anderen. Zoals eerder vermeld worden er in deze thesis alleen univariate unsupervised technieken geïmplementeerd omdat bedrijven zoals TrendMiner grote hoeveelheden data binnenkrijgen waardoor het niet altijd evident is om aan supervised technieken te voldoen. Supervised algoritmes hebben als train input een volledig gelabelde dataset met alle mogelijke anomalieën die zich kunnen afspelen. Dit is binnen deze context niet altijd eenvoudig. Dergelijke bedrijven ontvangen op korte tijd grote hoeveelheden data, waardoor het onmogelijk is om elk soort anomalie op voorhand te kennen. Elk van deze algoritmes worden daarna geïmplementeerd in een webapplicatie om een meer gevisualiseerde interpretatie te bekomen van hoe deze algoritmes werken.

4.1 Preprocessing

De eerste stap voor de algoritmes van K-Means en ARIMA bestaat uit een preprocessing stap van een tijdreeks waarbij er gecontroleerd wordt of de data stationair is. Zoniet wordt er differencing uitgevoerd om deze stationair te maken. Stationair betekent dat zowel het gemiddelde als de variantie van de data doorheen de tijd constant blijft. Bij een niet-stationaire dataset gaan de datapunten steeds een lagere of hogere waarde hebben doorheen de tijd. Dit gaat ervoor zorgen dat onze voorspellingen niet meer accuraat zijn en er dus meer false positives (FP) of false negatives (FN) worden berekend. False positives zijn datapunten die voorspeld zijn als anomalie maar in werkelijkheid geen anomalie zijn. False negatives zijn dan weer datapunten die voorspeld zijn als normaal maar in werkelijkheid net wel een anomalie zijn. Figuur 4.1 is een voorbeeld van een niet-stationaire tijdreeks.



Figuur 4.1: Voorbeeld van een niet-stationaire tijdreeks.

Om te testen of een tijdreeks stationair is kan men de statistische Augmented Dickey–Fuller test gebruiken. Dit is een hypothesetest met als null-hypothese dat de data een unit root heeft en dus niet-stationair is. Een unit root wil zeggen dat het gemiddelde en de variantie van de dataset niet constant blijft doorheen de tijd. Als de p-waarde van de test kleiner is dan een significantielevel dan is de test niet significant en zal de data stationair zijn. Als de p-waarde groter is, dan zal er dus een grotere kans zijn dat de data niet-stationair is. Als significantie level wordt er 95% gebruikt.

Als uit de test blijkt dat de tijdreeks niet-stationair is, kan men verschillende methodes toepassen om deze stationair te maken. Eén van deze methodes is het differencing algoritme. Hierbij wordt het verschil berekend tussen x_t en x_{t-1}, waarbij x de tijdreeks is en t een bepaalde timestamp. Een voorbeeld van het toepassen van differencing wordt weergegeven in tabel 4.1 waarbij de linker kolom de originele tijdreeks is en de rechter kolom de tijdreeks is na het uitvoeren van het differencing algoritme. Hierbij wordt elk datapunt afgetrokken van het vorige datapunt. Voor de eerste rij is er geen vorig datapunt dus zal er NaN staan. Als deze methode toegepast wordt op de dataset van figuur 4.1, krijgen we een stationaire tijdreeks zoals weergegeven in figuur 4.2. Merk op dat dit process meerdere keren kan uitgevoerd worden. Als de tijdreeks een kwadratische trend bevat moet deze stap twee keer uitgevoerd worden.

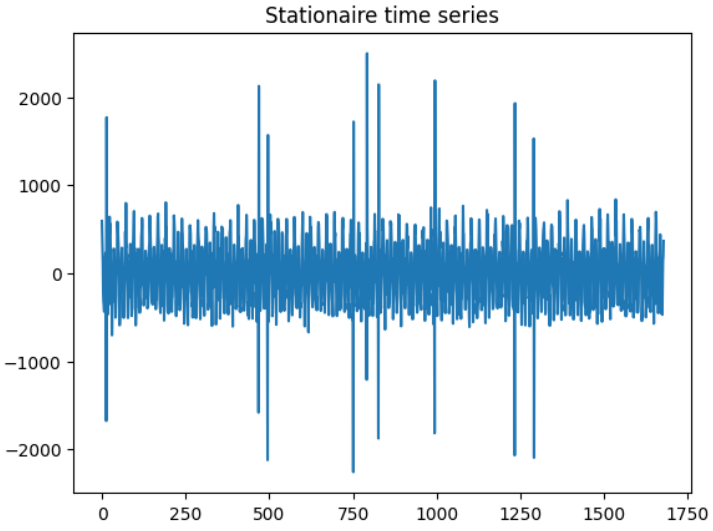
Tabel 4.1: Differencing

Tabel 4.2: Originele tijdreeks

10
13
9
... .

Tabel 4.3: Tijdreeks na differencing

NaN
3.0
-4.0
...



Figuur 4.2: Differencing toegepast op de data in figuur 4.1. Het resultaat is stationair.

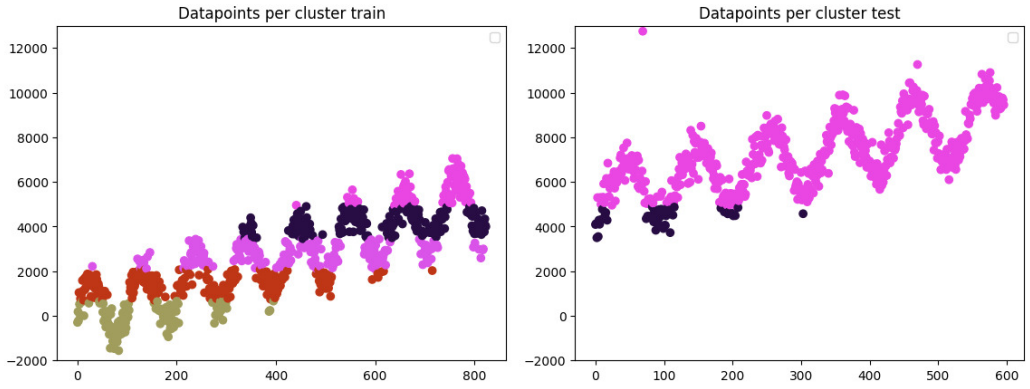
4.2 K-Means

K-Means is een uni/multivariate unsupervised techniek die op basis van een bepaald aantal *centroids* elk datapunt in een tijdreeks clustert. Op basis van deze informatie kan men anomalieën detecteren. Het detecteren van anomalieën met behulp van K-Means gebeurt in drie stappen:

1. Preprocessing van de tijdreeks.
2. Uitvoeren van het K-Means clustering algoritme.
3. Op basis van de gegeven clusters anomalieën detecteren.

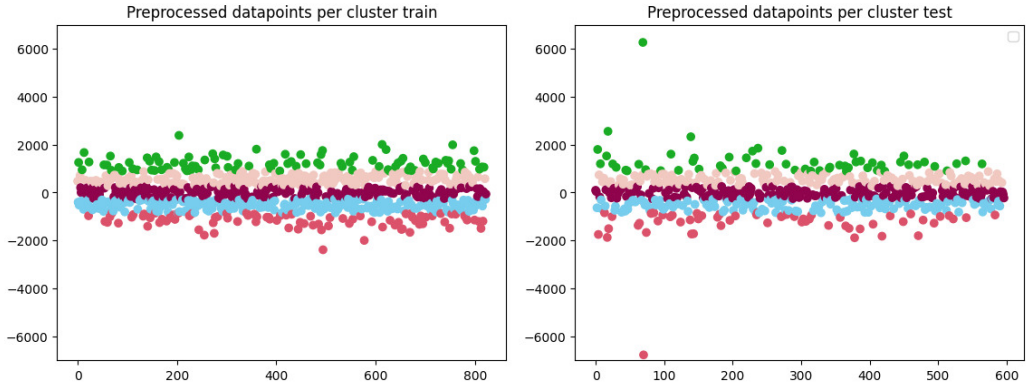
4.2.1 Preprocessing van een tijdreeks

De eerste stap binnen dit proces is het preprocessen van een gegeven tijdreeks. Het detecteren van anomalieën met behulp van K-Means gaat een beter resultaat opleveren bij een stationaire tijdreeks dan bij een niet-stationaire tijdreeks. Daarom wordt er de preprocessing stap gebruikt die beschreven is in sectie 4.1. Uit figuur 4.3 kan men afleiden dat, naarmate de niet-stationaire tijdreeks vordert, alle punten die in de testset liggen aan één bepaalde cluster toegewezen worden.



Figuur 4.3: Een univariate tijdreeks, met op de y-as de waarde waarop de clusters zijn berekend en de x-as de tijd. De linker grafiek zijn de datapunten waarop het K-Means algoritme is getraind terwijl de rechter grafiek de datapunten van de tijdreeksdataset zijn waarbij we de bijbehorende cluster willen voorspellen. Omdat de data niet-stationair is en hoger ligt dan al de andere datapunten zullen de datapunten in de testset fout geclusterd worden. Deze zullen namelijk altijd in de roze cluster terechtkomen.

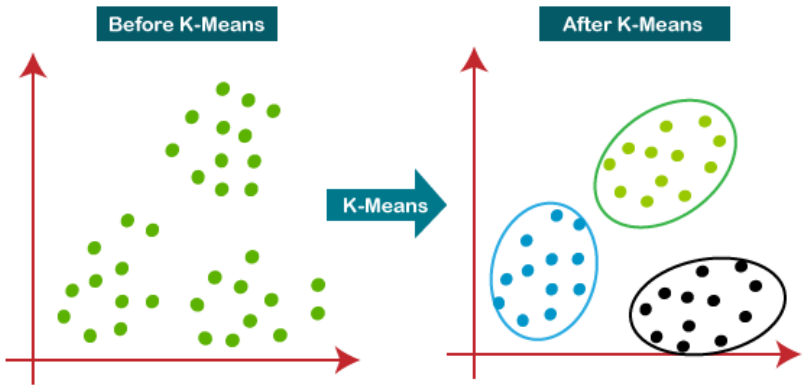
Na het uitvoeren van de preprocessing stap is de data stationair en zullen de toekomstige punten aan hun correcte cluster toegekend worden, zoals weergegeven op figuur 4.4.



Figuur 4.4: De linker grafiek zijn de preprocessed datapunten waarop het K-Means algoritme is getraind, terwijl de rechter grafiek de preprocessed datapunten van een tijdreeks zijn waarbij men de bijbehorende cluster wilt voorspellen. De y-as zijn de waardes waar clusters op berekend zijn en de x-as is de tijd.

4.2.2 Clustering

K-Means is een iteratief algoritme dat punten gaat onderverdelen in clusters. (1) De eerste stap van het algoritme is het aantal K-clusters bepalen. (2) Het kiezen van K-random cluster centroids. Dit kunnen zowel K-randompunten van de dataset zijn als nieuwe K-randompunten. (3) Het toewijzen van alle punten aan de dichtstbijzijnde cluster centroid. (4) Het nemen van het gemiddelde van elk punt. Dit zal de nieuwe waarde van onze centroid zijn. De afstand tussen centroids en datapunten kan berekend worden aan de hand van de Euclidische afstand. Stappen drie en vier worden iteratief uitgevoerd tot de waarde van de centroid niet meer aangepast wordt of tot geen enkel datapunt meer van cluster veranderd.



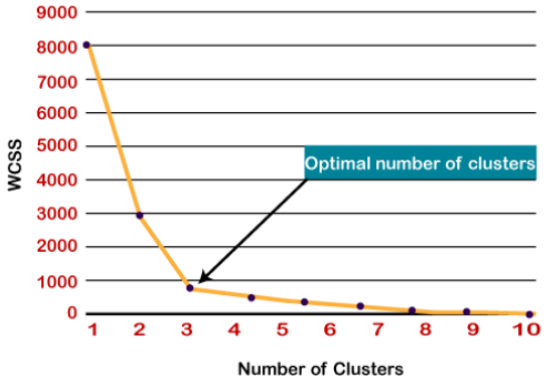
Figuur 4.5: Voorbeeld resultaat van het K-Means algoritme. [12]

De keuze van het aantal centroids kan gebeuren door het K-Means algoritme meerdere keren uit te voeren met elke keer een verschillend aantal centroids. Na elke training kan men de inertia ofwel de Within Cluster Sum of Squares (WCSS) als volgt berekenen:

$$\sum_{i=1}^{K} \sum_{x \epsilon X_i} (x - \mu_i)^2$$

Waarbij $X_1, ..., X_K$ de verzameling is van alle punten in cluster $1, ..., K$, K het aantal clusters zijn en μ_i de centroid is van de cluster i. De inertia wordt samen met de hoeveelheid clusters

per iteratie geplot in een grafiek zoals op figuur 4.6. Met behulp van deze grafiek selecteren we het aantal clusters zodanig dat de inertia bij een hoger aantal clusters minder hard daalt. Dit punt heet de elbow van de grafiek. We kunnen uit figuur 4.6 aflezen dat de inertia minder hard zal dalen als men meer dan 3 clusters neemt Deze drie clusters worden nu gebruikt om voor elk datapunt een bijbehorende cluster te vinden. De elbowgrafiek geeft dus weer wat de inertia of de within cluster sum of squares per cluster is. Met op de x-as het aantal clusters per keer dat het algoritme uitgevoerd worden en met op de y-as de WCSS (intertia). De clusters worden in deze implemenatie berekend op de waarde van een univariate tijdreeks zonder rekening te houden met het tijdstip.



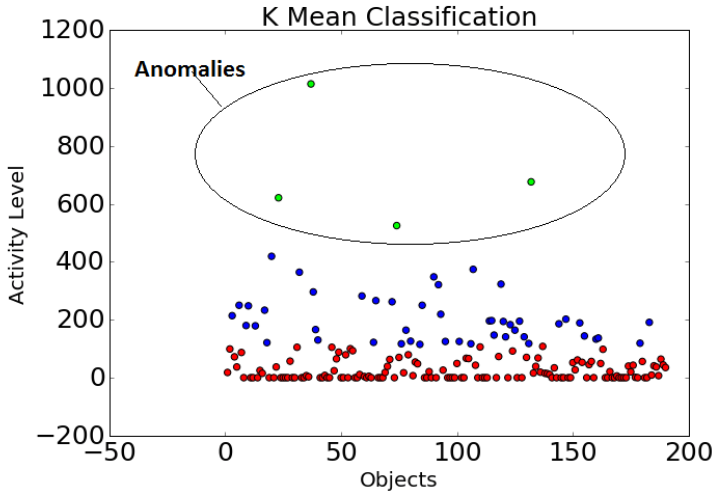
Figuur 4.6: Voorbeeld van een willekeurige elbowgrafiek [12].

4.2.3 Detecteren van anomalieën

Na het uitvoeren van het K-Means algoritme, op de y-waardes, hebben we voor elk datapunt een bijbehorende cluster. Anomalieën kunnen zich op twee manieren voordoen:

1. Heel de cluster is anomalie
2. Afzonderlijke punten binnen de cluster zijn anomalie

In het eerste geval worden alle punten binnen deze cluster gelabeld als anomalie. Om te bepalen of een volledige cluster teveel afwijkt van alle andere wordt er een threshold berekend op basis van het aantal punten van de gehele set datapunten. Als het aantal datapunten binnen een cluster kleiner is dan bijvoorbeeld 1% van de gehele dataset, kan er besloten worden dat deze cluster in zijn geheel een anomalie vormt. De bovenste cluster op figuur 4.7 wordt in dit geval in zijn geheel als anomalie beschouwd omdat het aantal datapunten binnen deze cluster, kleiner is dan 1% van de gehele set van punten.

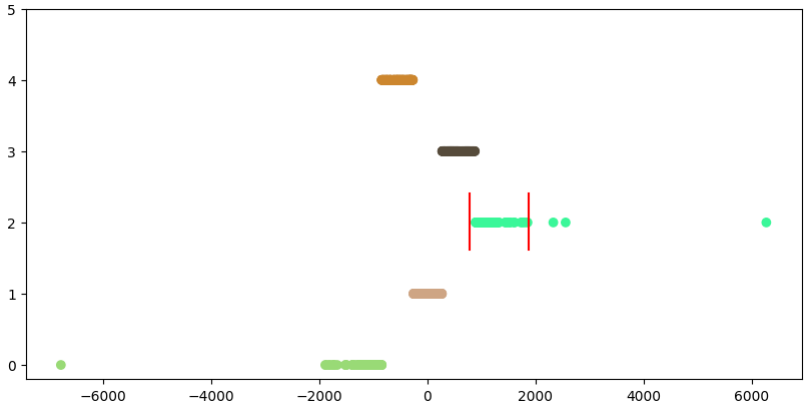


Figuur 4.7: Een cluster die in zijn geheel een anomalie vormt op basis van het aantal datapunten binnen deze cluster.

Als tweede kunnen afzonderlijke punten binnen een cluster gedetecteerd worden als een anomalie. Om dit te bepalen wordt er per cluster een minimum en een maximum threshold berekend op basis van de waarden van de datapunten binnen elke cluster. Als een punt kleiner is dan de minimum threshold of groter is dan de maximum threshold, dan is dit punt een anomalie. Deze thresholds worden als volgt berekend. $q25$ is het eerste kwartiel van alle datapunten binnen deze cluster, $q75$ is het derde kwartiel van alle datapunten binnen deze cluster en iqr de interkwartielafstand tussen het derde en het eerste kwartiel:

$$lower\_threshold = q25 - (1.5 * iqr)$$
$$upper\_thresh = q75 + (1.5 * iqr)$$

Figuur 4.8 geeft de boven- en ondergrens aan van de waarde binnen één cluster. Waarbij de Y-as de labels van de clusters zijn en de X-as de waarden van de tijdreeks. Voor cluster 1 wordt de boven- en ondergrens ook getoond in het rood. Als de waarden lager liggen dan de ondergrens of hoger liggen dan de bovengrens. Dan zijn deze waarden anomalieën.



Figuur 4.8: De y-as zijn de labels van de clusters terwijl de X-as de waarden zijn. Omdat K-Means getraind is op een univariate dataset is er dus maar één variabele. De twee rode lijnen zijn de onder- en bovengrens voor cluster twee.

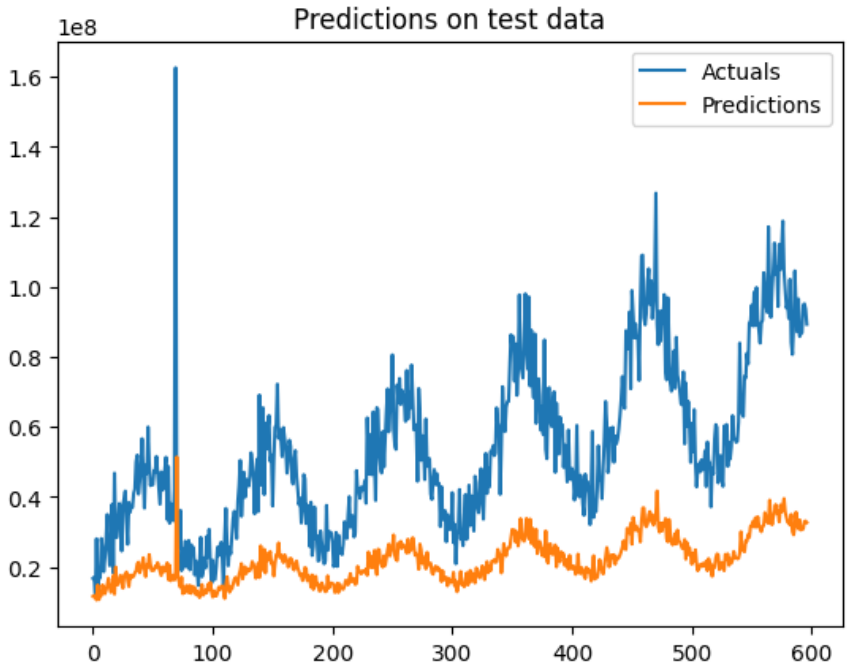
4.3 ARIMA

Autoregressive Integrated Moving Average (ARIMA) is een univariate unsupervised algoritme dat op basis van data uit het verleden voorspellingen kan maken in de toekomst. Het detecteren van anomalieën met behulp van ARIMA bestaat uit drie stappen:

1. Preprocessing van een tijdreeks.
2. Voorspellingen maken met behulp van het ARIMA-model.
3. Op basis van de voorspellingen en de echte waarden anomalieën detecteren.

4.3.1 Preprocessing van een tijdreeks

Net zoals bij K-Means moet er voor ARIMA dezelfde tussenstap gebeuren, namelijk het uitvoeren van een preprocessingstap om de data stationair te maken. Zoals eerder vermeld, wordt er bij een autoregressiemodel datapunten uit het verleden gebruikt. Als de variantie of het gemiddelde van de datapunten verschilt over tijd, zullen voorspellingen niet accuraat zijn. Dit komt omdat autoregressiemodellen ervan uitgaan dat het gemiddelde of de variantie over de tijd hetzelfde blijft. Hetzelfde geldt voor moving average modellen. Deze maken ook voorspellingen op basis van het verleden. Als het gemiddelde en de variantie sterk verschillen zullen de voorspellingen niet accuraat zijn. Figuur 4.9 is een voorbeeld van een autoregressiemodel waarbij voorspellingen gemaakt zijn op basis van de getoonde niet-stationaire testset. Om een tijdreeks stationair te maken wordt de differencingmethode toegepast zoals in sectie 4.1 vermeld is.



Figuur 4.9: Voorspellingen gemaakt door een autoregressiemodel op een niet-stationaire tijdreeks.

4.3.2 ARIMA-model

ARIMA is een statistisch model dat datapunten in de toekomst voorspelt op basis van data uit het verleden. Arima bestaat uit drie delen:

1. Autoregressive AR(p)
2. Integrated I(d)
3. Moving average MA(q)

Autoregressive AR(p)

Autoregressive maakt voorspellingen op basis van p, datapunten uit het verleden waarbij $\hat{y}_a$ het nieuwe voorspelde punt is, c een constante is, t het tijdstip van het huidige punt en een reeks α die de weights zijn van het regressiemodel.

$$\hat{y}_a = c + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + ... + \alpha_p y_{t-p}$$

Stel $p = 1$ en de coëfficiënt $\alpha_1 = 0$ dan merken we op dat het model alleen de constante als resultaat teruggeeft. Dit wil zeggen dat het vorige datapunt $(t-1)$ geen impact heeft op het voorspellen van het huidige datapunt (t). Als α positief is bestaat er een positieve correlatie, dit wil zeggen dat wanneer het vorige punt stijgt zal het huidige punt ook stijgen en omgekeerd. Een negatieve coëfficiënt duidt aan dat er een negatieve correlatie is, met als gevolg dat het huidige datapunt zal dalen als het vorige datapunt stijgt en omgekeerd [13].

$$\hat{y}_a = c + 0 y_{t-1}$$

Integrated I(d)

Het ARIMA-model werkt beter op stationaire data, daarom wordt er een preprocessingstap uitgevoerd om de data stationair te maken. Deze preprocessingstap kan echter ook direct door het ARIMA-model worden uitgevoerd. Het geïntegreerde deel van het model past het differencing d keer toe om de data stationair te maken. Hierbij staat d voor het aantal keer dat differencing moet worden uitgevoerd om de data stationair te maken. Het ARIMA-model converteert de voorspellingen, die zijn gemaakt op de gepreprocesste dataset, terug naar de originele tijdreeks zonder differencing. In deze implementatie wordt echter de gepreprocesste dataset gebruikt, samen met de voorspellingen op die dataset, om te visualiseren. Daarom wordt differencing op voorhand toegepast en wordt deze parameter niet gebruikt.

