TU WIEN Informatics

Evaluierung von Methoden zum Ranking von Performance Variablen

Erstellung eines Prototypen zur Leistungsevaluierung

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Business Informatics

eingereicht von

Sebastian Scholz
Matrikelnummer 1526884

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Thomas Grechenig
Mitwirkung: Dominik Hölbling

Wien, 10. November 2024

Unterschrift Verfasser

Unterschrift Betreuung

TU WIEN Informatics

Evaluating Methods for Generalized Ranking of Performance Variables

Prototyping an Individual Performance Evaluation Tool

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Sebastian Scholz
Registration Number 1526884

to the Faculty of Informatics

at the TU Wien

Advisor: Thomas Grechenig
Assistance: Dominik Hölbling

Vienna, 10th November, 2024

Signature Author

Signature Advisor

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Evaluating Methods for Generalized Ranking of Performance Variables

Prototyping an Individual Performance Evaluation Tool

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Sebastian Scholz
Registration Number 1526884

ausgeführt am
Institut für Information Systems Engineering
Forschungsbereich Business Informatics
Forschungsgruppe Industrielle Software
der Fakultät für Informatik der Technischen Universität Wien

Advisor: Thomas Grechenig

Wien, 10th November, 2024

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Sebastian Scholz

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 10. November 2024

Sebastian Scholz

Danksagung

Ich möchte mich zuallererst bei Dominik Hölbling für die umfassende Betreuung der Arbeit wie auch für die Möglichkeit bedanken, mein Studium der Informatik mit meiner Leidenschaft für Sport zu vereinen.

Ohne den fachlichen Input meines guten Freundes Max und den emotionalen und grammatikalischen Input meiner Freundin Kathi wäre diese Arbeit nicht das, was sie heute ist.

Zuletzt möchte ich vorallem meinem Vater danken, der mir dieses Studium nicht nur überhaupt ermöglicht hat, sondern mich - gerade in der Anfangszeit - immer ermutigt hat, diesen Weg auch zu Ende zu gehen.

Abstract

This work addresses the limitations in generalizations of scoring methods for sports performances. Various scoring methods capable of handling multiple performance indicators were evaluated, focusing on reproducibility, stability and ranking quality. Among the approaches included were purely statistical methods as well as those incorporating domain knowledge. The best-performing method was used to develop a system that enables automatic scoring and ranking of performances with multiple performance indicators. Furthermore, it enables comparison across different normative profiles. With the aim of ensuring optimal context, even from large datasets, normative profiles are selected dynamically.

A prototype was built to make the aforementioned methods and functionality accessible. The performance of the selected methods were evaluated on two datasets: a large dataset ranking countries by quality of living, and a smaller, high-dimensional dataset derived from latent variables. Results showed similar methods performance on large datasets, but on smaller datasets, linear combinations incorporating domain knowledge excelled in reproducibility. Future improvements could combine indicator weighting with linear combinations for enhanced scoring methods.

Keywords: *Unsupervised ranking, generalized ranking approach, multiple performance indicators, sports performance ranking, adaptable prototype*

Kurzfassung

Diese Arbeit befasst sich mit Einschränkungen bei der Automatisierung der Reihung von sportlichen Leistungen. Verschiedene Bewertungsmethoden, die in der Lage sind, mehrere Leistungsindikatoren zu vereinen, wurden evaluiert, wobei der Fokus auf Reproduzierbarkeit, Stabilität und Qualität der erstellten Ranglisten lag. Unter den betrachteten Ansätzen waren sowohl rein statistische Methoden als auch solche, die Domänenwissen integrieren. Die bestgeeignete Methode wurde im Anschluss verwendet um ein System zu entwickeln, das die Automatisierung der Bewertung und Reihung von verschiedensten sportlichen Leistungen ermöglicht. Zusätzlich wird - um einen Vergleich über verschiedene normative Profile hinweg zu ermöglichen - basierend auf den vorhandenen Daten automatisch das optimale normative Profil ermittelt.

Ein Prototyp wurde entwickelt um die zuvor genannten Methoden und Funktionen zugänglich zu machen. Die Leistung der ausgewählten Methoden wurde anhand von zwei Datensätzen bewertet: einem großen Datensatz, der Länder nach Lebensqualität reiht, und einem kleineren, hochdimensionalen Datensatz, der basierend auf latenten Variablen erstellt wurde. Die Ergebnisse zeigen eine ähnliche Qualität der Methoden bei großen Datensätzen, bei kleineren Datensätzen zeichnet sich jedoch die Methode der linearen Kombinationen, welche Domänenwissen berücksichtigt, durch eine hohe Reproduzierbarkeit aus. Zukünftige Verbesserungen könnten die Kombination von Indikatorgewichtung mit linearen Kombinationen beinhalten um die Bewertungsmethoden weiter zu optimieren.

Keywords: *Unüberwachtes Ranking, Verallgemeinerter Ranking-Ansatz, Mehrere Leistungsindikatoren, Reihung sportlicher Leistungen, Anpassbarer Prototyp*

Contents

CHAPTER 1

Introduction

In sports, it is often the goal to compare different performances - be that in competitions or simply to improve in training. This can be easily done in scored competitions, where the quality of the performance can be represented and evaluated by a single variable, for example running or weight lifting. When not looking at single-person competitions but group-sports (e.g. handball), it already becomes harder to evaluate the performance of a single person/player, since there is no score for each single player, just for the team as a whole. Similarly, there are sports performances that cannot be represented by a single score, e.g. in training, where no competition is available or no judges assign scores.

Evaluating performances that are not represented in a single score is still possible. However, the amount of data that needs to be analysed and compared increases drastically for each additional metric that defines a 'good performance'. Additionally, the amount of work to be done increases, when multiple different performances are to be ranked. This could for example be the case when multiple goals are present, such as in training routines, where one might track physical data about the athlete in addition to achieving certain goals. Furthermore, the domain knowledge needed to interpret said data correctly increases with each variable.

As portable devices become more ubiquitous so does tracking data while athletes or patients perform. With the greater amount of variables the correct interpretation of the data becomes increasingly hard. While domain experts are certainly able to do so, there are two problems. First, when a larger amount of performances is recorded, the amount of time needed to interpret them increases. Secondly, when the data is being processed on an end-user device (e.g. smartphone app), a domain expert may not always be present. As such, an automated way that is able to incorporate the domain knowledge could not only help with the speed of interpretation but also deliver the result to the end-user faster. As this problem arises for a multitude of performances, the solution should be adaptable to many different kinds as well - and thus be reusable in many projects.

1.1 What is a (good) sports performance?

Performance in sports involves the pursuit of excellence, where an athlete evaluates their progress both quantitatively and qualitatively in order to advance toward their desired goals [7]. Performance success plays a critical role in determining opportunities for participation in elite individual sports, influencing key aspects such as qualification for major competitions, acquisition of sponsorships, and allocation of government funding to sports organizations, which in turn provide financial support to athletes. Metrics such as winning performances and personal bests are commonly used as indicators of exceptional achievement in this context. Moreover, performance success serves as a key outcome for sports medicine and research in elite sports, with the goal of identifying and examining factors that impact performance outcomes (PO) [44]. As such, defining PO and developing objective methods for its assessment require careful consideration of the diverse range of factors that contribute to successful outcomes in sport.

1.1.1 Performance Indicators

In many sports a performance (or its outcome) can be represented by just a single score or variable. For example *time needed* (e.g. for swimming a certain length), *weight lifted*, *goals scored* can be used to assess a performance quickly. However, these variables are usually highly aggregated and do not tell the whole story of a certain performance. While this is desirable for competitions, the underlying factors that determine the outcome of a sports performance, are numerous and complex [37]. Even seemingly simple sports performances, such as running a race, involve the complex allocation and coordination of neural, psychological and physiological resources [37].

A single variable measured for a performance is referred to as *performance indicator*. A performance indicator tries to encapsulate some aspect of a performance. These variables can either express a single selected aspect or encompass a combination of aspects [19]. In order to be useful, a performance indicator should have a high correlation to the outcome of a certain performance. Performance indicators are often used by coaches or analysts to assess the performance of individuals, teams or certain elements of a team [19]. They may be used in isolation or comparatively with other athletes or teams.

Performance indicators based on bio-mechanical processes are often linked to the PO in a hierarchical relationship, see figure 1.1 with the example of long jump. These bio-mechanical performance indicators can often be broken down further into different performance indicators, that in sum define a certain aspect of the performance.

Bio-mechanical performance indicators are especially prevalent in sports that require good movement technique, such as acrobatic (e.g. gymnastics, diving), athletic (jumping and throwing) and cyclic (swimming, running) sports [55]. In the hierarchical modeling approach, the movement variables that contribute to the successful execution of a skill in a meaningful way can be considered performance indicators. These performance indicators usually are kinetic variables or other parameters, such as certain angles or speeds of body segments [19].

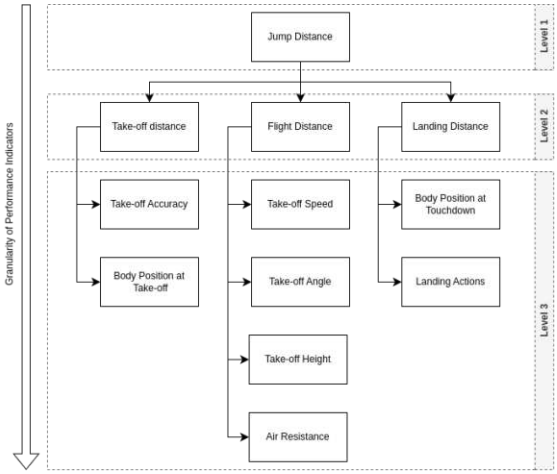


Figure 1.1: Hierarchical relationship of performance indicators of the long jump (adapted from [19]). The performance indicator *Jump Distance*, can be split into further, more granular, performance indicators. These can in turn again be seen as being composed of other, even more granular, performance indicators.

On the one hand, bio-mechanically based performance indicators are less common in team sports. On the other hand, *notational analysis* focuses more on studying the interaction between players in traditional team- and match-sports [19]. Performance indicators based on notational analysis focus on general match-indicators or technical-indicators. They reflect the physiological, psychological, technical and tactical demands of many sports. For example, in Rugby such indicators could be Turnovers, tackles or passes/possession over the course of a game or exercise.

The goal of measuring the factors that contribute to a performance is to understand the influence of certain variables on the PO [44]. This means that for training and improvement purposes athletes/coaches are incentivised to gather more and more data, as this can facilitate a deeper understanding of the individual factors contributing to the desired PO.

1.1.2 Frame of reference when comparing performances

After measuring multiple different performance indicators, athletes and coaches alike will most likely want to use the gathered data to gain some insight into the studied performance. Performances can only be evaluated relatively to each other. A single performance without any frame of reference can not be considered either good or bad [10]. Only when other performances that share the same goal are used as a frame of reference, a given performance can be evaluated. In a sense a single performance is not good or bad. A performance can only evaluated on if it is better (or worse) than other performances with the same goal. These reference values are also known as *normative performances* [18]. Usually, normative performances are created by taking the average

value for a performance indicator from a representative sample size.

As such, the frame of reference used to compare a certain performance can change the evaluation result of the given performance of interest. Elite athletes may look dominant when compared to amateurs, while looking average when being compared to other athletes of a similar level. A selection of a reasonable frame of reference is made in many kinds of sports competitions. Take for example weight classes in boxing, where only boxers within a certain range of weight compete against each other. As more weight is a huge advantage in most combat sports, a competitor may only be fairly assessed when competing against other athletes with a similar weight.

Similar to the hierarchical modeling approach outlined for performance indicators, the same concept can be applied to the physical baseline an athlete has. Bodily traits can be split into more granular variables to create a even more similar peer group that is used as a frame of reference when comparing the performance. For example, weight could be expanded by considering the variables *muscle mass* and *body fat percentage*. An athlete with a (significantly) higher body fat percentage will have a different physiology than one with a lower body fat percentage, even if both have the same weight in kilograms. On the other hand, these variables may also be combined to form more homogeneous peer groups, such as combining age and weight groups.

1.1.3 Evaluating Performances

Once an adequate frame of reference is found, a performance can be evaluated. For this, it needs to be compared to the other performances in the peer group. After it was determined if the performance was better or worse than its peers, it can be assessed accordingly. The same holds true when the improvement of an athlete is to be assessed. Here the frame of reference is clear, however, the performances of the same athlete still need to be compared with each other to see if an athlete did indeed improve.

For many athletes these PO are ordinal or continuous, meaning the performances are either assessed in a ranking (placed from best to worst) or they are assigned a number which represents the effectiveness in reaching the PO (in other words assigning points to the performances) [44].

If there is only one performance indicator to consider, then creating these ordinal or continuous placements is straight forward, as only a single number has to be compared. However, if it is not possible (or desired) to express the PO in a single variable (for example in complicated training exercises) comparing performances becomes increasingly difficult. On the other hand, certain trade-offs might exist between two or more performance indicators. In this case focusing on one performance indicator could come at the expense of another performance indicator.

In order to compare and evaluate performances that are defined by many different, sometimes interrelated, performance indicators a person needs a great deal of domain knowledge. Especially, if there is no aggregate performance indicator that may be used as a kind of end-sum to evaluate the performance.

1.2 Rationale

In practice, athletes' development is often guided not by theoretical frameworks or expert advice, but by decisions based on intuition, experience, and specific events. Coaches frequently rely on these factors to shape their training approaches and actions [9]. On the other when creating software to support performance analysis, developers and analysts alike have to reinvent the wheel - as there are few generalized software modules out there that can be easily adapted to fit a new PO.

1.2.1 Comparison of performance indicators

Formally speaking, a performance can be seen as a measurement of effectiveness in achieving a predefined goal [13]. Many performances have a precisely defined goal that can be easily expressed by a single number. If the PO is evaluated by a single number then arranging a set of performances is easily done, as only a single number has to be compared for each of the performances.

However, as noted above, using only a single performance indicator to assess a performance might not tell the whole story and leave out valuable insight on how the PO was reached. On the other hand, some performance might even lack such a single performance indicator by which they can be compared.

When trying to assess a performance into an ordinal scale every performance on said scale needs to be compared to all other performances - leading to a high number of comparison necessary. Additionally, for each comparison between two distinct pair of performances, all of the defined performance indicators need to be compared in order to get the full picture [34]. A process that, if done manually, takes a lot of time. An increase of combinations to be compared significantly increases the required workload and time. Additionally, selecting the correct result when evaluating performance indicators requires substantial domain expertise, especially when some indicators are interdependent.

An example would be a material arts exercise, where in a addition to hitting a target it is tracked how often the athlete gets hit himself and how complex the techniques used by the athlete are, thus tracking the level of difficulty of a certain training session. Here, simpler techniques might increase the amount of successful hits on the target, while decreasing the level of difficulty. Focusing more on defense might lower the hits taken by the athlete but might also decrease the successful hits landed by themselves. In this, simple, example there is the need to compare three interdependent variables for each pair of performances. Additionally, the PO cannot be expressed by a single aggregate performance indicator. This means that the person ranking the performances has to have a great deal of domain knowledge in order to correctly assess which of any two performances is better. Furthermore, different people might value different aspects (i.e. performance indicators) more or less. This would result in personal bias when ranking the performances, and ultimately in different rankings depending on who analyses and ranks the performances. Lastly, creating a continuous ranking, where distance between

two performances indicates how *much* better or worse one is, would increase the demand for domain knowledge even further.

Some supporting techniques are used to more easily compare lots of performance indicators, for example aids like radar charts [39]. While this provides some help, lots of manual analysis is still required. On the other hand, most machine learning approaches would need a lot of pre-analysed and sorted training data, something that is not available for every sport and certainly not for many more uncommon training exercises. Thus, the rankings have to be created and updated by hand. For example for 71 performances with 18 dependent variables each, 100+ hours of an 'experienced analyst' were needed, showcasing the enormous effort needed [51].

While there are many approaches for supporting performance analysis, fewer exist that try to rank a given set of performances, and even fewer have adapted a generic approach that can be used to rank a set of arbitrary performances (see section 1.3).

1.2.2 Comparing between different baselines

In recent times the traditionally rigid classification of athletes in sports has come under discussion [33]. The concrete classifications depend on sport, context and other factors. Mostly however, they aim at controlling variables that might give an athlete an advantage but are outside of their control, such as age. However, for many reasons one might want to compare athletes despite their inherent differences. In such a case, again, one is reduced to domain knowledge and other guesses as comparing performance indicators will not work due to inherently different predispositions. There currently is no system to eliminate inherent (dis-)advantages between athletes when comparing their performances, especially not when dealing with multiple different, interdependent performance indicators.

Comparison between different groups could also be done for non-physical differences, such as player positions, active time in a match (either time spent on the field, or specific time periods, such as first vs. second half-time), or during which match status (losing, winning, draw) a certain performance was given, or other groupings, or if other advantages were present (e.g. home advantage). Another possible factor to influence a performance is the matchup, as it is more likely to appear dominant when paired against a weaker opponent. Techniques that compare different baselines could also be used to analyze where a player's skills are best utilized, for example, determining if they perform better as a forward or defender based on specific performance indicators. Additionally, the system could control external advantages, such as wind or slope style in ski jumping [38].

1.3 State of the art

In this section, already existing approaches are listed and evaluated. The existing work has been split into three types. First, work exploring the relative evaluation of performances is given. Second, a list of commonly used techniques for gaining insight in performance data is listed. Here, *any* insight is considered, without filtering papers for a specific type

of insight that is sought after. Lastly, papers that are specific to *ranking* performances and their used techniques are explored in more detail.

1.3.1 (Normative) performance profiles as frame reference

Using normative profiles as a frame of reference when evaluating sports performances is a well known approach to performance analysis [18]. As different sports create different specialisations, different normative profiles are created for each one [49]. Many of the created profiles focus on athletes on the elite level, and less so on different performance groups. However, the profiles do differentiate between different physical baselines, one of the most common being gender. This shows that adopting the frame of reference to the studied subjects (e.g. athletes) is indeed beneficial and serves a better understanding of the given performance.

1.3.2 Performance analyses

Performance analysis techniques enable coaches and support staff to collect objective data from certain performances, providing insights into the delivered performance that help develop effective training routines, strategies and tactics. With the help of technology the number of measured performance indicators increased drastically [31]. New ways of generating insight from the given data have to be tested and evaluated [34].

Various performance analysis methods can be applied depending on the aspect being measured, the type of data analyzed, and the specific sport being examined. Below are some of the commonly identified techniques [26]:

- various statistical methods, such as Mann-Whitney U comparison , chi-squared test, linear models, ANOVA, regression ([51],[45],[20], [15])
- machine learning, such as support vector machine, multi-layer perceptron, convolutional network, recurrent network, gradient boosted tree ([29],[52])
- neural Networks ([42],[6])
- decision trees ([54],[21])
- social network analysis ([40],[50])
- notational analysis ([46],[43])
- self organizing maps ([8])

The above mentioned methods are mostly designed to draw insight from the given data, while the objectives vary - mostly a reduction in complexity or quantifying the difference between different groups were the goals of the work. Additionally, for classification tasks a ground truth is available most of the time. While showing that the mentioned methods

can indeed be used to deal with performance indicators, they do not attempt to rank the performances in any way.

Comparison between different performance profiles

Using a Mann-Whitney U collective comparison, different levels (Sub-Elite and Elite) of football players were compared in 18 different performance indicators [51]. A chi-squared test was used to differentiate the performances of winning and loosing handball teams [45]. The comparison across performance profiles, or different baselines, largely utilizes the same approaches as mentioned above. Mostly used are statistical comparisons.

1.3.3 Ranking performances

As stated above, most performance analysis approaches try to gather additional insights from the data in order to provide new or easier digestible information for coaches and analysts. However, none of these approaches try to rank the performances in an ordinal or continuous way - the preferred feedback type of the athletes themselves [44]. In the following, some works that do try to rank performances are analyzed, grouped by the employed techniques. Here, in order to gather insight from a wider scope, also papers that concern performances in domains other than sports are considered. While the domains - and thus the meaning of the data - varies greatly, there are often similar approaches from a statistical point of view. As can be seen, there are few approaches that try to create a generic method to ranking performances. Most of the work focuses on only a single PO, and does not consider other types of performances. This means that the same methodology needs to be evaluated, and implemented again for each new performance type that one wants to rank.

Principal Component Analysis

One very commonly used statistical tool is Principal Component Analysis (PCA). PCA is used to project data into a new coordinate system which has lower dimensionality than the original data - while keeping the highest possible variation. Using PCA, Dong et al. ranked the performance of Chinese hospitals [12]. They selected multiple measurements, which they grouped into three categories. The Hospitals where then ranked based on the resulting principal component value (two components were used). Donà et al. also used PCA to encode knee movements of race walkers, finding that by encoding the movement patterns of various athletes, a separation between different skill levels can be drawn [11].

Factor Analysis

Similarly to PCA, Factor Analysis (FA) is used to reduce the information given to a lower dimensionality. The so called Factors, are unobserved variables that highly correlate to the observed ones. Also in the domain of sports performances, Brasile compared different classes of wheelchair basketball players [4]. at first, various skills related to the sport

where measured, then a FA was performed on the data, finding that only one of the three classes could be separated clearly based on the results of the FA.

Linear Models and Regression

Kostrzewa et al. used a regression based approach to predict the sports performance of judo athletes [24]. Nine different variables were measured. The predicted variable was the Polish Judo Association ranking table score for each athlete. It was shown that for different weight groups, different variables correlate more with the predicted score. This work shows not only that regression can not only be used to estimate rankings, but also make a strong case for individualized ranking methods based on physical properties, e.g. the weight of an athlete.

Combining both PCA and regression, Sharma ranked the readiness of various different countries for e-tail [47]. An "e-tail readiness index" was created using seven variables. First, weights for the variables were calculated using PCA. Then, using these weights, a weighted linear combination was used to create the final score of each country. Not only using a linear combination but combining it with weights - here stemming from a PCA - ascribes different importance to different measures, since not every variable necessarily has the same impact on the resulting performance.

Heasman et al. created a ranking system for individual players in Australian football based in the individual impact of each player [16]. For this, three positive game actions and three negative game actions where weighted manually, by assigning an according weight to certain game actions - positive or negative respectively.. In addition to the weights the (individual) game time influenced the final score. Their main findings were, that such a scoring system could be easily replicated for other sports. On the other hand, they found significant differences between different positions, thus making the case that comparisons should be drawn between players on the same position. Finally, they found that the derived impact score is a reasonable method to distinguish between different player performances.

Self Organizing Maps

A generic approach to ranking is offered by utilising Self Organizing Maps (SOM), as presented by Martins et al. [35]. While SOM normally uses euclidean distances, Martins et al. adapted the algorithm to allow for different weighting of individual attributes (measurements). In their case the expert input was used to mark the data points, which are expected to be in the first and last place of the ranking. The weights for their adaption of SOM were then defined by a genetic algorithm, based on the expert input. The data they used was "obtained by sanitary inspections in Brazilian blood banks" [35], resulting in 85 data points, each with 77 measurements. While they did not apply a strict ranking, they binned the data points in ten rankings. The adapted SOM approach produced rankings that more closely resembled a desired Gaussian distribution more closely, than traditional approaches (like PCA).

Preliminary explanation of SOM This paragraph briefly explains the concept of SOMs and serves as a basic introduction to how they will be implemented in section 3.3.2. Initially described by Kohonen, Self Organizing Maps (SOM) are unsupervised Artificial Neural Network (ANN) Models [23]. ANNs are motivated by biological structures that have evolved to deal with a variety of different situations, as they try to mimic the structure of biological systems and employ an approach based on artificial neurons that are a simplification of their biological counterpart. SOMs specifically draw inspiration from selforganizing, biological systems, found e.g. in neurobiology [36]. They consist of a single layer of neurons arranged along an n-dimensional grid. These maps are used for visualization and clustering tasks by projecting high-dimensional data distributions into lower dimensions while still preserving the relations between data items [36].

SOMs utilize a competitive learning approach to arrange a map of neurons, represented by weight vectors. Through this the map is arranged to closely resemble the distribution of the input data. When a training example is fed to the SOM the distance of all weight vectors (neurons) to the input is calculated - usually a euclidean distance is used. Then, the most similar unit - the so called 'best matching unit' (BMU) - is selected. After a BMU is found, its weight and that of close neighbours, is adjusted to move closer to the input. In an iterative process, the magnitude of movement and the range of affected neighbours are decreased with each iteration.

The basic formula for the update process is as follows [36]:

$$w_i(n+1) = w_i(n) + \alpha(n) \cdot h\left(w_{bmu}(n), w_i(n), r(n)\right) \cdot ||x(n) - w_i(n)|| \tag{1.1}$$

Where

- $w_i(n)$ is weight vector of unit i at iteration n,
- n is the current iteration in the training,
- $\alpha(n)$ is a monotonically decreasing learning coefficient,
- $h\left(w_{bmu}(n), w_i(n), r(n)\right)$ is a neighborhood function that returns the distance between the BMU and the currently selected neuron w_i in step n,
- $r(n)$ is the neighborhood radius, shrinking with time and
- $x(n)$ is a random sample from the dataset, or the input for the given iteration n.

A basic algorithm to train a SOM is given in algorithm 1.1, which is run until the iteration limit λ is reached.

Algorithm 1.1: Algorithm to train a Self Organizing Map

```
initialize the map of size n × m
while s < λ do
    x_n ← random input vector from X
    w_bmu ← closest neuron to x_n
    update each unit i with 1.1
end
```

Preliminary explanation of genetic algorithms Again a brief introduction to genetic algorithms is given, as they are implemented in section 3.3.2. Genetic algorithms are algorithms inspired by nature that seek to solve optimization problems. Genetic algorithms use a heuristic approach and can be used for many optimization problems that may not have an efficient solution or are otherwise hard to solve - such as the traveling salesman problem [25].

The idea behind genetic algorithms is to translate the concept of evolution into an algorithmic approach that finds solutions for a given problem. Genetic algorithms rely on a collection of possible solutions called population. Each population representing a potential solution to the specific optimization problem that is to be solved by the algorithm. The possible solution space can be continuous (called vectors) or consist of bits (called bit strings) [25]. In the case of combinatorial problems, solutions often consist of symbols in a list. New solutions are created within the given solution space. Each of these newly created populations are referred to as a new generation.

After a new potential solution (generation) is generated, it has to be evaluated whether it provides a better result than solutions of previous generations. This is done via a so called *fitness function*. The fitness function returns a value that (most of the time) indicates a better solution, if the given value is higher. Thus, populations that result in a higher value are preferred.

