Fysiologische profielen: vergelijken van machine learning modellen om professionele voetballers te clusteren op basis van fysieke testdata

DE HAAGSE HOGESCHOOL

Afstudeerscriptie

Naam: Jort Stienstra

Studentnummer: 16105370

Datum: Juni 2023

Eerste begeleider: Onur Tezel, MSc

Tweede begeleider: Erik Wilmes, MSc

Opdrachtgever: Onur Tezel

Opleiding Bewegingstechnologie, De Haagse Hogeschool

Samenvatting

Inleiding: Uit wetenschappelijk onderzoek blijkt dat er variatie bestaat in de interne belasting die voetbalspelers ervaren, zelfs bij een gelijke externe belasting(Castagna et al., 2011). De fysieke capaciteiten van professionele voetballers kunnen hier invloed op hebben (Rabbani, et al., 2021). Dit verschil in interne belasting heeft ook invloed op de trainingsbelasting die spelers ervaren. Aangezien de trainingsbelasting een significante factor is bij het bepalen van de hersteltijd, kan het verschil in hersteltijd tussen spelers resulteren in de behoefte aan verdere optimalisatie van de periodisering (Nedelec et al., 2012). Daarom kan het voordelig zijn om speler op te delen in homogene subgroepen op basis van hun fysieke capaciteiten. Unsupervised machine learning modellen (UMLC) kunnen hierbij helpen.
Doel: Dit wetenschappelijk onderzoek vergelijkt drie UMLC-modellen om homogene subgroepen te identificeren in een dataset met anaerobe en aerobe gegevens van professionele voetballers. Met als doel op de best presterende model te vinden.
Methode: Het onderzoek betreft van dataset 41 professionele voetbalspelers (leeftijd: 17.7 ± 1.5) gedurende drie seizoenen, waarbij verschillende anaerobe en aerobe testen zijn uitgevoerd. De UMLC-modellen: K-means, DBSCAN en GMM zijn gebruikt om homogene subgroepen binnen de dataset te identificeren. De modellen zijn geëvalueerd aan de hand van interne validatietesten, namelijk de Silhouette index (SI) en Dunn index (DI) en visuele inspecties van de clusters.
Resultaten: K-means en GMM vertonen vergelijkbare resultaten op zowel de SI (0.16410, 0.15436 respectievelijk) als op de DI (0.10971, 0.17300 respectievelijk). De visuele tweedimensionaal-weergave van deze modellen tonen ook vergelijkbare clusters. Statistische analyse onthuld significante verschillen tussen de clusters voor alle zes de testvariabelen in het geval van K-means ($p < 0.05$), terwijl GMM significantie verschillen vertoonde voor alle testvariabelen behalve *Agility* ($p = 0.077$). Daarentegen liet DBSCAN een lagere SI-waarde zien (-0.0054) en een hogere DI-waarde (0.25923). Visueel waren de clusters van DBSCAN moeilijker van elkaar te onderscheiden, en de statistische analyse toont alleen significante verschillen tussen de clusters voor de testvariabelen *sprint 30m* en *Power output*.
Conclusie: Uit de bevindingen van dit onderzoek blijkt dat DBSCAN niet geschikt is voor het identificeren van fysiek homogene subgroepen in een dataset van professionele voetballers. Daarentegen slaagden zowel K-means als GMM er wel in om homogene subgroepen te vinden. Hoewel er kleine verschillen waren tussen de modellen, lijkt K-means het meest geschikte model vanwege andere factoren zoals het gebruiksgemak en de beschikbaarheid van wetenschappelijk onderzoek. Daarom verdient K-means de voorkeur voor verdere toepassing in het onderzoek naar fysiologische profielen in het kader van optimaliseren van periodisering bij professionele voetballers.

Inhoudsopgave

1 Inleiding 4
2 Methode 6
2.1 Participanten 6
2.2 Experimenteel design 6
2.3 Anaerobe fitheidstesten 6
Sprinttest 6
Behendigheidstest 7
Countermovement jump 7
Drop jump 7
2.4 Aerobe fitheidstesten 7
Yo-Yo onderbroken hersteltest 7
2.5 Unsupervised machine learning modellen 8
K-means 8
DBSCAN 8
GMM 9
2.6 Gegevensverwerking 9
2.7 Voorlopige analyse: 10
2.8 Cluster analyse 11
3 Resultaten 11
3.1 Voorlopige resultaten 11
3.2 Cluster resultaten 14
K-means 14
DBSCAN 16
GMM 18
3.3 Interne validatie resultaten 20
4 Discussie 21
4.1 DBSCAN 21
4.2 K-means en GMM 21
4.3 Clusters 22
4.4 Beperkingen 22
4.5 Aanbevelingen 23
4.6 Praktische relevantie 23
5 Conclusie 24
6 Literatuurlijst 25
7 Bijlage 27
Bijlage 1. Voorlopige analyse 27
Bijlage 2. Statistische toets resultaten 28
Bijlage 3. Python code 32

1 Inleiding

Tijdens voetbal wedstrijden wordt overwegend het aerobe energiesysteem aangesproken, met een gebruik van meer dan 90%. Het anaerobe systeem wordt voornamelijk aangesproken tijdens situaties waarbij er sprake is van duel tussen spelers (Chamari, et al., 2004). Het anaerobe systeem wordt geactiveerd wanneer er snel energie nodig is en er niet voldoende zuurstof beschikbaar is om het aerobe systeem te ondersteunen. Dit systeem wordt voornamelijk gebruikt bij korte, krachtige inspanningen zoals sprinten, gewichtheffen en andere activiteiten waarbij de spieren snel en krachtig moeten worden geactiveerd. In het anaerobe energiesysteem wordt energie gegenereerd door de anaerobe afbraak van glucose of glycogeen, waarbij zuurstof niet vereist is. Deze afbraak resulteert in de productie van ATP (adenosinetrifosfaat) en de vorming van lactaat als bijproduct. Het aerobe systeem daarentegen maakt gebruik van zuurstof om energie te produceren en wordt gebruikt bij langdurige, matige tot hoge intensiteit activiteiten zoals hardlopen, fietsen en zwemmen. Bij dit systeem worden vetten, koolhydraten en eiwitten afgebroken om ATP te produceren, de belangrijkste energiebron voor de cellen in het lichaam.

Onder begeleiding van coaches en trainers worden de aerobe en anaerobe fitheid van voetbalspelers getraind met als doel supercompensatie te bereiken. Tijdens de training ondergaat een voetbalspeler een belastingsperiode gevolgd door een herstelperiode, waarin het lichaam zich aanpast aan de stress van de trainingsbelasting en sterker en veerkrachtiger wordt. Het is essentieel om tijdens deze herstelperiode een geschikt moment te kiezen om het proces te onderbreken, zodat er gebruik kan worden gemaakt van supercompensatie en de fysieke fitheid van de voetbalspeler verbetert. Om beter inzicht te krijgen in de voortgang van voetbalspelers is het belangrijk om de anaerobe en aerobe capaciteiten objectiveren. Dit kan helpen om de fitheid van spelers ten opzichte van zichzelf en hun teamgenoten te beoordelen (Svensson & Drust, 2005).

De trainingsbelasting tijdens trainingen en wedstrijden kan worden opgesplitst in interne en externe belasting. Externe belasting verwijst naar de fysieke eisen die aan een individu worden gesteld, zoals afstand, snelheid en intensiteit. Interne belasting heeft betrekking op de fysiologische en psychologische respons van het individu op de externe belasting, zoals hartslag, lactaatniveaus, perceptie van inspanning en de mechanische krachten op pezen en spieren. Uit onderzoek blijkt dat er verschil zit in de interne trainingsbelasting die spelers ervaren bij een gelijke externe trainingsbelasting (Castagna et al., 2011; Impellizzeri et al., 2004). De anaerobe en aerobe capaciteiten van een voetbal speler kunnen van invloed zijn op de interne belasting die ze ervaren tijdens dezelfde externe belasting (Rabbani, et al., 2021). Als gevolg zit er ook een verschil in de trainingsbelasting. Het verschil in trainingbelasting kan dan weer leiden tot een verschil in de hersteltijd van een voetbal spelers (Nedelec et al., 2012). Op basis van de anaerobe en aerobe capaciteiten van een individu en daardoor een verschil in de herseltijd, zou de periodisering verder geoptimaliseerd kunnen worden. Periodisering is een trainingsstrategie waarbij het trainingsprogramma wordt opgedeeld in fases of periodes. Het doel van periodisering is om de trainingen te structureren om overbelasting te voorkomen, optimale prestaties te bevorderen en effectief herstel mogelijk te maken.

Slimani en Nikolaidis (2018) hebben in een systematische review het verschil beschreven tussen aerobe en anaerobe testen voor verschillende competitieniveaus, posities en leeftijden. Hierbij werden verschillende aerobe en anaerobe testen onafhankelijk van elkaar met elkaar vergeleken en op basis daarvan werden fysiologische profielen gecreëerd. Om echter betekenisvollere fysiologische profielen te genereren, moeten deze capaciteiten in samenhang met elkaar worden onderzocht, in

plaats van onafhankelijk van elkaar. Deze benadering vormt een meer holistisch beeld van de fysieke eigenschappen van een voetbalspeler en kan helpen bij het ontdekken van relaties tussen verschillende aerobe en anaerobe testen.

Machine learning (ML) kan hierbij helpen. ML-modellen zijn speciaal ontworpen om patronen en relaties binnen een dataset te identificeren, wat kan leiden tot nieuwe inzichten in complexe datasets. In ML worden datapunten doorgaans weergegeven als vectoren van kenmerken of attributen die de eigenschappen van de objecten beschrijven die worden geanalyseerd. ML is in wetenschappelijk onderzoek al gebruikt bij het analyseren van grote datasets om waardevolle informatie te verzamelen, zoals het voorspellen van blessures (Rossi, et al., 2018) en het uitvoeren van tactische analyses (Rein & Memmert, 2016). Unsupervised machine learning clustering (UMLC), een subdomein van ML, wordt gebruikt om patronen te ontdekken in grote en complexe datasets zonder voorkennis van de uitkomst en is reeds gebruikt om homogene groepen te creëren (Akhtar, et al., 2019; Misra & Yadav, 2020; Van Der Zwaard et al., 2019).

Op dit moment bestaat er geen wetenschappelijke kennis over welk UMLC-model het beste presteert bij het clusteren van een dataset met anaerobe en aerobe testgegevens van professionele voetbalspelers. Er zijn echter vergelijkbare onderzoeken uitgevoerd in de gezondheidszorg, zoals het onderzoek van Lukauskas en Ruzgas (2023), waaruit bleek dat de prestaties van clusteringmodellen afhankelijk zijn van de specifieke dataset. Het doel van dit onderzoek is om verschillende clusteringmodellen te vergelijken en inzicht te krijgen in hun prestaties. Binnen UMLC zijn er verschillende categorieën, centroid-, dichtheids-, kans- gebaseerd, die elk hun eigen methode van clusteren hebben. Centroid-gebaseerd clustering groepeert datapunten op basis van hun nabijheid tot een centraal punt, het centroid, door iteratief de centroids aan te passen om de afstand tussen de datapunten en hun toegewezen centroid te minimaliseren. Dichtheids-gebaseerd clustering groepeert datapunten op basis van hun nabijheid tot elkaar in termen van dichtheid, waarbij punten in gebieden met hoge dichtheid worden beschouwd als onderdeel van dezelfde cluster, terwijl punten in gebieden met lage dichtheid als ruis worden beschouwd. Kans-gebaseerd clustering wijst datapunten toe aan clusters op basis van hun waarschijnlijkheid om tot elke cluster te behoren, met behulp van een kansverdeling om de gegevens te modelleren en het datapunt toe te wijzen aan de cluster met de hoogste waarschijnlijkheid. Voor dit onderzoek is er per methode een model gekozen: K-means (centroid), Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (dichtheid) en Gaussian Mixture Models (GMM) (kans). Deze modellen zijn eerder toegepast in sportgerelateerd onderzoek. Elk model heeft zijn eigen voor- en nadelen die relevant zijn voor dit onderzoek.

In het onderzoek van Van der Zwaard et al. (2019) werd K-means clustering gebruikt om professionele wielrenners te groeperen op basis van hun antropometrische gegevens. Shelly et al. (2020) gebruikten K-means om de bestaande trainingsgroepen in het Amerikaanse football te valideren op basis van de spelersposities tijdens wedstrijden. K-means is eenvoudig te begrijpen en te implementeren en het presteert goed bij grote datasets. Het kan efficiënt clusters genereren wanneer de data goed gescheiden is. Een nadeel van K-means is echter dat het gevoelig is voor de initiële keuze van de centroids, wat kan resulteren in suboptimale oplossingen. Bovendien gaat K-means ervan uit dat de clusters sferisch zijn en vergelijkbare maten hebben, wat niet altijd overeenkomt met de complexe patronen in real-world datasets.

In recent werk door Das et al., (2023) werd DBSCAN clustering toegepast in het cricket om clusters te vormen die kunnen helpen bij data-gedreven besluitvorming met betrekking tot het wisselbeleid. Een voordeel van DBSCAN is dat het clusters van willekeurige vormen en maten kan identificeren, waardoor het flexibeler is dan K-means. Het algoritme is robuust tegen ruis en uitschieters en kan automatisch het aantal clusters bepalen op basis van de gegevens. Echter, DBSCAN kan gevoelig zijn

voor de keuze van hyperparameters, zoals de afstand tussen datapunten en het minimum aantal punten dat nodig is om een cluster te vormen.

Soto-Valero (2017) paste GMM clustering toe om voetbalspelers te groeperen op basis van gegevens uit een EA Sports FIFA-videogame. GMM kan complexe gegevensdistributies modelleren en clusters van willekeurige vormen identificeren. Het gebruik van GMM in clustering biedt probabilistische toewijzingen van datapunten aan clusters, wat waardevol kan zijn in bepaalde toepassingen, zoals het beoordelen van de onzekerheid van cluster-toewijzingen. Dit is vooral nuttig in situaties waarin datapunten ambigu zijn of wanneer er overlap is tussen clusters. Net als de andere modellen is GMM echter gevoelig voor de keuze van hyperparameters en kan het berekeningsintensief zijn voor grote datasets.

Voor het vergelijken van modellen worden twee verschillende interne validatietesten gebruikt, namelijk de "silhouette-index" (SI) (Rousseeuw, P. J. 1987) en de "Dunn-index" (DI) (Dunn, J. C. 1973). Beide testen zijn toegepast in onderzoek naar het vergelijken van UMLC-modellen (Abdulnassar & Nair, 2023; Nawrin et al., 2017). Abdulnassar & Nair (2023) rapporteerden SI-waarden tussen 0.19 en 0.86 en DI-waarden tussen 0.77 en 1.7. Deze testen bieden inzicht in de clustering en onderlinge afstand van de clusters, waardoor de kwaliteit van de clusters beoordeeld kan worden. Hierbij geldt dat optimale clusters zo ver mogelijk van elkaar afliggen en zo compact mogelijk zijn. De gevormde subgroepen met vergelijkbare fysieke capaciteiten kunnen vervolgens bijdragen aan de verbetering van de periodisering, waarbij rekening wordt gehouden met verschillen in hersteltijd en de sterke en zwakke punten van de subgroepen.

De onderzoeksvraag luidt : welk UMLC-model presteert het beste bij het identificeren van homogene subgroepen in anaerobe en aerobe testgegevens van professionele voetbalspelers?

2 Methode

2.1 Participanten

41 professionele voetbalspelers (17.7 ± 1.5) zijn gedurende drie seizoenen geobserveerd, verdeeld over drie teams: onder 19, jong Ajax en Ajax 1. Alle spelers waren afkomstig van dezelfde professionele voetbalclub en speelden op nationaal niveau. Om praktische redenen zijn alleen veldspelers meegenomen in dit onderzoek.

2.2 Experimenteel design

Het experimentele design omvatte het regelmatig onderwerpen van spelers aan laboratoriumtests gedurende drie seizoenen om hun aerobe en anaerobe capaciteit te evalueren. De tests die werden uitgevoerd waren gerelateerd aan de bewegingen die tijdens trainingen en wedstrijden worden verricht. De tests werden meerdere keren gedurende de seizoenen uitgevoerd.

2.3 Anaerobe fitheidstesten

Sprinttest

Een 30 meter sprinttest is uitgevoerd om anaerobe data te verzamelen bij voetbal-specifieke training. Deze test meet de tijd die nodig is om de sprint te voltooien en wordt in dit onderzoek aangeduid als de variabele *sprint 30m*.

