SAPIENZA
UNIVERSITÀ DI ROMA

# Formulation of an ML-based model for the Assessment of Maximum Sprint Capability in Elite Soccer Players


Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Ingegneria Gestionale

**Giacomo Bacchetta, 1840949**
**Edoardo Cesaroni, 1841742**

Advisor
Prof. Marco Sciandrone

Co-Advisor
Dr. Davide Gallo



Academic Year 2022/2023

Thesis defended on 21 July 2023
in front of a Board of Examiners composed by:

Prof. Marco Sciandrone (chairman)

Prof. Idiano D'Adamo

Prof. Alberto De Santis

Prof.ssa Marianna De Santis

Prof. Vincenzo Eramo

Prof. Luca Fraccascia

Prof. Roberto Li Voti

Prof. Massimo Roma

---

**Formulation of an ML-based model for the Assessment of Maximum Sprint Capability in Elite Soccer Players**

Sapienza University of Rome



This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: bacchetta.1840949@studenti.uniroma1.it
cesaroni.1841742@studenti.uniroma1.it

# Abstract

In recent years, machine learning techniques have gained widespread popularity, fueled by the availability of vast amounts of data across various industries.
The sports industry, including football, has also embraced these techniques to gain a competitive edge.
In our work, we focus on estimating and analyzing the maximum acceleration capability of football players using tracking data, which defines the player's position on the field at each frame during a match. This information can contribute to more precise tactical preparation and personalized training strategies.
We then develop a regression-based model that reconstructs a profile per player, capturing the relationship between acceleration and speed during a maximum sprint when the athlete performs at their peak capability.
By applying this model to data sampled at regular intervals during a match, we can identify variations in this profile, which reflect changes in the player's physical condition. It also allows us to evaluate how the time it takes for a player to cover specific distances at a certain initial speed varies over 90 minutes.
Furthermore, we employ unsupervised learning techniques, such as clustering, to identify players with similar athletic characteristics and performance. This approach deepens our understanding of player capabilities and enables data-driven decision-making in sports analysis and training. It provides a deeper level of insight into player performance, facilitating effective player evaluation, team selection, and training program optimization.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Machine learning and its application fields

> "Field of study that gives computers the ability to learn without being explicitly programmed."

This is the definition coined in 1959 by **Arthur Samuel**, an American pioneer in the field of computer gaming and artificial intelligence, marking the inception of **machine learning**. Since then, machine learning has undergone extensive research, development, and transformations, emerging as one of academia's most crucial and influential fields.
Machine learning aims to develop algorithms and models that can learn from data, improve their performance over time, and can effectively deal with three main problems:

1. **Regression**, is a *supervised learning*[1] problem where the purpose is to model and analyze the relationship between one or more continuous dependent variables and one or more independent ones. The goal is to estimate and predict the values of the dependent variable based on the available data;

2. **Classification**, where outputs are divided into two or more predefined classes, and the learning system aims to produce a model that assigns new inputs to classes. With the regression, described in the previous point, classification is a technique used in *supervised learning*;

[1] *Supervised learning* is a subcategory of machine learning which is defined by its use of labeled datasets to train algorithms that classify data or predict outcomes accurately.

3. **Clustering**, which is the most important expression of *unsupervised learning*[2], and it aims to spread a set of inputs into clusters without relying on prior experience, trying to maximize the *inter-cluster* (between each pair of different clusters) distance, or difference, and minimize the *intra-cluster* (into every cluster) distance.

Nowadays, thanks to extensive research and persistent academic studies, these machine learning techniques, which find their greatest expression within artificial intelligence, are implemented in various fields.

In the **finance sector**, they are used for fraud detection, identity theft prevention, stock market forecasting, and portfolio management (i.e. robo-advisors).
In the field of **health care**, these techniques play a crucial role in predicting the onset of specific diseases, facilitating AI-assisted surgeries, and enabling the development of personalized treatment options.
In the **energy sector**, they optimize energy consumption and make our homes smarter through the use of home automation. Still, they also can be used both in the wind and solar energy prediction using *Long Short-Term Memory* (LSTM) network.
In the **automotive industry**, they contribute to the development of autonomous vehicles and the optimization of the supply chain.
Of course, the **sports industry** could not be left out among these fields, where machine learning has increasingly gained value in recent years. Among the various tasks that machine learning can perform in this sector, we find analyzing not only players' technical-tactical performances but also their athletic abilities, supporting scouting efforts, enhancing accuracy in referee decisions, and assisting coaches in decision-making. It can be used as an auxiliary component that plays a crucial role in both the directional and managerial processes of a sports organization, as well as in enhancing fan engagement through the analysis of personal data.
In conclusion, machine learning offers sports organizations an opportunity to gain a competitive advantage, not only in terms of on-field success but also in commercial endeavors.

[2] *Unsupervised learning* uses machine learning to analyze and cluster unlabeled datasets.

## 1.2 The aim of the work

In the current context, we are referring to the sports industry, with a specific focus on the world's most renowned sport: **soccer**.
To clarify the objectives of this study, let's delve into an example: we suppose to have two players, Messi and Van Dijk, standing side by side on the back line of the field. If we were to instruct them to accelerate to their maximum capability, starting from a stationary position simultaneously, who would reach the halfway line first? The answer seems evident: "the faster of the two" (in this case, likely Van Dijk), and, considering a distance of approximately 55 meters, this reasoning holds valid.
Now, let's shift our focus to the penalty area line. With a reduced distance of 16 meters, it is reasonable to assume that the "quicker" player, the one with greater acceleration in the initial meters, possibly Messi, would be the first to arrive.
Drawing a parallel example, let's consider two cars: one equipped with a 120 horsepower turbo diesel engine and the other with a naturally aspirated engine boasting 200 horsepower. In a longer race, the car with 200 horsepower would likely prevail. However, in the first 100 meters, the turbocharged car would accelerate more rapidly and probably cross the finish line first.
From these examples, it becomes evident how crucial it is to distinguish between **maximum speed** and **maximum acceleration**.
Now, let's move on to a second scenario: picture the same two soccer players running at an equal speed of 8 $km/h$ on an athletic track, side by side. If we were to ask them to accelerate to their maximum capacity, who would cover the next 20 meters first? Would the outcome remain the same if their initial speed were 18 $km/h$ instead of 8 $km/h$? Furthermore, if Messi were running at 11 $km/h$ and Van Dijk at 7 $km/h$, who would complete the following 30 meters first? These inquiries shed light on the importance of considering the maximal acceleration capability at different velocities of each soccer player.

Based on these considerations, this work aims to develop a model that can accurately estimate a **soccer player's maximum acceleration capability** and the time required to cover a given distance during a game based on these evaluations.
The model will take the following inputs:

- The specific soccer player being analyzed;
- The distance that needs to be covered;
- The initial speed at which the sprint starts.

As an output, the model will provide the time taken by the player to complete the target distance while running in a straight line and accelerating to the best of their ability. Additionally, it will be crucial to take into account the timing within the game when the sprint is performed, to factor in the potential decline in acceleration capacity as the game progresses.
Once the model is built, the plan is to simulate various scenarios for each player, incorporating different distances to be covered and initial speeds, to estimate the corresponding travel times. Using these times, we will proceed with identifying meaningful clusters that classify players based on their similar acceleration capabilities.

In the end , it is crucial to underscore the tactical nature of this research, which focuses on assessing the acceleration capabilities of soccer players rather than the physiological or conditional parameters typically examined by sports scientists. The primary objective is to ascertain which soccer player exhibits greater efficiency in terms of acceleration compared to others.

## 1.3 Case study

The case study we will refer to consists of three matches played during the **Lega Serie A Tim 2022-2023 season**.
The data used for the development of this project were kindly offered to us by our co-advisor.
The matches in question are as follows:

- Lecce vs Milan on 14 January 2023;
- Milan vs Atalanta on 26 February 2023;
- Napoli vs Milan on 2 April 2023.

For each match, as mentioned above, we were provided with different data, that can be stored either in JSON format or in CSV or xlsx format:

- ***Match data***, which contains all the information related to the reference match, including date, time, teams and their formations, match-day number, and many other details;
- ***Tracking data***, which contains the positional data of each player, taking into account two coordinates (one along the length of the field and one along its width) at each timestamp, i.e., at each frame;

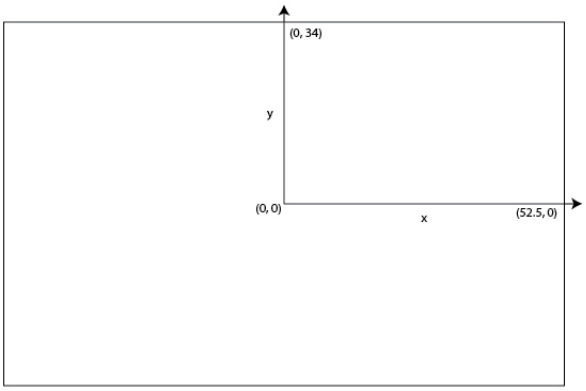



**Figure 1.1.** Coordinate system used to format the player's positioning data.

These data are obtained thanks to AI-powered video tracking technologies, which leverage computer vision and machine learning to generate tracking data from televised soccer broadcasts.
The raw data obtained in this manner requires a pre-processing stage to address the presence of noise, similar to other technologies like the Global

Positioning System (GPS) or Local Positioning Systems (LPS), used to acquire tracking data.

- ***Possession data***, which contains the sequence of possession phases throughout the entire match.

The data sources that were made available to us allowed us to collect information on around 70 players, considering both goalkeepers and substitutes, distributed among the 4 teams (Milan, Lecce, Napoli, Atalanta) as shown in the table:

| **Milan** | **Lecce** | **Napoli** | **Atalanta** | **Total** |
|---|---|---|---|---|
| 24 | 16 | 16 | 16 | 72 |

**Table 1.1.** Analysable players per match

# Chapter 2

# The model development

The purpose of this section is to provide a comprehensive description of the model development process, which includes data pre-processing, multiple regression, and clustering techniques.
We will first delve into the mathematical and analytical aspects of the model, followed by presenting the results obtained from applying it to the case study introduced in the previous section §1.3.

The implementation of the model presented is available on **GitHub**.
We would like to emphasize that the case study data is not included in the repository due to company privacy reasons.

## 2.1 Data pre-processing

The initial stage of the work involves **data pre-processing**, a term that encompasses the extraction, cleaning, and alignment of data from their source to ensure consistency with the intended model. The significance of this step increases with the volume of data that needs manipulation and the likelihood of encountering errors, as it directly impacts the success of the project.
As previously mentioned, tracking data, in particular, are prone to **noise** $\boldsymbol{\eta}$, and the extent of noise can vary between games and sampling frequencies, as explained in Van Breugel et al, 2020 [2]. The presence of noise becomes more critical since positional data must not only be used to calculate displacement but also velocity and acceleration during a match, amplifying the potential for errors.
The computation of derivatives from noisy measurement data is a common practice in physical, engineering, and biological sciences, often serving as a vital step in developing dynamic models or designing control systems. However, the mathematical formulation of numerical differentiation for noisy data is typically ill-posed, leaving

researchers to resort to ad hoc processes for selecting from various computational methods and their parameters.
Hence, it is crucial to give careful attention to this initial phase of the project, aiming to obtain data that are as reliable as possible. By doing so, the resulting implementation can yield accurate and robust outcomes.

### 2.1.1 Data extraction

The initial step in data pre-processing involves understanding the data sources and their formats. Once this is established, we can proceed with the **data extraction** process.
The data needed for our work is stored in three different files:

1. ***match_data***;
2. ***tracking_data***;
3. ***possession_data***.

The storage formats of these files may vary, including options such as xlsx, CSV, or JSON, depending on the specific match and the service provider employed within the football industry. The service provider, in this context, pertains to the organization responsible for gathering and delivering football-related data and information to their clientele.
First and foremost, it is important to extract information about the match being analyzed and the players of the respective teams.
This information is contained within the files called *match_data*. Using the *Pandas* library, we can convert the information from any file into a *Pandas dataframe*, as shown below.

| | start_time | end_time | number | first_name | last_name | trackable_object | team_opta_id | team |
|---|---|---|---|---|---|---|---|---|
| **0** | 00:00:00 | 00:45:02 | 56 | Alexis | Saelemaekers | 14740 | 120 | Milan |
| **1** | 01:25:21 | NaN | 40 | Aster | Vranckx | 28775 | 120 | Milan |
| **2** | 00:00:00 | 01:08:44 | 10 | Brahim | Diaz | 11673 | 120 | Milan |
| **3** | 00:00:00 | NaN | 1 | Ciprian | Tatarusanu | 1929 | 120 | Milan |
| **4** | 00:00:00 | 01:25:21 | 2 | Davide | Calabria | 2534 | 120 | Milan |
| **5** | 01:08:44 | NaN | 27 | Divock | Origi | 2792 | 120 | Milan |
| **6** | 00:00:00 | NaN | 23 | Fikayo | Tomori | 18976 | 120 | Milan |
| **7** | 00:00:00 | NaN | 4 | Ismael | Bennacer | 14322 | 120 | Milan |
| **8** | 00:45:02 | NaN | 30 | Junior | Messias | 28648 | 120 | Milan |
| **9** | 00:00:00 | NaN | 9 | Olivier | Giroud | 8216 | 120 | Milan |
| **10** | 00:00:00 | NaN | 20 | Pierre | Kalulu | 25829 | 120 | Milan |
| **11** | 00:00:00 | NaN | 17 | Rafael | Leao | 13008 | 120 | Milan |
| **12** | 00:45:02 | NaN | 21 | Sergino | Dest | 24022 | 120 | Milan |
| **13** | 01:25:21 | NaN | 24 | Simon | Kjaer | 9876 | 120 | Milan |
| **14** | 00:00:00 | 00:45:02 | 19 | Théo | Hernandez | 11694 | 120 | Milan |
| **15** | 00:00:00 | 01:25:21 | 32 | Tommaso | Pobega | 28701 | 120 | Milan |
| **17** | 00:00:00 | NaN | 29 | Alexis | Blin | 585 | 130 | Lecce |
| **18** | 01:01:59 | NaN | 25 | Antonino | Gallo | 19881 | 130 | Lecce |
| **20** | 00:00:00 | NaN | 6 | Federico | Baschirotto | 64281 | 130 | Lecce |
| **22** | 00:00:00 | 01:11:48 | 11 | Federico | Di Francesco | 3346 | 130 | Lecce |
| **23** | 00:00:00 | 01:28:18 | 27 | Gabriel | Strefezza | 20030 | 130 | Lecce |
| **24** | 00:00:00 | 01:01:59 | 97 | Giuseppe | Pezzella | 3982 | 130 | Lecce |
| **25** | 00:00:00 | 01:01:59 | 16 | Joan Cañellas | Gonzàlez | 72031 | 130 | Lecce |
| **26** | 01:11:48 | NaN | 31 | Joel | Voelkerling Persson | 185144 | 130 | Lecce |
| **28** | 01:11:48 | NaN | 22 | Lameck | Banda | 28888 | 130 | Lecce |
| **29** | 00:00:00 | 01:11:48 | 9 | Lorenzo | Colombo | 27074 | 130 | Lecce |
| **32** | 00:00:00 | NaN | 42 | Morten | Due Hjulmand | 29860 | 130 | Lecce |
| **33** | 01:28:18 | NaN | 28 | Rémi | Oudin | 12405 | 130 | Lecce |
| **34** | 00:00:00 | NaN | 93 | Samuel | Umtiti | 9533 | 130 | Lecce |
| **36** | 00:00:00 | NaN | 17 | Valentin | Gendrey | 12349 | 130 | Lecce |
| **37** | 00:00:00 | NaN | 30 | Wladimiro | Falcone | 14334 | 130 | Lecce |
| **38** | 01:01:59 | NaN | 32 | Youssef | Maleh | 28736 | 130 | Lecce |

**Figure 2.1.** *match_data* for Lecce-Milan

| | start_time | end_time | number | first_name | last_name | trackable_object | team_opta_id | team |
|---|---|---|---|---|---|---|---|---|
| **0** | 00:00:00 | 01:23:41 | 7 | Teun | Koopmeiners | 14512 | 456 | Atalanta |
| **1** | 01:23:41 | NaN | 22 | Matteo | Ruggeri | 33586 | 456 | Atalanta |
| **2** | 00:00:00 | 01:23:41 | 42 | Giorgio | Scalvini | 35197 | 456 | Atalanta |
| **3** | 01:23:41 | NaN | 23 | Lukas | Vorlicky | 408239 | 456 | Atalanta |
| **4** | 00:00:00 | 01:23:41 | 77 | Davide | Zappacosta | 2551 | 456 | Atalanta |
| **5** | 01:23:41 | NaN | 6 | Jose | Luis Palomino | 5424 | 456 | Atalanta |
| **6** | 00:00:00 | 01:08:14 | 11 | Ademola | Lookman | 11517 | 456 | Atalanta |
| **7** | 01:08:14 | NaN | 9 | Luis | Muriel | 6449 | 456 | Atalanta |
| **8** | 00:00:00 | 01:02:32 | 13 | Josè | Edérson | 32697 | 456 | Atalanta |
| **9** | 01:02:32 | NaN | 10 | Jeremie | Boga | 4930 | 456 | Atalanta |
| **10** | 00:00:00 | 01:28:10 | 17 | Rafael | Leao | 13008 | 120 | Milan |
| **11** | 01:28:10 | NaN | 56 | Alexis | Saelemaekers | 14740 | 120 | Milan |
| **12** | 00:00:00 | 01:28:10 | 30 | Junior | Messias | 28648 | 120 | Milan |
| **13** | 01:28:10 | NaN | 12 | Ante | Rebic | 913 | 120 | Milan |
| **14** | 00:00:00 | 01:13:36 | 9 | Olivier | Giroud | 8216 | 120 | Milan |
| **15** | 01:13:36 | NaN | 11 | Zlatan | Ibrahimovic | 11132 | 120 | Milan |
| **16** | 00:00:00 | 01:13:36 | 10 | Brahim | Diaz | 11673 | 120 | Milan |
| **17** | 01:13:36 | NaN | 90 | Charles | De Ketelaere | 25273 | 120 | Milan |
| **24** | 00:00:00 | NaN | 2 | Rafael | Toloi | 8822 | 456 | Atalanta |
| **25** | 00:00:00 | NaN | 17 | Rasmus Winther | Hojlund | 34128 | 456 | Atalanta |
| **26** | 00:00:00 | NaN | 15 | Marten | De Roon | 7027 | 456 | Atalanta |
| **27** | 00:00:00 | NaN | 1 | Juan | Musso | 14282 | 456 | Atalanta |
| **28** | 00:00:00 | NaN | 3 | Joakim | Maehle | 14767 | 456 | Atalanta |
| **29** | 00:00:00 | NaN | 19 | Berat | Djimsiti | 1327 | 456 | Atalanta |
| **40** | 00:00:00 | NaN | 19 | Theo | Hernandez | 11694 | 120 | Milan |
| **41** | 00:00:00 | NaN | 8 | Sandro | Tonali | 20041 | 120 | Milan |
| **42** | 00:00:00 | NaN | 33 | Rade | Krunic | 8782 | 120 | Milan |
| **43** | 00:00:00 | NaN | 20 | Pierre | Kalulu | 25829 | 120 | Milan |
| **44** | 00:00:00 | NaN | 28 | Malick | Thiaw | 26042 | 120 | Milan |
| **45** | 00:00:00 | NaN | 16 | Mike | Maignan | 7614 | 120 | Milan |
| **46** | 00:00:00 | NaN | 23 | Fikayo | Tomori | 18976 | 120 | Milan |