As can be seen in algorithm 1.2, the basic approach for genetic algorithms is based on two loops [25]. Initially, a population is initialized. Then, the nested loop generates new possible solutions with crossover and mutations until a population is complete.

Bézier Curves

Li et al. used a ranking method based on Bézier Curves to create an unsupervised ranking method that is geared towards multi-attribute objects [30]. In their paper, they rank countries based on four different parameters.

Preliminary explanation of Bézier curves Bézier curves are parametric curves, which are used in computer science to draw smooth, continuous curves that otherwise would be hard to represent mathematically. They serve as the basis to the approach of Li et al. which is implemented in section 3.3.2. Initially introduced by Pierre Bézier in

Algorithm 1.2: Basic Genetic Algorithm

```
initialize population
while termination condition not reached do
    while population not complete do
        crossover
        mutation
        phenotype mapping
        fitness function
    end
    select parental population
end
```

the 1960s while working at Rénault. Bézier Curves were used to specify the shape of cars in a mathematically exact way, instead of using wooden forms or similar, less precise, tools. Today, Bézier curves are used in a wide range of applications e.g. computer aided design, vector graphics or fonts.

Bézier curves can be constructed by using a set of discrete control points and the formula 1.2 [14]. The amount of control points used in the calculation is referred to as the *degree* of the Bézier curve.

$$b(t) = \sum_{i=0}^{n} B_i^n(t) b_i \tag{1.2}$$

Where $b_i(t)$ are the control points, and B_i^n refers to the so called *Bernstein Polynomials* [14].

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{1.3}$$

With the binomial coefficient being:

$$\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!}, & \text{if } 0 \leq i \leq n \\ 0, & \text{else} \end{cases} \tag{1.4}$$

Matrix form of Bézier curves Alternatively Bézier curves can be written in matrix form. A curve of the form

$$x(t) = \sum_{i=0}^{n} c_i C_i(t)$$

can be interpreted as a dot product [14]:

$$x(t) = (c_0, \ldots, c_n) \begin{pmatrix} C_0(t) \\ \vdots \\ C_n(t) \end{pmatrix}$$

A cubic Bézier curve can be written as (P being coordinate values in n-dimensions):

$$B(t) = (1-t)^3 \cdot P_0 + 3(1-t)^2 t \cdot P_1 + 3(1-t)t^2 \cdot P_2 + t^3 \cdot P_3 \tag{1.5}$$

A cubic Bézier curve in this form can be displayed as follows, and (temporarily) simplified by omitting the control points:

$$\begin{aligned} B(t) &= (1-t)^3 \cdot P_0 + 3(1-t)^2 t \cdot P_1 + 3(1-t)t^2 \cdot P_2 + t^3 \cdot P_3 \\ B(t) &= (1-t)^3 + 3(1-t)^2 t + 3(1-t)t^2 + t^3 \end{aligned} \tag{1.6}$$

$$\begin{aligned} B(t) = &(1-t)^3 \\ &+3 \cdot (1-t)^2 \cdot t \\ &+3 \cdot (1-t) \cdot t^2 \\ &+t^3 \end{aligned} \tag{1.7}$$

The equation above can be expanded into:

$$\begin{aligned} B(t) = \quad & -1 \quad t^3 + 3t^2 - 3t + 1 \\ & +3 \quad t^3 - 6t^2 + 3t + 0 \\ & -3 \quad t^3 + 3t^2 + 0t + 0 \\ & + \quad t^3 + 0t^2 + 0t + 0 \end{aligned} \tag{1.8}$$

That, in turn, can be rewritten as a dot product:

$$\begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ 3 \\ -3 \\ 1 \end{pmatrix} + \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ -6 \\ 3 \\ 0 \end{pmatrix} + \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \cdot \begin{pmatrix} -3 \\ 3 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{1.9}$$

That dot product can then be condensed into a single matrix operation by rearranging the formulas above.

$$\begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \cdot \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & t & t^2 & t^3 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix} \tag{1.10}$$

Adding the coordinates of the control points P_i as a single matrix yields the cubic Bézier curve in a matrix form:

$$B(t) = \begin{pmatrix} 1 & t & t^2 & t^3 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix} \cdot \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} \tag{1.11}$$

Similarly, Bézier curves of different degrees can be displayed as a matrix operation.

1.3.4 Comparing different rankings

When multiple different rankings, of the same performances, are given their (dis-)similarity needs to be quantified.

In order to compare the similarity of rankings objectively Kendall's Tau may be used. Kendall's Tau is a measurement for the ordinal association between two measured quantities. A popular usage of this metric is the measurement of rank correlations [41].

Kendall's Tau requires that two variables X and Y, are paired observations [41]. For both variables it needs to be at least ordinal and the calculation of the correlation between the two of them needs to be possible. Kendall's Tau is a measurement of the similarity between the ranks occurring when data is ordered by a certain property. For this, the data can either be scaled or the property can be used to create ranks.

For any n sample pairs in the two bivariant variables (X, Y) exist $m = \frac{1}{2}n(n-1)$ possible comparisons. These comparisons are drawn between the points (X_i, Y_i) and (X_j, Y_j) [41], where a pair is called *concordant* if $X_j - X_i$ and $Y_j - Y_i$ have the same sign. In other words, a pair of observations is concordant if they have the same ranking in Y and X. If the signs are different, then the pair is called *discordant*. Either $Y_i = Y_j$, $X_i = X_j$ or both is called a tie, and does not count as either concordant or discordant.

Here, C denoting the number of concordant pairs and D denoting the numbers of discordant pairs, Kendall's Tau is given as follows [41]:

$$\tau = \frac{C - D}{m} \tag{1.12}$$

The above formula can be extended to consider ties as well, the new form is referred to as τ_b [41].

$$\tau_b = \frac{C - D}{\sqrt{(C + D + X_0)(C + D + Y_0)}} \tag{1.13}$$

Where X_0 is the number of ties only in X and Y_0 as the number of ties in Y.

1.4 Solution approach

Within this thesis, approaches to the problems outlined above will be evaluated. The requirements for the ranking methods and the prototype were defined in an semi structured

expert interview. The defined requirements are described below. Additionally, unlike other approaches to ranking, the prototype needs to be usable for *any* given performance. This ensures that a generic way of ranking performances is provided that does not need to be specially adapted to the given performance type, as would be the case for e.g. many machine learning approaches.

The performance of each item in the system is evaluated using a scoring mechanism that quantifies its effectiveness. These scores rank performances from best to worst, providing a clear ordering for comparison. The calculation is designed to ensure that the distribution of scores follows a normal distribution, which enables more intuitive comparisons and analysis. This distribution ensures that extreme outliers (either high or low performances) are rare, while most performances fall around an average score.

To provide a meaningful frame of reference (normative profile) for each performance, the system dynamically creates subcategories based on the characteristics of available performances. The ordering and importance of the characteristics is defined by the user. This approach allows the system to form categories that are specifically tailored to the current performance type, ensuring each performance is compared to a peer group that is as homogeneous as possible. Additionally, mechanisms are introduced that prevent the groups from being too small for meaningful comparison.

Even when performances are placed in different subgroups, the system allows for cross-group comparison. A comparison score is calculated, enabling the evaluation of performances across different groups. This comparison is essential for identifying which performances excel beyond their immediate peers, offering a broader view of their effectiveness. The system highlights which performances outperform others, regardless of the subgroup to which they belong, allowing for better overall evaluation and optimization of the results.

A prototypical system will be built to evaluate the behaviour of the identified method for score creation, dynamic (sub)categorization and comparison between categories. Table 1.1 contains a summarizing of the main features the prototype is expected to provide.

1.5 Aims

The goal for this thesis is to evaluate methods that can be used to address the need for ranking and evaluating performances using a structured and automated approach. The chosen methods will be implemented into a prototype. The prototype will focus on dimensionality reduction, performance ranking and adaptability, making it suitable for integration into various domain-specific projects. The further aim of this work is to simplify the manually intensive way of ranking performances consisting of multiple, different performance indicators. The work will be provided as a reusable, customizable prototype, reducing the complexity of software development for tasks in this domain.

The primary objectives of this work are threefold:

No.	Feature	Description	Key Details
1	**Performance score calculation**	Assign a score to each performance based on its effectiveness in reaching the predefined goal. The resulting scores allow the performances to be ranked from best to worst.	- Scores rank performances by effectiveness. - Scores follow a normal distribution. - A single number that is easy to compare.
2	**Dynamic subcategory selection and creation**	Automatically categorize performances into subcategories (normative profiles) based on their properties as defined by the user.	- Subcategories are dynamically created based on available performances. - Each performance is placed into its appropriate subcategory. - Each performance is placed in only one subcategory. - Subcategories are only created if there are enough members to draw meaningful conclusions.
3	**Category comparison score calculation**	Calculate a comparison score for evaluating performances across different subcategories (normative profiles).	- Allows comparison of two or more performances across different subcategories. - Highlights, which performance is superior within a peer group.

Table 1.1: Planed features of the prototype, as defined by an expert.

1. Evaluating dimensionality reduction techniques The first objective is to evaluate various approaches to reduce the dimensionality of the performance data, i.e. the multiple performance indicators. Dimensionality reduction is used to compress the given information for a performance into a single variable: the score. This score is then subsequently used to rank the performances. By examining different methods, this work aims to create an effective ranking system that can handle generic performance data efficiently and generate rankings similar to those that would be created manually. This objective provides the base for feature 1, and by extension also for feature 3, in table 1.1.

2. Creating a descriptive schema for performance data The second objective is to design a schema that allows for the flexible description of performance data. This schema will defined how both the performance indicators and the information used for grouping are combined to automatically process the performance data. The schema must be versatile enough to accommodate various types and forms of data. Based on this schema, feature 2 in table 1.1 will be realized.

3. Developing a software prototype The third objective is to implement a software component that integrates the previous two objectives. This software will:

- **Utilize the schema** to parse and interpret the input data.
- **Automatically calculate scores** for the given performances. These scores rank the performance in accordance to their effectiveness. For this, the evaluated dimensionality reduction techniques are used.
- **Dynamically create normative performance profiles** by creating subcategories depending on the available baseline characteristics for the performances. These will be automatically updated and adapted as new performances are added.
- **Provide a comparison score** that allows performances from different subcategories (compared to different normative profiles) to be directly compared. This comparison score will help determine the better of two performances despite different starting conditions or characteristics.

This work will demonstrate how such a prototype can be utilized to solve the complex issues of ranking and comparing different types of performances. By integrating dimensionality reduction and automating the ranking, the prototype - and the selected methods used - will serve as a valuable tool for athletes and coaches alike.

1.6 Research questions

The concrete research questions (RQ) that will be answered within this thesis are the following:

- **RQ1:** Which mathematical method is most appropriate for a scoring system that attempts to rank performances in their effectiveness, especially if there is no single variable expressing the effectiveness directly?
  - **RQ1.1:** Are methods that incorporate domain knowledge more appropriate for addressing the given problem than those that do not?
- **RQ2:** Which method can be used to compare two performances when they are in different subcategories due to different physical baselines?

CHAPTER 2

Methods

2.1 Methodology

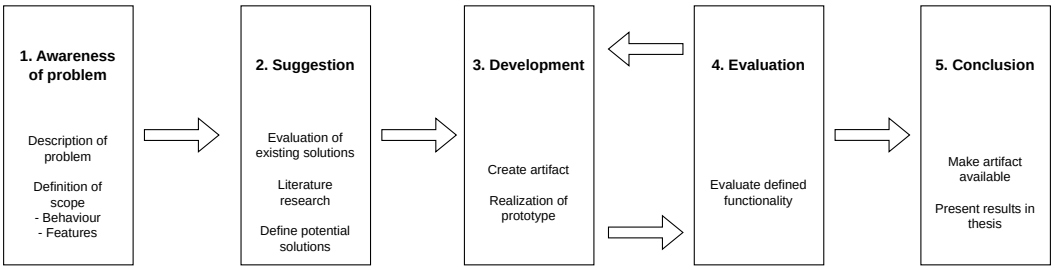


Figure 2.1: Five phases of design science according to Kuechler and Vaishnavi (adapted from [27]).

The base methodology for creating the software component will be *design science*, based on the approach described by Kuechler and Vaishnavi [27]. They identify five major activities in a design science research project: *Awareness of Problem*, *Suggestion*, *Development*, *Evaluation* and *Conclusion*. The resulting artifact at the end of the design process will be a prototype [53].

Awareness of Problem: Initially, the problem that is to be solved is described. For this, the requirements and expected behaviours are defined. The scope for the prototype and what kind of functionality it should provide are determined. The initial list of requirements is defined by addressing the research questions, which provide a starting point for needs and constraints that are to be addressed by the prototype.

Suggestion: After the problem and its scope are clearly defined, already existing solutions for the problem will be explored and evaluated. This phase ensures that the

approach that is being developed is indeed a novel one. For this, preexisting approaches to dimensionality reduction and ranking will be evaluated. Additionally, it will be checked whether these approaches have already been applied to a similar problem. At the end of this phase, some potential solutions shall be defined that can be further explored and refined in the next step.

To find potential approaches, an initial literature research [48] will be conducted to find ones that have already been applied to similar problems in the realm of performance analysis. A second literature research will be done for statistical methods that are used to reduce dimensionality in multivariate data - regardless of their specific problem space. Suitable approaches will be listed and checked whether they could cover the requirements defined in the previous step.

Development: In the development phase, the artifact is created based on the potential solutions found in the previous step. For this, the artifact is constructed and refined in iterative steps. For the most promising approaches found in the literature, initial implementations will be created, which in turn can then be compared to each other. The required metadata needed for the implemented scoring approaches will - partially - answer RQ1.1. Additionally, the functionality for creating categories will be added. Again, the meta-information needed will be part of RQ1.1. After functionality for both scores and categories have been implemented, a comparison between categories will be implemented.

Evaluation: The process cycles between the Development- and Evaluation-phase iteratively. This is done to further refine the artifact based on findings in the evaluation. For this project, the evaluation will be done in multiple ways since different, distinct functionalities are to be evaluated. For a detailed explanation of the evaluation done for design science see section 2.3.

Conclusion: The contributions of this thesis will be the artifact itself alongside the answered research questions. The most appropriate approaches for scoring and comparing performances will be selected for the final version of the artifact. Additionally, the schema used to encapsulate the domain knowledge is described in detail, so that the prototype can be used for different use cases.

2.2 Technical plan

This section explains the steps taken in this work from a more technical perspective, for an outline of the steps see figure 2.2, especially the box titled *Creating a generic prototype*. The initial step for this work is to define the various methods for scoring and ranking the performances, with some methods being taken from the literature and some other mathematical approaches having been adapted for this purpose. These methods are then implemented in a prototype, ensuring that they are able to deal with various

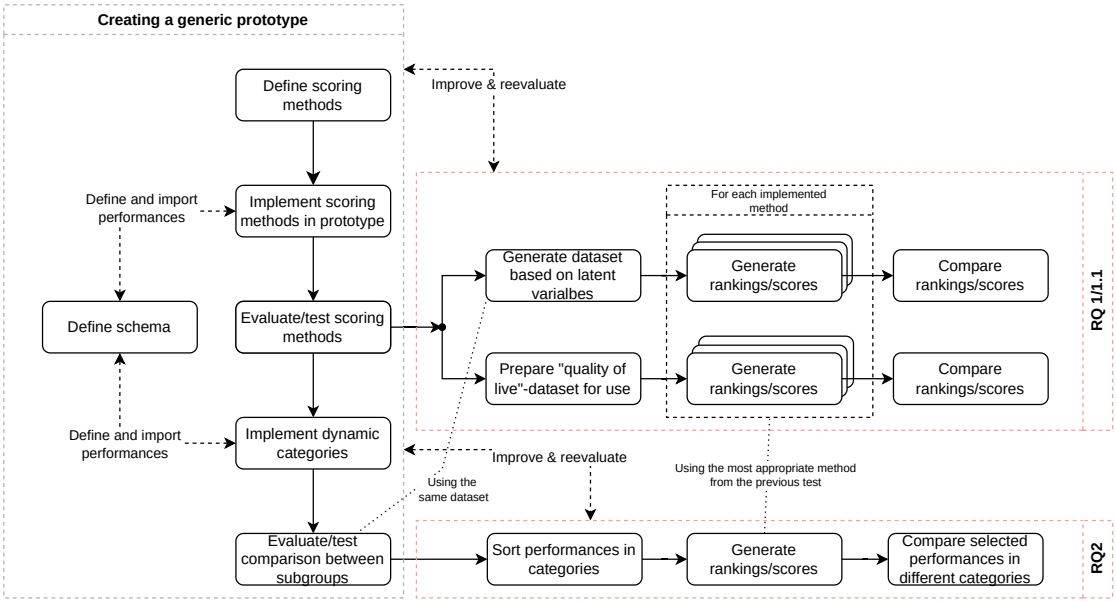


Figure 2.2: Outline of the steps in this work.

performances, as in this work two datasets will be used. Following this, the scoring methods are tested from a technical perspective. For the scientific evaluation see section 2.3.

After the datasets are defined and the methods evaluated, a schema is defined. This schema is iteratively developed by using it as a basis for the creation of dynamic classification of performances (functionality 2 in table 1.1). This enables automatic selection of normative profiles for subsequent evaluations.

After the categorization algorithm is implemented, the same generated dataset as for the validation of scoring methods is sorted into subcategories. The Comparison of performances with different baselines can then in turn be testes across these subcategories (feature 3 in table 1.1).

The prototype will be written in the python programming language. The advantages of this programming are its versatility and flexibility, which is important since the final requirements are not known at this point. Additionally, a rich ecosystem of libraries and frameworks exists for python - this is especially true for libraries related to data processing and machine learning. The following libraries will be used during the creation:

- pandas[1]: for tabular representation of data/performances, a widely used format in the python ecosystem.
- pydantic[2]: for data and schema validation.

[1] https://pandas.pydata.org/
[2] https://docs.pydantic.dev/

- scikit-learn[3]: contains many Machine Learning related tools and functionality related to data pre-processing.
- scipy[4] and numpy[5]: both libraries provide well tested mathematical and statistical functionality.

While integrated tightly there are four different main components in the architectural overview of the prototype:

- A class that represents the performances, which represents the coupling of domain knowledge and data.
- A function that handles the creation of subcategories depending on the given domain knowledge and available data.
- A function that uses the existing data and respective categories to produce rankings based on scores it creates.
- A function that uses the existing categories and the previously mentioned scoring function to create inter-class comparability.

2.3 Scientific methods

The Evaluation phase outlined in section 2.1 will be done using the methods defined by Hevner et al. [17]. In this thesis, functional as well as experimental testing (simulation) will be used to evaluate the artifact's behavior. See figure 2.2 for a conceptual overview. Again, the criteria for the scoring methods were given by an expert in a semi structured interview. Table 2.1 contains the list of evaluation criteria.

The evaluation will be twofold. In the first step the scoring methods themselves will be evaluated, thereby covering RQ1 and RQ1.1. Secondly, the comparison between different subcategories will be evaluated, thereby covering RQ2.

Evaluation for RQ1/1.1 In order to answer RQ1, multiple methods will be implemented in the prototype. These methods will then each be used to rank a given dataset. The evaluation of the resulting rankings (one per dataset) will be twofold. First, the rankings will be compared to each other and to an assumed ground truth. This ground truth can be either an external ranking, or some other underlying (latent) variables. It is important to note that the goal of this work is not an exact replication of a ground truth, as this would not be conductive to a generalized approach. Instead, the ground truth is a way to see how *sensible* the ranking is. Here, large deviations show a fundamental

[3] https://scikit-learn.org/stable/index.html
[4] https://scipy.org/
[5] https://numpy.org/

No.	Keyword	Description	Key Details
1	**Deterministic**	Rankings should not change when run on the same data	From the same input data, the product must always be a consistent ranking, with no random elements in the outcome.
2	**Monotonicity**	Rankings should not change when the same data is included in larger datasets	Rankings of existing performances should not change when additional data is introduced, preserving the order of previous results.
3	**Transitivity**	The ranking should respect transitivity: if A $>$ B and B $>$ C, then A $>$ C	The scoring method should reflect logical consistency across comparisons, ensuring that indirect comparisons preserve relative rankings.
4	**Reflective of Superiority**	If performance A is better than performance B, the ranking should reflect this	The method must capture the relative superiority of performances, so better performances rank higher than weaker ones based on objective criteria.
5	**Efficiency**	The ranking algorithm should be performant in terms of time and space complexity	The ranking calculation must scale efficiently to large datasets and be computationally feasible, also addition of new performances should be considered.
6	**Robustness**	The method should have no special requirement towards the shape of the performance	There should be no dependencies towards the form of performance indicators (aside from being numerical).

Table 2.1: Evaluation criteria for the scoring methods, as defined in a semi structured interview.

problem in the ranking, while small ones can be seen as just weighing certain aspects in the given performance differently. Secondly, the methods will be used to rank the performances within the subcategories generated in the step outlined above. Here it can be seen how stable the rankings generated by the methods are - e.g. if the ranking is $A < B < C$, then after the removal of B it should still be $A < C$. The results of these comparisons will answer RQ1. Additionally, it can be compared how different types of methods compare to each other. The methods can be sorted into two categories, those that do not incorporate domain knowledge and those that do. Comparing these two categories answers RQ1.1.

Evaluation for RQ2 Lastly, it will be tested how informative the comparison score between performances from different categories is. For this, the same datasets that were used for evaluating the scores are sorted into categories and then certain pairs of performances from different categories are compared. The result of this approach is then used to answer RQ2.

CHAPTER 3

Results

In this chapter, the implementation details of the prototype will be discussed in greater detail. As there are four main concepts realised in the prototype, four sections will each discuss one in detail. Firstly, a representation of domain knowledge is defined. A schema will be created, that describes the type of performance that is being observed. In addition, the schema encapsulates expert knowledge by containing further information about the collected data points. Secondly, the concept of categories and resulting subcategories will be explained, as they are the basic unit of a (sub-)dataset - on which later operations will be performed. These categories are created based on the schema defined in the previous section. Thirdly, the main requirements for a scoring method are explained and possible methods, which will be implemented and subsequently evaluated, are defined. Lastly, a method to compare records that are in different subcategories is implemented and explained.

3.1 A Schema to represent the data

In this section, the definition of the schema that represents the users domain knowledge are discussed. As the schema is the base for the behaviour of the software component, the schema needs to encode all the domain knowledge necessary for the software component to work as expected. The intentions behind the schema and its formal definition are discussed.

Initially, the usage and benefits of such a schema are described. This schema, as can be seen in figure 2.2, is the prerequisite for reading and working with performances in the prototype.

Why Create A Schema A schema specifies a formal definition and description of data, data types and the expected structure of files containing said data. In the context

of this work, a schema is used to describe performances and their individual variables. Since the software component needs to be able to process arbitrary types of performances - regardless of their number or types of measurements - the schema that a user defines in order to describe said performance must support this kind of flexibility. Essentially, the schema defines the expected structure of performances fed into the software component. All of the performances given as input to the program must conform to this schema. This means that all of the performances share the same structure and intentions. In other words the performances conforming to a certain schema all share the same goal, the same kinds of measurements and the same amount of measurements. All of the performances only differ in their values but not their structure, as the structure is fixed by the schema. This schema is used as a basis for the calculation of scores and sorting performances into various categories.

Since the schemas describing the input data are processed automatically, they too must conform to a certain structure. The shared structure ensures that their influence on the behaviour of the software component is well defined. For this, the schemas themselves must conform to another schema. Such schemas describing other schemas are commonly referred to as meta-schemas.

A meta-schema is a formal definition of a model and defines the allowed data, the data types and structure. If a schema conforms to a specific meta-schema, it is referred to as an instance of that meta-schema. If it does not conform to it, it is not a valid instance of said meta-schema.

As stated before the software component is required to be able to deal with arbitrary forms of performances. The word *arbitrary* in this context refers to an amount of measurements recorded and multiple different forms of possible categories, which are created in a customizeable fashion, that is not predefined. This means that, while the amount of measurements and categories may be arbitrary, the way in which both are set must follow a certain structure. The provided schema (or instance of the expected meta-schema) acts as the configuration file, defining the behaviour of the software component.

3.1.1 Defining a meta-schema

As the instances of the meta-schema will depend on the performance data that is to be processed, the contribution of this work does not create schemas, but a meta-schema to which the schemas must conform, in turn providing information on how to process performance records.

While - theoretically - the definition of the meta-schema can be specified in various forms, the software component is implemented in python, which is why the definition of the schema was done iteratively and implicitly by implementing a parser. The parser was extended in every step of the work in which new features were required, until all features were covered. The schema (the instance conforming to the meta-schema) is expected to be in the JSON file format[1] or defined as an object in the source code directly. This

[1] `https://www.json.org/json-en.html`

```
{
  "id_name": "ID",
  "group_min_size": 1,
  "personal_info": [],
  "performance_info": []
}
```

Listing 1: Root of a meta-schema instance.

ensures compatibility with python and also gives users a familiar format to work with.

As stated, the main form of definition is done via a parser. This parser also serves as a validation of the input files, creating a warning if a value is undefined.

The expected structure of the schema is a top-level object with two lists inside, as there are two types of measurements. First *personal information* is used to discern the individuals which delivered a certain performance. This information is expected to reflect the physical precondition of each person. As such, they are used to create categories of more similar persons performing (see section 3.2). The second list contains *performance information*, which reflects the effectiveness of the performance when reaching a certain goal (note that all performances adhering to a certain schema-instance are expected to have the exact same goal). These kinds of measurements are then used to rank the performances (see section 3.3). Additionally, two fields are expected that denote the *id* of each performance and the minimum size of a category.

The field `id_name` defines a field that is expected in each input file. The value of this field should be *unique* across all other performances. The unique id is used to identify certain performances no matter where they are in the resulting categories, as the current subcategory in which a performance resides can change overtime.

The field `group_min_size` is used to define the minimum amount of members a category needs before it will be created. This option prevents the creation of categories that may have too few members to be informative, such as categories with a single member. Since this threshold is highly dependent on not only the amount of data available but also the domain itself, it can be easily set and adapted by the expert user.

Note that the following examples are not intended to be a formal definition of the meta-schema, but rather a sort of visualization of what (parts of) a meta-schema instance may look like. Thus, they are kept in the JSON format. In the listing 1, the two lists containing definitions of the two types of measurements can be seen.

Personal Information

As mentioned before, the measurements concerning individual baselines are defined in the list accessed by the key `personal_info`. The basic information for these values provided are `name`, `position` and `split_value`. Additionally, an `is_id` property

```
"personal_info": [
    { "name": "name", "is_id": true },
    { "name": "gender", "position": 1, "split_value": 0 }
    { "name": "weight", "position": 2, "split_value": [60, 80] }
]
```

Listing 2: Example description of personal information items in the schema instance.

can be set, which indicates that the value in fields with the specified name are used to *uniquely* identify a record (i.e. names or timestamps). A brief example can be seen in listing 2.