Behendigheidstest

Een behendigheidstest met meerdere draaimomenten van 270 en 315 graden over een parcours van ongeveer 30 meter is intern ontwikkeld en uitgevoerd om anaerobe gegevens te verzamelen bij voetbal-specifieke training. De test meet de tijd tot voltooiing van het parcours als uitgangsparameter en wordt in dit onderzoek aangeduid als de variabele *Agility*.

Countermovement jump

De countermovement jump is een sprongtest die gegevens oplevert over de maximale hoogte van de sprong en de kracht (N/kg). In dit onderzoek worden de variabele aangegeven als *Max of jumpheight* en *Power output*, respectievelijk. De CMJ begint met een neerwaartse beweging gevolgd door een maximale opwaartse sprong. Er is wetenschappelijk onderzoek uitgevoerd waaruit blijkt dat de CMJ een betrouwbare test is (Marković et al., 2004).

Drop jump

De drop jump (DJ) is een sprongtest die de relative strength index (RSI) meet. RSI wordt gedefinieerd als de spronghoogte gedeeld door de contacttijd en wordt in dit onderzoek aangeduid als *RSI*. Bij de uitvoering van de DJ wordt een sprong gemaakt vanaf een verhoogd platform, waarbij het doel is om na de landing zo snel, zo hoog mogelijk te springen. Er is wetenschappelijk onderzoek verricht dat aantoont dat de DJ een betrouwbare test is (Flanagan et al., 2008).

Sprint- en sprongprestaties zijn veelvuldig bestudeerd en worden gebruikt om anaerobe fitheid te beoordelen. Onderzoek heeft aangetoond dat sprintprestaties valide testen zijn voor het meten van anaerobe vermogen, zoals gemeten met een Wingate-test (Nara et al., 2022). Onderzoek van Çakir-Atabek, (2014) heeft aangetoond dat sprongprestaties een valide test zijn voor anaerobe capaciteit.

2.4 Aerobe fitheidstesten

Yo-Yo onderbroken hersteltest

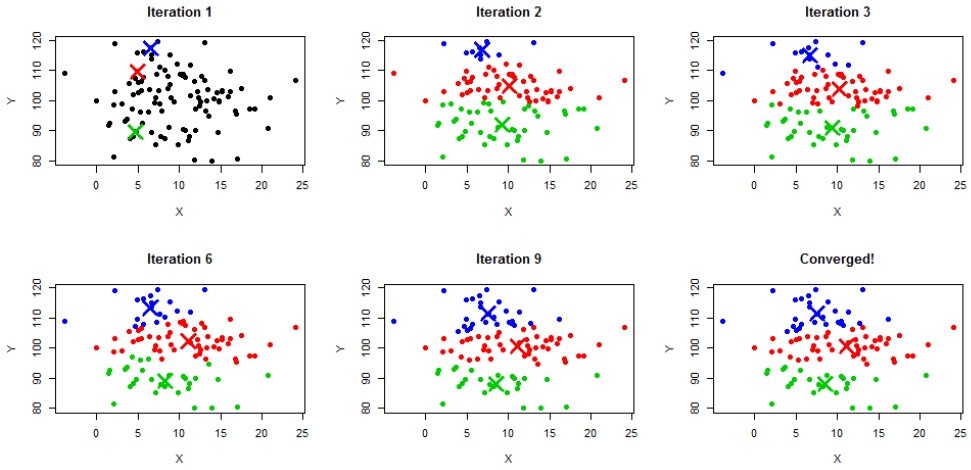
Een maximale Yo-Yo onderbroken hersteltest (Yo-Yo OH) wordt uitgevoerd om aerobe gegevens te verzamelen. Deze test meet de totaal afgelegd afstand in meters en wordt aangeduid als *Afstand*.

De validiteit en betrouwbaarheid van de Yo-Yo OH-test zijn onderzocht als een maat voor de beoordeling van de aerobe capaciteit. Krustrup et al. (2003) suggereert bijvoorbeeld dat de Yo-Yo OH-testprestaties correleert met VO2-max bij elite voetballers.

2.5 Unsupervised machine learning modellen

K-means

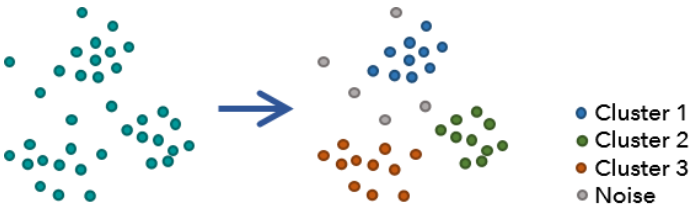
K-means is een clustering model dat in de centroid-gebaseerd categorie valt en werkt door *k* initiële centroids willekeurig te selecteren, waarbij *k* het aantal te vormen clusters vertegenwoordigt. Het algoritme berekent voor elk datapunt de Euclidische afstand tussen het datapunt en elk van de *k* centroids. Het datapunt wordt toegewezen aan het cluster met de dichtstbijzijnde centroid. Vervolgens herberekent het model de centroid van elke cluster als het gemiddelde van alle datapunten die aan dat cluster zijn toegewezen. Het proces wordt herhaald tot dat de centroids niet langer significant veranderen of een maximum aantal iteraties is bereikt.



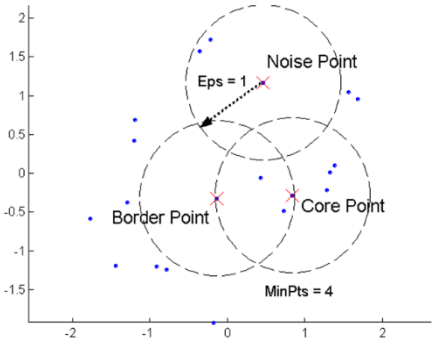
Figuur 1. Visualisatie van de K-means iteratie bij het vinden van de clusters(K = 3).

DBSCAN

DBSCAN is een clusteringmodel dat behoort tot de categorie van dichtheidsgebaseerde clusteringalgoritmen. Het algoritme maakt gebruik van twee belangrijke parameters: epsilon (ε) en minPts. ε is een afstandsdrempel die de grootte van de regio rond elk datapunt bepaalt, terwijl minPts het minimumaantal datapunten vertegenwoordigt dat aanwezig moet zijn in de ε-buurt van een datapunt om het als een kernpunt te beschouwen. Het algoritme identificeert eerst de kernpunten door te zoeken naar punten die minimaal minPts punten binnen hun ε-buurt hebben. Vervolgens voegt het punten toe aan een cluster als ze zich binnen de ε-afstand van een kernpunt bevinden. Hierdoor ontstaan clusters die worden gevormd door de verbondenheid van kernpunten en de grenspunten die bij de kernpunten horen. Niet-toegewezen punten die geen deel uitmaken van een cluster worden beschouwd als ruis.



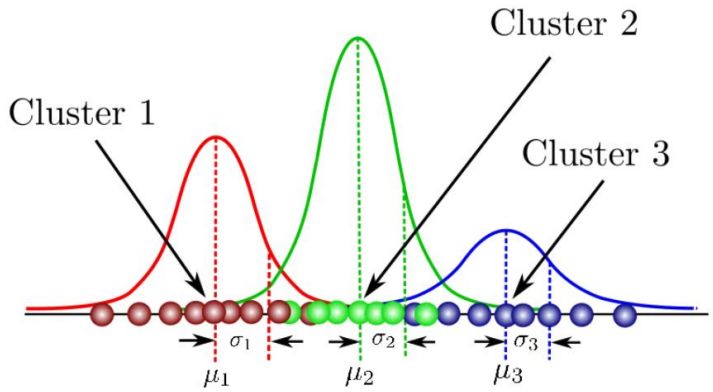
Figuur 2. Visualisatie van datapunten voor en na het gebruik van DBSCAN.



Figuur 3. Visualisatie van de kern- grens- en ruispunten van DBSCAN

GMM

GMM is een probabilistisch clusteringmodel dat gebruikmaakt van een combinatie van Gauss-verdelingen(normaal-verdelingen) om gegevens te modelleren. In een GMM wordt aangenomen dat elk datapunt een lineaire combinatie is van meerdere Gauss-verdelingen, die ook wel componenten worden genoemd. Elke component wordt gekarakteriseerd door zijn gemiddelde en standaarddeviatie, die worden gebruikt om de kansdichtheidsfunctie te berekenen voor die component. De verhoudingen tussen de componenten worden bepaald door gewichten, die aangeven hoeveel elk van de componenten bijdraagt aan het model. Deze gewichten bepalen de relatieve bijdrage van elke component aan de totale waarschijnlijkheid van een datapunt.



Figuur 4. Visualisatie van GMM. In dit voorbeeld zijn er 3 clusters die elk hun eigen Gauss-verdeling.

2.6 Gegevensverwerking

De dataset is opgebouwd door middel van verschillende stappen. Allereerst zijn de beste resultaten per seizoen, per speler en per test berekend. Voor elk van de teams, U19, Jong Ajax en Ajax 1, zijn aparte z-scores berekend om spelers te groeperen op basis van hun individuele prestaties in verhouding tot het gemiddelde van hun teamgenoten. Z-score zijn gebruikt om de dataset te standaardiseren, waardoor de schaal van de metingen gelijk wordt getrokken. Dit zorgt ervoor dat variabelen met grotere bereiken niet de overhand krijgen in het clustering proces en maakt het makkelijk om de clusters te interpreteren en vergelijken. Het toevoegen van een min-teken voor de *sprint 30m*- en *Agility*-tijd zorgt ervoor dat een snellere tijd als beter wordt beschouwd. Het gemiddelde en de standaarddeviatie van de leeftijd zijn berekend op basis van de leeftijden van alle afzonderlijke testmomenten.

Voor het opbouwen van de dataset zijn verschillende scenario's gevolgd, met de volgende resultaten: (1) Voor spelers die gedurende 2 of 3 seizoenen in hetzelfde team zijn gebleven, zijn de beste resultaten in z-scores per seizoen gebruikt om het gemiddelde te berekenen. (2) In situaties waarin een speler tijdens één seizoen een deel van de tests heeft uitgevoerd en in een volgend seizoen de resterende tests, zijn de resultaten samengevoegd. (3) Als een speler gedurende de seizoenen van team is veranderd, kunnen er meerdere datapunten, van dezelfde speler, met verschillend berekende z-scores ontstaan, afhankelijk van toewijzing aan verschillende teams.

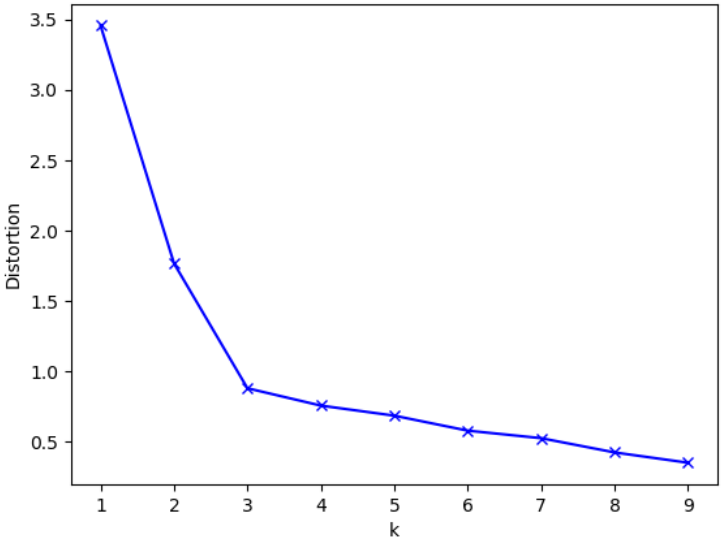
2.7 Voorlopige analyse:

In de voorlopige analyse zijn verschillende statistische methoden toegepast om de dataset te verkennen. Ten eerste is er gekeken of de data een normaalverdeling volgt. De uitkomsten van deze test worden meegenomen in de verdere statistische analyse.

Daarnaast zijn correlatiecoëfficiënten berekend tussen alle testparameters om mogelijke verbanden tussen deze parameters te onderzoeken. Om multicollineariteit te voorkomen, wordt één van de twee testparameters die een coëfficiënt hoger dan 0.7 vertonen, verwijderd uit het onderzoek (Ratner, 2009).

Om een visueel inzicht te verkrijgen in de clusters van professionele voetbalspelers, is een Principal Component Analysis (PCA) uitgevoerd. PCA wordt gebruikt om de oorspronkelijke dataset te transformeren naar een kleiner aantal nieuwe variabelen, bekend als principale components (PC), die het grootste deel van de informatie van de oorspronkelijke gegevens behouden. Deze PC's kunnen vervolgens gebruikt worden om tweedimensionale grafieken van de clusters te genereren die de meeste informatie weergeven. Om een beter begrip te krijgen van de PC's, wordt er gekeken naar de correlatie tussen de PC's en de verschillende testparameters. De resultaten van deze analyse kunnen helpen bij het begrijpen van de gevormde clusters. Hierdoor kan een verband worden gelegd tussen de clusters en de PC's. Dit biedt inzicht in welke testvariabelen de meeste invloed hebben op de vorming van de clusters en welke clusters goed of slecht presteren op de variabelen die correleren met de PC's.

Als laatste is de *elbow* methode gebruikt voor het bepalen van het optimale aantal clusters. Bij elke iteratie (k = i) berekent deze methode de som van de gekwadrateerde afstand tussen een datapunt en het toegewezen clustercentrum (Van Der Zwaard et al., 2019). Een *elbow* grafiek zoals getoond in figuur 5 wordt gemaakt om de visuele verandering van richting, de *elbow*, weer te geven en het aantal clusters te bepalen. In geval van figuur 5 is dat k = 3. De uitkomsten van de *elbow* methode worden gebruikt als het optimale aantal clusters voor alle modellen. Alle analyses en berekeningen zijn uitgevoerd met behulp van een zelfgemaakt Python script (Python 3.9.7, Python Software Foundation).



Figuur 5. Voorbeeld van een elbow plot, waar k = 3 een duidelijke elbow laat zien

2.8 Cluster analyse

Na het uitvoeren van de drie clustermodellen worden de clusters geanalyseerd met behulp van een statistische analyse. Voor de testparameters die een normale verdeling volgen, zijn One-way ANOVA-tests met een significantieniveau van $p<0.05$ en post hoc Tukey testen met een Benferroni correctie uitgevoerd. Voor de testparameters die niet normaal verdeeld zijn, zijn de Kruskal-Wallis-test en de Mann-Whitney U-test met een Bonferroni correctie gebruikt. Deze tests zijn toegepast om het verschil in fysieke capaciteiten tussen de clusters van onderzoek te bepalen.

Daarnaast zijn er interne validatietests uitgevoerd om de kwaliteit van de modellen te beoordelen. Hierbij worden de SI en de DI berekend om de modellen te beoordelen op basis van de gevormde clusters. De SI en de DI worden gebruikt als maatstaven voor de kwaliteit van clustering. De SI beoordeelt de gelijkenis van een datapunt met zijn eigen cluster ten opzichte van andere clusters (Rousseeuw, P. J. 1987), terwijl de DI de verhouding tussen de inter-clusterafstand en de intra-clusterafstand meet (Dunn, J. C. 1973). De uitkomst van de SI kan varieren tussen -1 en 1. Hierbij geldt dat een score dichtbij + 1 betekent dat datapunten correct zijn geclusterd, een score dichtbij 0 betekent dat datapunten ook bij een andere cluster kunnen behoren, een score van -1 betekent dat datapunten in de verkeerde cluster zijn ingedeeld (Shahapure, K. S., & Nicholas, C. 2020). De uitkomsten van de DI kunnen variëren tussen 0 en oneindig. Een score onder de 1 geeft aan dat er een zwakke neiging is om clusters te vormen, terwijl een score boven de 1 duidt op een sterke neiging om clusters te vormen (Rivera-Borroto et al., 2012).

Ook worden de clusters visueel geïnspecteerd om een beter beeld te krijgen van de clusters en de verschillen tussen de modellen. Voor de visualisatie van de clusters op een tweedimensionaal vlak wordt een spreidingsdiagram gebruikt. Daarnaast worden boxplots gebruikt om de prestaties van de clusters per testvariabele visueel weer te geven. Alle analyses zijn uitgevoerd met behulp van een zelf ontwikkeld script in Python (Python 3.9.7, Python Software Foundation).

