HAMK
Häme University
of Applied Sciences

Enhancing decision-making in football players using predictive analytics

Bachelor’s Thesis

Degree Programme in Computer Applications

Spring 2025

Tuan Dang

Abstract

DP Degree Programme in Computer Applications
Author Tuan Dang Year 2025
Subject Enhancing decision-making in football players using predictive analytics
Supervisors Dr. Jawad Yasin

The purpose of the thesis was to enhance football players' decision-making using predictive analytics and computer vision. The research aimed to support Calevala Interactive Oy by developing a system that analyses player behavior from match videos to improve tactical awareness and reduce in-game decision-making errors. The study focused on three research questions: how predictive analytics improve real-time decision-making, which performance metrics are most influential, and how effective AI-driven training is in practice. The thesis was commissioned by Calevala Interactive Oy.

This thesis takes a practical approach. It begins by reviewing important concepts such as machine learning, computer vision, and sports analytics. The project used object detection models to collect data from professional football match videos. Computer vision techniques helped extract player and ball movements. The data was then grouped using unsupervised learning methods, specifically clustering. K-Means clustering and the Elbow Method were used to find player behavior patterns. These results were visualized with heatmaps and movement tracks to support tactical training.

The research demonstrates that video-based predictive analytics can effectively improve decision-making by recognizing patterns in player and ball movements. The analysis indicates that metrics like positioning, possession time, and player proximity are key to successful decisions. Based on the findings, it is recommended that football organizations integrate video-based AI tools into their training to support more informed, data-driven strategies. Feedback from the commissioner confirmed that the system offers valuable support for ongoing development and real-world application.

Keywords Machine Learning, Artificial Intelligence, Computer Vision, Data Analyst.
Pages 54 pages and appendices 63 pages

Glossary

ML	Machine Learning
AI	Artificial Intelligence
CV	Computer Vision
SSL	Semi-supervised learning
ASI	Artificial Super Intelligence
TD	Temporal-difference
SL	Supervised Learning
UL	Unsupervised Learning
RL	Reinforcement Learning
DP	Dynamic programming
MDP	Markov Decision Process

Table of Contents

Figures

Tables

Equations

Program codes

Commands

Appendices

1 Introduction

Machine learning (ML) is a part of artificial intelligence (AI) that helps with tasks like sorting data, making predictions, or improving results (Nakayama et al., 2022, p. 1). It has a profound and diverse impact on many industries, from improving operational efficiency to improving customer experience and supporting decision-making.

The growing impact of Machine Learning spans various industries such as finance, social media, education and many more. As ML continues to evolve, its applications extend beyond traditional industries and into sports analytics, where data-driven insights are transforming performance evaluation and tactical decision-making.

This thesis aims to enhance football players' decision-making skills using predictive analytics, focusing on improving the performance of Calevala Interactive Oy's projects. The research will analyze player performance metrics, historical game data, and situational awareness to create an intelligent decision-support system that helps players and coaches optimize their strategies. The project will involve programming machine learning models using Python, and several machine learning libraries, such as Scikit-learn, and Pandas, will be installed to facilitate data analysis, model training, and performance evaluation. The following three research questions are answered throughout this thesis to achieve the goal:

- How can predictive analytics improve football players' real-time decision-making?
- How do key performance metrics influence successful in-game decisions?
- How effective is AI-driven training in enhancing tactical awareness and reducing decision-making errors?

The thesis consists of several chapters. Chapter 2 explains key theories, including machine learning, AI, sports analytics, and decision-making in football. Chapter 3 describes the research method, data collection, and ethical considerations. Chapter 4 focuses on AI-driven tactical analysis and training. Chapter 5 presents the main tools used, such as Python, OpenCV, YOLO, Visual Studio Code, and Anaconda. Chapter 6 explains how the project was developed and implemented. Chapter 7 shows the results based on the analysis.

2 The Evolution of Intelligence in Sports

This chapter explains how smart technologies are changing the world of sports. It focuses on computer vision, machine learning, and artificial intelligence, and how these tools help improve player performance, game analysis, and decision-making. Chapter 2.1 will introduce and history of computer vision. Chapter 2.2 will cover machine learning and its types, showing how machines learn from data to find patterns and make predictions. Chapter 2.3 will explain what artificial intelligence is, its history, and the types of AI used in sports. Chapter 2.4 will look at the basic ideas behind sports analytics and the tools used to study sports data. Finally, Chapter 2.5 will show how machine learning and AI are used in sports.

2.1 Computer Vision

Computer vision is the use of computers and special devices to copy how human vision works, mainly by analyzing pictures or videos to understand and gather 3D information about a scene (Gao et al., 2020, p. 8). The main purpose of computer vision is to build machines that can see and understand their surroundings, then use what they see to make choices or do something (Moin, 2023). This includes things like recognizing objects, rebuilding scenes, identifying faces, and dividing images into parts.

Computer vision started in the 1960s when new technology led to the idea of creating systems that could act like humans. But since computers were slow at that time, a new field was developed to help machines process images better. Around 1964, the focus was on fixing blurry satellite images, and by 1970, millions had been processed, leading to the first methods for comparing images. In 1965, a major step was made with an algorithm that turned camera images into simple outlines, using basic shapes like cubes and prisms to recognize objects. These shapes were adjusted and tested until a match was found (Ivanov, 2025, p. 178). Although simple today, this was a big achievement then and helped create early robots that could understand their surroundings and predict their next moves.

In recent years, computer vision has improved with better photo editing methods like HDR, image stitching, and texture filling, now called computational photography. At the same time, new ways to recognize objects and scenes using features and machine learning were developed. In the 2000s and 2010s, deep learning and CNNs made a big change by helping machines learn to recognize things more accurately and with less human help (Szeliski, 2022).

Computer vision is now used in many areas, such as analyzing medical and biological images (Grys et al., 2017), identifying types of rocks (Patel & Chatterjee, 2016), finding and classifying road damage, detecting damage in buildings (Cha et al., 2017), and more (Gao et al., 2020). It is also becoming more common in sports. For example, computer vision helps in sports broadcasts by showing 3D models of players or the ball, making it easier for viewers to understand positions and movements. It is also used in training, coaching, and even to support referees in making better decisions during matches (Thomas et al., 2017).

To sum up, computer vision has grown a lot—from early efforts in the 1960s to today's advanced systems that can understand complex pictures and videos. It is now widely used in healthcare, construction, and sports, helping machines "see" and make decisions based on what they observe. This progress has been made possible through machine learning, which helps computers learn patterns in visual data. The next part will explain what machine learning is and explain each type of machine learning.

2.2 Machine Learning

In recent years, the role of intelligence in sports has risen exponentially with the pressure from technological advancement. This chapter explores the history of intelligence in sports, starting with an introduction to machine learning. It identifies what machine learning is, how it works, and provides definition for each type of machine learning.

2.2.1 Introduction to Machine Learning

Machine learning is a scientific field that creates algorithms and statistical models, allowing computers to complete tasks without being explicitly programmed (Feng et al., 2024, p. 77). These learning algorithms are widely used in many everyday applications (Mahesh, 2020).

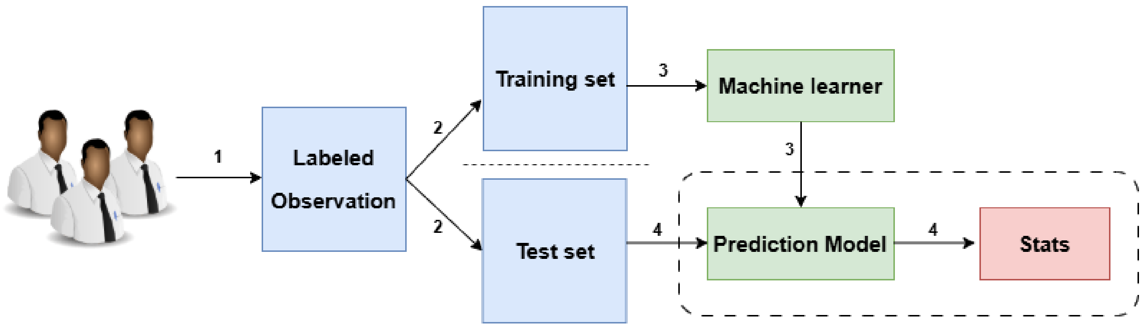
Machine learning models can be categorized based on the extent of human involvement with raw data, such as rewards, explicit feedback, or labeled examples. Sangeeta et al. (2024, p. 153) identifies four main types of machine learning models such as supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

2.2.2 Supervised learning

Supervised learning (SL) involves training a model to understand the relationship between input data and correct outputs, enabling it to predict results for new data (Cunningham et al., 2008, p. 23). It derives this mapping function from labeled training data made up of known examples (Mahesh, 2020). Organizations apply this method to tackle large-scale real-world problems, such as filtering spam emails (Byeon et al., 2023, p. 4) or forecasting stock prices.

Figure 1 explains the process of supervised learning. It starts with labeled observations, i.e., examples for which the outcomes are known. They are then divided into the training set and the test set. The training set enables the machine to learn and build from which the predictive model is derived. The test set checks how well the model performs. Statistics quantify how well the model performs in the final step.

Figure 1. Process of supervised learning (Salian, 2018)



Supervised learning usually helps with two kinds of problems. One is classification, where the goal is to determine which group something belongs to (Salian, 2018). The other type is regression. Regression models are used to predict continuous values, such as height, weight, or temperature (Nasteski, 2017, p. 3). The next part explains how unsupervised learning works in more detail.

2.2.3 Unsupervised learning

Unsupervised learning (UL) is a type of algorithm that finds patterns and structure in data without needing labeled answers like in supervised learning (Tyagi et al., 2022, p. 2). In this approach, the AI system is provided only with input data, without any corresponding output labels (Naeem et al., 2023, p. 1). The data includes many examples, but none come with the correct answer. The model looks for patterns, similarities, or structures in the data, helping uncover hidden relationships that might not be obvious.

Naeem et al. (2023) state that unsupervised learning techniques are mainly used for three key tasks: clustering, association, and dimensionality reduction.

Clustering is a type of unsupervised learning widely used and studied in computer vision (Caron et al., 2018, p. 1). These methods help find groups of similar items, where items in the same group share more features than those in other groups. This approach helps uncover meaningful insights by building a simplified model highlighting essential data patterns (Rodriguez et al., 2019, p. 2). According to Naeem (2023, p. 913), there are three types of clustering such as exclusive, overlapping, and hierarchical. Table 1 will define each category of clustering.

Table 1. Definition of each category of clustering

Type	**Definition**
Exclusive Clustering	Each data point belongs to only one group (Caron et al., 2018; Naeem et al., 2023, pp. 913–914).
Hierarchical Clustering (HCA)	Groups are formed like a tree: - **Agglomerative**: Each point starts alone, then groups are merged. - **Divisive**: All points start together, then are split into smaller groups. (Maji et al., 2018; Naeem et al., 2023, pp. 913–914)
Overlapping Clustering	A data point can belong to multiple groups at the same time (Ma, 2022; Naeem et al., 2023, pp. 913–914)

Association rule mining was first introduced by Agrawal, Imielinski, and Swami in 1993 during the ACM SIGMOD Conference on Data Management (Agrawal et al., 1993). Its goal is to find useful connections, common patterns, or relationships between groups of items in transaction databases or other types of data storage (Qiankun Zhao & Sourav S. Bhowmick, 2003).