The field `name` is mandatory and identifies the key by which a certain value can be accessed. The performance records are generally expected to be in a python dictionary. A dictionary (often referred to as dict) is a built-in datastructure used for storing and managing key value pairs. It is expected that each personal info object in the schema has a corresponding entry in all of the performance records, where the value of the name in the personal info object denotes the key in the dictionary, which holds the value for the associated variable.

Via the key `split_value` the creation of subcategories can be defined. The input can either be an array or a single number. For n breakpoints $n+1$ categories will be created (given records in the category exist), as for each value a new category is created and all performances where the measurement is below or equals this break point are put into said category. This means that all records, where the measurement is above the largest break point are put in the last category.

`position` is used to indicate the order of creation for subcategories. Personal information with a lower value in `position` are deemed more distinctive and the dataset will be split first along these variables. The creation process is strictly hierarchical, so that in the example above the first divide of the data is done for the gender variable, only then weight is considered for subcategory creation. For a more detailed explanation on categories see section 3.2.

Performance Information

In analogy to the process for personal information described above, the list `performance_info` holds information about the performance itself (i.e. performance indicators). An example schema is shown in listing 3. As shown by the example, one key difference is that the structure is not a flat list but that it allows nesting values into each other. Again, same as for the personal information, the attribute `name` denotes the key in the dict by which the values can be accessed. Due to the arbitrary nesting there is a differentiation between two types of elements, leaf elements which contain the `desired_result` key and internal nodes (or non-leaf nodes) which contain the key `measurements`.

```
"performance_info": [
    { "name": "weight_carried", "desired_result": "high" },
    { "name": "time", "desired_result": "low" },
    { "name": "mistakes", "desired_result": "low" },
    {
      "name": "speed",
      "measurements": [
        { "name": "part1", "desired_result": "high" },
        { "name": "part2", "desired_result": "high" }
      ]
    }
  ]
```

Listing 3: Example description of performance information items in the schema instance.

Leaf nodes are characterized 1) by not having child-nodes in the `measurements` attribute and 2) by containing the `desired_result` element. `desired_result` can take two values: *high* or *low*. This indicates which kind of performance is considered more efficient, i.e. if high values are desirable or not. For example, carried weight will be considered a part of the performance where, generally, a higher value is desired. At the same time, the amount of mistakes made during the performance should stay low. This field encodes domain knowledge and is essential when comparing the records in section 3.3.

3.1.2 Parsing records

The following section describes the implementation of the schema in regards to parsing data into python classes that are used in the software component as basis for all other operations. The schema is parsed and the two lists described above are extracted. Then, for each dict that represents the record a new record is created and if an id is present, it is set on creation time. Next, all key-value-pairs are traversed and - if defined in the schema - added to appropriate list in the record object. Keys that are not defined in the schema are ignored (see algorithm 3.1 for a representation in pseudo-code). This provides two advantages. Firstly, as the separation between personal info and performance info is done upon loading the object, later on, no distinction on which value belongs to which category has to be done, since they already reside in their appropriate list. Secondly, it ensures that only values that are defined in the schema - for which a behaviour is defined - are being processed further down the line.

3.2 Categories and subcategories

In this section, the functionality of category creation and automatic sorting of records is discussed (see table 1.1). First, the definition of a category within the context of this work is given. The process of how subcategories create a tree-like structure that gets

Algorithm 3.1: Read arbitrary data into a performance record object

```
1  Function ParseRecords(data, schema):
       Data: Input data is a dict representing a record of a performance and a
             Schema with the predefined structure
       Result: Output is a Object representing the record with measurements
               sorted into lists depending on their definition in the schema
2      personal_infos ← schema.personal_infos
3      performance_infos ← schema.performance_infos
4      id_name ← schema.id
5      id ← data.get(id_name)
6      new_record ← RecordObject(id=id)
7      foreach measurement m in data do
8          if m ∈ personal_infos then
9              new_record.personal_infos ← new_record.personal_infos.append(m)
10         else if m ∈ performance_infos then
11             new_record.performance_infos ←
                new_record.performance_infos.append(m)
12         else
13             throw error
14         end
15     end
16     return new_record
17 end
```

more and more refined the deeper the level. Secondly, it is evaluated how the creation order of categories may affect the resulting data structure. And lastly, implementation details are given, showing how the theoretical definitions are implemented in the software component.

3.2.1 Defining (sub)categories

Categories within the context of this work are collections of performances. The grouping is based on the individuals' personal information. As the personal information describes the baseline of each individual, grouping the performances into similar clusters increases comparability within the specific cluster (category of performances). Similar to age groups in competitions, the more similar the baseline is, the more expressive a comparison between two or more performances can be. While this may not be true in general (the personal information could be unrelated), within the context of this theses it is generally assumed that experts will select only those that are related to the outcome of the performance.

As the goal is to have as meaningful a comparison as possible, the categories should be as homogeneous as possible. The similarity within the category ensures that the underlying

factors have a similar effect on all the individuals whose performances are compared. For example, people with higher body weight also tend to be stronger, thus it makes more sense to compare people of similar weight. In order to facilitate as similar baselines as possible within any given category that is compared, each personal information can be used to split the data into subcategories.

When creating categories the initial dataset can be split into multiple different categories, at least two - since there should be some kind of differentiation - per personal information recorded. If a category is defined by a single variable the individuals are placed into multiple categories at the same time, as they might be of a certain height but also a certain weight. While this behaviour results in multiple categories, the similarity of the individuals inside of them remains quite heterogeneous as only a single variable defines their similarity. Not processing the categories further would increase comparability slightly but would omit a lot of possible information and chance to create more homogeneous categoryings. In order to enable more granular categories, thus improving comparability the categories need to be arranged in a tree-like structure. This means that multiple, different variables are used when creating categories. For example instead of creating the categories *male* and *above 60kg*, one might compare *all men above 60kg* - omitting males below 60kg. This newly created category's members are similar in not only one but two variables. The members of this category are the intersection of *male* and *above 60kg*, this drill-down into more and more specific categoryings can be extended as long as there are variables left to category by - selecting ever more homogeneous members.

While it is desired to have as homogeneous categories as possible, it is important to leave *some* heterogeneity - else the most homogenous categories would all only have a single member. Since comparing a single performance to itself will not result in a lot of knowledge, a certain context must be provided for each performance to be compared to. In other words, in order to be useful categories must have a certain minimum size. This minimum size may vary greatly depending on the domain, but nonetheless one needs at least two items in order to compare them. If a category does not meet the required minimum size their it should not be created, and the members should remain in the top-category, the next higher parent-node of the would-be category in the tree.

As an example, assume there is are performances with two personal information variables recorded for each individual, for example *weight* and *gender*. This means that the potential subcategories can be created with these variables. Gender will be split into *male* and *female*, while weight may be split along certain breaking points, e.g. 60kg and 80kg. Using only these two personal information variables means the individuals, staying on one level, could be sorted into five possible categories: *male*, *female*, *below 60kg*, *above 60kg and below 80kg* and *above 80kg*. With this setup however, a male individual with 70kg would have to be in two categories at the same time, namely *male* and *above 60kg and below 80kg*.

Using set notation the categories can be described as follows:

$$\begin{aligned} male &= \{\text{person} \mid \text{person is male}\} \\ female &= \{\text{person} \mid \text{person is female}\} \\ +80 &= \{\text{person} \mid \text{person weight is over 80kg}\} \\ 60-80 &= \{\text{person} \mid \text{person weight is between 60kg and 80kg}\} \\ -60 &= \{\text{person} \mid \text{person weight below 60kg}\} \end{aligned}$$

The sets described above contain all performances twice, since a person is both either male or female and weighs either under 60kg, between 60kg and 80kg or over 80kg. Using such a horizontal or flat division of categories would lead to many different categories. In each category individuals would share only a single baseline-trait (e.g. gender). This would not lead to the increased comparability that is desired. Instead the categories can be arranged as subsets. This is similar to asking *compare all male persons that weigh less than 80kg, but more than 60kg*. Using the set notation it can be described as follows:

$$\begin{aligned} male\_80+ &= male \cap +80 \\ male\_60-80 &= male \cap 60-80 \\ male\_60- &= male \cap -60 \\ female\_80+ &= female \cap +80 \\ female\_60-80 &= female \cap 60-80 \\ female\_60- &= female \cap -60 \end{aligned}$$

Visually, the concept can be represented as a tree. The top most *root* node would contain all performances without any differentiation. On the second level the data is then split by gender, into the two categories *male* and *female*. On the third and last level, the two gender-based categories are again split into the weight-based categories *below 60kg*, *between 60kg and 80kg* and *above 80kg*. A visual representation of the above can be seen in figure 3.1.

3.2.2 Creation order of categories

In the above example the order of selection did not matter, as the minimum size requirement was not taken into consideration. No change occurs if one selects all the woman from the category below 60kg ($female \cap -60$) or all woman and then select only those under the weight limit ($-60 \cap female$). However, the software component, created in this thesis, uses hierarchical categories with a minimum size, so the order of category creation directly influences the final categories produced. As the requirements for continuously adding new performances and supporting potentially hundreds of personal

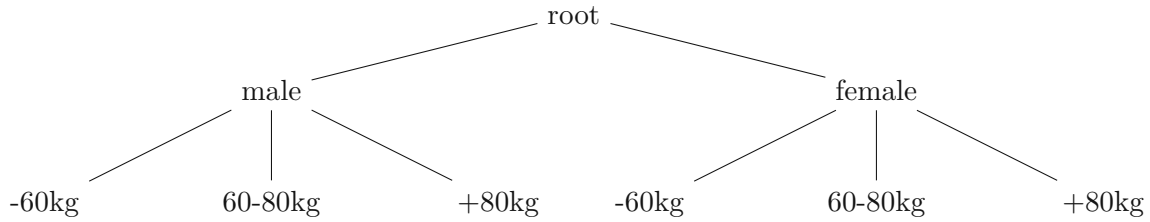


Figure 3.1: Tree-structure of possible categories resulting from using two personal information.

information items, the benefit of the hierarchical approach is that only the leaf nodes would need to be revisited and checked for new categories that can be created based on the new information.

Under the assumption, that for the above example the minimum size for a category to be created is five, then if a subcategory does not meet the required number of members, it will not be created and the members remain in the next closest parent node. Assuming the following cardinalities $|female| = 10$, $|male| = 4$, $|-60| = 7$, $|60-80| = 7$ and $|+80| = 0$, meaning there exist 2 men between 60kg and 80kg, 2 men below 60kg and similarly 5 women below 60kg and 5 women between 60kg and 80kg.

	Weight (kg)		
Gender	-60	60-80	+80
Male	2	2	-
Female	5	5	-

Table 3.1: Table of persons split in gender and weight.

If the gender-based categories have a lower position than weight (thus being created first) the category men is not created, resulting in a whole subbranch not being created and four records remaining in the root category. If weight-based categories are created in the first step, then two leave nodes - one per weight category - are missing. A visual representation is given in figures 3.3 and 3.2, the gray-dotted lines indicate categories that have not been created, while the number in brackets show the current members of the category.

The tree in figure 3.2 showcases the resulting structure, when the gender-based categories are created first and the weight-based later. Since the four remaining male performances do not meet the minimum size of five, the according category does not get created and they remain in the root category - indicating that they are not sorted into a category. The tree in figure 3.3 showcases the structure that results from the inverted creation order, weight categories first then split into gender categories. The structure is quite different, as while there are no performances in the root node, the male performances are split into the weight categories (two each), which in turn only create the female subcategory since the male subcategory does not meet the required minimum size of

five. The category $-60kg$ is shown with two members in the figure, which does not meet the defined minimum size. However, the members of the subcategory $female$ also weigh under 60kg, placing them in the $-60kg$ category, which is one level lower in the hierarchy - but still part of the $female$ category.

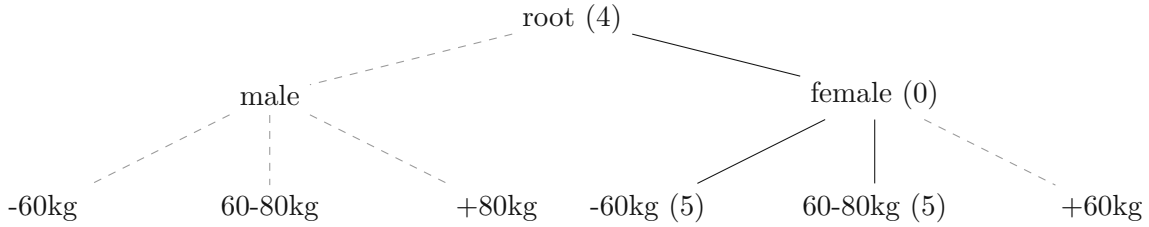


Figure 3.2: Tree-structure resulting from creating gender based categories first.

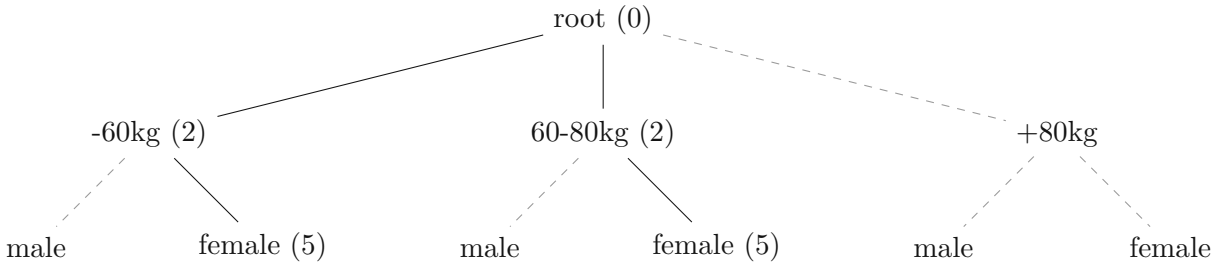


Figure 3.3: Tree-structure resulting from creating weight based categories first.

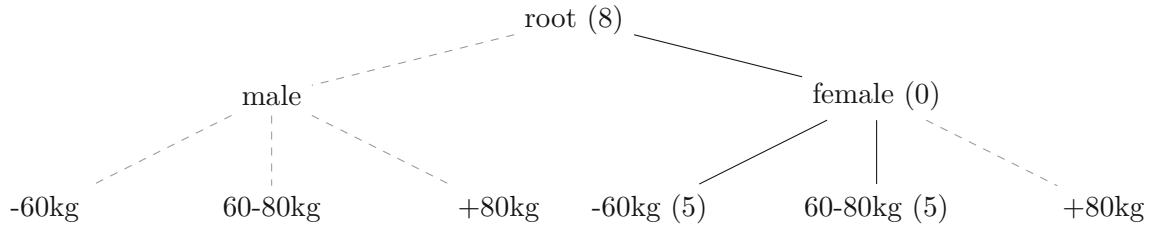
3.2.3 Updating categories

Since it is expected that new records will be added to the dataset over time, the categories are not static. However, since records will not be removed once created, categories can not be removed. When adding new records the overall amount of records increases and as such new categories may be created. When a new record (or multiple at the same time) is added, they are added to the root category. After that the records can be resorted, the procedure for this will be explained in more detail in section 3.2.4. One step per level in the tree can be observed. Keeping in line with the above example (see table 3.1), assume that four men with a weight between 60kg and 80kg are added to the dataset, starting out in the root node.

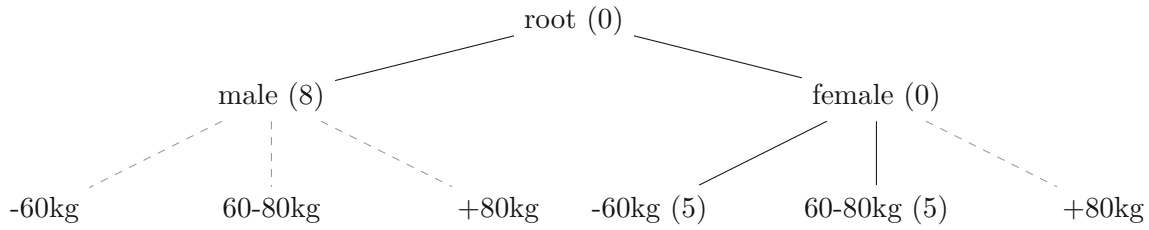
	Weight (kg)		
Gender	-60	60-80	+80
Male	2	6	0
Female	5	5	-

Table 3.2: Table of persons split in gender and weight.

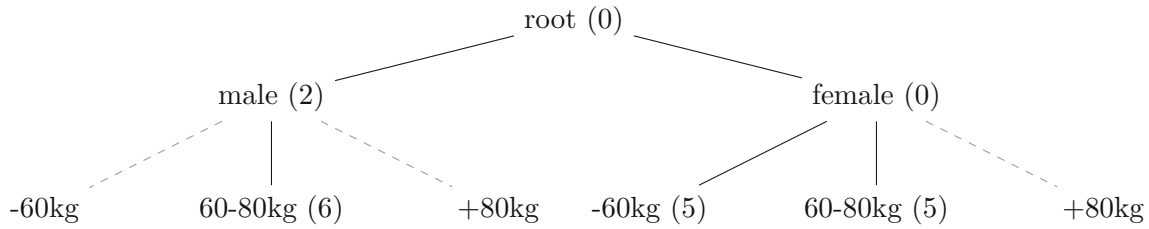
First, the sorting of the data, when gender categories are created first is examined, see figure 3.4 for a visual representation. In step 1, the four new records are added to the root node, there now being a total of eight male performances in this node. In step 2, the previously unavailable category male - since five is the minimum size - is now available and created, all performances from the root node are moved. Lastly, in step 3 the male category is again split by weight, the new subcategory 60-80kg is created and six records are moved into it. Since there are still not enough males below 60kg, the two performances from the initial creation stay in the male category.



(a) Step 1: records are added to the root node



(b) Step 2: records are moved into the respective gender category



(c) Step 3: records are again split into weight categories

Figure 3.4: Example of the three steps occurring when adding new records to the gender-based sorting tree.

Secondly, the steps occurring for the sorting order where weight categories are created first are shown in figure 3.5. Again, in step 1 the new performances are added to the root node. In step 2 the new performances are assigned to their appropriate weight category +60. In the last step 3 the new male subcategory is created in the +60 weight category, since there are now more than five members, and the performances are moved once more.

The update behaviour shows the heterogeneous outcomes caused by different prioritization (i.e., 'position') of personal information. While here the overall shapes stayed the same,

with three leave nodes in the end. It is important to keep in mind that with possible hundreds of personal information items and even more categories resulting from that the overall shape of the tree could vary greatly just based on the different order in which the categories are created.

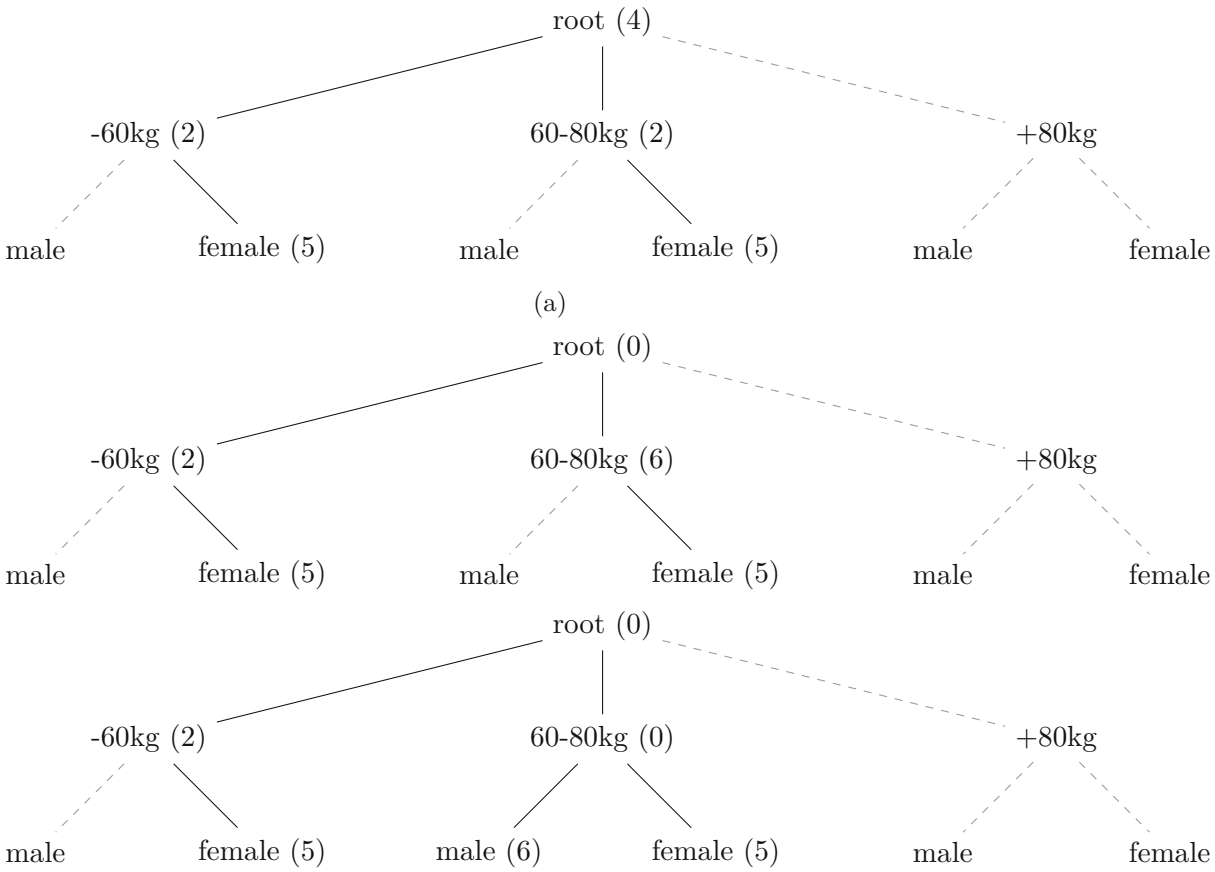


Figure 3.5: Example of the three steps occurring when adding new records to the weight-based sorting tree.

3.2.4 Implementation of categories

Algorithms 3.2,3.3,3.4 (split for readability) portray the algorithm used to create subcategories. The function used to do so is recursive. Since the object of each category holds itself subcategories, a tree-like structure - as discussed above - is created. This data-structures can have an arbitrary depth and on each level an arbitrary width, depending on the number of breakpoints and subcategories that meet the minimum size requirement. This structure lends itself to an recursive approach since the same operation has to be performed on every node, which in turn creates new nodes (on which the operation has to be performed again).

The function gets the root-node as a parameter, a flat list of personal information items defined by the schema, a minimum size that has to be met for a subcategory to be created and the current index. The list of personal information items to iterate over is already sorted according to the user defined preferences (see section 3.1), this ensures that the order of creation - discussed above - is the one that is expected. Typically the iteration will be started on the root node. Using the index, which is used to track the recursion depth, it is checked if there are any remaining categories that need to be processed. Each step in the recursion will - if categories are created - add a new level of depth to the tree. See part one of the algorithm 3.2.

Algorithm 3.2: Part 1 - Recursively create subcategories

```
1 Object category:
2 |   records: list of performances
3 |   subcategories: list of category object that are its subcategories
4 end
5 Function Recursivecategories(category, personal_infos, min_size, index):
  |   Input:
  |   category: object representing the current category.
  |   personal_infos: ordered list of categories to be created via a personal
  |   information.
  |   min_size: minimum size for a category.
  |   index: number tracking the current personal information being processed.
6 |   if index ≥ length of personal_infos then
7 |   |   return
8 |   end
9 |   current ← personal_info[index]
```

If a new category is still to be processed, the defining personal information object is selected from the array. The break points it defines (see section 3.1) are accessed and sorted ascending by value. Sorting the break points allows the algorithm to loop through them and move all performances with measures below the current break point into a new category list. The next break point that is processed - which is larger than the previous one - can then again simply remove all the performances from the list which are smaller then it. Lastly, the remaining measurements of the current performances are larger than any break point, they create the last category, e.g. +60 if the break point is 60. See part two of the algorithm 3.3.

In the third part of the algorithm, the map of records is being processed. An empty list is initialized to hold records that will be removed later on. Then, for each map entry - containing a subcategory and records that are within that subcategory - it is checked whether the subcategory exists already within the category. If this is the case, the records in the current map object are simply moved to the existing subcategory. Additionally, the records are also marked to be removed from the current category, by

Algorithm 3.3: Part 2 - Recursively create subcategories

```
10
    /* Break points are sorted by value so that it is
       possible to iterate over them and only check if a
       record is smaller than the current break point     */
11  break_points ← sort(current.split_value)
12  results ← empty list
       /* So that the original is not changed before the min
    size is checked */
13  data ← copy of records in category
14  for bp ∈ break_points do
15      current_range ← empty list
16      for record ∈ category.records do
17          measurement ← record.get_measurement(current)
18          if measurement ≤ bp then
19              remove record from data
20              current_range ← current_range + record
21          end
22          add the breakpoint and its records, in current_range, to the result map
23      end
24  end
    /* Data that has not been assigned to any break point
       since it exceeds the largest one                   */
25  if length of data > 0 then
26      results ← results + data
27  end
```

adding them to the 'to be removed'-list. On the other hand, if the subcategory does not already exist, it is checked if the minimum size for creation is satisfied. When this is the case, a new category-object is initialized with the associated records. Additionally, the records are again marked as to be removed and the newly created object is appended to the subcategories of the current category object. After the records are sorted, the records which where marked as to be deleted, are removed from the current category object. Finally, the algorithm is recursively called on all the subcategories in the current category object. When doing so the index is incremented by one, this ensures that the next personal information is processed creating the next category type - one level lower in the tree. See part three of the algorithm 3.4.

Algorithm 3.4: Part 3 - Recursively create subcategories

```
28
29      records_remove ← empty list
30      for entry ∈ results do
31          if entry exists in categories subcategories then
32              add records of entry to existing subcategory
33              records_remove ← records_remove + entry's records
34          else if length of res ≥ min_size then
                /* create a new object and append it to the
                   current one                                    */
35              category.subcategories ← category.subcategories +
                 category(records=res)
36              records_remove ← records_remove + entry's records
37          end
38      end
        /* Remove only records that have been put into a
           subcategory                                            */
39      category.records ← category.records - records_remove
        /* Recursively call the function on all subcategories  */
40      forall subcategory in category.subcategories do
41          Recursivecategories(subcategory, personal_infos, min_size, index + 1)
42      end
43  end
```

3.3 Calculating a score for each performance

The following section addresses the creation of scores and rankings of performances in regards to their effectiveness. The different approaches chosen to be examined will be explained in detail, alongside the desired properties that are expected of a ranking system. This section deals with the realisation of feature 1 defined in table 1.1.