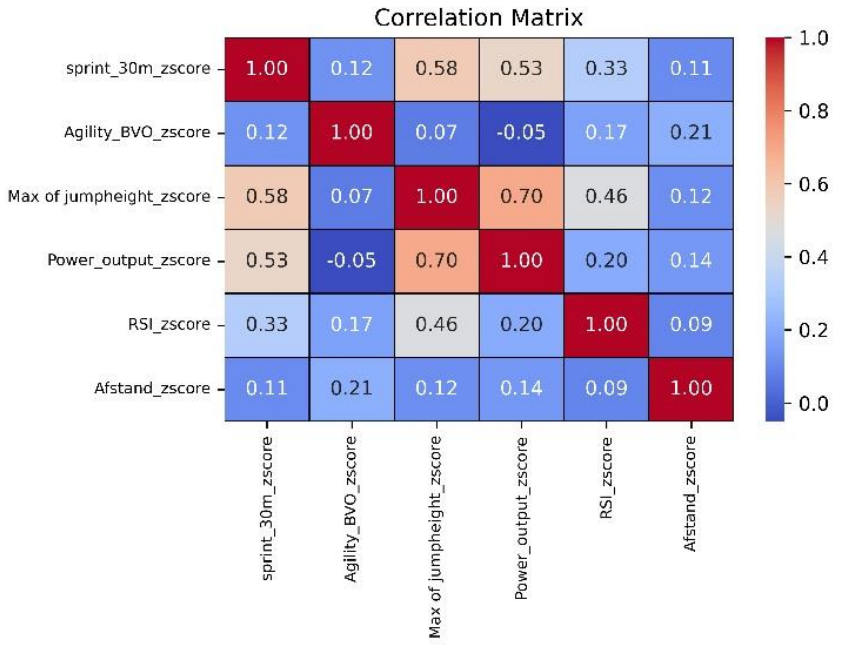
3 Resultaten

41 professionele voetbalspelers (leeftijd, 17.7 ± 1.5) zijn verdeeld in drie clusters en gevisualiseerd in een tweedimensionaal vlak met behulp van PC 1 en PC 2 (Figuur 9, 11, 13). De verschillen tussen de clusters worden weergegeven met behulp van boxplots (Figuur 10, 12, 14) en beschrijvende statistiek (Tabel 1, 2 ,3). Interne validatie scores worden weergegeven in tabel 4.

3.1 Voorlopige resultaten

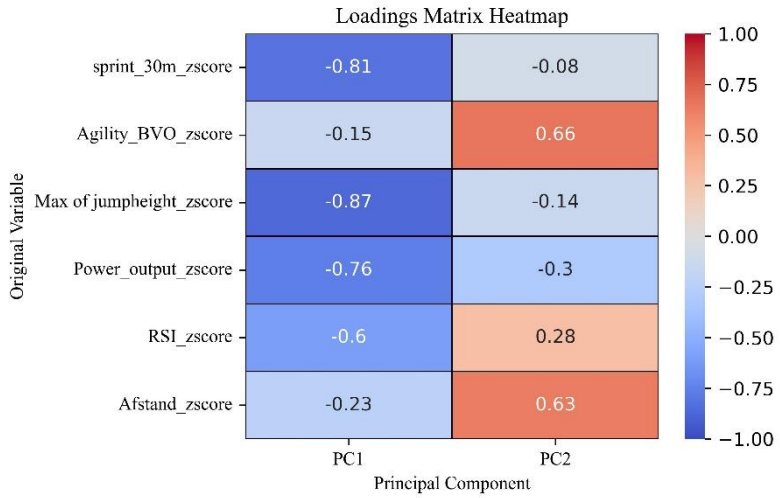
De voorlopige resultaten van het onderzoek omvatten verschillende aspecten. Allereerst is er een analyse uitgevoerd om de normaliteit van de testparameters te onderzoeken. Uit de resultaten bleek dat alle testparameters, behalve *Max of jumpheight* ($p = 0.0074$), een normale verdeling vertoonden ($p > 0.05$). Dit betekent dat voor alle testvariabelen, met uitzondering van *Max of jumpheight*, de One-way ANOVA-test en post hoc Tukey-test met Bonferroni-correctie zullen worden toegepast. Voor *Max of jumpheight* zal de Kruskal-Wallis-test en de Mann-Whitney U-test met Bonferroni-correctie worden gebruikt.

Vervolgens is er een correlatiematrix berekend om de onderlinge samenhang tussen de testparameters te onderzoeken. De resulterende correlatiecoëfficiënten toonden geen waarde hoger dan 0.7, wat aangeeft dat er geen sterke lineaire relaties zijn tussen de testparameters (figuur 6).



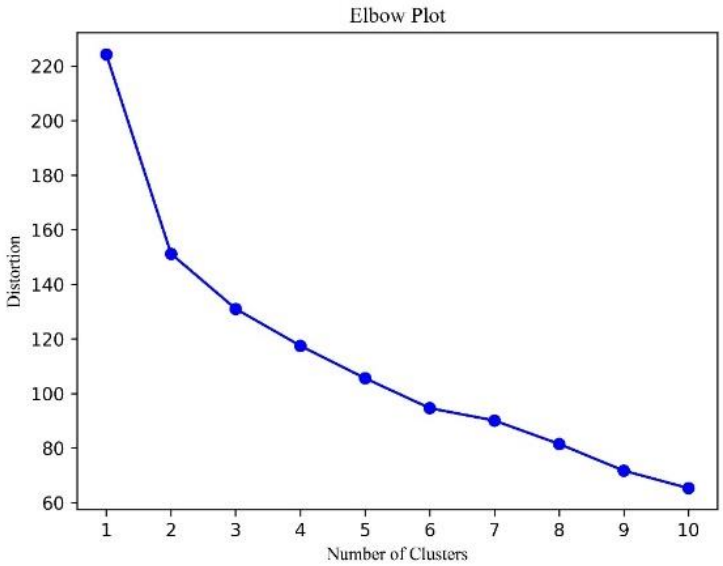
Figuur 6. Testvariabele in een correlatie matrix. Duidelijk correlatie tussen de "Max of jumpheight" en de "power output". De "Max of jumpheight" en de "power output", correleren ook matig met de "30 m sprint"

Daarna is er gekeken naar de correlatiematrix tussen de PC's en de testvariabele. PC 1 bleek sterk negatief te correleren met de *sprint 30m* (-0.81), *Max of jumpheight* (-0.87), *Power output* (-0.76) en matige correlatie te hebben met *RSI* (-0.6)(Ratner, 2009), waarbij negatieve waarden beter prestaties weerspiegelen. Om de leesbaarheid van de clustergrafieken te verbeteren, is ervoor gekozen om de uitkomst van PC 1 te vermenigvuldigen met -1, zodat een betere prestatie een positieve waarde vertegenwoordigt. PC 2 bleek gematigd te correleren met *Agility* (0.66) en *Afstand* (0.63) (Ratner, 2009) (figuur 7). PC 1 en PC 2 geven respectievelijk 43.18 % en 18.24 % van de informatie uit de dataset weer.



Figuur 7. Loadings Matrix heatmap. PC 1 correleer sterk negatief met sprint 30m, Max of jumpheight, Power output en matige met RSI. PC 2 correleer gematigd met Agility en Afstand.

Om het optimale aantal clusters te bepalen, is een *elbow* plot gebruikt. Hoewel er in figuur 8 bij *k* = 2 een kleine verandering van richting is, kan dit niet duidelijk als een *elbow* worden beschouwd zoals in figuur 5. Daarom zijn de modellen geëvalueerd met zowel *k* = 2 en *k* = 3. De resultaten toonden aan dat de clusters bij *k* = 3 zowel op PC 1 als PC 2 werden gescheiden, in tegenstelling tot alleen PC 1 bij *k* = 2 (bijlage 1, figuur 15, 17). Op basis hiervan is er gekozen voor *k* = 3.

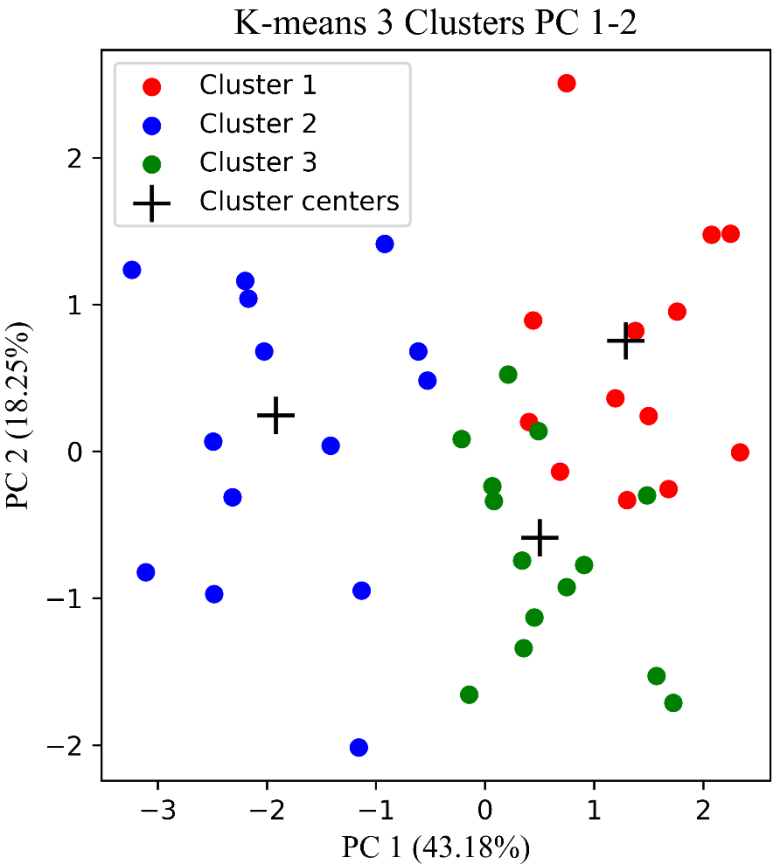


Figuur 8. Elbow plot of the distortion van datapunten gebaseerd op K = 1-10. Gevisualiseerd voor het kiezen van het optimale aantal clusters.

3.2 Cluster resultaten

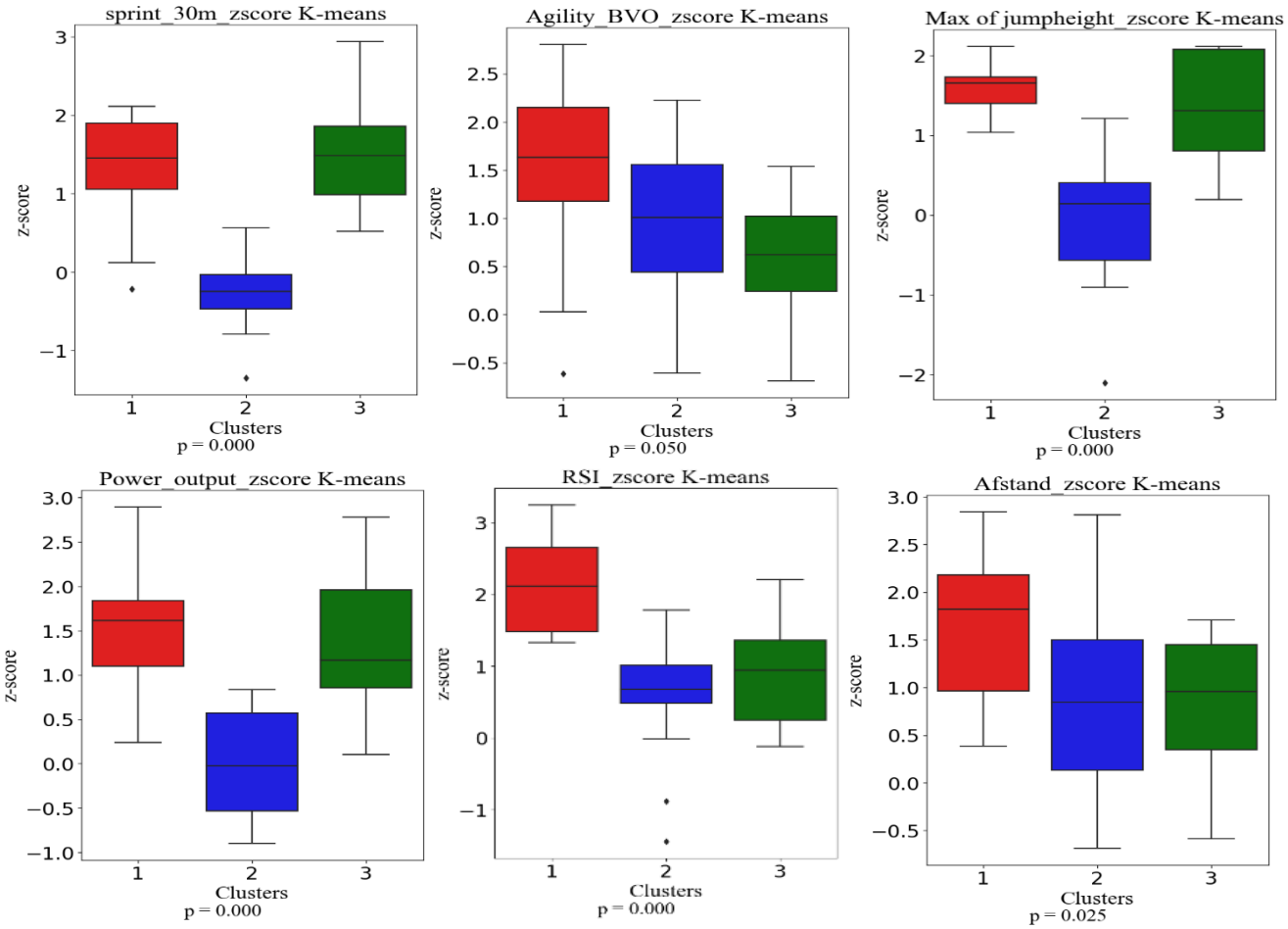
K-means

De resultaten van het K-means algoritme zijn gevisualiseerd in figuur 9. In dit figuur worden drie clusters waargenomen, elk met hun respectieve clustercentrum. Alle clusters hebben min of meer een gelijk aantal datapunten. Er is een duidelijke scheiding zichtbaar op de X-as tussen cluster 2 (blauw, 14 spelers) enerzijds en cluster 1 (rood, 13 spelers) en cluster 3 (groen, 14 spelers) anderzijds. Op de Y-as is er een minder uitgesproken scheiding tussen clusters 1 en 3 waar te nemen.



Figuur 9. K-means Cluster gevisualiseerd in 2D vlak met de principal components 1 en 2. Cluster 1 (rood) heeft 13 datapunten. Cluster 2 (blauw) heeft 14 datapunten. Cluster 3 (groen) heeft 14 datapunten.

Uit de analyse blijkt dat er statistisch significante verschillen zijn tussen de clusters in het K-means clustering model voor alle testparameters ($p < 0.05$). In figuur 10 worden de p-waarden en prestaties van alle testvariabelen weergegeven in boxplots.



Figuur 10. Boxplots van de testparameters en de k-means clusters. p-waardes zijn berekent met de one-way ANOVA bij een normale verdeling. Voor Max of jumpheight is de Kruskas-Wallis gebruikt.