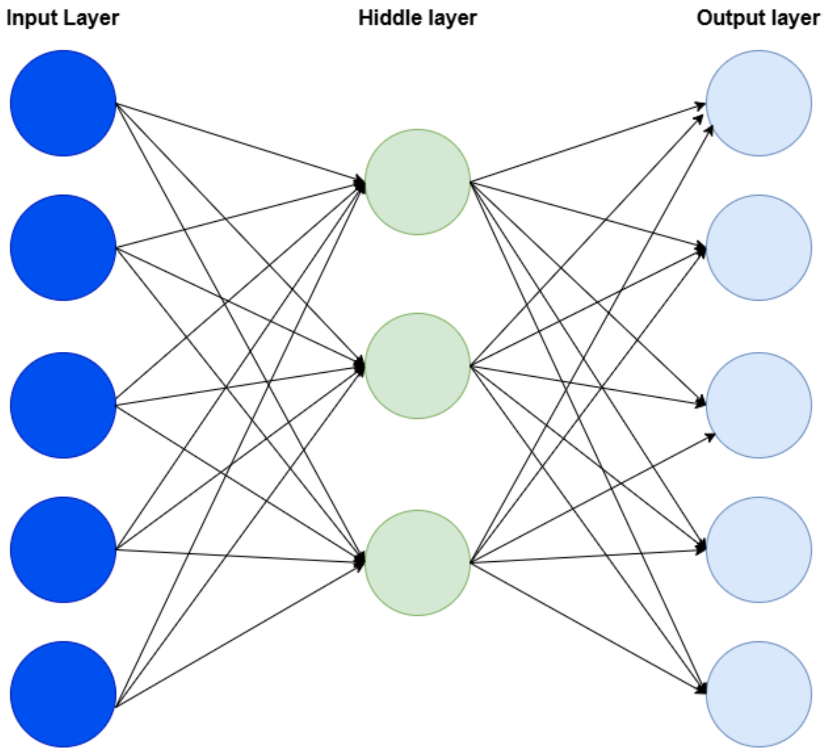
Dimensionality reduction encompasses techniques that remove redundant information and reduce the number of variables without compromising critical information, and such techniques—commonly used in machine learning and statistics—are a current research focus area (Sorzano et al., 2014). Saharawat (2023) and Zandavi et al. (2021) outlined several dimensionality reduction techniques that can be applied, such as Principal Component Analysis, t-Distributed Stochastic Neighbor Embedding (Smith-Vickery, 2023, p. 63), and Autoencoder. Table 2 shows the definition of each technique.

Table 2. Definition of each technique

Method	**Definition**
Principal Component Analysis (PCA)	Reduces a group of numbers into fewer new values called principal components to show patterns (Abdi & Williams, 2010; El-Ghany et al., 2024, p. 11).
t-Distributed Stochastic Neighbor Embedding (t-SNE)	Reduces complex data into a simple map that shows how data points are related (Belkina et al., 2019, p. 2; van der Maaten & Hinton, 2008).
Autoencoder	A neural network that compresses input data and rebuilds it to learn useful features (Bank et al., 2020; Michelucci, 2022, p. 2).

Figure 2 shows how an autoencoder works, using three layers: the input layer, hidden layer, and output layer. The input layer (on the left) takes in the original data and passes it to the hidden layer, where the data gets compressed. Then, the output layer (on the right) tries to reconstruct the original data from this compressed form.

Figure 2. How autoencoders work (IBM, 2021)



While unsupervised learning has many benefits, it also comes with significant challenges—it is often more difficult to apply than supervised methods. According to James et al. (2023, p. 504), “the exercise tends to be more subjective, and there is no simple goal for the analysis”, which highlights the open-ended nature of certain types of data exploration. These challenges are further compounded by several practical limitations, including computational complexity from large training datasets, longer training times, increased risk of inaccurate outcomes, the need for human validation of results, and a lack of transparency in how the data is grouped (IBM, 2021).

2.2.4 Semi-supervised learning

Semi-supervised learning (SSL) unites concepts in both supervised and unsupervised learning (Chapelle, Schölkopf, et al., 2006; Zhu & Goldberg, 2009) to utilize labeled and unlabeled data to help improve the capacity of models to cluster or predict new unseen data (Van Engelen & Hoos, 2020, p. 374). This is to imply that a model can be trained to label data with less labeled training data.

According to a survey on semi-supervised learning by Van Engelen and Hoos (2020), there are three assumptions: cluster assumption, smoothness assumption, low-density assumption and manifold assumption. Table 3 will describe each assumption.

Table 3. Definition of each assumption

Method	**Definition**
Cluster Assumption	Data forms clusters where each cluster mostly shares the same label (Chapelle, Zien, et al., 2006; Van Engelen & Hoos, 2020, p. 378). Unlabeled data helps to detect and define these clusters.
Smoothness Assumption	Nearby data points often share the same label, useful for handling unlabeled data (Van Engelen & Hoos, 2020).
Low-density Assumption	The boundary should run through low-density areas and steer clear of high-density regions to improve generalization (Chapelle, Schölkopf, et al., 2006; Feofanov et al., 2023, p. 2).
Manifold assumption	High-dimensional data lies on locally flat, lower-dimensional manifolds. In semi-supervised learning: (a) data lies on one or more such manifolds, and (b) points on the same manifold usually share a label (Van Engelen & Hoos, 2020, p. 378).

Figure 3 below builds on this by showing the low-density assumption: the optimal boundary (dashed line) passes through a sparse region between clusters. In contrast, the supervised model's boundary (solid line) may be misaligned due to the lack of labeled examples. In contrast, a supervised algorithm (solid line) may place the boundary incorrectly due to limited labeled data.

Figure 3. Smoothness and low-density assumptions (Van Engelen & Hoos, 2020)

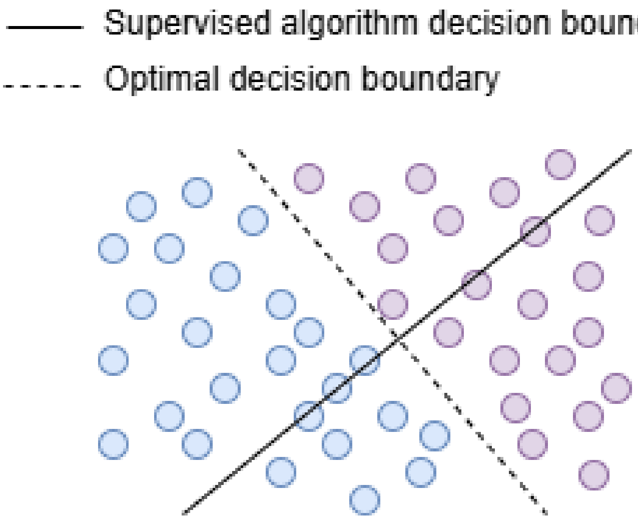
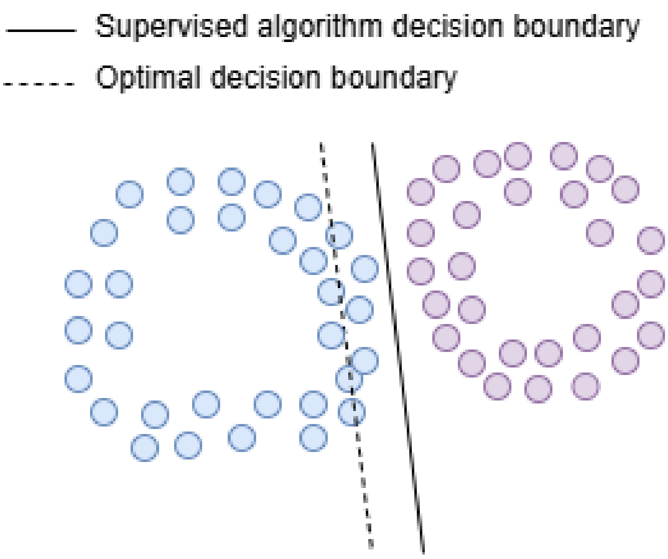


Figure 4 illustrates how mapping data to a lower-dimensional manifold helps reveal a more precise class separation, which leads to a more accurate decision boundary (dashed line) than the one learned by a standard supervised algorithm (solid line).

Figure 4. How mapping data to a lower-dimensional manifold helps reveal a more precise class separation (Van Engelen & Hoos, 2020, p. 377)



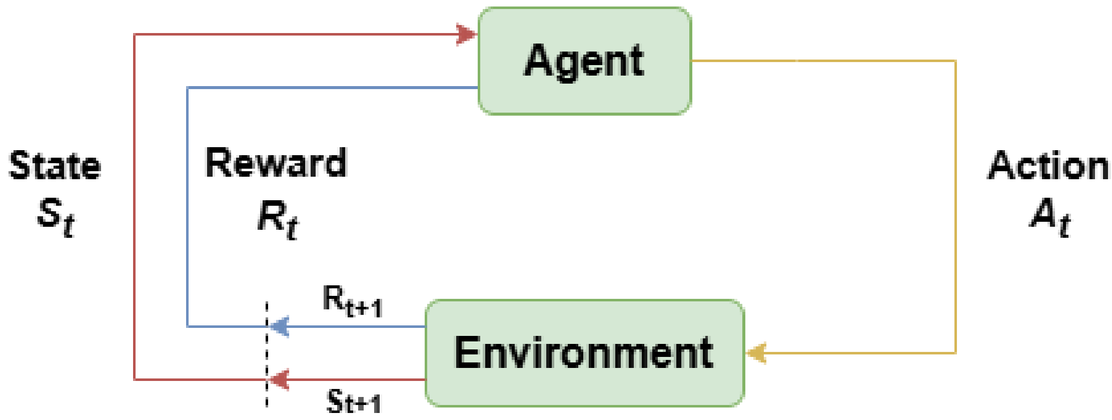
Semi-supervised learning is useful because it can use both labeled and unlabeled data. This can help the model perform better than using only labeled data (Zhu & Goldberg, 2009). This approach helps reduce the high cost and time associated with manual data labeling and often results in improved model accuracy—especially when labeled data is limited. Additionally, it is particularly effective for handling unstructured data types such as text, audio, or video, where large volumes of unlabeled data are commonly available.

SSL uses the setup of supervised learning and adds the advantages of unsupervised learning. While not appropriate for every situation, its inherent flexibility makes it a feasible option for a wide spectrum of project needs and goals (Chen, 2024).

2.2.5 Reinforcement learning

Reinforcement learning is about teaching an agent how to act by letting it try things out and learn from the results it gets from its environment. (Kaelbling et al., 1996, p. 237). An autonomous agent is a system that can make choices and take actions based on its surroundings without needing direct instructions from a person (Heaton, 2018; Murel & Kavlakoglu, 2024). Figure 5 illustrates how reinforcement learning works.

Figure 5: Reinforcement learning (Mahesh, 2020)



As shown in Figure 5, reinforcement learning consists of two main components: the environment and the agent. At each time step *t*, the environment is in a certain state S_t. The agent senses the environment and chooses an action A_t. It submits an action to the environment. The environment transitions to another state S_{t+1} and sends the reward R_{t+1} back to the agent. The reward sends feedback to the agent to indicate whether the action is right or wrong. The agent then uses this reward to learn and make better choices in the future. This process continues over time, helping the agent improve through experience.

RL involves an agent interacting with its environment to learn the best strategy or policy through trial and error. It is commonly applied to solve sequential decision-making problems across various natural sciences, social sciences, and engineering domains (Y. Li, 2018).

There are three main methods of reinforcement learning, each with a different way for an agent to learn and improve its decisions by interacting with the environment. These methods are Monte Carlo methods, dynamic programming, and temporal difference learning (Smadi, 2017, p. 91; Sutton & Barto, 1998).

Dynamic programming is a method used to solve problems by making a series of decisions step by step (Howard, 1966). Although DP has certain similarities with state-space search methods, it differs in that it evaluates actions incrementally at each state without explicitly considering complete sequences of actions (Richard S. Sutton, 1990). According to Murel and Kavlakoglu (2024), in this context, the agent does not select actions randomly but considers how each action will affect the next state. The reward r that the agent gets is based on its current state s, the action it chooses a, and the next state s' it moves to (Sammut & Webb, 2010, p. 299). This relationship is defined by the Markov Decision Process (MDP) reward function, as shown in Equation 1:

Equation 1: The reward function (Sutton & Barto, 1998, p. 96)

$$r_t(s,a) = \sum_{s' \in S} r_t(s,a,s') p_t(s'|s,a)$$

In Equation 1, $r_t(s,a)$ represents the expected reward for taking action a in state s at time t. The summation symbol $\sum_{s'}$ means that we are adding up the values for all possible next states s'. The term $r_t(s,a,s')$ represents the reward an agent gets when moving from state $\boldsymbol{s}$ to state s' by taking action a; this reward depends on the current state, the chosen action, and the next state (Sammut & Webb, 2010, p. 299). The term $p_t(s' \mid s,a)$ is the probability that the agent will end up in state s', given that it started in state s and chose action a. The reward function helps define the policy that guides the agent's actions. In reinforcement learning, dynamic programming methods aim to find the optimal policy that allows the agent to achieve the highest possible reward over time (Murel & Kavlakoglu, 2024). The Bellman equation is commonly used for this purpose, providing a way to calculate the value of each state. The Bellman equation is shown in Equation 2.