Moving Average MA(q)

Moving Average maakt voorspellingen op basis van het gemiddelde plus een lineaire combinatie van de residuen van de vorige q punten. Een residue is het verschil in waarde tussen het originele datapunt en het voorspelde datapunt en wordt als volgt berekend, waarbij ε de residue is, t het tijdstip van het huidige datapunt, y_{t-1} het vorige datapunt en $\hat{y}_{t-1}$ het vorige voorspelde datapunt is:

$$\varepsilon_{t-1} = |y_{t-1} - \hat{y}_{t-1}|$$

$\hat{y}_m$ is het punt dat we willen voorspellen, μ het huidige gemiddelde van de tijdreeksdataset, ε de residuen zijn en β de coëfficiënten van het lineair model [13]:

$$\hat{y}_m = \mu + \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + ... + \beta_q \varepsilon_{t-q}$$

Stel $q = 1$ en de coëfficiënt β van dit punt is $\beta = 0$ dan merken we op dat het model alleen de constante als resultaat terug geeft, wat in dit geval het gemiddelde van de tijdreeks (gemeten tot het punt t) is.

$$\hat{y}_m = \mu + 0\varepsilon_{t-1}$$

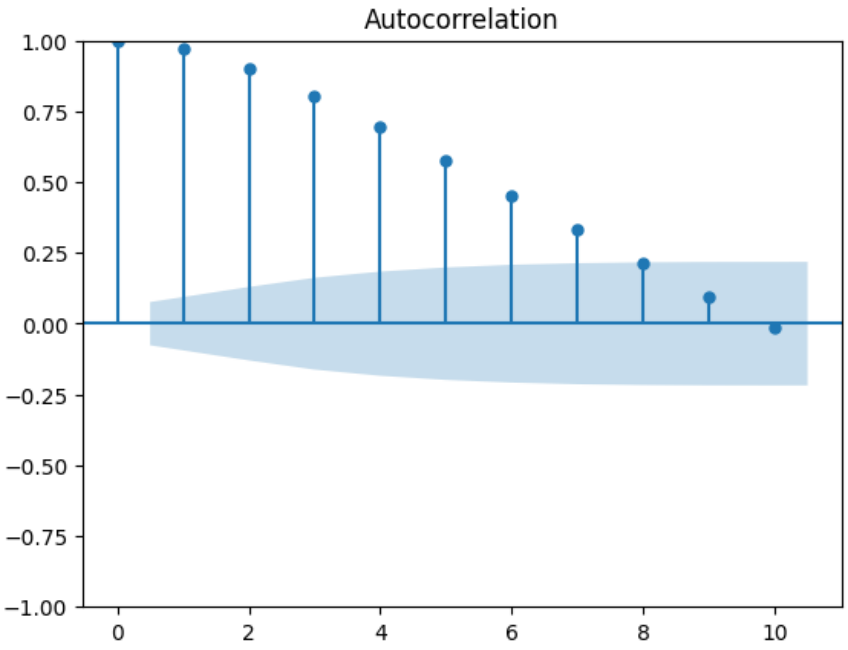
AR+MA

De formule van AR en MA worden als volgt gecombineerd om een voorspelling te maken, waarbij $\hat{y}$ het te voorspellen punt is, c een constante is, t het tijdstip van het huidige punt en een reeks α die de weights zijn van het regressiemodel, μ het huidige gemiddelde van de tijdreeksdataset, ε de residuen zijn en β de coëfficiënten van het Moving Average model.

$$\hat{y} = c + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + ... + \alpha_p y_{t-p} + \mu + \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + ... + \beta_q \varepsilon_{t-q}$$

Autocorrelatie (ACF)

Om waarden te voorspellen moeten zowel de parameter p en q gekend zijn. Autocorrelatie helpt bij het bepalen van q voor het MA-gedeelte van ARIMA. De autocorrelatie berekent de correlatie tussen de datapunten van de tijdreeks en hun voorgaande punten (lags) van dezelfde tijdreeks [14]. De observatie t hangt dus af van de observaties: $t-1, t-2, ..., t-q$. Met andere woorden een voorspelling hangt af van de waarden uit het verleden. Door de autocorrelatie kan over de gehele dataset bekeken worden hoeveel lags uit het verleden belangrijk zijn voor het voorspellen van de nieuwe datapunten. Figuur 4.10 geeft een autocorrelation histogram weer. De parameter q bepaalt dus hoeveel datapunten uit het verleden gebruikt worden om voorspellingen te maken. In dit geval, voor figuur 4.10 zou de parameter q gelijk zijn aan zeven.

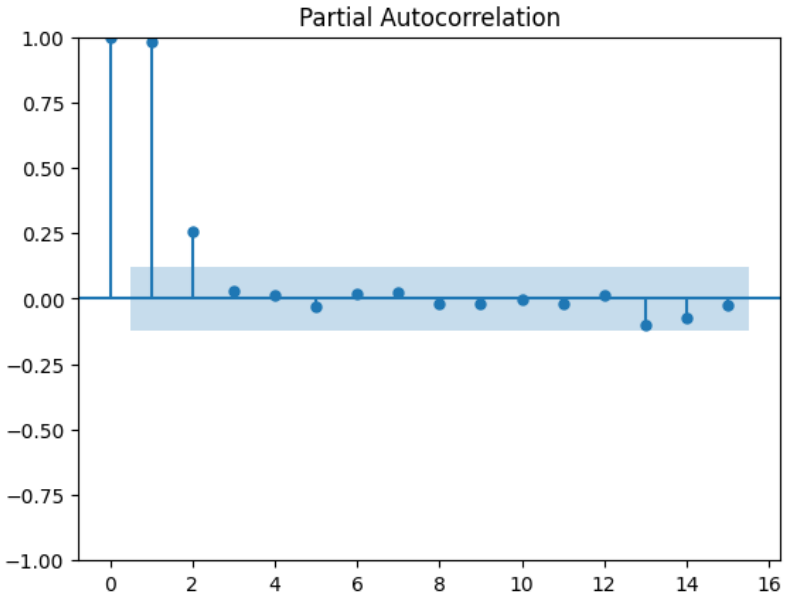


Figuur 4.10: Autocorrelation histogram. De x-as geeft de lags weer terwijl de y-as de significantie weergeeft van de lags.

Zoals men kan afleiden gaat punt nul (het huidige punt) een hoge correlatie hebben met zichzelf. Daarnaast kan men zien dat tot lag twee significant zijn voor het voorspellen van het huidige datapunt. Dat kan men zien omdat de correlatie groter is dan het blauwe oppervlak in figuur 4.10. Als de correlatiewaarden binnen het blauwe oppervlakte vallen, zullen deze niet sterk gecorreleerd zijn met elkaar. Als geen enkel punt buiten dit oppervlakte valt, bestaan er geen correlaties tussen de verschillende datapunten. Dit kan te maken hebben met een random gegenereerde dataset waarbij het niet mogelijk is om datapunten te voorspellen [15].

Partial autocorrelatie (PACF)

De partial autocorrelatie helpt om de parameter p te bepalen. Deze geeft de correlatie weer tussen de datapunten van een tijdreeks en hun voorgaande punten (lags) van dezelfde tijdreeks zonder de indirecte relaties met de voorgaande datapunten [16]. Dit wil zeggen dat een observatie t enkel afhangt van de observatie $t - q$, terwijl bij autocorrelatie hangen de observatie t af van de observaties: $t - 1, t - 2, ..., t - q$. Parameter p zegt dus hoeveel datapunten uit het verleden gebruikt worden om voorspellingen te maken.



Figuur 4.11: Partial autocorrelatie histogram. De x-as geeft de lags weer terwijl de y-as de significantie weergeeft van de lags.

4.3.3 Detecteren van anomalieën

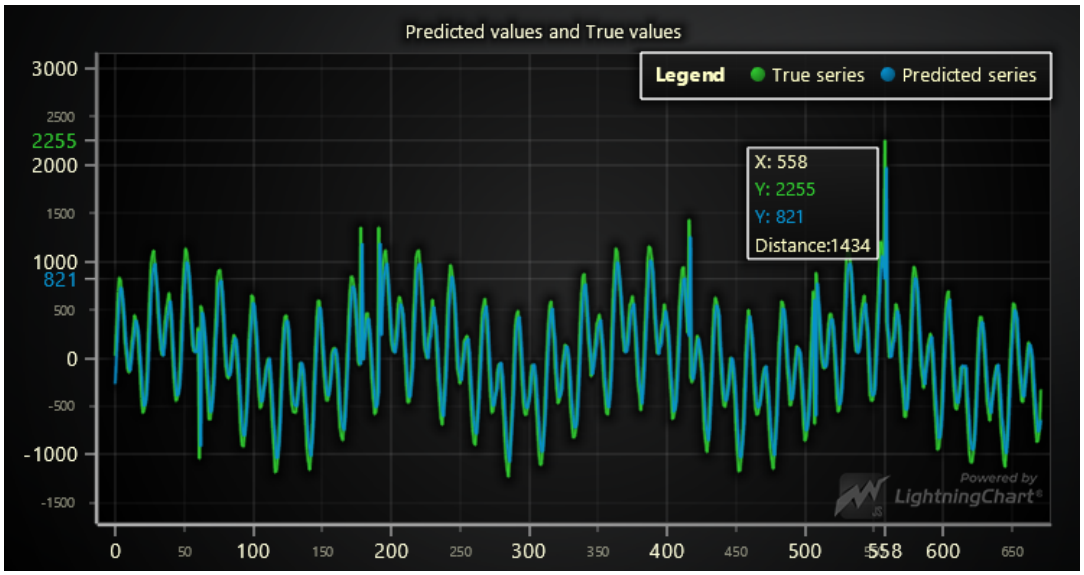
Om anomalieën te detecteren worden er eerst voorspellingen gemaakt met behulp van het ARIMA-model, waarna ze worden vergeleken met de originele waarden. Wanneer het verschil tussen een originele waarde uit de tijdreeks en de voorspelde waarde groter is dan een threshold, wordt dat punt gelabeld als anomalie. Het berekenen van de afstanden gebeurt als volgt, waar $original\_values$ de originele waarden zijn van de tijdreeks, $predictions$ de voorspelde waarde zijn, n het aantal waardes en i de huidige waarde. Daarnaast wordt voor elke afstand de absolute waarde genomen omdat er in dit geval alleen een bovengrens als

$$distances = [abs(original\_values_i - predictions_i)]_{i=1}^{n}$$

Eenmaal de afstanden gekend zijn kan er met behulp van de interkwartielafstand een threshold berekend worden, $q25$ is het eerste kwartiel van de berekende afstanden, $q75$ is het derde kwartiel en iqr de interkwartielafstand tussen het derde en het eerste kwartiel. Er wordt nu de bovengrens gebruikt omdat we alleen geïnteresseerd zijn in afstanden die hoger liggen dan alle anderen:

$$threshold = q75 + (1.5 * iqr)$$

Figuur 4.12 geeft de voorspelde waarde (in het blauw) samen met de overeenkomstige waarde uit de gegeven tijdreeks (in het groen).



Figuur 4.12: Voorspelde waarde (in het blauw) samen met de overeenkomstige waarde uit een tijdreeks (in het groen). De waarden worden samen met de afstand weergegeven op een kader in de grafiek.

4.4 DWT-MLEAD

Discrete Wavelet Transformation and Maximimum Likelihood Estimation for Anomaly Detection is een anomaliedetectie algoritme bestaande uit vier stappen [17]:

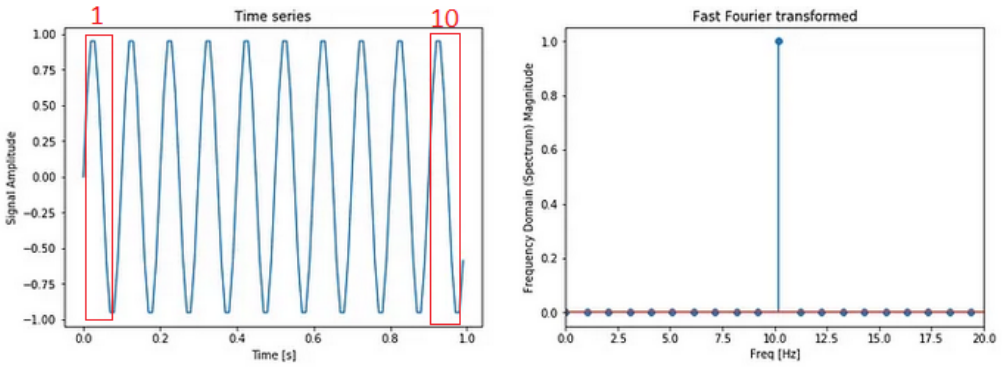
1. Decompositie van het signaal (discrete wavelet transformaties)
2. Sliding windows
3. Maximimum Likelihood Estimation
4. Berekenen van anomalieën

Wavelet-transformaties worden gebruikt om een decompositie te maken van een gegeven inputsignaal (tijdreeks) in zowel het tijds- als frequentiedomein. Voor een beter inzicht wordt er eerst uitgelegd waarom dit soort transformaties geschikt zijn voor deze taak.

4.4.1 Frequentieanalyse in tijdreeksen

Een tijdreeks kan ook voorgesteld worden in het frequentiedomein door er frequentieanalyse op uit te voeren. Een frequentiedomein representeert de frequentiecomponenten van een tijdreeks. Deze componenten stellen voor hoe vaak een gebeurtenis zich afspeelt binnen een tijdreeks. Een voorbeeld hiervan is het resultaat van een Fourier-transformatie, waarbij er berekend wordt hoe vaak een herhaaldelijke gebeurtenis (cyclussen) zich afspelen binnen een tijdreeks door deze te gaan ontbinden met behulp van sinusfuncties. De linker grafiek op Figuur 4.13 geeft het tijdsdomein weer (tijdreeks) en de rechter grafiek op figuur 4.13 geeft het frequentiedomein ervan weer. De y-as op de rechter grafiek stelt de amplitude voor en de x-as de frequentie. De frequentie en amplitude voor de tijdreeks op figuur 4.13 worden als volgt berekend:

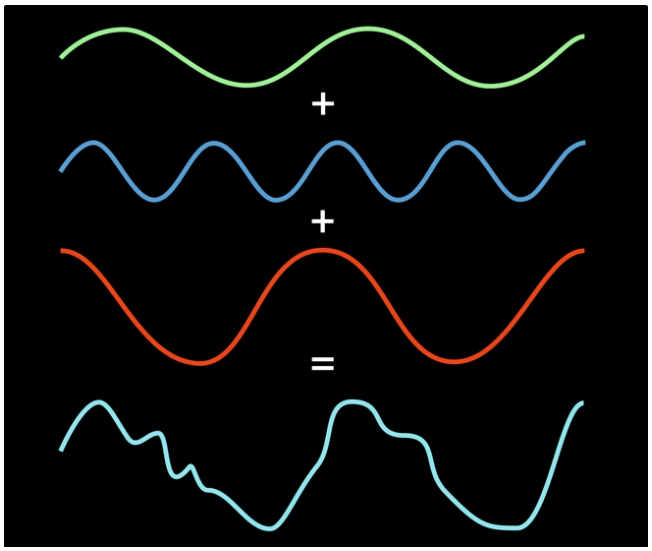
1. Bereken hoe vaak een cyclus zich afspeelt binnen een bepaalde tijdreeks (in dit geval 1 seconde).
2. Bepaal de hoogte van de cyclus.



Figuur 4.13: Voorbeeld van Fourier-transformatie, waarbij de linker figuur $f(t)$ voorstelt en de rechter figuur $F(w)$ [18].

Door de tijdreeks te ontbinden in sinusfuncties kunnen we berekenen hoe dikwijls een cyclus voorkomt. Dit is mogelijk omdat we weten hoe dikwijls deze cyclus binnen elke sinusfunctie voorkomt. De tijdreeks op figuur 4.13 heeft geen decompositie nodig omdat deze op zich al een sinusfunctie is. We zien dus dat er één cyclus is die zich tien keer afspeelt, dit wordt weergegeven aan de hand van de rode cirkels op de frequentiegrafiek in figuur 4.13.

Natuurlijk is het niet altijd zo dat een tijdreeks voorgesteld wordt door één sinusfunctie. Figuur 4.14 geeft weer wat de overeenkomstige sinusfuncties zijn voor een bepaalde tijdreeks. Waarbij het voor elke sinusfunctie mogelijk is om de frequentie te berekenen. Uit elk van deze sinusfuncties kan men de uiteindelijke amplitude afleiden van het frequentiedomein voor een gegeven tijdreeks.



Figuur 4.14: Fourier-transformatie waarbij elke sinusfunctie (groen, blauw, rood) samengevoegd kan worden om de originele tijdreeks voor te stellen [19].

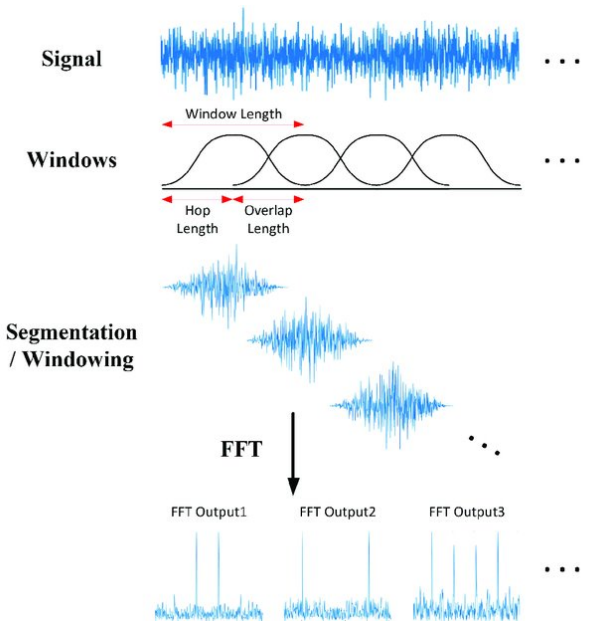
4.4.2 Nadelen van een Fourier-transformatie

Een Fourier-transformatie [20], zoals eerder vermeld, is een wiskundige techniek die wordt gebruikt om een signaal te ontleden naar een frequentiedomein. Nadeel aan een Fourier-transformatie, deze houdt geen rekening met de tijd wanneer een frequentie zich afspeelt. Frequentie is de hoeveelheid van herhaaldelijke gebeurtenissen in een tijdreeks of een gegeven signaal. Het uitvoeren van een Fourier-transformatie is geschikt voor stationaire data omdat er geen rekening wordt gehouden met wanneer een frequentiecomponent zich afspeelt in de tijd. De formule van een Fourier-transformatie wordt hieronder weergegeven, waarbij $f(t)$ de tijdreeks is en $F(w)$ het resultaat van het uitvoeren van de Fourier-transformatie en stelt de componenten in het frequentiedomein voor:

$$F(w) = \int_{-\infty}^{+\infty} f(t)e^{-iwt}\,dt \tag{4.1}$$

Short-Time Fourier-transformatie

de Short-Time Fourier-transformatie is een uitbreiding van de Fourier-transformatie. Deze transformatie gaat de data opsplitsen in stukken (windows) waarbij er op elk stukje data een Fourier-transformatie wordt uitgevoerd. Het is dus mogelijk om een niet-stationaraire tijdreeks als input te hebben maar elk window moet wel stationair zijn. Zoals weergegeven in figuur 4.15 wordt van een gegeven inputsignaal windows gemaakt van een bepaalde grootte, om daarna een Fourier-transformatie uit te voeren op elk van deze windows.

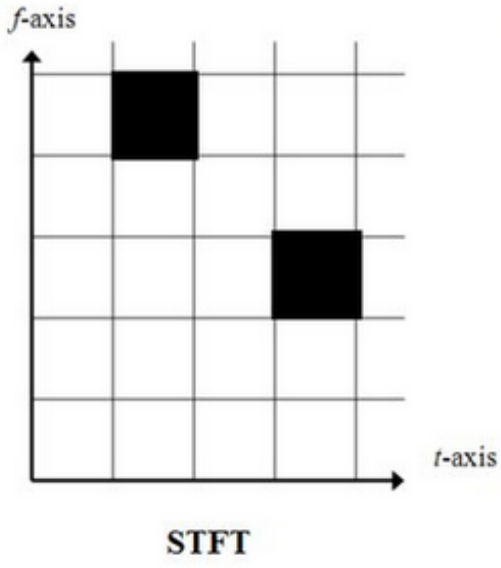


Figuur 4.15: Short-Time Fourier-transformatie [21].

Wanneer we de functie van de Fourier-transformatie 4.1 vergelijken met die van de Short-Time transformatie 4.2 ziet men dat de berekening hetzelfde blijft. Enkel wordt er nu een window functie aan toegevoegd die vermenigvuldigd wordt met het origineel signaal. Deze functie heeft een parameter τ die ervoor zorgt dat het window over het gegeven signaal schuift. Op elke verschuiving van het window wordt er een standaard Fourier-transformatie berekend. Wiskundig gezien kan de Short-Time Fourier-transformatie als volgt berekend worden, waarbij functie $f(t)$ het input signaal is, τ bepaalt de translatie van het gegeven window, een functie W die het window berekent en een exponent e^{-iwt} [22]:

$$F(\tau, w) = \int_{-\infty}^{+\infty} f(t)W(t-\tau)e^{-iwt}\,dt \tag{4.2}$$

Het nadeel van deze transformatie is dat het window, waarmee men over het signaal schuift, een vaste grootte heeft. Daarnaast, binnen elk window, is er niet geweten welke frequenties wanneer afspelen. Figuur 4.16 geeft weer dat op eender welk moment met eender welke frequentie de frequentie- en tijdsresolutie vast opgesplitst zijn. Hoe smaller deze rechthoeken hoe meer tijdsinformatie er kan worden afgeleid, maar hoe slechter de frequentie-informatie kan worden afgeleid. Het omgekeerde geldt wanneer de rechthoeken heel breed zijn.

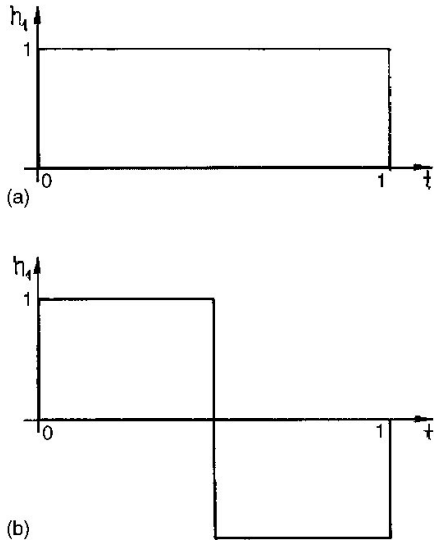


Figuur 4.16: Tijd-/Frequentiedomein van een Short-Time Fourier-transformatie.

4.4.3 Discrete wavelet transformatie (DWT)

Discrete wavelet transformatie (DWT) bestaat uit een wavelet functie die schuift over een gegeven tijdreeks en enkel berekeningen maakt op dat stukje van de tijdreeks waardoor er zowel frequentie-informatie als tijdsinformatie opgehaald wordt. Fourier-transformaties bestuderen de tijdreeks in zijn geheel terwijl DWT de tijdreeks deel per deel analyseert. Een wavelet kan geschaald worden in verschillende groottes en lengtes. Omdat men de tijdreeks in aparte delen bestudeert is het mogelijk om bij te houden welke informatie uit het frequentie-domein zich afspeelt in het tijdsdomein. Door de mogelijkheid van schaling van een wavelet, kunnen verschillende groottes van windows worden geanalyseerd wat niet het geval is bij Short-Time Fourier-transformatie.