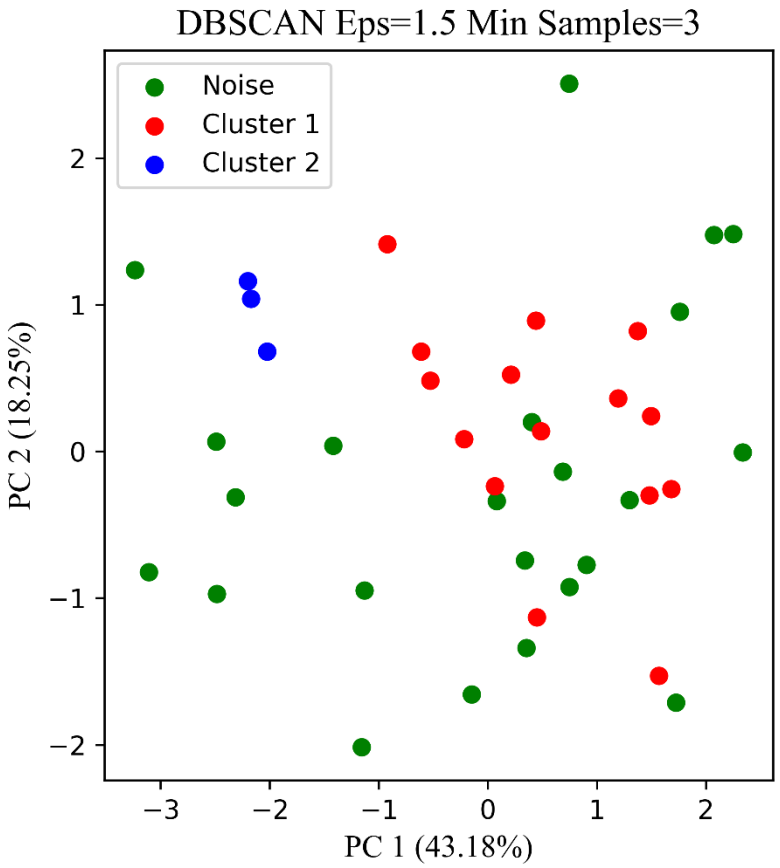
De post hoc testen (tabel 1) laten zien dat Cluster 1 significant betere prestaties vertoont dan Cluster 2 op alle variabelen, behalve op de variabele *Agility*, waar geen significant verschil werd waargenomen ($p = 0.2503$). Daarnaast presteert Cluster 1 significant beter dan Cluster 3 op de variabelen *RSI* en *Afstand* ($p < 0.001$). Cluster 2 vertoont significant slechtere prestaties dan Cluster 3 op de variabelen *30-meter sprint*, *Max of jumpheight* en *Power output* ($p < 0.001$).

Tabel 1. Post hoc p-waardes voor K-means. Tukey test voor alle testvariabele met een normaal verdeling. Voor Max of jumpheight is de Mann-Whitney U-test gebruikt.

K-means	1 vs. 2	1 vs. 3	2 vs. 3
		p-waarde	
30- meters sprint	<0.001	0.7747	<0.001
Agility	0.2503	0.0416	0.6322
Max of jumpheight	<0.001	0.4816	<0.001
Power output	<0.001	0.824	<0.001
RSI	<0.001	<0.001	0.5262
Afstand	0.0488	0.0414	0.977

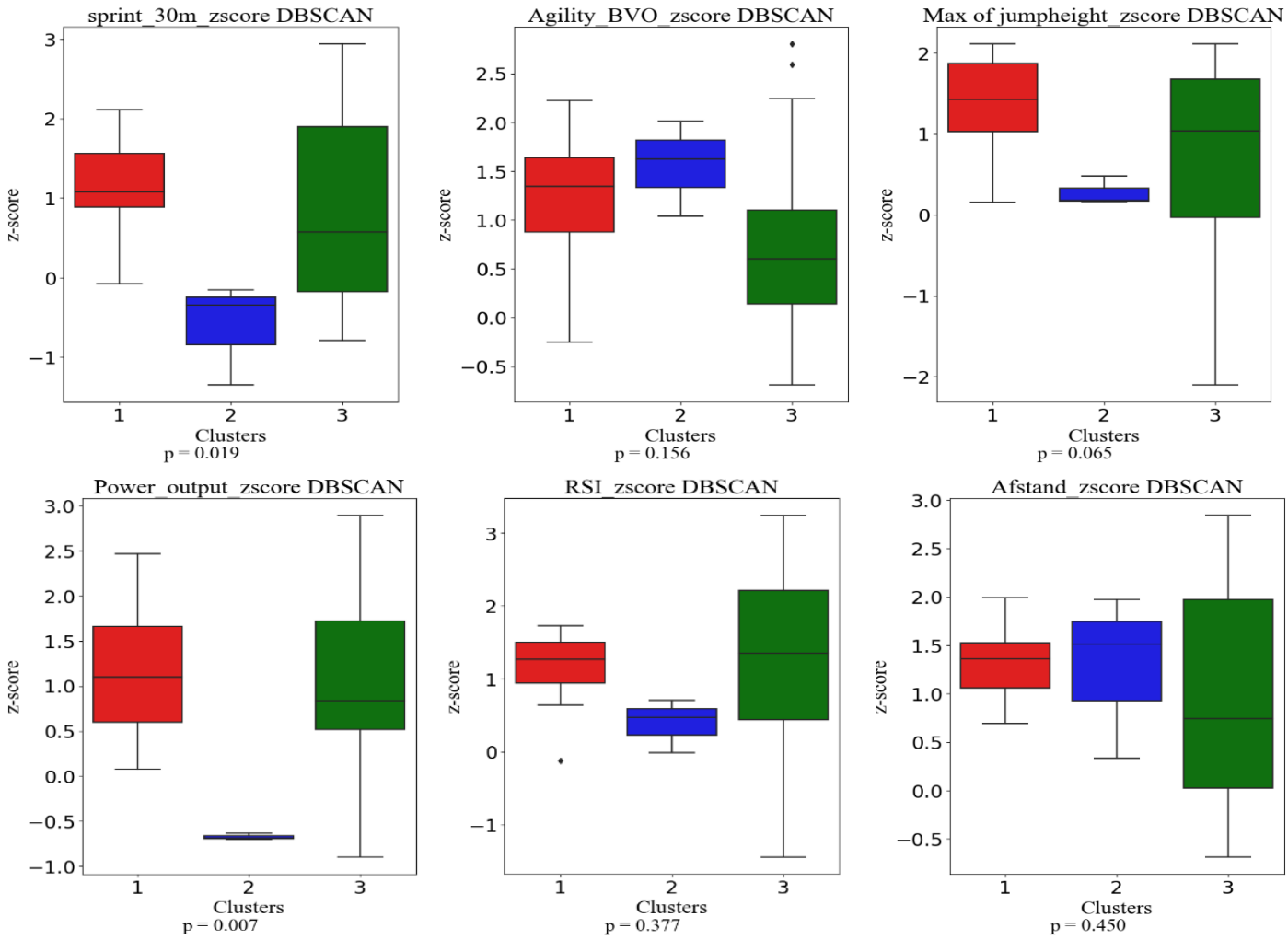
DBSCAN

De DBSCAN-clusters zijn weergegeven in Figuur 11, waarbij de formatie van de clusters zichtbaar is. DBSCAN laat grote verschillen zien tussen het aantal spelers in de clusters. Cluster 1 bevat slechts drie spelers, terwijl cluster 2 er 15 spelers bevat. Noise, bestaande uit datapunten die niet aan een specifiek cluster zijn toegewezen, bevat 23 datapunten en is verspreid over beide assen. De clusters 1 en 2 worden voornamelijk van elkaar gescheiden op de x-as.



Figuur 11. DBSCAN Cluster gevisualiseerd in 2D vlak met de principal components 1 en 2. Cluster 1 (blauw) heeft 15 datapunten. Cluster 2 (groen) heeft 15 datapunten. De Noise groep (zwart) heeft 23 datapunten

Bij DBSCAN-clustering zijn dezelfde statistische tests uitgevoerd als bij de K-means clustering. Uit de resultaten bleek dat er significante verschillen waren voor de variabelen *sprint 30m* en *Power output* ($p < 0.05$). In figuur 12 worden de p-waarden voor alle testvariabelen weergegeven, samen met de prestaties van elke cluster per testvariabele.



Figuur 12. Boxplots van de testparameters en de DBSCAN clusters. p-waardes zijn berekent met de one-way ANOVA bij een normale verdeling. Voor "Max of jumpheight" is de Kruskas-Wallis gebruikt.

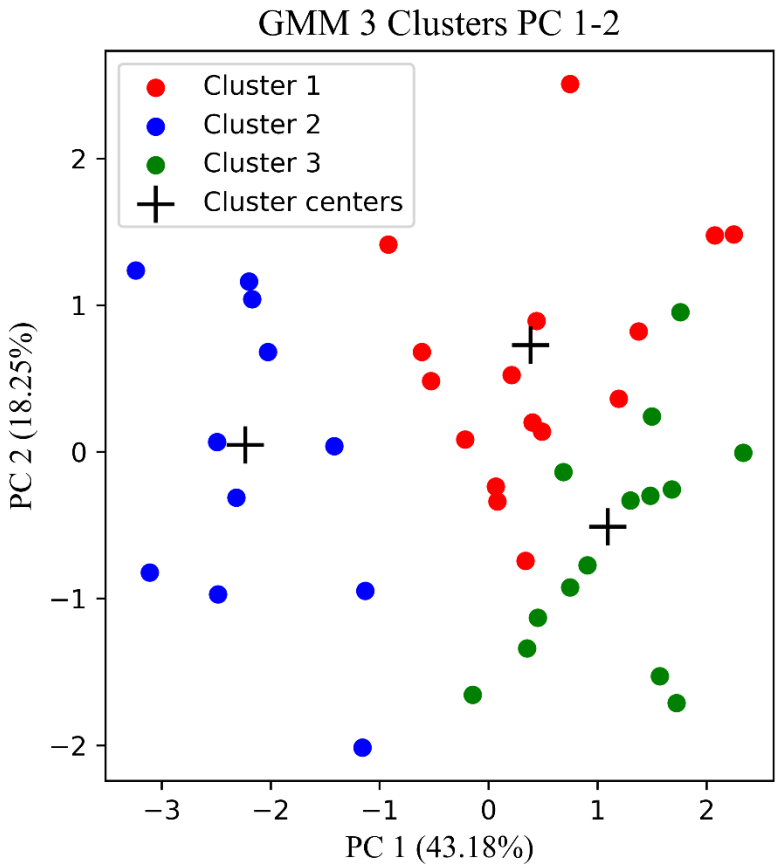
Post-hoc tests voor deze variabelen onthulden specifieke clusterprestaties en worden beschreven in tabel 2. Voor de vergelijking tussen Cluster 1 en 2 presteerde Cluster 1 significant beter op zowel de *sprint 30m* als de *Power output* ($p < 0.05$). Daarentegen werden er geen significante verschillen waargenomen tussen Cluster 1 en de *Noise* voor deze variabelen ($p > 0.05$). Verder presteerde Cluster 2 significant slechter dan de *Noise* op zowel de *sprint 30m* als de *Power output* ($p < 0.05$).

Tabel 2. p-waardes DBSCAN voor vergelijkingen tussen de verschillende clusters per testvariabele

DBSCAN	1 vs. 2	1 vs. 3	2 vs. 3
		p-waarde	
30- meters sprint	0.0141	0.5702	0.043
Agility	-	-	-
Max of jumpheight	-	-	-
Power output	0.0066	0.9534	0.0077
RSI	-	-	-
Afstand	-	-	-

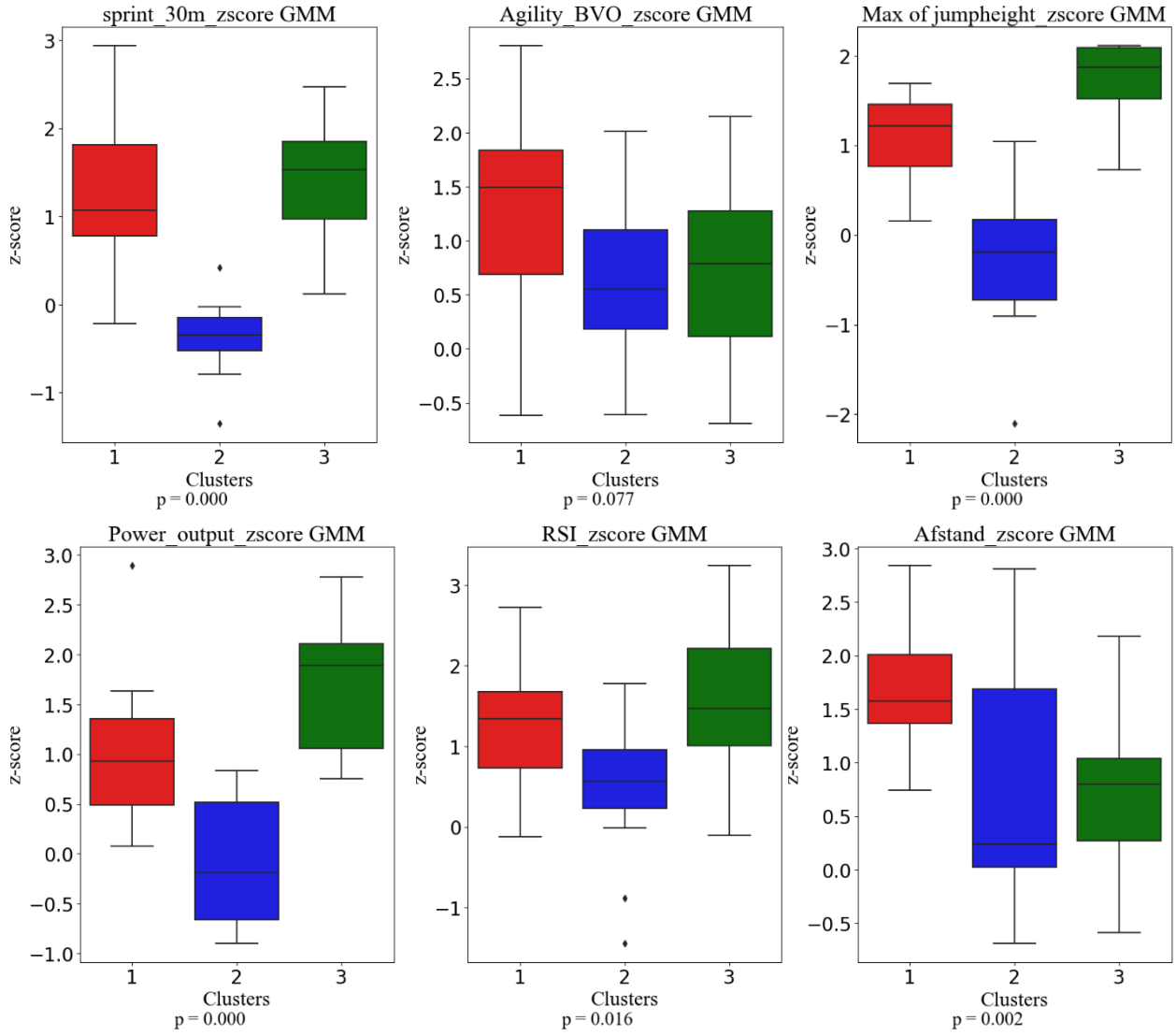
GMM

De resultaten van de GMM-clustering worden gepresenteerd in Figuur 13, waarin de drie clusters zichtbaar zijn, samen met hun respectieve clustercentrum. GMM laat grotere verschillen zien tussen het aantal spelers per clusters. Cluster 2 (blauw, 11 spelers) wordt gescheiden van cluster 1 (rood, 16 spelers) en cluster 3 (groen, 14 spelers) op de X-as. Op de Y-as is er een scheiding zichtbaar tussen clusters 1 en 3.



Figuur 13. GMM cluster gevisualiseerd in 2D vlak met de principal components 1 en 2. Cluster 1 (rood) heeft 16 datapunten. Cluster 2 (blauw) heeft 11 datapunten. Cluster 3 (groen) heeft 14 datapunten.

Voor een gedetailleerde data-analyse worden boxplots weergegeven in Figuur 14, samen met de resultaten van de statistische analyse. Uit de statistische analyse van het GMM blijkt dat er significante verschillen zijn tussen de clusters op de verschillende testvariabele ($p < 0.05$), met uitzondering van *Agility* ($p < 0.077$).



Figuur 14. Boxplots van de testparameters en de GMM clusters. p-waardes zijn berekent met de one-way ANOVA bij een normale verdeling. Voor Max of jumpheight is de Kruskas-Wallis gebruikt.

De beschrijvende statistieken van de post-hoc testen zijn te vinden in tabel 3. De resultaten van de post-hoc testen laten zien dat Cluster 1 significant beter presteert dan Cluster 2 op de meeste variabelen, met uitzondering van *RSI* waarvoor geen significant verschil werd gevonden ($p = 0.084$). Daarnaast presteert Cluster 1 significant beter dan Cluster 3 op de variabele *Afstand* en significant slechter dan Cluster 3 op *Max of jumpheight* en *Power output* ($p < 0.05$). Cluster 2 presteert significant beter dan Cluster 3 op alle variabelen, behalve *Afstand* ($p = 0.9995$).