Equation 2: Bellman optimality equation (Murel & Kavlakoglu, 2024)

$$v_t(s) = \max_{a \epsilon A(x)} \{r_t(s,a) + \sum_{s\prime \in S} p(s'|s,a) v_{t+1}(s') \}$$

Equation 2 defines $v_t(s)$ as the total expected reward that the agent can obtain starting from state s at time t until the decision-making process ends. According to Murel and Kavlakoglu (2024), it “assumes that the agent begins by occupying state s at time t”. They also explain that “the equation ultimately divides the reward at time t into the immediate reward $r_t(s,a)$” and the agent’s total expected reward. By using this method, the agent tries to make the best decision in each state by choosing the action that gives the highest possible reward over time (Murel & Kavlakoglu, 2024; Puterman & Patrick, 2017).

Another reinforcement learning method is the Monte Carlo method, developed by John von Neumann and Stanislaw Ulam (Eckhardt, 1987). While dynamic programming makes decisions by predicting future states and rewards, Monte Carlo methods learn from experience using examples of what happened, what actions were taken, and what rewards were received during real or simulated interactions (Alonso & Mondragón, 2013, p. 142; Sutton & Barto, 1998, p. 113).

Temporal-difference (TD) learning is one of the main methods used in today's reinforcement learning (Kaelbling et al., 1996; Sutton, 1988; Sutton & Barto, 1998; Szepesvári, 2010). Temporal difference learning is a valuable form of learning when a reward is long in coming. It can be applied to make good predictions also and aid in making good decisions by performing actions leading to good predict results (Tesauro, 1995). It is similar to Monte Carlo techniques in learning based on experience and without any need to model the environment. It is similar to dynamic programming in updating an estimate in terms of other learned values before experiencing the outcome (Sutton & Barto, 1998, p. 143). What sets TD apart is its use of the difference between expected and actual rewards—this "temporal difference" guides the learning process as the agent adjusts its predictions step by step. Among its various forms, two key types are “State–action–reward–state–action SARSA and Q-learning” (Murel & Kavlakoglu, 2024; Sutton & Barto, 1998, p. 167). SARSA follows an on-policy approach, improving the same policy it uses to implement. At the same time, Q-learning is off-policy, learning an optimal policy regardless of the agent’s current behavior by separating exploration and exploitation strategies (Murel & Kavlakoglu, 2024).

2.3 Artificial Intelligence

Since the Industrial Revolution, technology has grown a lot and taken over many tough jobs that people used to do. One big step in this growth is Artificial Intelligence (AI). AI is a part of science that makes intelligent machines and computer programs that can do jobs that typically need human thinking. It uses a lot of data (called big data) to do these jobs well. In the past, AI was just an idea seen in movies or talks about the future. But today, it is part of our daily life and is used in many areas such as factories, hospitals, and delivery systems. Because AI can do things people can't, it helps make work faster and better (PK, F. A., 1984).

2.3.1 History of Artificial Intelligence

In 1950, Alan Turing suggested that computers could be used to imitate how humans think and act intelligently (Naderisorki et al., 2024, p. 2). That same year, he also shared a way to test if a machine can think in his paper "Computing Machinery and Intelligence." This test was designed, as later described by Haenlein and Kaplan (2019) and Kaul et al. (2020), "to determine whether computers were capable of human intelligence."

In 1956, six years later, people first used the term "Artificial Intelligence" (AI) (Oh, 2024, p. 10). A group of researchers, including Marvin Minsky and John McCarthy, organized a workshop at Dartmouth College in America to invite experts in various areas to design machines in such a manner that they would be similar to human beings (Haenlein & Kaplan, 2019). During this time, John McCarthy defined AI as "the science and engineering of making intelligent machines" (Amisha et al., 2019; Hamet & Tremblay, 2017; Kaul et al., 2020). This event is now known as the beginning of AI research (Haenlein & Kaplan, 2019).

2.3.2 Types of AI

Artificial Intelligence includes two main types: narrow AI (weak AI), which handles specific tasks such as language translation or facial recognition, and general AI (also called strong AI), which tries to think, learn, and solve problems like humans across many different situations (Thea Gasser, 2019, p. 7,8).

Weak AI, as explained by Kumar and Rajesh (2024, p. 210), is designed to perform a single task efficiently, such as tracking the weather, analyzing data, or playing games like chess. They follow set rules and patterns to complete tasks but don't "think" or "understand" as humans do. For instance, a chess program can play the game well by calculating possible moves, but it doesn't truly grasp the game's strategy as a person would.

In contrast, strong AI refers to machines with real intelligence, or at least something very close to it. These systems can learn from what happens, make their own choices, and handle new situations without needing new instructions. For example, Google’s DeepMind uses machine learning to improve itself over time, much like how humans learn. Strong AI systems could be seen as more independent, which raises questions about how they should be treated legally—more like adults than tools (Rex Martinez, 2019).

Super AI, or Artificial Super Intelligence (ASI), is a possible future stage of AI where machines become much smarter than humans in thinking and understanding (Zohuri, 2023). Even if Super AI were possible, it implies that a machine is superior to a human being. It would not only interpret and respond to humans' emotions but be capable of thinking for itself and generating ideas that humans cannot think of due to the inherent limitations of the human brain (Alberto Carrio, 2023, p. 25). To better understand Super AI, it is important to see how it differs from artificial general intelligence (AGI), or AI. Unlike narrow AI, which can only work within a narrow scope of domains or tasks, Super AI could use its intelligence in many domains, from science to the arts. What makes Super AI different is that it could upgrade itself independently, learning exponentially at a speed beyond the confines of man and creating new ideas at a speed beyond man's understanding (Maria Johnsen, 2025, p. 15).

To sum up, dividing AI into strong and weak types is useful for classification and helps people better understand their fundamental differences (Bory et al., 2024). Although still only a theoretical idea, Super AI represents a step beyond strong AI, imagining machines that could completely surpass human intelligence. According to Suchman (2023) and Cugurullo (2024), many studies focus too much on big, theoretical ideas about strong AI—like machines having consciousness—and ignore the simpler, more practical ways weak AI is used in everyday life. Recognizing how weak AI works in daily activities, along with the ideas of strong AI and Super AI, can help give a more complete and balanced view of AI in the modern world.

2.4 Foundations of sports analytics

Sports analytics is the study of sports to understand and improve how players and teams perform (Graça, 2024, p. 3). It helps make better business decisions by collecting, managing, and analyzing sports data to make useful predictions (Michael Cohen & Matthew Sloane, 2015; Sarlis, 2024).

Using analytics in sports is not a new idea (Fujii, 2025, p. 3). The roots of sports analytics can date back to the 19th century. Among the first of many milestones are Henry Chadwick's development of the box score for the baseball game in 1859, quantifying player performance (EBSCO, 2022), and Earnshaw Cook's publication of "Percentage Baseball" in 1964, popularizing the application of statistics for the game of baseball as a means of team strategy improvement and player evaluation (Albert & Bennett, 2001, p. 170). This provided the foundation for using analytics within sports.

While these initial concepts arose some time ago, sports analytics gained significant traction in the early 2000s. A major step in baseball's growth was the 2003 book Moneyball, which showed how the Oakland Athletics, with Billy Beane as their leader, used statistics to choose players and became successful (Pinheiro, 2022, p. 37). Their tale, depicted in the book as well as the movie Moneyball, proved that employing novel statistics such as on-base percentage, rather than conventional statistics, assisted teams in identifying under-valued players and constructing stronger teams. This method shift altered the way many teams approached as well as selected players (EBSCO, 2022).

In short, statistics and data have played a crucial role in shaping sports today, ranging from the first innovations of the box score to more recent methods such as Moneyball philosophy. As data impacts sport more than ever, technologies of the future, such as ML and AI, are pushing development even further. The section below discusses how ML and AI are currently applied within sports, transforming athlete performance, team tactics, and spectator engagement.

2.5 Machine learning and artificial intelligence applications in sports

Over the past 20 years, artificial intelligence (AI) has improved the way people watch and study sports (Chmait & Westerbeek, 2021), from how athletes are trained and managed to how their performances are evaluated. The shift embodies a massive trend towards decisions based on data. As technology continues to evolve, AI increasingly becomes more important to help optimize the performance of athletes, game strategy, and fan interactions. According to Allied Market Research (2024), AI market in sports could grow to around $29.7 billion by 2032, highlighting how much these technologies are shaping the industry.

AI is extensively used today in sports. Computers can capture complete sports data with the help of cameras and wearable sensors. With this data, AI facilitates the preparation of training programs suitable for a particular athlete and the development of more effective game plans. AI visual technology can even detect the ball very accurately, even down to its minute details. Such technology is already assisting referees for a number of sports, such as cricket, tennis, soccer, baseball, and billiards (B. Li & Xu, 2021, p. 55).

One fine example is LaLiga, Spain's top soccer league, which utilizes AI and machine learning for player performance improvement. LaLiga and Microsoft developed a Media Coach system, which gathers about 3.5 million data points per game from 16 camera equipment placed within each stadium. It tracks players, referees, and ball movements in real-time and gives complete reports to 42 clubs. LaLiga even came up with a Goal Probability module that calculates how probable a player is to score depending upon location regarding the goalkeeper, number of defenders, and the player's goal-scoring history. This is how sports are getting brighter, fairer, and more entertaining (*Sport Analytics Leverage AI and ML to Improve the Game*, 2024).

As AI becomes more sophisticated, it will play an even larger part in sports, rendering refereeing more precise, effective, and unbiased. AI will bring greater accuracy through better machine learning and quicker data processing, adding even greater precision to sports and altering how competitions are overseen. The question is whether AI will change sports and how leagues, referees, and organizations will adapt to its increasing function (Anna Martin, 2025).

3 Data Analytics in Football

Football is the world's most popular sport and plays a major role in the global economy. As big data technology has developed, artificial intelligence has been used more and more in the analysis of football data (Wen, 2024). Performance science experts are now a key part of football support teams. They help improve strategies related to athlete performance, talent identification, and medical services (Bartlett & Drust, 2021; Lolli et al., 2025, p. 1).

3.1 Introduction to Data Analytics in Football

Windt et al. (2021) stated that data analytics has become more important for sports scientists to create organized ways to study football in teams (Lolli et al., 2025, p. 1). Rapid technological advancements in data collection, storage, and analysis have transformed football analytics and related fields over the last few decades (Cavus & Biecek, 2022). With the ability to track minute data, coaches are better informed regarding player and team performance, and there are opportunities to observe player movement, formation, and tactics with greater rigor (Seweryn et al., 2024).

People started using data in football a long time ago (Flanagan, 2022, p. 1). In the early 1950s, Charles Reep started recording game data (Anderson & Sally, 2013, p. 15). In the 1990s, sports data companies like Prozone and Opta Sports began to appear. They helped change football by turning it into a game focused more on numbers and analysis (Biermann, 2019, p. 125) and Opta Sports, the first of its kind (Anderson & Sally, 2013, p. 10). In the past several years, football employment data has grown much more advanced. Biermann (2019) even went so far as to state that football has passed an important point of no return in its digitalization. Liverpool FC, for example, depend highly on data to secure their 2019 Champions League, Club World Cup, and 2020 Premier League titles (Flanagan, 2022, p. 1).

There are four types of football data types such as: positional data, physical data, expected goals (xG), and tactical data. Table 4 will show the method of collection, metrics measured of each type of data.