After performances are recorded, verified and a schema for them has been defined, they can be evaluated, optionally they may be sorted into categories first. However, in the literature multiple different approaches to ranking (sports) performances can be found. Most of these approaches are evaluated in a narrow domain, focusing on the resulting ranking rather than the generalized usability of the used ranking method.

One of the main goals of the prototype is to provide a sensible ranking for many different kinds of performances. As such, one main contribution of this thesis is to evaluate different approaches and methods that can be used to create a ranking for a given set of performances.

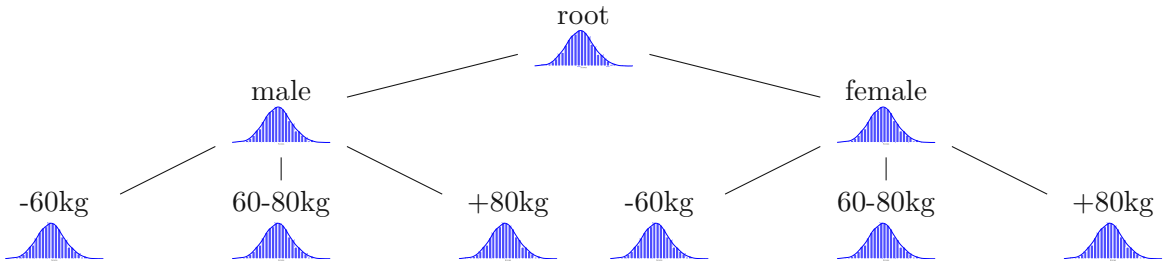


Figure 3.6: Visualization of separate, normally distributed, scores for each subcategory. Here it is assumed that every subcategory contains members. The same (theoretically possible) subcategories were used as explained in section 3.2.

3.3.1 Requirements for the scoring methods

As performances need to be classified as *better* or *worse* in relation to each other, they need to have a numerical value denoting their effectiveness in achieving a certain shared goal. The goal is to express the effectiveness of each performance in a single numerical value. These values can be compared to each other easily and a ranking can be created based on these comparisons. For such a ranking, the most effective performance shall be assigned the first position (i.e. rank 1) and the worst - or least effective - performance the last rank. These ranks are assigned based on the scores calculated by the individual methods, that will be defined in this section. Scores are created for each subcategory separately for each subcategory, resulting in an independent ranking for every single subcategory and its members. Figure 3.6 shows the conceptual idea of having multiple separate scores in each subcategory. Each of these resulting scores should ideally be normally distributed.

For the following chapter these definitions are given:

- $\mathbf{P}$ is the totality of the performances in a single category.
- $\mathbf{V}$ contains all the variables measured for the performances in $\mathbf{P}$. Unless stated otherwise $\mathbf{V}$ only contains performance indicators.
- For n performances, each with m measurements, a $\mathbf{X} = n \times m$ matrix can be created, where each row $x_i \in \mathbf{X}$ contains a performance from $\mathbf{P}$ and $x_{i,j}$ denotes the variable $v_j \in \mathbf{V}$ measured for the specific observation in row i.
- $\mathbf{S} = |\mathbf{P}| \times 1$ contains the scores generated from the performances in $\mathbf{X}$. The value $s_i \in \mathbf{S}$ denotes the score for performance p_i.

The function creating the score $s_i \in \mathbf{S}$ of performance $p_i \in \mathbf{P}$ is denoted as $S(p_i)$. For simplification reasons in this chapter - unless stated otherwise - only the *performance information* is considered, while the *personal information* is ignored. The function $S()$ should fulfill the following properties:

- $\exists S(p) = s, \forall p \in \mathbf{P}$, the function $S()$ can calculate a score for any given performance p, that follow the same given schema.
- $p_i = p_j \rightarrow s_i = s_j$, if two performances p_i and p_j contain the exact same measurement values, then also their computed scores $S(p_i) = s_i$ and $S(p_j) = s_j$ are the same.

After a score s_i is assigned to each individual performance p_i, the scores can be ranked. Ranking the performances puts them in order by their score value and assigns the resulting position as rank, i.e. the p_1 has rank 1, p_2 rank 2 and so forth. Whether the best performance is assigned the lowest or the highest score depends on the chosen scoring method.

3.3.2 Scoring methods

In the following section the scoring methods that are compared and implemented in this work are explained in detail. Multiple different methods can be used to derive the score s_i for a performance p_i. As above, the data matrix takes the form $\mathbf{X} = n \times m$, where the rows $x_{i,\cdot}$ denote the performances $p_i \in \mathbf{P}$ and the columns $x_{\cdot,j}$ the measurements $v_j \in \mathbf{V}$.

The $desired - results$ vector encodes the domain knowledge from the schema as defined in section 3.1. If a larger value v_j indicates a more effective performances then $desired - results_j = 1$ or simply $high$, else if a lower value is associated with a more effective performance the value at $desired - results_j$ is low or -1.

Principal Component Analysis

Principal component analysis (PCA) is a tool used in many different disciplines. It is primarily used for reduction of dimensionality by identifying the most meaningful way to re-express a given dataset. The expected effect is noise reduction, revealing a structure in the data that was previously obscured [28].

PCA achieves the reduction by creating linear combinations of the data that contain maximum variance while being uncorrelated with previous linear combinations [22]. These linear combinations are called *principal components*, and are created using the eigenvectors of the covariance matrix [22]. In this work for the purpose of scoring the first principal component will be used to assign a score to each observation.

Factor Analysis

The basic idea behind Factor Analysis (FA) is that correlations exist for multiple observed variables that are influenced by unobserved variables, called factors [3]. These factors provide a link between observable variables and unobservable (latent) variables. The goal is to determine how many latent variables are needed to explain the correlations between the observed variables. These latent variables can then be interpreted and used to predict latent variables given changes in observed values.

For the purpose of this work a single factor is assumed and used to score a performance - similar to PCA.

Min-Max Distance Measure

The *min-max-distance measure* 3.1 (min-max) computes the score by comparing each measurement to the best value found in the category - where best depends on the given schema and can either be the lowest or the highest observation. For each single measurement $x_{i,j}$ the distance to the best occurring instance of v_j is summed up. If the desired value is low then the minimum is taken as the best value, else if high the maximum is used, denoted by the $extreme(v_j)$ method. This method compares the performance to the best possible performance that can be constructed given the available data. While ignoring some real world constraints (such as interdependence in variables making some combinations unobtainable, or humanly not possible combinations of values), this function gives the distance to a theoretical optimal performance.

$$
\begin{aligned}
s_i &= \sum_{j=0}^{m} |extreme(v_j) - x_{i,j}| \\
extreme(v_j) &= \begin{cases} max(v_j), & \text{if } desired - results_j \text{ is } high \\ min(v_j), & \text{if } desired - results_j \text{ is } low \end{cases}
\end{aligned}
\tag{3.1}
$$

Pairwise Distance Measure

The *pairwise-distance measure* 3.2 compares all performances to each other pair by pair, measurement by measurement. When comparing the measurements of two performances p_i and p_j, it is checked first if the desired value is high or low for each measurement, and then which of the two performances performs better in regards to this. The 'winning' measurement adds a point to the score s_i of the performance p_i, while performing worse than p_j deducts a point from the score.

$$
\begin{aligned}
s_i &= \sum_{j=0; i \neq j}^{n} \sum_{k=0}^{m} cmp(x_{i,k}, x_{j,k}) \\
cmp(x, y) &= \begin{cases} 1, & \text{if } desired(v) \text{ is } high \text{ and } x > y \\ -1, & \text{if } desired(v) \text{ is } high \text{ and } x < y \\ 1, & \text{if } desired(v) \text{ is } low \text{ and } x < y \\ -1, & \text{if } desired(v) \text{ is } low \text{ and } x > y \\ 0, & \text{else} \end{cases}
\end{aligned}
\tag{3.2}
$$

Linear Combination

Linear combinations (LC) is a very intuitive method of calculating scores are . For many types of games the end result can be represented as a sum of individual sub-scores, e.g.

scores per round or mistakes per round. Domain knowledge then determines if a higher score or a lower score is better. The following equation 3.3 is used to calculate a linear combination that can be used as the score of a performance.

$$\begin{aligned} s_i &= \sum_{j=0}^{m} (d_j * x_{i,j}) \\ d_j &= \begin{cases} 1, & \text{if } desired_j \text{ is } high \\ -1, & \text{if } desired_j \text{ is } low \end{cases} \end{aligned} \tag{3.3}$$

Ranking Principal Curve

Principal Curves are an extension of PCA and can be used to order observations [30] (the method will be referenced to as RPC in the following sections). Based on Bézier curves (see section 10) unsupervised rankings can be implemented [30]. The idea behind this unsupervised approach is to minimize the error when fitting a Bézier curve through a given dataset. By adapting the control points $\mathbf{P}$, the curve is fitted to the data in an iterative fashion until convergence.

Li et al. define the RPC curve as following:

$$f(s) = \sum_{r=0}^{3} B_r^3(s) p_r, \;\; s \in [0,1] \tag{3.4}$$

Equation 3.4 can be formulated as a nonlinear optimization problem, where $J(\mathbf{P}, s)$ denotes the difference between the optimal solution and the calculated solution for the given control points $\mathbf{P}$ [32].

$$J(\mathbf{P}, \mathbf{s}) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{PMz}_i\|^2 \tag{3.5}$$

Next, equation 3.6 determines the points s_i on the curve that have the minimum residuals when reconstructing x_i by $f(s_i)$.

$$\begin{aligned} &\left(\frac{\partial \mathbf{PMz}}{\partial s}\right)^T (\mathbf{x}_i - \mathbf{PMz})\Big|_{s=s_i} = 0 \\ &\mathbf{s} = (s_1, s_2, ..., s_n), \mathbf{z}_i = (1, s_i^1, s_i^2, ..., s_i^k)^T \\ &\mathbf{P} \in [0,1]^{d \times 4}, s_i \in [0,1] \\ &i = 1, 2, ..., n \end{aligned} \tag{3.6}$$

Local minimizer $(\mathbf{P}^*, \mathbf{s}^*)$ can then be found iteratively by equations 3.7 and 3.8. Here t indicates the t-th iteration during the optimization process.

$$\mathbf{P}^{(t+1)} = \arg\min \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{PMz}_i^{(t)}\|^2 \tag{3.7}$$

$$\left(\frac{\partial \mathbf{PMz}}{\partial s}\right)^T (\mathbf{x}_i - \mathbf{PMz})\Big|_{s=s_i^{(t+1)}} = 0 \tag{3.8}$$

When associating $\mathbf{X}$ with $\mathbf{Z}$:

$$\mathbf{Z} = \begin{pmatrix} 1 & & 1 & \dots & 1 \\ s_1^1 & s_2^1 & s_3^1 & \dots & s_n^1 \\ s_1^2 & s_2^2 & s_3^2 & \dots & s_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_1^k & s_2^k & s_3^k & \dots & s_n^k \end{pmatrix} = (\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n) \tag{3.9}$$

Then equation 3.4 can be rewritten in a matrix form:

$$\begin{aligned} J(\mathbf{P}, \mathbf{s}) &= \|\mathbf{X} - \mathbf{PMZ}\|_F \\ &= tr(\mathbf{X}^T\mathbf{X}) - 2tr(\mathbf{PMZX}^T) + tr(\mathbf{PMZZ}^T\mathbf{M}^T\mathbf{P}^T) \end{aligned} \tag{3.10}$$

$$\frac{\partial J}{\partial \mathbf{P}} = 2\left(\mathbf{P}\left(\mathbf{MZ}\right)\left(\mathbf{MZ}\right)^T - \mathbf{X}\left(\mathbf{MZ}\right)^T\right) = 0 \tag{3.11}$$

Which leads to the solution for the minimum point of equation 3.5:

$$\mathbf{P} = \mathbf{X}\left(\mathbf{MZ}\right)\left((\mathbf{MZ})(\mathbf{MZ})^T\right)^+ = \mathbf{X}(\mathbf{MZ})^+ \tag{3.12}$$

Since 3.10, 3.11 are computationally expensive, Li et al. [30] give the following equation 3.13, using a preconditioner $\mathbf{D}$. The L2 norms of the columns of $(\mathbf{MZ}^{(t)})(\mathbf{MZ}^{(t)})^T$ are used in the diagonal matrix $\mathbf{D}$.

$$\mathbf{P}^{(t+1)} = \mathbf{P}^{(t)} - \gamma^{(t)}\left(\mathbf{P}^{(t)}\left(\mathbf{MZ}^{(t)}\right)\left(\mathbf{MZ}^{(t)}\right)^T - \mathbf{X}\left(\mathbf{MZ}^{(t)}\right)^T\right)\mathbf{D}^{-1} \tag{3.13}$$

With $\lambda_{min}^{(t)}$ and $\lambda_{max}^{(t)}$ being the min/max eigenvalues of $(\mathbf{MZ}^{(t)})(\mathbf{MZ}^{(t)})^T$ $\gamma^{(t)}$ is a parameter that is set to ensure convergence of the sequence $\mathbf{P}^{(t)}$. Li et al. suggest setting it to the result of equation 3.14.

$$\gamma^{(t)} = \frac{2}{\lambda_{min}^{(t)} + \lambda_{max}^{(t)}} \tag{3.14}$$

A basic implementation of the RPC-learning approach can be seen in algorithm 3.5. $\mathbf{P}^{(0)}$ is initialized using *desired*, where $p_0 = \frac{1}{2}(1 - desired)$ and $p_0 = \frac{1}{2}(1 + desired)$. p_1 and p_2 are initialized using random points in the input set $\mathbf{X}$.

Two adaptions were made to the approach proposed by Li et al.. Firstly, due to limitations in the implementation, an implementation [1] of Brent's method [5] was used, instead of Golden Section Search, to find the root of the given equation 3.11. For similar reasons the preconditioner $\mathbf{D}$ in equation 3.13 was omitted. Despite these adaptions the algorithm was found to perform reasonably well.

Algorithm 3.5: RPC Learning Algorithm

input : $\mathbf{X}$, dataset to be ranked
input : ξ, small positive number
output : $\mathbf{s}^*$, scores of $\mathbf{X}$ as a vector

Initialize $\mathbf{P}^{(0)}$
while $\Delta J > \xi$ **do**
  Use a minimizer to adapt $\mathbf{s}^{(t)}$ from equation 3.8
  Compute $\mathbf{P}^{(t+1)}$ using equation 3.13
  if $\Delta J < 0$ **then**
    break
  end
end

Self Organizing Maps

As discussed in section 1.3.3 Self Organizing Maps (SOM) are Artificial Neural Networks. SOMs have been combined with genetic algorithms in order to create rankings of multidimensional data [35].

The base approach is as follows:

1. Initialize a network/map in a $1 \times x$-shape, where x is a multiple of the amount of observations that are given.
2. Randomly generate the initial population of chromosomes with values, that are within a range informed by an expert.
3. While the resulting distribution is not of adaptable quality:
   3.1. Set the values of the first and last neuron to be the values of the best and worst performance. The other neurons are initialized randomly.
   3.2. Create a new set of distances (chromosomes) via the genetic algorithm.
   3.3. Train the SOM on the provided samples and the current recombination of distances.
   3.4. Calculate how closely the distribution of rankings resembles the targeted distribution.
4. Train a significant number of networks with the chosen distance and generate a final ranking by means of the average of the ranking position in all trained networks.

In a standard SOM, the squared euclidean distance is usually used to quantify the similarity between input and neurons [35]. As such, the distance is given as:

$$d^2 = \sum (E_i - w_i)^2$$

E_i and w_i are the values of the i-th attribute of the input E and neuron w respectively.

For the purpose of this scoring method additional weights for each attribute are introduced. The weight vector is adapted by the genetic algorithm in order to modify the resulting distribution of scores to be closely related to a desired distribution. β_i is the weight of the i-th attribute, determining how much influence the attribute i has on the overall distance between E and w.

$$d^2 = \sum \beta_i (E_i - w_i)^2 \tag{3.15}$$

The genetic algorithm converges towards a β vector that will align the scores most closely with the chosen target distribution, which is defined by the fitness function (see section 6).

The map takes the shape of a vector, essentially being one row of neurons. The endpoints of this row are fixed and hold the values of the best and worst performance as defined by an expert. As SOMs cluster similar inputs near each other, performances that are similar in their effectiveness will activate neurons that are close to each other. The fixed endpoints then force the network to sort the remaining data between the best and worst performance. Due to the clustering behaviour of SOMs, performances closer to the top performance can be interpreted as being better. Likewise, performances closer to the bottom performance can be seen as worse. After training the SOM, the position of the neuron closest to the input can be seen as the score for the given input.

While the behaviour described above forces the SOM to sort the performances into a ranking, the added weight vector β influences the distribution (or placement) on the map. The fitness function used to guide the mutation of the β vector is maximized. In this case the fitness functions returns a value [2] that indicates how close to a normal distribution the resulting scores are.

3.3.3 Calculation of scores

All of the methods discussed in section 3.3.2 have been implemented in the prototype. In addition to the methods, two transformations are performed on the data. Firstly, the incoming data is transformed into a value range of $[0, 1]$. This is done by applying the following transformation to the data:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.16}$$

Secondly, after the scores have been calculated with one of the methods mentioned above, a power transformation [56] is applied to the data. This second transformation shifts the resulting scores closer to a normal distribution.

For methods that rely on random initialization (RPC and SOM), each time five sets of scores have been calculated, the average of these is returned.

3.4 Comparison between subcategories

In section 3.2 the intention and utility behind the creation of subcategories was explained. In addition, in section 3.3 the scoring and ranking of performances in a certain subcategory was explored. When splitting the dataset into multiple categories the followup question arises: *can performances be compared across different subcategories?*. Here the realization of feature 3 as defined in table 1.1 is discussed.

The basic premise of this question is that, for different physical preconditions, a direct comparison between performances often does not make sense. As one person's physique may be better suited for a certain task, that means their measured values will most likely be better than those of another person that has a less optimal physical precondition for the given task. One example for this is weight lifting, where a heavier person tends to be able to outperform a lighter person, which is why weight classes have been introduced. Different classes exist in many kinds of sports, since athletes are generally compared to other athletes of similar physique (e.g. weight-classes in boxing or lifting). If one wants to compare performances across different classes, the question comes up which person performed better in regards to their physical precondition.

So the question is not *which of two performers is better?* but rather *which of the performances is better, in regards to their given physical preconditions?*. To answer this, the performances (or their scores) can not be compared directly, as this would heavily skew results in favour of whoever has better physical preconditions for the given goal of the performance.

In order to roughly compare two performances with different physical preconditions they each need to be compared to other performances with similar physical preconditions first. This way a baseline is established and it can be seen which performance achieves a greater efficiency than others with the same physical preconditions. Then, in a second step, the two initially different performances can be compared to each other.

The comparison to the peer category is done by using the implementation of the scores explained in the previous section 3.3 (using the scoring method LC). After the comparison to the peer category is done, it can be quantified how each performance performed in regards to their respective peer category.

The basic approach when comparing within the category is to calculate the scores, and then evaluate how the score of a specific performance compares to an *average* performance in the category. The distance between a specific score and the average then show if a specific performance is above or below average. A positive distance to the average score indicates performing better than the average, while a negative distance indicates performing worse than the average[2].

Once each performance is scored in relation to its peers, the results can be compared and

[2]This assumes that the scoring method indicates better performances with a higher score, for other approaches the meaning of the distance is reversed.

ranked. A performance performs better than another if it outperforms its peers more than the other performance does.

3.4.1 Extending peer categories

The approach described above has one caveat, as - due to the implementation of the subcategories (see section 3.2) - if a record is the only member of a subcategory there are no peers to compare it to. To alleviate this problem, if a subcategory has fewer member than the minimum size (see section 3.1), the closest subcategory will be added to the pool of performances during the comparison. Algorithm 3.6 shows the process of comparing records and selecting the closest subcategory. For this extension of the peer category, the subcategory which is closest in the metric that splits the subcategories is used.

For example consider three possible weight categories $-60kg$, $60-80kg$ and $+80kg$, with a minimum size of 5. The current sizes of the categories are $|-60kg| = 6$, $|60-80kg| = 7$ and $|+80kg| = 2$. Since the category $+80kg$ does not meet the minimum size, their category would be extended with the the next closest category - in this case $60-80kg$.

Algorithm 3.6: Read arbitrary data into a performance record object

Data: list of ids for the chosen records that will be compared
Result: List of scores associated to the corresponding record id

```
result ← map of id and corresponding score
forall id ∈ list of ids do
    record ← record with the given id
    category ← subcategory of record
    peers ← performances in the subcategory
    if category has fewer members than minimum size for categories then
        categoryMetric ← current metric that defines subcategories, e.g. height or
          weight
        subcategory ← subcategory whose records are closest to the current
          record in the categoryMetric
        peers ← merge the current peers with the performances in selected
          subcategory
    end
    scores ← scores for all peers
    average score ← calculate the average from scores
    put difference between score of current performance and average into the
      result map
end
```

CHAPTER 4

Evaluation

4.1 Applying scoring methods

Evaluation was done using two different kinds of datasets. The first dataset contains 171 different countries around the world and ranks them based on four different properties. It is of note, that a preexisting ranking exists for this dataset. While the use case for the prototype assumes that there is no such thing as a ground truth that algorithms can be trained on, for evaluation only purposes this preexisting ranking provides a sense of scale on how close the performances are to the 'true' ranking.

The other dataset has been created randomly. The intent of this smaller set is to be broken down into categories and to evaluate on the one hand the behaviour of the methods when applied to the total dataset (all performances) and on the other hand their behaviour when being applied only to a subset of the total dataset.

4.1.1 Nonlinear Quality of Life Index

Nonlinear Quality of Life Index (NQLI) is a ranking that consists of four different properties [57]:

- GDP per capita by Purchasing Power Parities (GDP)
- Life expectancy at birth (LEaB)
- Infant mortality rate (IMR)
- Tuberculosis incidence (Tub)

Countries have been ranked in their quality of life using these four metrics. The general aim of this ranking is to reduce biases present in other quality of life rankings (such as

The Economist Intelligence Unit's quality-of-life index and The Life Quality Index) [57]. For this reason, the data has been ranked based on PCA, to ensure that there is no human bias in the ranking.

While the dataset is not from within the sports domain, it can be used to create rankings nonetheless. The absence of human bias and its size make this dataset a good candidate to evaluate the scoring approaches described earlier in this chapter. For this, the data has been extracted from the original paper and normalized. Subsequently, each scoring approach has been used to rank the dataset independently. This dataset is used especially to see how the algorithms behave on larger datasets. It can be checked how closely the resulting scores follow a normal distribution. The complete list of scores can be found in the appendix (table 1).

Data exploration: The dataset consists of 171 countries, with each being given four different metrics - non of which are undefined. As can be seen in table 4.1 in the rows *min* and *max*, the scale of the four different measurements varies drastically.

Table 4.1: Summary statistics about the input data for NQLI

	GDP	LEaB	Tub	IMR
count	171	171	171	171
mean	10496.9	67.229	65.988	41.129
std	12214.3	10.712	77.784	39.584
min	330	41.681	2	0
max	70014	82.27	422	165
norm	0	0	0	0

Figure 4.1 shows a box plot for each attribute. As one can see aside from LEaB, all of the attributes are spread out widely and contain outliers. The fact that LEaB does not contain outliers can be expected, as the possible value range is set by natural circumstances of human lifespan itself - setting an upper limit to the values. GDP contains one very high outlier of 70014, belonging to Luxembourg, which is nearly 150% of the next highest value (47551, Norway). Overall, the impact of the variables is fairly similar on the calculated score (NQLI), as can be seen in the correlations (table 4.2). As expected, higher GDP and LEaB have a positive influence on the quality of life, while higher infant mortality or tuberculosis incidence indicates a worse quality of life. Additionally, an - especially strong - negative correlation can be seen between LEaB and infant mortality rate. This can be expected as a high IMR is directly connected to lower average life spans. LEaB also has a high correlation with the overall score (NQLI), which again is to be expected, as longer life spans can indicate better living circumstances and access to medical care. Similarly, there is a strong negative correlation between the overall score and IMR.