**Figure 2.2.** *match_data* for Milan-Atalanta

| | start_time | end_time | number | first_name | last_name | trackable_object | team_opta_id | team |
|---|---|---|---|---|---|---|---|---|
| **0** | 00:00:00 | 01:22:35 | 33 | Rade | Krunic | 8782 | 120 | Milan |
| **1** | 01:22:35 | NaN | 14 | Tiemoue | Bakayoko | 10361 | 120 | Milan |
| **2** | 00:00:00 | 01:22:35 | 4 | Ismael | Bennacer | 14322 | 120 | Milan |
| **3** | 01:22:35 | NaN | 90 | Charles | De Ketelaere | 25273 | 120 | Milan |
| **4** | 00:00:00 | 01:12:58 | 17 | Rafael | Leao | 13008 | 120 | Milan |
| **5** | 01:12:58 | NaN | 27 | Divock | Origi | 2792 | 120 | Milan |
| **6** | 00:00:00 | 01:12:58 | 9 | Olivier | Giroud | 8216 | 120 | Milan |
| **7** | 01:12:58 | NaN | 12 | Ante | Rebic | 913 | 120 | Milan |
| **8** | 00:56:03 | NaN | 56 | Alexis | Saelemaekers | 14740 | 120 | Milan |
| **9** | 00:00:00 | 01:20:26 | 3 | Min-Jae | Kim | 26471 | 459 | Napoli |
| **10** | 01:20:26 | NaN | 5 | Juan | Jesus | 11857 | 459 | Napoli |
| **11** | 00:00:00 | 01:16:06 | 18 | Giovanni | Simeone | 11324 | 459 | Napoli |
| **12** | 01:16:06 | NaN | 81 | Giacomo | Raspadori | 14365 | 459 | Napoli |
| **13** | 00:00:00 | 01:07:13 | 68 | Stanislav | Lobotka | 11233 | 459 | Napoli |
| **14** | 01:07:13 | NaN | 91 | Tanguy | N'Dombele | 11177 | 459 | Napoli |
| **15** | 00:00:00 | 01:07:13 | 20 | Piotr | Zielinski | 8733 | 459 | Napoli |
| **16** | 01:07:13 | NaN | 11 | Hirving | Lozano | 12014 | 459 | Napoli |
| **17** | 00:00:00 | 01:07:13 | 21 | Matteo | Politano | 7246 | 459 | Napoli |
| **18** | 01:07:13 | NaN | 7 | Eljif | Elmas | 13970 | 459 | Napoli |
| **28** | 00:00:00 | NaN | 19 | Theo | Hernandez | 11694 | 120 | Milan |
| **29** | 00:00:00 | NaN | 8 | Sandro | Tonali | 20041 | 120 | Milan |
| **30** | 00:00:00 | NaN | 24 | Simon | Kjaer | 9876 | 120 | Milan |
| **31** | 00:00:00 | NaN | 16 | Mike | Maignan | 7614 | 120 | Milan |
| **32** | 00:00:00 | NaN | 23 | Fikayo | Tomori | 18976 | 120 | Milan |
| **33** | 00:00:00 | NaN | 2 | Davide | Calabria | 2534 | 120 | Milan |
| **34** | 00:00:00 | 00:56:03 | 10 | Brahim | Diaz | 11673 | 120 | Milan |
| **42** | 00:00:00 | NaN | 6 | Mario | Rui | 6915 | 459 | Napoli |
| **43** | 00:00:00 | NaN | 77 | Khvicha | Kvaratskhelia | 24477 | 459 | Napoli |
| **44** | 00:00:00 | NaN | 22 | Giovanni | Di Lorenzo | 14319 | 459 | Napoli |
| **45** | 00:00:00 | NaN | 99 | Andre | Anguissa | 723 | 459 | Napoli |
| **46** | 00:00:00 | NaN | 13 | Amir | Rrahmani | 13777 | 459 | Napoli |
| **47** | 00:00:00 | NaN | 1 | Alex | Meret | 13642 | 459 | Napoli |

**Figure 2.3.** *match_data* for Napoli-Milan

As depicted in Figure [2.1], Figure [2.2], and Figure [2.3], each player is characterized by a set of attributes that encompass their first and last name, jersey number, team affiliation, and playing time (captured by the *start_time* and *end_time* variables). Furthermore, two additional attributes, namely the ***trackable object*** and ***team opta ID***, are also included.
The *trackable_object* code is a unique key that identifies a player within the tracking provider and it also serves to identify the same player in different matches, while the *team_opta_id* is a unique code associated with each team of the Serie A, the Italian league.
Among all this information, the relevant ones for our work are undoubtedly the player's last name, which is essential for recognizing which player it refers to, and their corresponding *trackable_object*, which allows us to identify the player within the *tracking_data* files and across multiple analyzed matches.
Next, we move on to processing the information contained within the *tracking_data*. This is the most complex file of the three, as each provider constructs and manages it differently, resulting in variations in both structure and format.
In general, the tracking data for each player is collected within a dictionary, where the *trackable_object* serves as the key, and the values consist of the x and y coordinates representing the player's position within the game rectangle in each timestamp of the match. The data are sampled at a frequency that can vary from 10 Hz (10 frames per second) to 5 Hz (5 frames per second).
Using the extracted information from the match and tracking files, we build a match-specific database referred to as *players*. This database is represented as a dictionary and saved as a JSON file. In this database, the player's unique *trackable_object* identifier serves as the key, while the associated values comprise the player's details from the *match_data* file. Additionally, the values also include the sequence of player position data recorded at each timestamp, providing a comprehensive record of the player's movements throughout the match.

To help the reader understand the structure of the database, we provide below an illustrative example related to a single study element.

```
players={
    14740:{
        'last_name': Saelemakers,
        'number': 77,
        'team': Milan,
        'start_time': hh:mm:ss.cc,
        'end_time': hh:mm:ss.cc,
        'tracking': [[period_1, frame_1, x_1, y_1],
                     [period_2, frame_2, x_2, y_2],
                     [period_3, frame_3, x_3, y_3],
                     ...]
                }
        ...}
```

To conclude the data extraction phase, we proceed with the processing of data within the *possession_data* file. This file indicates the time intervals during which the ball is in play, excluding moments when the game is stopped by the referee or during a corner kick or a throw-in, and the team that has control of the game.
From this information, we obtain a list of non-effective game time instances, which helps us further reduce the *tracking_data* of the players to only the moments of actual gameplay.

### 2.1.2 Data wrangling

Once the most important data have been extracted and the superfluous ones eliminated, we can proceed to the actual **processing** and **manipulation** of the former.
The aim of this step is to generate a list of instantaneous speeds and accelerations for each player in the players' dictionary. It is important to note that, similar to other studies, we make the assumption that players predominantly move in a forward direction and, consequently, we do not take into account information regarding backward or sideways movements (Osgnach et al., 2010 [3]).
To calculate the player's shift in each frame, we use the player's position along the x-axis and y-axis. This enables us to determine the displacement along each axis using the following equations:

$$\Delta x_i = x_i - x_{i-1} \tag{2.1}$$

$$\Delta y_i = y_i - y_{i-1} \tag{2.2}$$

where $i$ is the current frame and *i-1* is the previous one.
By considering these axial shifts, we can subsequently compute the distance $\Delta space_i$ covered by a player between two consecutive frames in meters.

$$\Delta space_i = \sqrt{\Delta x_i^2 + \Delta y_i^2} \tag{2.3}$$

We are now able to compute, for each frame, the instant speed $s$ sustained by the player in question. This is done by performing the simple ratio between the space traveled and the time taken to travel it:

$$s_i = \frac{\Delta space_i}{\Delta t} \tag{2.4}$$

and $\Delta t$ can be equal to 0.1 seconds or 0.2 seconds; it depends on the sampling rate used by the service provider during the match.

Continuing with the calculation of instantaneous acceleration using an additional finite difference of instantaneous velocities would further amplify the existing error in the velocity obtained through this method. This occurs, as we said previously because the derived velocity values inherit the noise present in the original tracking data.
Consequently, it becomes necessary to reduce the noise in the data, which can be achieved using a Python package called ***PyNumDiff***.
Suppose we have a generic time series affected by noise, denoted as $\eta$, and expressed as:

$$y = x + \eta \tag{2.5}$$

*PyNumDiff* is a Python package that provides several methods for calculating the numerical derivative $\hat{\dot{x}}$. As mentioned earlier, this step is crucial in the analysis, and the package also provides smoothed estimates $\hat{x}$ of the original time series, aiming to remove noise from the data.
PyNumDiff implements four distinct families of numerical differentiation methods: **finite difference** computation combined with a smoothing filter, **local approximation** utilizing linear models, methods based on **Kalman filtering**, and **regularization methods** based on total variation.
Each of these methods requires a set of parameters $\Phi$, which can include parameters such as the size of the smoothing window, number of iterations, filter order, cutoff frequency, and more. Determining the appropriate values for these parameters is often done through techniques like grid search or visual analysis of the data, depending on the data analyst's subjectivity.
This package introduces a general approach to streamline the parameter selection process for any chosen method. Its goal is to find the optimal set of parameters $\Phi$ that minimizes the **loss function L**. This loss function is defined as a weighted sum of two metrics computed from the derivative estimation: the **accuracy of the integral of the derivative** and the **smoothness of the derivative itself**. By optimizing the parameters, this approach ensures a balance between fidelity to the original data and achieving a desired level of smoothness in the estimation of the derivative.
The loss function is defined as follows:

$$L = RMSE(trapz(\hat{\dot{x}}(\Phi)) + \mu, y) + \gamma(TV(\hat{\dot{x}}(\Phi))) \tag{2.6}$$

where $\text{RMSE}(u, v) = ||u - v||_2$ denotes the Root-Mean-Squared Error, $trapz(\cdot)$ is the **discrete-time trapezoidal numerical integral** and $\mu$ resolves the unknown integration constant

$$\mu = \frac{1}{m}\sum_{k=0}^{m}(trapz(\hat{\dot{x}}(\Phi)) - y) \tag{2.7}$$

The hyperparameter $\gamma$ plays a crucial role in determining the desired smoothness of the resulting derivative. It can be set using a heuristic approach based on the following formula:

$$log(\gamma) = -1.6log(f) - 0.71log(\Delta t) - 5.1 \tag{2.8}$$

where, $f$ represents the estimated cutoff frequency, which can be obtained by counting the actual number of peaks per second in the data or by analyzing the power spectrum.

Examining the loss function, the first term ensures the accuracy of the derivative estimation by minimizing the discrepancy between the integral of the derivative and the original data. On the other hand, the second term promotes the smoothness of the derivative estimation itself. Specifically, the second term corresponds to the total variation:

$$TV(\hat{\dot{x}}(\Phi)) = \frac{1}{m} \left| \hat{\dot{x}}_{0:m-1}(\Phi) - \hat{\dot{x}}_{1:m}(\Phi) \right|_1 \tag{2.9}$$

Here, $m$ represents the timestamps in the data, and $|\cdot|_1$ denotes the $L1$ norm.

### 2.1.3 Case study

We have explored the various numerical derivation methods proposed using the optimization framework described in the previous section to select the optimal parameters.
Ultimately, we choose an **iterative first-order centered finite difference with an averaging filter** as the preferred method. This choice is made due to its superior performance in terms of computation time and the quality of the smoothed velocity estimate and its derivatives.
The other methods, with excessive smoothing of the data, attenuate sharp peaks in the data but also lead to an underestimation of the derivative's magnitude. Any outliers or anomalous peaks are subsequently addressed during the model construction phase.
The only parameter that needs to be set for this method is the number of iterations, which determines the level of data smoothing. Through the optimization process, we determined the appropriate number of iterations as shown in the following table:

| **Match** | **N° of iterations** |
|---|---|
| **Lecce-Milan** | 10 |
| **Milan-Atalanta** | 1 |
| **Napoli-Milan** | 1 |

**Table 2.1.** Parameter per match

The number of iterations selected for the method varies among different matches. Notably, Milan-Atalanta and Napoli-Milan have the same sample rate of 5 Hz, resulting in identical parameter settings. On the other hand, Lecce-Milan has a sample rate of 10 Hz.
This discrepancy is likely due to the increased necessity for data smoothing when sampled at a higher frequency. The higher frequency leads to a greater amplification of errors during the derivative estimation process using finite differences, as the time interval in the denominator becomes smaller.
By adjusting the number of iterations, the method effectively mitigates the impact of these errors, ensuring reliable and accurate results across different sampling rates.

The results we obtain, which will later be used to complete the work, can be represented as follows.

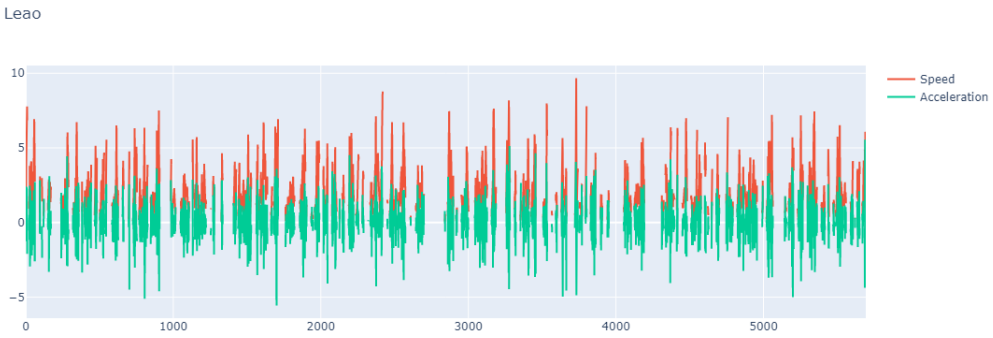



**Figure 2.4.** Graphical representation of speed and acceleration trends during a match

The image in Figure [2.4] displays the speed (in red) and acceleration (in green) as functions of time, measured in seconds.
There are gaps or discontinuities in these curves. These gaps occur during specific frames when the game is paused, such as when the ball is out of play, during a free-kick or corner kick, or when a throw-in is taking place.