Tabel 3. p-waardes GMM voor vergelijkingen tussen de verschillende clusters per testvariabele

GMM	1 vs. 2	1 vs. 3	2 vs. 3
		p-waarde	
30- meters sprint	<0.001	0.7256	<0.001
Agility	-	-	-
Max of jumpheight	<0.001	0.002	<0.001
Power output	<0.001	0.0146	<0.001
RSI	0.084	0.6334	0.0139
Afstand	0.007	0.005	0.9995

3.3 Interne validatie resultaten

Uit de interne validatietests blijkt dat de K-means (0.16410) en GMM (0.15436) vergelijkbare waarde scores behalen op de SI, terwijl het DBSCAN (-0.0054) een lagere score heeft dan de andere twee modellen.

Bij de DI behaalt daarentegen het DBSCAN-model de hoogste score (0.25923), gevolgd door GMM (0.17300) en vervolgens K-means (0.10971).

Tabel 4. Uitkomsten van de interne validatie testen

Model	Silhouette index	Dunn index
K-means	0.16410	0.10971
DBSCAN	-0.0054	0.25923
GMM	0.15436	0.17300

4 Discussie

Uit eerdere onderzoeken is gebleken dat er een verschil bestaat in de interne belasting van spelers bij dezelfde externe belasting. Dit kan worden beïnvloed door de anaerobe en aerobe capaciteiten van voetbalspelers. Het verschil in trainingsbelasting heeft een directe invloed op de hersteltijd van spelers, wat de noodzaak aantoont om spelers te clusteren op basis van anaerobe en aerobe testgegevens. Deze clusters kunnen gebruikt worden bij het optimaliseren van de periodisering. Periodisering heeft als doel om trainingen te structureren ter voorkoming van overbelasting, bevordering van optimale prestaties en effectief herstel mogelijk te maken. Het doel van dit onderzoek was om het model te identificeren dat de beste prestaties levert bij het identificeren van subgroepen in een dataset met anaerobe en aerobe testgegevens van professionele voetbalspelers. Hiervoor zijn drie modellen gebruikt: k-means, DBSCAN en GMM. De prestaties van de modellen werden geëvalueerd aan de hand van twee interne validatietests (SI en DI), een visuele inspectie van de clusters en het toetsen van statistische verschillen tussen de clusters.

4.1 DBSCAN

DBSCAN bleek duidelijk ongeschikt voor het identificeren van subgroepen in de gebruikte dataset. Het basis idee achter DBSCAN is dat clusters gebieden zijn met een hoge dichtheid, terwijl datapunten in gebieden met een lage dichtheid als ruis worden beschouwd. Tijdens de preliminaire analyse zijn herhaaldelijk aanpassingen gemaakt aan de epsilon en het minimale aantal datapunten, dit resulteerde in twee mogelijke uitkomsten. In het ene scenario werden slechts twee clusters gevonden, samen met een groot aantal ruispunten, zoals te zien is in Figuur 11. In het andere scenario ontstond er één grote cluster met slechts een klein aantal ruispunten (figuur 16). Een mogelijke oorzaak hiervan is de hoge fysieke homogeniteit onder professionele voetballers (Silvestre et al., 2006). Deze homogeniteit kan ertoe leiden dat er geen gebieden met een verandering in dichtheid zijn die clusters van elkaar kunnen scheiden. Bovendien werden er weinig significante verschillen gevonden bij DBSCAN. Op basis van deze bevinding kan geconcludeerd worden dat DBSCAN niet geschikt is voor het identificeren van subgroepen in deze dataset. Om die reden zal DBSCAN niet verder worden meegenomen bij het vergelijken van de modellen.

4.2 K-means en GMM

De interne validatiescores van zowel K-means als GMM zijn laag voor de SI (0.16410, 0.15436 respectievelijk) en de DI (0.10971, 0.17300 respectievelijk). Een SI waarde dicht bij nul geeft aan dat clusters overlappen (Shahapure & Nicholas, 2020). Op dezelfde manier duidt een waarde van de DI die dichter bij 0 ligt op een slechte clusteringoplossing (Rivera-Borroto et al., 2012). Hoewel deze scores kunnen worden gebruikt om de modellen onderling te vergelijken, zijn de scores over het algemeen laag en bevinden ze zich dicht bij nul. De lage scores van de clusters kunnen waarschijnlijk worden toegeschreven aan de homogeniteit van de professionele voetballers, zoals eerder vermeld. Hierdoor zijn de modellen niet in staat om compacte clusters te vinden die ver van elkaar verwijderd zijn, wat resulteert in lage scores bij de interne validatietests.

Bij het onderling vergelijken van de interne validatiescores kan worden gesteld dat GMM beter presteert dan K-means. Echter, de verschillen in de interne validatiescores tussen de twee modellen zijn zeer klein en bieden geen duidelijk inzicht in welk model beter presteert dan het andere. Op basis van de resultaten van de interne validatietests kan geconcludeerd worden dat er geen model is dat beter presteert op de interne validatietesten en daardoor superieur is in het identificeren van optimale clusters.

4.3 Clusters

Op het gebied van visuele waarneming vertonen de clusters van k-means en GMM veel overeenkomsten. De verschillen tussen de twee modellen worden voornamelijk waargenomen in het aantal datapunten per cluster en de mate van scheiding tussen de clusters. Visueel gezien lijken de clusters van GMM beter van elkaar gescheiden te zijn in een tweedimensionale ruimte, waarbij er geen overlappende datapunten van verschillende clusters voorkomen, in tegenstelling tot K-means. Het is echter belangrijk om te benadrukken dat PC 1 en PC 2 samen slechts 61.43% van de variatie in de gegevens vertegenwoordigen, waardoor dit een vertekend beeld kan geven.

Bij het verder interpreteren van de clusters kunnen we concluderen dat zowel k-means als GMM vergelijkbare clusteringresultaten tonen. Cluster 1 behaalt de beste prestaties op zowel PC 1, die sterk correleert met anaerobe testen (*sprint 30m, Max of jumpheight*, *Power output*, *RSI*), als op PC 2, die matig tot sterk correleert met *Agility* en *Afstand*. Op basis hiervan kan geconcludeerd worden dat cluster 1 gedefinieerd kan worden als de topgroep (TG) met zowel goede anaerobe als aerobe prestaties. Cluster 2 presteert in beide gevallen slechter op PC 1 dan de andere clusters en vertoont vergelijkbare prestaties op PC 2 als cluster 1 en cluster 3 samen. Cluster 2 kan beschouwd worden als de niet-anaerobe groep (NAG) met slechtere prestaties op de anaerobe testen en vergelijkbare prestaties ten opzichte van cluster 1 en cluster 3 samen op aerobisch vlak. Cluster 3 laat een betere anaerobe prestatie zien dan cluster 2 en een mindere prestatie dan cluster 1 op aeroob vlak. Cluster 3 kan gekarakteriseerd worden als de anaerobe groep (AG) met goede prestaties op anaeroob vlak en mindere prestaties op aeroob vlak. Deze studie implementeerde k-means en GMM-algoritmen om de fysieke capaciteiten van professionele voetballers te analyseren en was succesvol in het identificeren van drie homogene subgroepen. Dit onderzoek vertoont overeenkomsten met eerdere studies van Van der Zwaard et al. (2019) en Soto-Valero (2017), waarbij respectievelijk k-means en GMM werden toegepast. Van der Zwaard et al. gebruikte k-means om professionele wielrenners op basis van antropometrische gegevens in te delen in subgroepen. Deze methode resulteerde in de identificatie van drie clusters die gerelateerd waren aan lichaamstype en specialisatie binnen het wielrennen. Soto-Valero (2017) paste GMM toe om voetbalspelers te groeperen op basis van gegevens uit een EA Sports FIFA-videogame. Deze methode leidde tot de identificatie van vier clusters die correspondeerden met de veldposities: aanvallers, middenvelders, verdedigers en doelmannen.

Bij het vergelijken van de statistisch significante verschillen tussen de clusters lijkt K-means betere resultaten te behalen. Alle zes testvariabelen vertonen significante verschillen tussen de clusters bij K-means. Bij GMM is alleen de variabele *Agility* niet significant verschillend ($p = 0.077$). Ondanks dit verschil in statistische toetsen, vertonen beide modellen hetzelfde aantal significante verschillen bij de post-hoc testen, namelijk 11. Op basis van deze bevindingen kan er geen definitieve conclusie worden getrokken over welk model beter presteert.

4.4 Beperkingen

Dit onderzoek heeft enkele beperkingen met betrekking tot de gebruikte dataset. Ten eerste is er een beperking met betrekking tot de samenstelling van de dataset. De dataset bevat vijf anaerobe testen en slechts één aerobe test, wat kan leiden tot een vertekend beeld als gevolg van de oververtegenwoordiging van anaerobe testen. Daarnaast kunnen er meerdere datapunten van dezelfde speler in de database voorkomen, met verschillende berekende z-scores, doordat alleen de z-scores zijn samengevoegd die zijn berekend voor dezelfde teams. Dit kan de homogeniteit van de dataset mogelijk verder hebben versterken. Bovendien kan het combineren van testgegevens van verschillende seizoenen, waarbij een deel van de testen in seizoen 1 werd uitgevoerd en een ander

deel in seizoen 2, invloed hebben op de resultaten. Dit is met name relevant gezien de mogelijke aanzienlijke vooruitgang die jongere spelers in één jaar kunnen boeken (Slimani & Nikolaidis, 2018).

Een tweede beperking van dit onderzoek is dat de bevindingen afhankelijk zijn van de gebruikte dataset. Eerdere onderzoeken hebben aangetoond dat de prestaties van clusteringmodellen worden beïnvloed door de specifieke dataset die wordt gebruikt (Lukauskas & Ruzgas, 2023). Het toevoegen van extra variabele, vooral als deze geen anaerobe of aerobe gegevens bevatten, kan van invloed zijn op de prestaties van de modellen en de geïdentificeerde clusters.

Tijdens de voorlopige analyse werd waargenomen dat kleine verschillen in de dataset, zoals verschillende methoden voor het berekenen van z-scores of het samenvoegen van gegevens bij ontbrekende waarden of spelers die meerdere keren voorkomen als gevolg van teamwisselingen, weinig invloed hadden op de resultaten.

4.5 Aanbevelingen

Uit de bevindingen van deze studie volgen enkele aanbevelingen. (1) In dit onderzoek zijn fysiologische groepen van spelers samengesteld op basis van voornamelijk anaerobe fysieke testen. Om betere fysiologische profielen op te stellen, is het noodzakelijk om meer aerobe informatie toe te voegen aan de dataset, zodat er een betere balans ontstaat tussen anaerobe en aerobe testen. Hierbij zouden variabelen zoals 1 minuut hartslagherstel na afloop van de Yo-Yo-test en de VO2max kunnen worden opgenomen. (2) Voor vervolg onderzoek is K-means het meest geschikte model, omdat dit model al veelvuldig wordt gebruikt in ander wetenschappelijk onderzoek naar het clusteren van atleten. Hoewel andere centroid-gebaseerde modellen, zoals k-medoids of k-median, ook interessant kunnen zijn vanwege hun verschillende berekeningsmethoden voor de afstand tussen datapunten en de centroid. (3) Indien de elbow-methode bij vervolg onderzoek geen duidelijke uitkomst biedt, is het raadzaam om verder onderzoek te doen naar het aantal clusters, in plaats van de analyse te beperken tot slechts twee of drie clusters, zoals in dit onderzoek is gedaan. Door een breder scala aan clusteraantallen te verkennen, kan er meer inzicht worden verkregen in de dataset en de verschillen tussen spelers. (4) Vervolgens is het noodzakelijk om de gevonden clusters te valideren. Onderzoek toont aan dat er verschillen zijn in de interne trainingsbelasting die spelers ervaren bij een gelijke externe trainingsbelasting (Castagna et al., 2011; Impellizzeri et al., 2004). Om de gevonden clusters te valideren en de periodisering verder te optimaliseren op basis van verschillen in hersteltijd, is het belangrijk om te onderzoeken of deze verschillen ook bestaan tussen de geïdentificeerde clusters. (5) Ten slotte kan er toekomstig onderzoek worden uitgevoerd naar clustering, niet gebaseerd op fysieke capaciteiten, maar op basis van de interne belasting van spelers bij een vergelijkbare externe belasting. Hiervoor wordt trainingsdata verzameld, bestaande uit parameters zoals hartslag, lactaatniveaus en perceptie van inspanning, die kunnen worden gebruikt voor het clusteren van spelers. Deze clusters kunnen helpen bij het valideren van clusters die zijn geïdentificeerd op basis van fysieke capaciteiten of kunnen op zichzelf worden gebruikt om de periodisering te verbeteren.

4.6 Praktische relevantie

Het formeren van subgroepen biedt de mogelijkheid om spelers doelgerichte begeleiding te bieden in hun trainingsprogramma. Bijvoorbeeld, de NAG zou mogelijk baat kunnen hebben bij aanvullende anaerobe training om hun prestaties op dat gebied te verbeteren, terwijl de AG mogelijk zou kunnen profiteren van extra aerobe training. Indien vervolgonderzoek aantoont dat er verschillen bestaan in de interne belasting bij dezelfde externe belasting tussen de clusters, kunnen deze verschillen worden gebruikt om de periodisering verder te optimaliseren op basis van verschillen in hersteltijd.

5 Conclusie

Deze studie is de eerste die UMLC-modellen met elkaar vergelijkt bij het vinden van subgroepen op basis van anaerobe en aerobe gegevens bij professionele voetballers, die kunnen worden gebruikt voor een verdere individualisering van de periodisering. Het is gebleken dat DBSCAN niet goed in staat is om significante verschillen tussen de subgroepen te vinden, terwijl zowel K-means als GMM daarin wel slagen. Beide modellen tonen drie clusters met verschillen in de fysieke capaciteiten.

Dat gezegd hebbende zijn de verschillen tussen de modellen relatief klein, dus er moeten ook andere factoren in overweging worden genomen bij het bepalen van het best presterende model voor verder onderzoek. Aangezien K-means al uitgebreid is gebruikt in onderzoek binnen de sportwereld en het model gemakkelijk te begrijpen en te gebruiken is, zelfs voor niet-datawetenschappers, verdient K-means een sterke overweging voor verder onderzoek met betrekking tot fysiologische profiel.