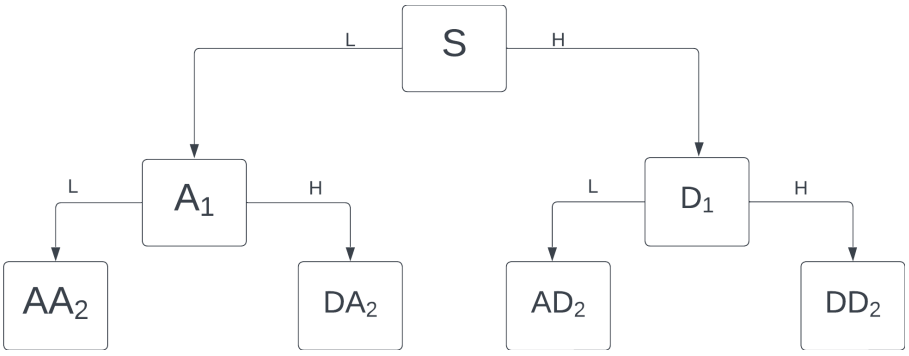
DWT ontleedt een tijdreeks in verschillende componenten door gebruik te maken van een verzameling van basisfuncties (wavelets). Wavelets zijn een set van wiskundige functies die een tijdreeks ontleden in verschillende frequenties op verschillende schalen waarbij de schalen de verschillende tijdsintervallen van de tijdreeks zijn. Elke wavelet transformatie bestaat uit een high pass en een low pass filter. De high pass filter refereert naar de *mother wavelet* en de low pass filter naar de father wavelet, ook wel de scaling functie genoemd. Figuur 4.17.a geeft een mother wavelet weer terwijl figuur 4.17.b een scaling functie weergeeft van een haar wavelet. Omdat een wavelet geschaald wordt moet de data niet stationair zijn. Er is dus geen preprocessingstap nodig die het originele signaal omvormt naar een stationair signaal.



Figuur 4.17: (a) is de scaling functie en (b) is de wavelet functie van een haar wavelet [23].

Het DWT algoritme is een iteratief algoritme dat meerdere keren op het resultaat van de vorige iteratie uitgevoerd wordt. In de eerste iteratie wordt de gehele tijdreeks genomen als input voor de high en de low pass filter. Het resultaat van de high pass filter zijn de gedetailleerde coëfficiënten. Het resultaat van de low pass filter zijn de benaderde (approximated) coëfficiënten. Na het uitvoeren van de high of low pass filter zal het aantal datapunten van de benaderde of van de gedetailleerde coëfficiënten de helft zijn van die uit het vorige level. De lengte van beide coëfficiënten samen is terug de lengte van het aantal datapunten uit het vorige level. Dit wordt iteratief gedaan tot het aantal datapunten gelijk is aan twee.

Figuur 4.18 geeft het iteratief proces weer van het DWT algoritme waarbij S de originele tijdreeksdataset is, A het resultaat is van de low pass filter, L en D het resultaat van de high pass filter H. Bijvoorbeeld $AD2$ duidt aan dat het signaal eerst door de high pass filter is gegaan wat resulteert in $D1$ met 1 het eerste level en daarna door een low pass filter wat resulteert in $AD2$ met 2 als level twee. Het maximaal aantal levels is $log_2(len(S))$ met $len(S)$ het aantal datapunten in de tijdreeks S.



Figuur 4.18: DWT-boomstructuur [24].

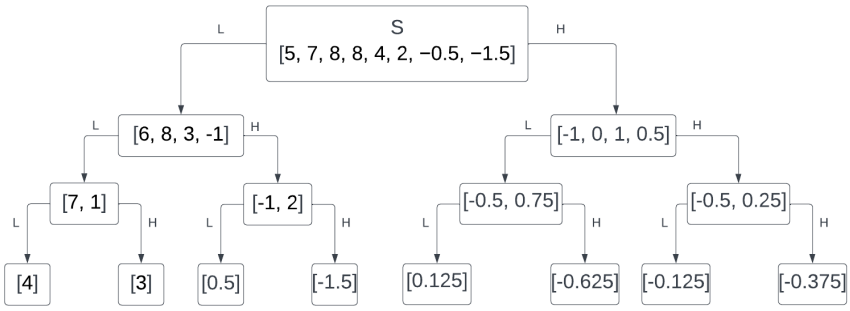
Een iteratieve decompositie met gebruik van haar wavelets verloopt als volgt, gegeven de high pass filter H_i en de low pass filter L_i toegepast op X. Met X als input voor de high en de low pass filter. Voor de eerste iteratie zal X gelijk zijn aan de tijdreeks S. Voor de tweede iteratie zal X gelijk zijn aan de benaderde/gedetailleerde coëfficiënten die afkomstig zijn van de eerste iteratie. Dit kan iteratief uitgevoerd worden tot de nodes van het laatste level telkens één datapunt bevat. Merk op dat Haar wavelets niet de enige mogelijke familie van wavelets zijn die gebruikt kunnen worden:

$$L_i = \frac{X_{2i} + X_{2i+1}}{2} \; H_i = \frac{X_{2i} - X_{2i+1}}{2} \;\; voor : i = 0, ..., \frac{len(X)}{2}$$

We illustreren de werking aan de hand van de volgende tijdreeks (deze worden ook gebruikt in figuur 4.19).

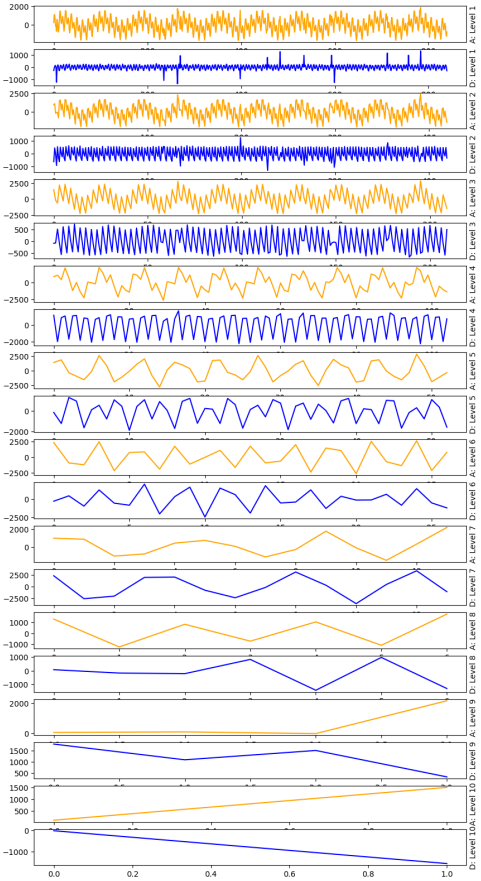
$$X = [5,\ 7,\ 8,\ 8,\ 4,\ 2,\ -0.5,\ -1.5]$$

De binaire boom in figuur 4.19 geeft het resultaat weer van het DWT algoritme, hier is terug L de low pass filter en H de high pass filter. Merk op dat de dataset op elk level gehalveerd wordt tot er maar één datapunt meer overblijft met als root node de originele tijdreeks S.

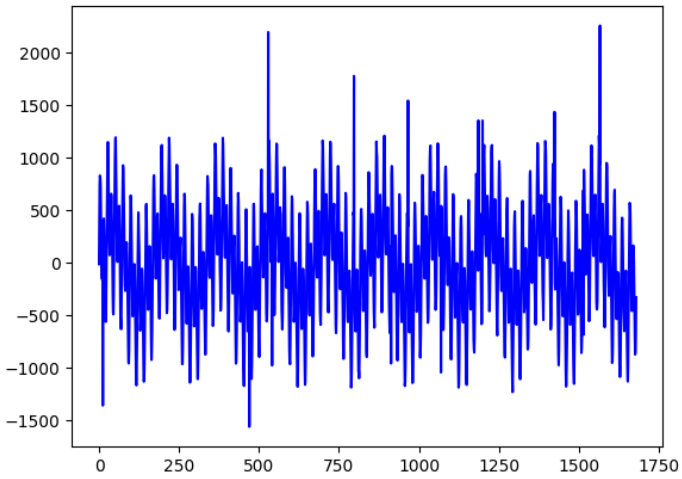


Figuur 4.19: Uitgewerkte boomstructuur van het DWT algoritme op een verzameling van punten genoteerd als S.

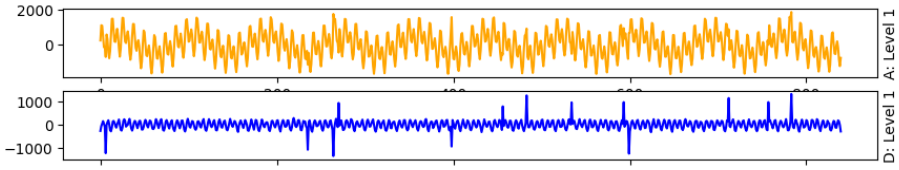
Figuur 4.20 toont per level de gedetailleerde en de benaderde coëfficiënten voor een echte tijdreeksdataset uit figuur 4.21. Het algoritme laat toe om vanuit verschillende standpunten naar een gegeven tijdreeks te kijken. Als men kijkt naar het eerste level en de benaderde coëfficiënten vergelijkt met die van de gedetailleerde coëfficiënten kan men uit beide andere details afleiden. Zo zal men uit de gedetailleerde coëfficiënten het verschil tussen twee opeenvolgende datapunten kunnen bekijken. Wanneer een datapunt plots veel hoger is dan het vorige datapunt wordt dit weergegeven in de gedetailleerde coëfficiënten. De benaderde coëfficiënten geven ons een benaderde weergave van het oorspronkelijk signaal (tijdreeks). Figuur 4.22 zoomt in op de eerste benaderde en gedetailleerde coëfficiënten van level 1.



Figuur 4.20: Oranje grafieken zijn de benaderde coëfficiënten en de blauwe de gedetailleerde voor de dataset uit figuur 4.21, de x-as is genormaliseerd zodat op elk level de grafiek dezelfde grootte heeft.

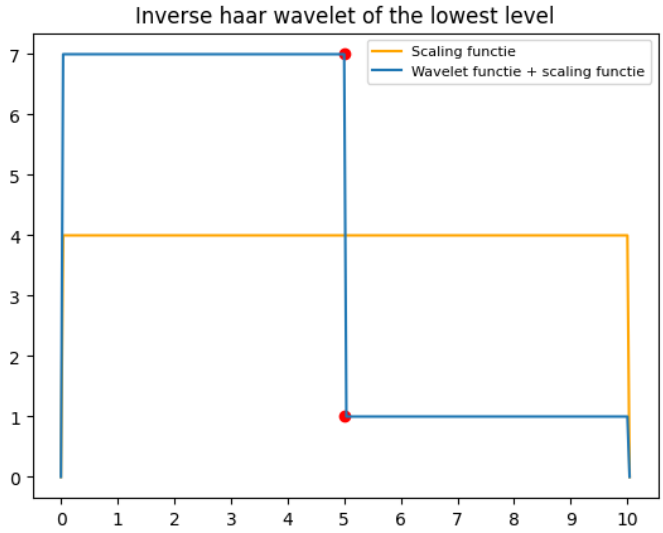


Figuur 4.21: Een gegeven tijdreeks.



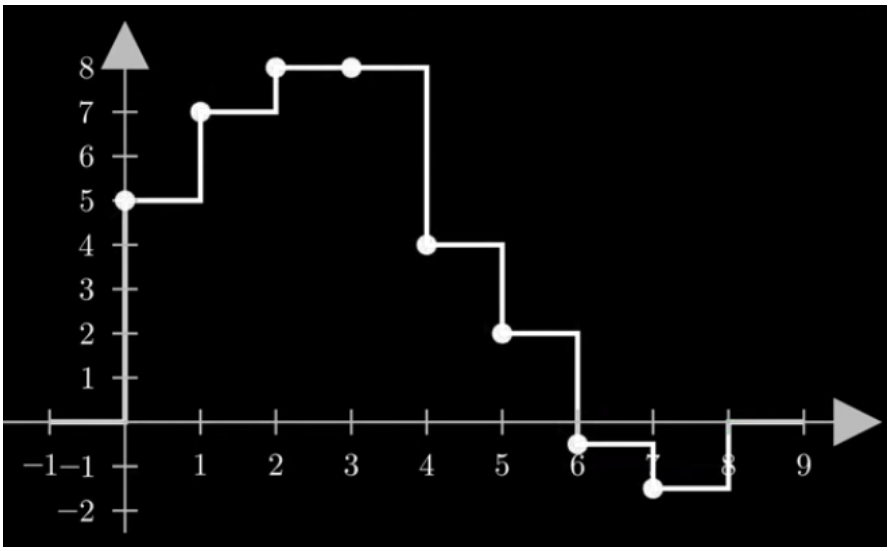
Figuur 4.22: De oranje grafiek zijn de benaderde coëfficiënten terwijl de blauwe de gedetailleerde zijn voor het eerste level van figuur 4.20.

Het is ook mogelijk om het originele signaal te reconstrueren met de informatie uit alle levels. Om het signaal te reconstrueren wordt er gestart vanaf de leaf nodes (het laagste level). Dit is dus omgekeerd bij de decompositie, deze start van de root node. Vanuit het laagste level (leaf nodes) kan heel het signaal gereconstrueerd worden door alleen gebruik te maken van de wavelet, scaling functie en de coëfficiënten. Voor elk level wordt er in eerste instantie de scaling functie vermenigvuldigd met de benaderde coëfficiënten van dat level. Daarna wordt de wavelet functie vermenigvuldigd met de gedetailleerde coëfficiënten van dat level. Deze twee resultaten worden bij elkaar opgeteld. Figuur 4.23 laat zien wat het resultaat is na de eerste inverse iteratie, de oranje grafiek is de scaling functie die vermenigvuldigd is met de benaderde coëfficiënt van het eindlevel in figuur 4.19 die als waarde vier heeft. De blauwe grafiek is het resultaat nadat de wavelet functie werd vermenigvuldigd met de gedetailleerde coëfficiënt drie, van het eindlevel in figuur 4.19 en daarna werd opgeteld bij de scaling functie. De y-waarde van de punten die in het rood worden aangegeven corresponderen met de benaderde coëfficiënten van level twee. Men kan in principe ook vanaf eender welk level starten om opnieuw het originele signaal te bekomen.



Figuur 4.23: Inverse van de haar wavelet transformatie voor het laagste level.

Als eindresultaat kan heel de dataset opnieuw gereconstrueerd worden aan de hand van deze functies. Figuur 4.24 geeft weer hoe een heel signaal opgebouwd is door gebruik te maken van de coëfficiënten en de wavelet/scaling functie startende vanuit de leave nodes tot aan de originele tijdreeks.



Figuur 4.24: Inverse van de haar wavelet transformatie voor alle levels. De x-as representeert het tijdstip (in dit geval de index) en de y-as representeert de overeenkomstige waarde.

4.4.4 Sliding windows

Het resultaat van de vorige stap, namelijk de coëfficiënten die per level berekend worden zoals aangegeven op figuur 4.20, waarbij level 0 (root node) de originele tijdreeks is. Deze worden nu opgedeeld in verschillende windows. De grootte van de windows verschilt per level. Zo zullen windows in eerdere levels groter zijn dan windows in latere levels. Dit is omdat de lengte van de coëfficiënten per level halveert. Men kan dus niet dezelfde grootte van windows gebruiken in level één als in de laatste levels. De grootte van de coëfficiënten voor level één kan bestaan uit bijvoorbeeld duizend(en) datapunten terwijl het voorlaatste level bestaat uit twee datapunten. Men kan dus bijvoorbeeld voor level één een window van grootte zes hebben en voor het voorlaatste level een window van grootte één. Een window van zes datapunten zou dus niet mogelijk zijn in het voorlaatste level. De grootte van de windows wordt als volgt berekend, waarbij $l \in \{l', ..., E\}$ en $E = log_2(len(S)) - 1$ het maximum level is min één (het laatste level bestaat uit één enkel datapunt en kan dus niet worden opgedeeld in windows), $len(S)$ het aantal datapunten in een tijdreeks S is en l' het startlevel is [25]:

$$max(2, E - l - l' + 1)$$

Het startlevel l' is een willekeurig gekozen level dat kleiner moet zijn dat het eindlevel $1 < l' < E$, met 0 als root wat dus de originele tijdreeks vertegenwoordigt en E als maximum level. Het startlevel bepaalt vanaf welk level de coëfficiënten in overweging worden genomen bij het detecteren van anomalieën. Coëfficiënten die in hogere levels liggen (dus dichter bij de root) zullen niet opgenomen worden om anomalieën te detecteren. Het huidige level wordt aangeduid met l dit bepaalt in welk level men zich bevindt. Om een voorbeeld te geven wordt de verzameling van benaderde coëfficiënten gebruikt van level één uit figuur 4.19 namelijk:

$$[6, 8, 3, -1]$$

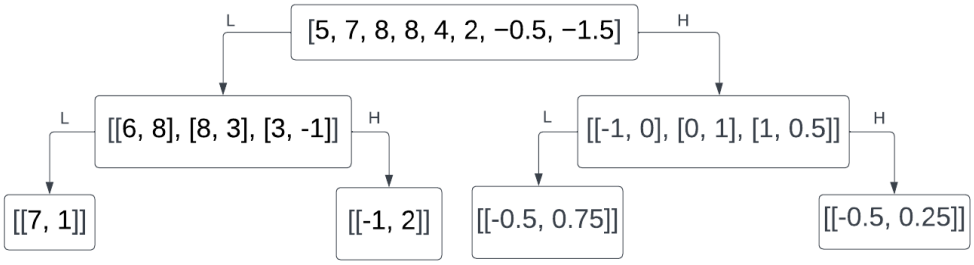
De grootte van elk window is gelijk aan twee want $max(2, 2 - 1 - 1 + 1) = 2$, waarbij het startlevel $l' = 1$ is, het huidig level $l = 1$ is (we nemen de coëfficiënten van level één). E is in dit voorbeeld $E = 2$.

$$W1: [6, 8]$$

$$W2: [8, 3]$$

$$W3: [3, -1]$$

Dit proces wordt op elk level voor zowel de gedetailleerde als de benaderde coëfficiënten uitgevoerd. Figuur 4.25 toont het resultaat van een gegeven DWT-decompositie uit figuur 4.19. De root stelt nog steeds de originele tijdreeks voor waarbij de nakomelingen van de root de coëfficiënten zijn die opgedeeld zijn in windows. Merk op dat het laatste level weggelaten is. Het is namelijk niet mogelijk om met een window van twee over één datapunt te schuiven, daarom wordt het laatste level uit ons resultaat gehaald. Het resultaat na het opsplitsen in windows, zal in de volgende paragraaf gebruikt worden om een multivariate gaussian distribution op te maken.



Figuur 4.25: Sliding window representatie van de originele DWT-boomstructuur uit figuur 4.19. Het laatste level is weggelaten omdat we de verzameling ter grootte van 1 niet gebruiken om anomalieën te detecteren. De root node stelt nog steeds de originele tijdreeks voor.

4.4.5 Maximum Likelihood Estimation

In deze sectie willen we berekenen hoe waarschijnlijk het is dat een window binnen een reeks van windows valt. Per level zijn er nu 2 reeksen met windows namelijk de windows van de gedetailleerde coëfficiënten en de windows van de benaderde coëfficiënten. Waarbij m het maximum level - 1 is (laatste heeft maar één datapunt en wordt dus niet verder gebruikt) en j het aantal windows voor level één. Level m bestaat maar uit twee datapunten en zal dus maar één window hebben. Level nul bestaat uit de originele tijdreeks.

$$Level1 : A[window_1, ..., window_j],\ D[window_1, ..., window_j]$$

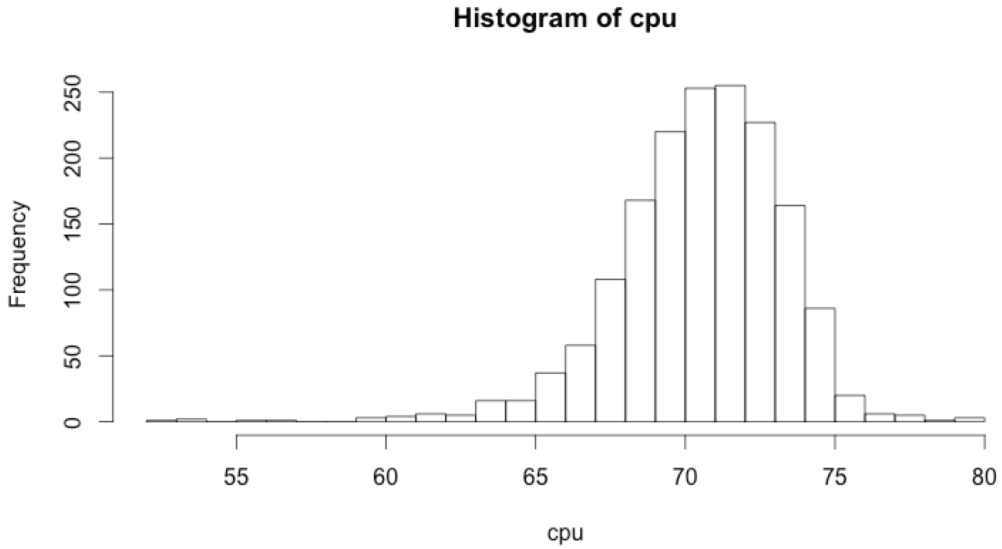
$$...$$

$$Level_m : A[window_1],\ D[window_1]$$

Met deze resultaten is het mogelijk om voor elk level, per reeks van windows, een multivariate Gaussian-distributie te berekenen. Deze distributie helpt ons om de verdeling van de windows binnen één reeks te bepalen. Met behulp van deze verdeling kunnen we detecteren welke windows anomalie zijn en welke niet. Merk op dat MLE niet de enigste techniek is om anomalie windows te detecteren. Het is mogelijk om andere algoritmes toe te passen zoals clusteringtechnieken die deze windows kan clusteren om zo anomalie windows te detecteren. In de volgende paragrafen bespreken we hoe de multivariate Guassian-distributie berekenen.

Distributie van data

Een distributie refereert naar het patroon of vorm van de data wanneer we deze in een grafiek plotten. Het beschrijft hoe de data verdeeld of geclusterd zijn rond verschillende waarden. Figuur 4.26 is een voorbeeld van een distributie van het gebruik van een CPU. We kunnen hieruit aflezen dat deze CPU meestal een gebruik rond de 70% heeft. Met behulp van deze distributie is het nu mogelijk om anomalieën te detecteren. Uit figuur 4.26 kan men aflezen dat er enkele keren een gebruik onder de 60% gemeten is, dit kan duiden op een anomalie. Als het CPU-gebruik onder de 60% is, kan dit aanduiden op een applicatie die gecrasht is of deze opnieuw opgestart is. Een verbruik hoger dan 75% kan dan weer aanduiden dat de CPU overbelast werd, wat dus ook een anomalie is [26].

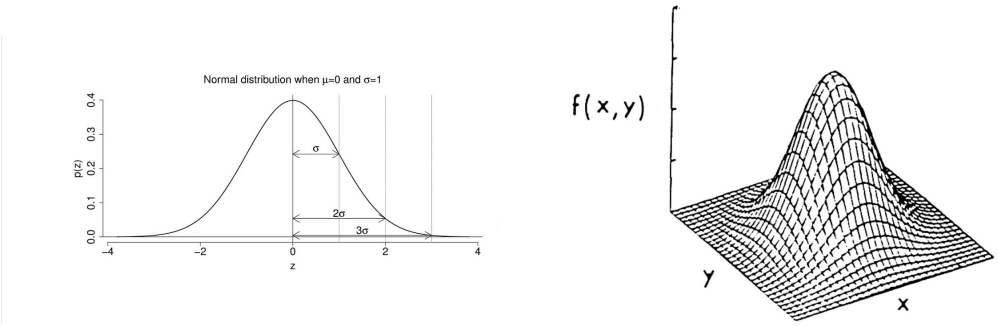


Figuur 4.26: Distributie van een tijdreeks bestaande uit CPU-metingen [26].