## 2.2 AS Profile: a new version

The aim of the work conducted in the previous section §2.1 was to obtain the most accurate measurements of instantaneous velocities and accelerations, as they are the key information for addressing the objective of this thesis.
To define what **maximum acceleration capability** is and how we can quantify it, it is necessary to examine how *acceleration events* or *sprints* are treated in the football context (Osgnach, 2022 [4]).
In the performance analysis of soccer players, it is typical to consider an acceleration event when the instantaneous acceleration exceeds 2.5 $m/s^2$ for at least 0.5 $s$.
However, it is essential to acknowledge the limitations associated with this approach. Firstly, the threshold of 2.5 $m/s^2$ is arbitrarily determined, as is the minimum duration for exceeding this threshold.
Secondly, this criterion fails to account for the increasing difficulty of surpassing this threshold at higher speeds, thus disregarding a range of accelerations that involve greater effort performed at high velocities. This aspect can be found in the studies of Varley and Aughey, 2013 [5], who demonstrated that nearly half of the actions with maximal accelerations $> 2.78$ $m/s^2$ come from a velocity of less than 1 $m/s$, and only 8% of these maximal accelerations occurs when the starting speed is greater than 3 $m/s$.
Hence, it is evident that employing a universal acceleration threshold fails to consider individual variations and relies on subjective choices that have become prevalent in practice. To effectively examine accelerating events in soccer, it is crucial to account for the unique characteristics of each player and adopt a personalized approach to analyzing such events.
To avoid such subjective choices and, more importantly, to obtain a comprehensive understanding of accelerations across the entire velocity spectrum, we aim to quantify the maximum acceleration capability by relying on individual instantaneous values, both of the speed and the acceleration.
In the literature, a common alternative approach used to assess the velocities and accelerations of team sports athletes is the creation of an **Activity Profile** (**AP**), discussed by Sweeting et al., 2017 [6]. This method is based on instantaneous values and deviates from the minimum duration of threshold crossings. During games or training sessions, an athlete's instant speed and acceleration are divided into different zones using predefined threshold values. The AP considers the time and distance covered in each zone or category to evaluate these variables.
Despite offering a comprehensive view of speed and acceleration ranges for different players and overcoming the constraint of minimum threshold duration, there are still challenges associated with the arbitrary definition of global thresholds. The

selection of these thresholds remains a critical issue, and there is no consensus on the optimal values to use.
Furthermore, the speed profile and the acceleration one do not interact with each other, leading to potential inaccuracies in assessing athletes' acceleration capabilities. The same acceleration can be considered low or maximal depending on the speed at which it occurs. Consequently, we may "overestimate" accelerations that reach the threshold but are significantly below the player's maximum capabilities because they occur at low speeds. Conversely, we may "underestimate" significant accelerations that fall below the threshold but bring the player closer to their maximum potential.
The lack of consensus in establishing standard criteria for classifying accelerating efforts in team sports is evident from the variations in speed and acceleration categories used in studies. For instance, some research (Osgnach et al., 2010 [3]) has used categories like walking, jogging, low-speed running, etc., while others (Buchheit et al., 2014 [7] and Couderc et al., 2017 [8]) have employed different categories such as moderate or high. However, the rationale behind these selections often remains undisclosed.
In May 2021, Morin et al. [1] introduced a novel approach to assess the mechanical performance of soccer players' sprints, addressing several limitations discussed earlier. The aim was to provide insights similar to the force-velocity profile of sprints, but without relying on instrumented treadmills or track-embedded multiple force plate systems, which are typically unavailable to most practitioners. Force-velocity and power-velocity profiles are widely regarded as the reference standard for monitoring the overall mechanical capability to generate horizontal external force during sprinting (Samozino et al., 2013 [9]). However, these profiles necessitate individualized tests and specialized equipment for evaluation.
To tackle these challenges, Morin et al. developed the **Acceleration-Speed A-S profile**, a novel methodology grounded in in-situ data collection. This approach allows for data acquisition during regular training sessions or matches, eliminating the need for a specific test. The A-S profile utilizes raw acceleration and speed data obtained during these sessions.
The A-S profile can be utilized to determine a player's maximum acceleration through the use of **linear regression analysis**. This approach offers valuable insights into athletes' sprint performance and enhances our understanding of their mechanical capabilities during acceleration.
To construct the A-S profile following Morin et al.'s instructions, the following steps are undertaken:

1. Setting a minimum speed threshold of 3 $m/s$, as accelerations that occur below this value are not considered maximal;

2. Velocity sub-intervals are defined with a step size of 0.2 $m/s$, starting from 3 $m/s$ and progressing up to the player's maximum velocity achieved;

3. Only positive accelerations are taken into account, and the two highest acceleration values within each velocity sub-interval are identified;

4. A preliminary linear regression analysis is conducted using the two maximum acceleration values for each velocity sub-interval;

5. Subsequently, to enhance the overall accuracy of the model any data points falling outside the 95% confidence interval are excluded;

6. Finally, a second and final linear regression is performed using the refined dataset, devoid of outliers. This regression is utilized to construct the individual in-situ A-S profile.

The proposed model, with appropriate modifications, aims to establish the correlation between a given running speed and the corresponding maximum acceleration produced by a soccer player.
However, we can identify certain limitations of this model in relation to our specific objectives. Specifically, the velocity sub-intervals considered in the original model begin at 3 $m/s$, whereas our intention is to analyze accelerations across the entire velocity spectrum, including the lower initial velocities during a sprint.

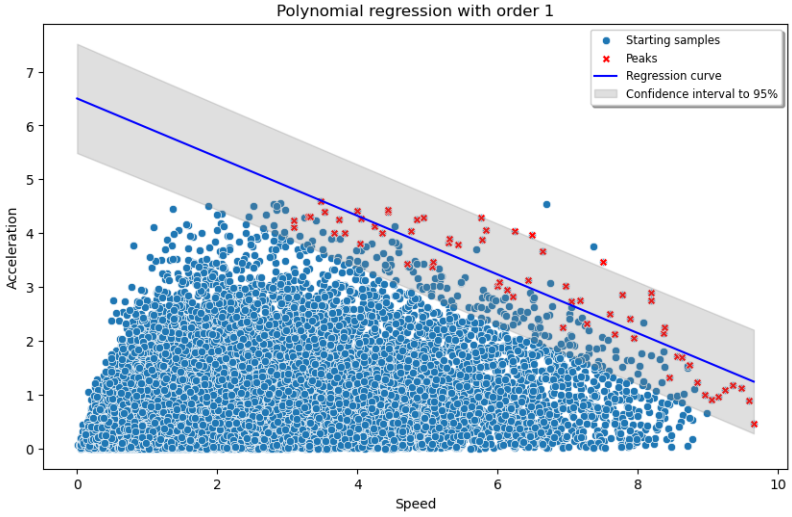



**Figure 2.5.** Rafael Leao: AS profile by Morin et al. [1].

Upon examining the typical arrangement points in the Acceleration-Speed graph in Figure [2.5], it becomes apparent that assuming a first-degree linear relationship across the entire velocity range is not appropriate. The data exhibits a noticeable curvature, indicating the need for a higher-degree polynomial regression model to capture the underlying pattern.
Once we made the decision to encompass the complete velocity range, ranging from 0 $m/s$ to the maximum individual velocity, we also questioned whether limiting the analysis to only two peaks per sub-interval and utilizing a 0.2 $m/s$ amplitude for the sub-intervals would be suitable.

Consequently, three hyperparameters must be determined during the model construction process:

1. The **degree $k$** of the polynomial function that establishes the relationship between velocity and acceleration;
2. The **number of peaks $\nu$** to consider within each speed sub-interval.
3. The **amplitude $\gamma$** of the speed sub-intervals.

The choice of these hyperparameters is critical and should be based on the specific requirements of the study, as they can influence the accuracy and precision of the final model. The subsequent section §2.2.1 will delve into the discussion regarding the selection of these hyperparameters.

Once the values of the last two hyperparameters have been determined, we proceed to identify the data points that need to be fitted by the polynomial regression model, and we calculate the regression coefficients using the least squares method (Sheldon and Ross, 2008 [10] and Hastie et al., 2001 [11]).
Let's consider a generic training dataset of dimension $n$ comprised of ordered pairs ($x$, $y$), where $x$ represents the independent variable (**speed**) and $y$ represents the dependent variable (**acceleration**). Our purpose is to find a polynomial function of degree $k$ that provides the best possible approximation of this data.
This polynomial function can be defined for a generic sample $i$ as:

$$\hat{y}_i(x_i) = \theta_0 + \theta_1 x_i + \theta_2 x_i^2 + \ldots + \theta_k x_i^k \tag{2.10}$$

where the coefficients $\theta_0, \theta_1, \ldots, \theta_k$ are the regressors we want to estimate and $\hat{y}_i$ is the predicted value.

To estimate the regressors, we use the **least squares method**. The objective is to minimize the average squared difference between the values predicted by the

polynomial function and the observed values of the training data, known as the ***Mean Squared Error MSE***:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\|Y - X\theta\|^2$$

where $X$ is a matrix of dimension $(n \times (k+1))$ given by:

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^k \\ 1 & x_2 & x_2^2 & \dots & x_2^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^k \end{bmatrix} \tag{2.11}$$

and $Y$ is a column vector of dimension $(n \times 1)$ given by:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2.12}$$

The Hessian matrix $\nabla^2 MSE(\theta) = \frac{2}{n}X^T X$ is symmetric and positive semidefinite: for any $z \in R^{k+1}$, we have $\langle \frac{2}{n}X^T Xz, z\rangle = \frac{2}{n}\langle Xz, Xz\rangle = \frac{2}{n}\|Xz\|^2 \geq 0$. Therefore, the cost function is a convex function of $\theta$, and the vanishing of the gradient is a necessary and sufficient condition for a global minimum:

$$\nabla MSE(\theta) = -\frac{2}{n}X^T(Y - X\theta) = 0$$

This leads to a system of $k+1$ linear equations in $k+1$ unknowns:

$$X^T X\theta = X^T Y$$

The closed-form solution for estimating the regressors using the *least squares method* is as follows:

$$\hat{\theta} = \begin{bmatrix} \hat{\theta}_0 \\ \hat{\theta}_1 \\ \vdots \\ \hat{\theta}_k \end{bmatrix} = (X^T X)^{-1} X^T Y \tag{2.13}$$

Furthermore, the confidence interval for a specific point $x$ is determined as follows:

$$\hat{y} \pm t_{\alpha/2,n-(k+1)} \cdot \sqrt{\frac{MSE \cdot n}{n-(k+1)} \cdot (1 + x^T (X^T X)^{-1} x)} \tag{2.14}$$

where $t_{\alpha/2,n-(k+1)}$ is the critical t-value for a confidence level $1-\alpha$ with $n-(k+1)$ degrees of freedom.

The computation of this confidence interval is contingent upon the residuals obtained from the regression conforming to a Gaussian distribution.
To validate this assumption, we opted to utilize the ***Anderson-Darling test***, which can be implemented using the Python package *scipy.stats*.

To conclude, it is important that the final parameters obtained, and thus the model, are significant; the test we conducted is ***Student's t-test***.
The null hypothesis $H_0$ assumes that the parameter is equal to zero ($\theta_i = 0$), indicating no relationship between the independent variable and the dependent variable. The alternative hypothesis $H_1$ assumes that the parameter is not equal to zero ($\theta_i \neq 0$), suggesting a significant relationship.

The $t$-statistic for each estimated parameter can be calculated by dividing the absolute value of the estimated parameter by its standard error. Mathematically, it can be expressed as:

$$t_{\hat{\theta}_i} = \frac{|\hat{\theta}_i|}{\sigma_{\hat{\theta}_i}} \tag{2.15}$$

and for a given t-statistic $t_{\hat{\theta}_i}$, the p-value is calculated as:

$$p_{value} = 2 \cdot (1 - CDF(t_{\hat{\theta}_i}, df))$$

where $CDF(t_{\hat{\theta}_i}, df)$ represents the cumulative distribution function of the t-distribution with $df = n - (k+1)$ degrees of freedom.
A significance threshold ($\alpha = 0.05$ in the code) is applied to determine whether the parameters are statistically significant. If the p-value is below the threshold, the null hypothesis is rejected, indicating that the parameter is considered statistically significant. Conversely, if the p-value is above the threshold, the parameter is considered non-significant.

$$\begin{cases} \theta_i & \text{significant,} \quad \text{if } p_{value} \leq 0.05; \\ \theta_i & \text{non significant,} \quad \text{otherwise.} \end{cases} \tag{2.16}$$

Lastly, it is important to note that players who have played for less than 25 minutes have an incomplete set of cloud points, and consequently, they were excluded from the analysis. This exclusion also extends to goalkeepers. It is essential to recognize that the available data may not offer a comprehensive representation of maximal acceleration points across all potential running speeds.

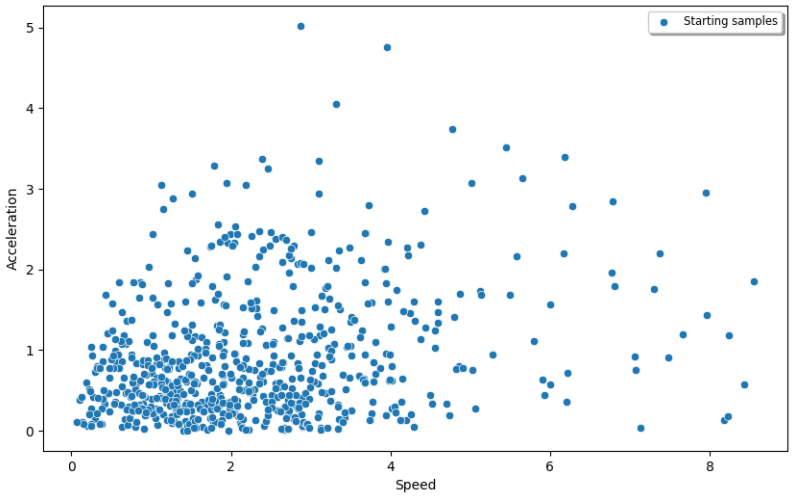



**Figure 2.6.** Juan Jesus: example of a player with low-density data

### 2.2.1 Case study

Once we have defined the model structure, which, as mentioned earlier (§2.2), is governed by three hyperparameters, we proceed with the selection of the model itself by implementing a ***grid search*** to find the optimal values of the hyperparameters themselves.
*Grid search* is an iterative procedure used for tuning hyperparameters. It involves running the model multiple times, with different combinations of hyperparameter values. In our case, the hyperparameters to be set are as follows:

- Degree of the polynomial **k**: [2, 3, 4, 5];
- Number of peaks $\boldsymbol{\nu}$: [2, 3, 4];
- Amplitude of the speed interval $\boldsymbol{\gamma}$: [0.1, 0.15, 0.2].

The choice of parameter values for the grid search is a critical aspect, as different combinations of small amplitude and a high number of peaks would result in considering a larger number of points for regression. Since our goal is to focus only on the maximal values, we deem it appropriate to consider these two sets of parameters because:

1. In the case of a higher number of points ($\nu = 4$ and $\gamma = 0.1$), we have approximately 360 points (considering that the highest speed is typically around 9 $m/s$).
2. Conversely, in the case of $\gamma = 0.2$ and $\nu = 2$, we initially consider approximately 90 points.

By carefully selecting these parameter sets, we can control the number of data points used for regression while still capturing the maximal values, ensuring that the model focuses on the most relevant aspects of the data.

However, our approach extends beyond a simple grid search. We also incorporate ***K-Fold Cross Validation*** to obtain a more robust evaluation of the model's performance. *K-Fold Cross Validation* is a technique commonly used to assess the effectiveness of machine learning models. In this technique, the dataset is divided into $K$ subsets of similar size, known as **folds**. The model is then trained and evaluated $K$ times, with each iteration using a different fold as a surrogate for the test set, referred to as the **validation set**, while the remaining folds serve as the training set. This process ensures that each data point is utilized both for training and evaluation, reducing the risk of overfitting and providing a more reliable

estimate of the model's performance.
Since a significant amount of information is required for this procedure, we decided to validate our model by merging the data of all three matches.
This approach allows us to use a larger dataset for training and validation, increasing the reliability and robustness of the results. By pooling data from several matches, there is a wider range of patterns and variations in data, leading to a more comprehensive evaluation of the model's performance.
Through the aggregation of results from the $K$ iterations, we achieve a comprehensive assessment of the model's performance, taking into account its consistency across multiple validation sets. This methodology empowers us to make more informed decisions regarding the model's suitability and its ability to generalize to unseen data. We selected a value of $K$ equal to 5 for the folds, which is a commonly used value in practice. This implies that in each iteration of the grid search, a model is trained using a specific combination of hyperparameters on five distinct training sets. The model's performance is then evaluated by calculating the average validation error across these five iterations.
The validation error is measured using the **Mean Squared Error**, defined as:

$$MSE_{VS} = \frac{1}{n_{VS}} \sum_{i=1}^{n_{VS}} (y_i - \hat{y}_i)^2 \tag{2.17}$$

where $n_{VS}$ represents the number of data points in the validation set $VS$, $y_i$ is the observed output for the $i$-th data point, and $\hat{y}_i$ is the predicted value obtained from the model.
Below, therefore, are the results obtained in terms of *validation error*:

| | | $\gamma$ | | |
|---|---|---|---|---|
| $k$ | $\nu$ | **0,1** | **0,15** | **0,2** |
| **2** | **2** | 2,910286 | 2,533932 | 2,538972 |
| | **3** | 2,964856 | 2,723904 | 2,638455 |
| | **4** | 2,905484 | 2,816248 | 2,640348 |
| **3** | **2** | 2,799436 | 2,348244 | 2,361138 |
| | **3** | 2,750433 | 2,578432 | 2,427209 |
| | **4** | 2,602653 | 2,626384 | 2,537845 |
| **4** | **2** | 2,867327 | 2,494879 | 2,564304 |
| | **3** | 2,745394 | 2,664595 | 2,521122 |
| | **4** | 2,628061 | 2,678039 | 2,598541 |
| **5** | **2** | 2,828461 | 2,472953 | 2,517025 |
| | **3** | 2,793082 | 2,602684 | 2,529528 |
| | **4** | 2,626727 | 2,652741 | 2,57793 |

Analyzing the table above, we can draw initial conclusions.
By selecting the model with the lowest validation error for each degree, we find the following models with the highest fitting quality:

1. $M_1(k = 2, \nu = 2, \gamma = 0.15)$
2. $M_2(k = 3, \nu = 2, \gamma = 0.15)$
3. $M_3(k = 4, \nu = 2, \gamma = 0.15)$
4. $M_4(k = 5, \nu = 2, \gamma = 0.15)$

For each degree, the combination of hyperparameters $\nu = 2$ and $\gamma = 0.15$ yields the best results. However, it is important to note that only $M_1$ and $M_2$ have all parameters found to be statistically significant, which directs our attention to these two models.
Our choice ultimately falls on $M_1$ due to several reasons.
Firstly, it has a lower number of degrees of freedom compared to $M_2$. By selecting a second-degree function, we limit the potential for multiple changes in concavity, which can occur with a third-degree function. This becomes particularly relevant when dealing with data points that exhibit low density throughout the speed and acceleration spectrum, which may occur when analyzing an individual player's data without combining information from all three matches.
Secondly, our decision aligns with the principle of Occam's razor, which favors simplicity among equally valid options. Both $M_1$ and $M_2$ exhibit an R-squared value exceeding 0.8 (0.877 for the former and 0.891 for the latter), indicating a good level of fit. However, by choosing the simpler model, $M_1$, we avoid unnecessary complexity without significantly sacrificing performance.
Lastly, examining the plotted data reveals a clear parabolic trend in the maxima, further supporting the appropriateness of a second-degree function.
Considering that the increase in validation error between $M_1$ and $M_2$ is only 7.9%, and based on the aforementioned considerations, we prefer the more parsimonious model, $M_1$.