Table 4. Method of collection, metrics measured of each type of data

Type of Data	Method of Collection	Metrics Measured	Reference
Positional Tracking	Global navigation satellite systems, global positioning systems, local radio-based local positioning, and optical-based tracking systems	Player coordinates (x,y)	Beato et al. (2018), Beato & Jamil (2018), Castellano et al. (2014), Hoppe et al. (2018), Leser et al. (2011), Ogris et al. (2012), Rago et al. (2020), Teixeira et al. (2021, 2022)
Physical Performance	Wearable GPS trackers and inertial measurement units (IMUs)	Distance covered, sprint count, high intensity runs, player load, acceleration/deceleration patterns	Losada-Benitez et al. (2023), Lewis et al. (2022)
Expected Goals (xG)	A statistical model that uses event data like shot location, distance to goal, angle, shot type, assist type, and play type.	Probability of the shot resulting in a goal	Hewitt & Karakuş (2023, pp. 3–4, 9)
Tactical Analysis	Historical match data	Team formation, team style, selected players	Beal et al. (2020, p. 1,3)

3.2 Role of AI and Machine Learning in Football Analytics

Machine learning has become an integral tool in football for predicting various performance metrics (Sun, 2023, p. 1). A review by Rico-González et al. (2023) identified that machine learning applications in football have been used in predicting match results, physical and physiological performance, technical and tactical performance, and talent detection. The authors classified 32 papers reviewed in their work into three broad categories: Injury prediction (n = 7), performance prediction (n = 21), and talent detection (n = 5). The classification indicates the versatility of ML in capturing multi-dimensioned aspects of football performance (Rico-González et al., 2023).

With machine learning and deep learning, these algorithms are even more accurate and useful in football (Wang, 2024, p. 2113). One such advanced use is in the use of such systems for tracking the position and movement of players in the field (Forcher et al., 2022). Such systems detect the position of each player as well as how they move to other players automatically through the processing of video input from multiple cameras (Teranishi et al., 2023), which allows coaches and analysts to observe team formations, the extent of players' adherence to position as well as their movement under different situations (Wang, 2024, p. 2114). Wang (2024, p. 2114) concludes that soccer teams can become more competitive by using machine vision to understand player behavior and team strategies through data.

In conclusion, machine learning and deep learning have made football analysis faster and more accurate. With video input and tracking technologies, teams can better understand how players move, follow positions, and play as a team. It helps coaches make better decisions and gives teams an advantage through data insights.

4 Methodology

Chapter 4 introduces the research methodology used in this thesis. It explains how the project was designed, what kind of data was collected, and how it was processed. Section 4.1 describes the approach taken to analyze football players' behavior using unsupervised learning methods. Section 4.2 explains the data sources and how the match videos were collected and processed. Section 4.3 covers the ethical considerations related to data usage and AI training to ensure the system is fair and responsible. Section 4.4 outlines the steps taken to prepare the system for implementation, including data preprocessing, model setup, and evaluation planning.

4.1 Research Design and Approach

This study uses an unsupervised learning approach to analyze football player behavior based on in-game data collected from match videos. Instead of relying on labeled outcomes like match results, the system groups players based on features such as pass count, turnovers, and reaction time. The Elbow Method is used to find the best number of groups, and K-Means is used to spot patterns in how players perform (Fernando & Fianty, 2024, p. 1486). These clusters are then used to generate targeted training recommendations for different player types. This approach helps uncover hidden patterns in the data without requiring labeled examples.

4.2 Data Sources and Collection Methods

The data used in this project was sourced from a small number of football match videos originally provided in the KFL Bundesliga dataset on Kaggle. Although the contest has ended and the dataset is no longer publicly available, some of the test videos have been preserved and are still accessible online. These videos were used to extract player and ball movement information through video analysis techniques. Python libraries such as OpenCV and pandas were used to process the visual data, and the extracted features were used for clustering and pattern analysis. Instead of using large match datasets, the analysis focuses on insights drawn from these limited video samples.

4.3 Ethical Considerations in Data Usage and AI Training

Using data in a fair and responsible way is very important. Since this project uses match videos from public games and professional competitions, it does not involve any private or personal data. The use of these videos respects the original terms of sharing, and proper credit is given to the source. When applying AI models, it is important to avoid bias—for example, not giving unfair advantages to players from certain teams or positions. The use of AI in training or scouting should be open and easy to understand, so coaches and players know how the system works. These AI tools are meant to support human decisions, not replace them. Finally, being clear about how predictions are made helps build trust and ensures technology is used responsibly in football training.

4.4 Preparing for Implementation

After collecting and processing the data from football match videos, the next step is to apply computer vision techniques and machine learning models to analyze the data. This includes using pre-trained object detection models to track player and ball movements, saving this information into CSV files, and then analyzing it using clustering algorithms in Jupyter Notebook.

The methods described in Sections 4.1 to 4.3 are directly used in the implementation stage. For example, K-Means clustering is applied to the player data to find behavioral groups, and the Elbow Method is used to select the best number of clusters. The results are then visualized with heatmaps and running tracks. These outputs are useful for coaches to give feedback or design training based on player performance.

This section sets the foundation for the development process, which is fully described in Chapter 6.

The next chapter will introduce the tools and technologies used to carry out this project. These tools were chosen because they are popular, easy to use, and well-suited for working with images, videos, and machine learning.

5 Technologies and tools

This chapter provides an overview of the main tools and technologies used in the development of this project. Each tool was chosen based on its usefulness, popularity, and ability to support tasks such as programming, data analysis, and image processing.

5.1 Python

Python is a programming language established and maintained by the Python Software Foundation (Trifunovic, 2019). It is also described as both a scripting language and an object-oriented language (Python, n.d.; Rossum & Drake, 2006, p. 3). Python is an easy to learn, powerful programming language. It also has an excellent number of scientific libraries for data science and is hence a growingly relevant and important tool (Rossum & Drake, 2006, p. 3).

Python is still the top choice for scientific computing, data science, and machine learning because it helps work faster and more efficiently using powerful libraries and easy-to-understand commands (Raschka et al., 2020, p. 1). For this project, Python is chosen because it has many useful libraries for machine learning, such as scikit-learn and TensorFlow. These libraries support unsupervised learning methods like clustering, which are used in this work to encourage group player behavior and find hidden patterns in football data.

5.2 Open Source Computer Vision

Open Source Computer Vision (OpenCV), according to Alifya (2020, p. 549), is “an open source computer vision and machine learning software library” aimed at image and video analysis. Initially created by Intel over a decade ago and subsequently shared, OpenCV enables the learning of basic concepts of image processing without the need to study lengthy manuals or textbooks (Culjak et al., 2012).

According to Bradski & Kaehler (2011), OpenCV is mainly built using C and C++, and it works on operating systems like Linux, Windows, and macOS. It also supports other languages such as Python, Ruby, and Matlab through additional interfaces.

OpenCV will be used in this project because it is a free and easy-to-use library for working with images and videos. It helps with many tasks like object detection and tracking, which are important in this project. OpenCV also works with different operating systems and programming languages, including Python, which makes it flexible and useful for real-time image processing.

5.3 You Only Look Once

You Only Look Once (YOLO) is a popular and widely used algorithm, especially known for detecting objects in images. It was first introduced in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi in their well-known paper called “You Only Look Once: Unified, Real-Time Object Detection”. Since then, many updated versions have been released. Some smaller or modified versions like YOLO-LITE also exist. Table 5 will show each version and time release of each version.

Table 5. YOLO model (Alif & Hussain, 2024)

Type	Year	Key Features & Improvements
YOLOv1	2015	First YOLO model. Detect objects by dividing the image into a grid.
YOLOv2	2016	Used anchor boxes and trained better to make results more accurate (Das et al., 2025, p. 205).
YOLOv3	2018	Used a deeper network (Darknet-53) and detected small objects better.
YOLOv4	2020	Faster and more accurate with new backbone (CSPDarknet53) and PANet.
YOLOv5	2020	Developed by Ultralytics. Easy to use, with models of different sizes (small to large).
YOLOv6	2022	Improved speed and accuracy; started using anchor-free detection.
YOLOv7	2022	Supports more tasks (like pose detection); better learning system (E-ELAN).
YOLOv8	2023	New design, anchor-free, better performance.
YOLOv9	2023	More accurate and efficient, but not much public info available.
YOLOv10	2024	Latest version with the best real-time performance so far.

This project will use YOLOv8 because it is fast, accurate, and easy to use. Unlike older versions like YOLOv4 and YOLOv5, YOLOv8 has a new design that does not use anchors, which makes training simpler and results better. Although YOLOv9 and YOLOv10 are newer, they do not have much information available or are harder to use. YOLOv8 is lightweight, works well in real time, and is well supported, so it is a good and practical choice for this project.

5.4 Anaconda

Anaconda, according to Loy (2019), is "a free and open-source software that includes Python" and R, which are often used in data science, machine learning, and AI. It comes with over 250 useful tools and packages that help make installing and using them easier.

Anaconda runs on various operating systems, and thus the programs can be executed without modifying the code. It comes with popular tools such as Jupyter Notebook, Spyder, and RStudio, which are useful for data analysis and visualization. These tools make Anaconda a favorite platform for data science work. It also supports the sharing of code and collaboration with ease using its web-based platform, Anaconda Cloud.

In this project, Anaconda is used to manage Python libraries and environments. It allows users to install all the necessary packages with one command and makes it easy to switch between different Python versions using the conda command. The process of changing the Python version and setting up the environment will be explained in Chapter 6.

5.5 Visual Studio Code

Visual Studio Code is a free and popular code editor developed by Microsoft. It is used to write, edit, and manage code in many programming languages such as Python, JavaScript, and C++. The editor works on Windows, macOS, and Linux. It offers useful features like syntax highlighting, auto-completion, and built-in error checking to make coding easier and faster.

In this project, Visual Studio Code is used as the main code editor for writing and organizing the source files. It provides a user-friendly interface and supports many programming languages, including Python, which is the main language used in this work.

6 Development and implementation

This chapter explains how the project was built step by step. It includes downloading the necessary software, setting up the environment, and modifying the code to process football videos. The chapter has two main parts. The first part (6.1) shows how to install tools like Anaconda, Python, Visual Studio Code, and required packages. It also explains how to clone the project files. The second part (6.2) covers how the code was used to detect players, track ball possession, save match data, and perform analysis with clustering and heatmaps. These steps turn the research plan into a working system.

6.1 Download Software

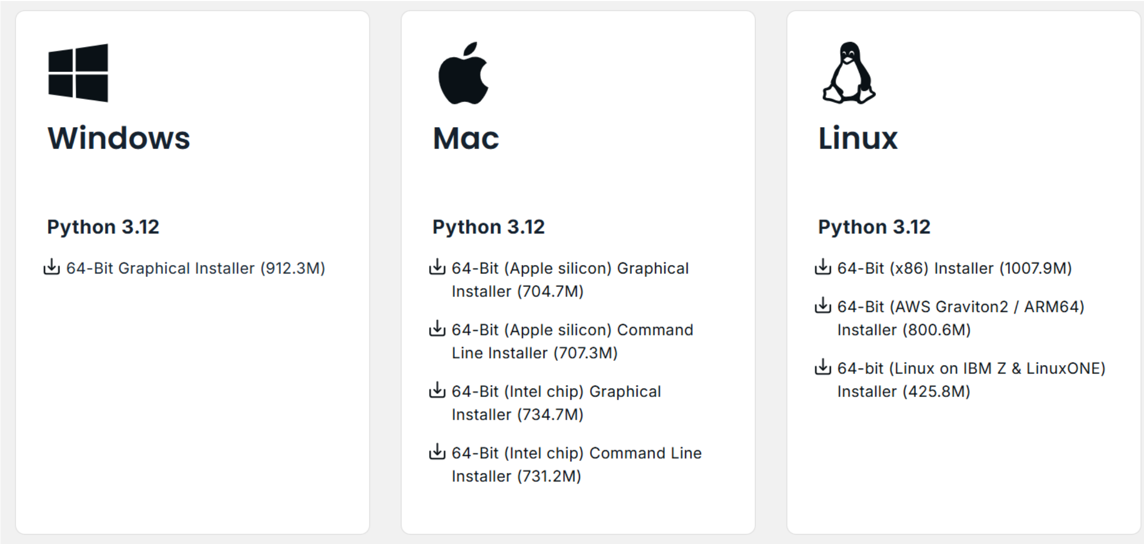
This section explains how to download and install all the software needed for the project. It includes tools that help with programming, running machine learning models, and working with video data. The main tools used in this project include Python for programming, Visual Studio Code for writing the code, and several useful Python libraries. It also explains how to clone the project files to your computer. These steps are important to prepare the system before starting the actual implementation.

6.1.1 Anaconda Navigator

Anaconda can be installed by visiting an official website. The Anaconda website has free download links for Windows, Mac, and Linux.