Merk op dat de verdeling uit figuur 4.26 bestaat uit univariate waarden. Omdat men bij DWT-MLEAD voor elke reeks van windows een distributie willen berekenen is het niet mogelijk om een univariate Guassian-distributie te gebruiken. De coëfficiënten zijn hier namelijk opgedeeld in windows, daarom volstaat een univariate Guassian-distributie niet. Als oplossing wordt er een multivariate Gaussian-distributie berekent. Dit maakt het mogelijk om een distributie te maken voor een reeks van vectoren (windows).

Multivariate gaussian-distributie

Per reeks van windows worden de parameters van een multivariate Gaussian-distributie berekend. Een standaard Gaussian-distributie zou hier niet functioneren omdat elk window binnen een reeks van windows een vector met grootte n is, waarbij n de grootte van het window voorstelt [25]. Een multivariate Guassian-distributie is een Guassian-distributie met meerdere variabelen. Figuur 4.27 geeft als voorbeeld aan de linkerkant een univariate Guassian-distributie en aan de rechterkant een multivariate Gaussian-distributie weer.



Figuur 4.27: De linker Guassian-distributie is een univariate distributie waarbij σ de standaardafwijking is en μ het gemiddelde. De rechtergrafiek is een bivariate distributie met x en y de features en $f(x, y)$ de frequentie van beide features [27].

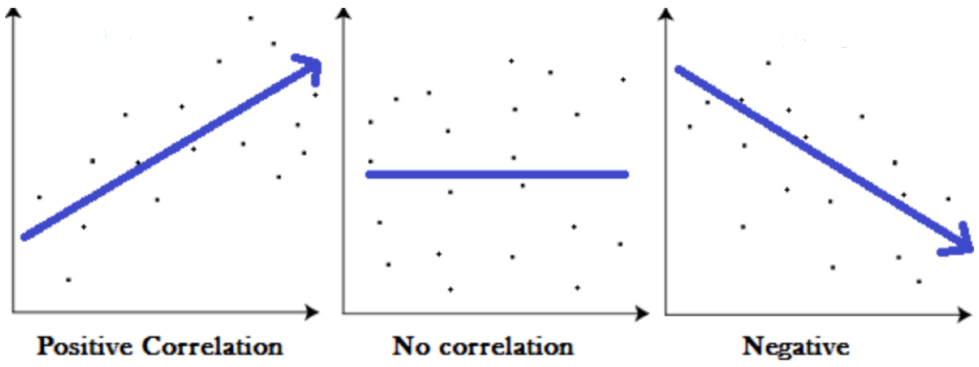
De parameters van de multivariate Gaussian-distributie worden later in het proces gebruikt om te bepalen welk window binnen dezelfde reeks van windows anomalie is. Om verder te gaan op figuur 4.25 zal nu voor elke reeks van coëfficiënten, vanaf level één, een multivariate Gaussian-distributie opgemaakt worden. Merk op dat het laatste level, dus de bladeren van de boom, uit figuur 4.25 maar één window bevat, het is dus zinloos om een multivariate Gaussian-distributie te bereken voor een verzameling met grootte één.

Voor een standaard Gaussian-distributie worden er twee parameters berekend namelijk de standaardafwijking en het gemiddelde van een reeks getallen. Bij een multivariate Gaussian-distributie worden er ook twee parameters berekend namelijk: het gemiddelde van alle windows en een covariantiematrix die berekend wordt op basis alle windows [28]. Het gemiddelde berekenen van alle vectoren (windows) gebeurt door een *pointwise* gemiddelde te nemen, stel u en v zijn beide vectoren met dezelfde grootte n:

$$u = (u_0, u_1, ..., u_n),\ v = (v_0, v_1, ..., v_n)$$

$$mean(u, v) = \frac{\sum_{j=1}^{n} u_j + v_j}{2}$$

De covariantie berekent het lineair verband tussen twee variabelen. Een voorbeeld van een covariantie tussen twee variabelen wordt getoond op figuur 4.28.



Figuur 4.28: Positieve, geen en negatieve covariantie tussen twee variabelen.

Als de covariantie voor twee variabelen nul is dan is er geen lineair verband tussen beide variabelen. Met andere woorden de ene variabele heeft geen invloed op de andere. Wanneer de ene variabele daalt bestaat er geen lineair verband waardoor de andere zal dalen of stijgen. Een positieve covariantie betekent, als één van de twee variabelen toeneemt in waarde, de

andere variabele ook toeneemt. Een negatieve covariantie betekent dat er een inverse relatie bestaat tussen beide variabelen. Als de ene stijgt zal de andere dalen. De diagonalen gaan dus een lineair verband berekenen tussen een variabele en zichzelf, dit is de variantie van die waarde. Veronderstel dat windows binnen een bepaald level van coëfficiënten drie punten bevat, en de sequentie van windows $[X, Y, Z], ..., [X_z, Y_z, Z_z]$ waarbij $\hat{X}$ het gemiddelde is van alle X-waarden over alle windows, $\hat{Y}$ het gemiddelde is van alle Y-waarden van alle windows, $\hat{Z}$ het gemiddelde is van alle Z-waarden van alle windows en z het aantal windows zijn. Deze gemiddelden kunnen afgeleid worden uit de gemiddelde vector. Dan is de covariantie van de drie variabelen [29]:

$$cov(X, Y) = \frac{\sum_{i=1}^{z}(X_i - \bar{X})(Y_i - \bar{Y})}{z}$$
$$cov(X, Z) = \frac{\sum_{i=1}^{z}(X_i - \bar{X})(Z_i - \bar{Z})}{z}$$
$$cov(Y, Z) = \frac{\sum_{i=1}^{z}(Y_i - \bar{Y})(Z_i - \bar{Z})}{z}$$

Onderstaande matrix geeft weer wat het resultaat kan zijn voor het berekenen van de covariantie van een reeks windows waarbij elke window bestaat uit drie variabelen, die overeenkomen met variabelen X, Y, Z respectievelijk.

$$\begin{bmatrix} variance(X) & covariance(X,Y) & covariance(X,Z) \\ covariance(Y,X) & variance(Y) & covariance(Y,Z) \\ covariance(Z,X) & covariance(Z,Y) & variance(Z,Z) \end{bmatrix}$$

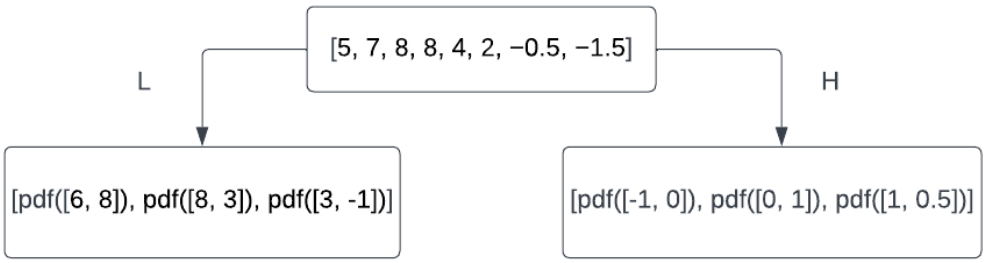
Likelihood

Eenmaal de parameters voor de multivariate Gaussian-distributie gekend zijn, kan er voor elke window binnen een reeks van windows een probability density waarde berekend worden. Deze waarde geeft aan wat de kans is dat deze binnen een distributie valt. Een hoge score gaat dus aangeven dat er veel andere windows zijn die ongeveer dezelfde waarden hebben. Windows die gedragen als een anomalie hebben niet veel andere windows die dezelfde waarde hebben. Deze anomalie windows hebben dus een lagere probability density score. Een ander voorbeeld hiervan is, stel je hebt een reeks van windows en één window wijkt heel hard af van alle andere windows dan zal deze een lage probability density score hebben omdat er weinig andere windows zijn die dezelfde waarden hebben.

Als we dit voor de gedetailleerde coëfficiënten windows en de benaderde coëfficiënten windows uitvoeren op de verschillende levels krijgt men onderstaand resultaat waarbij *pdf* de probability density functie is, $m - 1$ als eindlevel en j het aantal windows in level één is:

$$Level1 : A[pdf(window_1), ..., pdf(window_j)],\ D[pdf(window_1), ..., pdf(window_j)]$$
$$...$$
$$Level_{m-1} : A[pdf(window_1), pdf(window_2)],\ D[pdf(window_1), pdf(window_2)]$$

Als we dit gaan toepassen op het resultaat uit figuur 4.25, bekomen we het resultaat uit figuur 4.29. Merk op dat het laatste level uit figuur 4.25 bestaat uit slechts één window waardoor we voor deze verzameling geen distributie berekend hebben. Als we dit toch hadden gedaan zou de probability density score heel hoog zijn waardoor dit window in het volgende hoofdstuk niet als anomalie beschouwd zou worden. Daarom wordt dit level niet getoond in figuur 4.29.

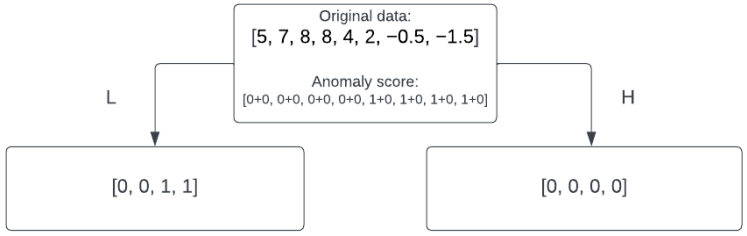


Figuur 4.29: Voor elk van de windows wordt een probabilty density score berekend voor een gegeven multivariate Gaussain-distributie voor die node (voor level 1). Pdf is de probabilty density functie. Level twee is niet getoond omdat het niet zinvol is om een Guassian-distributie te berekenen in een reeks met lengte één.

4.4.6 Anomaliedetectie

Voortgaand op het resultaat van de vorige stap, kan er nu per level per reeks windows een threshold berekend worden op basis van de probability density scores. Als threshold wordt er het eerste percentiel berekend van een reeks scores. Als een window een kleinere score heeft dan de threshold dan kan er geconcludeerd worden dat dit window een anomalie is.

Het resultaat hiervan is een label per window. Met label 1 dat duidt op een anomalie en label 0 dat duidt op geen anomalie. Er wordt een afvlakkingsfunctie uitgevoerd dat de windows opnieuw omvormt tot een verzameling zonder windows, elk punt krijgt nu de label van het window. Zoals aangegeven in sectie 4.4.4 is het mogelijk dat meerdere datapunten in meerdere windows voorkomen. Daarom wordt er een horizontale reductie toegepast om dezelfde datapunten, die in meerdere windows anomalie zijn, samen te nemen en een label één te geven [17]. Wanneer een coëfficiënt (datapunt) in een bepaald window w_t anomalie is maar in het volgende window w_{t+1} niet, dan zal dit punt het label één krijgen. Als laatste wordt er een verticale reductie uitgevoerd om deze labels terug te linken aan hun originele datapunten uit de tijdreeks. Te beginnen van het onderste level van de boom kan men nu de scores per datapunt propageren naar het bovenliggende level tot aan de root node, daar worden dan voor elk van de punten de gepropageerde scores bij elkaar opgeteld. Merk op dat we dus nu starten vanuit de bladeren van de boom en eindigen bij de root node (origine tijdreeks). Het eindresultaat wordt weergegeven in figuur 4.30 waarbij we per datapunt een label nul of één krijgen. Volgende paragrafen verduidelijken hoe we dit eindresultaat bekomen.

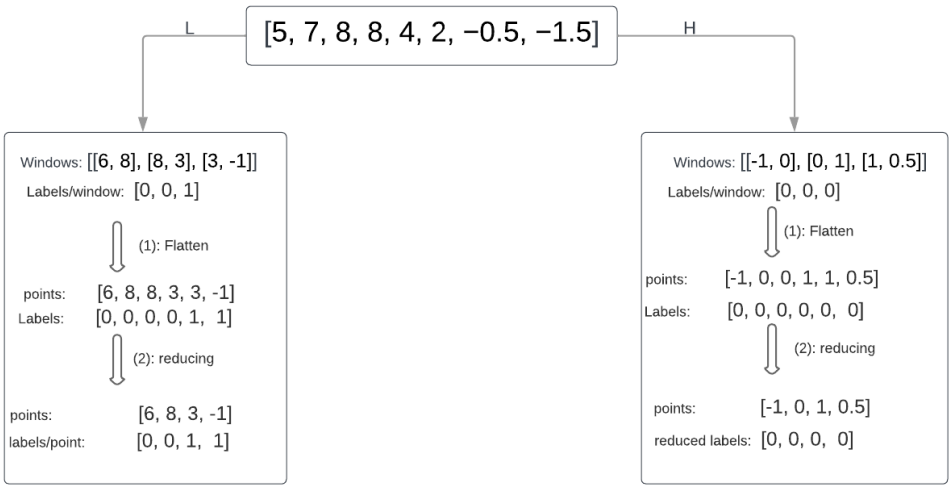


Figuur 4.30: Scores per datapunt voor elk level en elke reeks van coëfficiënten. De root node bevat de originele tijdreeks en de anomalie score per datapunt. De labels uit level één worden bepaald door de pdf te vergelijken met een threshold, als deze waarde kleiner is dan de threshold dan kan men concluderen dat dit window anomalie is. Zoals aangegeven in figuur 4.29 zijn er twee punten in een window en aangezien het laatste window anomalie is zullen de laatste twee punten anomalie zijn en dus label één krijgen.

Horizontale reductie

Om terug het voorbeeld aan te halen uit figuur 4.29, voor de benaderde coëfficiënten heeft men per window een label dat aanduidt of het een anomalie window is of niet. Als voorbeeld labelen we window $[3, -1]$, uit figuur 4.29, als anomalie. Window $[3, -1]$ krijgt dus label één en de andere windows respectievelijk, $[6, 8]$ en $[8, 3]$, label nul. Als we voor elk datapunt binnen elk van deze windows hetzelfde label geven als het label van hun window, krijgen we het resultaat zoals weergegeven op figuur 4.31. Figuur 4.31 geeft voor elke node de stappen weer die nodig zijn om de labels per datapunt voor zowel de gedetailleerde als de benaderde coëfficiënten te berekenen. Omdat elk window, in dit voorbeeld dat bestaat uit twee variabelen, zijn er in totaal zes labels (elk datapunt in elk window krijgt het label van dat window). Merk dus op dat dezelfde datapunten meerdere keren voorkomen.

Hierna kunnen we dus een horizontale reductie uitvoeren om van deze zes labels terug vier labels te maken (label per datapunt van de originele coëfficiënten). Dit doen we door datapunten, die meerdere keren voorkomen, het maximum label te geven. Bijvoorbeeld datapunt 3, bij de benaderde coëfficiënten, komt twee keer voor (één keer in een normaal window en één keer in een anomalie window). Dit datapunt zal dus label één krijgen wat overeenkomt met het hoogste label voor dat datapunt. Een volledig uitgewerkt resultaat kan men vinden in figuur 4.31.



Figuur 4.31: Horizontale reductie per node gebaseerd op het voorbeeld uit figuur 4.29. De root node stelt de originele dataset voor. Voor beide afstammelingen van de rootnode is er als voorbeeld de originele windows voor dat level getoond samen met de labels per window. Stel window $[3, -1]$ is een anomalie en heeft dus label één en al de andere label nul.(1) Afvlakken van de matrix (de matrix is de verzameling van windows) zorgt ervoor dat men nu een verzameling van datapunten heeft waarbij één punt meerdere keren kan voorkomen. Elk punt krijgt het label van dat window. (2) Daarna wordt er gereduceerd zodat elk punt opnieuw één keer voorkomt. Als reductiefunctie wordt er het grootste label genomen dat bij dit punt hoort (0 of 1). Het resultaat is label per datapunt.

Een meer conceptueel voorbeeld wordt hieronder weergegeven: stel, een gegeven level x van gedetailleerde coëfficiënten D waarbij elk window bestaat uit drie datapunten:

$$Levelx : D\,[[y_1, y_2, y_3], [y_2, y_3, y_4], [y_3, y_4, y_5]]$$

Voor elk van deze windows kan men de pdf-score berekenen. Als deze waarde kleiner is dan de threshold ε, dan krijgt dat window het label één en anders nul. In onderstaand voorbeeld veronderstellen we dat windows één en twee een pdf-waarde kleiner dan een threshold hebben. Het derde window heeft een pdf-waarde die hoger is dan een threshold (en dus geen anomalie is).

$$Levelx : D\,[pdf([y_1, y_2, y_3]) < \varepsilon, pdf([y_2, y_3, y_4]) < \varepsilon, pdf([y_3, y_4, y_5]) > \varepsilon]$$

$$Levelx : D\,[1, 1, 0]$$

Vervolgens krijgt elk datapunt binnen een window het label van dat window, voor het bovenstaande voorbeeld krijgt men het volgende resultaat per datapunt. Zoals eerder vermeld kan een datapunt in meerdere windows voorkomen.

y_1 komt enkel voor in het eerste window het bovenstaand voorbeeld heeft maar één label. y_2 komt voor in het eerste en tweede window en zal dus twee labels krijgen. y_3 komt voor in het eerste, tweede en derde window en heeft drie dus labels. y_4 komt voor in het tweede en derde window en heeft dus twee labels. y_5 komt enkel voor in het derde window en heeft dus één label:

$$y_1 : [1]$$
$$y_2 : [1, 1]$$
$$y_3 : [1, 1, 1]$$
$$y_4 : [1, 0]$$
$$y_5 : [0]$$

Omdat datapunt y_2 tot en met datapunt y_4 een lijst van labels heeft, moeten deze herleid worden naar één label per datapunt. De reductie die toegepast wordt is het maximum nemen van alle labels. Als een datapunt in één window gelabeld is als anomalie dan zal deze uiteindelijk het label één krijgen. Onderstaand voorbeeld geeft weer hoe dit in de praktijk werkt:

$$y_1 : [max(1)]$$
$$y_2 : [max(1, 1)]$$
$$y_3 : [max(1, 1, 1)]$$
$$y_4 : [max(1, 0)]$$
$$y_5 : [max(0)]$$

Dit geeft dus als eindresultaat het volgende, waarbij elk datapunt nu elk één label heeft:

$$Levelx : D\,[1, 1, 1, 1, 0]$$

Deze bewerking wordt uitgevoerd op elk level voor elke reeks van windows (benaderde en gedetailleerde coëfficiënt windows). Als eindresultaat krijgt men dus labels per datapunt voor elk level.

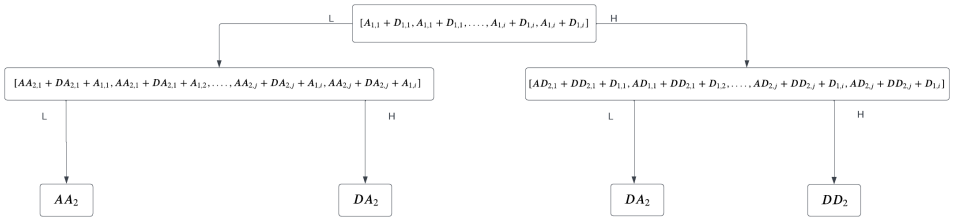
Verticale reductie

Zoals weergegeven in figuur 4.31 hebben we als eindresultaat labels per datapunten voor elke reeks van coëfficiënten met één dat aanduid op een anomalie en nul op een normaal datapunt. Deze labels worden allemaal gepropageerd naar de root node (de originele datapunten van de tijdreeks), dit heet verticale reductie omdat men labels vanuit meerdere levels (verticaal) reduceert tot de root node.

Zoals eerder aangegeven, wordt het aantal datapunten per level gehalveerd. Stel in het voorbeeld van figuur 4.31 komt een datapunt in level één twee keer voor in de originele tijdreeks (root node), dit voor zowel de gedetailleerde als de benaderde coëfficiënten. Met andere woorden zal komt een datapunt in level x overeen met twee datapunten in level $x-1$ waarbij x het aantal levels voorstelt en waarbij level nul de originele tijdreeks is.

Om verder te gaan met het voorbeeld op figuur 4.31, zullen de labels per level van beide coëfficiënten gecombineerd worden tot aan de root node (wat dus in dit geval maar één level is). Het resultaat hiervan kan afgeleid worden uit figuur 4.30. Om dit te verduidelijken, wordt in figuur 4.32 de werkwijze weergegeven met een meer conceptueel voorbeeld. DA zijn de gedetailleerde coëfficiënten van de benaderde coëfficiënten uit level één. AA zijn de benaderde coëfficiënten van de benaderde coëfficiënten uit level één. AD zijn de benaderde coëfficiënten van de gedetailleerde coëfficiënten uit level één.

DD zijn de gedetailleerde coëfficiënten van de gedetailleerde coëfficiënten uit level één. i is het aantal datapunten voor beide reeksen van coëfficiënten in level één en j het aantal datapunten voor beide reeksen van coëfficiënten in level twee. Omdat elke label in level twee tweemaal voorkomt in level één, tellen we twee keer de labels uit de gedetailleerde/benaderde coëfficiënten uit level twee samen met de benaderde of gedetailleerde coëfficienten uit level één met elkaar op. Dit wordt weergegeven in figuur 4.32, waarbij bijvoorbeeld $AA_{2,1} + DA_{2,1} + A_{1,1}$ twee keer wordt berekend.



Figuur 4.32: Voorbeeld van horizontale reductie met A de benaderde coëfficiënten en D de gedetailleerde.

Eindresultaat

Als eindresultaat hebben we voor elk datapunt een anomaliescore die aangeeft in hoeveel levels dit datapunt gelabeld wordt als anomalie. Hierop kan men een threshold zetten, deze threshold bepaalt in hoeveel levels een datapunt gelabeld moet worden als anomalie om in de originele tijdreeks daadwerkelijk gelabeld te worden als anomalie. Voor de threshold kan men bijvoorbeeld het gemiddelde van de scores berekenen maar merk op dat puntanomalieën eerder zichtbaar worden in de lagere levels dus de score voor puntanomalieën zal niet groot zijn.

4.4.7 Padding

Omdat de lengte van elke reeks coëfficienten bestaat uit de helft van het aantal punten uit de reeks van coëfficiënten van het vorige level zal het aantal levels gelijk zijn aan $\log_2(len(S))$ met $len(S)$ het aantal datapunten van de originele tijdreeks S. Als $\log_2(len(S))$ niet gelijk is aan een geheel getal, dan wordt er padding toegevoegd aan deze tijdreeks zodat $\log_2(len(S))$ een geheel getal is. De exacte hoeveelheid padding wordt als volgt berekend, waarbij $len(S)$ het aantal datapunten is van de originele tijdreeks S:

$$padding = len(S) - 2^{ceil(\log_2(len(S)))}$$

De opvulling die hier gebruikt wordt is het kopiëren van de laatste waarden. Omdat padding toegevoegd wordt zal men dus meer scores hebben dan effectieve datapunten in de originele tijdreeks. Dit wordt opgelost door de scores die buiten de tijdreeks vallen te verwijderen.