The *norm*-row in table 4.1 contains the p-values of a Shapiro-Wilks test, applied to each input attribute separately. The point-null hypothesis of the test is that the given data

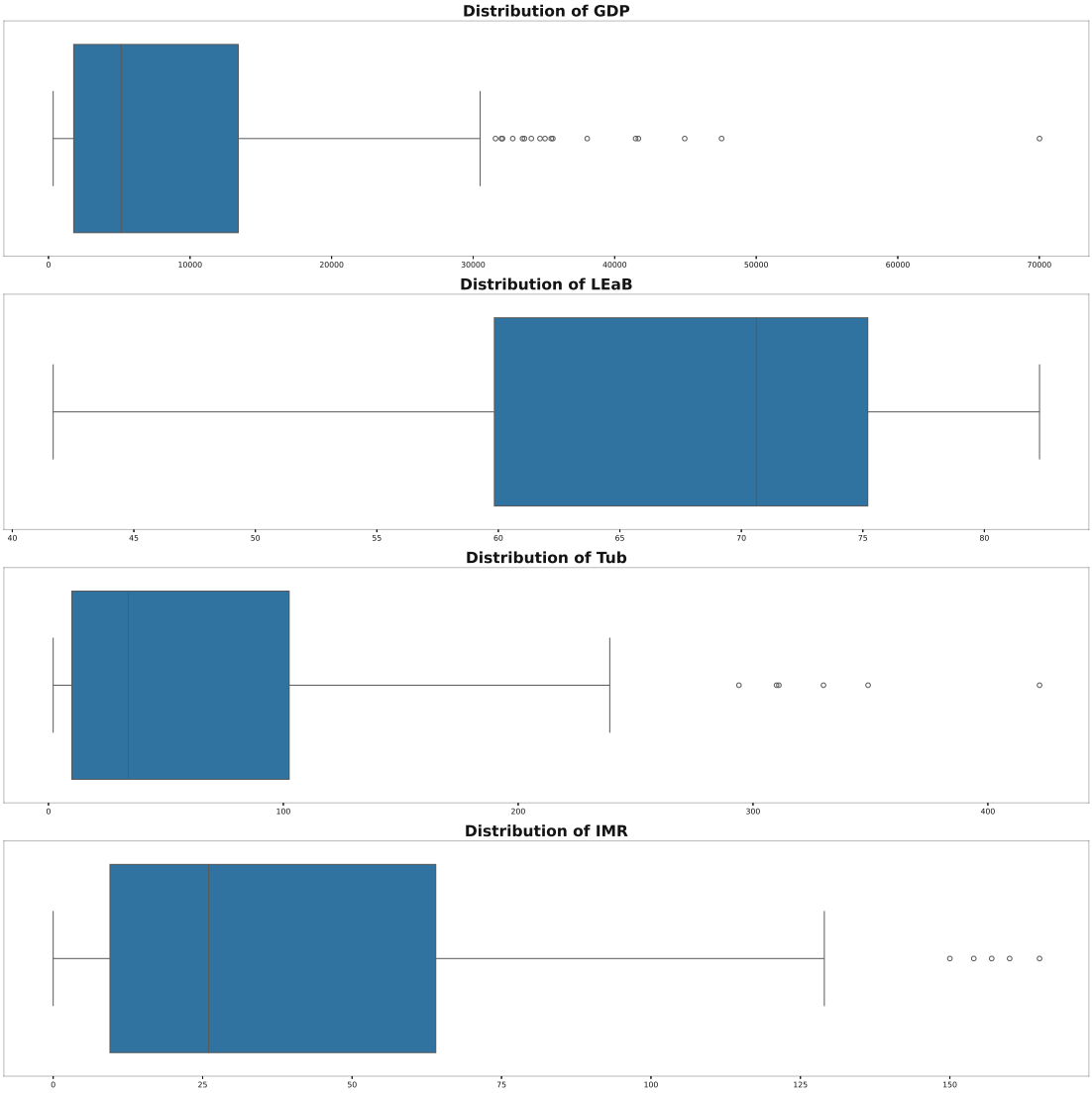


Figure 4.1: Box plot of each input attribute

Table 4.2: Correlation between input parameters as well as the calculated score

	GDP	LEaB	Tub	IMR	NQLI
GDP	1	0.65	-0.47	-0.58	0.8
LEaB	0.65	1	-0.8	-0.92	0.95
Tub	-0.47	-0.8	1	0.68	-0.84
IMR	-0.58	-0.92	0.68	1	-0.88
NQLI	0.8	0.95	-0.84	-0.88	1

was drawn from a normal distribution. All of the p-values were well below the chosen $\alpha = 5\%$ ($p = 0.05$) level. Thus, no support was found that the measurements follow a normal distribution. Additionally, the box plots in figure 4.1 show that most of the data is heavily skewed to one side. This indicates that there are few wealthy countries (with a high GDP), where people have longer lifespans on average. On the other hand, there are more countries with a high infant mortality and a high count of tuberculosis incidents.

Calculation of scores All the methods to generate a score, as described in section 3.3.2, have been used to rank the same data as the Nonlinear Quality of Life Index [57]. When comparing and evaluating these ratings, a direct replication of the NQLI is not the main goal. Rather, a sensible ranking should be the result of an appropriate ranking method. This, by extent, results in an overlap with the given NQLI. However, slight deviations are acceptable and do not pose a significant problem. As such, the evaluation of scores will be twofold. Firstly, the correlation (using Kendall's Tau) between them will be looked at. In other words, it will be assessed how similar the rankings are. Secondly, the first and last five countries in each ranking will be examined more closely. The tails (first and last five countries) are expected to be populated by outliers, since the countries which are significantly better or worse than the others are expected to be ranked accordingly. Therefore, strong deviations to the given NQLI could hint at the method not reacting correctly to these outliers. In other words, the method does not reliably identify very good or bad performances.

In figure 4.2, box plots are shown for all the generated scores. Furthermore, the same countries and their scores can be seen in table 2. The box plots show that the scoring algorithm *centered* the scores, in contrast to the input data. While the input data (figure 4.1) showed outliers in three of the four box plots, no single box plot shows values outside of the $1.5 \times IQR$ (interquartile range).

Table 4.3 shows the correlation between all of the ranking methods. It is apparent that most of the generated rankings have a very high overlap. Only the method based on SOM shows a strong divergence from the other methods, with a correlation lower than 0.1 to each of the other methods. Another unexpected result is the complete overlap of LC and min-max. Both methods are based on an unweighted linear combination. However, min-max requires knowledge about the overall set (the minimum and maximum values of each performance indicator), while LC does not. Thus, when adding new performances, min-max potentially has to be recalculated for all performances while LC can be calculated for the newly added performances only. This property makes LC more effective than min-max - given this equivalence holds true for other datasets.

Assuming a high correlation to the NQLI indicates a good ranking, LC and min-max can also be seen as producing the best rankings for the given dataset. However, as rankings produced by other methods are not too far off further testing is necessary to see which of the ranking methods is more appropriate.

Next, it will be looked at how the selection of the extremes in the ranking (i.e. the best and worst countries) compare. Table 4.4 contains the five best performing countries for

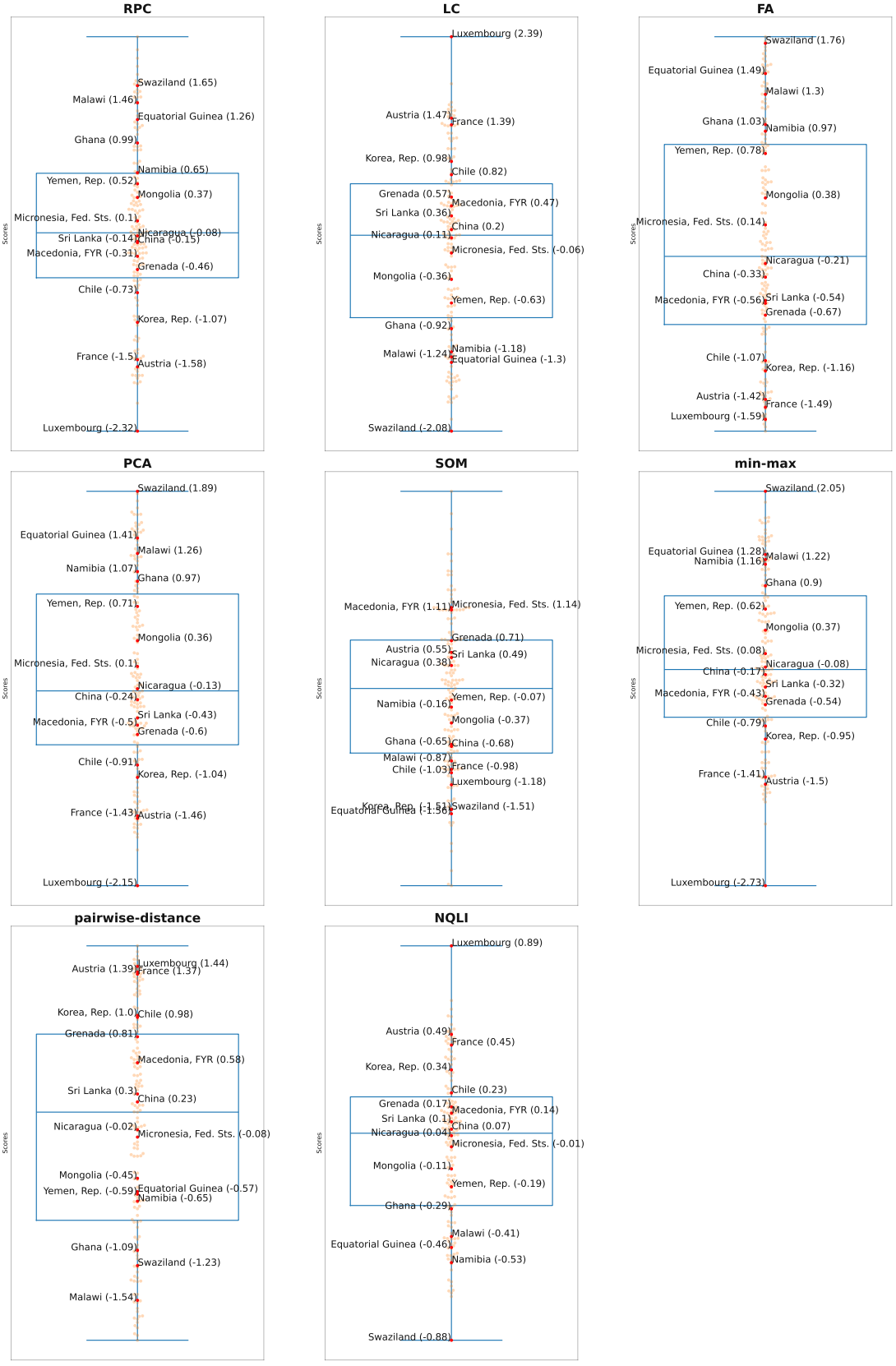


Figure 4.2: Box plots of generated scores. Additionally, every 10th (by NQLI-Ranking) country has been marked and its score written in brackets. Some methods inversed the ranking, where the highest score denotes the worst country.

Table 4.3: Kendall's Tau Correlation between the different ranking approaches

	RPC	LC	FA	PCA	SOM	min-max	pairwise-distance	NQLI
RPC	1	0.9	0.9	0.91	0.07	0.9	0.88	0.89
LC	0.9	1	0.92	0.97	0.05	1	0.92	0.96
FA	0.9	0.92	1	0.95	0.07	0.92	0.89	0.89
PCA	0.91	0.97	0.95	1	0.06	0.97	0.91	0.93
SOM	0.07	0.05	0.07	0.06	1	0.05	0.06	0.04
min-max	0.9	1	0.92	0.97	0.05	1	0.92	0.96
pairwise-distance	0.88	0.92	0.89	0.91	0.06	0.92	1	0.9
NQLI	0.89	0.96	0.89	0.93	0.04	0.96	0.9	1

each method, while table 4.5 contains the five worst performing ones. Each country's rank assigned by the NQLI, written in brackets next to it. As discussed above, Luxembourg is an outlier, especially for GDP. Thus, it can be expected that Luxembourg performs very well overall. Almost all methods rank Luxembourg as the best performing country, in accordance with the NQLI. The only two exceptions are pairwise-distance and SOM. The SOM method notably diverges, as no country in its top five was selected by any other method. On the other hand, *pairwise-distance* mostly selected countries in its top five that occur in other rankings - however, Luxembourg is still ranked highly in place eight. Other methods (RPC, LC, PCA, and min-max) ranked the same four countries in places one to five, even having the same countries in rank one and two as the NQLI-Ranking. Additionally, Singapore (rank four in NQLI) was ranked as third by these methods.

Table 4.4: The five best ranked countries per method

	1	2	3	4	5
RPC	Luxembourg (1)	Norway (2)	Singapore (4)	Iceland (7)	Switzerland (8)
LC	Luxembourg (1)	Norway (2)	Singapore (4)	Iceland (7)	Switzerland (8)
FA	Luxembourg (1)	Iceland (7)	Switzerland (8)	Australia (12)	Japan (18)
PCA	Luxembourg (1)	Norway (2)	Singapore (4)	Iceland (7)	Switzerland (8)
SOM	Ecuador (76)	Pakistan (123)	Bhutan (124)	Benin (125)	Lesotho (165)
min-max	Luxembourg (1)	Norway (2)	Singapore (4)	Iceland (7)	Switzerland (8)
pairwise-distance	Norway (2)	Iceland (7)	Switzerland (8)	Canada (9)	Sweden (15)
NQLI	Luxembourg (1)	Norway (2)	Kuwait (3)	Singapore (4)	United States (5)

Similar behaviour can be observed for the bottom five countries. Again, SOM has the most divergent ranking from any of the methods. It is noteworthy, that for SOM the top and bottom countries do not correspond in any way to the NQLI ranking, where they can be found in the 100-150 range. Other methods show significant tendencies to rank countries in accordance to the NQLI, most top five countries are from the top 10 in the NQLI ranking or from the bottom 10 respectively.

Overall, there is no obvious best performing method for this given dataset. However, it is of note that SOM did perform very poorly and that min-max has a strong disadvantage compared to LC, while giving the same ranking.

Table 4.5: The five worst ranked countries per method

	167	168	169	170	171
RPC	Central African Republic (158)	Afghanistan (159)	Zambia (164)	Lesotho (165)	Zimbabwe (170)
LC	Afghanistan (159)	Zambia (164)	Sierra Leone (169)	Zimbabwe (170)	Swaziland (171)
FA	Afghanistan (159)	Angola (162)	Zambia (164)	Sierra Leone (169)	Swaziland (171)
PCA	Afghanistan (159)	Angola (162)	Zambia (164)	Sierra Leone (169)	Swaziland (171)
SOM	Norway (2)	Iran (67)	Fiji (82)	Moldova (97)	Mauritania (136)
min-max	Afghanistan (159)	Zambia (164)	Sierra Leone (169)	Zimbabwe (170)	Swaziland (171)
pairwise-distance	Burundi (157)	Central African Republic (158)	Congo, Dem. Rep. (163)	Sierra Leone (169)	Zimbabwe (170)
NQLI	South Africa (167)	Djibouti (168)	Sierra Leone (169)	Zimbabwe (170)	Swaziland (171)

4.1.2 Boxing exercise

As a second dataset to evaluate a fictional boxing exercise has been created. The (semi-)randomly created measurements for each data point are as follows:

- **height:** The height of the person performing the exercise.
- **weight:** The weight of the person performing the exercise.
- **avg heart rate:** The average heart rate of the person during the exercise. This metric helps to track the overall fitness level.
- **avg punch impact:** The average force or impact of each punch thrown during the exercise. This metric measures the power behind each punch.
- **avg punch speed:** The average speed of each punch thrown during the exercise. This metric indicates how quickly punches are delivered.
- **number of punches:** Total number of punches that have been thrown during the exercise. This metric can be seen as an indicator for endurance for this specific exercise.
- **punches defended:** The number of punches successfully blocked or dodged during the exercise. This metric helps gauge defensive skills and reaction time.
- **punches missed:** The number of punches thrown that did not hit the target. This metric helps to evaluate accuracy and control.

Generating a test dataset

A test dataset of 20 observations with the above measurements has been created. In an initial step latent variables have been created, which then influence the performance indicators. The two latent variables are called *skill* and *physical*, which represent a persons boxing specific skill and general physical fitness status respectively. Both are in the interval of $[-1, 1]$, where a higher number represents a better fitness status or higher skill level. Likewise, a lower number indicates a worse physical status or lower skill level. This approach, enables easier comparison between observations. Since it can be expected, that in the sports domain some latent variables influence each performance, this way it can be checked, how different methods respond to them.

The *physical* variable influences the measurements *avg heart rate*, *number of punches*, *avg punch speed*. The *skill* variable influences the measurements *avg punch impact*, *punches defended* and *punches missed*. The measurement variables were taken from a normal distribution, where the latent variables modified the mean (center) of the distribution. If appropriate, the values drawn from these normal distributions were validated and checked to be in a certain value range, i.e. *punches defended* should not be negative or *punches missed* should not be higher than *number of punches*. While the aim of the usage of latent variables is to model a more realistic dataset, it can in no way capture the complexity of real life data - as such it is not intended to be a substitute for a 'real' dataset. Additionally to the discussed latent variables weight influences the *avg punch impact* measurement.

The complete list of generated values can be seen in the appendix in table 3. The performances with the ids 0 and 1 had their latent variables manually set to be the best or worst respectively.

Data Exploration Table 4.6 provides summarized statistics about the created dataset. As noted above, there are two measurements that are seen as *personal information* (height and weight), two latent variables (*skill* and *physical*) as well as six measurements that represent *performance indicators*. Figure 4.3 shows box plots of the generated performance data (and the two latent variables).

Figure 4.4 shows the influence of the latent variables on the generated measurements. In four out of the six performance indicators, it can be clearly seen that the manually created performances with the id 0 and 1 (as best and worst performances respectively), are indeed performing as best and worst respectively. As indicated by the corresponding values, where skill or physical are either 0 or 1, being clearly separated from the other values.

Additionally, table 4.7 shows the correlations between the different variables. Due to the latent variables many of the performance indicators are correlated to each other. The methods Principal Factor Analysis and Factor Analysis may be affected, since they try to extract information based on underlying properties within the data. Other methods, such as pairwise-distance, do not rely on it at all.

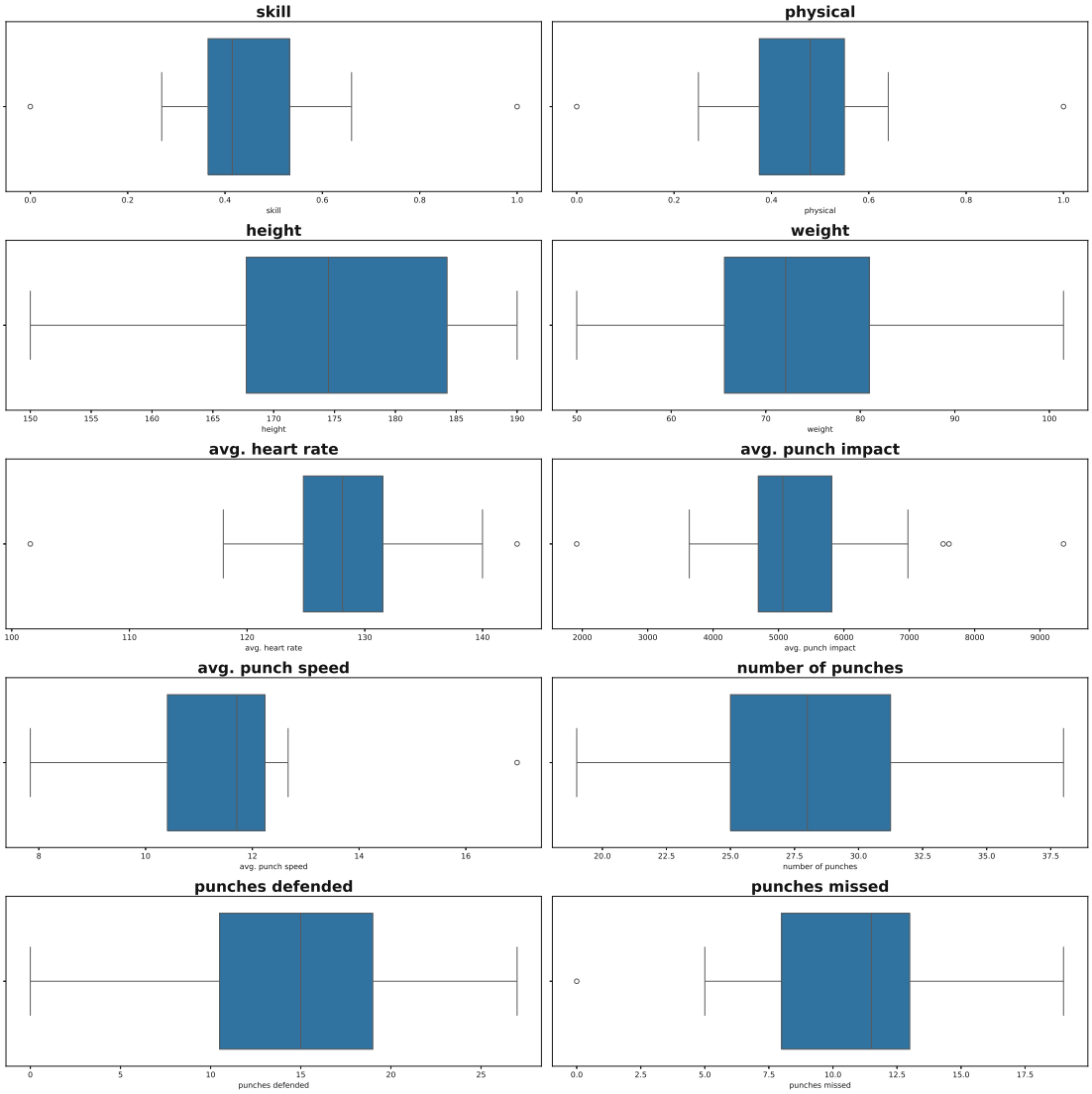


Figure 4.3: Generated variables of the boxing dataset.

Table 4.6: Summary statistics for input data of the boxing exercise.

	count	mean	std	min	max
height	20	174.25	10.43	150	190
weight	20	73.95	13.19	50	101.5
skill	20	0.45	0.2	0	1
physical	20	0.48	0.19	0	1
avg. heart rate	20	127.43	8.55	101.58	142.94
avg. punch impact	20	5348.82	1596.77	1914.69	9358.76
avg. punch speed	20	11.43	1.88	7.84	16.96
number of punches	20	28.1	4.83	19	38
punches defended	20	14.8	7.18	0	27
punches missed	20	10.5	4.11	0	19

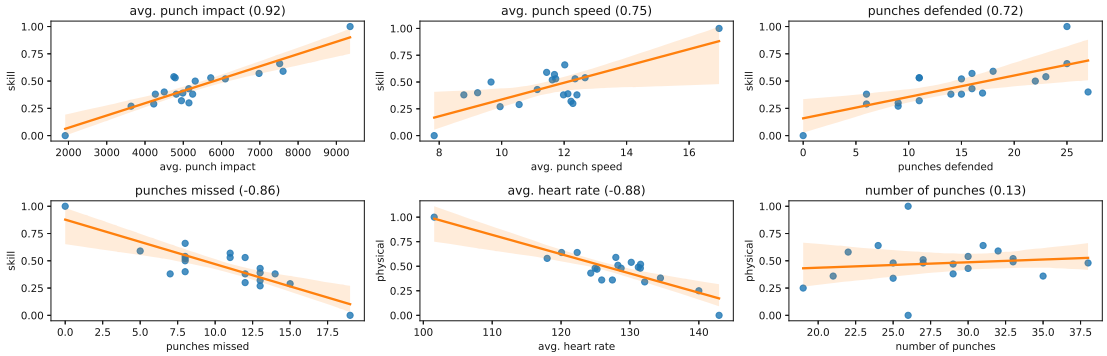


Figure 4.4: Plots showing the correlation between the latent variables (y-axis) and the affected performance indicators. The exact value of correlation is given in the heading.

Table 4.7: Complete table of correlations for the test dataset.

	height	weight	skill	physical	avg. heart rate	avg. punch impact	avg. punch speed	number of punches	punches de-fended	punches missed
height	1	0.65	0.5	0.58	-0.59	0.7	0.6	0.23	0.29	-0.41
weight	0.65	1	0.04	0.08	-0.07	0.4	0.11	0.33	-0	0.03
skill	0.5	0.04	1	0.77	-0.69	0.92	0.75	0.03	0.72	-0.86
physical	0.58	0.08	0.77	1	-0.88	0.73	0.7	0.13	0.56	-0.8
avg. heart rate	-0.59	-0.07	-0.69	-0.88	1	-0.63	-0.68	0.06	-0.38	0.64
avg. punch impact	0.7	0.4	0.92	0.73	-0.63	1	0.7	0.19	0.64	-0.77
avg. punch speed	0.6	0.11	0.75	0.7	-0.68	0.7	1	0.07	0.33	-0.62
number of punches	0.23	0.33	0.03	0.13	0.06	0.19	0.07	1	0.12	-0.09
punches defended	0.29	-0	0.72	0.56	-0.38	0.64	0.33	0.12	1	-0.69
punches missed	-0.41	0.03	-0.86	-0.8	0.64	-0.77	-0.62	-0.09	-0.69	1

Generated scores for the complete dataset

Initially, all the performances were ranked without separation between categories. As stated earlier, two performances were created manually to be at the top and bottom

respectively. The performance with id 0 is expected to be ranked among the best ones, as both its latent variables for skill and physical were the highest. On the other hand, the performance with id 1 has been created to be ranked at the bottom, as it has the worst latent variables for *skill* and *physical*. As expected the performance with id 0 did very well. It performed best in *avg. heart rate*, *avg. punch impact*, *avg. punch speed* and *punches missed*. Only in two measurements did it not perform as well: *number of punches* (seventh place) and *punches defended* (second place). Similarly, the performance with id 1 did not do well in many measurements. It performed worst in *avg. heart rate*, *avg. punch impact*, *avg. punch speed*, *punches defended* and *punches missed*. Only in *number of punches* it did not place last, being in seventh place[1]. The average placement of the two specially crafted performances in the measurement *number of punches*, can be explained by the low correlation of number of punches to the latent variables.

Table 4.8: Generated ranking per method for each performance.

id	RPC	LC	FA	PCA	SOM	min-max	pairwise-distance
0	20	1	20	20	8	1	1
1	19	20	1	1	4	20	20
2	6	12	8	9	19	12	10
3	7	13	10	11	19	13	11
4	14	2	19	18	15	2	2
5	17	16	4	5	17	16	17
6	2	4	15	14	16	4	4
7	4	7	11	12	5	7	5
8	8	14	7	6	19	14	13
9	1	9	6	10	5	9	7
10	9	8	16	15	12	8	8
11	18	17	3	3	3	17	19
12	10	11	14	13	13	11	14
13	15	15	12	8	10	15	15
14	11	5	13	16	1	5	9
15	3	3	18	19	14	3	2
16	13	6	17	17	9	6	6
17	5	19	5	4	2	19	16
18	16	10	9	7	11	10	12
19	12	18	2	2	7	18	18

A look at the ranks per method (table 4.8) shows that 3 of 7 methods (LC, min-max, pairwise-distance) correctly identified the performance with id 0 as the best performance. The same three methods also correctly ranked the performance with id 1 last. It is of note, that the methods FA and PCA reverse this, ranking the performance with id 0 as worst and the performance with id 1 as best. The method RPC ranks both these performances on the last two positions. Lastly, SOM puts both of the performances in the middle of the ranking.

[1]The *number of punches* measurement includes duplicated values, thus the same rank is assigned to multiple performances.