For the sake of completeness, we provide the values of both the ***AIC*** (Akaike Information Criterion) and the ***BIC*** (Bayesian Information Criterion).
The AIC is defined as:

$$AIC = 2(k + 1) - 2L \tag{2.18}$$

The BIC is defined as:

$$BIC = (k+1)\ln(n) - 2L \tag{2.19}$$

where $L$ is the log-likelihood of the model. Both criteria were calculated using all the data from all players in the three matches but outside of the K-fold cross-validation. Below is the table showing the AIC and BIC values for the two models:

| | ***AIC*** | ***BIC*** |
|---|---|---|
| $\mathbf{M_1}$ | 594.9724 | 1435.891 |
| $\mathbf{M_2}$ | 584.2621 | 1410.871 |

**Table 2.2.** AIC and BIC values for $M_1$ and $M_2$

Upon examination, we observe a small difference of 1.8% in the AIC values and 1.7% in the BIC values between the two models. This indicates a negligible difference between the two models in terms of information loss and complexity penalty.

Graphically, what we obtain by exploiting these hyperparameters and reconstructing the maximum acceleration profile, taking into account all the players within our dataset, is this:

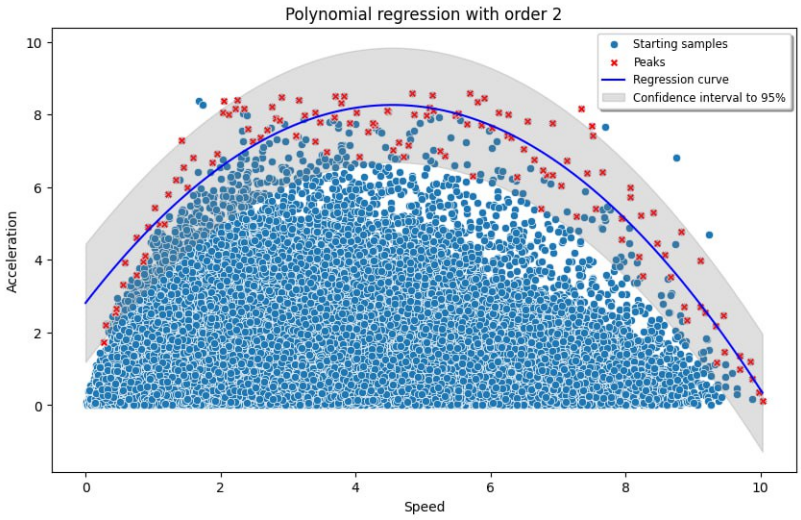



**Figure 2.7.** Acceleration profile on all players

and it is described by the following formula:

$$\hat{a} = \hat{\theta}_0 + \hat{\theta}_1 \cdot s + \hat{\theta}_2 \cdot s^2 \tag{2.20}$$

where $\hat{a}$ represents the estimated maximum acceleration at a given speed $s$. Obviously, once the hyperparameters have been determined, the model can be applied to the data of each individual player to define his acceleration profile.

## 2.3 Stamina trend estimation

After we have determined how and which hyperparameters to use for constructing a player's acceleration profile, we can take our analysis a step further by sampling the profile at different moments during a game.
It is highly likely that the parabolic shape of the profile will undergo changes, either becoming wider or flatter, indicating variations in the player's acceleration pattern. But how can we interpret these changes? One way is by focusing on the vertex of the parabola.
Specifically, we can examine the y-coordinate of the vertex, which corresponds to the maximum acceleration value $a_{max}$:

$$\hat{a} = \hat{\theta}_0 + \hat{\theta}_1 \cdot s + \hat{\theta}_2 \cdot s^2 \tag{2.21}$$

$$\hat{a}_{max} = -\frac{\hat{\theta}_1^2 - 4\hat{\theta}_0\hat{\theta}_2}{4\hat{\theta}_2} \tag{2.22}$$

identifies the player's maximum acceleration, a downward shift of the parabola results in a decline in the player's physical condition, the **stamina**, under analysis.

### 2.3.1 Case study

Upon attempting to apply the analysis mentioned above to our case study, we encounter a limitation: the available data for each player is insufficient to make accurate estimations. As a result, we need to resort to approximating the average stamina decay by considering all the players from the merged data of the three matches at our disposal.
To address the lack of exhaustive data, we implemented a global acceleration profile by sampling the data at each quarter of the game, which corresponds to intervals of approximately 22 minutes and a half. This approach allows us to capture the overall trend of stamina decay throughout the matches.
Here are the results obtained from our analysis:

| Minute | Maximum Acceleration | Stamina |
|---|---|---|
| 22:30 | 7.39 $m/s^2$ | $\sim$ 100 % |
| 45:00 | 7.20 $m/s^2$ | 95,51 % |
| 67:30 | 6.89 $m/s^2$ | 93,23 % |
| 90:00 | 6.42 $m/s^2$ | 86,88 % |

**Table 2.3.** Stamina trend both in absolute and in percentage values

The values included in the table above derive from an analysis of the various curves obtained at each quarter.

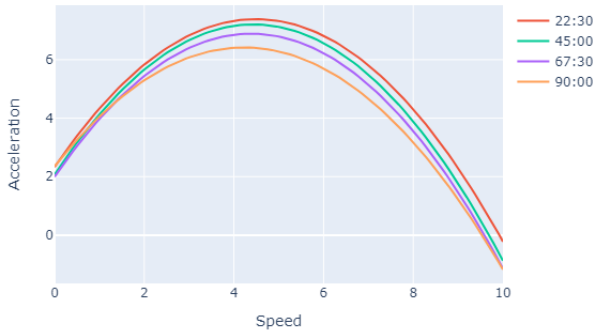



**Figure 2.8.** Acceleration profile considering the merged data from all three matches in the different four quarters

As we can see from the image in Figure [2.8] and the Table [2.3], there is a clear drop in the maximum sustainable acceleration, which coincides with the upper peak of the curve describing the acceleration profile.
Once we have collected these samples, our goal is to estimate a function that describes the evolution of stamina throughout the entire game, aiming to provide valuable and practical insights. To accomplish this, we can employ either a first-degree or second-degree polynomial regression. We have opted for the latter choice as it allows us to capture a short initial interval where stamina remains relatively constant, typically observed in the first few minutes of the game.
Specifically, the independent variable is the maximum acceleration capability defined in its percentage form, while the dependent variable is the playing time expressed in seconds.

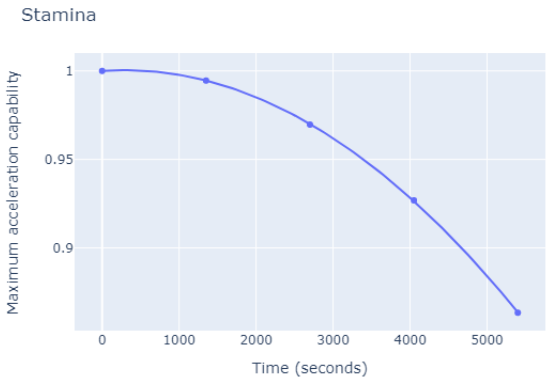



**Figure 2.9.** Stamina percentage trend

## 2.4 Sprint simulation

Once we have determined the hyperparameters to be used within the multiple regression in §2.2.1, we can proceed to construct and identify the **acceleration profile** of each player within our database, thus analyzing the relationship between instantaneous velocity and instantaneous acceleration.
In particular, the hyperparameters we refer to, and which were defined thanks to the previously mentioned 5-fold cross-validation, are as follows:

- Degree $\boldsymbol{k}$ of the polynomial function, it is 2;
- Number $\boldsymbol{\nu}$ of peaks, equal to 2;
- Speed sampling rate $\boldsymbol{\gamma}$, it is equal to 0.15 m/s.

By then applying the model to each i-th player, we obtain many different second-degree polynomial curves:

$$\hat{a_i} = \hat{\theta}_{0i} + \hat{\theta}_{1i} \cdot s_i + \hat{\theta}_{2i} \cdot s_i^2 \tag{2.23}$$

where the vector $\hat{\theta}_i = [\hat{\theta}_{0i}, \hat{\theta}_{1i}, \hat{\theta}_{2i}]$ is unique for each player and determines its acceleration profile, $\hat{a}$ represents the estimated maximum acceleration at a given speed $s$.
Graphically, this (i.e. Rafael Leao's acceleration profile) can be represented as follows:

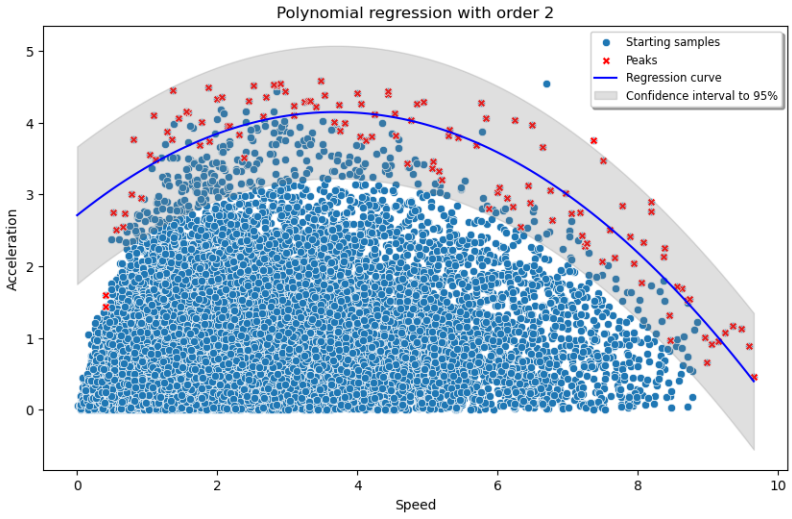
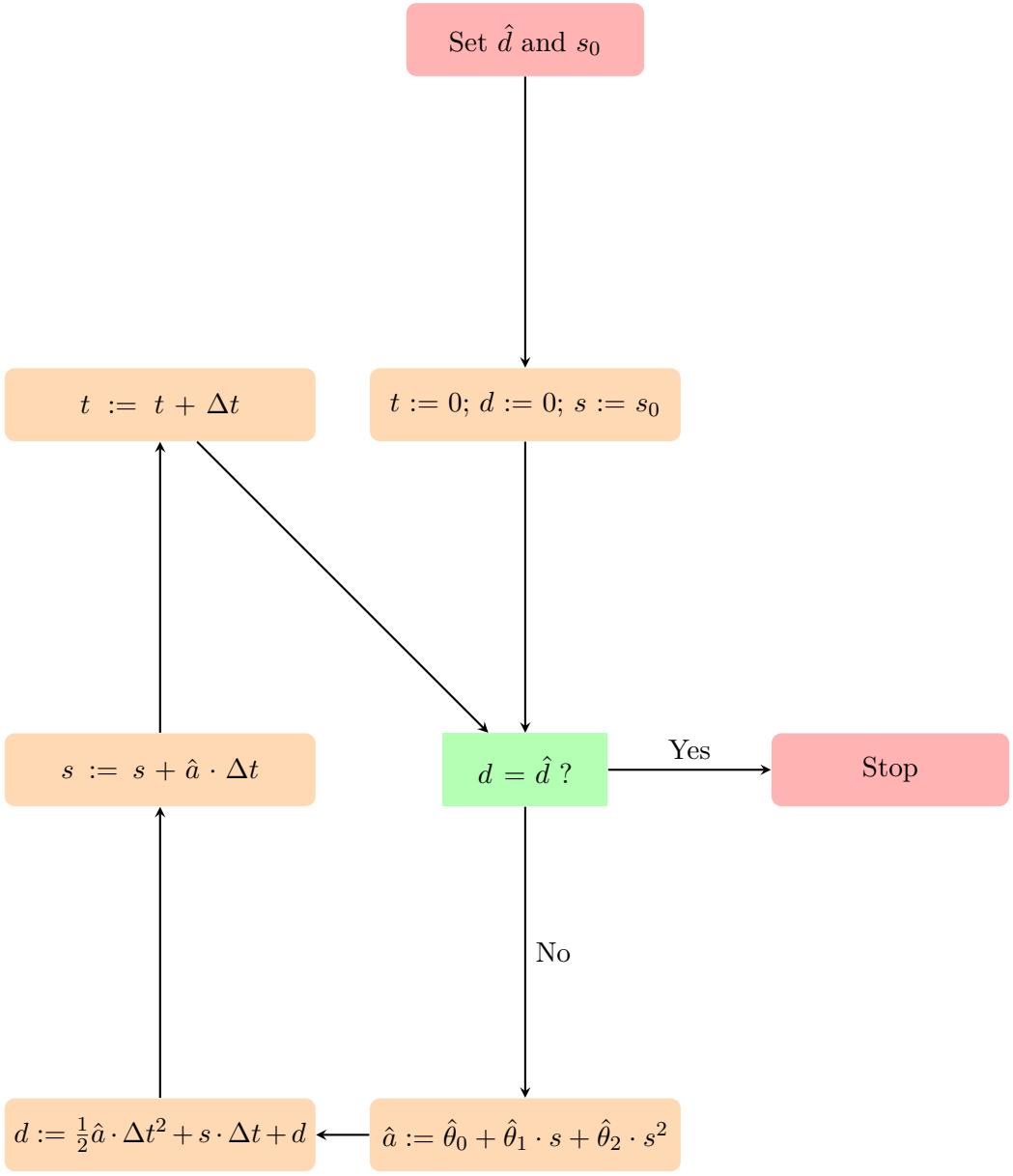



**Figure 2.10.** Acceleration profile of a single player

After acquiring the acceleration profiles of all the players at our disposal, we can move on to the sprint simulation phase, considering both long and short distances and both stationary and moving starts; these simulations are carried out using a simple iterative algorithm, the flow-chart of which is presented below:




where $\Delta t$ is the sampling rate, $d$ the runned distance and $\hat{d}$ the target one set by ourselves.
The algorithm operates in a straightforward manner: it calculates the maximum acceleration for a given speed at a sampling rate of 0.1 seconds using the player's maximum acceleration profile until the target distance is reached. From the maximum acceleration value, the subsequent velocity and the distance traveled at each timestamp

are calculated using the equations of **uniformly accelerated motion**. This allows for the estimation of the player's motion and progression throughout the sprint simulation.
Although the selection of distances and starting speeds for the simulations is subjective, we have opted for a set of six distinct distances, including three short and three long distances. Additionally, we have chosen four different starting speeds to account for variations in initial velocity.
Specifically, the simulations within our work involve the following distances:

| $\hat{\mathbf{d}}$ | 10 | 20 | 35 | 50 | 75 | 100 |
|---|---|---|---|---|---|---|

**Table 2.4.** Distances for sprint simulation expressed in meters

and the following starting speeds:

| $\mathbf{s_0}$ | 0 | 2.5 | 5.5 | 7 |
|---|---|---|---|---|

**Table 2.5.** Staring speed for sprint simulation expressed in $m/s$

The implementation of this algorithm enables the simulation and estimation of running times for different football players under various conditions, while also providing valuable features for the final phase of our work. In this phase, we aim to identify groups of footballers based on their athletic performance using clustering techniques.
It is worth noting that this procedure can be adjusted to incorporate the estimated stamina percentage obtained in the previous section. To do so, we simply multiply the acceleration value by the estimated physical condition percentage. However, it is important to acknowledge that this modification may result in inferior outcomes and longer running times, as the reduced stamina affects the player's performance.

### 2.4.1 Case study

After identifying the parameters that describe the acceleration profiles of the players in our database, we proceed to implement the recursive algorithm for sprint simulations. These simulations provide insights into the patterns of acceleration and speed over time.
To illustrate this, let's consider an example of a sprint simulation for Rafael Leao over a distance of 30 meters, starting from a standing position.

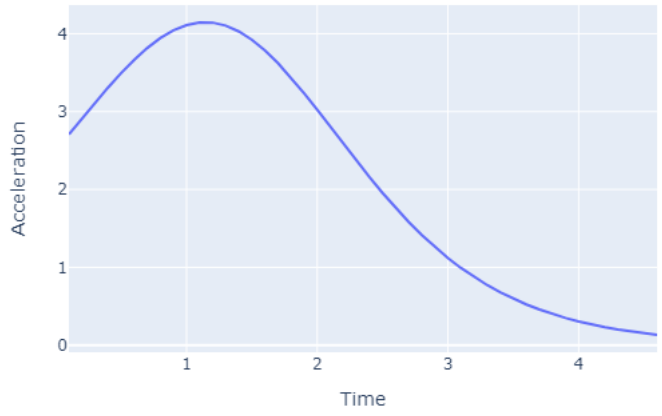



**Figure 2.11.** Example of acceleration trend on time in a sprint simulation

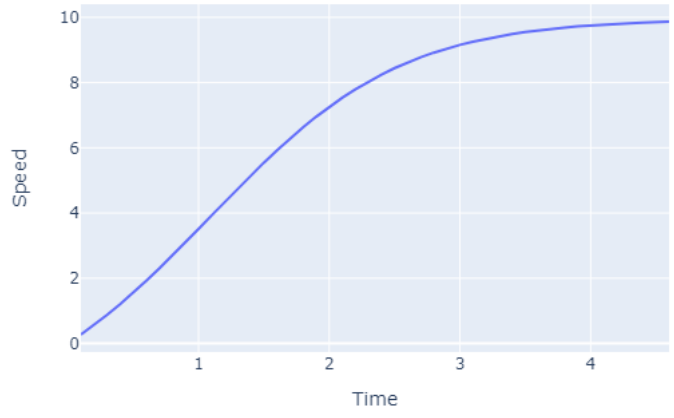



**Figure 2.12.** Example of speed trend on time in a sprint simulation

As shown in Figure [2.11], the simulated sprint closely follows the expected acceleration profile. It exhibits a gradual increase in both acceleration and speed initially, as illustrated in Figure [2.12]. However, once the maximum acceleration is reached, acceleration starts to decrease while speed remains constant at its maximum value until the end of the sprint. This behavior is consistent with the velocity profile resembling an inverse exponential function, similar to the one utilized in the study by Morin et al., 2019 [12]. The equation for this function is $v(t) = v_{\max} \cdot (1 - e^{-t/\tau})$, where $v(t)$ represents the velocity at time $t$, $v_{\max}$ denotes the maximum velocity, and $\tau$ represents the time constant. It is worth noting that the use of exponential models in analyzing sprint running velocity over time dates back almost a century, as initially presented by Furusawa et al., 1927 [13].