6 Literatuurlijst

Abdulnassar, A., & Nair, L. R. (2023). Performance analysis of Kmeans with modified initial centroid selection algorithms and developed Kmeans9+ model. Measurement: Sensors, 25, 100666. https://doi.org/10.1016/j.measen.2023.100666

Akhtar, F., Li, J., Pei, Y., Xu, Y., Rajput, A., & Wang, Q. (2019). Optimal Features Subset Selection for Large for Gestational Age Classification Using GridSearch Based Recursive Feature Elimination with Cross-Validation Scheme. In *Lecture notes in electrical engineering*. Springer Science+Business Media. https://doi.org/10.1007/978-981-15-3250-4_8

Castagna, C., Impellizzeri, F. M., Chaouachi, A., Bordon, C., & Manzi, V. (2011). Effect of Training Intensity Distribution on Aerobic Fitness Variables in Elite Soccer Players: A Case Study. *Journal of Strength and Conditioning Research*, *25*(1), 66–71. https://doi.org/10.1519/jsc.0b013e3181fef3d3

Çakir-Atabek, H. (2014). Relationship between anaerobic power, vertical jump and aerobic performance in adolescent track and field athletes. *Journal of Physical Education and Sport*. https://doi.org/10.7752/jpes.2014.04100

Chamari, K., Hachana, Y., Ahmed, Y. B., Galy, O., Sghaier, F., Chatard, J.-C., Hue, O., Wisloff, U. (2004). Field and laboratory testing in young elite soccer players. *British Journal of Sports Medicine*, *38*(2), 191–196. https://doi.org/10.1136/bjsm.2002.004374

Das, N. R., Mukherjee, I., Patel, A. D., & Paul, G. (2023). An intelligent clustering framework for substitute recommendation and player selection. *The Journal of Supercomputing*. https://doi.org/10.1007/s11227-023-05314-z

Dunn, J. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of cybernetics*, *3*(3), 32–57. https://doi.org/10.1080/01969727308546046

Flanagan, E. P., Ebben, W. P., & Jensen, R. L. (2008). Reliability of the Reactive Strength Index and Time to Stabilization During Depth Jumps. *Journal of Strength and Conditioning Research*, *22*(5), 1677–1682. https://doi.org/10.1519/jsc.0b013e318182034b

Impellizzeri, F. M., Rampinini, E., Coutts, A. J., Sassi, A., & Marcora, S. M. (2004). Use of RPE-Based Training Load in Soccer. *Medicine and Science in Sports and Exercise*, *36*(6), 1042–1047. https://doi.org/10.1249/01.mss.0000128199.23901.2f

Krustrup, P., Mohr, M., Amstrup, T., Rysgaard, T., Johansen, J., Steensberg, A., Pedersen, P. U., & Bangsbo, J. (2003). *The Yo-Yo Intermittent Recovery Test: Physiological Response, Reliability, and Validity. Medicine & Science in Sports & Exercise, 35(4), 697–705.* https://doi.org/10.1249/01.mss.0000058441.94520.32

Lukauskas, M., & Ruzgas, T. (2022). Review and Comparative Analysis of Unsupervised Machine Learning Application in Health Care. In *Algorithms for intelligent systems* (pp. 751–759). Springer Nature. https://doi.org/10.1007/978-981-19-6004-8_56

Marković, G., Dizdar, D., Jukić, I., & Cardinale, M. (2004). Reliability and Factorial Validity of Squat and Countermovement Jump Tests. *Journal of Strength and Conditioning Research*, *18*(3), 551–555. https://doi.org/10.1519/00124278-200408000-00028

Nara, K., Kumar, P., Rathee, R., & Kumar, J. (2022). The compatibility of running-based anaerobic sprint test and Wingate anaerobic test: a systematic review and meta-analysis. *Pedagogy of Physical Culture and Sports*, *26*(2), 134–143. https://doi.org/10.15561/26649837.2022.0208

Nawrin, S., Rahman, M. R., & Akhter, S. (2017). Exploreing K-Means with Internal Validity Indexes for Data Clustering in Traffic Management System. International Journal of Advanced Computer Science and Applications, 8(3). https://doi.org/10.14569/ijacsa.2017.080337

Nedelec, M., McCall, A., Carling, C., Legall, F., Berthoin, S., & Dupont, G. (2012). Recovery in Soccer. *Sports Medicine*, *42*(12), 997–1015. https://doi.org/10.2165/11635270-000000000-00000

Rabbani, A., Wong, D. P., Clemente, F. M., & Kargarfard, M. (2021). Internal training load and fitness profile between adult team versus junior team soccer players. *Kinesiology*. https://doi.org/10.26582/k.53.1.8

Ratner, B. (2009). The correlation coefficient: Its values range between +1/−1, or do they? *Journal of Targeting, Measurement and Analysis for Marketing*, *17*(2), 139–142. https://doi.org/10.1057/jt.2009.5

Rein, R., & Memmert, D. (2016). Big data and tactical analysis in elite soccer: future challenges and opportunities for sports science. *SpringerPlus*, *5*(1). https://doi.org/10.1186/s40064-016-3108-2

Rivera-Borroto, O. M., Rabassa-Gutiérrez, M., Del Corazón Grau-Ábalo, R., Marrero-Ponce, Y., & López, R. (2012b). Dunn's index for cluster tendency assessment of pharmacological data sets. Canadian Journal of Physiology and Pharmacology, 90(4), 425–433. https://doi.org/10.1139/y2012-002

Rossi, A., Pappalardo, L., Cintia, P., Iaia, F. M., Fernández, J., & Medina, D. (2018). Effective injury forecasting in soccer with GPS training data and machine learning. *PLOS ONE*, *13*(7), e0201264. https://doi.org/10.1371/journal.pone.0201264

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

Shahapure, K. S., & Nicholas, C. (2020). *Cluster Quality Analysis Using Silhouette Score*. https://doi.org/10.1109/dsaa49011.2020.00096

Silvestre, R., West, C., Maresh, C. M., & Kraemer, W. J. (2006). Body Composition and Physical Performance in Men's Soccer: A Study of a National Collegiate Athletic Association Division I Team. *Journal of Strength and Conditioning Research*, *20*(1), 177. https://doi.org/10.1519/r-17715.1

Slimani, M., & Nikolaidis, P. T. (2018). Anthropometric and physiological characteristics of male soccer players according to their competitive level, playing position and age group: a systematic review. *Journal of Sports Medicine and Physical Fitness*, *59*(1). https://doi.org/10.23736/s0022-4707.17.07950-6

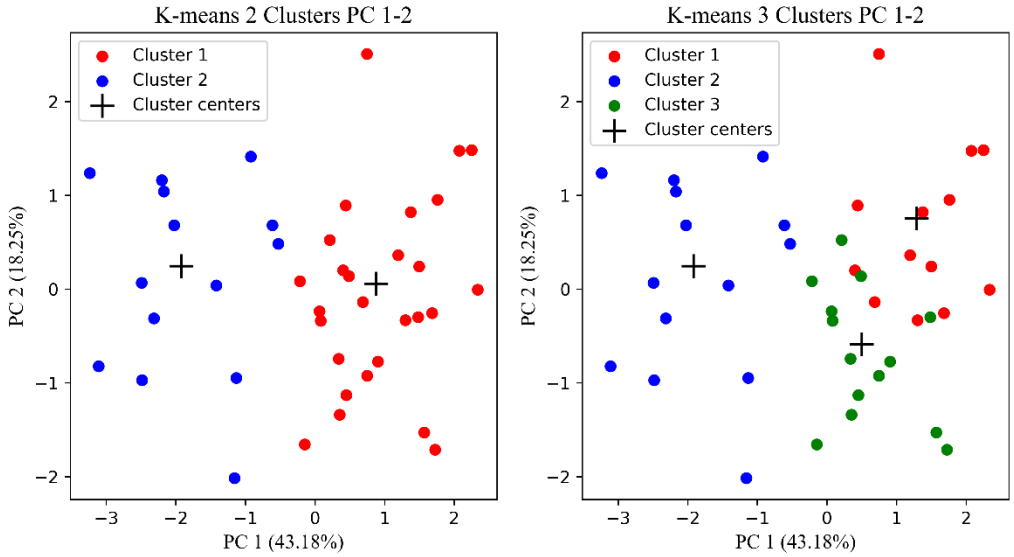
Soto-Valero, C. (2017). A Gaussian mixture clustering model for characterizing football players using the EA Sports' FIFA video game system. *Revista Internacional de Ciencias del Deporte*, *13*(49), 244–259. https://doi.org/10.5232/ricyde2017.04904

Svensson, M., & Drust, B. (2005). Testing soccer players. *Journal of Sports Sciences*, *23*(6), 601–618. https://doi.org/10.1080/02640410400021294

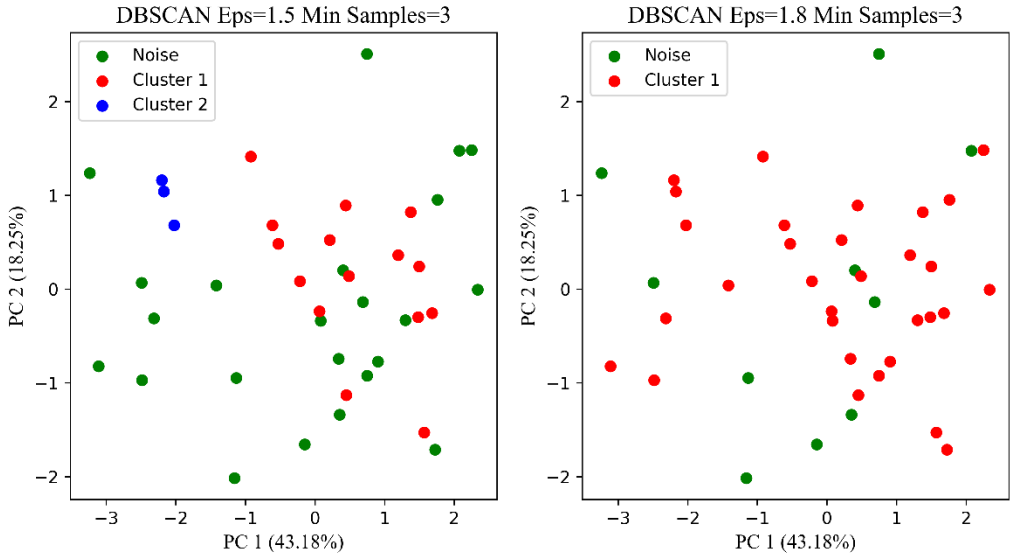
Van Der Zwaard, S., De Ruiter, C., Jaspers, R. T., & De Koning, J. J. (2019). Anthropometric Clusters of Competitive Cyclists and Their Sprint and Endurance Performance. *Frontiers in Physiology*, *10*. https://doi.org/10.3389/fphys.2019.01276

7 Bijlage

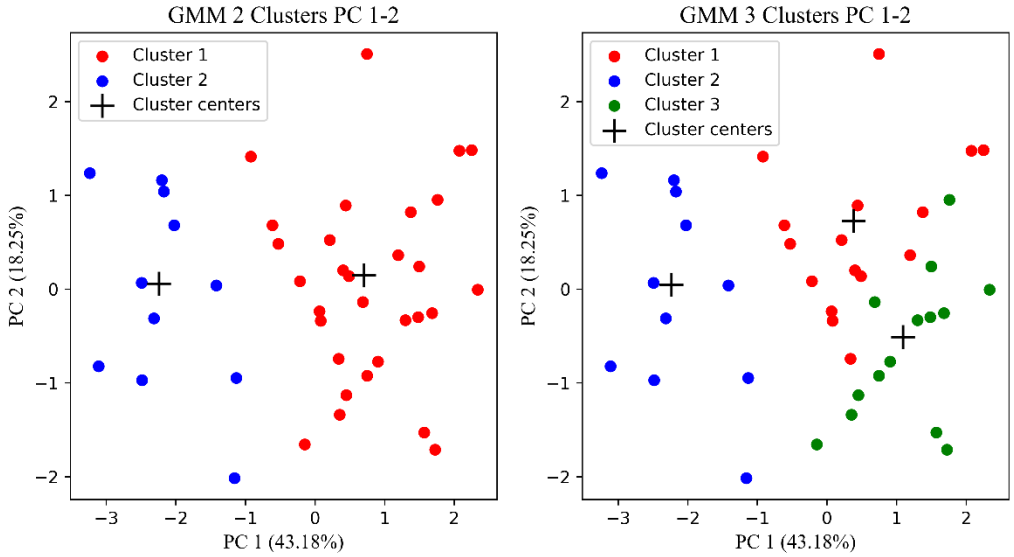
Bijlage 1. Voorlopige analyse



Figuur 15. Visualisatie van de k-means clusters voor k = 2 en k = 3, voor de voorlopige analyse



Figuur 16. Visualisatie van de DBSCAN clusters voor Eps = 1.5 en Eps = 1.8 en de Min Samples = 3, voor de voorlopige analyse



Figuur 17. Visualisatie van de GMM clusters voor k = 2 en k = 3, voor de voorlopige analyse

Bijlage 2. Statistische toets resultaten

Normaal verdeling

```
sprint_30m_zscore: p-value = 0.6048
sprint_30m_zscore is normally distributed
Agility_BVO_zscore: p-value = 0.8094
```

```
Agility_BVO_zscore is normally distributed
Max of jumpheight_zscore: p-value = 0.0074
Max of jumpheight_zscore is not normally distributed
Power_output_zscore: p-value = 0.7982
Power_output_zscore is normally distributed
RSI_zscore: p-value = 0.8812
RSI_zscore is normally distributed
Afstand_zscore: p-value = 0.5056
Afstand_zscore is normally distributed
```

K-means one-way ANOVA & post hoc Tukey test

```
one-way ANOVA K-means
Variabele: sprint_30m_zscore - F-statistic: 29.451460997826707 - p-valu
e: 1.8857657805471455e-08
Post hoc Tukey-test
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj   lower   upper  reject
-----------------------------------------------------
     0      1  -1.5525    0.0  -2.165 -0.9401   True
     0      2   0.1716 0.7747 -0.4409   0.784  False
     1      2   1.7241    0.0  1.1231  2.3251   True
-----------------------------------------------------

one-way ANOVA K-means
Variabele: Agility_BVO_zscore - F-statistic: 3.2408503859979967 - p-val
ue: 0.05016976126376346
Post hoc Tukey-test
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj   lower   upper  reject
-----------------------------------------------------
     0      1  -0.5268 0.2503 -1.3207   0.267  False
     0      2  -0.8202 0.0416  -1.614 -0.0263   True
     1      2  -0.2933 0.6322 -1.0724  0.4857  False
-----------------------------------------------------

one-way ANOVA K-means
Variabele: Power_output_zscore - F-statistic: 17.555319728556512 - p-va
lue: 3.9834307284351356e-06
Post hoc Tukey-test
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj   lower   upper  reject
-----------------------------------------------------
     0      1  -1.4733    0.0  -2.145 -0.8016   True
     0      2  -0.1637  0.824 -0.8354   0.508  False
     1      2   1.3096 0.0001  0.6505  1.9687   True
-----------------------------------------------------

one-way ANOVA K-means
Variabele: RSI_zscore - F-statistic: 14.853609530200549 - p-value: 1.71
32620083589813e-05
Post hoc Tukey-test
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj   lower   upper  reject
```

```
---------------------------------------------------
     0      1  -1.5343    0.0 -2.2555  -0.813   True
     0      2   -1.218 0.0006 -1.9393 -0.4968   True
     1      2   0.3162 0.5262 -0.3915   1.024  False
---------------------------------------------------

one-way ANOVA K-means
Variabele: Afstand_zscore - F-statistic: 4.070375295115309 - p-value: 0
.025019572746820765
Post hoc Tukey-test
Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================
group1 group2 meandiff p-adj   lower   upper  reject
---------------------------------------------------
     0      1  -0.8305 0.0488 -1.6574 -0.0035   True
     0      2   -0.855 0.0414 -1.6819  -0.028   True
     1      2  -0.0245  0.997  -0.836   0.787  False
---------------------------------------------------
```

K-means Kruskas-Wallis & poc hoc Mann-Whitney U test

```
Kruskal wallis result: 1.1054830170325874e-05
Comparison between group 1 and group 2: U = [180.], p = [1.75036166e-05
] (significant after Bonferroni correction)
Comparison between group 1 and group 3: U = [106.], p = [0.48166366] (n
ot significant after Bonferroni correction)
Comparison between group 2 and group 3: U = [15.], p = [0.00015023] (si
gnificant after Bonferroni correction)
```

DBSCAN one-way ANOVA & post hoc Tukey test

```
one-way ANOVA voor DBSCAN
Variable: sprint_30m_zscore - F-statistic: 4.400959769028761 - p-value:
0.019093198546205357
Post hoc Tukey test
Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================
group1 group2 meandiff p-adj   lower   upper  reject
---------------------------------------------------
     0      1  -1.7597 0.0141 -3.2068 -0.3126   True
     0      2   -0.317 0.5702 -1.0764  0.4424  False
     1      2   1.4427  0.043  0.0382  2.8473   True
---------------------------------------------------

one-way ANOVA voor DBSCAN
Variable: Agility_BVO_zscore - F-statistic: 1.9511384800886864 - p-valu
e: 0.15608943096841596
one-way ANOVA voor DBSCAN
Variable: Power_output_zscore - F-statistic: 5.608720123569537 - p-valu
e: 0.007338831295881084
Post hoc Tukey test
Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================
group1 group2 meandiff p-adj   lower  upper  reject
---------------------------------------------------
     0      1  -1.7953 0.0066 -3.1396 -0.451   True
     0      2  -0.0852 0.9534 -0.7906 0.6202  False
     1      2   1.7101 0.0077  0.4054 3.0148   True
```

```
--------------------------------------------------

one-way ANOVA voor DBSCAN
Variable: RSI_zscore - F-statistic: 1.0005351664943432 - p-value: 0.377
16180420707984
one-way ANOVA voor DBSCAN
Variable: Afstand_zscore - F-statistic: 0.8147232174741306 - p-value: 0
.45034591034381266
```

DBSCAN Kruskas-Wallis & poc hoc Mann-Whitney U test

```
Kruskal wallis result: 0.06531351610525382
Comparison between group 1 and group 2: U = [42.], p = [0.01715686] (no
t significant after Bonferroni correction)
Comparison between group 1 and group 3: U = [227.], p = [0.10681862] (n
ot significant after Bonferroni correction)
Comparison between group 2 and group 3: U = [22.], p = [0.35230769] (no
t significant after Bonferroni correction)
```