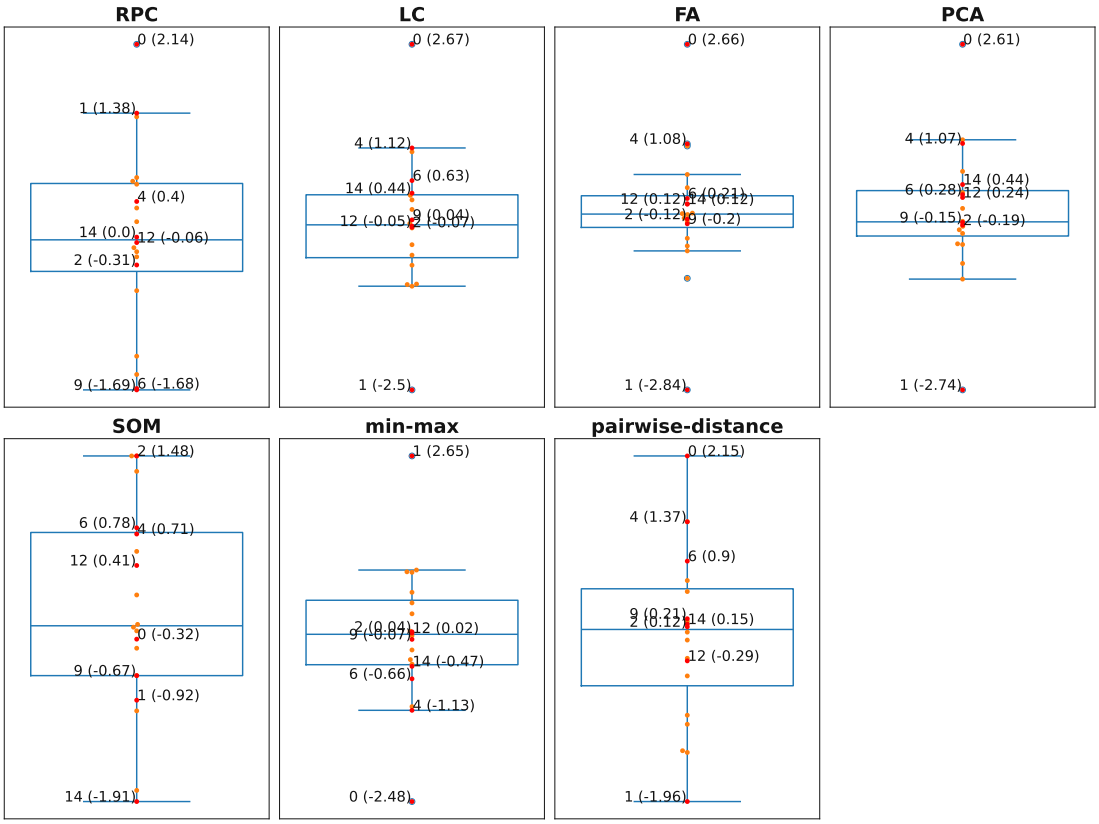


Figure 4.5: Box plots of generated scores. Additionally, every fourth (by latent skills) performance has been marked and its score written in brackets. Additionally, the performances with ids 0 and 1 have been marked. Some methods inversed the ranking, where the highest score denotes the worst performance

Table 4.9 show the overall similarity between the different ranking methods (and the latent ranking). Interestingly, other than for the NQLI the overall correlation between the methods is lower, which might be due to the smaller dataset. However, min-max and LC again have a perfect correlation of 1. Aside of the two formerly mentioned occurrences, there does not seem to be any other strong correlation between the methods.

Under the assumption that a ranking should reveal which performance was generated by the most optimal latent variables, a more direct comparison to the sum of the two latent parameters should indicate a 'better' ranking method. Additionally to table 4.9, figure 4.6 shows a visualization of the (dis-)similariy of each ranking to the ranking based on the latent variables alone.

The highest (positive) correlation between the latent variables can be observed for LC (and min-max), next closest is the pairwise-distance method. However, FA and PCA also have a very high (negative) correlation. This indicates that both of these methods

Table 4.9: Kendall's Tau between different methods when ranking all the available performances. The sum of *physical* and *skill* has been included as *latent*.

	RPC	LC	FA	PCA	SOM	min-max	pairwise-distance	latent
RPC	1	0.17	-0.08	-0.14	-0.14	0.17	0.31	-0.04
LC	0.17	1	-0.75	-0.82	-0.1	1	0.84	0.67
FA	-0.08	-0.75	1	0.86	0.15	-0.75	-0.71	-0.71
PCA	-0.14	-0.82	0.86	1	0.1	-0.82	-0.78	-0.68
SOM	-0.14	-0.1	0.15	0.1	1	-0.1	-0.14	-0.14
min-max	0.17	1	-0.75	-0.82	-0.1	1	0.84	0.67
pairwise-distance	0.31	0.84	-0.71	-0.78	-0.14	0.84	1	0.61
latent	-0.04	0.67	-0.71	-0.68	-0.14	0.67	0.61	1

also identified the patterns in the dataset. The fact that both methods also ranked the performances with ids 0 and 1 on the extremes of the ranking (best/worst) could indicate that PCA and FA could also be used as suitable methods. Ignoring the negative correlation, they performed better for the given dataset than LC - although just by a slim margin.

4.2 Comparing performances in different subcategories

In order to see how the different methods behave in smaller subcategories, the dataset was split using a predefined schema (see section 3.2.4, the schema can be seen in the appendix in listing 4). The creation of subcategories resulted in four different subcategories, with varying sizes. For further reference the categories have each been assigned a symbol. These subcategories were:

1. ♣: height below 170cm
2. ♢: height below 170cm and weight 60-80kg
3. ♡: height 170-190cm and weight 60-80kg
4. ♠: height 170-190 and weight above 80kg

Figure 4.7, shows how each of these categories relate to each other.

Table 4.10 shows the ranks per method after splitting the performances into subcategories (denoted by their respective symbol).

As the rankings should be predictable, the ranking between two performances should be the same regardless of other performances being present. For example, if performance a is ranked above performance b in the total ranking, the same performances should be ranked in the same order (a over b) in a category with fewer members. Table 4.11 contains the Kendall's Tau between the ranking in each category and the ranking of the same subset in the total ranking (see table 4.8). E.g. for category ♣ it compares

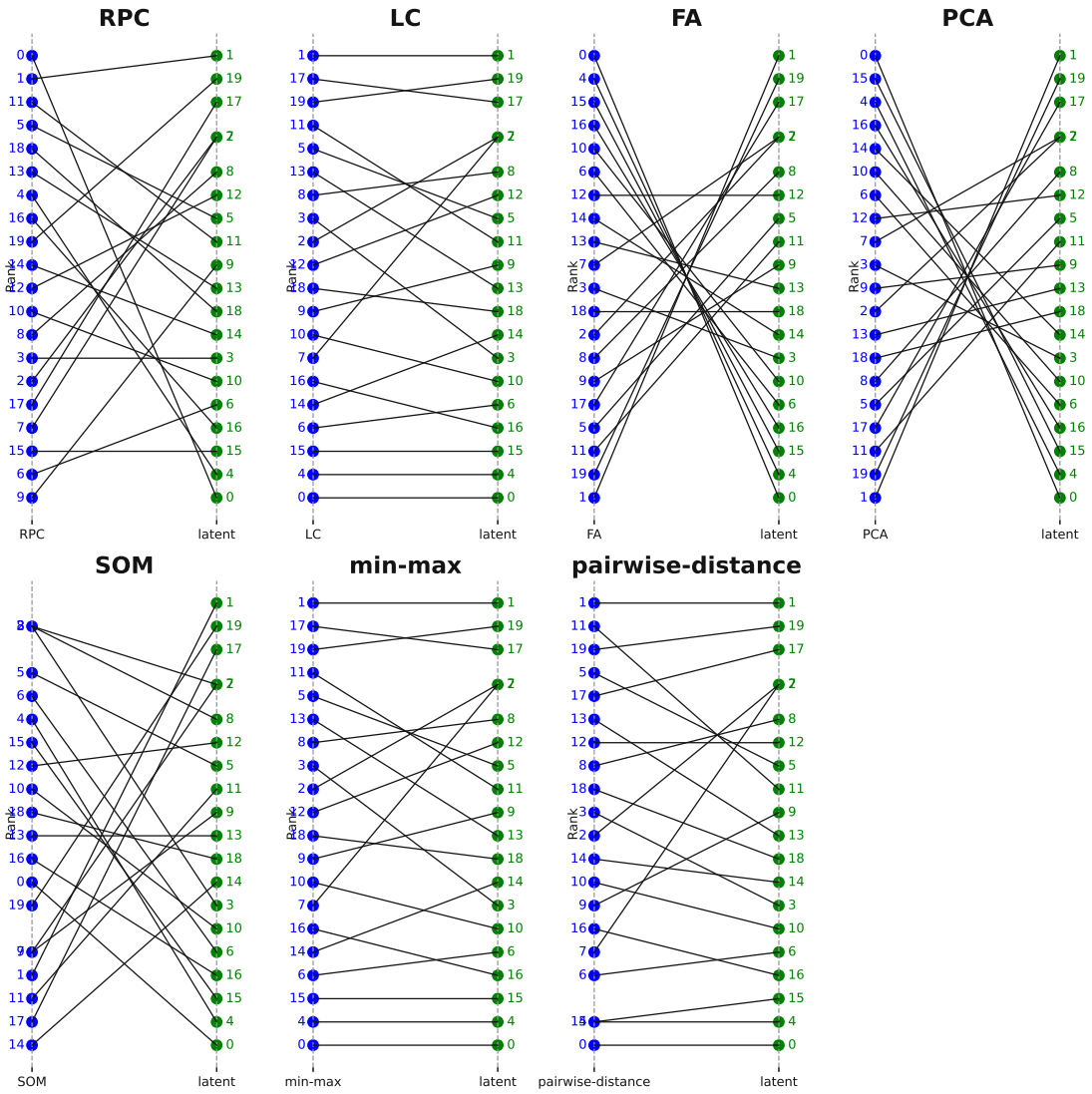


Figure 4.6: Comparison of created ranking compared to latent, where latent is the sum of both latent variables *skill* and *physical*. The ids are given next to the points and ordered according to the rankings. Additionally, the same ids on both sides are connected to showcase (dis-)similarity.

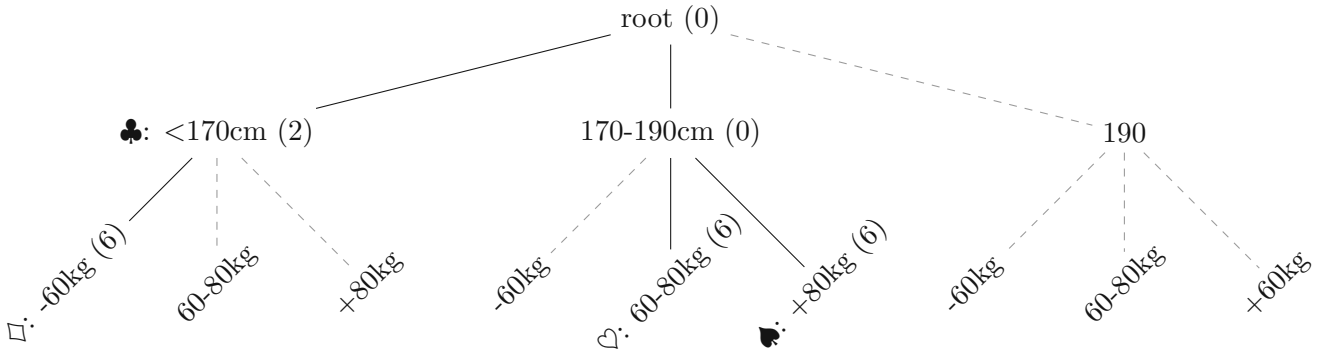


Figure 4.7: Category structure resulting after the boxing exercises were sorted. Not-existing categories are greyed out. For existing categories their member count is added in brackets. Additionally, the symbols used to refer to them are prefixed.

the ranking of performances with ids 13 and 16 as defined in table 4.10 (ranked in the category) and table 4.8 (ranking in the total category).

Category ♣ contains only two members. The rankings for this category highly correlate, since there are only two ways to rank two performances. It can be seen in table 4.11 that most of the methods kept their rankings the same (i.e. having a correlation of 1). Only FA and RPC reversed their ranking.

Category ♢ contains six members. Among these is the performance with id 1, which was created to be among the worst in the dataset. Aside from SOM, all of the methods rank the performance as the lowest of the six in the category. Overall, only LC and min-max show a complete replication of the total ranking. PCA ranked the items in the same order as in the total ranking, however it reversed the order. The performance that was ranked best in the total ranking, was ranked worst in the category and so forth - as can be seen by the perfect inverse correlation of -1. Of note is also that FA, which was otherwise performing similar to PCA, had a much worse correlation of only -0.6. RPC and SOM have a low correlation of 0.07 and -0.33 respectively.

Category ♡ contains six performances performance. Among these is the performance with id 0. As it was created using the highest possible latent factors, it is expected that the methods also rank it the highest in this category. Indeed LC, min-max and pairwise-distance assigned the highest rank. In line with their total ranking, FA and PCA assign it the lowest rank.

In the last category ♠, not a single method reached a perfect correlation. However, for most of them the correlation was very high. Notably, five of the methods have the exact same correlation of 0.87.

Overall, methods that either project the data into lower dimensionality (FA, PCA) or depend on random initialization (RPC, SOM), seem to prove less stable than other methods that depend on linear combinations or other mathematical calculations. Figure

Table 4.10: Scores per method for each subcategory of the boxing dataset. The categories are indicated via symbols. Each ranking was generated considering only performances within the category.

id	Category	RPC	LC	FA	PCA	SOM	min-max	pairwise-distance
13	♣	1	2	2	1	2	2	2
16	♣	2	1	1	2	1	1	1
1	♢	6	6	6	6	6	6	6
12	♢	4	3	2	2	3	3	3
14	♢	5	1	3	1	4	1	2
17	♢	1	5	1	4	2	5	4
18	♢	3	2	4	3	5	2	1
19	♢	2	4	5	5	1	4	5
11	♡	6	6	1	1	1	6	6
10	♡	3	3	5	5	6	3	3
7	♡	1	2	2	4	5	2	2
3	♡	4	4	4	3	3	4	4
0	♡	2	1	6	6	4	1	1
5	♡	5	5	3	2	2	5	5
9	♠	1	5	1	3	4	5	4
2	♠	5	4	2	2	2	4	5
4	♠	6	1	6	6	6	1	1
6	♠	2	3	4	4	1	3	3
8	♠	4	6	3	1	3	6	6
15	♠	3	2	5	5	5	2	2

Table 4.11: Kendall's Tau for each category and method. Compared were the resulting rankings of members within a specific category to their ranking when the dataset was not split into categories.

	♣	♢	♡	♠
RPC	-1	0.47	0.33	0.87
LC	1	1	1	0.87
FA	-1	-0.6	0.73	0.87
PCA	1	-1	1	0.87
SOM	1	-0.07	0.07	-0.28
min-max	1	1	1	0.87
pairwise-distance	1	0.87	1	0.97

4.8 visualizes the divergences. On the first vertical line in each subchart is the order of the total ranking, the second one the one resulting from categories and lastly the third line the one resulting from latent variables. The more the line connecting ids cross, the less stable - therefore worse - the ranking method performs for this particular dataset.

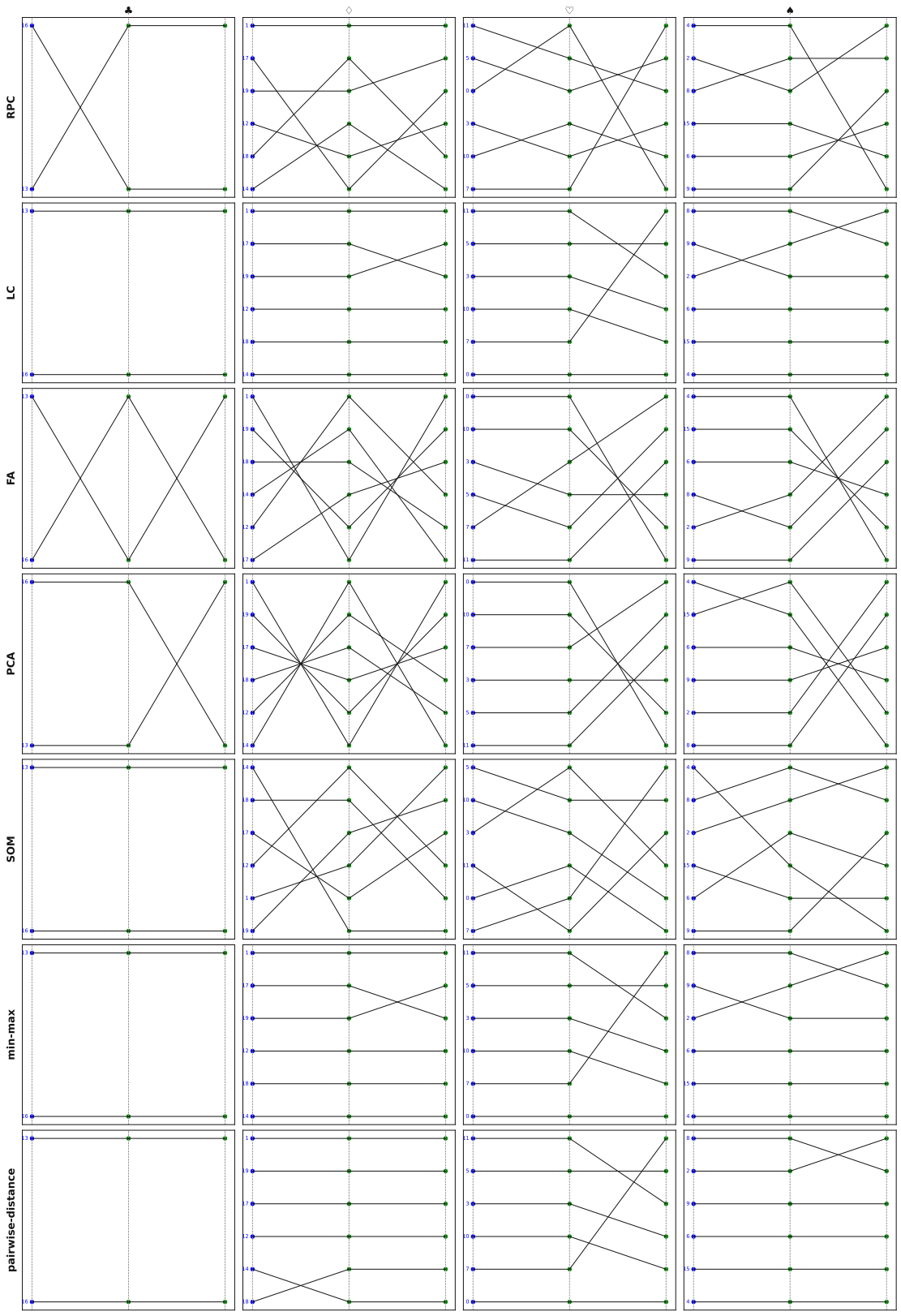


Figure 4.8: Comparing the rankings of categories and their members to the full ranking. Along the first line the ids are arranged by the ranks generated inside categories, on the second line the rankings as created when the full dataset is available, the last row the rank when the sum of both latent variables is used. Rows contain the methods while columns the categories.

4.2.1 Comparison of different scoring methods

While no definitive answer can be given on which of the tested methods is the most appropriate for any given dataset, some generalizations can be made based on the achieved results that should hold true for other datasets as well.

The only two methods used without any additional input of expert knowledge are PCA and FA. While they produced similar rankings for large datasets (see tables 4.3, 4.9), with PCA having a better selection for the top countries (table 4.4), they differed widely for smaller datasets, where PCA outperformed FA (tables 4.10, 4.11). However, even if arranged correctly, neither PCA nor FA were stable in the direction of their ranking (e.g. the total ranking was $A < B < C$, however a different ranking was then $A > C$). This indicates that while both methods can pickup the overall structure in the data, they lack a concept of 'which is better'.

While given more domain knowledge than PCA and FA, the SOM-based method still did not contain any information regarding the measurements themselves, but only the 'best' and 'worst' performances as initial starting weights. For all of the given tests, SOM turned out to be the worst performing approach. It did not provide a reliable ranking when splitting data into different subcategories. Additionally, it was not able to differentiate good from bad performances reliably (see tables 4.10, 4.4/4.5). More than just the resulting scores, also the mechanics used by this method are less suited for the given tasks, as neural networks such as SOMs need to be trained. While they may not need to be retrained for every single newly added performance, it is necessary to do so at a certain threshold. Moreover, retraining would - most likely - also require a redefinition of the 'best' and 'worst' performances, thus requiring time-consuming manual intervention.

Similarly to SOM, Ranking Principal Curve (RPC) utilises domain knowledge about the measurements directly (via the α-vector used in the initialization of $\mathbf{P}$). On the one hand, with RPC the ranking on the NQLI was reasonable. On the other hand, while correctly ranking the performance with id 1 the worst, it failed to rank the performance with id 0 as the best in the overall ranking for the boxing exercises (see table 4.8). Additionally, the stability of the ranking when breaking into categories was low, with one being exceptionally low (table 4.11).

4.2.2 Evaluation of different scoring methods based on predefined criteria

Each of the chosen methods has been analyzed using the criteria defined in section 2.3, table 2.1. In the following, each method explained and implanted in section 3.3 is evaluated. A summary and overview of the evaluation can be seen in table 4.12.

PCA is not deterministic, as it is possible for the ranking of performances to be inverted, even when applied to the same dataset. This lack of determinism means that the results may vary across runs. PCA does also not satisfy monotonicity and transitivity, again

because the order of the ranking can reverse. PCA does reflect superiority for larger data sets, however it struggles with smaller ones, such as the subcategories for the boxing exercise. PCA tends to perform well, since there are highly optimized algorithms present, however it still needs to be rerun for the complete dataset upon every new performance added. Finally, when considering robustness, PCA can generally be seen as robust.

FA is a non deterministic method, as the ranking of performances can be inverted when applied to the same dataset, leading to inconsistent results across different runs. Similar to PCA, it does therefore also not satisfy monotonocitiy or transitivity. In terms of reflection of superiority, FA tends to perform well on larger datasets, but can be problematic for smaller datasets, where the reversal of rankings is more common. Again similar to PCA, FA has to be rerun on every new performance added, making it less efficient. Finally, robustness is generally good.

min-max is a deterministic method, meaning that it consistently produces the same ranking results when applied to the same dataset. Min-max satisfies monotonicity and transitivity[2], as it assigns a fixed score for all given performances based on the distance to a calculated optimum. In terms of reflection of superiority, min-max generally performs well, and incorporating weights can further enhance its ability to reflect the relative importance of performances. However, its efficiency is relatively low, as the distances for each new performance must be re-evaluated against all other performances in the dataset, though some optimizations could be implemented to improve this process. In terms of robustness, min-max has no special requirements towards the data.

pairwise-distance is a deterministic method, meaning that it consistently produces the same results when applied to the same dataset. Similar to min-max, pairwise distance produces a fixed score based on the distance between pairs, which means that monotonicity and transitivity are given (but not strict). Regarding reflection of superiority, its performance depends on the context, as all performance indicators are treated equally, which may not always accurately reflect the relative importance of each indicator. This could be modified by assigning a weight to the performance indicators. However, still only absolute better/worse comparison is used, which might loose some nuance. In terms of efficiency, the method is very inefficient, as it requires an exponential number of comparisons, because each performance must be compared to every other performance in the dataset, and the entire process must be repeated when a new performance is added. However, robustness is high.

LC is a deterministic method, consistently yielding the same results when applied to the same dataset. Since LC assigns scores based on a sum of performance indicators monotonicity and transitivity [2] are given. In terms of reflection of superiority, LC

[2]In the previous results, there have been changes in the order when the dataset changed (mostly for smaller subsets). However, as noted, this is due to the application of a transformation *after* applying the scoring methods. This behavior was absent when not applying the transformation.

generally performs well, and its ability to reflect the relative importance of performance indicators, can be improved by using appropriate weights. Efficiency is very high with LC, as only a single calculation is required when a new performance is added, making it computationally efficient. Additionally, robustness is high, and LC consistently performs reliably across different scenarios.

RPC is a nondeterministic method due to the randomization in the starting weights. In regards to reflection of superiority, RPC performed similar to other methods for large data sets, however, it performed poorly for smaller ones. Monotonicity and transitivity are not given for RPC, likely due to the random elements present during the initialization. In terms of efficiency, the initial investment required to fit the comparison curve is high, making it computationally intensive at first. However, once set up, the method is effective. Robustness is high, as with RPC there are no special requirements for the data set.

SOM is a nondeterministic method, as its results can be highly random, leading to different outcomes even with the same dataset. In terms of monotonicity or transitivity, SOM does not satisfy these properties, as the order of rankings is quite random and unreliable. Regarding reflection of superiority, SOM performs poorly, regardless of the dataset size, failing to accurately reflect the relative importance of different performance indicators. Efficiency is low due to the very high initial computational cost of training the model, though the trained weights can be reused up to a point, after which re-training becomes necessary. Overall, the efficiency is considered bad. However, in terms of robustness, SOM is reliable, as it does not have specific requirements regarding the form or structure of the input performances.

4.3 Evaluation of comparison between subcategories

As stated previously, the same data and schema as in section 4.1.2 will be used to test the comparison of performances across different subcategories. For this, the best/worst and average performances were selected from each category (see table 4.13 for the ids). The method of comparison is outlined in section 3.4.

As the category ♣ only has two members, namely the performances with ids 13 and 16, it does not meet the minimum size of five as defined in the schema. Since the minimum size indicates the amount of performances needed to draw a meaningful comparison, the category can not processed as is. As shown in figure 4.7, the members of category ♣ are those that are under a height of 170cm. While this particular category has only two members, there are actually six more performances that fit the description of *under a height of 170cm*. The category ♢ contains members which are under 170cm of height *and* weigh less than 60kg. The tree-like structure created a new category to allow members of ♢ to be compared on a more specific level. However, members of the subcategory ♢ still belong to the super-category ♣ and can therefore be used to provide context to the two members of category ♣ (see section 3.4.1).

Table 4.13 shows the results of comparing the selected records. The first two rows *mean* and *median* contain the distance to the average performance. These two different metrics were chosen to see if outliers (such as the performance with id 0) affect the overall ranking, since median is less sensitive to outliers. The values in the two columns show the distance to the respective average, as well as the resulting rank in brackets. The next two columns "LC (total)" and "LC (Category)" contain the rank the performance received in the total ranking and in its specific subcategory respectively. Lastly, the column "Category" indicates the subcategory the performance has been assigned to. Similarly, figure 4.9 shows a graphical representation of the new ranking. The right column contains the respective peer group from which the average was calculated. The orange colored bar highlights the current id of interest for which the comparison score (in the left column) is calculated. A clear pattern can be seen, showing that the scores start high(er) then go towards the middle and then get worse further down the ranking. The figure also shows the extended group for performances with id 13 and 16. These two performances are the sole members of group ♣. As two members do not meet the minimum size requirement, their peer group has been extended using the closest subcategory (♢).

As can be seen for both ranking methods (mean and median) they conform to the category ranking in the sense that performances ranked top of their category are also ranked highest here - regardless of their total rank. Secondly, mid and low ranking performances are also sorted appropriately. In the comparison ranking, the performance with id 6 is ranked lower than the performance with id 12, despite the fact that the performance with id 6 performed much better in the total ranking (rank 4 vs 11). However, the performance with id 12 performed better *when compared to its peers*. Due to this it can be assumed, that the performance with id 12 performed better in light of its physical baseline than the performance with id 6 did.