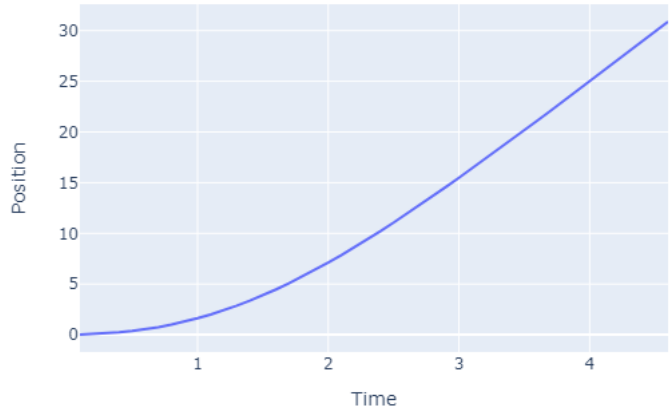



**Figure 2.13.** Example of position trend on time in a sprint simulation

In Figure [2.13], we observe the displacement of the player being examined relative to the starting point. The initial growth of the displacement is faster than that of a parabolic curve, indicating an increase in acceleration. However, once the acceleration decrease, the displacement becomes nearly linear, suggesting a constant speed maintained by the player.

Now, after having made a brief digression about the acceleration and velocity trends during these simulations, we report below the results obtained from the simulations carried out on all the players in our case study.

| d | s_0 | Saelemaekers | Diaz | Calabria | Tomori | Bennacer | Messias | Giroud | Kalulu | Leao | Dest | Hernandez | Pobega | Blin | Gallo | Baschirotto |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 0.0 | 2.6 | 2.6 | 2.6 | 2.3 | 2.5 | 2.4 | 2.5 | 2.4 | 2.4 | 2.7 | 2.4 | 2.6 | 2.8 | 3.0 | 2.6 |
| | 2.5 | 1.8 | 1.8 | 1.8 | 1.7 | 1.8 | 1.7 | 1.8 | 1.7 | 1.8 | 1.8 | 1.7 | 1.8 | 1.9 | 1.9 | 1.8 |
| | 5.5 | 1.4 | 1.4 | 1.5 | 1.4 | 1.4 | 1.4 | 1.4 | 1.4 | 1.4 | 1.4 | 1.4 | 1.4 | 1.5 | 1.5 | 1.4 |
| | 7.0 | 1.3 | 1.3 | 1.3 | 1.3 | 1.3 | 1.2 | 1.3 | 1.3 | 1.2 | 1.3 | 1.2 | 1.3 | 1.4 | 1.3 | 1.3 |
| 20 | 0.0 | 3.9 | 3.8 | 3.8 | 3.5 | 3.7 | 3.5 | 3.7 | 3.5 | 3.5 | 3.9 | 3.5 | 3.9 | 4.1 | 4.3 | 3.9 |
| | 2.5 | 3.0 | 3.0 | 3.1 | 2.8 | 3.0 | 2.8 | 2.9 | 2.8 | 2.9 | 3.0 | 2.8 | 3.0 | 3.2 | 3.1 | 3.0 |
| | 5.5 | 2.6 | 2.6 | 2.7 | 2.5 | 2.6 | 2.4 | 2.5 | 2.5 | 2.4 | 2.6 | 2.4 | 2.7 | 2.8 | 2.7 | 2.6 |
| | 7.0 | 2.5 | 2.5 | 2.6 | 2.4 | 2.5 | 2.3 | 2.4 | 2.4 | 2.3 | 2.4 | 2.3 | 2.5 | 2.7 | 2.5 | 2.5 |
| 35 | 0.0 | 5.6 | 5.5 | 5.7 | 5.1 | 5.5 | 5.1 | 5.3 | 5.2 | 5.1 | 5.6 | 5.0 | 5.7 | 6.0 | 6.1 | 5.6 |
| | 2.5 | 4.8 | 4.8 | 4.9 | 4.5 | 4.8 | 4.4 | 4.6 | 4.5 | 4.4 | 4.7 | 4.4 | 4.8 | 5.1 | 4.9 | 4.8 |
| | 5.5 | 4.3 | 4.3 | 4.5 | 4.1 | 4.4 | 4.0 | 4.2 | 4.2 | 4.0 | 4.3 | 4.0 | 4.5 | 4.7 | 4.5 | 4.4 |
| | 7.0 | 4.2 | 4.2 | 4.4 | 4.0 | 4.2 | 3.9 | 4.1 | 4.0 | 3.8 | 4.1 | 3.8 | 4.4 | 4.6 | 4.3 | 4.2 |
| 50 | 0.0 | 7.3 | 7.2 | 7.5 | 6.8 | 7.2 | 6.7 | 7.0 | 6.9 | 6.6 | 7.3 | 6.5 | 7.5 | 7.9 | 7.9 | 7.3 |
| | 2.5 | 6.5 | 6.5 | 6.7 | 6.1 | 6.5 | 6.0 | 6.2 | 6.2 | 5.9 | 6.4 | 5.9 | 6.6 | 7.0 | 6.7 | 6.5 |
| | 5.5 | 6.1 | 6.0 | 6.4 | 5.8 | 6.1 | 5.6 | 5.8 | 5.8 | 5.5 | 6.0 | 5.5 | 6.3 | 6.6 | 6.2 | 6.1 |
| | 7.0 | 5.9 | 5.9 | 6.2 | 5.7 | 6.0 | 5.5 | 5.7 | 5.7 | 5.3 | 5.8 | 5.3 | 6.2 | 6.5 | 6.1 | 6.0 |
| 75 | 0.0 | 10.2 | 10.1 | 10.6 | 9.5 | 10.1 | 9.3 | 9.7 | 9.6 | 9.1 | 10.1 | 9.1 | 10.5 | 11.1 | 10.8 | 10.2 |
| | 2.5 | 9.4 | 9.3 | 9.8 | 8.8 | 9.4 | 8.6 | 9.0 | 8.9 | 8.4 | 9.2 | 8.4 | 9.7 | 10.2 | 9.7 | 9.4 |
| | 5.5 | 9.0 | 8.9 | 9.4 | 8.5 | 9.0 | 8.2 | 8.6 | 8.6 | 8.0 | 8.8 | 8.0 | 9.3 | 9.8 | 9.2 | 9.0 |
| | 7.0 | 8.8 | 8.7 | 9.3 | 8.4 | 8.9 | 8.1 | 8.5 | 8.5 | 7.8 | 8.6 | 7.9 | 9.2 | 9.6 | 9.1 | 8.9 |
| 100 | 0.0 | 13.1 | 12.9 | 13.6 | 12.3 | 13.0 | 11.9 | 12.5 | 12.4 | 11.6 | 12.9 | 11.6 | 13.5 | 14.2 | 13.8 | 13.2 |
| | 2.5 | 12.3 | 12.2 | 12.8 | 11.6 | 12.3 | 11.2 | 11.8 | 11.7 | 10.9 | 12.0 | 11.0 | 12.7 | 13.4 | 12.6 | 12.3 |
| | 5.5 | 11.8 | 11.7 | 12.5 | 11.3 | 11.9 | 10.8 | 11.4 | 11.3 | 10.5 | 11.6 | 10.6 | 12.3 | 12.9 | 12.2 | 11.9 |
| | 7.0 | 11.7 | 11.6 | 12.3 | 11.2 | 11.8 | 10.7 | 11.2 | 11.2 | 10.3 | 11.5 | 10.4 | 12.2 | 12.8 | 12.1 | 11.8 |

**Figure 2.14.** Results of the sprint simulation (1 of 4)

| d | s_0 | Di Francesco | Strefezza | Pezzella | Gonzàlez | Colombo | Due Hjulmand | Umtiti | Gendrey | Maleh | Krunic | Kim | Simeone | Lobotka | Zielinski | Politano |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 0.0 | 2.7 | 2.8 | 2.8 | 2.6 | 2.7 | 2.7 | 2.7 | 2.6 | 2.8 | 2.5 | 2.6 | 2.7 | 2.7 | 2.7 | 2.7 |
| | 2.5 | 1.8 | 1.9 | 1.9 | 1.8 | 1.8 | 1.8 | 1.9 | 1.8 | 1.9 | 1.8 | 1.8 | 1.9 | 1.9 | 1.9 | 1.9 |
| | 5.5 | 1.4 | 1.5 | 1.4 | 1.5 | 1.4 | 1.5 | 1.5 | 1.4 | 1.5 | 1.4 | 1.4 | 1.4 | 1.5 | 1.5 | 1.5 |
| | 7.0 | 1.3 | 1.4 | 1.3 | 1.3 | 1.3 | 1.4 | 1.4 | 1.3 | 1.4 | 1.3 | 1.3 | 1.3 | 1.4 | 1.4 | 1.3 |
| 20 | 0.0 | 3.9 | 4.1 | 4.0 | 3.9 | 3.8 | 4.0 | 4.1 | 3.7 | 4.1 | 3.7 | 3.8 | 3.9 | 4.0 | 4.1 | 3.9 |
| | 2.5 | 3.0 | 3.2 | 3.0 | 3.1 | 2.9 | 3.1 | 3.2 | 3.0 | 3.2 | 3.0 | 3.0 | 3.1 | 3.2 | 3.2 | 3.1 |
| | 5.5 | 2.6 | 2.7 | 2.5 | 2.7 | 2.5 | 2.7 | 2.9 | 2.5 | 2.7 | 2.6 | 2.6 | 2.6 | 2.8 | 2.8 | 2.6 |
| | 7.0 | 2.5 | 2.6 | 2.4 | 2.6 | 2.3 | 2.6 | 2.8 | 2.4 | 2.6 | 2.4 | 2.4 | 2.5 | 2.7 | 2.7 | 2.5 |
| 35 | 0.0 | 5.7 | 5.9 | 5.6 | 5.7 | 5.4 | 5.8 | 6.1 | 5.4 | 5.9 | 5.4 | 5.5 | 5.6 | 5.9 | 6.0 | 5.6 |
| | 2.5 | 4.7 | 5.0 | 4.6 | 4.9 | 4.5 | 5.0 | 5.3 | 4.6 | 5.0 | 4.7 | 4.7 | 4.8 | 5.1 | 5.2 | 4.8 |
| | 5.5 | 4.3 | 4.6 | 4.1 | 4.5 | 4.0 | 4.6 | 4.9 | 4.1 | 4.6 | 4.3 | 4.3 | 4.3 | 4.7 | 4.8 | 4.3 |
| | 7.0 | 4.2 | 4.4 | 4.0 | 4.4 | 3.9 | 4.4 | 4.8 | 3.9 | 4.4 | 4.1 | 4.1 | 4.1 | 4.6 | 4.6 | 4.1 |
| 50 | 0.0 | 7.4 | 7.8 | 7.2 | 7.5 | 6.9 | 7.7 | 8.2 | 6.9 | 7.7 | 7.1 | 7.2 | 7.3 | 7.9 | 8.0 | 7.3 |
| | 2.5 | 6.5 | 6.8 | 6.2 | 6.7 | 6.0 | 6.8 | 7.3 | 6.1 | 6.8 | 6.4 | 6.4 | 6.5 | 7.0 | 7.1 | 6.5 |
| | 5.5 | 6.0 | 6.4 | 5.7 | 6.3 | 5.6 | 6.4 | 7.0 | 5.7 | 6.4 | 6.0 | 6.0 | 6.0 | 6.6 | 6.7 | 6.0 |
| | 7.0 | 5.9 | 6.2 | 5.6 | 6.2 | 5.5 | 6.3 | 6.9 | 5.5 | 6.2 | 5.8 | 5.8 | 5.8 | 6.5 | 6.6 | 5.8 |
| 75 | 0.0 | 10.2 | 10.8 | 9.9 | 10.5 | 9.6 | 10.8 | 11.6 | 9.6 | 10.8 | 9.9 | 10.0 | 10.1 | 11.0 | 11.2 | 10.1 |
| | 2.5 | 9.3 | 9.9 | 8.9 | 9.8 | 8.7 | 9.9 | 10.7 | 8.8 | 9.9 | 9.2 | 9.2 | 9.3 | 10.2 | 10.4 | 9.3 |
| | 5.5 | 8.9 | 9.4 | 8.4 | 9.4 | 8.2 | 9.5 | 10.4 | 8.3 | 9.4 | 8.8 | 8.8 | 8.8 | 9.8 | 9.9 | 8.8 |
| | 7.0 | 8.7 | 9.3 | 8.3 | 9.2 | 8.1 | 9.4 | 10.2 | 8.2 | 9.3 | 8.6 | 8.6 | 8.6 | 9.7 | 9.8 | 8.6 |
| 100 | 0.0 | 13.1 | 13.8 | 12.5 | 13.6 | 12.2 | 13.9 | 14.9 | 12.2 | 13.8 | 12.7 | 12.8 | 13.0 | 14.2 | 14.4 | 12.9 |
| | 2.5 | 12.1 | 12.9 | 11.6 | 12.8 | 11.3 | 13.0 | 14.1 | 11.4 | 12.9 | 12.0 | 12.0 | 12.1 | 13.4 | 13.6 | 12.1 |
| | 5.5 | 11.7 | 12.5 | 11.1 | 12.4 | 10.9 | 12.6 | 13.7 | 10.9 | 12.5 | 11.6 | 11.6 | 11.6 | 13.0 | 13.2 | 11.6 |
| | 7.0 | 11.6 | 12.3 | 10.9 | 12.3 | 10.7 | 12.5 | 13.6 | 10.8 | 12.3 | 11.5 | 11.5 | 11.5 | 12.9 | 13.0 | 11.4 |

**Figure 2.15.** Results of the sprint simulation (2 of 4)

| d | s_0 | Tonali | Kjaer | Rui | Kvaratskhelia | Di Lorenzo | Anguissa | Rrahmani | Koopmeiners | Scalvini | Zappacosta | Lookman | Edérson | Boga | Toloi | Hojlund |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **10** | **0.0** | 2.4 | 2.7 | 2.6 | 2.7 | 2.7 | 2.6 | 2.8 | 2.6 | 2.5 | 2.4 | 2.6 | 2.7 | 2.8 | 2.4 | 2.4 |
| | **2.5** | 1.8 | 1.9 | 1.9 | 1.9 | 1.9 | 1.9 | 1.9 | 1.9 | 1.8 | 1.8 | 1.8 | 1.9 | 1.9 | 1.7 | 1.8 |
| | **5.5** | 1.4 | 1.5 | 1.5 | 1.5 | 1.4 | 1.5 | 1.5 | 1.5 | 1.5 | 1.4 | 1.4 | 1.5 | 1.5 | 1.4 | 1.4 |
| | **7.0** | 1.3 | 1.4 | 1.3 | 1.3 | 1.3 | 1.3 | 1.4 | 1.4 | 1.4 | 1.3 | 1.3 | 1.3 | 1.3 | 1.3 | 1.3 |
| **20** | **0.0** | 3.6 | 4.0 | 3.8 | 4.0 | 3.9 | 3.9 | 4.1 | 3.9 | 3.8 | 3.6 | 3.8 | 3.9 | 4.1 | 3.5 | 3.6 |
| | **2.5** | 2.9 | 3.2 | 3.1 | 3.1 | 3.0 | 3.1 | 3.2 | 3.2 | 3.1 | 3.0 | 3.0 | 3.1 | 3.2 | 2.8 | 2.9 |
| | **5.5** | 2.6 | 2.7 | 2.6 | 2.7 | 2.5 | 2.7 | 2.8 | 2.7 | 2.7 | 2.6 | 2.6 | 2.6 | 2.7 | 2.5 | 2.5 |
| | **7.0** | 2.5 | 2.6 | 2.5 | 2.5 | 2.3 | 2.5 | 2.7 | 2.6 | 2.6 | 2.4 | 2.5 | 2.5 | 2.6 | 2.3 | 2.4 |
| **35** | **0.0** | 5.3 | 5.9 | 5.5 | 5.8 | 5.4 | 5.6 | 6.0 | 5.7 | 5.6 | 5.3 | 5.6 | 5.6 | 5.9 | 5.2 | 5.2 |
| | **2.5** | 4.7 | 5.0 | 4.8 | 4.9 | 4.5 | 4.9 | 5.1 | 5.0 | 4.9 | 4.6 | 4.8 | 4.8 | 5.0 | 4.4 | 4.5 |
| | **5.5** | 4.3 | 4.6 | 4.3 | 4.5 | 3.9 | 4.4 | 4.7 | 4.6 | 4.6 | 4.2 | 4.5 | 4.4 | 4.5 | 4.1 | 4.1 |
| | **7.0** | 4.2 | 4.5 | 4.2 | 4.3 | 3.7 | 4.3 | 4.6 | 4.5 | 4.4 | 4.1 | 4.3 | 4.2 | 4.4 | 4.0 | 4.0 |
| **50** | **0.0** | 7.0 | 7.7 | 7.2 | 7.6 | 6.8 | 7.4 | 8.0 | 7.6 | 7.5 | 6.9 | 7.4 | 7.4 | 7.7 | 6.8 | 6.8 |
| | **2.5** | 6.4 | 6.9 | 6.5 | 6.7 | 5.9 | 6.6 | 7.1 | 6.9 | 6.8 | 6.3 | 6.6 | 6.6 | 6.8 | 6.1 | 6.2 |
| | **5.5** | 6.0 | 6.4 | 6.0 | 6.2 | 5.4 | 6.2 | 6.7 | 6.4 | 6.4 | 5.8 | 6.3 | 6.1 | 6.3 | 5.7 | 5.8 |
| | **7.0** | 5.9 | 6.3 | 5.9 | 6.1 | 5.2 | 6.0 | 6.6 | 6.3 | 6.3 | 5.7 | 6.1 | 5.9 | 6.1 | 5.6 | 5.6 |
| **75** | **0.0** | 9.9 | 10.8 | 10.1 | 10.5 | 9.2 | 10.3 | 11.2 | 10.7 | 10.5 | 9.7 | 10.4 | 10.2 | 10.7 | 9.5 | 9.5 |
| | **2.5** | 9.3 | 10.0 | 9.3 | 9.7 | 8.3 | 9.5 | 10.3 | 9.9 | 9.8 | 9.0 | 9.6 | 9.4 | 9.8 | 8.8 | 8.9 |
| | **5.5** | 8.9 | 9.5 | 8.9 | 9.2 | 7.7 | 9.1 | 9.9 | 9.5 | 9.5 | 8.6 | 9.3 | 8.9 | 9.3 | 8.4 | 8.5 |
| | **7.0** | 8.8 | 9.4 | 8.7 | 9.1 | 7.5 | 8.9 | 9.8 | 9.4 | 9.3 | 8.4 | 9.1 | 8.8 | 9.1 | 8.3 | 8.4 |
| **100** | **0.0** | 12.8 | 13.9 | 12.9 | 13.5 | 11.6 | 13.2 | 14.4 | 13.7 | 13.6 | 12.4 | 13.4 | 13.1 | 13.7 | 12.2 | 12.3 |
| | **2.5** | 12.2 | 13.0 | 12.1 | 12.6 | 10.7 | 12.5 | 13.5 | 13.0 | 12.9 | 11.7 | 12.6 | 12.3 | 12.7 | 11.5 | 11.6 |
| | **5.5** | 11.8 | 12.6 | 11.7 | 12.2 | 10.1 | 12.0 | 13.2 | 12.6 | 12.5 | 11.3 | 12.3 | 11.8 | 12.3 | 11.1 | 11.2 |
| | **7.0** | 11.7 | 12.5 | 11.5 | 12.0 | 9.9 | 11.8 | 13.1 | 12.5 | 12.4 | 11.2 | 12.1 | 11.6 | 12.1 | 11.0 | 11.1 |