To download and install Anaconda, the operating system preferred Python version, and the appropriate 64-bit graphical installer must be selected based on the system's processor type (Figure 6). In this case, the Windows 64-bit graphical installer is selected, and the corresponding link is then clicked on the official website to begin the download.

Figure 6: Download version of Anaconda



Anaconda also provides Miniconda, which is a smaller version of Anaconda. It includes only the essentials: Python, Conda, and a few basic packages, allowing users to install only what is needed. This makes Miniconda lightweight and more flexible for custom environments.

After downloading and completing the setup of Anaconda, the Anaconda Navigator can be found by searching for it on the device and then opening the application.

6.1.2 Python

There are several ways to install Python, and one common method is to visit the official Python website and select the preferred version for installation. After the process is complete, Python becomes available on the system. However, as noted in Chapter 6.1.1, Python is already included and installed with Anaconda.

The current version of Python on Anaconda is 3.12 as shown in Figure 6. However, this project will use version 3.11.7. So, the downgrade can be done using Command 1.

Command 1: Downgrade Python version

```
conda create -n myenv python=3.11
```

Then activate the environment using Command 2.

Command 2: Activate the environment

```
conda activate myenv
```

Finally, use Command 3 to check is the downgrade done successfully

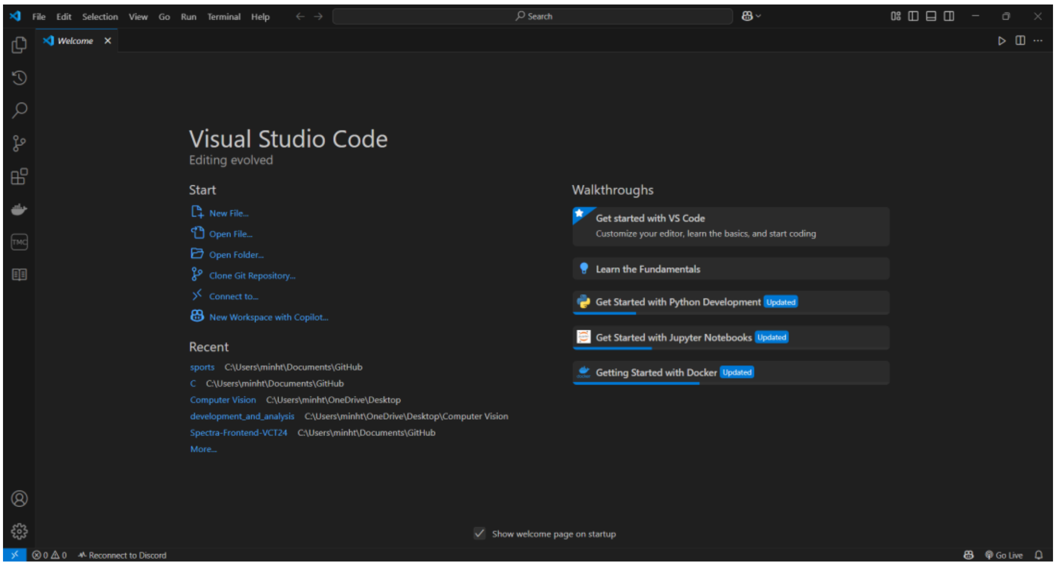
Command 3: Check Python version

```
python --version
```

6.1.3 Visual Studio Code

Visual Studio Code (VS Code) can be installed on the official website, or on Anaconda Navigator. After installation is complete, VS Code can be opened. Figure 7 shows the UI of VS code.

Figure 7: UI of VS Code



Now, let’s move on to install all the packages that are required for the project.

6.1.4 Packages

Appendix 2 shows the list of packages used along with their versions. It is important to install all packages correctly, as many of them depend on each other. Changing the version of one package can automatically change the versions of others, which may lead to errors if some versions are not compatible (Trifunovic, 2019, p. 45). All the Python needed packages can be installed by running the command below, as long as they are listed in the requirements.txt file.

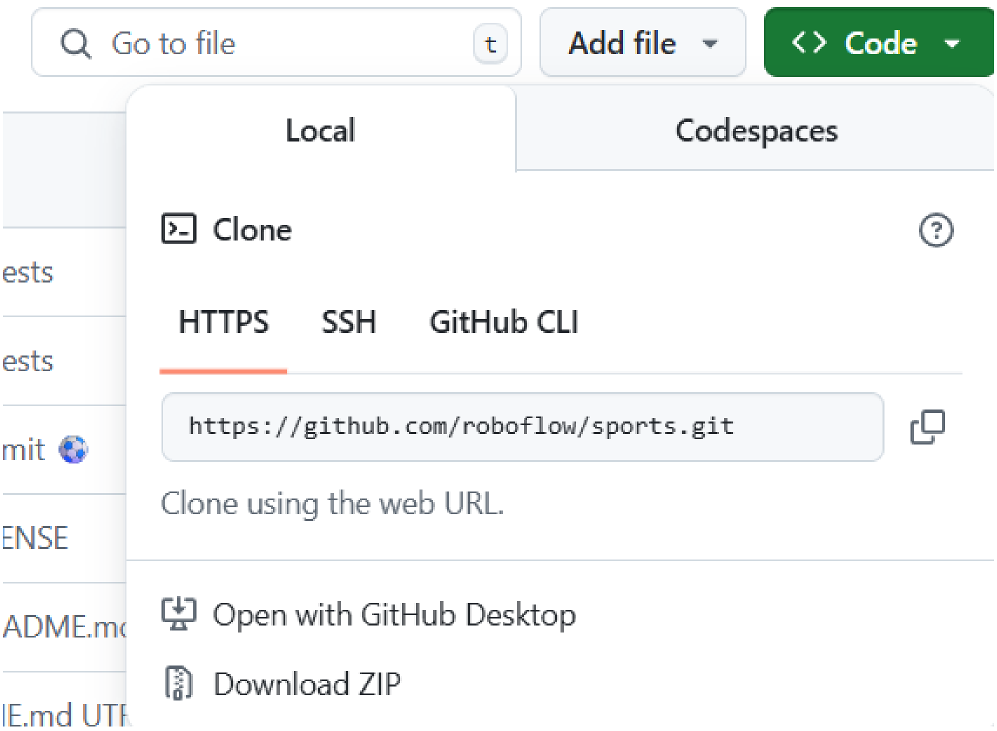
Command 4: Install Required Python Packages from requirements.txt

```
pip install -r requirements.txt
```

6.1.5 Cloning the project

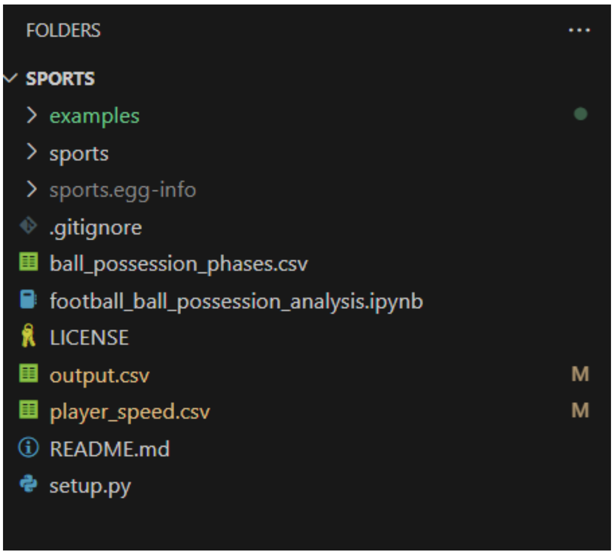
Training a model can take a long time, so a pre-trained player detection model created by Roboflow is used for this project. To clone the project to the device, the "Code" button on the project page should be clicked, which will display options as shown in Figure 8.

Figure 8: Cloning the Roboflow repository



Clicking “Open with GitHub Desktop” launches the GitHub Desktop application. After the setup is complete, the repository is successfully cloned to the local machine. To access all project files in Visual Studio Code, click “Open in Visual Studio Code.” Once opened, the folder structure of the project is displayed in the VS Code Explorer sidebar (Figure 9). This view includes all files and directories in the project, such as Python scripts, and configuration files. The main folder named “SPORTS” contains subfolders like examples and sports, along with key files such as setup.py, and README.md.

Figure 9: Project directory view in Visual Studio Code



Now, let’s move on to the soccer folder by clicking on examples/soccer. Inside this folder, there are several files, including a script named *setup.sh*. This script is important because it helps set up the necessary files for the project. When running, it creates a data folder, if it doesn't already exist, and downloads all the required models and videos from Google Drive using gdown. To run the script, first open a terminal in the same folder, give it permission with command like Command 5.

Command 5. Script to run setup.sh

```
> chmod +x setup.sh

> ./setup.sh
```

Make sure *gdown* is installed. Once completed, all downloaded files will be placed in the data folder. Appendix 3 will show the code inside the file *setup.sh*. After running the command, all the files are stored in *data* folder like in Figure 10.

Figure 10: Project files and folders in the *examples/soccer* directory after running setup

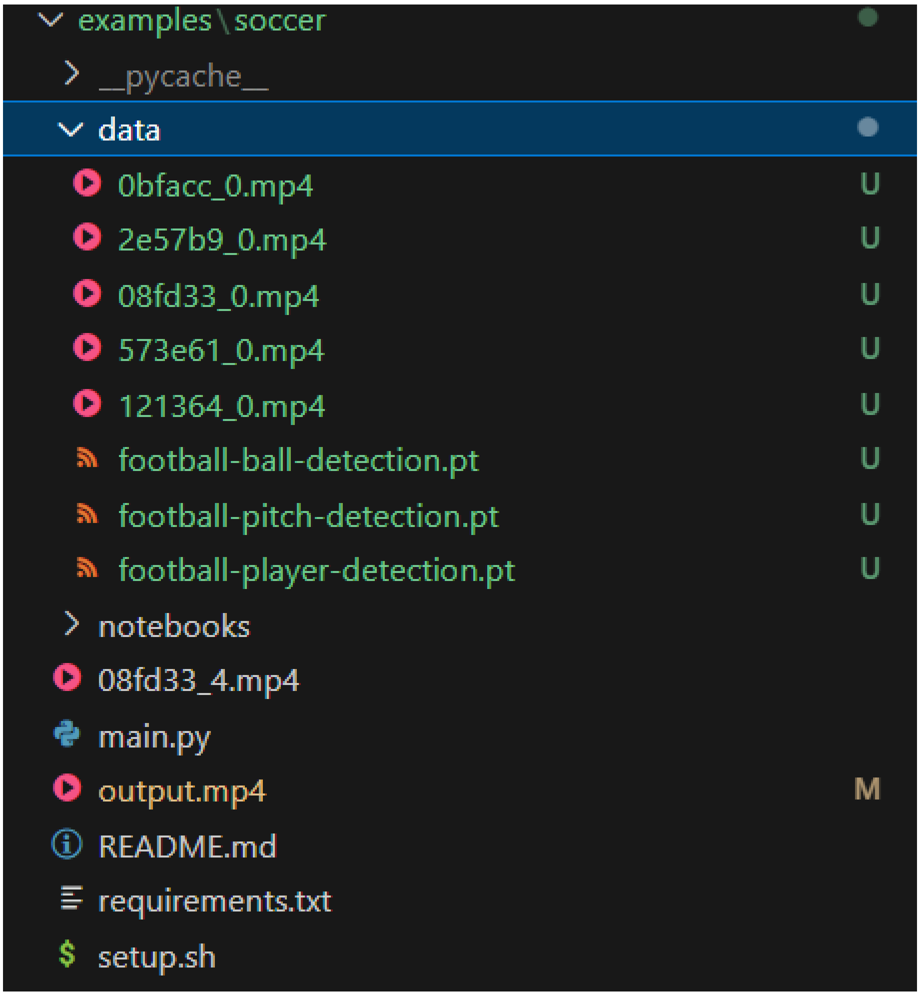


Figure 10 shows five videos in the *data* folder to test if the code works correctly. It also contains three PyTorch model files, which are pre-trained models used to detect objects in football videos. These models are called football-ball-detection.pt, football-pitch-detection.pt, and football-player-detection.pt. Each model is designed to detect a specific element in the match: the ball, the field, or the players. All three models are used in this project to support the object detection process.

After installing all the necessary files and packages, the next step is to move on to the implementation part.