Hoofdstuk 5

Resultaten

In dit hoofdstuk wordt er besproken hoe men deze algoritmes, op een kwalitatieve manier, kunnen testen. Omdat de meeste tijdreeksen voornamelijk bestaan uit normale data, en er dus weinig anomalieën zijn (skewed datasets), is het belangrijk om de juiste metrieken te gebruiken. Het proces om deze resultaten te genereren gebeurt als volgt:

1. Opsplitsen van de tijdreeksdataset in een train- en testset.
2. Berekenen van alle parameters van elk model op de trainset.
3. Berekenen van anomalieën op de testset.
4. Berekenen van de testresultaten.

De eerste stap is het opsplitsen van de tijdreeks in twee aparte verzamelingen, met name de train- en de testset. De parameters (inclusief thresholds) van de modellen worden berekend aan de hand van de trainset. De uiteindelijke testresultaten worden berekend aan de hand van de testset. Een testset is hier van belang zodat het model getest wordt op ongeziene data. De trainset bestaat uit 60% van de originele tijdreeks terwijl de testset uit 40% van de originele tijdreeks bestaat.

5.1 Bepalen van de parameters

Zoals eerder vermeld worden de parameters van elk model berekend aan de hand van een trainset. De trainset bevat 60% van de originele tijdreeks.

5.1.1 DWT-MLEAD

De belangrijkste parameters die berekend worden op de traindataset zijn die van de multivariate Gaussian-distributies, deze worden gebruikt om binnen elke reeks van coëfficiënten sequenties van anomalieën te detecteren. De eerste stap is dus het bepalen van het start- en eind level op de dataset, als startlevel wordt er voor elke dataset het eerste level genomen. Het eindlevel is afhankelijk van de grootte van de dataset omdat de testdata kleiner is dan de traindata zal de testdata een lager maximaal level hebben dan de traindata. Daarom wordt het eindlevel berekend op het aantal datapunten in de testset. Belangrijk om te weten is dat de testdata zelf NIET gebruikt wordt om de parameters van de multivariate Gaussian-distributie te berekenen. De lengte van de testdata wordt hier enkel gebruikt zodat we bijvoorbeeld niet voor level negen in de trainset een multivariate Gaussian-distributie berekenen terwijl de testdata maar maximaal zes levels heeft.

Het eindlevel wordt als volgt berekend, waarbij *ceil* het resultaat naar boven afrond, *max_level* gelijk is aan $\log_2(n)$ waarbij n het aantal datapunten is:

$$end\_level = ceil((\frac{max\_level}{2}) + 2)$$

Daarna wordt er een decompositie van de trainset uitgevoerd met behulp van wavelet transformaties, zie sectie 4.4. Op basis van deze coëfficiënten kunnen de multivariate Guassian-distributies worden berekend die later gebruikt worden om anomalieën te detecteren. De "minimum anomaly count"wordt op twee gezet zoals in de paper vermeld staat.

5.1.2 ARIMA

In eerste instantie wordt er gecontroleerd of de tijdreeksdataset stationair is, zo niet wordt er eerst een differencing methode toegepast, die beschreven wordt in sectie 4.1. Daarna wordt de trainset gebruikt om de parameters van het ARIMA-model te bepalen. Deze parameters zijn: p, d en een threshold die bepaald of een datapunt een anomalie is of niet.

Zoals eerder vermeld is het mogelijk om de parameters p en q af te lezen van de ACF- en de PACF-grafiek maar dit is niet evident in een geautomatiseerd systeem. Daarom wordt er een grid search uitgevoerd. Bij een grid search worden verschillende modellen getraind met elk een andere combinatie van p en q, waarbij p en q waarden kunnen hebben van één tot vijf. Er worden dus 25 modellen getraind. Voor elk getraind model kan men dan de root mean squared error (RMSE) score berekenen. De parameters p en q van het model met de laagste RMSE-score op de testset worden gebruikt om de testresultaten te genereren.

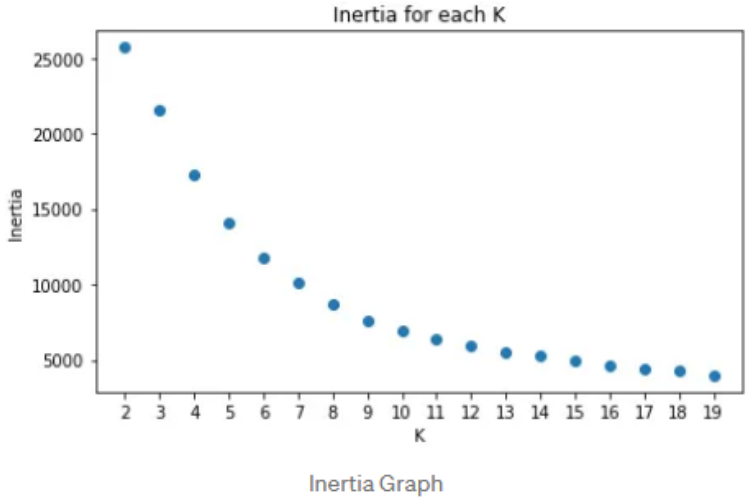
RMSE is het gemiddelde verschil tussen de voorspelde waarden en de originele tijdreeks en wordt als volgt berekend, waarbij N het aantal voorspellingen zijn, y de originele waarde en $\hat{y}$ de voorspelde waarde:

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

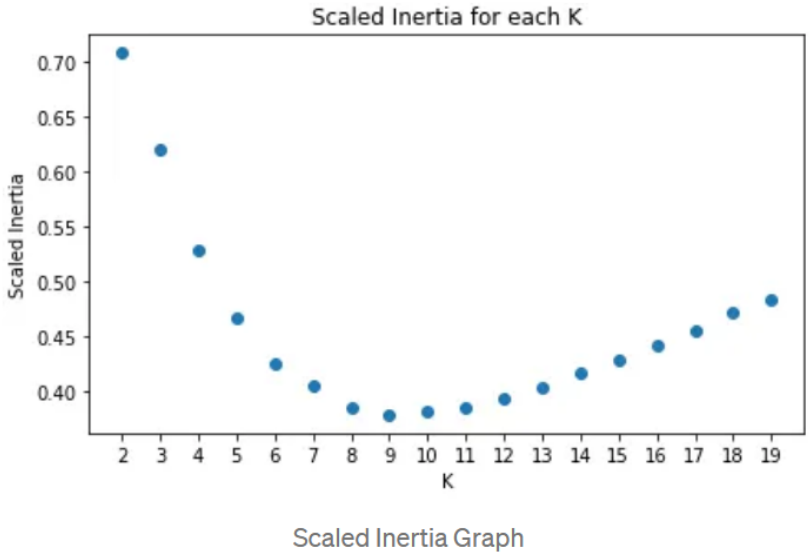
5.1.3 K-Means

Zoals bij de vorige modellen worden de parameters van K-Means ook berekend op de trainset. Als de originele dataset niet stationair is, wordt deze eerst stationair gemaakt zoals uitegelegd in sectie 4.1. Daarna wordt het optimaal aantal clusters gezocht voor deze trainset. Omdat het niet praktisch is om voor elke dataset handmatig naar de elbow-grafiek te kijken, wordt er een berekening gemaakt op basis van de gegeven *inertia*. Inertia is de kwadratische som van de verschillen tussen elk datapunt en hun bijbehorende cluster. Deze inertia kan dan geschaald worden zodat we op een duidelijke manier kunnen aflezen wat het optimaal aantal clusters zijn voor de trainset. Het berekenen van de inertia gebeurt per hoeveelheid van clusters. Er wordt telkens getraind met een bepaald aantal clusters gaande van 1 tot 20 waarbij het model in de eerste iteratie één cluster heeft en bij de 20ste iteratie 20 clusters. De geschaalde inertia per cluster kan dan als volgt berekend worden, waarbij K het aantal clusters is per iteratie en α een error parameter is [30]:

$$scaled\_inertia = \frac{inertia_K}{intertia_1} + \alpha \times K$$



Figuur 5.1: Elbow-grafiek, waarbij de x-as het aantal clusters zijn per iteratie en de y-as de inertia [30].



Figuur 5.2: Grafiek van de geschaalde inertia per hoeveelheid clusters, waarbij de x-as het aantal clusters zijn per iteratie en de y-as de geschaalde inertia [30].

Uit de elbow-grafiek op Figuur 5.1 kan men afleiden dat het optimaal aantal clusters tussen de 7 en 9 ligt. Het bepalen van het optimaal aantal clusters met behulp van deze grafiek is niet evident in tegenstelling tot de geschaalde inertia op figuur 5.2. Bij de reeks van geschaalde inertia's is het optimaal aantal clusters, het punt waar de geschaalde inertia niet meer gaat dalen, wat in dit geval dus is bij 9 clusters.

5.2 Metrieken

5.2.1 TP, FP, FN, TN

Allereerst wordt het aantal true positives (TP), false positives (FP), false negatives (FN) en het aantal true negatives (TN) berekend. True positives zijn datapunten die we detecteren als een anomalie en ook een anomalie zijn in de originele tijdreeks. False positives zijn diegene die het model detecteert als anomalie maar in werkelijkheid geen zijn. False negatives zijn datapunten die niet gedetecteerd worden als een anomalie maar in werkelijkheid wel één zijn. Deze voorspellingen hebben in het productieproces de grootste kost. Stel we hebben een productieproces waarbij elk onderdeel in het proces gemonitord wordt met behulp van verschillende sensoren. Wanneer een werkelijke anomalie niet wordt gedetecteerd, loopt het productieproces gewoon door wat grote gevolgen kan hebben op het eindproduct. Als laatste zijn er de true negatives, deze zijn datapunten die niet gedetecteerd worden als een anomalie en in werkelijkheid ook geen zijn. Met deze waarden kan men dan een confusion matrix opstellen zoals weergegeven in figuur 5.3. Een confusion matrix is een matrix met in de cellen het aantal true positives, false positives, false negatives en het aantal true negatives.

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Figuur 5.3: Confusion matrix [31].

5.2.2 Precision & recall

Met behulp van deze waarden kan men de precision en de recall score berekenen. Deze kunnen dan ook gecombineerd worden om een kwalitatief testresultaat te bekomen.

Precision

Precision berekent hoeveel van de positieve predicties (TP en TN) correct voorspeld zijn. Dit houdt dus in hoeveel procent van de positieve voorspellingen correct zijn. Precision kan als volgt berekend worden, waarbij TP true positives zijn en FP de false positives zijn [32]:

$$precision = \frac{TP}{TP + FP}$$

Omdat precision berekend wordt aan de hand van de TP en FP zal het geen impact hebben als er weinig anomalieën zijn. Precision wordt namelijk berekend op hoeveel anomalieën er net juist gedetecteerd zijn. Stel een tijdreeks met 1000 normale datapunten waarvan 10 datapunten die gelabeld zijn als anomalie. Van deze 10 anomalieën is er maar 1 anomalie juist gedetecteerd en van de normale datapunten zijn er 2 gedetecteerd als FP, dan zal de precision er als volgt uitzien [33]:

$$precision = \frac{1}{1 + 2}$$

Het aantal TN ten opzichte van het aantal TP zal geen impact hebben. Deze berekend namelijk hoeveel van de gedetecteerde anomalieën daadwerkelijk ook anomalie zijn. Het is hier belangrijk om het aantal FP te minimaliseren. Een ander probleem onstaat wanneer er maar bijvoorbeeld maar één of twee anomalieën in de tijdreeks zitten en het model bijvoorbeeld 5 FP genereert. Ondanks het model wel de correcte anomalieën vindt zal de precision naar de lage kant zijn.

Recall

Recall berekend hoeveel van de positieve predicties daadwerkelijk correct geclassificeerd zijn, over alle positieve cases. Er worde de verhouding berekend tussen het aantal TP en het totaal aantal anomalieën van de originele testset. De formule van recall wordt hieronder weergegeven, waarbij TP de true positives en FN de flase negatives zijn [32]:

$$recall = \frac{TP}{TP + FN}$$

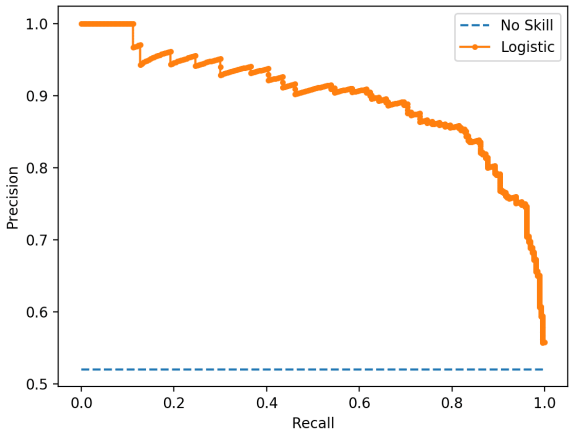
Omdat recall de verhouding detecteerd tussen de TP en de FN hebben skewed klassen geen impact op deze score. Stel een tijdreeks met 1000 datapunten waarvan 10 datapunten gelabeld zijn als anomalie. Van deze 10 anomalieën is er maar 1 anomalie juist gedetecteerd en vijf normale datapunten gelabeld zijn als anomalie (FN), dan zal de recall score gelijk zijn aan:

$$recall = \frac{1}{1 + 10}$$

99% van de tijdreeks bevat normale data en maar 1% van deze data is anomalie. Toch zal dit geen impact hebben op de recall score omdat deze het aantal TN niet in rekening brengt [33]. Wat bijvoorbeeld wel een impact heeft is dat als de tijdreeks bijvoorbeeld twee anomalieën bevat waarvan maar één gedetecteerd wordt, zal de recall score maar 50% bedragen [32].

Precision en recall curve

Omdat precision geen rekening houdt met het aantal FN en recall geen rekening houdt met het aantal FP, is het belangrijk om deze twee scores samen te evalueren. Dit kan men doen aan de hande van een precision-recall curve, zoals weergegeven in figuur 5.4. De precision en recall curve toont de tradeoff tussen precision en recall. Een hoge oppervlakte onder de curve betekend dat men een hoge recall en precision heeft. Er wordt een tradeoff gemaakt tussen precision en recall als de precision laag is maar de recall hoog of beide laag zijn zal de oppervlakte onder deze grafiek minimaal zijn wat insinueert dat het model niet goed zal classificeren. één punt zal dus overeenkomen met een instantie van een model met één bepaalde threshold. Door verschillende parameters te laten variëren krijgt men een precision/recall-curve [34].



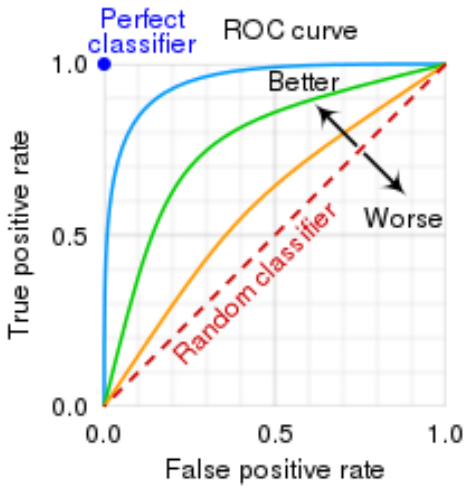
Figuur 5.4: De oranje lijn is de tradeoff tussen precision en recall terwijl de blauwe lijn een random classifier voorstelt [34].

5.2.3 ROC

Receiving operator characteristics is een grafiek met op de x-as de false positive rate (FPR) en de y-as de true positive rate (TPR). Hierbij wordt elke TPR en FPR berekend op verschillende parameters van het model, zoals bij de precision-recall curve. De TPR is gelijk aan de recall. De FPR wordt als volgt berekend:

$$FPR = \frac{FP}{FP + TN}$$

Figuur 5.5 is een voorbeeld van een ROC-curve voor een binaire classificatie. Een perfect classificatiemodel zou een TPR van één en een FPR van nul aanhouden, aangezien we zoveel mogelijk correcte voorspellingen willen en zo weinig mogelijk fout gedetecteerde voorspellingen. De stippellijn op grafiek 5.5 duidt een random model aan dat random classificaties maakt. Hoe meer een grafiek bij de y-as ligt hoe beter het model is. De blauwe grafiek op figuur 5.5 ligt dichter bij de y-as. Het model dat de blauwe grafiek voorstelt heeft een lage FPR en een hoge TPR en heeft een beter resultaat dan het model van de groene grafiek.



Figuur 5.5: ROC-curve voor een binaire classificatie [35].

Zoals eerder besproken hebben skewed tijdreeksen geen invloed op de recall-score, omdat men hier enkel de positieve klassen evalueert en niet de negatieve (normale datapunten). Bij de false positive rate is dit wel een probleem. Het is namelijk mogelijk om alles te labelen als een normaal datapunt waardoor men een lage FPR uitkomt. Stel een tijdreeks met 1000 datapunten waarvan elk datapunt gelabeld wordt als normaal, dan hebben we een FPR van nul zoals hieronder wordt weergegeven:

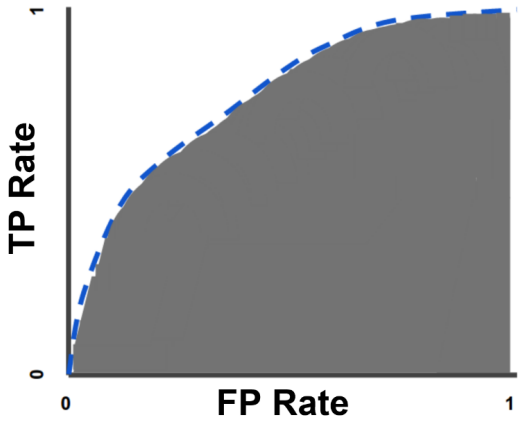
$$FPR = \frac{0}{0 + 1000}$$

Maar omdat er de tradeoff gemaakt wordt tussen TPR (recall) en FPR kan de score niet gemanipuleerd worden door heel de tijdreeks als normaal te labelen. Stel we nemen het vorig voorbeeld, waarbij elk datapunt gelabeld is als normaal dan zal er dus een lage FPR maar ook een lage TPR zijn. Door deze twee te combineren kan men de nauwkeurigheid van het algoritme afleiden.

5.2.4 AUC

Zoals eerder vermeld kan men de nauwkeurigheid van een algoritme bepalen aan de hand van de oppervlakte onder de precision-recall- en de ROC-curve. Om deze oppervlakte om te zetten in een score wordt Area Under Curve (AUC) gebruikt. Een AUC van één wilt zeggen dat bij de precision-recall-curve de precision altijd één is tot aan het einde van de x-as. Bij de ROC-curve is dit wanneer de TPR altijd één zal zijn en de FPR altijd nul. Bij deze twee voorbeelden zal de oppervlakte onder de curve het grootste zijn. Figuur 5.6 is een voorbeeld van een ROC-curve, waarbij het grijze gedeelte de prestaties van het model aanduidt. Dit gedeelte willen we maximaliseren.

Voor AUC-ROC scores, zullen scores tussen 0 en 0.50 refereren naar een random classificatie-model. Dit insinueert dat het model slechter of gelijk scoort aan een random model. Tussen 0.50 en 0.70 refereert naar een model dat datapunten slecht discrimineert en dus moeilijker onderscheid kan maken tussen anomalieën en normale data. Tussen de 0.70 en de 0.80 heeft het model een acceptabele discriminatie, tussen de 0.8 en 0.9 heeft het model een goede discriminatie en met alles boven de 90 heeft het model een excellente discriminatie. Voor de ROC-PR zal een excellent algoritme een AUC-score hebben van één wat terug aanduidt dat het vlak onder de grafiek maximaal is [36].



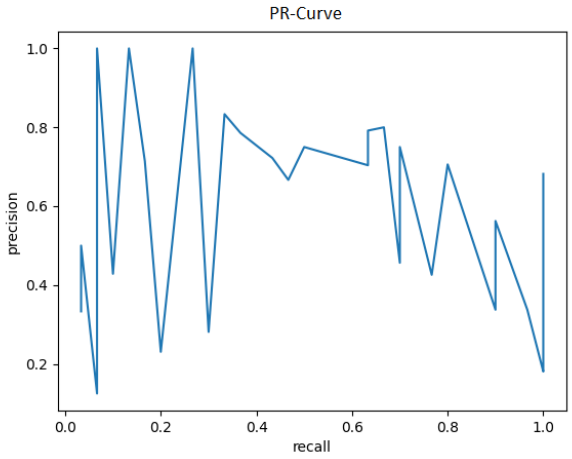
Figuur 5.6: ROC-curve met AUC voor een binaire classificatie [37].

5.3 Overzicht van de resultaten

Om testresultaten te creëren laat men elke parameter van elk model variëren. Als resultaat heeft men meerdere instanties van één soort algoritme waarbij elke instantie getest wordt. Daarvan worden de AUC-ROC en de AUC-PR scores berekend. Dit geeft een overzicht van de prestaties van een algoritme. Om dit te verduidelijken nemen we volgend voorbeeld, waarbij D d verzameling van alle instanties voor het DWT-MLEAD algoritme voorstelt, d een instantie is van het DWT-MLEAD en waarbij elke instantie verschillend is van elkaar. A is de verzameling van instanties voor het ARIMA-model waarbij elk instantie aangeduid is met a en verschillend is van elkaar. K is de verzameling van instanties voor het K-Means model waarbij elk instanties aangeduid is met k en verschillend is van elkaar. n is het aantal instanties voor DWT-MLEAD, m is het aantal instanties voor ARIMA en j is het aantal instanties voor K-Means:

$$D[d_1, d_2, ...d_n], A[a_1, a_2, ..., a_m], K[k_1, k_2, ..., k_j]$$

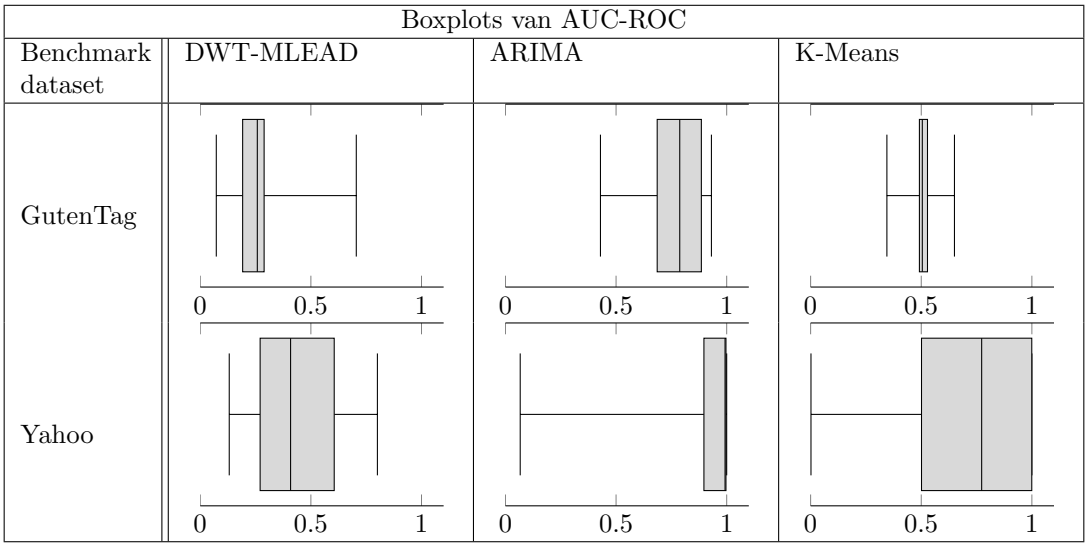
Elke instantie heeft dus andere parameters. Een parameter kan bijvoorbeeld het aantal clusters zijn voor het K-Means algoritme. Hierdoor testen we verschillende instanties met elk andere parameters. Voor elke instantie wordt dan de threshold apart gevarieerd van zijn minimum tot maximum waarde. Deze threshold bepaald of een datapunt een anomalie is. Bijvoorbeeld bij ARIMA is de threshold het verschil tussen het originele datapunt en het voorspelde. Omdat testresultaten gemaakt zijn op meerdere instanties van een model, worden de metingen gesorteerd. Voor de AUC-ROC-score worden de metingen gesorteerd op FPR en TPR en voor AUC-PR worden ze gesorteerd op precision en recall. Figuur 5.7 geeft het resultaat weer als ze niet gesorteerd zijn. Het sorteren zorgt ervoor dat men de correcte curve bekomt zoals eerder vermeld in figuur 5.4.