**Figure 2.16.** Results of the sprint simulation (3 of 4)

| d | s_0 | De Roon | Maehle | Djimsiti | Thiaw |
|---|---|---|---|---|---|
| 10 | 0.0 | 2.5 | 2.3 | 2.5 | 2.5 |
| | 2.5 | 1.9 | 1.7 | 1.7 | 1.8 |
| | 5.5 | 1.5 | 1.4 | 1.4 | 1.4 |
| | 7.0 | 1.3 | 1.3 | 1.3 | 1.3 |
| 20 | 0.0 | 3.8 | 3.5 | 3.7 | 3.7 |
| | 2.5 | 3.1 | 2.9 | 3.0 | 2.9 |
| | 5.5 | 2.7 | 2.6 | 2.6 | 2.6 |
| | 7.0 | 2.6 | 2.5 | 2.5 | 2.4 |
| 35 | 0.0 | 5.6 | 5.3 | 5.5 | 5.3 |
| | 2.5 | 4.9 | 4.7 | 4.8 | 4.6 |
| | 5.5 | 4.5 | 4.4 | 4.4 | 4.2 |
| | 7.0 | 4.4 | 4.3 | 4.3 | 4.1 |
| 50 | 0.0 | 7.4 | 7.1 | 7.3 | 7.0 |
| | 2.5 | 6.7 | 6.5 | 6.6 | 6.3 |
| | 5.5 | 6.3 | 6.2 | 6.3 | 5.9 |
| | 7.0 | 6.2 | 6.1 | 6.2 | 5.8 |
| 75 | 0.0 | 10.5 | 10.1 | 10.4 | 9.8 |
| | 2.5 | 9.8 | 9.4 | 9.6 | 9.1 |
| | 5.5 | 9.4 | 9.1 | 9.3 | 8.7 |
| | 7.0 | 9.2 | 9.0 | 9.2 | 8.6 |
| 100 | 0.0 | 13.5 | 13.0 | 13.4 | 12.6 |
| | 2.5 | 12.8 | 12.4 | 12.6 | 11.9 |
| | 5.5 | 12.4 | 12.1 | 12.3 | 11.5 |
| | 7.0 | 12.3 | 12.0 | 12.2 | 11.4 |

**Figure 2.17.** Results of the sprint simulation (4 of 4)

## 2.5 Players' clustering

Thanks to what has been obtained through the sprint simulation algorithm presented in §2.4, we can now proceed to the last phase of our work, which, as already mentioned in the introduction §1.2, involves the identification of clusters of players who exhibit similar athletic performance.
In particular, this section aims to identify which are the fastest players both over short distances and over long distances; so we consider the number of clusters to be 2 at each distance ($K = 2$).
To do this, we resort to various **clustering** techniques, which are machine learning techniques with an *unsupervised approach* in which groups of *entities*, characterized by several *attributes*, are created according to the similarity between them.
The general objective of these techniques is to maximize the distance between entities in different clusters and to minimize the distance between entities in the same cluster.
Applying this concept to our specific domain, we consider the following:

- The **entities** in our study are the players from the teams and matches under analysis.
- The **attributes** we focus on are the travel times obtained from the simulated sprints conducted in the previous section.

Among the various clustering techniques that have been developed over the years, we have chosen to explore two main classes: ***Partitional methods*** and ***Hierarchical methods***. Within each class, we have selected two specific algorithms to examine in detail:

1. ***K-Means Algorithm*** and ***K-Medoids Algorithm***;
2. ***Hierarchical Agglomerative Algorithm*** and its variant ***BIRCH Algorithm***.

The ***K-means Algorithm*** is one of the most popular algorithms used for data analysis. It begins by initializing $K$ cluster centroids within the feature space and then iteratively performs two steps: **assignment** and **update**.
In the *assignment* step, each data point is assigned to the cluster with the nearest centroid based on a distance metric, typically Euclidean distance. This step determines the initial clustering of the data points.
In the *update* step, the centroids of the clusters are recalculated as the geometric mean of the data points assigned to each cluster. This step adjusts the centroids to better represent the data points within each cluster.
The algorithm continues these steps iteratively until convergence is achieved. Convergence

occurs when the assignment of data points to clusters no longer changes significantly or when a maximum number of iterations is reached.
Through this iterative process, the *K-means Algorithm* aims to minimize the within-cluster sum-of-squares by optimizing the following objective function:

$$\min_{\{v_i\}_{i=1}^{K}} \sum_{i=1}^{K} \sum_{x \in C_i} \|x - v_i\|_2^2 \tag{2.24}$$

where $K$ is the number of clusters, $C_i$ represents the $i$-th cluster, $\mathbf{x}$ is a data point, $\boldsymbol{v}_i$ is the centroid of cluster $C_i$. The objective function measures the squared distance between each data point and its assigned centroid. By minimizing the sum of these squared distances, the algorithm aims to create compact and well-separated clusters.
Although the *K-means Algorithm* is widely recognized for its simplicity and scalability, it does come with a few limitations. One of these limitations is the requirement to specify the number of clusters ($K$) in advance, which may pose a challenge in some cases. However, in our specific case, this is not a problem as we have already determined the desired number of clusters.
Another limitation of the *K-means Algorithm* is its sensitivity to the initial placement of centroids. The algorithm's convergence and resulting cluster assignments can vary depending on the initial centroid positions. This sensitivity highlights the importance of using proper initialization techniques, such as the *K-means++*[1] method, to improve the algorithm's performance and mitigate this issue.
In the following figure we can see how this algorithm works:

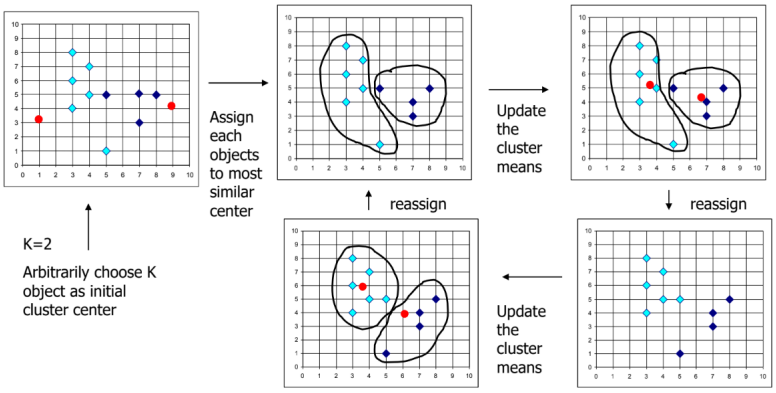



**Figure 2.18.** Functioning of the *K-Means Algorithm*

[1] *K-means++* is an algorithm for choosing the starting centroids for the *K-means Clustering Algorithm*.
Once the first centroid has been chosen from the data at our disposal, the centroids of the other clusters are chosen iteratively by selecting the point furthest from the previously selected centroids.

In our case, the *K-Means Algorithm* is appropriately implemented thanks to the *scikit-learn* Python package after applying a consistent **grid search** to identify the best method of initialization of the starting centroids (between a *random* type and a *K-means++* one) and the maximum number of iterations allowed.

The ***K-medoids Algorithm*** is an alternative clustering algorithm that addresses some of the limitations of the *K-means Algorithm*. Both algorithms aim to minimize a function that measures the distance between data points within a cluster and the designated center of the cluster.
In the case of ***K-medoids***, the algorithm minimizes the sum of actual distances to medoids by optimizing the following objective function:

$$\min_{\{m_i\}_{i=1}^{K}} \sum_{i=1}^{K} \sum_{x \in C_i} \|x - m_i\|_2 \tag{2.25}$$

where $m_i$ is the medoid of cluster $C_i$, and the objective function computes the total distance of each data point to its assigned medoid. The goal is to find the configuration of medoids that results in the smallest total distance.
Unlike *K-means*, the *K-medoids* uses data points called ***medoids*** as representatives of the cluster centers. Medoids are selected as the most centrally located objects within their clusters, having the least total distance to other members of the cluster. In contrast, the centroid used in *K-means* is an "artificial" point calculated as the geometric mean or barycenter of all points in the cluster.
The *K-Medoid Algorithm* operates similarly to the *K-means Algorithm*, using an iterative process with two steps. It employs the ***Partitioning Around Medoids*** (**PAM**) technique, which starts with an initial set of medoids. Given a current set of medoids $m_1, \ldots, m_K$, the goal is to minimize the total distance by assigning each observation to the closest (current) cluster center. In each iteration, the algorithm computes the cost of swapping each medoid with any non-medoid point. If replacing a medoid with a non-medoid point improves the total distance of the resulting clustering, the swap is made. This process continues until convergence is achieved, meaning the medoids no longer change significantly, or a predetermined number of iterations is reached.
One advantage of the *K-medoids Algorithm* is its robustness to outliers. Since medoids are actual data points, they are less affected by extreme values compared to centroid-based methods like *K-means*.
However, it's worth noting that the *K-medoids Algorithm* can be computationally more intensive than *K-means*, especially for large datasets. This is because dissimilarities between all data points and medoids need to be calculated at each iteration. Fortunately, in our case, the dataset we are considering is small, so this computational overhead

is not a concern.
Also in this case, the *K-Medoid Algorithm* is appropriately implemented using the *scikit-learn-extra* Python package. We also perform a consistent **grid search** to identify the best method of initializing the starting medoids. This grid search includes options such as selecting the initial medoids randomly, using a heuristic approach that picks the K points with the smallest sum distance to every other point, or applying the *K-medoids++* method, which follows an approach based on *K-means++*. Additionally, we also tune the maximum number of iterations allowed to ensure convergence.

***Hierarchical clustering*** is the second class of clustering techniques we are evaluating for this section. Its objective is to group similar data points into clusters based on their pairwise similarities or dissimilarities.
Hierarchical clustering algorithms produce hierarchical structures or trees, where clusters at each level of the hierarchy are created by merging clusters from the lower level. This hierarchical structure can be represented as a dendrogram, which visually depicts the clustering process and provides insights into the relationships and similarities between clusters and data points. It offers a highly interpretable and complete description of hierarchical clustering in a graphical format, which contributes to the popularity of hierarchical clustering methods.
By analyzing the dendrogram, one can observe how clusters are formed and gain insights into the optimal number of clusters. The height at which clusters merge in the dendrogram is proportional to the inter-group dissimilarity between the clusters. A higher merging height suggests greater dissimilarity between clusters, while a lower merging height indicates higher similarity. The choice of the optimal number of clusters can be determined by identifying significant changes in the merging heights, which correspond to distinct clusters in the data.

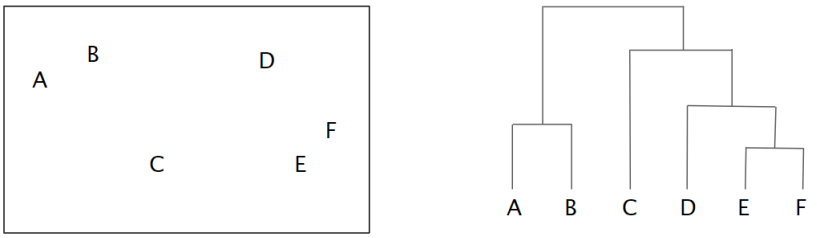



**Figure 2.19.** Example of a dendrogram

There are two main approaches to hierarchical clustering: ***agglomerative*** (bottom-up) and ***divisive*** (top-down). The agglomerative approach starts with individual data points and iteratively merges them into larger clusters based on their similarity. On the other hand, the divisive approach begins with a single cluster containing all the

data and recursively divides it into smaller sub-clusters. While both approaches have their merits, the agglomerative method has been more extensively studied in the literature. Therefore, we have chosen to focus on agglomerative methods for our analysis.

***Hierarchical Agglomerative Algorithm*** starts by assigning each data point to a separate cluster, forming the original forest of singleton clusters. The algorithm then calculates pairwise distances or dissimilarities between clusters using a chosen distance metric. At each iteration, the two closest (least dissimilar) clusters are merged into a single cluster, reducing the total number of clusters by one at the next higher level.
When two clusters $s$ and $t$ are merged into a single cluster $u$, $s$ and $t$ are removed from the forest, and $u$ is added to the set of clusters. This process continues iteratively until only one cluster remains in the forest, which becomes the root of the dendrogram.
A distance matrix is maintained at each iteration, reflecting the distance of the newly formed cluster $u$ with the remaining clusters $v$ in the forest. The distance between the newly formed cluster $u$ and each cluster $v$ can be calculated using different methods, known as *linkage criteria*, which are used to update the distance matrix. Here are some commonly used linkage criteria:

- *Single linkage*, measures the distance between the closest points of two clusters (least dissimilar);

$$d(u, v) = min(dist(u[i], v[j])) \quad \forall i \in u; \forall j \in v \tag{2.26}$$

- *Complete linkage*, measures the distance between the farthest points of two clusters (most dissimilar);

$$d(u, v) = max(dist(u[i], v[j])) \quad \forall i \in u; \forall j \in v \tag{2.27}$$

- *Average linkage*, calculates the average distance between all pairs of points from two clusters;

$$d(u, v) = \sum_{ij} \frac{d(u[i], v[j])}{|u| \cdot |v|} \quad \forall i \in u; \forall j \in v \tag{2.28}$$

  where $|u|$ and $|v|$ are the cardinalities of the two clusters.

- *Weighted linkage*

$$d(u,v) = \frac{dist(s,v) + dist(t,v)}{2} \tag{2.29}$$

and the cluster $u$ is composed by both cluster $s$ and cluster $t$;

- *Centroid linkage*

$$d(s,t) = ||c_s - c_t||_2 \tag{2.30}$$

where $c_s$ and $c_t$ are the centroids of clusters $s$ and $t$, respectively. When two clusters $s$ and $t$ are merged to form a new cluster $u$, the centroid of $u$ is computed by taking into account all the original objects in clusters $s$ and $t$. The distance between the newly formed cluster $u$ and a remaining cluster $v$ in the forest is then calculated as the Euclidean distance between the centroids of $u$ and $v$.

- *Median linkage*, measures the distance as the previous method and, when cluster $s$ and cluster $t$ are merged in a new cluster $u$, the centroid is the average between the centroids of $s$ and $t$;

- *Ward linkage*

$$d(u,v) = \sqrt{\frac{|v|+|s|}{T}d(v,s)^2 + \frac{|v|+|t|}{T}d(v,t)^2 - \frac{|v|}{T}d(s,t)^2} \tag{2.31}$$

where $u$ is the newly joined cluster consisting of clusters $s$ and $t$, $v$ is an unused cluster in the forest, $T = |v|+|s|+|t|$, and $|\cdot|$ is the cardinality of its argument.

The selection of a specific linkage criterion plays a crucial role in determining the dissimilarity between clusters. Different criteria can lead to distinct clustering outcomes. To identify the most suitable method, a grid search approach is employed. The results of this search will be presented in the subsequent section §2.5.1.
To implement this we resort to the *Scipy* Python library which, unlike the previously mentioned *scikit-learn*, has numerous functions for implementing hierarchical clustering techniques.