6.2 Implementation

This section explains how the code was prepared and used in the project. It is divided into two parts: the first part describes how the original code was modified to perform computer vision tasks and save data to CSV file, and the second part shows how the processed data was analyzed and visualized using a Jupyter Notebook.

6.2.1 Modifying the code for computer vision tasks

Now, let's begin to code. The author of the repository that has mentioned in chapter 6.1.5 has provided data model to test and some other important file at the main folder. In this thesis, the file main.py in examples/soccer folder will be mentioned most.

In the main.py file, the author of that code has written mostly complete. Unfortunately, when testing by using test video, it showed some errors. For example, run team classification by running run_team_classification function. Run Command 6 as shown below.

Command 6. Run team classification

```
~\anaconda3\python.exe -m examples.soccer.main --source_video_path
examples/soccer/data/08fd33_0.mp4 --target_video_path
examples/soccer/output.mp4 --mode TEAM_CLASSIFICATION
```

--source_video_path is the path to the input video that will be used for processing, --mode tells the program which function to run, in Command 6 it is team classification mode, --target_video_path is the path where the output video will be saved. In command 3, it returns a video called *output.mp4* , which is stored in examples/soccer folder.

When watching the output video, some errors can be seen in the classification. One common issue is that the team ID of a player is not stable and changes between frames. Figure 11 shows an example of this error.

Figure 11: Team ID of the player changes automatically

The issue of where the same player changes team color across frames happens because the TeamClassifier predicts team IDs separately for each frame based on the player's shirt appearance. When lighting or angle changes, the same player might be classified into a different team. This causes the team color to flicker even though the player remains the same. The problem occurs because there is no memory of the player's identity between frames. Program code 1 will show the place that makes it have the problem.

Program code 1: Problem part in team_classification function

```
players_team_id = team_classifier.predict(crops)
```

To fix this, a simple method is used by assigning each player a fixed team ID based on their tracker ID when they first appear and keeping it the same in all following frames. This approach ensures consistent team labeling and prevents the color from changing for the same player.

In team classification, run_all function adds some important improvements compared to run_team_classification. The old method predicts the team of each player based only on their shirt color in every frame, which can cause the team color to change if the lighting or camera angle is different. To fix this, run_all function uses a dictionary to remember each player’s team based on their tracking ID. Once a team is given to a player, it stays the same for the rest of the video. This makes the team labeling more stable and reduces color switching. However, there is still one problem. If two players from different teams stand very close to each other, like one in front of the other, the system might confuse them. This can lead to the wrong team ID being assigned because the visual appearance and tracking may not be clear enough to tell them apart. Program code 2 shows the key improvement in run_all function is the use of a dictionary called player_team_fixed to store the team ID of each player based on their tracking ID, which is the variable tid in Program code 2.

Program code 2: Fixed Team Assignment Using Tracking ID in run_all function

```
player_team_fixed = {}
if tid not in player_team_fixed:
    if class_id == PLAYER_CLASS_ID and i < len(players_team_id):
        player_team_fixed[tid] = players_team_id[i]
    elif class_id == GOALKEEPER_CLASS_ID and i - 
len(players_team_id) < len(goalkeepers_team_id):
        player_team_fixed[tid] = goalkeepers_team_id[i - 
len(players_team_id)]
    else:
```

```
        player_team_fixed[tid] = REFEREE_CLASS_ID
```

In addition to improving team consistency, run_all function also integrates a ball detection function written by the author of the original codebase. This feature detects the ball in each frame and estimates which player is currently in control by checking the position of the ball relative to each player's foot.

The author of this thesis also developed a function called run_all, which is responsible for detecting player IDs and tracking their positions (x, y) on the football field over time. This function processes the video frame by frame and records the exact location of each player at every time point. Program code 3 below demonstrates how the player positions are collected.

Program code 3: Player Position Transformation Using Tracking ID

```
for i, (tid, class_id) in enumerate(zip(tracked_ids,
detections.class_id)):
    foot_xy = np.array([(x1 + x2) / 2, y2])
    transformed_xy =
transformer.transform_points(np.array([foot_xy]))[0]
```

All tracked data such as timestamp, team ID, player ID, x, y and hasBall are recorded into a CSV file for further analysis. This combined functionality allows for more detailed insights into player movements and game events. The pseudocode for the run_all function is provided in Appendix 4. Table 6 describes each feature in the CSV file.

Table 6. Definition of each feature in CSV file

Feature	**Purpose**
time	The timestamp (in seconds) show when the data was recorded in the video.
teamId	The team label (e.g., 0 or 1) that the player belongs to.
playerId	A unique ID for each player, used to track them across frames.
x	X-coordinate of the player on the pitch (scaled)
y	Y-coordinate of the player on the pitch (scaled)
hasBall	True if the player has the ball at that time, otherwise False

With the output.csv, several useful tasks can be done in a Jupyter Notebook. The data can be used to find which players had the ball most often and for how long, helping identify the most active players in the match. Players can also be grouped by their team ID (teamId) to compare activity between teams. By counting how many times a player has hasBall is true, it is possible to estimate how often each player made a pass or controlled the ball. If a player's team ID changes during the video, it can be flagged as a labeling error. All this information is helpful for analyzing the match and preparing features for building machine learning models later.

6.2.2 Project Setup and Data Analysis in Jupyter Notebook

The test video used in the implementation is 0bfacc_0.mp4, located in the examples/soccer/data/ folder. The structure of this folder is shown in Figure 10. To begin, create a Jupyter Notebook file and add Command 7 in a cell to run the run_all function and start the team classification process.

Command 7. Execute run_all Function for Team Classification and Ball Tracking

```
!python examples/soccer/main.py --source_video_path
examples/soccer/data/0bfacc_0.mp4 --target_video_path
examples/soccer/output.mp4 --mode ALL
```

As the command has explained in chapter 6.2.1, it will return output video called output.mp4 and output.csv will be stored in sports folder. Now, let’s check the output.csv file by running a code cell in the Jupyter Notebook. To explore the contents of the generated CSV file, the data can be loaded and previewed using the Pandas library in a Jupyter Notebook. Program Code 4 reads the output.csv file and displays the first five rows, helping users understand the structure and type of data collected during the video analysis.

Program code 4: Show data in output.csv file

```
import pandas as pd
df = pd.read_csv("output.csv")
df.head()
```

After running Program code 4, the output will display a small sample of the data. Table 7 shows the first five rows of the CSV file, which include the time of the frame, team ID, player ID, player position (x, y), and whether the player had possession of the ball (hasBall).

Table 7. Sample data from output.csv file

	time	teamId	playerId	x	y	hasBall
0	0.16	1	4	6311.219727	5685.019043	False
1	0.16	1	3	4595.560547	6307.313477	False
2	0.16	1	2	7254.991699	2911.208496	False
3	0.16	0	8	4473.628418	2883.530762	False
4	0.16	0	16	5561.456543	2797.586182	False

As shown in Table 7, there are six main columns in the CSV file. All of them have been explained previously in Table 6.

Next, the movement path and heatmap of a specific player will be visualized. The player selected for analysis is player ID 9. The code used to generate the movement footage and heatmap is provided in Appendix 5 and Appendix 6, respectively. Figure 12 and Figure 13 show the running track and heatmap of player ID 9 in video 0bfacc_0.mp4.

Figure 12: Running track of player ID 9 in video 0bfacc_0.mp4.

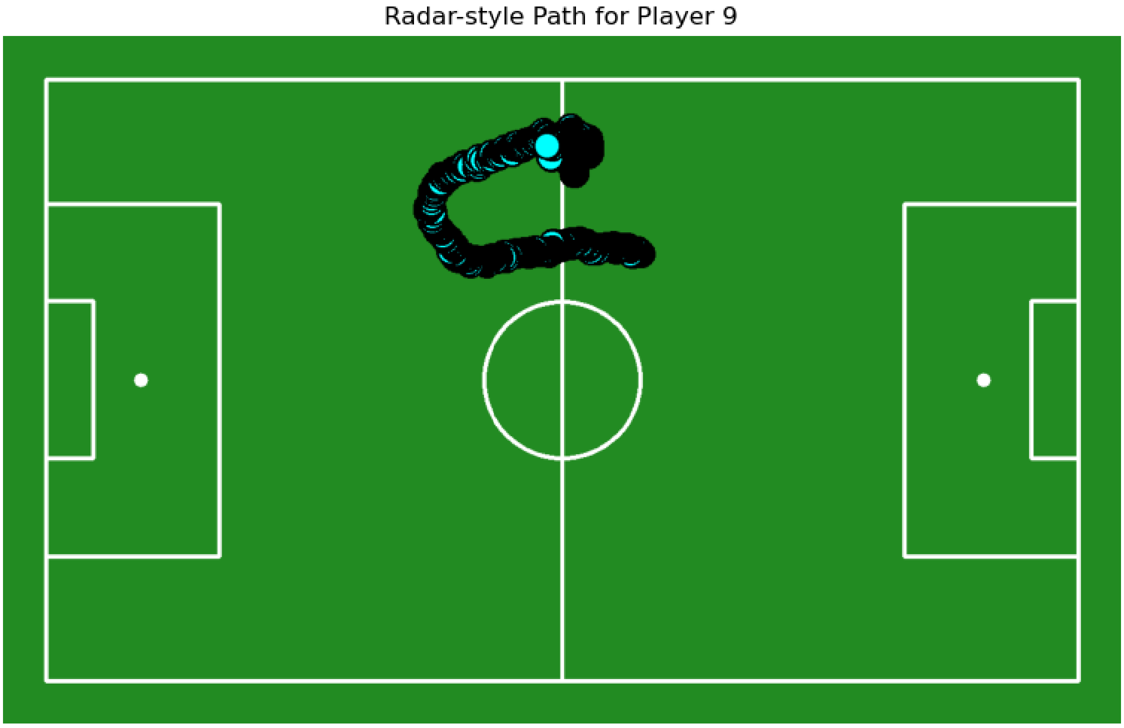


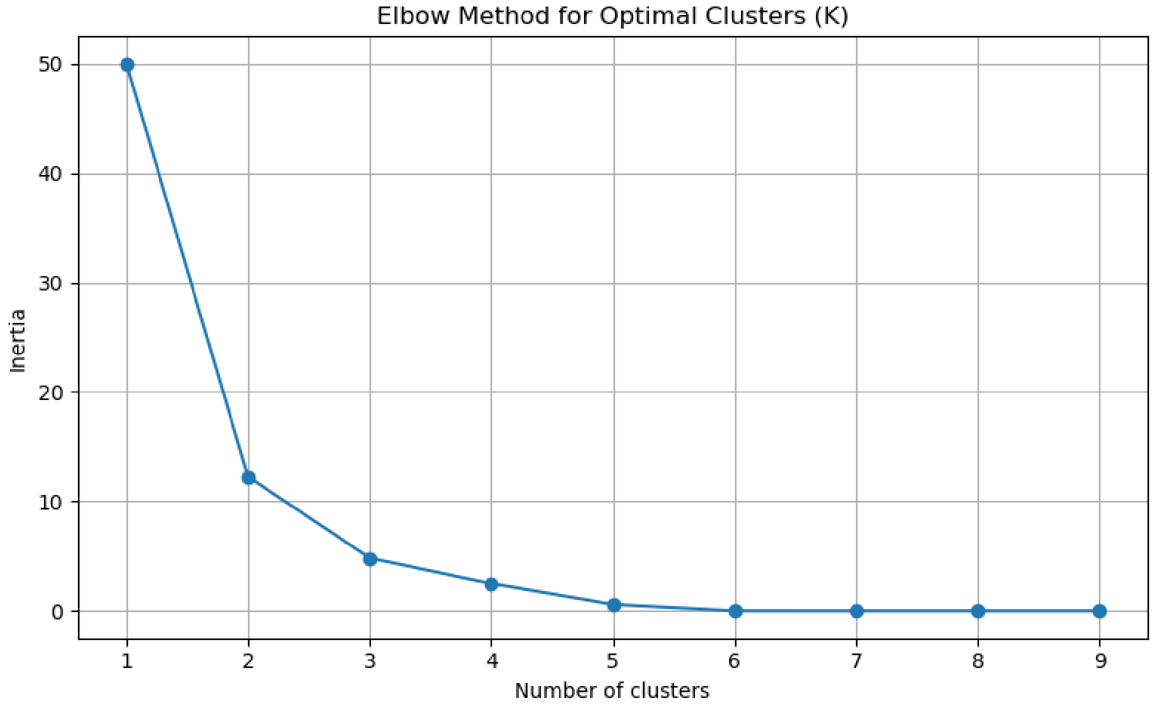
Figure 13: Heat map of player ID 9.