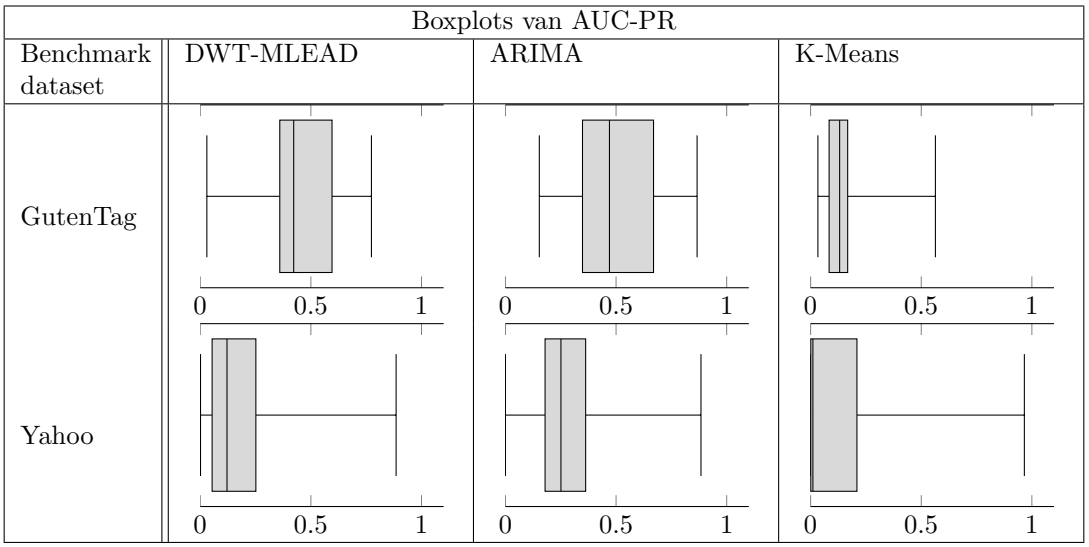
Figuur 5.7: PR-curve waarbij precision en recall niet gesorteerd is.

Met de GutenTag-tijdreeks generator zijn er ongeveer 30 tijdreeksen gegenereerd. Elk van de dataset heeft gevarieerde anomalieën zoals: verandering in de frequentie van een sequentie, verandering van het gemiddelde van een sequentie, verandering van de variantie van een sequentie, lokale puntanomalieën (deze kunnen kleiner of groter zijn dan de vorige punten) en globale puntanomalieën. Zowel de trainset als de testset kan anomalieën bevatten zodat de algoritmes in een unsupervised omgeving getest worden.

Als tweede verzameling van datasets wordt er de Yahoo-benchmark dataset gebruikt. Hiervan worden een aantal tijdreeksen datasets (CSV-bestanden) geselecteerd zodat we een ruime diversiteit aan anomalieën hebben. Uit de gehele set zijn er 50 tijdreeksen uitgekozen die verschillende soorten anomalieën bevatten.



Tabel 5.1: Boxplots van de AUC-scores van de ROC-curve, gemeten op elke tijdreeks van de benchmark datasets.



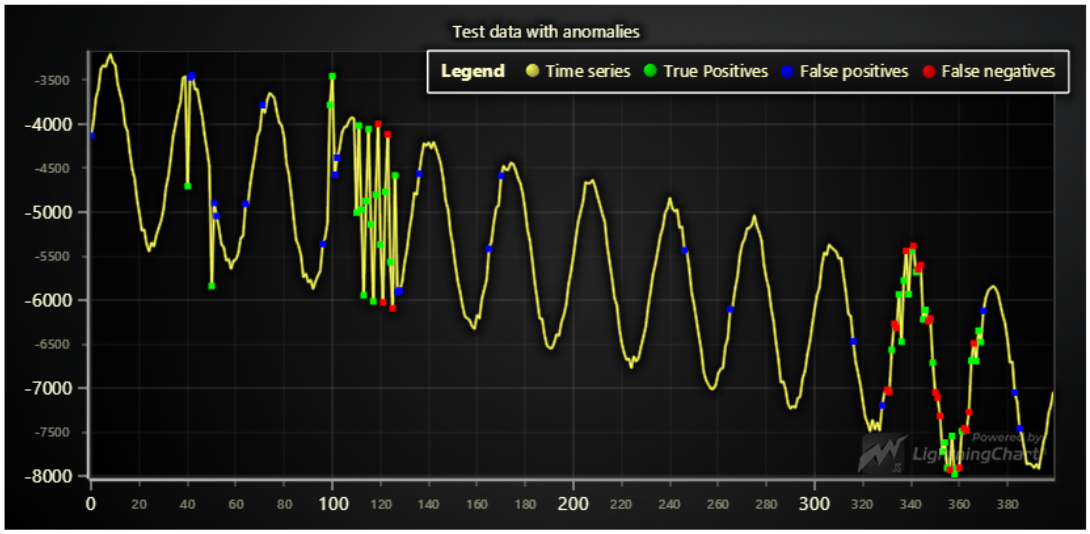
Tabel 5.2: Boxplots van de AUC-scores van de PR-curve, gemeten op elke tijdreeks van de benchmark datasets.

	DWT-MLEAD		ARIMA		K-Means	
	AUC -PR	AUC -ROC	AUC -PR	AUC-ROC	AUC -PR	AUC-ROC
Gut1(f)	49%	16%	86%	88%	23%	47%
Gut2(f)	77%	27%	36%	78%	6%	50%
Gut4(f)	75%	27%	53%	74%	8%	49%
Gut5(f,p)	73%	21%	76%	87%	9%	49%
Gut6(m)	45%	19%	46%	42%	56%	54%
Gut7(v)	59%	24%	43%	66%	8%	47%
Gut8(f,m)	38%	30%	28%	68%	4%	46%
Gut9(f,m,t)	35%	18%	31%	80%	5%	62%
Gut10(f,m,t)	26%	24%	31%	80%	5%	65%
Gut11(m)	60%	70%	17%	51%	16%	50%
Gut12(m,p)	59%	32%	15%	49%	26%	57%
Gut13(p)	2%	25%	34%	66%	14%	59%
Gut14(p,t)	3%	7%	34%	89%	7%	51%
Gut16(f,v,o,t)	41%	26%	56%	81%	8%	51%
Gut17(f,v,o,t)	35%	14%	74%	89%	13%	39%
Gut18(f,v,o)	30%	15%	47%	72%	15%	50%
Gut19(f,v,o,t)	41%	28%	66%	89%	13%	52%
Gut20(f,v,o)	47%	21%	43%	64%	17%	49%
Gut21(f,v,o)	36%	16%	49%	70%	18%	49%
Gut22(f)	54%	30%	45%	76%	11%	50%
Gut23(f)	66%	56%	79%	91%	16%	51%
Gut24(p,v,f,t)	42%	28%	55%	80%	27%	62%
Gut25(p,v,f,t)	15%	27%	47%	89%	3%	34%
Gut26(v,t)	42%	25%	84%	93%	10%	45%
Gut27(v)	70%	49%	66%	78%	14%	52%
A3-TS1(p,t)	2%	29%	48%	99%	0%	32%
A3-TS2(p,t)	4%	17%	33%	98%	6%	32%
A3-TS7(p)	10%	64%	44%	99%	33%	97%
A3-TS12(p)	7%	59%	38%	99%	42%	85%
A3-TS15(p,t)	11%	52%	22%	99%	1%	58%
A3-TS21(p,t)	3%	25%	54%	98%	8%	48%
A3-TS25(p,t)	12%	26%	16%	98%	0%	38%
A3-TS28(p,t)	0%	52%	0%	99%	0%	51%
A3-TS29(p,t)	10%	15%	17%	99%	0%	33%
A3-TS31(p)	7%	66%	21%	99%	17%	70%
A3-TS35(p,t)	0%	23%	0%	99%	0%	7%
A3-TS46(p,t)	12%	32%	14%	99%	2%	63%
A3-TS47(p,t)	8%	27%	32%	99%	18%	61%
A3-TS49(p)	9%	50%	54%	99%	34%	62%
A3-TS79(p,t)	2%	25%	20%	99%	0%	40%
A3-TS94(p,t)	12%	31%	20%	99%	0%	3%
A3-TS95(p,t)	3%	13%	52%	98%	0%	49%
A3-TS100(p,t)	12%	73%	36%	99%	33%	93%
A4-TS2(p,t)	12%	27%	16%	99%	12%	79%
A4-TS26(p,m)	12%	31%	18%	99%	25%	67%
A4-TS52(p)	8%	41%	59%	99%	25%	78%
A4-TS65(p)	11%	61%	36%	99%	38%	75%
A4-TS71(p,t)	9%	36%	39%	92%	19%	62%
A4-TS89(p,t)	6%	55%	23%	98%	2%	52%
A4-TS97(p,t)	4%	26%	36%	98%	2%	76%

real17(m,v)	54%	14%	82%	89%	81%	85%
real25(m,v)	88%	27%	88%	44%	96%	100%
real37(v,p)	34%	38%	18%	81%	21%	80%
real38(m,p)	32%	80%	22%	84%	26%	69%
real46(m)	23%	13%	7%	6%	9%	0%
real51(p)	11%	37%	29%	99%	0%	13%
real58(f)	88%	15%	88%	58%	82%	100%
real66(f,v)	49%	27%	85%	90%	84%	94%
Synth10(p)	25%	18%	25%	50%	0%	48%
Synth13(p,t)	25%	61%	25%	72%	0%	100%
Synth18(p,t)	0%	35%	0%	99%	0%	99%
Synth2(p,t)	16%	78%	16%	33%	0%	37%
Synth25(p,t)	25%	42%	25%	99%	0%	100%
Synth28(p,t)	25%	62%	25%	99%	0%	100%
Synth30(p,t)	0%	57%	0%	100%	0%	100%
Synth34(p,t)	25%	40%	25%	80%	0%	100%
Synth40(p,t)	25%	60%	25%	74%	0%	91%
Synth42(p,t)	0%	63%	0%	100%	0%	100%
Synth6(p,t)	0%	64%	0%	99%	0%	97%
Synth85(p,t)	12%	51%	25%	99%	0%	100%
Synth89(p,t)	16%	77%	22%	66%	0%	100%
Synth9(p,t)	0%	60%	0%	99%	0%	100%
Synth94(p,t)	25%	54%	25%	99%	5%	98%
Synth98(p,t)	25%	62%	27%	68%	0%	100%
Synth100(p)	25%	58%	25%	99%	0%	100%

Tabel 5.3: Elke dataset apart geëvalueerd met p puntanomalieën, f frequentie-anomelieën, v variantieanomalieën, m sequentieanomalieën met een verschillend gemiddelde, t duidt aan dat een tijdreeks trending bevat.

Het eerste wat opvalt is, ondanks ARIMA bedoeld is om puntanomalieën te detecteren, er toch een redelijke nauwkeurigheid gehaald op sequentieanomalieën. ARIMA presteert voornamelijk goed op sequentieanomalieën met een hogere variantie of frequentie. Figuur 5.8 is een tijdreeks met drie soorten anomalieën: puntanomalieën, sequentieanomalie met een verschillende frequentie en een sequentieanomalie met een verschillende variantie. Puntanomalieën worden, zoals verwacht, gedetecteerd door het ARIMA-model. Bij sequentieanomalieën vallen datapunten dichter opelkaar en zullen plotse stijgingen zichtbaar worden waardoor er een groter verschil ontstaat tussen het voorspelde punt en het echte punt. Een sequentieanomalie met een hogere variantie zal hier (deels) gedetecteerd worden, omdat een variantie noise simuleert waardoor datapunten iets hoger of lager liggen. Deze plotse stijgingen en dalingen worden door ARIMA gedetecteerd. ARIMA heeft echter geen notie hoe abnormaal een hele sequentie is, het bepaalt per punt of dit een anomalie is of niet. Daarom zullen zulke sequenties niet in zijn geheel een anomalie vormen. Omdat ARIMA punt per punt analyseert zal deze minder FP genereren.

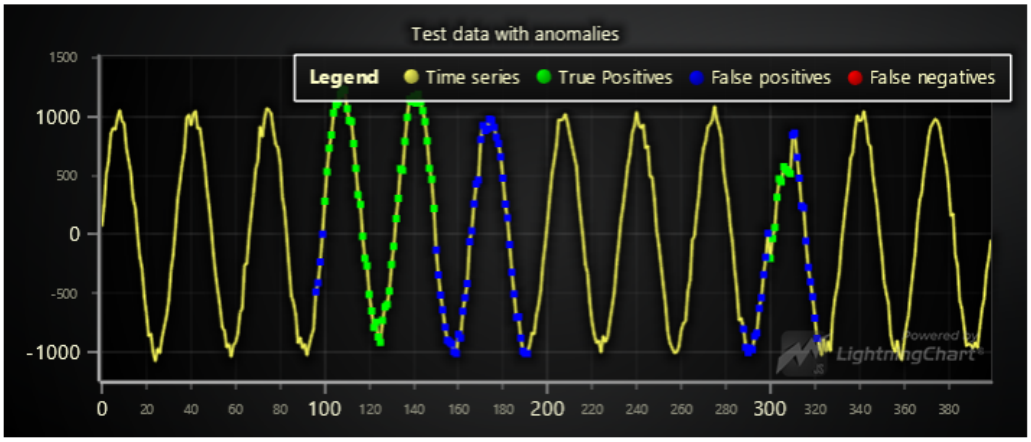


Figuur 5.8: Resultaat van ARIMA op sequentieanomalieën met een hoge frequentie en een hoge variantie. De groene punten zijn de TP, de blauwe punten zijn de FP en de rode punten zijn de FN.

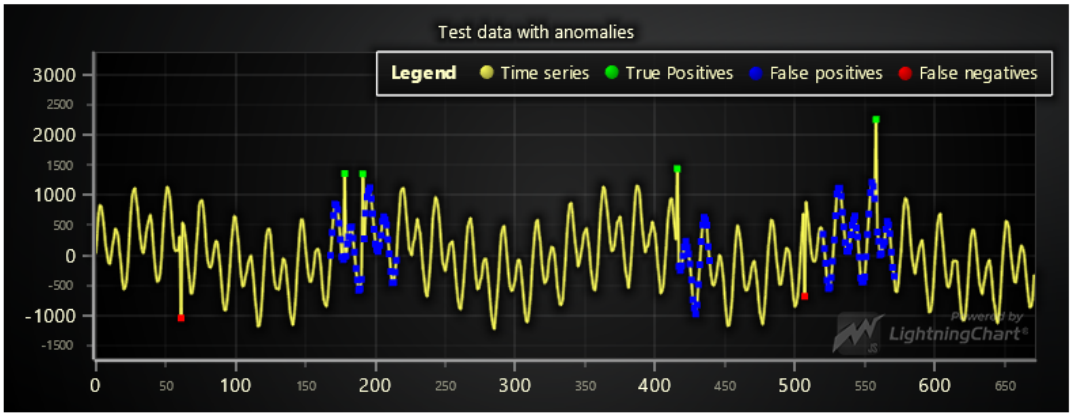
Op eerste zicht lijken de AUC-PR scores, van tabel 5.2 beter voor de GutenTag-benchmark dataset bij het ARIMA-model wat zou impliceren dat ARIMA beter werkt op sequentieanomalieën. Omdat meerdere instanties van het ARIMA-model zijn getest, en dus niet meteen op de optimale p en q waardes zal bij de meeste instanties het model de data under- of overfitten waardoor de voorspelling niet accuraat zijn. Dit heeft als resultaat dat er meer FP en FN zijn wat dus een grotere impact heeft op de precision en recall. Dit is ook voornamelijk omdat de tijdreeksen uit de Yahoo-dataset minder anomalieën bevatten en hoofdzakelijk bestaan uit puntanomalieën t.o.v. de GutenTag-benchmark dataset die sequentieanomalieën bevat.

DWT-MLEAD scoort, volgens de metingen in tabel 5.2 en 5.1, zeer laag. Als men kijkt naar de boxplot voor de precision-recall auc-scores valt er op dat de mediaan zich rond de 45% bevindt, wat niet goed is. De reden hiertoe is dat DWT-MLEAD behoorlijk wat FP detecteert, zoals figuur 5.9 aantoont. De reden hiervan is dat sommige delen van de sequentie herkend worden in latere levels. Omdat DWT-MLEAD sommige delen in latere levels detecteert zal dit een groter domein van punten aanspreken in de originele tijdreeks waardoor er meer FP zijn. Daarnaast is het voor DWT-MLEAD niet eenvoudig om de correcte parameters te voorspellen zoals de scores aangeven. De meeste instanties van het DWT-MLEAD algoritme presteren ondermaats. Wat opvalt is dat er een behoorlijke score wordt gehaald op de Yahoo-benchmark dataset. Zo zal DWT-MLEAD toch in staat zijn om, voornamelijk globale puntanomalieën,

te detecteren. Figuur 5.10 is een tijdreeks met verschillende globale puntanomalieën. DWT-MLEAD zal dan ook heel het gebied rond dit punt beschouwen als anomalie en zal een hele sequentie detecteren als FP. De reden waarom de AUC-ROC-score hoger is voor DWT-MLEAD op de Yahoo-benchmark dataset is omwille dat het aantal FP veel minder effect als men naar de formule van FPR kijkt. Omdat DWT-MLEAD instaat is om puntanomalieën te detecteren zoals aangegeven in figuur 5.10, zal de AUC-ROC-score bij sommige datasets hoger liggen omdat men hier de meeste anomalieën vindt terwijl bij sommige GutenTag-tijdreeksen niet altijd heel de sequentie gevonden wordt. Dit is omdat elke instantie van het model getest wordt over een verschillende range van levels. Niet elk detail is in elk level zichtbaar. Er zullen maar enkele instanties goed presteren op de testresultaten.

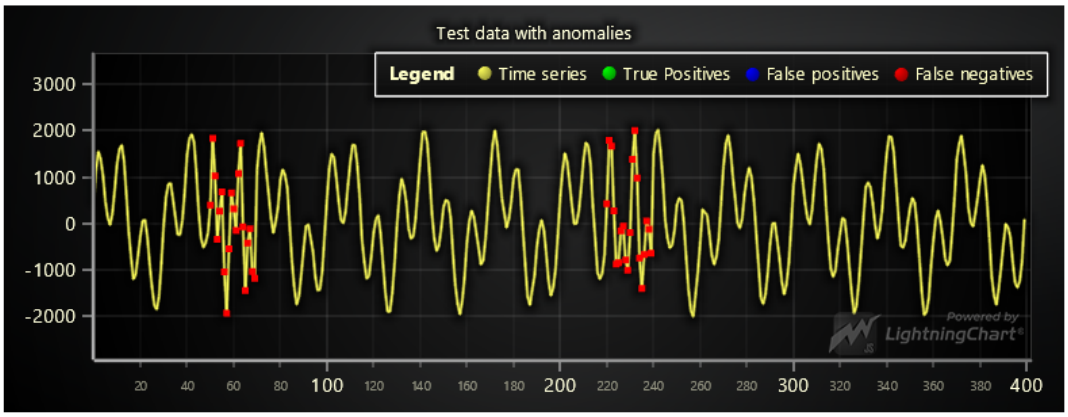


Figuur 5.9: Resultaat op de Gut-12-tijdreeks. Deze tijdreeks bevat sequentieanomalieën. De groene punten zijn de TP, de blauwe punten zijn de FP en de rode punten zijn de FN.

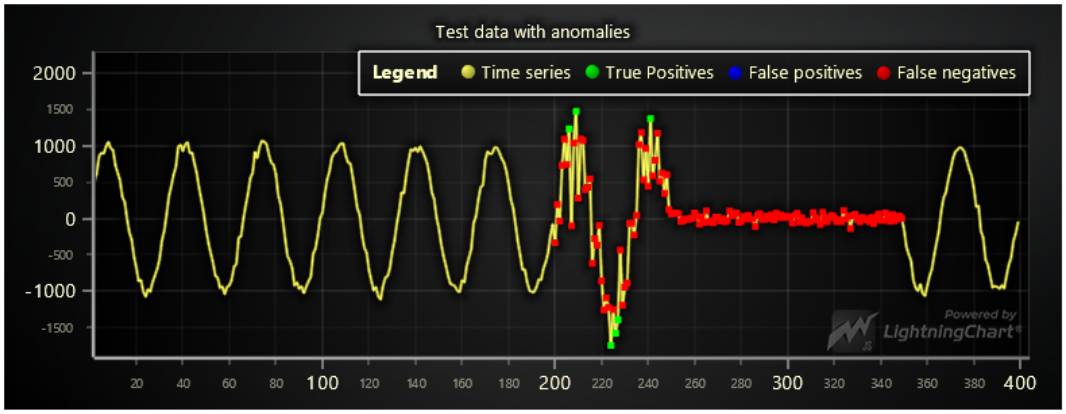


Figuur 5.10: Resultaat van DWT-MLEAD op de A3-TS12, waarbij DWT-MLEAD heel het gebied rond de anomalie mee detecteert wat resulteert in een hoge FPR. De groene punten zijn de TP, de blauwe punten zijn de FP en de rode punten zijn de FN.

We merken op dat K-Means voornamelijk enkel globale puntanomalieën detecteert. De tijdreeks op figuur 5.11 toont aan dat K-Means geen enkel van de gevonden sequentieanomalieën vindt. Dit is omdat de implementatie van K-Means enkel een anomalie detecteert als deze een waarde heeft die hoger is dan een bovengrens, lager is dan een ondergrens (per cluster) of als de cluster minder dan bijvoorbeeld 1% van de data bevat. In dit geval wordt de cluster in zijn geheel gelabeld als anomalie. Uit Figuur 5.12 leidt men af dat alleen de meest onderste en de meest bovenste anomalieën ook gedetecteerd worden als anomalie. Al de andere datapunten van deze sequentieanomalieën zullen niet gedetecteerd worden door deze K-Means implementatie.

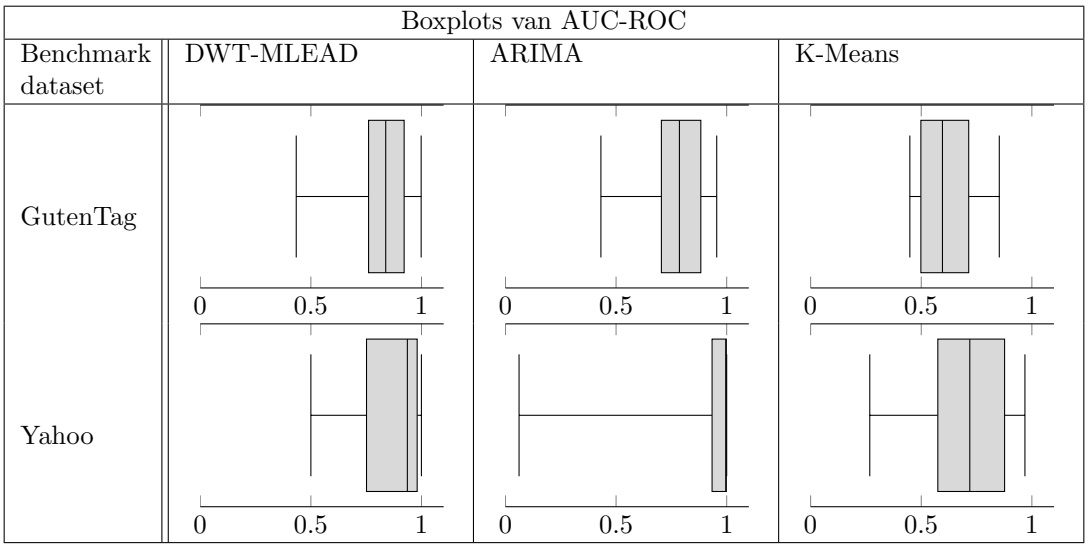


Figuur 5.11: Resultaat van K-Means op de Gut-22. De groene punten zijn de TP, de blauwe punten zijn de FP en de rode punten zijn de FN.