Furthermore, we also consider a variant of hierarchical algorithms called the ***BIRCH algorithm*** that stands for Balanced Iterative Reducing and Clustering using Hierarchies. The BIRCH algorithm aims to overcome one of the limitations of agglomerative clustering, which is the inability to undo previous clustering decisions.
The *BIRCH* algorithm operates in two phases.
In the first phase, known as ***micro-clustering***, *BIRCH* constructs a tree-like

structure called the **Cluster Feature Tree** (**CFT**). The CFT is a height-balanced tree that compactly represents the dataset by summarizing and reducing the data in *Clustering Feature nodes* (CF Nodes). This reduced data is then passed on to the second phase, known as ***macro-clustering***.
During the *macro-clustering* phase, the leaf nodes from the CFT is fed into a global clustering algorithm, typically an agglomerative clustering algorithm, and the number of desired clusters, denoted as $K$, can be specified for the macro-clustering step.
Each **CF Node** of the CFT contains a set of sub-clusters known as **Clustering Feature sub-clusters**. Their use offers the advantage of not needing to store all the available data; only a triplet of values is stored for each sub-cluster $i$, which represents key information. This triplet is denoted as $CF_i(N_i, \overrightarrow{LS_i}, SS_i)$, where:

- $\boldsymbol{N_i}$ is the number of data points in the $i$ sub-cluster in a $d$-dimensional feature space;
- $\boldsymbol{\overrightarrow{LS_i}} = \sum_{j=1}^{N_i} \overrightarrow{x_j}$, as the linear sum of the data points;
- $\mathbf{SS_i} = \sum_{j=1}^{N_i} (\overrightarrow{x_j})^2$, as the square sum of the data points.

The CFT consists of three types of nodes.
At level zero, there is a single root node, which serves as the starting point of the tree. Non-leaf nodes are present at intermediate levels of the tree and have at least one child node; these nodes represent the hierarchical merging of sub-clusters. Finally, leaf nodes are at the bottom level of the tree and do not have any child nodes; they contain sub-clusters of points in the feature space.
The structure of the CFT depends on two parameters: the ***branching factor*** $\boldsymbol{\beta}$ and the ***threshold*** $\boldsymbol{T}$.
The *branching factor* determines the maximum number of sub-clusters that can be stored at each node of the hierarchical tree. The choice of its value depends on the specific dataset and the desired level of granularity in the clustering results.
The *threshold* parameter is used to control the compactness of the sub-clusters. It specifies the maximum radius, or distance, allowed for a sub-cluster in the feature space. The threshold value influences the level of detail in the clustering results, with a smaller threshold resulting in more compact sub-clusters and potentially finer-grained clusters.
The construction of the Cluster Feature Tree in the BIRCH algorithm is done dynamically as objects are inserted: this makes the method incremental, meaning that the tree grows and adapts as new samples are added.
When a sample is inserted into the CFT, it is merged with the nearest sub-cluster

$i$ contained in the leaf node which results in the smallest **radius $R_i$** after merging. The radius is calculated as:

$$R_i = \sqrt{\frac{SS_i}{N_i} - \frac{LS_i^2}{N_i^2}} \tag{2.32}$$

This merging process is subject to constraints imposed by the threshold and branching factor conditions.
If the radius of the resulting sub-cluster exceeds the threshold and the number of sub-clusters in the node surpasses the branching factor, a temporary space is allocated to accommodate this new sample.
To maintain the branching factor condition, the two farthest sub-clusters in the node are selected, and two groups of sub-clusters are divided into two separate CF nodes, becoming nodes at the next level of the tree, based on the distance between these farthest sub-clusters. A leaf node remains a leaf as long as the branching condition is satisfied.
Once a maximum of B sub-clusters are formed in each leaf node of the tree, we proceed with the application of an agglomerative algorithm to the sub-cluster in the leaf node to obtain the final $K$ clusters. In this case, the method for calculating the distance between clusters can be one of the following: *Ward*, *Average*, *Complete*, and *Single*; these are previously described.
By tuning these parameters, we can achieve different trade-offs between clustering granularity, compactness, and computational efficiency.
The other advantage of using *BIRCH* over hierarchical clustering is its ability to handle large datasets more efficiently, reducing memory requirements and speeding up the clustering process, while effectively handling noise and outliers.
For this algorithm, too, we make use of the *scikit-learn* library, which also allows us to perform a grid search for the determination of parameters such as the *threshold* and the *branching factor*.

Once different clustering techniques have been introduced, we need to find the one that best meets our needs. To do this, it is important to analyze the goodness and quality of the solution returned by each technique introduced; we, therefore, calculate three different indices:

1. ***Mean Silhouette Coefficient S***, it is used to evaluate both the cohesion within each cluster of data and the separation between different clusters. Its value can be from -1 to +1, where a high value indicates that the objects are well-matched to their own cluster and poorly matched to neighboring clusters. The index can be computed with any distance metric (i.e. Euclidean distance, which is the default metric used in the function *sklearn.metrics.silhouette_score()*).

The Mean Silhouette Coefficient, denoted as $S$, is calculated as the average of individual silhouette coefficients $s(i)$ for all data points in the entire dataset. It can be expressed as:

$$S = \tilde{s}(K) \tag{2.33}$$

where $\tilde{s}(K)$ represents the mean $s(i)$ for a specific number of clusters $K$.
To calculate the Mean Silhouette Coefficient, we use the mean intra-cluster distance $a(i)$ and the mean nearest-cluster distance $b(i)$ for each sample $i$. Let

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, j \neq i} d(i, j) \tag{2.34}$$

be the mean distance between the point $i$ and all the points in the same cluster, where $|C_i|$ is the number of points belonging to cluster $C_i$. In this way, we can interpret $a(i)$ as a measurement of how well the point is assigned to its cluster (in this case, the smaller the value and the better the assignment). After this, we define the means dissimilarity of point $i$ to some cluster $C_k$ as the mean of the distance from $i$ to all points in $C_k$. For each data point in $C_i$, we define

$$b(i) = min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \tag{2.35}$$

as the smallest mean distance of $i$ to all points in any other clusters. The cluster with the smallest mean dissimilarity is said to be the neighbor cluster of $i$ because it is the next best-fit cluster for that point. We can finally define the Silhouette index $s(i)$ of one data point $i$ as

$$s(i) = \frac{b_i - a_i}{max\{a(i), b(i)\}} \sum_{j \in C_k} d(i, j) \quad if \ \ |C_i| > 1 \tag{2.36}$$

and

$$s(i) = 0 \quad if \ \ |C_i| = 1 \tag{2.37}$$

An $s(i)$ close to 1 means that the data is appropriately clustered, if $s(i)$ is close to $-1$ this means that this point would be more appropriate if it was clustered in its neighbor cluster. Thus the mean of all the different $s(i)$ is a measure of how appropriately the data have been clustered. If there are too many or too few clusters, some of the clusters will display much narrower silhouettes than the rest.

2. ***Davies-Bouldin index DB***, is used to evaluate the quality of a clustering algorithm by measuring the goodness of split. In practice, it is calculated as the average similarity of each cluster with a cluster most similar to it, where similarity is defined as the ratio of within-cluster distances to between-cluster distances. The better the clusters are separated and the better the result of the performed clustering performance. The minimum score is zero, with lower values indicating a better clustering procedure. The DB index is computed as follows:

$$DB = \frac{1}{K}\sum_{i=1}^{K} R_i \tag{2.38}$$

where $K$ is the number of clusters and $R_i$ is defined as the maximum similarity between cluster $C_i$ and any other cluster:

$$R_i = max_{j=1...K, j\neq i}(R_{ij}) \quad i = 1, ..., K \tag{2.39}$$

The similarity between clusters $C_i$ and $C_j$ is computed as:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \tag{2.40}$$

where $d_{ij}$ represents the Euclidean distance between the centroids of clusters $C_i$ and $C_j$, and $s_i$ is the average distance from the points in cluster $C_i$ to its centroid $v_i$:

$$s_i = \frac{1}{|C_i|}\sum_{x\in C_i} d(x, v_i) \tag{2.41}$$

3. ***Calinski-Harabasz index CH*** (Wei, 2020 [14]), also known as *Variance Ratio Criterion*, is the ratio of the sum of between-clusters dispersion and of inter-cluster dispersion for all clusters. A high CH means better clustering since observations in each cluster are closer together (more dense), while clusters themselves are further away from each other (well separated).
The CH index is computed using the following formula:

$$CH = \frac{BGSS}{WGSS}\frac{N-K}{K-1} \tag{2.42}$$

where, $N$ is the number of data points, ***BGSS*** is the between-group sum of squares (inter-cluster dispersion)

$$BGSS = \sum_{i=1}^{K} |C_i| \cdot ||v_i - b||^2 \tag{2.43}$$

where $b$ is the barycenter of the dataset.
Instead, ***WGSS***, the within-group sum of squares (intra-cluster dispersion), has the following form

$$WGSS = \sum_{i=1}^{K} \sum_{x \in C_i} ||x - v_i||^2 \tag{2.44}$$

and $v_i$ is the centroid of the i-th cluster.

It must be said that those presented are quite heterogeneous indices, while the Silhouette index and the Calinski-Harabasz must be maximized, the Davies-Bouldin index must be minimized, and while the first two return values around one, the last index can even reach values on the order of hundreds, in the best case.
Therefore, to perform a proper comparison between the different clustering methods, we must first normalize the values of each index to the highest value

$$\overline{S_i} = \frac{S_i}{\max_{\forall j} S_j} \quad \overline{DB_i} = \frac{DB_i}{\max_{\forall j} DB_j} \quad \overline{CH_i} = \frac{CH_i}{\max_{\forall j} CH_j} \tag{2.45}$$

and then, for each clustering technique, calculate a total scoring $TS$ value as follows:

$$TS_i = \omega_S \overline{S_i} - \omega_{DB} \overline{DB_i} + \omega_{CH} \overline{CH_i} \tag{2.46}$$

where $i$ is the i-th clustering algorithm to be evaluated and the vector $\omega$ contains the weights to be assigned to each scoring type. For convenience, we have chosen to assign equal weights to each index, and thus

$$\omega_S = \omega_{DB} = \omega_{CH} = \frac{1}{3} \tag{2.47}$$

Lastly, regarding the grid searches implemented for each individual clustering technique considered, we followed a similar approach as described earlier. However, due to the nature of using a single algorithm, we were unable to perform any normalization.
As a result, the scoring metric minimized within the grid search is the following, referred to as the ***custom scorer*** **CS**:

$$CS_i = S_i - DB_i + \frac{CH_i}{500} \tag{2.48}$$

### 2.5.1 Case Study

Shifting the focus to our case study, the dataset to which we will apply the clustering technique is therefore one in which each player is described by a set of features. Specifically, the features represent the vector of travel times obtained from the simulation conducted in the previous section §2.5.
The objective is to identify fast and slow players based on their travel times over short and long distances. To achieve this, each clustering technique is implemented by specifying the number of clusters as 2. This choice allows us to divide the players into two distinct groups: one representing fast players and the other representing slow players.
Before proceeding with the implementation of the clustering techniques introduced in the previous chapter §2.5 and choosing the appropriate one, it is advisable to perform an internal grid search within the agglomerative hierarchical algorithm. In fact, in the previous section, we mentioned the importance of first identifying the right metric for calculating the dissimilarity between clusters.
To do this, we executed the hierarchical clustering algorithm using each distance metric (e.g., *Single*, *Complete*, *Average*, *Weighted*, etc.) and then evaluated their scoring and *TS* as presented in the equation 2.46.

| | **S** | **DB** | **CH** | **TS** |
|---|---|---|---|---|
| ***Single*** | 0,463553 | 0,320912 | 6,52213 | 0,433926 |
| ***Complete*** | 0,49226 | 0,61114 | 60,10199 | 0,67803 |
| ***Average*** | 0,533123 | 0,62254 | 81,62112 | 1 |
| ***Weighted*** | 0,43720 | 0,50933 | 25,33632 | 0,312338 |
| ***Centroid*** | 0,533123 | 0,62254 | 81,62112 | 1 |
| ***Median*** | 0,49226 | 0,61114 | 60,10199 | 0,67803 |
| ***Ward*** | 0,533123 | 0,62254 | 81,62112 | 1 |

**Table 2.6.** Grid search choosing the best distance method in agglomerative clustering

The three methods that demonstrate the best performance are *Average*, *Centroid*, and *Ward*. Among these options, we choose ***Ward*** as our preferred method. *Ward* distance is the most commonly used method in hierarchical clustering implementations, including the widely-used scikit-learn library in Python. It is extensively utilized, well-tested, and easily accessible for data analysis purposes. Ward distance aims to minimize the internal variance within each cluster during the merging process. This approach leads to the creation of more compact and coherent clusters, with greater homogeneity within each cluster. Additionally, Ward distance takes into account the shape of the clusters during the merging process. This flexibility allows

it to identify clusters of different shapes and sizes, adapting to the underlying data structures more effectively.

Before presenting the final results, it is appropriate to identify the right clustering technique to adopt, and we do this by implementing the chosen techniques over the **entire feature space** (i.e. considering both long-distance and short-distance simulation features) and taking the one that maximizes the value of ***TS*** 2.46.
Thus, the values of the three indices recorded are the following:

| | **S** | **DB** | **CH** |
|---|---|---|---|
| ***K-Means Method*** | 0.529962 | 0.626428 | 82.404552 |
| ***K-Medoids Method*** | 0.529962 | 0.626428 | 82.404552 |
| ***Hierarchical Algorithm*** | 0.533124 | 0.622544 | 81.621128 |
| ***BIRCH Algorithm*** | 0.533124 | 0.622544 | 81.621128 |

**Table 2.7.** Scoring of clustering algorithms

And applying the normalization described above, we obtain:

| | $\overline{S}$ | $\overline{DB}$ | $\overline{CH}$ | **TS** |
|---|---|---|---|---|
| ***K-Means Method*** | 0.994069 | 1.000000 | 1.000000 | 0.331356 |
| ***K-Medoids Method*** | 0.994069 | 1.000000 | 1.000000 | 0.331356 |
| ***Hierarchical Algorithm*** | 1.000000 | 0.993800 | 0.990493 | 0.332231 |
| ***BIRCH Algorithm*** | 1.000000 | 0.993800 | 0.990493 | 0.332231 |

**Table 2.8.** Normalized scoring of clustering algorithms

From the table above, we can see that, taking the *TS* values into account, we can see that hierarchical clustering (both *Agglomerative* and its variation *BIRCH*) techniques perform better than Partitioning ones.

In particular, after evaluating the performance of the *BIRCH* and *Agglomerative Hierarchical* algorithms, we decided to opt for the agglomerative hierarchical algorithm. Despite the BIRCH algorithm not offering any significant improvement in performance, the agglomerative hierarchical algorithm was chosen for its lower complexity and intuitive visualization and analysis capabilities.

Let us now visualize the results obtained for the short-distance sprints, in green the fastest player and in orange the slowest ones.

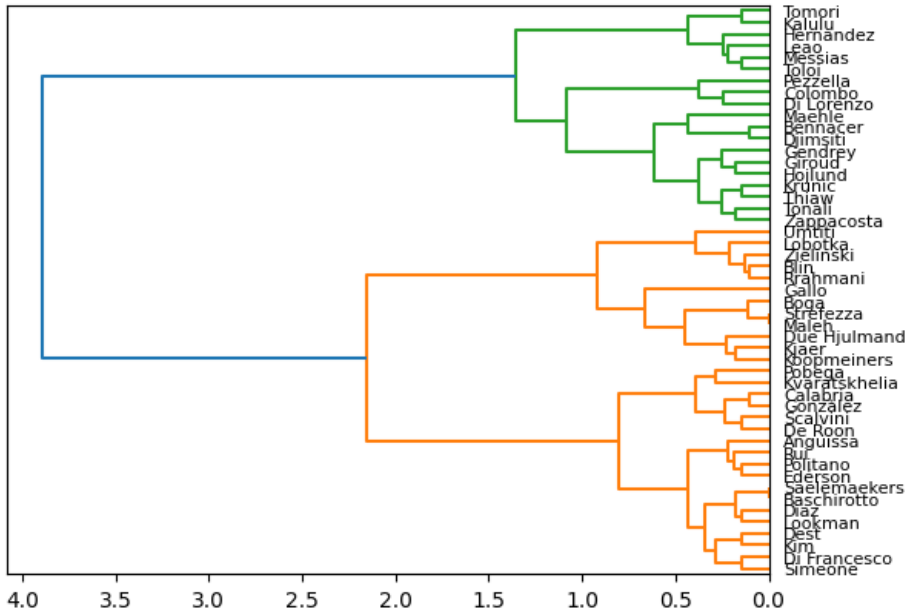



**Figure 2.20.** Clustering on short distances

From the obtained dendrogram, depicted in Figure [2.20], we can observe that, over short distances, the fastest players are more or less as expected. Specifically, there is a cluster, representing the more accelerative players (depicted in green), predominantly consisting of wingers and forwards. Their presence in the green cluster aligns with their primary role of exploiting the speed and making dynamic runs.

Interestingly, we also observe the presence of central midfielders and central defenders in the green cluster. These players are known for their exceptional athletic abilities, which differentiates them from their counterparts in the other cluster (depicted in orange).

Lastly, we emphasize that the obtained results are still influenced by the initial velocities considered in the sprint simulations. In fact, by focusing separately on both standing starts (0 $m/s$ and 2.5 $m/$s) and moving starts (5.5 $m/s$ and 7 $m/s$), we conclude that while the results remain unchanged in the former case, the number of players in the green one increases in the latter. This expanded set of players includes individuals such as Baschirotto, Kim, and Simeone, who may not excel in accelerating from a standstill but demonstrate good acceleration capacity or the ability to maintain high accelerations even at higher speeds.