The purpose of these two figures is to show where the player moved on the field and how active they were in different areas. The running track shows the exact path the player took during the match, while the heatmap highlights the zones where the player spent the most time. This information can help coaches understand the player's positioning, movement patterns, and involvement in the game. It is useful for reviewing performance and planning training based on real match data.

In addition to movement analysis, unsupervised learning such as K-Means clustering was used to group players based on their in-game behavior. By combining stats like passes, turnovers, and reaction time, this method helps coaches identify different player types and design training that fits each group. Together with heatmaps, clustering gives a fuller view of both where and how a player performed. Before applying clustering, the Elbow Method was used to find the best number of clusters. This method helps identify the point were increasing the number of clusters no longer gives better separation. The implementation of the Elbow Method is shown in Appendix 7.

Figure 14 shows the output generated after executing code in Appendix 7.

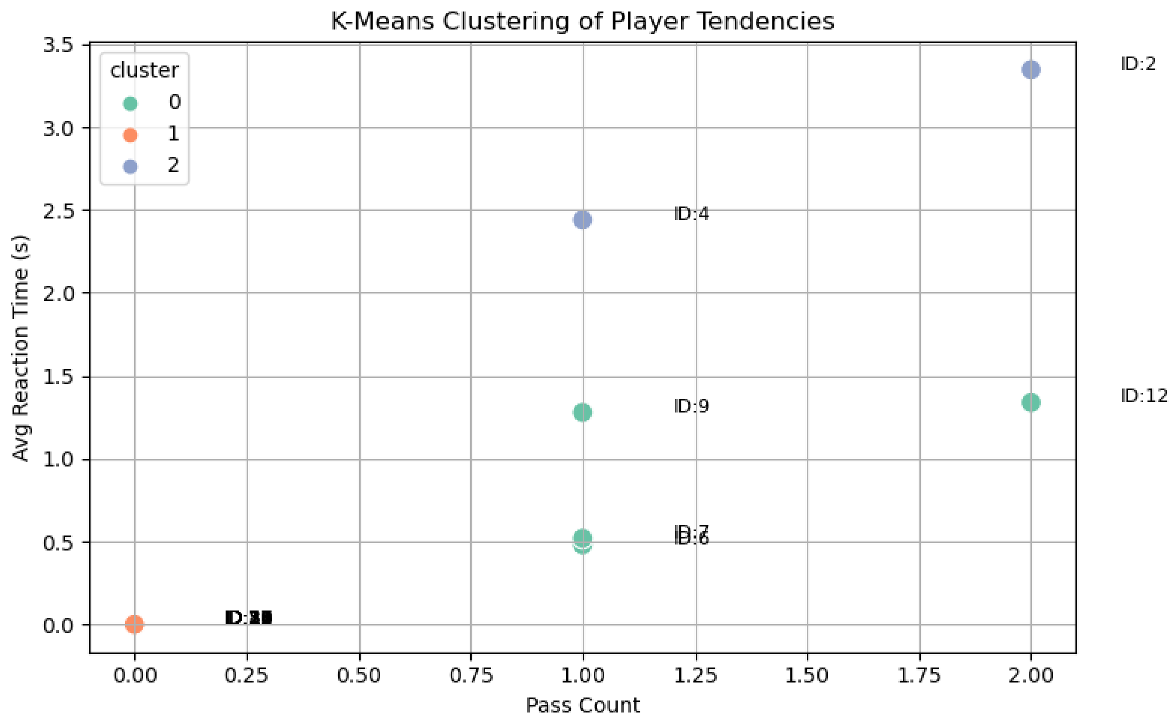
Figure 14: Elbow Method Result Showing Optimal Number of Clusters



As can be seen in figure 14, Elbow Method graph used to determine the optimal number of clusters (K). The curve shows a clear bend at K = 3, indicating that using three clusters provides a good balance between simplicity and clustering performance. This value is selected for the K-Means model in the player grouping analysis.

After selecting K = 3 as the best number of groups using the Elbow Method, the K-Means algorithm was used to divide players based on their passing, turnovers, and reaction time. This clustering helps identify patterns in player behavior. The implementation is shown in Appendix 8, and the result is visualized in Figure 15.

Figure 15: K-Means Clustering of Player Tendencies



As can be seen in Figure 15, K-Means clustering result shows three player groups based on pass count and average reaction time. Each dot represents a player, and the color indicates the cluster they belong to. This helps coaches identify different player styles, such as quick passers, hesitant players, or low-involvement players.

Based on the clustering results, training recommendations were assigned to each group of players. Each cluster represents a different behavioral pattern, such as quick passers or hesitant decision-makers. The function suggest_training maps each cluster to a specific type of training focus. The code implementation is shown below and is used to generate targeted feedback for every player. Full code is provided in Program code 5.

Program code 5: Function to assign training recommendations based on clustering results

```
def suggest_training(cluster):
    if cluster == 0:
        return "Maintain speed, introduce quick-play pressure
drills"
    elif cluster == 1:
        return "Involve more: movement, confidence, build-up
drills"
    elif cluster == 2:
        return "Reduce hesitation: reaction training"
    elif cluster == 3:
        return "Decision focus: passing in pressure zones,
simulated scrimmages"
```

```
    else:
        return "General development"

summary_all["Training Recommendation"] =
summary_all["cluster"].apply(suggest_training)

# Step 8: Show final player recommendations
summary_all[["playerId", "cluster", "Training Recommendation"]]
```

This function gives a training suggestion to each player based on the group (cluster) they are in. The groups are made by looking at things like how long players hold the ball, how many passes they make, and how often they lose the ball. Each group is matched with a type of training that fits their playing style. This helps coaches plan the right drills for each player. Table 8 shows a table of training recommendations based on cluster

Table 8. Player Clusters and Training Recommendations

	playerId	cluster	Training Recommendation
0	2	2	Reduce hesitation: reaction training
1	3	1	Involve more: movement, confidence, build-up d...
2	4	2	Reduce hesitation: reaction training
3	5	1	Involve more: movement, confidence, build-up d...
4	6	0	Maintain speed, introduce quick-play pressure ...
5	7	0	Maintain speed, introduce quick-play pressure ...
6	8	1	Involve more: movement, confidence, build-up d...
7	9	0	Maintain speed, introduce quick-play pressure ...
8	10	1	Involve more: movement, confidence, build-up d...

Table 8 presents the result after running the code that assigns a training recommendation to each player based on their cluster. The suggest_training function maps each cluster to a specific type of training focus. This allows coaches to see which players fall into which group and what type of practice is recommended for each. The table includes the player ID, the assigned cluster, and the suggested training.

7 Result

The system developed in this thesis was able to analyze football match videos using computer vision and predictive analytics. It successfully tracked players, detected ball possessions, and collected key performance data such as passes, turnovers, and movement. Two output files output.csv and radar_output.csv were generated, storing detailed actions and positions of players. These files were used in Jupyter Notebook to create visualizations such as running tracks, player statistics, and heatmaps.

The heatmaps showed where each player spent most of their time on the pitch, helping coaches understand player positioning and activity levels. This kind of visual feedback is important for analyzing tactical roles and movement efficiency. For example, some players covered a wide area, while others remained in a small zone, which may indicate hesitation or limited involvement.

K-Means clustering was also applied to group players based on behavior like reaction time, pass count, and turnovers. The Elbow Method showed that three clusters gave the best result. Each cluster was linked to a training suggestion, helping coaches assign the right drills to different player types. This shows that unsupervised learning can be useful in real match analysis.

However, the project still has some limitations. The test videos were short and did not cover a full match. The system may not perform as well on longer or more complex footage. In some cases, team ID tracking is not fully reliable. For example, when players stand too close together or overlap, the team color detection can switch by mistake. Tracking accuracy also drops in crowded scenes or fast-paced movements.

In the future, the project can be improved by adding a simple user interface (UI) to make it easier for coaches to use. The team ID tracking logic can also be enhanced for better accuracy. Despite these issues, the project shows how computer vision can extract useful information from match videos, and how unsupervised learning like clustering can help identify player behavior patterns.

8 Summary

The research challenges were addressed by a pragmatic approach to video analytics and machine learning. Football matches were analyzed using video footage and fed into a computer vision program to track player movement and ball location. The core performance indicators, like players' positioning, possession time, and players' distance, were computed from the same data. They were used to train algorithms by using machine learning. The result was that predictive analytics were able to pick up patterns and aid players to make better on-field decisions.

Along the journey, some skills were acquired, and learnings were incorporated. Working with machine learning and computer vision and dealing with massive video data helped in gaining a better understanding of how AI systems work and how they are used. Having the correct features and clean data became essential while trying to increase model precision. One also developed a good sense of how useful data-driven tools could be when implemented in sports analytics, not just for coaches and teams but even to aid players in personal development.

This work can also be extended by combining different types of data, such as psychological or biometric data, or even by combining several camera angles to achieve better results. More work includes real-time usage on real-life games, or even by using reinforcement learning to aid players in adapting to dynamic games. More training data and better models will make AI systems more useful as a tool to train and aid in tactical planning in football.

References

A. Ramesh Kumar & M. Rajesh. (2024). *A Comprehensive Study on the Role of Artificial Intelligence in Modern Education System*. *24*(4), 211–214.

Abdi, H., & Williams, L. J. (2010). Principal component analysis. *WIREs Computational Statistics*, *2*(4), 433–459. https://doi.org/10.1002/wics.101

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 207–216. https://doi.org/10.1145/170035.170072

Albert, J., & Bennett, J. (2001). *Curve ball: Baseball, statistics, and the role of chance in the game*. Copernicus.

Alberto Carrio. (2023). *The ethics of AI in sport. Taking athletes' rights and wellbeing seriously*. Barcelona School of Management. https://www.researchgate.net/publication/375278970_The_ethics_of_AI_in_sport_Taking_athletes'_rights_and_wellbeing_seriously

Alif, M. A. R., & Hussain, M. (2024). *YOLOv1 to YOLOv10: A comprehensive review of YOLO variants and their application in the agricultural domain* (arXiv:2406.10139). arXiv. https://doi.org/10.48550/arXiv.2406.10139

Alifya Khan, Pratyusha Trivedi, Karthik Ashok, Prof. Kanchan Dhuri, & Vidyalankar Institute of Technoly, Mumbai. (2020). Natural Language Toolkit based Morphology Linguistics. *International Journal of Engineering Research And*, *V9*(01), IJERTV9IS010320. https://doi.org/10.17577/IJERTV9IS010320

Allied Market Research. (2024). *Artificial Intelligence in Sports Market Size, Share, Competitive Landscape and Trend Analysis Report, by Component, by Deployment Model, by Technology, by Application, by Game Type: Global Opportunity Analysis and Industry Forecast, 2023-2032* [Dataset]. https://www.alliedmarketresearch.com/artificial-intelligence-in-sports-market-A12905

Alonso, E., & Mondragón, E. (2013). Associative Reinforcement Learning—A Proposal to Build Truly Adaptive Agents and Multi-agent Systems: *Proceedings of the 5th International Conference on Agents and Artificial Intelligence*, 141–146. https://doi.org/10.5220/0004175601410146

Amisha, Malik, P., Pathania, M., & Rathaur, V. (2019). Overview of artificial intelligence in medicine. *Journal of Family Medicine and Primary Care*, *8*(7), 2328. https://doi.org/10.4103/jfmpc.jfmpc_440_19

Anderson, C., & Sally, D. (2013). *The Numbers Game: Why Everything You Know About Football is Wrong* (illustrated ed.). Penguin UK.