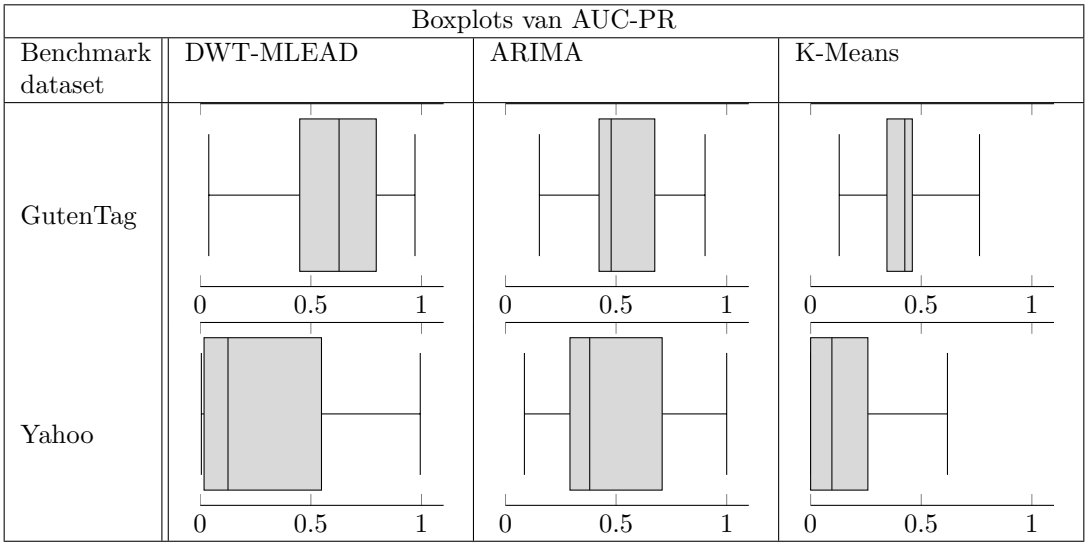


Figuur 5.12: Resultaat van K-Means op de Gut-22. De groene punten zijn de TP, de blauwe punten zijn de FP en de rode punten zijn de FN.

Tabellen 5.4 en 5.5 zijn de AUC-ROC- en AUC-PR-scores voor elke tijdreeks van een benchmark dataset maar met nu enkel de threshold die gevarieërd wordt. Er wordt nu maar één enkele instantie van het model getest. Hierbij worden de parameters gekozen zoals weergegeven in sectie 5.1. In deze testresultaten wordt enkel de threshold gevarieërd en niet al de andere parameters van de modellen. We kunnen hier duidelijk zien dat de scores nu beter zijn voor DWT-MLEAD en ARIMA. Dit is omdat andere instanties bij ARIMA de dataset niet goed fitten, of bij DWT-MLEAD de anomalieën in een gegeven range van levels niet vinden waardoor de meeste voorspellingen voor de andere instanties niet goed zijn.



Tabel 5.4: Boxplots van AUC-ROC-scores voor elk model waar enkel de threshold varieert. Elke tijdreeks heeft een AUC-ROC-score, deze worden samengenomen om een boxplot te genereren.



Tabel 5.5: Boxplots van AUC-PR-scores voor elk model waar enkel de threshold varieert. Elke tijdreeks heeft een AUC-PR-score, deze worden samengenomen om een boxplot te genereren.

5.4 Conclusie van de resultaten

Het detecteren van anomalieën is een uitdagende taak waarbij geen enkel algoritme in staat is om elk type anomalie te detecteren. Zoals eerder vermeld, is het in de praktijk vaak noodzakelijk om een combinatie van verschillende algoritmes te gebruiken om zoveel mogelijk anomalieën op te sporen. Het specifieke type data en het soort anomalie dat men wil detecteren, zijn de belangrijkste factoren bij het bepalen van het meest geschikte algoritme, wat we ook terugzien in de resultaten van vorige paragraaf.

In het geval van de anomaliedetectie van tijdreeksen zou een combinatie van DWT-MLEAD en ARIMA een goede start kunnen zijn om verschillende soorten anomalieën te detecteren. Uit de testresultaten blijkt dat DWT-MLEAD beter in staat is om sequenties van anomalieën te detecteren in vergelijking met ARIMA, zoals we kunnen zien in tabel 5.4 en 5.5. Aan de andere kant is ARIMA meer geschikt voor het detecteren van puntanomalieën. Daarom is het raadzaam dat bedrijven zoals TrendMiner een combinatie van verschillende anomaliedetectie technieken gebruiken om zo veel mogelijk anomalieën op te sporen.

Een belangrijk aspect bij het evalueren van de resultaten van de anomaliedetectie is de vergelijking van voorspellingen op puntniveau met de oorspronkelijke labels van de tijdreeks. Het is ook vermeldenswaardig dat algoritmes een hele sequentie als anomalie kunnen detecteren als bijvoorbeeld één punt van de sequentie als anomalie wordt gekenmerkt, of als de meerderheid van de sequentie correct wordt gedetecteerd. Het is mogelijk dat punten rondom een geïdentificeerde anomalie ook afwijken van de overige observaties, waardoor de context en het patroon van de anomalie beter begrepen kunnen worden.

Het DWT-MLEAD algoritme vertoont echter een aanzienlijk aantal FP wanneer het start- en eindlevel niet optimaal gekozen is. Dit betekent dat het moeilijker is om de juiste parameters voor dit algoritme te bepalen. Het correct instellen van deze levels is belangrijk om ervoor te zorgen dat de relevante coëfficiënten op de juiste manier worden geanalyseerd. Aan de andere kant genereert ARIMA minder FP omdat het puntsgewijze voorspellingen maakt in plaats van sequenties van data te analyseren.

Het selecteren van de optimale parameters en levels voor anomaliedetectie algoritmes is een belangrijke overweging en vereist vaak een iteratief proces van experimentatie en optimalisatie. Het is cruciaal om de specificaties en vereisten van het specifieke anomaliedetectieprobleem in overweging te nemen, evenals de kenmerken van de beschikbare tijdreeksen. Door een combinatie van verschillende algoritmes te gebruiken, kunnen bedrijven de robuustheid en nauwkeurigheid van hun anomaliedetectiesysteem vergroten, waardoor ze effectief kunnen reageren op verschillende anomalieën in hun gegevens.

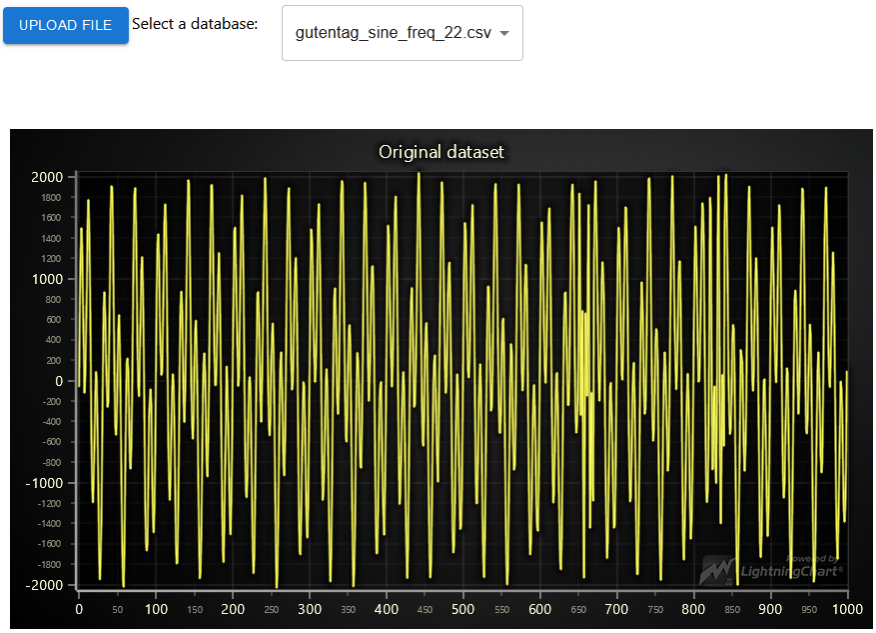
Hoofdstuk 6

Exploratietool

Om beter te begrijpen waarom datapunten gedetecteerd worden als anomalie, is er een interactieve webapplicatie gemaakt die voor elk van deze drie modellen visualiseert waarom datapunten gelabeld worden als anomalie. Om de applicatie te gebruiken is het van belang dat men weet wat de parameters van elk van deze modellen zijn en wat hun invloed is op het detecteren van anomalieën.

De webapplicatie bestaat uit een frontend en een backend. De frontend is een React-webapplicatie terwijl de backend een Python Flask-server is. De algoritmes worden geïmplementeerd in de Python-backend. De algoritmes worden interactief getoond in de React-frontend.

Zoals aangegeven op figuur 6.1 is er een dropdown menu waar de gebruiker de mogelijkheid heeft om een reeks van tijdreeksen te kiezen die al op de server staan. Het is mogelijk om een eigen tijdreeks toe te voegen en deze te uploaden naar de webserver via de "UPLOAD FILE"knop. Eenmaal de tijdreeks geanalyseerd is wordt deze gevisualiseerd op het scherm. Daarna is het mogelijk om elk van de algoritmes apart te analyseren door ze aan te duiden in het tab menu onderaan de grafiek.



Figuur 6.1: Selecteren van dataset en visualisatie van de dataset.

6.1 Metrieken voor individuele instanties

In hoofdstuk 5 zijn families van algoritmes getest. Voor elke instantie per soort algoritme, zijn er testresultaten gemaakt. Hiervoor is gebruikgemaakt van de PR-curve en de ROC-curve. In de webapplicatie bestuderen we telkens één instantie van een algoritme met telkens één bepaalde threshold. Daarvoor is het dus mogelijk om andere scores te gebruiken zoals Index of Balanced Accuracy en F1-score.

6.1.1 IBA - Index of Balanced Accuracy

Index of Balanced Accuracy is een evaluatiescore voor imbalanced classificatie problemen. Deze meet hoe dominant een klasse is met de hoogste individuele nauwkeurigheidsgraad ten opzichte van de andere klasse. IBA wordt als volgt berekend, TPR is de true positive rate, TNR is de true negative rate en α waarbij α een waarde heeft tussen nul en één. Deze geeft aan hoe belangrijk de dominantie factor is:

$$IBA_\alpha = (1 + \alpha \times (TPR - TNR)) \times TNR \times TPR$$

De dominantie-index is de (TPR-TNR) deze berekent welke klasse er dominanter is dan de andere. De waarde schommelt tussen -1 en 1. Omdat zowel de TPR als de TNR een waarde heeft tussen 0 en 1 zal bij een dominantie-index van nul het algoritme het beste score (TPR is dan gelijk aan 1 en TNR is ook gelijk aan één). Als één van beide heel laag is zal de waarde dichterbij 1 of -1 liggen. Deze factor wordt vermenigvuldigd met een α waarde die bepaalt hoe belangrijk deze dominantiefactor is.

Als voorbeeld nemen we een tijdreeks met 1000 datapunten waarvan 10 anomaliën zijn en stel dat het detectiemodel van deze 10 anomalieën er maar één gedetecteerd heeft. Dan is de onderverdeling van de klassen als volgt: 990TN, 1TP en 9FN, 0FP. De TPR zal dan gelijk zijn aan 0.1 en de TNR gelijk aan 1, omdat er geen FP zijn en de data voornamelijk bestaat uit normale datapunten. In dit voorbeeld spreekt men dus van een skewed tijdreeks omdat het aantal normale datapunten veel groter is dan het aantal anomalieën. De IBA-score wordt alsvolgt berekend, waarbij we α gelijkstellen aan 0.5:

$$TPR = \frac{TP}{TP + FN} = \frac{1}{1 + 9} = 0.1$$
$$TNR = \frac{TN}{TN + TNR} = \frac{990}{990 + 0}$$

$$IBA_{0.5} = (1 + 0.5 \times (0.1 - 1)) \times 1 \times 0.1 = 0.055$$

Ondanks dat de TNR heel goed scoort heeft men een dominantiefactor die dicht bij -1 ligt. Deze factor zorgt ervoor dat de uiteindelijke IBA-score zeer laag is (in dit geval gelijk aan 0.55).

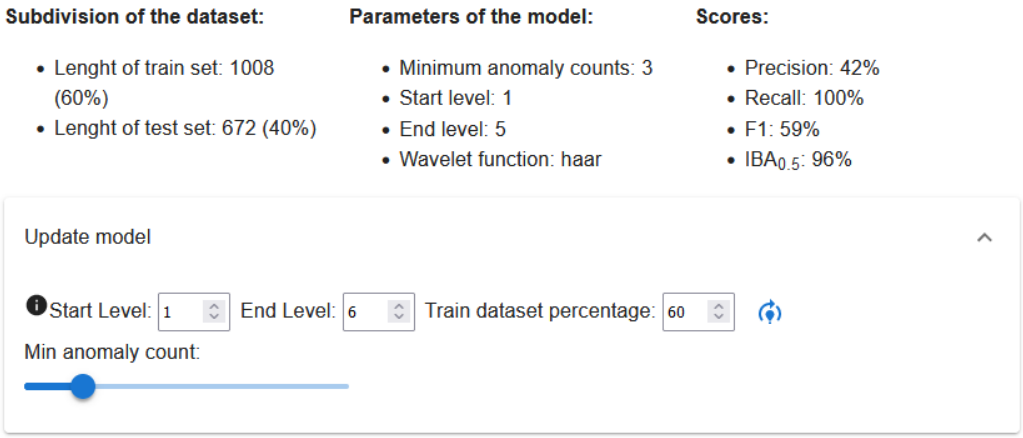
6.1.2 F1

Het is ook mogelijk om precision en recall, zie hoofdstuk 5, te combineren in één score, namelijk de F1-score, dit is het harmonisch gemiddelde tussen precision en recall [32]:

$$F1 = 2(\frac{precision \times recall}{precision + recall})$$

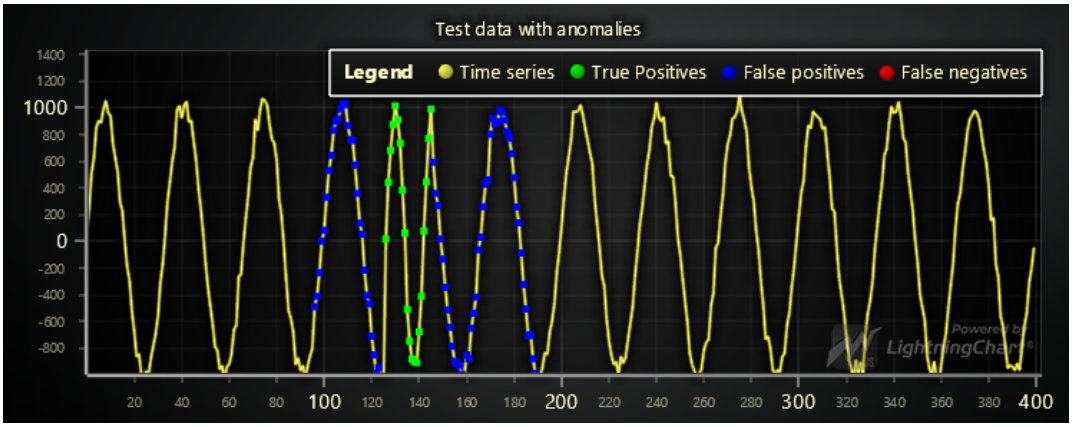
6.2 DWT-MLEAD

Het eerste algoritme van de webapplicatie is DWT-MLEAD. Bovenaan het overzicht staan de belangrijkste parameters. Als startwaarde worden de standaardparameters gebruikt zoals beschreven in sectie 5.1. Naast deze parameters staan ook de precision, de recall, de F1- en de IBA-score. Ze worden berekend met de huidige ingevulde parameters en threshold. De standaard parameters kunnen aangepast worden om duidelijker inzicht te krijgen waarom datapunten gelabeld zijn als anomalie. Door het aanpassen van deze parameters wordt het overzicht, samen met de scores, vernieuwd. Zoals figuur 6.2 aangeeft is het mogelijk om het start- en eindlevel aan te passen samen met de minimale anomalie count. Ook het percentage van de trainset kan worden gewijzigd. Als dit 100% is worden de parameters berekend op de gehele tijdreeks alsook het detecteren van de anomalieën.



Figuur 6.2: Parameters van het DWT-MLEAD model in de webapplicatie.

Na het trainen van het model, worden de anomalieën samen met de testtijdreeks gevisualiseerd zoals weergegeven in figuur 6.3. De gedetecteerde anomalieën worden geclassificeerd als false positives, true positives en false negatives.



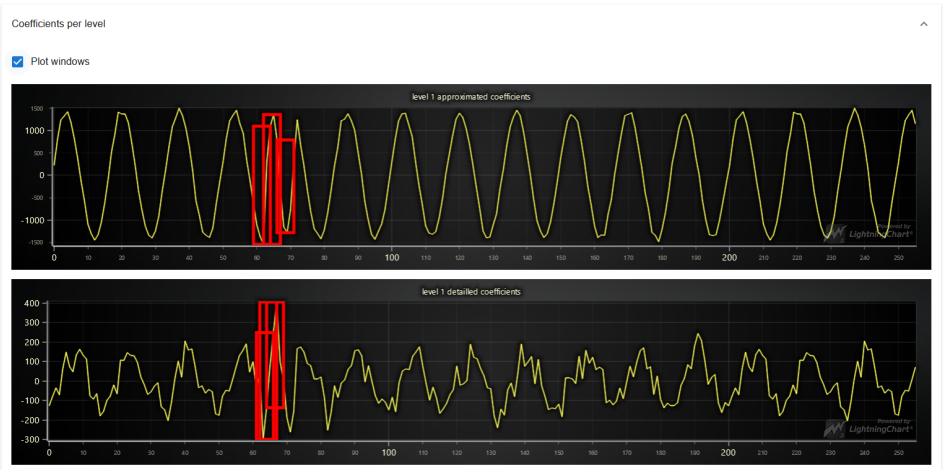
Figuur 6.3: Parameters van het DWT-MLEAD model in de webapplicatie.

De anomalie gelabelde windows, van de verschillende levels van benaderde en gedetailleerde coëfficiënten, worden omgevormd naar de originele dataset en worden samen met de data gevisualiseerd. Het is mogelijk om in de visualisatie enkel de gedetailleerde coëfficiënten, de benaderde coëfficiënten of allebei te tonen samen met de originele tijdreeks. Omdat dit onoverzichtelijk kan worden, naarmate er meer windows zijn, kan men filteren op level.

Alleen de windows die op een bepaald level als anomalie gelabeld worden, zullen dan getoond worden in de grafiek. Dit kan aangepast worden door de range slider te verslepen zoals figuur 6.4 weergeeft. De rode windows zijn afkomstig van de benaderde coëfficiënten terwijl de groene afkomstig zijn van de gedetaillerde coëfficiënten. De blauwe rechthoek geeft de anomalieën weer die een anomalie count hebben die groter of gelijk is aan de huidige "min anomaly count". Deze wordt weergegeven op de slider in figuur 6.2.

Figuur 6.4: Originele tijdreeks met als groene vierkanten de anomalie windows van de gedetailleerde coëfficiënten en de rode vierkanten de anomalie windows van de benaderde coëfficiënten. Het blauwe vierkant zijn de datapunten met een anomalie score die groter zijn dan de "min anomaly count".

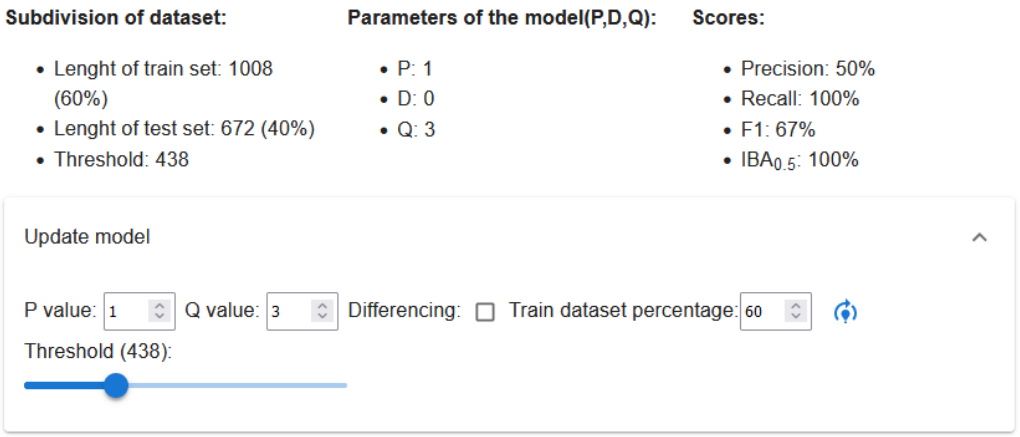
Als laatste kunnen de coëfficiënten per level apart worden geplot samen met de windows die gedetecteerd zijn als anomalie. Figuur 6.5 geeft weer hoe dit gevisualiseerd wordt voor het eerste level in de webapplicatie. Elk level heeft twee grafieken: één voor de benaderde coëfficiënten en één voor de gedetailleerde. De windows die gelabeld zijn als anomalie, zijn te herkennen door een rood vierkant. Dit vierkant kan worden uitgeschakeld zodat alleen nog de coëfficiënten zichtbaar zijn.



Figuur 6.5: Gedetailleerde en benaderde coëfficiënten voor level één, de andere levels worden op dezelfde manier gevisualiseerd.