GMM one-way ANOVA & post hoc Tukey test

```
one-way ANOVA voor GMM
Variable: sprint_30m_zscore - F-statistic: 23.41169547797436 - p-value:
2.3664174739415023e-07
Post hoc Tukey test
Multiple Comparison of Means - Tukey HSD, FWER=0.05
==================================================
group1 group2 meandiff p-adj   lower   upper  reject
--------------------------------------------------
     0      1  -1.5791    0.0 -2.2447 -0.9134   True
     0      2   0.1955 0.7256 -0.4265  0.8174  False
     1      2   1.7745    0.0  1.0897  2.4593   True
--------------------------------------------------

one-way ANOVA voor GMM
Variable: Agility_BVO_zscore - F-statistic: 2.7393345820613972 - p-valu
e: 0.077381961036819
one-way ANOVA voor GMM
Variable: Power_output_zscore - F-statistic: 21.314696374344663 - p-val
ue: 6.201745572443769e-07
Post hoc Tukey test
Multiple Comparison of Means - Tukey HSD, FWER=0.05
==================================================
group1 group2 meandiff p-adj   lower   upper  reject
--------------------------------------------------
     0      1  -1.0547 0.0009 -1.7051 -0.4043   True
     0      2   0.7353 0.0146  0.1276  1.3431   True
     1      2     1.79    0.0   1.121  2.4591   True
--------------------------------------------------

one-way ANOVA voor GMM
Variable: RSI_zscore - F-statistic: 4.592679563509992 - p-value: 0.0163
5122536721287
Post hoc Tukey test
Multiple Comparison of Means - Tukey HSD, FWER=0.05
==================================================
group1 group2 meandiff p-adj   lower  upper  reject
```

```
---------------------------------------------------
     0      1  -0.7926  0.084 -1.6711  0.086  False
     0      2   0.3085 0.6334 -0.5124 1.1293  False
     1      2    1.101 0.0139  0.1973 2.0048   True
---------------------------------------------------

one-way ANOVA voor GMM
Variable: Afstand_zscore - F-statistic: 7.559791894818219 - p-value: 0.
0017220657482302244
Post hoc Tukey test
Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================
group1 group2 meandiff p-adj   lower   upper  reject
---------------------------------------------------
     0      1  -1.0268 0.0077 -1.8106 -0.2431   True
     0      2  -1.0171 0.0046 -1.7494 -0.2848   True
     1      2   0.0097 0.9995 -0.7965  0.8159  False
---------------------------------------------------
```

GMM Kruskas-Wallis & poc hoc Mann-Whitney U test

```
Kruskal wallis result: 2.637481981710753e-06
Comparison between group 1 and group 2: U = [166.], p = [0.00013112] (s
ignificant after Bonferroni correction)
Comparison between group 1 and group 3: U = [37.], p = [0.00195486] (si
gnificant after Bonferroni correction)
Comparison between group 2 and group 3: U = [3.], p = [5.72788597e-05]
(significant after Bonferroni correction)
```

Interne validatie

	Model	Sillhouette score	Dunn Index
0	K-means	0.164100	0.109717
1	DBSCAN	-0.005396	0.259236
2	GMM	0.154365	0.173001

Bijlage 3. Python code

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, DBSCAN
from sklearn.mixture import GaussianMixture
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
import seaborn as sns
from sklearn.metrics import silhouette_score
from scipy.stats import f_oneway, kruskal
from sklearn.metrics.pairwise import pairwise_distances
import matplotlib.font_manager as fm
```

```
# inlezen van de dataset
df = pd.read_excel("Z:\Physiological profiles
(Projectfolder)\Datasets\z_scores_mean_diff_teams.xlsx")

# Verwijderen van alle variabele dit niet nodig zijn voor dit onderzoek
X_zscores_remove = df.drop(['Season', 'Team', 'Player_id',
'Age_zscore','sprint_5m_zscore', 'sprint_10m_zscore',
            'sprint_20m_zscore', 'peakforce_zscore',
           'Force_output_zscore', 'weight_zscore', 'jumpheight_zscore',
            'Average of contacttime_zscore', 'RPE_zscore',
            'HRtop_zscore', 'Fatigue_zscore', 'peakpower_zscore'],
axis=1)


X_zscores_remove.head()

# Alle spelers verwijderen die een NaN waarde hebben
X_zscores = X_zscores_remove.dropna(axis = 0, how ='any')
column_age = X_zscores['Age']

# Lengte van de column
length = column_age.count()

# gemiddelde van de column
mean = column_age.mean()

# Standard deviatie of de column
std_deviation = column_age.std()

print("Length:", length)
print("Mean:", mean)
print("Standard Deviation:", std_deviation)

# Age variabele laten vallen
X_zscores = X_zscores.drop(["Age"], axis=1)
display(X_zscores)
Preliminaire analyse

from scipy.stats import shapiro
# Test of de data een normale verdeling volgt
# Als de p-waarden groter zijn dan 0,05, dan volgt de data een normale
verdeling.
for col in X_zscores.columns:
    stat, p = shapiro(X_zscores[col])
    print(f"{col}: p-value = {p:.4f}")

    # Histogram
    plt.hist(X_zscores[col], bins=10)
    plt.title(f'Frequentieverdeling {col}', fontname='Times New Roman')
    plt.xlabel(col, fontname='Times New Roman')
    plt.ylabel('Frequentie', fontname='Times New Roman')
```

```
    # Opslaan van figuur
    plt.savefig(f'normaal_verdeling_{col}.png', dpi=300)
    plt.close()  # Figuur sluiting om geheugen vrij te maken

    if p > 0.05:
        print(f"{col} is normally distributed")
    else:
        print(f"{col} is not normally distributed")

import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline
# Genereer een willekeurige correlatiematrix
corr_matrix = X_zscores.corr()

# Een heatmap maken met seaborn
fig, ax = plt.subplots()
sns.heatmap(corr_matrix, cmap='coolwarm', annot=True, fmt='.2f',
linewidths=.5, linecolor = 'black', ax=ax)

# De labels x- en y-as instellen
variables = ['sprint_30m_zscore', 'Agility_BVO_zscore', 'Max of
jumpheight_zscore',
             'Power_output_zscore', 'RSI_zscore', 'Afstand_zscore']
ax.set_xticklabels(variables, fontsize=10, rotation=90, fontname='Times
New Roman')
ax.set_yticklabels(variables, fontsize=10, fontname='Times New Roman')

# Title
ax.set_title('Correlation Matrix', fontname='Times New Roman')

# Pas de lay-out aan en sla de grafiek op als een afbeelding met een hoge
resolutie
fig.tight_layout()
plt.savefig('Correlation Matrix.jpg', dpi=600)

%matplotlib inline
# PCA-analyse
pca = PCA(n_components=2)
pca.fit(X_zscores)

# Krijg de ladingen van de PCA
loadings = pca.components_.T * np.sqrt(pca.explained_variance_)

# Transformeer de data naar de principal components
X_pca = pca.transform(X_zscores)

# Maak een dataframe van de ladingen
loadings_df = pd.DataFrame(loadings, columns=['PC{}'.format(i) for i in
range(1, pca.n_components_+1)], index=X_zscores.columns)
```

```
# Een heatmap maken met seaborn
plt.figure(figsize=(6, 4))
sns.heatmap(loadings_df, annot=True, cmap='coolwarm', center=0, vmin=-1,
vmax=1, linewidths=.5, linecolor='black')
plt.title('Loadings Matrix Heatmap', fontname='Times New Roman')
plt.xlabel('Principal Component', fontname='Times New Roman')
plt.ylabel('Original Variable', fontsize=10, fontname='Times New
Roman')  # increase the fontsize
plt.xticks(fontsize=10, fontname='Times New Roman')
plt.yticks(fontsize=10, fontname='Times New Roman')
plt.tight_layout()  # adjust the layout of the plot to prevent
overlapping
plt.savefig('Loadings Matrix.jpg', dpi=600)  # increase the dpi

# De variatie van de PC x 100 om het precentage te krijgen
PCA_expl = pca.explained_variance_ratio_ * 100

# Druk de uitgelegde variantieverhouding van elke hoofdcomponent af
print(PCA_expl)
print(sum(PCA_expl))

X_pca[:, 0] = -1 * X_pca[:, 0]


import matplotlib.pyplot as plt

# Het maken van elbow plot voor het K-means algoritme
Distortion = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_zscores)
    Distortion.append(kmeans.inertia_)

# Visualize results
plt.plot(range(1, 11), Distortion, marker='o', linestyle='-', color='b')
plt.xticks(range(1, 11))
plt.xlabel("Number of Clusters", fontname='Times New Roman')
plt.ylabel("Distortion", fontname='Times New Roman')
plt.title('Elbow Plot', fontname='Times New Roman')

plt.savefig('Elbow.jpg', dpi=600)
plt.show()


from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Parameters for K-means
random_state = 42
n_clusters = [2, 3]
fig, axs = plt.subplots(1, len(n_clusters), figsize=(10, 5))

for i, n in enumerate(n_clusters):
```

```
    kmeans_zscores = KMeans(n_clusters=n, random_state=random_state)
    kmeans_labels_zscores = kmeans_zscores.fit_predict(X_zscores)

    # Get the cluster centers in 2D space
    cluster_centers = kmeans_zscores.cluster_centers_
    cluster_centers_2d = np.dot(cluster_centers, pca.components_[:2,
:].T)
    cluster_centers_2d[:, 0] = -1 * cluster_centers_2d[:, 0]

    # Plot the clusters on PC1 and PC2
    ax = axs[i]
    cluster_colors = ['red', 'blue', 'green']
    scatter_plots = []
    for j, color in enumerate(cluster_colors[:n]):
        cluster_points = X_pca[kmeans_labels_zscores == j]
        scatter_plots.append(ax.scatter(cluster_points[:, 0],
cluster_points[:, 1], c=color, label='Cluster {}'.format(j+1)))

    ax.scatter(cluster_centers_2d[:, 0], cluster_centers_2d[:, 1], s=200,
marker='+', c='black', label='Cluster centers')
    ax.set_xlabel('PC 1 ({:.2f}%)'.format(PCA_expl[0]), fontname='Times
New Roman', fontsize=12)
    ax.set_ylabel('PC 2 ({:.2f}%)'.format(PCA_expl[1]), fontname='Times
New Roman', fontsize=12)
    ax.set_title('K-means {} Clusters PC 1-2'.format(n), fontname='Times
New Roman', fontsize=14)

    # Add legend
    ax.legend(handles=scatter_plots+[ax.collections[-1]],
labels=['Cluster {}'.format(j+1) for j in range(n)] + ['Cluster
centers'],
              loc='best', fontsize=10)

#plt.tight_layout()
plt.savefig('K-means_Clusters_2D.png', dpi=600)
plt.show()

from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt

# Parameters for DBSCAN
eps_values = [1.5, 1.8]
min_samples = 3
fig, axs = plt.subplots(1, len(eps_values), figsize=(10, 5))

cluster_colors = [ 'blue','red']

for i, eps in enumerate(eps_values):
    dbscan_zscores = DBSCAN(eps=eps, min_samples=min_samples)
    dbscan_labels_zscores = dbscan_zscores.fit_predict(X_zscores)
    dbscan_labels_zscores = np.where(dbscan_labels_zscores != -1,
dbscan_labels_zscores + 1, dbscan_labels_zscores)
```

```
    # Plot the clusters on PC1 and PC2
    ax = axs[i]
    unique_labels = np.unique(dbscan_labels_zscores)
    scatter_plots = []
    for j, label in enumerate(unique_labels):
        if label == -1:
            cluster_points = X_pca[dbscan_labels_zscores == label]
            scatter_plots.append(ax.scatter(cluster_points[:, 0],
cluster_points[:, 1], c='green', label='Noise'))
        else:
            cluster_points = X_pca[dbscan_labels_zscores == label]
            cluster_color = cluster_colors[j % len(cluster_colors)]
            scatter_plots.append(ax.scatter(cluster_points[:, 0],
cluster_points[:, 1], c=cluster_color, label='Cluster {}'.format(label)))

    ax.set_xlabel('PC 1 ({:.2f}%)'.format(PCA_expl[0]), fontname='Times
New Roman', fontsize=12)
    ax.set_ylabel('PC 2 ({:.2f}%)'.format(PCA_expl[1]), fontname='Times
New Roman', fontsize=12)
    ax.set_title('DBSCAN Eps={} Min Samples={}'.format(eps, min_samples),
fontname='Times New Roman', fontsize=14)

    # Add legend
    ax.legend(handles=scatter_plots, labels=['Noise'] + ['Cluster
{}'.format(label) for label in unique_labels if label != -1],
              loc='best', fontsize=10)

#plt.tight_layout()
plt.savefig('DBSCAN_Clusters_2D.png', dpi=600)
plt.show()

from sklearn.mixture import GaussianMixture
import matplotlib.pyplot as plt

# Parameters for GMM
random_state = 42

# Perform GMM clustering with 2, 3, and 4 clusters
n_clusters = [2, 3]
fig, axs = plt.subplots(1, len(n_clusters), figsize=(10, 5))

for i, n in enumerate(n_clusters):
    gmm_zscores = GaussianMixture(n_components=n,
random_state=random_state)
    gmm_labels_zscores = gmm_zscores.fit_predict(X_zscores)

    # Get the cluster centers in 2D space
    cluster_centers = gmm_zscores.means_
    cluster_centers_2d = np.dot(cluster_centers, pca.components_[:2,
:].T)
    cluster_centers_2d[:, 0] = -1 * cluster_centers_2d[:, 0]
    print(gmm_labels_zscores)
```

```
    # wissel 1 en 0 om voor de boxplot
    gmm_labels_zscores = np.where(gmm_labels_zscores == 1, 0,
np.where(gmm_labels_zscores == 0, 1, gmm_labels_zscores))
    print(gmm_labels_zscores)

    # Plot the clusters on PC1 and PC2
    ax = axs[i]
    cluster_colors = ['red', 'blue', 'green', 'purple']
    scatter_plots = []
    for j, color in enumerate(cluster_colors[:n]):
        cluster_points = X_pca[gmm_labels_zscores == j]
        scatter_plots.append(ax.scatter(cluster_points[:, 0],
cluster_points[:, 1], c=color, label='Cluster {}'.format(j+1)))

    ax.scatter(cluster_centers_2d[:, 0], cluster_centers_2d[:, 1], s=200,
marker='+', c='black', label='Cluster centers')
    ax.set_xlabel('PC 1 ({:.2f}%)'.format(PCA_expl[0]), fontname='Times
New Roman', fontsize=12)
    ax.set_ylabel('PC 2 ({:.2f}%)'.format(PCA_expl[1]), fontname='Times
New Roman', fontsize=12)
    ax.set_title('GMM {} Clusters PC 1-2'.format(n), fontname='Times New
Roman', fontsize=14)

    # Add legend
    ax.legend(handles=scatter_plots+[ax.collections[-1]],
labels=['Cluster {}'.format(j+1) for j in range(n)] + ['Cluster
centers'],
              loc='best', fontsize=10)

#plt.tight_layout()
plt.savefig('GMM_Clusters_2D.png', dpi=600)
plt.show()




# Maak een voorbeeld van een DataFrame dat de clusterlabels bevat
X_zscores_cluster_labels = X_zscores.assign(
    K_means_cluster_labels=kmeans_labels_zscores,
    DBSCAN_cluster_labels=dbscan_labels_zscores,
    GMMs_cluster_labels=gmm_labels_zscores
)

X_zscores_cluster_labels.head()
print(X_zscores_cluster_labels.columns)

X_zscores_cluster_K_means_labels = X_zscores.assign(
    K_means_cluster_labels=kmeans_labels_zscores)

print(X_zscores_cluster_K_means_labels.columns)

# Statistical tests voor de data die normaal verdeeld is (one-way ANOVA
```

```
met post hoc Tukey-test, inclusief Bonferroni-correctie)

# Lijst van groepen
groups_kmeans =
[X_zscores_cluster_labels[X_zscores_cluster_labels['K_means_cluster_label
s'] == i] for i in range(3)]

# Lijst van variabelen
var_list = ['sprint_30m_zscore', 'Agility_BVO_zscore',
'Power_output_zscore', 'RSI_zscore', 'Afstand_zscore']

# Voer one-way ANOVA-test uit voor elke variabele
for var in var_list:
    f_statistic, p_value = f_oneway(*[group[var].dropna() for group in
groups_kmeans])
    print("one-way ANOVA K-means")
    print("Variabele:", var, "- F-statistic:", f_statistic, "- p-value:",
p_value)