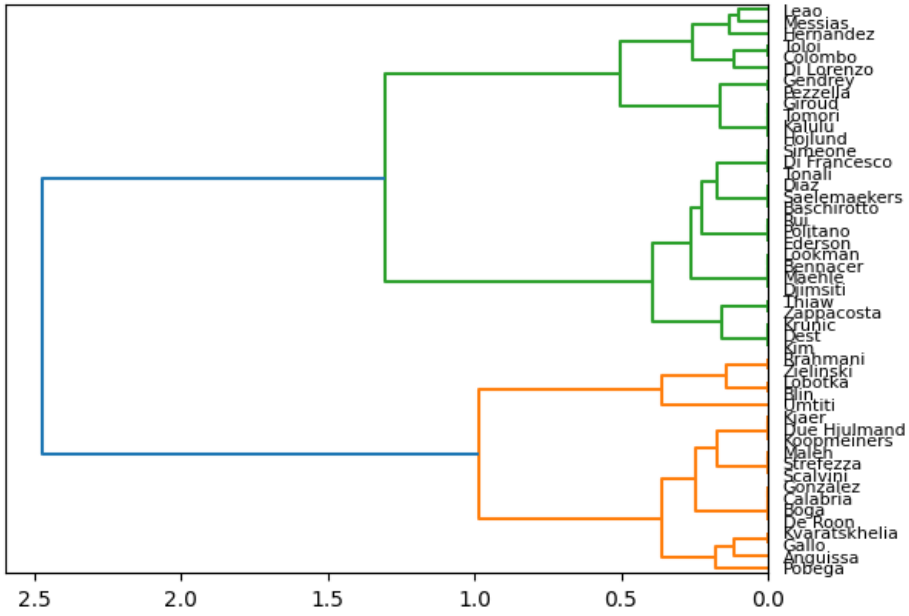



**Figure 2.21.** Clustering on short distances and high starting speed

The clusters of slow players and fast players over long distances are constructed as follows (in green the fastest players and in orange the slowest ones):

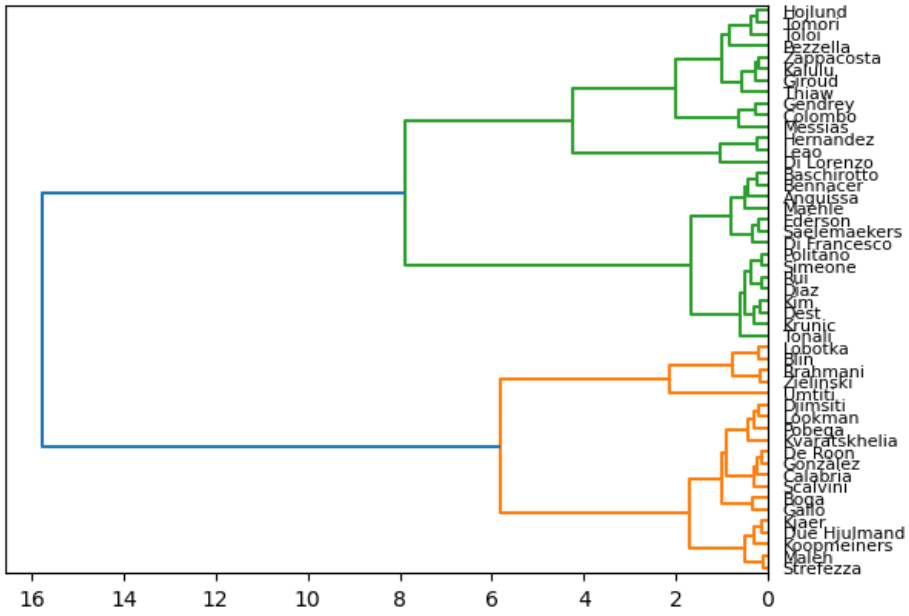



**Figure 2.22.** Clustering on long distances

Analyzing the results obtained from implementing the clustering algorithm on the features representing sprint times over long distances, we immediately notice an increase in the size of the green cluster.
In longer distances, in addition to the ability to achieve strong initial accelerations to reach maximum speed quickly, the absolute value of this maximum speed becomes crucial. Typically, players reach their maximum speed within the first 3-4 seconds and maintain it for the remainder of the sprint.
In this case, we observe that the players who are slower in longer distances are also among the slower ones in shorter distances, indicating that they not only have poor acceleration capabilities but also lower maximum speeds.
Among the faster players, we see that many of those who show good times in shorter distances and high initial speeds also join this group. This confirms the hypothesis that these players have difficulties in rapidly reaching high speeds, but once they reach them, they become competitive even with initially faster players.
When distinguishing between standing starts and moving starts, we do not observe any changes in the clustering results. This supports the notion that the absolute value of the maximum speed achieved becomes more important when covering longer distances. It is noteworthy that all players, including those with lower initial accelerations, are able to reach their maximum speed with such distances.

In conclusion, we employ the *Hierarchical Agglomerative Algorithm* to identify players with similar athletic performance across the entire feature space, encompassing all combinations of simulated starting distances and speeds.
As mentioned in the previous section §2.5, by examining the dendrogram, we can observe the formation of clusters and determine the optimal number of clusters. Clusters that merge at higher values, compared to the merge values of the sub-clusters located lower in the hierarchy, are considered potential natural clusters.
From the analysis of the previous dendrograms in Figure [2.20] and in Figure [2.22], it is evident that the optimal number of clusters is indeed two.

However, for a more detailed analysis, we can focus on the level where sub-clusters are formed within the hierarchy.

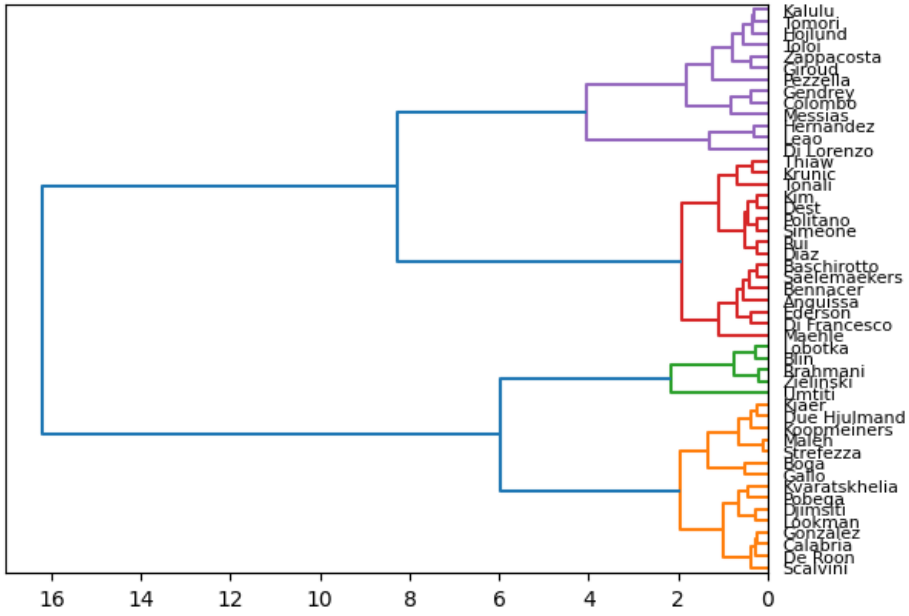



**Figure 2.23.** Clustering results on the whole feature space

This dendrogram is obtained by considering all the available features without distinguishing between short and long distances, as done previously.
This allows us to identify groups of players who are similar from an athletic perspective, which can be useful in various aspects ranging from tactical preparation for a match to player scouting for finding similar profiles.
To provide this detailed analysis, we can distinguish the four clusters shown in Figure [2.23]. Firstly, we can identify the group of players with the highest overall speed and the best acceleration capabilities (in purple). On the other hand, there

is a group (in green) consisting of players with the lowest acceleration capacity and maximum speed, making them the slowest in each simulation.
Furthermore, there are two additional intermediate groups in terms of maximum speed.
The first group (in red) primarily consists of players who, despite being among the slowest in the sprint simulations over short distances or when starting from a standstill, show a change in trend once the simulation is conducted over longer distances. The second group (in orange) includes players with poor acceleration abilities even at higher speeds, as they are unable to improve their times over longer distances. However, they can achieve a higher maximum speed compared to the green group.

# Chapter 3

# Conclusion

In this work, our focus was on the field of applied statistics and machine learning, specifically in the context of sports, with a particular emphasis on elite soccer players.

The first step of the work (§2.2) has been to develop a model for constructing the **acceleration profile** of soccer players, analyzing the relationship between instantaneous speed and instantaneous acceleration.
We have found that this relationship can be described by a parabolic function.
This curve captures the variations in height and amplitude across different players, highlighting the uniqueness and individuality of each player's athletic profile. This finding has reinforced the notion that each player has their own specific capabilities and performance characteristics.
However, before achieving this result, we encountered various challenges and conducted extensive testing. One of the tests involved evaluating the model using polynomial functions of degrees 3 and 4.
Unfortunately, both the third-degree polynomial and the fourth-degree polynomial have exhibited a critical issue. Specifically, the curves displayed a change in concavity, causing the acceleration to increase again at maximum speeds, resulting in unrealistically low sprint times during the simulations. This behavior has been inconsistent with the expected performance of the players and could have led to inaccurate estimations of their maximum sprint capability.
It is worth mentioning that this behavior has consistently been observed in third-degree polynomial models. However, for players with limited data available (where we have only had information from a single match), this behavior has also been observed in some of the fourth-degree polynomial models. It is possible that with a larger dataset, including multiple matches for each player, a fourth-degree polynomial could have provided a better description of the acceleration profile.

Another inherent challenge in this work is the management of **outliers**, which arises also from deriving acceleration values from player positions. To overcome this challenge, we have explored the application of robust regression techniques, such as **Huber regression**, known for their effectiveness in handling outliers. The significance of each value had to be carefully determined, as our focus was specifically on modeling the maximum values that truly represent a player's peak performance. However, the application of Huber regression has introduced its own set of challenges. It has necessitated the tuning of multiple hyperparameters through grid search, and these hyperparameters could vary from player to player, potentially limiting the model's universality. Additionally, the profiles generated using Huber regression often have appeared "flattened" due to the identification of visually significant maximum values as outliers. In light of these challenges, the approach of performing a preliminary regression and removing outliers based on the confidence interval has been deemed the most suitable and robust solution for our modeling task. It has allowed for the identification of meaningful maximum acceleration values while mitigating the impact of outliers on the overall profile construction.
When considering the encountered obstacles, we acknowledge the limitations posed by the limited quantity and quality of the available data. These constraints have influenced the depth and breadth of our analysis. Additionally, the data itself has presented challenges due to the different sampling frequencies utilized. Football is an intermittent sport characterized by dynamic physical demands influenced by various factors, including individual player characteristics, playing style, team formation, and specific match tactics (such as defensive or offensive strategies). Therefore, relying on data from a single match may provide only a partial view of the players' athletic performances. As a result, players like Kvaratskhelia and Lookman may have been "underestimated" by the model due to the limited range of speeds and accelerations captured in the specific match included in our dataset.
Despite these, the model can offer several **advantages** and benefits.
Firstly, it is less influenced by subjective choices, such as global thresholds, that do not consider the specific characteristics of players. By basing the model on match data, it avoids the limitations of relying on specific athletic tests, which do not reflect actual in-game performance. The model captures the various accelerations and sprints of different durations providing a more comprehensive and realistic assessment of players' acceleration and running speed capacities in a football context with the presence of a ball, opponents, teammates, and multidirectional movements. Furthermore, the model retains the benefits highlighted by Morin et al., 2021 [1], such as the evaluation of the profile at different moments during the season, which can help identify performance declines or assist in injury prevention. It provides continuous information to sports and medical staff about players' physical fitness

without requiring specific testing and the associated reluctance. Additionally, the model allows for targeted training interventions by identifying deficits in both acceleration and speed capabilities. This information can be utilized by coaches and trainers to design specific training programs tailored to improve the identified areas of weakness.

A last notable advantage of this model is its graphical representation, which allows us to assess the reliability and accuracy of the profile.

It is not solely reliant on a single quantitative measure derived from data processing but provides insights into the methodology used to obtain the profile. In other words, the model is **highly interpretable**: can be interpreted by data analytical experts and understood by professionals who are not experts in the data analysis domain.

In general, many successful AI-based models are considered black-box in nature, making it challenging to understand the inner workings and decision-making processes. The use of linear regression, on the other hand, is an intrinsically interpretable method. Unlike neural networks or unsupervised approaches, it allows for a better understanding of the relationship between the model's input and predictions.

The second step of the work (§2.3) has involved examining the **trend of stamina** by conducting a regression analysis on the maximum accelerations sampled at regular intervals. This analysis has performed by aggregating all available data from the players and the three matches. However, it is noteworthy that as more data becomes available for an individual player, the same methodology can be applied to obtain a personalized estimation of stamina.

To ensure a sufficient amount of data for calculating the profile at each sample, despite using the aggregated data from the three matches, a sampling interval of every quarter of the game has been employed. By incorporating more data, a denser evaluation can be performed, such as recalculating the profile every fifteen minutes. This enables a more granular assessment of stamina levels throughout a match.

Furthermore, utilizing the acquired acceleration profiles, we have conducted **simulations of sprints** (§2.4) over both short and long distances and recorded the corresponding times. Upon visualizing the outcomes, we have discovered that the obtained values are reasonable and align with real-world observations. These simulated times, which take into account the players' maximum acceleration effort, can serve as personalized benchmarks for individual players during sprint training sessions.

Lastly, utilizing the sprint simulation times, we have employed **clustering** techniques to identify the fastest groups of players for both short and long distances (§2.5). This approach has provided valuable insights into categorizing player performance and can inform strategic decision-making in team selection and training programs.

It is worth highlighting the significance of using an agglomerative method, which,

through the construction of a dendrogram, enhances the clarity of player similarities and provides interpretability to the clustering process. Additionally, it has enabled us to assess the optimal number of clusters to form, rather than relying solely on numerical indices to validate the quality of the obtained clustering.

In conclusion, our work is part of a context in which, as explained in Ofoghi et al., 2013 [15], the widespread use of effective analytical approaches in sports performance analysis is hindered by several obstacles. These include a lack of belief that performance analysis outcomes can effectively improve future athlete performances in major competitions, a lack of a well-defined data analytical framework, and a lack of confidence and mutual understanding between sports data analysts and professional coaches/athletes.
On the other hand, this study and the interpretability of its model, which are a significant step forward in understanding the **maximum sprint capability** of soccer players, push the sports industry to keep up with the times and align with other industries that leverage analytical models as true tools for gaining a competitive advantage.
We hope that our study serves as a reference for future research aimed at improving the understanding of player performance, optimizing game tactics, personalizing training programs, evaluating opponent performances, and conducting scouting activities to identify new talents.

# Bibliography

[1] J.B Morin, Y. LeMate, C. Osgnach, A. Barnabò, A. Pilati, P. Samozino, and P.E. di Prampero. Individual acceleration-speed profile in-situ: A proof of concept in professional football players. *Journal of Biomechanics*, 2021.

[2] F. Van Breugel, J.N. Kutz, and B. Brunton. Numerical differentiation of noisy data: A unifying multi-objective optimization framework. *IEEE Access*, 2020.

[3] C. Osgnach, S. Poser, R. Bernardini, R. Rinaldo, and P.E. di Prampero. Energy cost and metabolic power in elite soccer: a new match analysis approach. *Med Sci Sports Exerc*, 2010.

[4] C. Osgnach. How easy is it to stumble over acceleration and deceleration? https://www.gpexe.com/2022/03/11/stumble-over-acceleration-deceleration/, 2022.

[5] M.C. Varley and R.J. Aughey. Acceleration profiles in elite australian soccer. *International Journal of Sports Medicine*, 2013.

[6] A.J. Sweeting, S.J. Cormack, S. Morgan, and R.J. Aughey. When is a sprint a sprint? a review of the analysis of team-sport athlete activity profile. *Frontiers in Physiology*, 2017.

[7] M. Buchheit, H. Al Haddad, B.M. Simpson, D. Palazzi, P.C. Bourdon, V. Di Salvo, and A. Mendez-Villanueva. Monitoring accelerations with gps in football: Time to slow down? *International Journal of Sports Physiology and Performance*, 2014.

[8] A. Couderc, C. Thomas, M. Lacome, J. Piscione, J. Robineau, R. Delfour-Peyrethon, R. Borne, and C. Hanon. Movement patterns and metabolic responses during an international rugby sevens tournament. *International Journal of Sports Physiology and Performance*, 2017.

[9] P. Samozino, JB. Morin, S. Dorel, and G. Rabita. A simple method for measuring power, force and velocity properties of sprint running. January 2013.

[10] S.M. Ross and F. Morandin. *Probabilità e statistica per l'ingegneria e le scienze*. 2008.

[11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. 2001.

[12] J.B. Morin, P. Samozino, M. Murata, M.R. Cross, and R. Nagahara. A simple method for computing sprint acceleration kinetics from running velocity data: Replication study with improved design. *Journal of Biomechanics*, 2019.

[13] K. Furusawa, A.V. Hill, and J.L. Parkinson. The dynamics of "sprint" running. *Proceedings of the Royal Society of London. Series B, Containing Papers of a Biological Character*, 1927.

[14] Haitian Wei. How to measure clustering performances when there are no ground truth, January 2020. https://shorturl.at/boxCT.

[15] B. Ofoghi, J. Zeleznikow, C. Macmahon, and M. Raab. Data mining in elite sports: A review and a framework. *Measurement in Physical Education and Exercise Science*, July 2013.

[16] Arthur Samuel. Arthur samuel (computer scientist). Wikipedia, April 2023. https://en.wikipedia.org/wiki/Arthur_Samuel_(computer_scientist).

[17] A. Periklis. Data clustering techniques. April 2002.

[18] M. Ghesmoune, M. Lebbah, and H. Azzag. State-of-the-art on clustering data streams. *Big Data Analytics*, 2016.