The comparison-based ranking results in a gauge on how different performances were compared to their peers. As such they might not accurately reflect the real efficiency in reaching a predefined goal but can give hints as to whether a person is performing well for their given physical precondition. The approach of comparing to the average performance of the category gives an adaptive metric that can react reliably to new performances being added, while the grouping and extension mechanism ensures that each performance is evaluated in light of an appropriate peer category. Furthermore, using a single score based on the single score calculated in section 3.3 gives a good measurement on how much better or worse a specific performance is. This also enables comparison between the different subcategories, as the numerical values quantifying the how much can be compared to each other. This second comparison - while losing some insights - can be used as a indicator on which of any given person performs better while factoring out different physical preconditions.

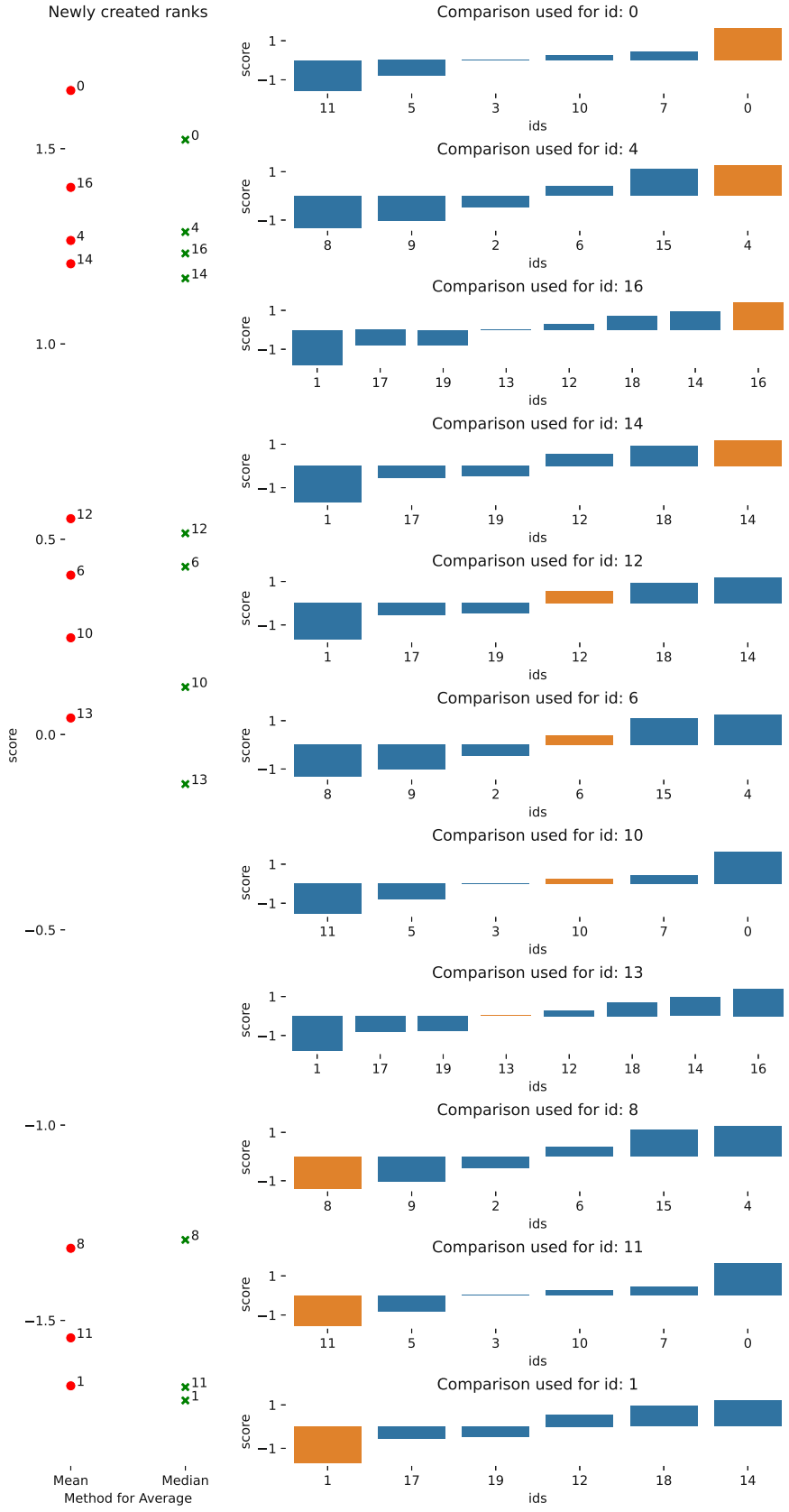


Figure 4.9: Comparison of performances in different subcategories. The left columns contains the newly created rankings for both definitions of average. The right column contains the scores of the performances in the respective subcategories, that were used to calculate the average.

Method	Deter-ministic	Mono-tonocity	Trans-itivity	Reflection of Superi-ority	Efficiency	Ro-bustness
PCA	✗	✗	✗	for large datasets only	needs to be rerun on complete dataset for each new performance	✓
FA	✗	✗	✗	for large datasets only	needs to be rerun on complete dataset for each new performance	✓
min-max*	✓	✓	✓	very good, could be adapted with weights	needs to be rerun on complete dataset for each new performance, but optimization would be possible	✓
pairwise-distance*	✓	✓	✓	only absolute better/worse comparison, no relativity, which might lose some nuance	very low, complete pairwise rerun needed with n^2 comparisons for n elements	✓
LC*	✓	✓	✓	very good, could be adapted with weights	very good, for each performance the calculation needs to be run only once	✓
RPC*	✗	✗	✗	mediocre for large datasets, bad for small ones	high initial effort, weights can then be reused until too many new performances have been added	✓
SOM*	✗	✗	✗	the worst of all tested methods	highest initial effort, weights can then be reused until too many new performances have been added	✓

Table 4.12: Overview of evaluation criteria per method, as defined in table 2.1. A ✓ indicates that a property is given. On the other hand, ✗ indicates that a certain property is not fulfilled. Methods marked with a * utilize domain knowledge in one way or another.

Table 4.13: Comparison of performances in different subcategories, with the assigned rank (for the given selection) in brackets.

id	mean	median	LC (total)	LC (Category)	Category
0	1.649 (1)	1.523 (1)	1	1	♡
4	1.265 (3)	1.287 (2)	2	1	♠
16	1.401 (2)	1.232 (3)	6	1	♣
14	1.205 (4)	1.168 (4)	5	1	♢
12	0.553 (5)	0.515 (5)	11	3	♢
6	0.408 (6)	0.43 (6)	4	3	♠
10	0.248 (7)	0.122 (7)	8	3	♡
13	0.042 (8)	-0.127 (8)	15	2	♣
8	-1.315 (9)	-1.294 (9)	14	6	♠
11	-1.545 (10)	-1.671 (10)	17	6	♡
1	-1.668 (11)	-1.705 (11)	20	6	♢

CHAPTER 5

Discussion

In this thesis a prototype has been created with the functionality that was outlined in section 1.5. The prototype was realized in python with the main intention being that it can be imported and used as a library.

The main functions provided by the prototype are as follows:

- Validating a given schema that represents the domain knowledge used in various different functions and features.
- Importing records (which conform to the given schema) and representing them in a class coupling them to the respective schema.
- Based on the given schema, the imported records can be sorted automatically into subcategories that get created according to the logic detailed in section 3.2, resulting in a tree-like structure of more and more granular categories. These categories can be updated as new records are being added, making dynamic sorting possible. Realizing feature 2 from table 1.1.
- For a given category a score based ranking can be calculated, which can then be used to rank the performances in accordance to their effectiveness. In other words the performances can be ranked from good to bad (see section 3.3). Realizing feature 1 from table 1.1.
- When performances are sorted into categories, performances from different categories may be picked and compared with regards to their physical preconditions. This second ranking is based on the previously mentioned scores and compares how good a performance is in regards to similar performances (see section 3.4). Realizing feature 3 from table 1.1.

Secondly, different performance scoring methods were implemented, tested with two different datasets and subsequently evaluated against criteria defined by an expert (see table 4.12 for the summary of criteria).

5.1 Interpretation of the rankings provided by different scoring methods

It has been shown that multiple mathematical methods can be used to create scores for a set of given performances. While for a big dataset all of the tested methods - apart from SOM - performed reasonably well, with a smaller dataset methods based on linear combinations performed the most stable and selected the most sensible ranks. Especially the method LC (defined in equation 3.3) is the most appropriate choice for the given datasets (see table 4.12), answering **RQ1**. Additionally, for the analyses datasets, the methods PCA and FA did not provide a reliable order of ranks, e.g. they ranked $A < B < C$ in one dataset and then $A > B > C$ in a subcategory. This makes them unsuitable for an automated ranking system that does not need human intervention to determine the order of rankings.

For large datasets (i.e. the NQLI with 171 performances) the incorporation of domain knowledge did not lead to improvement. For the NQLI data the correlation between methods that rely on domain knowledge and ones that do not was quite high (LC and PCA had a correlation of 0.97). The non-domain-knowledge-based methods also had a sensible ranking for the best/wort performing countries. However, similar to **RQ1**, when datasets had fewer members the non-domain-knowledge-based methods performed worse than the methods that do rely on domain knowledge. Since the knowledge based scoring methods are more appropriate for smaller categories, **RQ1.1** can be answered, stating that incorporating domain knowledge is indeed the more appropriate approach to the analyses problem.

In summary, for large datasets, the choice of method has relatively little impact on the results. However, differences between methods become more pronounced when working with smaller datasets or when considering the stability of rankings. Reproducibility, in particular, has often been overlooked in much of the literature. Methods that incorporate domain knowledge offer a key advantage in reproducibility, as they provide a fixed ranking order. In contrast methods that rely solely on statistical techniques (such as PCA or FA) may maintain similar rankings but can sometimes reverse the order, which creates issues when the dataset is updated. Specifically, these statistical methods require re-running of the entire dateset whenever new performance data is added.

On smaller datasets, methods that can evaluate a performance independently of the rest of the dataset tend to perform better than those that require the entire dataset for calculation. This brings two significant benefits. First, these methods are more efficient, as they do not need to be re-run on the entire dataset when new data is added. Second,

the scores generated are independent of the normative profile, it is not affecting how hey are calculated, only how they are interpreted.

5.2 Interpretation of the comparison of records across subcategories

It has been shown that, based on the methods found by **RQ1**, a comparability score across subcategories can be calculated by using the distance to an average performance of the respective peer category. It has been found that, based on the grouping logic outlined in section 3.2, the peer category defining an average performance needs to be large enough to calculate a sensible distance for each single performance. Once the categories are large enough the definition of "average" (mean and median were compared) did not result in a difference in the end ranking of the chosen performances. **RQ2** can thus be answered by the implementation of this two step approach to comparing records in regards to their physical baseline.

5.3 Limitations

One major limitation while evaluating was the lack of expert knowledge to create *multiple* ground truths that could then be compared - or similarly having a domain expert evaluate the resulting rankings for each scoring method. As such the evaluation has been limited to a sensibility check by a non-expert and to evaluating how similar the rankings were. However, some methods were able to be rejected in certain circumstances (such as SOM generally, or PCA and FA for smaller datasets).

Similarly, no real world test could be done - dealing with problems such as data drift or bias in different measurements (e.g. different equipment or person recording the measurements). Similarly, while the results are assumed to be generalize, only two datasets have been used to evaluate the methods presented.

As mentioned above, the evaluation was limited by the absence of real-world datasets and a lack of expert knowledge. In particular, a more in-depth analysis of the comparison scores between subcategories would have been beneficial and could have been more thoroughly explored with input from an expert for the respective real-world dataset.

5.4 Outlook

As with many software components, only real world usage will reveal the shortcomings and strengths of this prototype. While the current state should provide a reasonable feature set and starting point depending on the specific context in which it will be used, different necessities may arise. A few such opportunities for improvement are explored below.

5.4.1 Additional categorization approaches

In this iteration of the prototype the breakpoints for splitting subcategories are quite rigid, as they are hard coded values that stay the same regardless of their members. This could be done in a more flexible way, for example by allowing users to specify relative breakpoints. Instead of specifying that the subcategories are to be $-50kg$ and $+50kg$ (here the breakpoint is 50), a relative breakpoint could be set that splits the subcategory into thirds (lightest, middle and heaviest). The advantage here would be that the subcategories adapt to the distribution of the current members´ attributes - for example a male only category can be expected to have different weight values than a female only.

5.4.2 Further customizing LC

The scope of this thesis was focused on selecting one kind of scoring method. As such, a linear combination was selected (see equation 3.3). While supported by the prototype, no evaluation work was done on how different weights (w_j in 5.1) could affect the ranking. As such, the equation could easily be adapted to incorporate weights defined by experts. Similarly, these weights could be preselected by some kind of algorithm to provide a baseline, e.g. utilising PCA.

$$
\begin{aligned}
s_i &= \sum_{j=0}^{m} (w_j * d_j * x_{i,j}) \\
d_j &= \begin{cases} 1, & \text{if } desired_j \text{ is } high \\ -1, & \text{if } desired_j \text{ is } low \end{cases}
\end{aligned}
\tag{5.1}
$$

Another possible point of adjustment could be data reprocessing. As some of the methods expect a normalization into the value range of $[0, 1]$, the same scaling has been chosen for all methods. Here, additional steps can be explored when focusing on fewer methods.

5.4.3 Insight into subcategories and meta-groups

Further research could benefit from a meta-analytical approach that compares top-performances in different subcategories, allowing for a refined understanding of key performance indicators across different normative profiles. For instance an analysis could reveal that in lower weight classes speed is a more influential factor, whereas, in higher weight classes, strength may be distinguishing feature. This comparative approach would enable dynamic insight into the different influence that performance indicators have in different subcategories. Such insights could inform tailored strategies to optimize performance in specific contexts, guiding more precise interventions and enhancing the broader understanding of performance dynamics across diverse categories.

5.4.4 Visualization and export of resulting categories

In its current form there is no way to easily visualize the resulting subcategories and their members. For purposes such as presentations, an easy way to export the resulting tree structure could prove useful. Additionally, exporting the data and scores into common formats such as CSV could also be of use, when the data is to be processed outside of the python programming language.

Similarly, an import/export function - within the python environment - could be of use in order for the current data structure to be be preserved across invocations of the program.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] Optimization and root finding (scipy.optimize). https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.brentq.html#scipy.optimize.brentq, . Accessed: 2024-06-30.

[2] scipy.stats.goodness_of_fit — scipy v1.13.1 manual. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.goodness_of_fit.html, . Accessed: 2024-06-30.

[3] David J Bartholomew. Factor analysis and latent variable modelling. In *International Encyclopedia of Statistical Science*, pages 501–503. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[4] Frank M. Brasile. Performance evaluation of wheelchair athletes: More than a disability classification level issue. *Adapted Physical Activity Quarterly*, 7(4):289–297, oct 1990. doi: 10.1123/apaq.7.4.289.

[5] Richard P Brent. *Algorithms for minimization without derivatives*. Prentice Hall, 1973.

[6] Heike Brock, Yuji Ohgi, and James Lee. Learning to judge like a human: convolutional networks for classification of ski jumping errors. In *Proceedings of the 2017 ACM International Symposium on Wearable Computers*, UbiComp '17. ACM, September 2017. doi: 10.1145/3123021.3123038. URL http://dx.doi.org/10.1145/3123021.3123038.

[7] D. Chuckravanen, S. Bulut, G.B. Kürklü, and G. Yapali. Review of exercise-induced physiological control models to explain the development of fatigue to improve sports performance and future trend. *Science & Sports*, 34(3):131–140, June 2019. ISSN 0765-1597. doi: 10.1016/j.scispo.2018.10.017. URL http://dx.doi.org/10.1016/j.scispo.2018.10.017.

[8] Hayden Croft, Bobby Willcox, and Peter Lamb. Using performance data to identify styles of play in netball: an alternative to performance indicators. *International Journal of Performance Analysis in Sport*, 17(6):1034–1043, November 2017. ISSN 1474-8185. doi: 10.1080/24748668.2017.1419408. URL http://dx.doi.org/10.1080/24748668.2017.1419408.

[9] Christopher Cushion and John Lyle. *Conceptualising Sport-Coaching: Some Key Questions and Issues*, pages 117–134. Springer Fachmedien Wiesbaden, Wiesbaden, 2016. ISBN 978-3-658-10171-8. doi: 10.1007/978-3-658-10171-8_7. URL https://doi.org/10.1007/978-3-658-10171-8_7.

[10] Lysann Damisch and Thomas Mussweiler. On the relativity of athletic performance: a comparison perspective on performance judgments in sports. In Markus Raab, Joseph G. Johnson, and Hauke R. Heekeren, editors, *Mind and Motion: The Bidirectional Link between Thought and Action*, volume 174 of *Progress in Brain Research*, pages 13–24. Elsevier, 2009. doi: https://doi.org/10.1016/S0079-6123(09)01302-8. URL https://www.sciencedirect.com/science/article/pii/S0079612309013028.

[11] Giulia Donà, Ezio Preatoni, Claudio Cobelli, Renato Rodano, and Andrew J. Harrison. Application of functional principal component analysis in race walking: An emerging methodology. *Sports Biomechanics*, 8(4):284–301, nov 2009. doi: 10.1080/14763140903414425.

[12] Shengjie Dong, Ross Millar, Chenshu Shi, Minye Dong, Yuyin Xiao, Jie Shen, Xianqun Fan, and Guohong Li. Rating hospital performance in china: Review of publicly available measures and development of a ranking system. *Journal of Medical Internet Research*, 23(6):e17095, June 2021. ISSN 1438-8871. doi: 10.2196/17095. URL http://dx.doi.org/10.2196/17095.

[13] Richard Dwight. Searching for real maintenance performance measures. *Journal of Quality in Maintenance Engineering*, 5(3):258–275, sep 1999. doi: 10.1108/13552519910282728.

[14] Gerald Farin. *Curves and Surfaces for CAGD*. The Morgan Kaufmann Series in Computer Graphics. Morgan Kaufmann, 5 edition, October 2001.

[15] Miguel-Angel Gómez, Alberto Lorenzo, Sergio-José Ibañez, and Jaime Sampaio. Ball possession effectiveness in men's and women's elite basketball according to situational variables in different game periods. *Journal of Sports Sciences*, 31(14):1578–1587, October 2013. ISSN 1466-447X. doi: 10.1080/02640414.2013.792942. URL http://dx.doi.org/10.1080/02640414.2013.792942.

[16] Jarryd Heasman, Brian Dawson, Jason Berry, and Glenn Stewart. Development and validation of a player impact ranking system in australian football. *International Journal of Performance Analysis in Sport*, 8(3):156–171, November 2008. ISSN 1474-8185. doi: 10.1080/24748668.2008.11868457. URL http://dx.doi.org/10.1080/24748668.2008.11868457.

[17] Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS quarterly*, pages 75–105, 2004. doi: 10.2307/25148625.

[18] Mike Hughes, Steve Evans, and Julia Wells. Establishing normative profiles in performance analysis. *International Journal of Performance Analysis in Sport*, 1 (1):1–26, July 2001. ISSN 1474-8185. doi: 10.1080/24748668.2001.11868245. URL http://dx.doi.org/10.1080/24748668.2001.11868245.

[19] Mike D Hughes and Roger M Bartlett. The use of performance indicators in performance analysis. *Journal of sports sciences*, 20(10):739–754, 2002.

[20] Sergio J. Ibañez, Javier Garcia-Rubio, Miguel-Ángel Gómez, and Sergio Gonzalez-Espinosa. The impact of rule modifications on elite basketball teams' performance. *Journal of Human Kinetics*, 64(1):181–193, September 2018. ISSN 1899-7562. doi: 10.1515/hukin-2017-0193. URL http://dx.doi.org/10.1515/hukin-2017-0193.

[21] Lachlan P. James, Sam Robertson, G. Gregory Haff, Emma M. Beckman, and Vincent G. Kelly. Identifying the performance characteristics of a winning outcome in elite mixed martial arts competition. *Journal of Science and Medicine in Sport*, 20(3):296–301, March 2017. ISSN 1440-2440. doi: 10.1016/j.jsams.2016.08.001. URL http://dx.doi.org/10.1016/j.jsams.2016.08.001.

[22] Ian Jolliffe. Principal component analysis. In *International Encyclopedia of Statistical Science*, pages 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[23] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[24] Maciej Kostrzewa, Radosław Laskowski, Michal Wilk, Wies ław Błach, Angelina Ignatjeva, and Magdalena Nitychoruk. Significant predictors of sports performance in elite men judo athletes based on multidimensional regression models. *International Journal of Environmental Research and Public Health*, 17(21):8192, nov 2020. doi: 10.3390/ijerph17218192.

[25] Oliver Kramer. *Genetic Algorithm Essentials*. Springer International Publishing, 2017. ISBN 9783319521565. doi: 10.1007/978-3-319-52156-5. URL http://dx.doi.org/10.1007/978-3-319-52156-5.

[26] Sarka Krizkova, Hana Tomaskova, and Erfan Babaee Tirkolaee. Sport performance analysis with a focus on racket sports: A review. *Applied Sciences*, 11(19):9212, October 2021. ISSN 2076-3417. doi: 10.3390/app11199212. URL http://dx.doi.org/10.3390/app11199212.

[27] Bill Kuechler and Vijay Vaishnavi. On theory development in design science research: anatomy of a research project. *European Journal of Information Systems*, 17 (5):489–504, October 2008. ISSN 1476-9344. doi: 10.1057/ejis.2008.40. URL http://dx.doi.org/10.1057/ejis.2008.40.

[28] Takio Kurita. *Principal Component Analysis (PCA)*, page 1–4. Springer International Publishing, 2020. ISBN 9783030032432. doi: 10.1007/978-3-030-03243-2_649-1. URL http://dx.doi.org/10.1007/978-3-030-03243-2_649-1.

[29] Shitanshu Kusmakar, Sergiy Shelyag, Ye Zhu, Dan Dwyer, Paul Gastin, and Maia Angelova. Machine learning enabled team performance analysis in the dynamical environment of soccer. *IEEE Access*, 8:90266–90279, 2020. ISSN 2169-3536. doi: 10.1109/access.2020.2992025. URL http://dx.doi.org/10.1109/ACCESS.2020.2992025.

[30] Chun-Guo Li, Xing Mei, and Bao-Gang Hu. Unsupervised ranking of multi-attribute objects based on principal curves, 2014.

[31] Dario G. Liebermann, Larry Katz, Mike D. Hughes, Roger M. Bartlett, Jim McClements, and Ian M. Franks. Advances in the application of information technology to sport performance. *Journal of Sports Sciences*, 20(10):755–769, January 2002. ISSN 1466-447X. doi: 10.1080/026404102320675611. URL http://dx.doi.org/10.1080/026404102320675611.

[32] T. Linder, A. Krzyzak, B. Kégl, and K. Zeger. Learning and design of principal curves. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 22(03): 281–297, mar 2000. ISSN 1939-3539. doi: 10.1109/34.841759.

[33] Sigmund Loland. Classification in sport: A question of fairness. *European Journal of Sport Science*, 21(11):1477–1484, June 2021. ISSN 1536-7290. doi: 10.1080/17461391.2021.1923816. URL http://dx.doi.org/10.1080/17461391.2021.1923816.

[34] Felicity Lord, David B Pyne, Marijke Welvaert, and Jocelyn K Mara. Methods of performance analysis in team invasion sports: A systematic review. *Journal of Sports Sciences*, 38(20):2338–2349, June 2020. ISSN 1466-447X. doi: 10.1080/02640414.2020.1785185. URL http://dx.doi.org/10.1080/02640414.2020.1785185.

[35] W. Martins and J. Carlos Meira e Silva. Multidimensional data ranking using self-organising maps and genetic algorithms. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, IJCNN-01. IEEE. doi: 10.1109/ijcnn.2001.938739. URL http://dx.doi.org/10.1109/IJCNN.2001.938739.

[36] Dubravko Miljković. Brief review of self-organizing maps. In *2017 40th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 1061–1066. IEEE, 2017.

[37] Jo Morrison. An evolutionary perspective on sport and performance enhancement. *Performance Enhancement & Health*, 11(4):100263, October 2023. ISSN 2211-2669. doi: 10.1016/j.peh.2023.100263. URL http://dx.doi.org/10.1016/j.peh.2023.100263.

[38] Wolfram Müller. Determinants of ski-jump performance and implications for health, safety and fairness. *Sports medicine*, 39:85–106, 2009.

[39] P. O'Donoghue. Normative profiles of sports performance. *International Journal of Performance Analysis in Sport*, 5(1):104–119, June 2005. ISSN 1474-8185. doi: 10.1080/24748668.2005.11868319. URL http://dx.doi.org/10.1080/24748668.2005.11868319.

[40] Juan Ortega, Carlos Evangelio, Filipe Clemente, Fernando Martins, and Sixto González-Víllora. Analysis of physiological, technical, and tactical analysis during a friendly football match of elite u19. *Sports*, 4(2):35, June 2016. ISSN 2075-4663. doi: 10.3390/sports4020035. URL http://dx.doi.org/10.3390/sports4020035.

[41] Llukan Puka. *Kendall's Tau*, page 713–715. Springer Berlin Heidelberg, 2011. ISBN 9783642048982. doi: 10.1007/978-3-642-04898-2_324. URL http://dx.doi.org/10.1007/978-3-642-04898-2_324.

[42] N A Rahmad and M A As'ari. The new convolutional neural network (cnn) local feature extractor for automated badminton action recognition on vision based data. *Journal of Physics: Conference Series*, 1529(2):022021, April 2020. ISSN 1742-6596. doi: 10.1088/1742-6596/1529/2/022021. URL http://dx.doi.org/10.1088/1742-6596/1529/2/022021.

[43] Gaetano Raiola, Ylenia Giugno, Isabella Scassillo, and Pio Alfredo Di Tore. An experimental study on aerobic gymnastic: performance analysis as an effective evaluation for technique and teaching of motor gestures. *Journal of Human Sport and Exercise*, 8(Proc2):297–306, 2013. ISSN 1988-5202. doi: 10.4100/jhse.2012.8.proc2.32. URL http://dx.doi.org/10.4100/jhse.2012.8.Proc2.32.

[44] Benjamin P. Raysmith, Jenny Jacobsson, Michael K. Drew, and Toomas Timpka. What is performance? a scoping review of performance outcomes as study endpoints in athletics. *Sports*, 7(3):66, March 2019. ISSN 2075-4663. doi: 10.3390/sports7030066. URL http://dx.doi.org/10.3390/sports7030066.