    # Voer Tukey-test uit voor significante variabelen
    if round(p_value, 2) <= 0.05:
        tukey_results =
pairwise_tukeyhsd(endog=X_zscores_cluster_labels[var],
                                        groups=X_zscores_cluster_labels['K_
means_cluster_labels'],
                                        alpha=0.05)
    print("Post hoc Tukey-test")
    print(tukey_results)
    print(' ')

#Statistical tests voor de data die niet-normaal verdeeld is (Kruskas-
Wallis met post hoc Mann Whitney U, inclusief Bonferroni-correctie)

# Lijst van variabelen
var_list = ['Max of jumpheight_zscore']

H, p = kruskal(*[group[var] for group in groups_kmeans for var in
var_list])
print("Kruskal wallis result:", p)

for i, j in combinations(range(len(groups_kmeans)), 2):
    U, p = mannwhitneyu(groups_kmeans[i][var_list],
groups_kmeans[j][var_list])
    if p * 3 < 0.05:
        print(f"Comparison between group {i+1} and group {j+1}: U = {U},
p = {p} (significant after Bonferroni correction)")
    else:
        print(f"Comparison between group {i+1} and group {j+1}: U = {U},
p = {p} (not significant after Bonferroni correction)")

groups_DBSCAN =
[X_zscores_cluster_labels[X_zscores_cluster_labels['DBSCAN_cluster_labels
'] == i] for i in range(3)]
```

```
var_list = ['sprint_30m_zscore', 'Agility_BVO_zscore',
       'Power_output_zscore', 'RSI_zscore', 'Afstand_zscore']

for var in var_list:
    f_statistic, p_value = f_oneway(*[group[var].dropna() for group in
groups_DBSCAN])
    print("one-way ANOVA voor DBSCAN")
    print("Variable:", var, "- F-statistic:", f_statistic, "- p-value:",
p_value)


    if round(p_value,2) <= 0.05:
        tukey_results =
pairwise_tukeyhsd(endog=X_zscores_cluster_labels[var],
                                          groups=X_zscores_cluster_labels
['DBSCAN_cluster_labels'],
                                          alpha=0.05)
        print("Post hoc Tukey test")
        print(tukey_results)
        print(' ')

var_list = ['Max of jumpheight_zscore']

H, p = kruskal(*[group[var] for group in groups_DBSCAN for var in
var_list])
print("Kruskal wallis result:", p)

for i, j in combinations(range(len(groups_DBSCAN)), 2):
    U, p = mannwhitneyu(groups_DBSCAN[i][var_list],
groups_DBSCAN[j][var_list])
    if p * 3 < 0.05:
        print(f"Comparison between group {i+1} and group {j+1}: U = {U},
p = {p} (significant after Bonferroni correction)")
    else:
        print(f"Comparison between group {i+1} and group {j+1}: U = {U},
p = {p} (not significant after Bonferroni correction)")


groups_GMM =
[X_zscores_cluster_labels[X_zscores_cluster_labels['GMMs_cluster_labels']
== i] for i in range(3)]

var_list = ['sprint_30m_zscore', 'Agility_BVO_zscore',
       'Power_output_zscore', 'RSI_zscore', 'Afstand_zscore']

for var in var_list:
    f_statistic, p_value = f_oneway(*[group[var].dropna() for group in
groups_GMM])
    print("one-way ANOVA voor GMM")
    print("Variable:", var, "- F-statistic:", f_statistic, "- p-value:",
p_value)
```

```
    if round(p_value,2) <= 0.05:
        tukey_results =
pairwise_tukeyhsd(endog=X_zscores_cluster_labels[var],
                                          groups=X_zscores_cluster_labels
['GMMs_cluster_labels'],
                                          alpha=0.05)
        print("Post hoc Tukey test")
        print(tukey_results)
        print(' ')

var_list = ['Max of jumpheight_zscore']

H, p = kruskal(*[group[var] for group in groups_GMM for var in var_list])
print("Kruskal wallis result:", p)

for i, j in combinations(range(len(groups_GMM)), 2):
    U, p = mannwhitneyu(groups_GMM[i][var_list], groups_GMM[j][var_list])
    if p * 3 < 0.05:
        print(f"Comparison between group {i+1} and group {j+1}: U = {U},
p = {p} (significant after Bonferroni correction)")
    else:
        print(f"Comparison between group {i+1} and group {j+1}: U = {U},
p = {p} (not significant after Bonferroni correction)")

%matplotlib inline
var_list = ['sprint_30m_zscore', 'Agility_BVO_zscore',
'Power_output_zscore', 'RSI_zscore', 'Afstand_zscore']

#Loop over elke variabele
for i, var in enumerate(var_list):
# Maak een nieuw figuur voor elke variabele
    fig, ax = plt.subplots(figsize=(6, 8))

    # Filter de data voor K-means
    X_Kmeans_var = X_zscores_cluster_labels[['K_means_cluster_labels',
var]]
    X_Kmeans_var = X_Kmeans_var + 1

    # Maak de boxplot voor K-means
    sns.boxplot(x='K_means_cluster_labels', y=var, data=X_Kmeans_var,
ax=ax, palette=cluster_colors)

    # Voeg de p-waarde toe aan de plot
    groups = [group[var].dropna() for _, group in
X_zscores_cluster_labels.groupby('K_means_cluster_labels')]
    pval = f_oneway(*groups)[1]
    ax.text(0.3, -0.15, 'p = {:.3f}'.format(pval),
transform=ax.transAxes, fontsize=20, fontname='Times New Roman')

    # Pas de plot aan voor K-means
    ax.set_title('{} K-means'.format(var), fontsize=24, fontname='Times
New Roman')
    ax.set_xlabel('Clusters', fontsize=22, fontname='Times New Roman')
```

```
    ax.set_ylabel('z-score'.format(var), fontsize=22, fontname='Times New
Roman')
    ax.tick_params(axis='both', which='major', labelsize=20)

    # Bewaar het figuur
    fig.savefig('k-means_boxplots{}.png'.format(var),
bbox_inches='tight')
    plt.close(fig)  # Sluit het figuur om geheugen vrij te maken

var = 'Max of jumpheight_zscore'

fig, ax = plt.subplots(figsize=(6, 8))

X_Kmeans_var = X_zscores_cluster_labels[['K_means_cluster_labels', var]]
X_Kmeans_var = X_Kmeans_var + 1

sns.boxplot(x='K_means_cluster_labels', y=var, data=X_Kmeans_var, ax=ax,
palette=cluster_colors)

ax.set_title('{} K-means'.format(var), fontsize=24, fontname='Times New
Roman')
ax.set_xlabel('Clusters', fontsize=22, fontname='Times New Roman')
ax.set_ylabel('z-score'.format(var), fontsize=22, fontname='Times New
Roman')
ax.tick_params(axis='both', which='major', labelsize=20)

groups = [group[var] for group in groups_kmeans]
H, p = kruskal(*groups)
ax.text(0.3, -0.15, 'p = {:.3f}'.format(p), transform=ax.transAxes,
fontsize=20, fontname='Times New Roman')


fig.savefig('K-means_boxplot_{}.png'.format(var), bbox_inches='tight')
plt.close(fig)

%matplotlib inline
var_list = ['sprint_30m_zscore', 'Agility_BVO_zscore',
       'Power_output_zscore', 'RSI_zscore', 'Afstand_zscore']

for i, var in enumerate(var_list):
    fig, ax = plt.subplots(figsize=(6, 8))

    X_DBSCANs_var = X_zscores_cluster_labels[['DBSCAN_cluster_labels',
var]]
    X_DBSCANs_var = X_DBSCANs_var + 1

    sns.boxplot(x='DBSCAN_cluster_labels', y=var, data=X_DBSCANs_var,
ax=ax, palette=cluster_colors)

    ax.set_title('{} DBSCAN'.format(var), fontsize=24, fontname='Times
New Roman')
    ax.set_xlabel('Clusters', fontsize=22, fontname='Times New Roman')
    ax.set_ylabel('z-score'.format(var), fontsize=22, fontname='Times New
```

```
Roman')
    ax.tick_params(axis='both', which='major', labelsize=20)

    groups = [group[var].dropna() for _, group in
X_zscores_cluster_labels.groupby('DBSCAN_cluster_labels')]
    pval = f_oneway(*groups)[1]
    ax.text(0.3, -0.15, 'p = {:.3f}'.format(pval),
transform=ax.transAxes, fontsize=20, fontname='Times New Roman')

    fig.savefig('DBSCAN_boxplots{}.png'.format(var), bbox_inches='tight')
    plt.close(fig)

var = 'Max of jumpheight_zscore'

fig, ax = plt.subplots(figsize=(6, 8))
X_GMMs_var = X_zscores_cluster_labels[['DBSCAN_cluster_labels', var]]
X_GMMs_var = X_GMMs_var + 1

sns.boxplot(x='DBSCAN_cluster_labels', y=var, data=X_GMMs_var, ax=ax,
palette=cluster_colors)

ax.set_title('{} DBSCAN'.format(var), fontsize=24, fontname='Times New
Roman')
ax.set_xlabel('Clusters', fontsize=22, fontname='Times New Roman')
ax.set_ylabel('z-score'.format(var), fontsize=22, fontname='Times New
Roman')
ax.tick_params(axis='both', which='major', labelsize=20)

groups = [group[var] for group in groups_DBSCAN]
H, p = kruskal(*groups)
ax.text(0.3, -0.15, 'p = {:.3f}'.format(p), transform=ax.transAxes,
fontsize=20, fontname='Times New Roman')

fig.savefig('DBSCAN_boxplot_{}.png'.format(var), bbox_inches='tight')
plt.close(fig)

%matplotlib inline
var_list = ['sprint_30m_zscore', 'Agility_BVO_zscore',
       'Power_output_zscore', 'RSI_zscore', 'Afstand_zscore']

for i, var in enumerate(var_list):
    fig, ax = plt.subplots(figsize=(6, 8))

    X_GMMs_var = X_zscores_cluster_labels[['GMMs_cluster_labels', var]]
    X_GMMs_var = X_GMMs_var + 1

    sns.boxplot(x='GMMs_cluster_labels', y=var, data=X_GMMs_var, ax=ax,
palette=cluster_colors)

    ax.set_title('{} GMM'.format(var), fontsize=24, fontname='Times New
Roman')
    ax.set_xlabel('Clusters', fontsize=22, fontname='Times New Roman')
    ax.set_ylabel('z-score'.format(var), fontsize=22, fontname='Times New
```

```
Roman')
    ax.tick_params(axis='both', which='major', labelsize=20)

    groups = [group[var].dropna() for _, group in
X_zscores_cluster_labels.groupby('GMMs_cluster_labels')]
    pval = f_oneway(*groups)[1]
    ax.text(0.3, -0.15, 'p = {:.3f}'.format(pval),
transform=ax.transAxes, fontsize=20, fontname='Times New Roman')

    fig.savefig('GMMs_boxplots{}.png'.format(var), bbox_inches='tight')
    plt.close(fig)

var = 'Max of jumpheight_zscore'

fig, ax = plt.subplots(figsize=(6, 8))

X_GMMs_var = X_zscores_cluster_labels[['GMMs_cluster_labels', var]]
X_GMMs_var = X_GMMs_var + 1

sns.boxplot(x='GMMs_cluster_labels', y=var, data=X_GMMs_var, ax=ax,
palette=cluster_colors)

ax.set_title('{} GMM'.format(var), fontsize=24, fontname='Times New
Roman')
ax.set_xlabel('Clusters', fontsize=22, fontname='Times New Roman')
ax.set_ylabel('z-score'.format(var), fontsize=22, fontname='Times New
Roman')
ax.tick_params(axis='both', which='major', labelsize=20)

groups = [group[var] for group in groups_GMM]
H, p = kruskal(*groups)
ax.text(0.3, -0.15, 'p = {:.3f}'.format(p), transform=ax.transAxes,
fontsize=20, fontname='Times New Roman')

fig.savefig('GMMs_boxplot_{}.png'.format(var), bbox_inches='tight')
plt.close(fig)

# Berekenen van de silhouette-index
# K-means
silhouette_avg_k_means = silhouette_score(X_zscores,
kmeans_labels_zscores)
print(f"The average silhouette score for the K-means clustering is
{silhouette_avg_k_means}")

# DBSCAN
silhouette_avg_DBSCAN = silhouette_score(X_zscores,
dbscan_labels_zscores)
print(f"The average silhouette score for the DBSCAN clustering is
{silhouette_avg_DBSCAN}")

# GMM
silhouette_avg_GMM = silhouette_score(X_zscores, gmm_labels_zscores)
print(f"The average silhouette score for the GMM clustering is
```

```
{silhouette_avg_GMM}")

# Berekenen van de Dunn index
# Dunn index(DI) = min.seperation/max.diameter

# Compute pairwise distances between data points
distances = pairwise_distances(X_zscores)

# Dunn-index voor K-means
# Vind de minimale afstand tussen punten in verschillende clusters
min_distance = np.inf
for i in range(len(X_zscores)):
    for j in range(i+1, len(X_zscores)):
        if kmeans_labels_zscores[i] != kmeans_labels_zscores[j]:
            if distances[i][j] < min_distance:
                min_distance = distances[i][j]


# De maximale diameter van elke cluster
max_diameters = []
for label in np.unique(kmeans_labels_zscores):
    cluster_points = X_zscores[kmeans_labels_zscores == label]
    max_diameter = 0
    for i in range(len(cluster_points)):
        for j in range(i+1, len(cluster_points)):
            if distances[i][j] > max_diameter:
                max_diameter = distances[i][j]
    max_diameters.append(max_diameter)

#print("Maximum diameter of each cluster:", max_diameters)
Dunn_Index_k_means = min_distance/max_diameter
print("Dunn Index score voor K-means:", Dunn_Index_k_means)


# Dunn-index voor DBSCAN
# Vind de minimale afstand tussen punten in verschillende clusters
min_distance = np.inf
for i in range(len(X_zscores)):
    for j in range(i+1, len(X_zscores)):
        if dbscan_labels_zscores[i] != dbscan_labels_zscores[j]:
            if distances[i][j] < min_distance:
                min_distance = distances[i][j]




# De maximale diameter van elke cluster
max_diameters = []
for label in np.unique(dbscan_labels_zscores):
    cluster_points = X_zscores[dbscan_labels_zscores == label]
    max_diameter = 0
    for i in range(len(cluster_points)):
        for j in range(i+1, len(cluster_points)):
```

```
            if distances[i][j] > max_diameter:
                max_diameter = distances[i][j]
    max_diameters.append(max_diameter)


Dunn_Index_DBSCAN = min_distance/max_diameter
print("Dunn Index score voor DBSCAN:", Dunn_Index_DBSCAN)

# Dunn-index voor GMM
# Vind de minimale afstand tussen punten in verschillende clusters
min_distance = np.inf
for i in range(len(X_zscores)):
    for j in range(i+1, len(X_zscores)):
        if gmm_labels_zscores[i] != gmm_labels_zscores[j]:
            if distances[i][j] < min_distance:
                min_distance = distances[i][j]



# De maximale diameter van elke cluster
max_diameters = []
for label in np.unique(gmm_labels_zscores):
    cluster_points = X_zscores[gmm_labels_zscores == label]
    max_diameter = 0
    for i in range(len(cluster_points)):
        for j in range(i+1, len(cluster_points)):
            if distances[i][j] > max_diameter:
                max_diameter = distances[i][j]
    max_diameters.append(max_diameter)

Dunn_Index_GMM = min_distance/max_diameter
print("Dunn Index score voor GMM:", Dunn_Index_GMM)

# Definieer de namen van de modellen en de namen van de tests
models = ['K-means', 'DBSCAN', 'GMM']
test_names = ['Sillhouette score', 'Dunn Index']

# Definieer de resultaten voor elke test
results = [
    [silhouette_avg_k_means, Dunn_Index_k_means],
    [silhouette_avg_DBSCAN, Dunn_Index_DBSCAN],
    [silhouette_avg_GMM, Dunn_Index_GMM]
]

# Maak het DataFrame aan met de modelnamen als eerste kolom
results_df = pd.DataFrame({'Model': models})

# Voeg de testresultaten toe als kolommen aan het DataFrame
for i, test_name in enumerate(test_names):
    results_df[test_name] = [row[i] for row in results]

# Toon het uiteindelijke DataFrame
display(results_df)
```