6.3 ARIMA

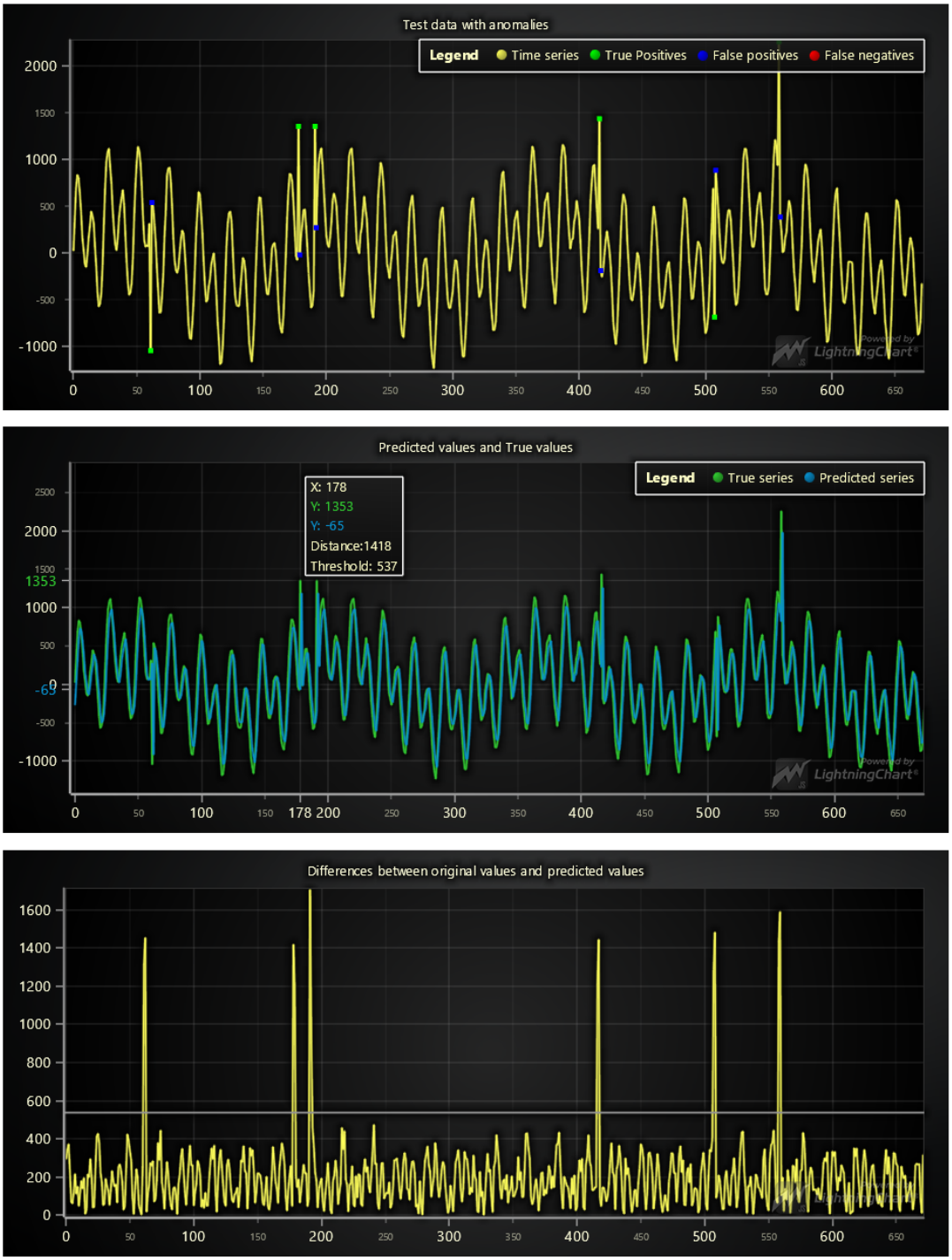
Bovenaan vindt men, net zoals bij de DWT-MLEAD implementatie, de parameters van het algoritme terug alsook de scores van het model met de huidige parameters. Daarnaast is het ook mogelijk om de parameters door middel van het gegeven formulier aan te passen. De standaardparameters zijn de parameters die bepaald worden in sectie 5.1. Zoals figuur 6.6 weergeeft, wordt de lengte van de train- en testtijdreeks getoond samen met de parameters: P, D en Q. P is de orde van het AR-gedeelte, D geeft weer of er differencing nodig is om de data stationair te maken als dit nul is dan is de dataset al stationair. Q is de orde van het MA-gedeelte zoals beschreven in sectie 4.3. Het is mogelijk om deze waarden aan te passen samen met de threshold die gebruikt wordt om te bepalen of een datapunt een anomalie is of niet. Daarnaast kan men differencing aan- en uitzetten om de impact ervan op de dataset weer te geven. Net zoals bij de DWT-MLEAD implementatie wordt er hier ook een foutmelding gegeven als een parameter onjuist ingevuld is.



Figuur 6.6: Headers van het ARIMA-model samen met een formulier om deze parameters aan te passen.

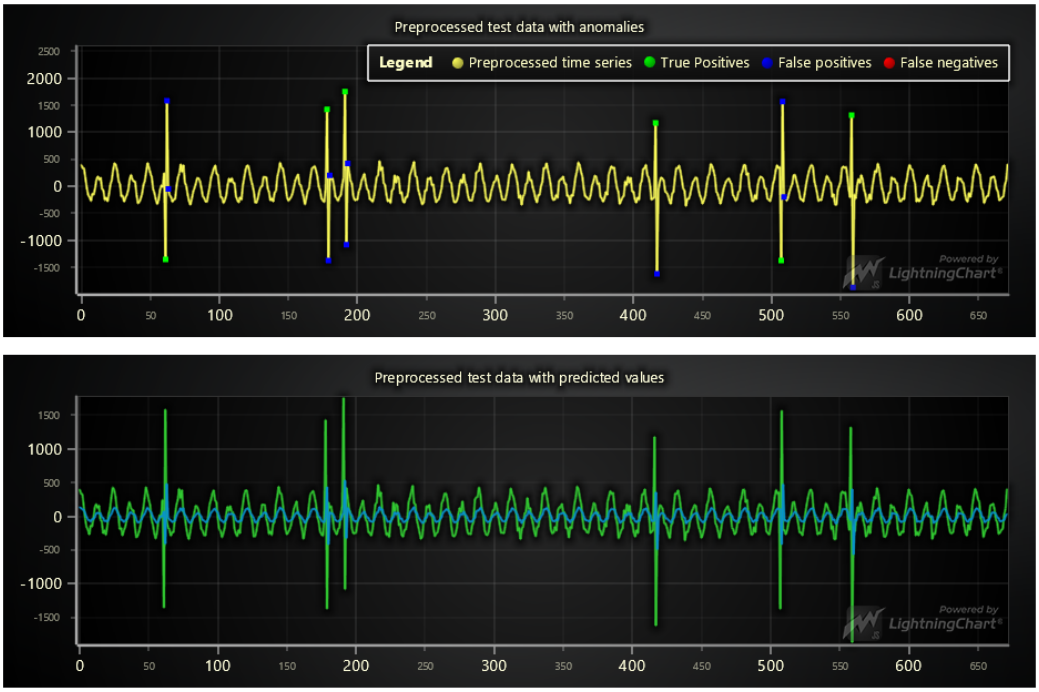
Figuur 6.7 geeft de grafieken weer van de resultaten van het ARIMA algoritme op een gegeven testtijdreeks zonder differencing. De gedetecteerde anomalieën worden weergegeven aan de hand van true positives, false positives en false negatives. True negatives zijn al de andere datapunten, deze zijn datapunten die oorspronkelijk normaal zijn en ook niet gedetecteerd worden als anomalie. De tweede grafiek (de grafiek met als titel "Predicted values and true values") visualiseert de originele tijdreeks samen met de voorspellingen die gemaakt zijn door middel van het ARIMA-model. Als men met de muis over deze grafiek beweegt ziet men de waarde voor het originele datapunt samen met de overeenkomstige voorspelde waarde. Dit maakt het eenvoudiger om de afstand tussen de twee datapunten te vergelijken met een threshold.

De onderste grafiek geeft de verschillen weer tussen de originele waarden en de overeenkomstige voorspelde waarden. De dunne witte lijn in deze grafiek duidt op de threshold. Wanneer men de slider aanpast in figuur 6.6 dan zal deze lijn mee verschuiven. Als een afstand boven de threshold ligt worden deze voorspeld als anomalie. Dit maakt het duidelijker waarom sommige datapunten gelabeld worden als anomalie en de andere niet.



Figuur 6.7: Visualisatie in de webapplicatie webapplicatie om de beslissingen, waarom een datapunt anomalie is of niet in de context van het ARIMA algoritme, begrijpbaarder te maken. De bovenste grafiek toont de datapunten met de gelabelde anomalieën, de tweede grafiek zijn de originele waarden met de overeenkomstige voorspelde waarden. De onderste grafiek toont de verschillen tussen beide samen met de threshold.

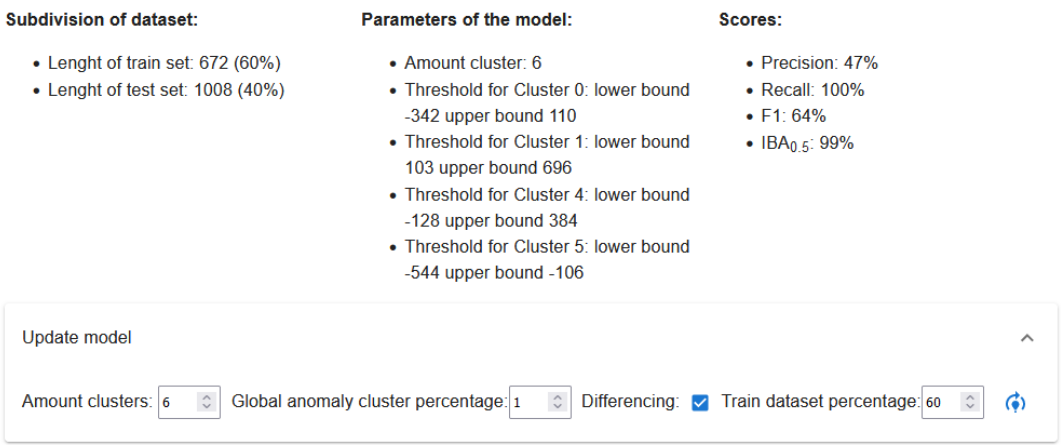
Als men differencing aanvinkt of een niet-stationaire tijdreeks neemt, ziet men dat er extra grafieken zijn. Het is nu mogelijk om te bekijken hoe de tijdreeks eruitziet na het toepassen van het differencing algoritme samen met de voorspellingen die het ARIMA algoritme maakt op deze bewerkte tijdreeks. Alles wordt nu berekend op de bewerkte tijdreeks. Het algoritme neemt dus de bewerkte tijdreeks om voorspellingen te maken, threshold te berekenen en om datapunten te labelen. De verschillen zoals weergegven in figuur 6.7 zijn nu de verschillen tussen de originele en de voorspelde waarden van de bewerkte tijdreeks. Om de voorspellingen te visualiseren op de originele dataset worden deze terug omgevormd zodat deze overeenkomen met de data in de originele tijdreeks. Ook al is de originele dataset stationair dan is het nog steeds mogelijk om deze optie aan te duiden. Het is in beide gevallen mogelijk om dit uit te schakelen. Figuur 6.8 geeft weer welke grafieken erbij komen bij het uitvoeren van differencing. Deze grafieken worden samen gevisualiseerd met de grafieken uit figuur 6.7.



Figuur 6.8: Een gegeven tijdreeks waarop voorspelling met behulp van ARIMA zijn gemaakt. De bovenste grafiek geeft weer wat de labels zijn van de datapunten (TP, FP of FN) terwijl de onderste grafiek de voorspelde waarden samen met de originele waarden zijn.

6.4 K-Means

Zoals bij de vorige algoritmes, is er voor K-Means ook een overzicht gemaakt dat de belangrijkste parameters samen met de scores toont op het scherm zoals weergegeven in figuur 6.9. Daarnaast is het ook mogelijk om verschillende parameters van het model aan te passen zoals: het aantal clusters, globaal cluster anomaliepercentage, het toepassen van differencing en het percentage van het aantal datapunten uit deze tijdreeks dat gebruikt wordt als testdata. Het globale cluster anomaliepercentage bepaalt hoeveel punten een cluster, in vergelijking met de testdata, minimaal moet hebben om niet als een globale anomalie beschouwd te worden zoals eerder vermeld in hoofdstuk 4.



Figuur 6.9: Parameters van het K-Means algoritme.

Bij een dataset zonder differencing, of wanneer er de optie "differencingüit staat, zijn er twee grafieken die getoond worden. De eerste grafiek gaat, net zoals bij de vorige algoritmes, de gedetecteerde anomalieën labelen als false positives, false negatives en true negatives. Deze worden samen met de originele tijdreeks weergegeven in de bovenste grafiek, in de onderste grafiek op figuur 6.10 worden de punten met hun bijbehorende cluster gevisualiseerd. Dit geeft een overzicht welk datapunt bij welke cluster hoort om eventueel een beter optimaal aantal clusters te definiëren in de parameters op figuur 6.9.

Figuur 6.10: De bovenste grafiek geeft de classificaties van de gedetecteerde anomalieën samen met de originele tijdreeks weer, terwijl de onderste grafiek de punten per cluster weergeeft.

Omdat K-Means, net zoals ARIMA, beter werkt op stationaire data, is het mogelijk om ook de bewerkte dataset te tonen. De bewerkte dataset is de originele tijdreeks waar het differencing algoritme op toegepast wordt om de data stationair te maken. Dit zorgt ervoor dat we voorspellingen maken op de bewerkte dataset. Als er differencing toegepast is op de dataset worden er twee extra grafieken getoond zoals weergegeven op figuur 6.11. De bovenste grafiek geeft weer hoe de tijdreeks eruit ziet na het toepassen van het differencing algoritme. Daarnaast worden de gedetecteerde anomalieën opnieuw geclassificeerd en getoond op de grafiek. De onderste grafiek geeft weer wat de datapunten per cluster zijn voor de bewerkte dataset. Deze worden dan terug omgevormd naar de datapunten per cluster op de originele dataset zoals al eerder getoond in figuur 6.10.

Figuur 6.11: Extra grafieken na het toepassen van het differencing algoritme.

Hoofdstuk 7

Conclusies

In deze masterthesis heb ik mij kunnen verdiepen in het boeiende onderwerp van anomaliedetectie en het visualiseren van de redenen waarom een model datapunten als anomalie classificeert. Gedurende het academiejaar heb ik mij gericht op het implementeren van verschillende univariate unsupervised technieken en het ontwikkelen van een webapplicatie om keuzes en parameters visueel weer te geven.

Anomaliedetectie is een breed vakgebied met diverse technieken, en hoewel ik me hier voornamelijk heb gericht op univariate unsupervised methoden, besef ik dat er nog veel andere benaderingen bestaan die niet in deze thesis zijn behandeld. Het zou interessant zijn om dit onderzoek uit te breiden naar andere technieken en deze te implementeren in de webapplicatie. Hierdoor zou de applicatie nog waardevoller worden voor praktische toepassingen.

Tijdens het onderzoek kwam ik tot de constatering dat de huidige algoritmes weliswaar goed presteren op train- en testsets, maar dat ze nog niet zijn getest in een streaming omgeving. Het aanpassen en testen van deze algoritmes voor real-time verwerking van data zou een waardevolle volgende stap zijn in het verbeteren van de prestaties en het vergroten van de toepasbaarheid in organisaties die grote hoeveelheden data verwerken.

Een uitdaging die ik ben tegengekomen bij het uitvoeren van dit onderzoek, was het verkrijgen van kwalitatieve testresultaten. Er zijn verschillende soorten anomalieën die kunnen optreden, waardoor het vinden van geschikte datasets lastig kan zijn. Hoewel benchmark datasets vaak een mix van puntanomalieën en sequentieanomalieën bevatten, zijn ze meestal gericht op globale anomalieën. Het zou waardevol zijn om in een toekomstig onderzoek nieuwe methoden te ontwikkelen om kwalitatieve testresultaten te verkrijgen voor verschillende soorten anomalieën.

Persoonlijk heb ik veel geleerd tijdens het uitvoeren van dit onderzoek. Het implementeren van de algoritmes en het ontwikkelen van de webapplicatie hebben mijn programmeervaardigheden versterkt. Het analyseren en visualiseren van de modelbeslissingen heeft me een dieper inzicht gegeven in de complexiteit van anomaliedetectie. Ik ben trots op de resultaten die ik heb behaald en ik geloof dat deze thesis een waardevolle bijdrage levert aan het begrip en de toepassing van anomaliedetectie.

In de toekomst zou het interessant zijn om dit onderzoek verder uit te breiden, zowel op technisch als op praktisch gebied. Het aanpassen van de algoritmes voor streaming omgevingen en het ontwikkelen van nieuwe methoden voor kwalitatieve testresultaten kunnen de bruikbaarheid en effectiviteit van anomaliedetectie verder verbeteren. Bovendien kan de webapplicatie worden uitgebreid met meer algoritmes en functionaliteiten, waardoor het een waardevol instrument wordt voor professionals in diverse domeinen.

Bibliografie

[1] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A. Lozano. A Review on Outlier/Anomaly Detection in Time Series Data. *ACM Computing Surveys (CSUR)*, 54(3), 4 2021.

[2] KMI - Recente waarnemingen te Ukkel. URL: `https://www.meteo.be/nl/klimaat/klimaat-van-belgie/recente-waarnemingen-te-ukkel`, geraadpleegd op 26-11-2022.

[3] Georgios Meditskos, Pierre Marie Plans, Thanos G. Stavropoulos, Jenny Benois-Pineau, Vincent Buso, and Ioannis Kompatsiaris. Multi-modal activity recognition from egocentric vision, semantic enrichment and lifelogging applications for the care of dementia. *Journal of Visual Communication and Image Representation*, 51:169–190, 2 2018.

[4] Ronghua Sun, Qianxun Wang, and Liang Guo. Research Towards Key Issues of API Security. *Communications in Computer and Information Science*, 1506 CCIS:179–192, 2022.

[5] Principal Component Analysis (PCA) Explained Visually with Zero Math — by Casey Cheng — Towards Data Science. URL: `https://towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d`, geraadpleegd op 27-12-2022.

[6] Using Moving and Tumbling Windows to Highlight Trends — VMware Aria Operations for Applications Documentation. URL: `https://docs.wavefront.com/query_language_windows_trends.html`, geraadpleegd op 26-05-2023.

[7] GitHub - numenta/NAB: The Numenta Anomaly Benchmark. URL: `https://github.com/numenta/NAB`, geraadpleegd op 21-01-2023.

[8] GitHub - waico/SKAB: SKAB - Skoltech Anomaly Benchmark. Time-series data for evaluating Anomaly Detection algorithms. URL: `https://github.com/waico/SKAB`, geraadpleegd op 21-01-2023.

[9] Gutentag - A good Timeseries Anomaly Generator. URL: `https://github.com/HPI-Information-Systems/gutentag/blob/main/doc/index.md`, geraadpleegd op 22-01-2023.

[10] S5 - A Labeled Anomaly Detection Dataset. URL: `https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70`, geraadpleegd op 19-05-2023.

[11] Deep Learning Based Anomaly Detection for Muti-dimensional Time Series: A Survey. URL: `https://link.springer.com/chapter/10.1007/978-981-16-9229-1_5`, geraadpleegd op 19-05-2023.

[12] K-Means - Clustering. URL: `https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning`, geraadpleegd op 23-01-2023.

[13] Time Series Data Analysis with sARIMA and Dash — by Gabriele Albini. URL: `https://towardsdatascience.com/`

time-series-data-analysis-with-sarima-and-dash-f4199c3fc092, geraadpleegd op 09-05-2023.

[14] Autocorrelation and Autocovariance: Calculation, Examples, and More. URL: https://blog.quantinsti.com/autocorrelation-autocovariance/, geraadpleegd op 14-06-2023.

[15] Quick way to find p, d and q values for ARIMA. URL: https://analyticsindiamag.com/quick-way-to-find-p-d-and-q-values-for-arima/, geraadpleegd op 09-05-2023.

[16] Understanding Partial Auto-correlation And The PACF – Time Series Analysis, Regression, and Forecasting. URL: https://timeseriesreasoning.com/contents/partial-auto-correlation, geraadpleegd op 14-06-2023.

[17] Olga Valenzuela. Proceedings ITISE 2017 : International Work-Conference on Time Series, Granada, September, 18-20 2017. 2017.

[18] Deconstructing Time Series using Fourier Transform — by Khairul Omar. URL: https://medium.com/@khairulomar/deconstructing-time-series-using-fourier-transform-e52dd535a44e, geraadpleegd op 05-04-2023.

[19] Sneller algoritme voor Fourier-transformaties ontwikkeld. URL: https://tweakers.net/nieuws/79444/sneller-algoritme-voor-fourier-transformaties-ontwikkeld.html, geraadpleegd op 27-05-2023.

[20] Fourier Transformation: Introduction. URL: https://www.thefouriertransform.com/#introduction, geraadpleegd op 05-04-2023.

[21] Hohyub Jeon, Yongchul Jung, Seongjoo Lee, and Yunho Jung. Area-efficient short-time fourier transform processor for time–frequency analysis of non-stationary signals. *Applied Sciences (Switzerland)*, 10(20):1–10, 10 2020.

[22] Short-time Fourier transform. URL: https://en.wikipedia.org/wiki/Short-time_Fourier_transform, geraadpleegd op 05-04-2023.

[23] Ü Lepik. Numerical solution of differential equations using Haar wavelets. *Mathematics and Computers in Simulation*, 68(2):127–143, 4 2005.

[24] Fatemeh Safara, Shyamala Doraisamy, Azreen Azman, Azrul Jantan, and Sri Ranga. Wavelet packet entropy for heart murmurs classification. *Advances in Bioinformatics*, 2012, 2012.

[25] Olga Valenzuela. Proceedings ITISE 2017 : International Work-Conference on Time Series, Granada, September, 18-20 2017. 2017.

[26] Anomaly Detection with the Normal Distribution. URL: https://anomaly.io/anomaly-detection-normal-distribution/index.html, geraadpleegd op 28-05-2023.

[27] F. Gregory Ashby and W. William Lee. Predicting Similarity and Categorization From Identification. *Journal of Experimental Psychology: General*, 120(2):150–172, 1991.

[28] Chuong B Do. The Multivariate Gaussian Distribution. Technical report, 2008.

[29] Covariance: What is Covariance? URL: https://www.cuemath.com/covariance-formula, geraadpleegd op 06-04-2023.

[30] An Approach for Choosing Number of Clusters for K-Means — by Or Herman-Saffar. URL: https://towardsdatascience.com/an-approach-for-choosing-number-of-clusters-for-k-means-c28e614ecb2c, geraadpleegd op 05-05-2023.

[31] Machine Learning Accuracy: True-False Positive/Negative. URL: https://research.aimultiple.com/machine-learning-accuracy/, geraadpleegd op 11-05-2023.

[32] Sepand Haghighi, Masoomeh Jasemi, Shaahin Hessabi, and Alireza Zolanvari. PyCM: Multiclass confusion matrix library in python. *Journal of Open Source Software*, 3(25):729, may 2018.

[33] A Look at Precision, Recall, and F1-Score — by Teemu Kanstrén. URL: `https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec`, geraadpleegd op 20-05-2023.

[34] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE*, 10(3), 3 2015.

[35] Receiver operating characteristic. URL: `https://en.wikipedia.org/wiki/Receiver_operating_characteristic`, geraadpleegd op 14-05-2023.

[36] Receiver Operating Characteristic Curve in Diagnostic Test Assessment. URL: `https://www.sciencedirect.com/science/article/pii/S1556086415306043`, geraadpleegd op 14-05-2023.

[37] Classification: ROC Curve and AUC — Machine Learning . URL: `https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc`, geraadpleegd op 14-05-2023.

[38] A Deep Dive on ARIMA Models. — by Matt Sosna. URL: `https://towardsdatascience.com/a-deep-dive-on-arima-models-8900c199ccf`, geraadpleegd op 27-04-2023.

[39] V. García, R. A. Mollineda, and J. S. Sánchez. Index of balanced accuracy: A performance measure for skewed class distributions. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5524 LNCS:441–448, 2009.

[40] RPubs - Gaussian White Noise Time Series. URL: `https://rpubs.com/ks190995/118845`, geraadpleegd op 27-04-2023.

[41] Dong Han, Valerie Renaudin, and Miguel Ortiz. Smartphone based gait analysis using STFT and wavelet transform for indoor navigation. *IPIN 2014 - 2014 International Conference on Indoor Positioning and Indoor Navigation*, pages 157–166, 2014.

[42] Time series: Introduction. URL: `https://en.wikipedia.org/wiki/Time_series`, geraadpleegd op 26-11-2022.

[43] Understanding the Covariance Matrix. URL: `https://datascienceplus.com/understanding-the-covariance-matrix/`, geraadpleegd op 06-04-2023.