[45] Jose M. Saavedra, Sveinn Þorgeirsson, Milan Chang, Hafrún Kristjánsdóttir, and Antonio García-Hermoso. Discriminatory power of women's handball game-related statistics at the olympic games (2004-2016). *Journal of Human Kinetics*, 62(1):221–229, June 2018. ISSN 1899-7562. doi: 10.1515/hukin-2017-0172. URL http://dx.doi.org/10.1515/hukin-2017-0172.

[46] Mohamad Nizam Mohamed Saphie, Diyanna Athirah Jamsari, Afrina Mohd Razi, Jamiaton Kusrin, and Wahidah Tumijan. Comparison of action performances and outcome between winners and losers in young female silat tempur matches. *Revista de Artes Marciales Asiáticas*, 14(2s):21–24, November 2019. ISSN 2174-0747. doi: 10.18002/rama.v14i2s.5964. URL http://dx.doi.org/10.18002/rama.v14i2s.5964.

[47] Soumitra Sharma. Principal component analysis (pca) to rank countries on their readiness for e-tail. *Journal of Retail & Leisure Property*, 7(2):87–94, March 2008. ISSN 1750-2098. doi: 10.1057/rlp.2008.1. URL http://dx.doi.org/10.1057/rlp.2008.1.

[48] Hannah Snyder. Literature review as a research methodology: An overview and guidelines. *Journal of Business Research*, 104:333–339, November 2019. ISSN 0148-2963. doi: 10.1016/j.jbusres.2019.07.039.

[49] Christopher Thomas, Thomas Dos'Santos, and Paul Jones. A comparison of dynamic strength index between team-sport athletes. *Sports*, 5(3):71, September 2017. ISSN 2075-4663. doi: 10.3390/sports5030071. URL http://dx.doi.org/10.3390/sports5030071.

[50] B. Travassos, J. Bourbousson, P.T. Esteves, R. Marcelino, M. Pacheco, and K. Davids. Adaptive behaviours of attacking futsal teams to opposition defensive formations. *Human Movement Science*, 47:98–105, June 2016. ISSN 0167-9457. doi: 10.1016/j.humov.2016.02.004. URL http://dx.doi.org/10.1016/j.humov.2016.02.004.

[51] Mark Waldron and Paul Worsfold. Differences in the game specific skills of elite and sub-elite youth football players: Implications for talent identification. *International journal of performance analysis in sport*, 10(1):9–24, 2010.

[52] Sebastian Wenninger, Daniel Link, and Martin Lames. Performance of machine learning models in application to beach volleyball data. *International Journal of Computer Science in Sport*, 19(1):24–36, June 2020. ISSN 1684-4769. doi: 10.2478/ijcss-2020-0002. URL http://dx.doi.org/10.2478/ijcss-2020-0002.

[53] Thomas Wilde and Thomas Hess. Methodenspektrum der wirtschaftsinformatik: Überblick und portfoliobildung, 01 2006.

[54] Carl T. Woods, James Veale, Job Fransen, Sam Robertson, and Neil French Collier. Classification of playing position in elite junior australian football using technical skill indicators. *Journal of Sports Sciences*, 36(1):97–103, January 2017. ISSN 1466-447X. doi: 10.1080/02640414.2017.1282621. URL http://dx.doi.org/10.1080/02640414.2017.1282621.

[55] M.R. Yeadon and J.H. Challis. The future of performance-related sports biomechanics research. *Journal of Sports Sciences*, 12(1):3–32, February 1994. ISSN 1466-447X. doi: 10.1080/02640419408732156. URL http://dx.doi.org/10.1080/02640419408732156.

[56] I-K Yeo. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, December 2000.

[57] ANDREI Zinovyev and Alexander N Gorban. Nonlinear quality of life index. *arXiv preprint arXiv:1008.4063*, 2010.

Appendix

Table 1: Complete scores for the NQLI data

country	RPC	LC	FA	PCA	SOM	min-max	pairwise-distance	NQLI	Rank
Luxembourg	-2.325	2.389	-1.594	-2.147	-1.183	-2.729	1.436	0.892	1
Norway	-2.002	1.856	-1.578	-1.781	1.843	-1.981	1.567	0.647	2
Kuwait	-1.716	1.55	-1.129	-1.395	-0.628	-1.6	1.19	0.608	3
Singapore	-1.777	1.645	-1.465	-1.6	0.323	-1.716	1.417	0.578	4
United States	-1.684	1.566	-1.238	-1.456	-1.469	-1.619	1.365	0.575	5
Ireland	-1.683	1.554	-1.409	-1.515	-0.876	-1.605	1.386	0.539	6
Iceland	-1.757	1.618	-1.662	-1.651	-0.876	-1.682	1.619	0.519	7
Switzerland	-1.766	1.591	-1.636	-1.617	-0.876	-1.649	1.526	0.517	8
Canada	-1.679	1.526	-1.498	-1.525	-0.328	-1.572	1.495	0.508	9
Netherlands	-1.607	1.494	-1.426	-1.481	0.553	-1.533	1.445	0.503	10
Austria	-1.584	1.466	-1.416	-1.458	0.553	-1.5	1.386	0.495	11
Denmark	-1.469	1.406	-1.272	-1.37	0.553	-1.431	1.33	0.486	12
Australia	-1.685	1.512	-1.607	-1.551	-1.223	-1.555	1.461	0.486	13
United Arab Emirates	-1.384	1.296	-1.083	-1.221	-0.461	-1.305	1.183	0.479	14
Sweden	-1.597	1.483	-1.549	-1.519	-0.056	-1.521	1.498	0.475	15
Belgium	-1.488	1.397	-1.358	-1.393	0.723	-1.421	1.312	0.471	16
United Kingdom	-1.486	1.378	-1.353	-1.376	0.723	-1.399	1.29	0.465	17
Japan	-1.637	1.473	-1.699	-1.558	-0.98	-1.509	1.405	0.46	18
Germany	-1.462	1.382	-1.385	-1.393	0.723	-1.403	1.371	0.454	19
Finland	-1.426	1.375	-1.354	-1.381	0.723	-1.395	1.374	0.453	20
France	-1.502	1.391	-1.486	-1.431	-0.98	-1.413	1.368	0.447	21
Italy	-1.49	1.386	-1.563	-1.455	-0.98	-1.408	1.421	0.428	22
Spain	-1.402	1.296	-1.438	-1.35	-0.98	-1.305	1.262	0.418	23
Bahrain	-1.044	1.013	-0.835	-0.948	0.421	-0.993	0.964	0.395	24
Greece	-1.233	1.207	-1.261	-1.234	-0.549	-1.205	1.218	0.393	25
New Zealand	-1.289	1.236	-1.376	-1.29	-1.678	-1.237	1.321	0.386	26
Cyprus	-1.255	1.236	-1.348	-1.284	-0.549	-1.237	1.343	0.384	27
Israel	-1.275	1.242	-1.412	-1.309	-1.678	-1.244	1.346	0.38	28
Slovenia	-1.068	1.12	-1.15	-1.144	-0.362	-1.109	1.202	0.36	29
Malta	-1.105	1.118	-1.296	-1.186	-0.005	-1.106	1.237	0.336	30
Korea, Rep.	-1.072	0.975	-1.161	-1.038	-1.506	-0.953	0.999	0.336	31
Puerto Rico	-1	1.121	-1.263	-1.192	0.189	-1.11	1.302	0.329	32
Portugal	-1.005	1.028	-1.17	-1.085	0.189	-1.009	1.064	0.324	33
Czech Rep.	-0.877	1.004	-0.994	-1.017	0.189	-0.983	1.171	0.321	34
Oman	-0.819	0.885	-0.79	-0.852	0.122	-0.857	1.005	0.309	35
Saudi Arabia	-0.677	0.638	-0.402	-0.534	1.005	-0.603	0.664	0.288	36
Barbados	-0.788	0.845	-0.921	-0.869	0.189	-0.815	1.074	0.263	37
Hungary	-0.537	0.75	-0.62	-0.72	-0.725	-0.717	0.857	0.259	38
Slovak Republic	-0.597	0.77	-0.723	-0.765	-1.08	-0.738	0.917	0.252	39
Estonia	-0.518	0.711	-0.602	-0.687	0.491	-0.677	0.787	0.251	40
Chile	-0.73	0.824	-1.071	-0.911	-1.026	-0.794	0.983	0.233	41
Poland	-0.579	0.749	-0.809	-0.782	-1.394	-0.716	0.892	0.231	42
Croatia	-0.59	0.733	-0.835	-0.779	-1.394	-0.7	0.888	0.227	43

Continued on next page

Table 1: Complete scores for the NQLI data

country	RPC	LC	FA	PCA	SOM	min-max	pairwise-distance	NQLI	Rank
Lithuania	-0.339	0.558	-0.45	-0.535	-0.99	-0.522	0.639	0.203	44
Costa Rica	-0.643	0.742	-1.08	-0.854	-0.279	-0.709	0.904	0.195	45
Cuba	-0.535	0.757	-1.089	-0.883	-1.715	-0.724	0.92	0.191	46
Latvia	-0.281	0.512	-0.396	-0.486	-0.983	-0.476	0.543	0.187	47
Argentina	-0.499	0.579	-0.666	-0.606	1.11	-0.543	0.708	0.184	48
Uruguay	-0.492	0.616	-0.781	-0.673	0.706	-0.58	0.762	0.18	49
Libya	-0.416	0.519	-0.51	-0.51	1.11	-0.483	0.683	0.17	50
Grenada	-0.464	0.572	-0.665	-0.598	0.706	-0.536	0.81	0.17	51
Malaysia	-0.407	0.48	-0.557	-0.512	-0.687	-0.444	0.607	0.169	52
Mexico	-0.651	0.484	-0.579	-0.481	0.15	-0.448	0.67	0.168	53
Mauritius	-0.272	0.494	-0.434	-0.48	-0.032	-0.458	0.616	0.16	54
Trinidad and Tobago	-0.273	0.328	-0.017	-0.192	0.194	-0.294	0.467	0.154	55
Bulgaria	-0.272	0.475	-0.472	-0.482	0.781	-0.439	0.581	0.153	56
Venezuela	-0.37	0.458	-0.479	-0.46	1.11	-0.422	0.575	0.152	57
Belize	-0.411	0.5	-0.697	-0.564	0.706	-0.464	0.607	0.147	58
Panama	-0.452	0.463	-0.628	-0.508	1.11	-0.427	0.569	0.145	59
Albania	-0.4	0.522	-0.774	-0.603	0.706	-0.486	0.623	0.142	60
Macedonia, FYR	-0.311	0.471	-0.559	-0.503	1.11	-0.435	0.578	0.142	61
Bosnia and Herzegovina	-0.331	0.471	-0.653	-0.536	1.11	-0.435	0.543	0.137	62
Lebanon	-0.306	0.364	-0.263	-0.312	1.375	-0.33	0.527	0.131	63
Tunisia	-0.278	0.398	-0.476	-0.419	1.11	-0.363	0.476	0.121	64
Syria	-0.174	0.428	-0.558	-0.482	1.11	-0.392	0.463	0.119	65
Brazil	-0.217	0.321	-0.302	-0.311	0.284	-0.287	0.371	0.113	66
Iran	-0.271	0.284	-0.147	-0.214	2.171	-0.252	0.367	0.112	67
Colombia	-0.169	0.345	-0.383	-0.361	0.284	-0.311	0.387	0.109	68
Belarus	0.034	0.288	-0.141	-0.254	-0.249	-0.255	0.3	0.105	69
Romania	-0.247	0.27	-0.324	-0.289	0.08	-0.238	0.396	0.104	70
Sri Lanka	-0.14	0.357	-0.535	-0.429	0.486	-0.323	0.3	0.103	71
Jamaica	-0.181	0.306	-0.245	-0.274	1.375	-0.273	0.416	0.103	72
Turkey	-0.205	0.289	-0.242	-0.261	0.607	-0.256	0.335	0.101	73
Tonga	-0.089	0.295	-0.293	-0.296	0.284	-0.262	0.297	0.094	74
Jordan	-0.11	0.309	-0.326	-0.313	-0.259	-0.275	0.422	0.094	75
Ecuador	-0.361	0.259	-0.489	-0.323	-1.952	-0.227	0.358	0.09	76
Samoa	-0.055	0.238	-0.206	-0.224	1.141	-0.206	0.232	0.079	77
Armenia	-0.169	0.23	-0.373	-0.272	-0.747	-0.198	0.2	0.074	78
Russia	0.213	0.138	0.16	-0.046	1.076	-0.11	0.213	0.073	79
El Salvador	-0.057	0.206	-0.191	-0.199	0.381	-0.176	0.177	0.072	80
China	-0.154	0.202	-0.326	-0.242	-0.68	-0.172	0.229	0.07	81
Fiji	0.186	0.201	-0.073	-0.174	2.666	-0.171	0.142	0.069	82
Paraguay	-0.028	0.19	-0.244	-0.212	0.061	-0.16	0.086	0.065	83
Thailand	0.136	0.151	-0.087	-0.155	-0.967	-0.122	0.155	0.063	84
Algeria	-0.203	0.153	-0.171	-0.136	0.381	-0.124	0.2	0.058	85
Honduras	-0.04	0.143	-0.221	-0.167	0.381	-0.114	0.018	0.052	86

Continued on next page

Table 1: Complete scores for the NQLI data

country	RPC	LC	FA	PCA	SOM	min-max	pairwise-distance	NQLI	Rank
Egypt	-0.001	0.131	-0.044	-0.09	0.663	-0.103	0.138	0.051	87
Peru	-0.215	0.107	-0.259	-0.151	-0.74	-0.08	0.197	0.045	88
Vietnam	-0.119	0.124	-0.424	-0.23	-1.484	-0.096	0.051	0.044	89
Maldives	0.021	0.104	-0.081	-0.091	1.141	-0.077	0.021	0.043	90
Nicaragua	-0.076	0.108	-0.205	-0.129	0.381	-0.081	-0.018	0.041	91
Georgia	-0.075	0.095	-0.176	-0.112	0.381	-0.068	0.001	0.038	92
Ukraine	0.214	0.042	0.077	-0.013	0.993	-0.018	0.011	0.028	93
Guatemala	0.001	0.037	-0.008	-0.016	1.141	-0.013	-0.008	0.023	94
Suriname	0.031	-0.001	0.082	0.037	1.211	0.024	0.018	0.011	95
Vanuatu	0.083	-0.003	0.02	0.015	1.141	0.025	-0.157	0.011	96
Moldova	0.279	-0.04	0.044	0.02	1.845	0.06	-0.23	0.002	97
Morocco	-0.031	-0.027	-0.018	0.027	0.381	0.048	-0.127	0.002	98
Cape Verde	0.036	-0.037	-0.072	-0.001	-1.193	0.058	-0.187	-0.001	99
Iraq	0.116	-0.046	0.109	0.08	0.663	0.066	-0.18	-0.002	100
Micronesia, Fed. Sts.	0.097	-0.064	0.139	0.099	1.141	0.083	-0.084	-0.009	101
Kazakhstan	0.298	-0.099	0.312	0.172	-0.107	0.116	-0.028	-0.023	102
Kyrgyzstan	0.273	-0.219	0.245	0.231	0.322	0.231	-0.462	-0.055	103
Indonesia	0.057	-0.198	0.058	0.146	-0.276	0.21	-0.254	-0.056	104
Uzbekistan	0.231	-0.224	0.246	0.239	0.472	0.235	-0.405	-0.057	105
Philippines	-0.038	-0.223	-0.031	0.131	-0.06	0.235	-0.251	-0.068	106
Turkmenistan	0.365	-0.266	0.449	0.346	-1.369	0.275	-0.254	-0.073	107
Comoros	0.452	-0.352	0.501	0.421	-0.053	0.356	-0.65	-0.091	108
Azerbaijan	-0.219	-0.32	0.328	0.38	-1.153	0.327	-0.144	-0.102	109
Guyana	0.307	-0.353	0.422	0.388	-0.201	0.357	-0.398	-0.104	110
Mongolia	0.367	-0.363	0.38	0.363	-0.373	0.367	-0.451	-0.108	111
Nepal	0.378	-0.466	0.471	0.473	-0.654	0.465	-0.781	-0.137	112
Sao Tome and Principe	0.325	-0.474	0.54	0.522	-0.922	0.473	-0.663	-0.137	113
Solomon Islands	0.362	-0.471	0.52	0.503	1.114	0.469	-0.645	-0.139	114
Bolivia	0.328	-0.488	0.524	0.511	-0.073	0.485	-0.454	-0.156	115
Eritrea	0.956	-0.57	0.795	0.651	0.994	0.564	-0.994	-0.157	116
Tajikistan	0.286	-0.54	0.517	0.547	-0.17	0.535	-0.709	-0.166	117
Laos	0.428	-0.566	0.625	0.604	-0.922	0.56	-0.7	-0.171	118
India	0.506	-0.581	0.657	0.621	-0.922	0.575	-0.659	-0.178	119
Bangladesh	0.393	-0.582	0.559	0.58	-1.219	0.576	-0.841	-0.181	120
Yemen, Rep.	0.524	-0.63	0.776	0.714	-0.073	0.622	-0.592	-0.188	121
Korea, Dem. Rep.	0.225	-0.575	0.411	0.51	-0.762	0.568	-0.722	-0.193	122
Pakistan	0.148	-0.656	0.631	0.682	-2.054	0.647	-0.601	-0.21	123
Bhutan	0.229	-0.668	0.623	0.67	-2.506	0.658	-0.531	-0.229	124
Benin	0.603	-0.781	0.906	0.863	-2.309	0.768	-0.887	-0.232	125
Papua New Guinea	0.758	-0.735	0.801	0.757	-0.582	0.723	-0.841	-0.235	126
Myanmar	0.64	-0.774	0.848	0.82	-1.064	0.761	-0.971	-0.237	127

Continued on next page

Table 1: Complete scores for the NQLI data

country	RPC	LC	FA	PCA	SOM	min-max	pairwise-distance	NQLI	Rank
Sudan	0.952	-0.834	0.951	0.878	-0.48	0.819	-0.832	-0.268	128
Timor-Leste	0.762	-0.813	0.829	0.808	-0.461	0.799	-0.818	-0.276	129
Gabon	0.489	-0.668	0.848	0.754	0.82	0.658	-0.24	-0.278	130
Ghana	0.994	-0.921	1.034	0.973	-0.654	0.905	-1.092	-0.287	131
Madagascar	0.785	-0.903	0.944	0.927	-0.467	0.887	-1.167	-0.288	132
Senegal	1.174	-0.932	1.056	0.971	0.024	0.916	-1.045	-0.301	133
Haiti	0.665	-0.925	0.866	0.9	-0.461	0.909	-1.102	-0.316	134
Togo	0.54	-0.99	0.875	0.949	-0.274	0.973	-1.233	-0.344	135
Mauritania	0.961	-1.051	1.085	1.068	2.372	1.032	-1.106	-0.349	136
Cameroon	1.468	-1.123	1.306	1.202	0.436	1.103	-1.078	-0.349	137
Tanzania	1.237	-1.112	1.183	1.136	0.814	1.092	-1.323	-0.363	138
Guinea	0.913	-1.151	1.174	1.18	1.752	1.131	-1.357	-0.369	139
Somalia	1.565	-1.257	1.378	1.305	0.167	1.236	-1.485	-0.397	140
Malawi	1.457	-1.243	1.304	1.257	-0.869	1.222	-1.538	-0.411	141
Kenya	1.308	-1.23	1.261	1.234	1.622	1.209	-1.338	-0.417	142
Cambodia	0.667	-1.125	0.976	1.059	0.141	1.106	-1.139	-0.419	143
Uganda	1.522	-1.263	1.333	1.279	-0.869	1.242	-1.485	-0.42	144
Ethiopia	1.196	-1.236	1.224	1.224	0.519	1.215	-1.557	-0.421	145
Burkina Faso	1.216	-1.349	1.398	1.392	1.426	1.327	-1.371	-0.422	146
Congo, Rep.	1.172	-1.215	1.237	1.219	-0.621	1.195	-0.975	-0.433	147
Cote d'Ivoire	0.916	-1.259	1.192	1.237	-1.192	1.238	-1.285	-0.44	148
Guinea-Bissau	1.715	-1.456	1.562	1.51	1.616	1.432	-1.696	-0.444	149
Niger	1.365	-1.519	1.567	1.57	0.108	1.495	-1.542	-0.453	150
Equatorial Guinea	1.264	-1.304	1.49	1.415	-1.563	1.283	-0.57	-0.46	151
Nigeria	1.712	-1.415	1.503	1.447	0.247	1.393	-1.366	-0.462	152
Liberia	0.588	-1.483	1.374	1.487	-0.773	1.46	-1.552	-0.464	153
Chad	1.525	-1.515	1.546	1.538	-0.962	1.491	-1.366	-0.486	154
Mali	1.656	-1.535	1.57	1.553	1.57	1.51	-1.609	-0.491	155
Rwanda	1.609	-1.478	1.486	1.473	-0.419	1.454	-1.681	-0.496	156
Burundi	1.522	-1.504	1.492	1.496	0.195	1.48	-1.763	-0.497	157
Central African Republic	1.784	-1.565	1.598	1.574	1.57	1.54	-1.782	-0.505	158
Afghanistan	1.971	-1.753	1.817	1.796	1.312	1.726	-1.676	-0.512	159
Mozambique	1.673	-1.523	1.518	1.508	-0.673	1.499	-1.734	-0.516	160
Namibia	0.652	-1.185	0.975	1.072	-0.164	1.165	-0.654	-0.528	161
Angola	1.69	-1.694	1.726	1.724	0.931	1.667	-1.219	-0.533	162
Congo, Dem. Rep.	1.588	-1.689	1.62	1.658	0.404	1.663	-1.893	-0.561	163
Zambia	2.068	-1.724	1.694	1.681	0.126	1.697	-1.705	-0.606	164
Lesotho	1.877	-1.709	1.643	1.655	-2.489	1.683	-1.638	-0.614	165
Botswana	1.093	-1.469	1.412	1.437	0.147	1.446	-0.579	-0.633	166
South Africa	1.146	-1.438	1.299	1.344	-0.831	1.415	-0.623	-0.652	167
Djibouti	0.911	-1.608	1.332	1.475	1.253	1.583	-1.243	-0.655	168

Continued on next page

Table 1: Complete scores for the NQLI data

country	RPC	LC	FA	PCA	SOM	min-max	pairwise-distance	NQLI	Rank
Sierra Leone	1.583	-1.948	1.776	1.871	0.051	1.917	-1.85	-0.664	169
Zimbabwe	2.218	-1.759	1.666	1.659	1.365	1.731	-1.835	-0.68	170
Swaziland	1.655	-2.084	1.758	1.894	-1.506	2.049	-1.229	-0.876	171

Table 2: Ranks of different methods for a selection of countries

country	RPC	LC	FA	PCA	SOM	min-max	pairwise-distance	NQLI
Luxembourg	1	1	5	1	21	1	8	1
Austria	14	14	14	12	120	14	12	11
France	15	17	10	15	29	17	16	21
Korea, Rep.	29	34	29	32	10	34	35	30
Chile	37	37	35	35	25	37	36	41
Grenada	49	48	47	48	126	48	44	50
Macedonia, FYR	59	59	53	56	144	59	57	60
Sri Lanka	78	65	56	63	116	65	74	71
China	77	80	70	76	49	80	78	81
Nicaragua	82	89	81	88	108	89	95	91
Micronesia, Fed. Sts.	98	101	102	101	152	101	97	101
Mongolia	117	111	108	108	64	111	110	111
Yemen, Rep.	124	122	123	124	76	122	116	121
Ghana	141	132	136	135	51	132	138	131
Malawi	152	145	146	146	40	145	156	141
Equatorial Guinea	149	149	155	151	9	149	114	151
Namibia	129	141	134	138	74	141	121	161
Swaziland	161	171	169	171	10	171	144	171

Table 3: Complete list of the boxing dataset

id	height	weight	skill	physical	avg. heart rate	avg. punch impact	avg. punch speed	number of punches	punches de-fended	punches missed
0	190	72	1	1	101.58	9358.76	16.96	26	25	0
1	150	66.1	0	0	142.94	1914.69	7.84	26	0	19
2	185	101.5	0.32	0.43	124.31	4951.63	12.22	30	11	13
3	177	68.8	0.53	0.47	125.22	5718.13	11.73	29	11	12
4	186	94.4	0.59	0.64	120.07	7607.21	11.44	31	18	5
5	177	69.2	0.38	0.48	124.94	4816.55	8.79	25	14	12
6	172	88	0.57	0.48	131.51	6979.82	11.69	38	16	11
7	182	72	0.39	0.36	127.45	4987.98	12.12	35	17	13
8	186	90.8	0.3	0.48	128.72	5148.1	12.28	27	9	12
9	185	80.6	0.38	0.49	131.21	5244.81	11.98	33	15	14
10	184	78.6	0.52	0.51	128.25	6097.88	11.62	27	15	8
11	173	75.8	0.29	0.58	117.98	4228.42	10.56	22	6	15
12	169	72.2	0.5	0.34	132.14	5311.5	9.66	25	22	8
13	167	53.7	0.53	0.36	125.9	4788.54	12.35	21	11	11
14	166	64.2	0.4	0.59	127.95	4502.14	9.23	32	27	8
15	176	82	0.66	0.54	130.19	7518.74	12.02	30	25	8
16	163	50	0.54	0.64	122.34	4753.42	12.67	24	23	8
17	169	76	0.43	0.25	140	5142.16	11.14	19	16	13
18	168	60.3	0.38	0.52	131.54	4269.59	12.41	33	6	7
19	160	62.8	0.27	0.38	134.43	3636.39	9.95	29	9	13

```
{
  "id_name": "id",
  "group_min_size": 5,
  "personal_info": [
    {
      "name": "height",
      "position": 0,
      "split_value": [
        170,
        190
      ]
    },
    {
      "name": "weight",
      "position": 1,
      "split_value": [
        60,
        80
      ]
    }
  ],
  "performance_info": [
    {
      "name": "number_of_punches",
      "desired_result": "high"
    },
    {
      "name": "avg_punch_speed",
      "desired_result": "high"
    },
    {
      "name": "avg_punch_impact",
      "desired_result": "high"
    },
    {
      "name": "punches_defended",
      "desired_result": "high"
    },
    {
      "name": "avg_heart_rate",
      "desired_result": "low"
    },
    {
      "name": "punches_missed",
      "desired_result": "low"
    }
  ]
}
```

Listing 4: Json schema for boxing exercise