Anna Martin. (2025, March 11). *The landscape of athletics is influenced by the addition of artificial intelligence*. https://www.thejustice.org/article/2025/03/the-landscape-of-athletics-is-influenced-by-the-addition-of-artificial-intelligence-brandeis

Bank, D., Koenigstein, N., & Giryes, R. (2020). *Autoencoders* (Version 2). arXiv. https://doi.org/10.48550/ARXIV.2003.05991

Bartlett, J. D., & Drust, B. (2021). A framework for effective knowledge translation and performance delivery of Sport Scientists in professional sport. *European Journal of Sport Science*, *21*(11), 1579–1587. https://doi.org/10.1080/17461391.2020.1842511

Beal, R., Chalkiadakis, G., Norman, T. J., & Ramchurn, S. D. (2020). *Optimising Game Tactics for Football* (arXiv:2003.10294). arXiv. https://doi.org/10.48550/arXiv.2003.10294

Beato, M., Coratella, G., Stiff, A., & Iacono, A. D. (2018). The Validity and Between-Unit Variability of GNSS Units (STATSports Apex 10 and 18 Hz) for Measuring Distance and Peak Speed in Team Sports. *Frontiers in Physiology*, *9*, 1288. https://doi.org/10.3389/fphys.2018.01288

Beato, M., & Jamil, M. (2018). Intra-system reliability of SICS: Video-tracking system (Digital.Stadium®) for performance analysis in soccer. *The Journal of Sports Medicine and Physical Fitness*, *58*(6). https://doi.org/10.23736/S0022-4707.17.07267-X

Belkina, A. C., Ciccolella, C. O., Anno, R., Halpert, R., Spidlen, J., & Snyder-Cappione, J. E. (2019). Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature Communications*, *10*(1), 5415. https://doi.org/10.1038/s41467-019-13055-y

Biermann, C. (2019). *Football Hackers: The Science and Art of a Data Revolution*. Kings Road Publishing.

Bory, P., Natale, S., & Katzenbach, C. (2024). Strong and weak AI narratives: An analytical framework. *AI & SOCIETY*. https://doi.org/10.1007/s00146-024-02087-8

Bradski, G. R., & Kaehler, A. (2011). *Learning OpenCV: Computer vision with the OpenCV library* (1. ed., [Nachdr.]). O'Reilly.

Byeon, H., Verma, G., Sen, A. C., & Haider, S. A. (2023). *SUPERVISED LEARNING ALGORITHMS – CLASSIFICATION AND REGRESSION ALGORITHMS*. Xoffencer International Publication. https://www.researchgate.net/publication/374587046_SUPERVISED_LEARNING_ALGORITHMS_-_CLASSIFICATION_AND_REGRESSION_ALGORITHMS

Caron, M., Bojanowski, P., Joulin, A., & Douze, M. (2018). *Deep Clustering for Unsupervised Learning of Visual Features* (Version 2). arXiv. https://doi.org/10.48550/ARXIV.1807.05520

Castellano, J., Alvarez-Pastor, D., & Bradley, P. S. (2014). Evaluation of Research Using Computerised Tracking Systems (Amisco® and Prozone®) to Analyse Physical Performance in Elite Soccer: A Systematic Review. *Sports Medicine*, *44*(5), 701–712. https://doi.org/10.1007/s40279-014-0144-3

Cavus, M., & Biecek, P. (2022). Explainable expected goal models for performance analysis in football analytics. *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*, 1–9. https://doi.org/10.1109/DSAA54385.2022.10032440

Cha, Y.-J., Chen, J. G., & Büyüköztürk, O. (2017). Output-only computer vision based damage detection using phase-based optical flow and unscented Kalman filters. *Engineering Structures*, *132*, 300–313. https://doi.org/10.1016/j.engstruct.2016.11.038

Chapelle, O., Schölkopf, B., & Zien, A. (Eds.). (2006). *Semi-supervised learning*. MIT Press.

Chapelle, O., Zien, A., & Schölkopf, B. (Eds.). (2006). *Semi-supervised learning*. MIT Press.

Chen, M. (2024, October 29). *What Is Semi-Supervised Learning?* What Is Semi-Supervised Learning? https://www.oracle.com/fi/artificial-intelligence/machine-learning/semi-supervised-learning/

Chmait, N., & Westerbeek, H. (2021). Artificial Intelligence and Machine Learning in Sport Research: An Introduction for Non-data Scientists. *Frontiers in Sports and Active Living*, *3*, 682287. https://doi.org/10.3389/fspor.2021.682287

Cugurullo, F. (2024). The obscure politics of artificial intelligence: A Marxian socio-technical critique of the AI alignment problem thesis. *AI and Ethics*. https://doi.org/10.1007/s43681-024-00476-9

Culjak, I., Abram, D., Pribanic, T., Dzapo, H., & Cifrek, M. (2012). A brief introduction to OpenCV. *2012 Proceedings of the 35th International Convention MIPRO*, 1725–1730.

Cunningham, P., Cord, M., & Delany, S. J. (2008). Supervised Learning. In M. Cord & P. Cunningham (Eds.), *Machine Learning Techniques for Multimedia* (pp. 21–49). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-75171-7_2

Das, A., Pathan, F., Jim, J. R., Kabir, M. M., & Mridha, M. F. (2025). Deep learning-based classification, detection, and segmentation of tomato leaf diseases: A state-of-the-art review. *Artificial Intelligence in Agriculture*, *15*(2), 192–220. https://doi.org/10.1016/j.aiia.2025.02.006

EBSCO. (2022). *Sports analytics*. https://www.ebsco.com/research-starters/sports-and-leisure/sports-analytics

Eckhardt, R. (1987). Stan Ulam, John von Neumann, and the Monte Carlo Method. *Los Alamos Science (Special Issue)*, 131–141.

El-Ghany, S. A., Elmogy, M., A. Mahmood, M., & Abd El-Aziz, A. A. (2024). A Robust Tuberculosis Diagnosis Using Chest X-Rays Based on a Hybrid Vision Transformer and Principal Component Analysis. *Diagnostics*, *14*(23), 2736. https://doi.org/10.3390/diagnostics14232736

Feng, P., Bi, Z., Wen, Y., Pan, X., Peng, B., Liu, M., Xu, J., Chen, K., Liu, J., Yin, C. H., Zhang, S., Wang, J., Niu, Q., Li, M., & Wang, T. (2024). *Deep Learning and Machine Learning, Advancing Big Data Analytics and Management: Unveiling AI's Potential Through Tools, Techniques, and Applications* (arXiv:2410.01268). arXiv. https://doi.org/10.48550/arXiv.2410.01268

Feofanov, V., Tiomoko, M., & Virmaux, A. (2023). *Random matrix analysis to balance between supervised and unsupervised learning under the low density separation assumption*. *202*, 10008–10033. https://proceedings.mlr.press/v202/feofanov23a.html

Fernando, L., & Fianty, M. I. (2024). Optimizing Motorcycle Sales: Enhancing Customer Segmentation with K-Means Clustering and Data Mining Techniques. *Journal of Information Systems and Informatics*, *6*(3), 1484–1498. https://doi.org/10.51519/journalisi.v6i3.799

Flanagan, C. A. (2022). Stats Entertainment: The Legal and Regulatory Issues Arising from the Data Analytics Movement in Association Football. Part One: The Development of Data Analytics and Property Rights in Data. *Entertainment and Sports Law Journal*, *19*(1). https://doi.org/10.16997/eslj.1163

Forcher, L., Altmann, S., Forcher, L., Jekauc, D., & Kempe, M. (2022). The use of player tracking data to analyze defensive play in professional soccer—A scoping review. *International Journal of Sports Science & Coaching*, *17*(6), 1567–1592. https://doi.org/10.1177/17479541221075734

Fujii, K. (2025). *Machine Learning in Sports: Open Approach for Next Play Analytics* (1st ed. 2025). Springer Nature Singapore. https://doi.org/10.1007/978-981-96-1445-5

Gao, K., Mei, G., Piccialli, F., Cuomo, S., Tu, J., & Huo, Z. (2020). Julia language in machine learning: Algorithms, applications, and open issues. *Computer Science Review*, *37*, 100254. https://doi.org/10.1016/j.cosrev.2020.100254

Graça, M. M. S. (2024). *Exploring MotoGP Race Dynamics: A Machine Learning Study Using Fictional Data* [Master's Thesis, Instituto Superior Técnico]. https://www.proquest.com/docview/3143984481

Grys, B. T., Lo, D. S., Sahin, N., Kraus, O. Z., Morris, Q., Boone, C., & Andrews, B. J. (2017). Machine learning and computer vision approaches for phenotypic profiling. *Journal of Cell Biology*, *216*(1), 65–71. https://doi.org/10.1083/jcb.201610026

Haenlein, M., & Kaplan, A. (2019). A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence. *California Management Review*, *61*(4), 5–14. https://doi.org/10.1177/0008125619864925

Hamet, P., & Tremblay, J. (2017). Artificial intelligence in medicine. *Metabolism*, *69*, S36–S40. https://doi.org/10.1016/j.metabol.2017.01.011

Heaton, J. (2018). Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning: The MIT Press, 2016, 800 pp, ISBN: 0262035618. *Genetic Programming and Evolvable Machines*, *19*(1–2), 305–307. https://doi.org/10.1007/s10710-017-9314-z

Hewitt, J. H., & Karakuş, O. (2023). *A Machine Learning Approach for Player and Position Adjusted Expected Goals in Football (Soccer)*. https://doi.org/10.48550/ARXIV.2301.13052

Hoppe, M. W., Baumgart, C., Polglaze, T., & Freiwald, J. (2018). Validity and reliability of GPS and LPS for measuring distances covered and sprint mechanical properties in team sports. *PLOS ONE*, *13*(2), e0192708. https://doi.org/10.1371/journal.pone.0192708

Howard, R. A. (1966). Dynamic Programming. *Management Science*, *12*(5), 317–348. https://doi.org/10.1287/mnsc.12.5.317

IBM. (2021, September 23). *What is unsupervised learning?* What Is Unsupervised Learning? https://www.ibm.com/think/topics/unsupervised-learning

Ivanov, A. (2025). A COMPREHENSIVE REVIEW OF THE HISTORY AND METHODS OF COMPUTER VISION. *Telecommunication and Information Technologies*, *86*(1). https://doi.org/10.31673/2412-4338.2025.011767

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An Introduction to Statistical Learning: With Applications in Python*. Springer International Publishing. https://doi.org/10.1007/978-3-031-38747-0

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, *4*, 237–285. https://doi.org/10.1613/jair.301

Kaul, V., Enslin, S., & Gross, S. A. (2020). History of artificial intelligence in medicine. *Gastrointestinal Endoscopy*, *92*(4), 807–812. https://doi.org/10.1016/j.gie.2020.06.040

Leser, R., Baca, A., & Ogris, G. (2011). Local Positioning Systems in (Game) Sports. *Sensors*, *11*(10), 9778–9797. https://doi.org/10.3390/s111009778

Lewis, G., Towlson, C., Roversi, P., Domogalla, C., Herrington, L., & Barrett, S. (2022). Quantifying volume and high-speed technical actions of professional soccer players using foot-mounted inertial measurement units. *PLOS ONE*, *17*(2), e0263518. https://doi.org/10.1371/journal.pone.0263518

Li, B., & Xu, X. (2021). Application of Artificial Intelligence in Basketball Sport. *Journal of Education, Health and Sport*, *11*(7), 54–67. https://doi.org/10.12775/JEHS.2021.11.07.005

Li, Y. (2018). *Deep Reinforcement Learning* (Version 1). arXiv. https://doi.org/10.48550/ARXIV.1810.06339

Lolli, L., Bauer, P., Irving, C., Bonanno, D., Höner, O., Gregson, W., & Di Salvo, V. (2025). Data analytics in the football industry: A survey investigating operational frameworks and practices in professional clubs and national federations from around the world. *Science and Medicine in Football*, *9*(2), 189–198. https://doi.org/10.1080/24733938.2024.2341837

Losada-Benitez, J. A., Nuñez-Sánchez, F. J., & Barbero-Álvarez, J. C. (2023). Quantifying technical load and physical activity in professional soccer players during pre-season matches with IMU technology. *Frontiers in Physiology*, *14*, 1274171. https://doi.org/10.3389/fphys.2023.1274171

Loy, J. (2019). *Neural network projects with Python: The ultimate guide to using Python to explore the true power of neural networks through six projects*. Packt Publishing.

Ma, H. (2022). Design of Chinese Linguistics Teaching System Based on K-means Clustering Algorithm. In Z. Xu, S. Alrabaee, O. Loyola-González, X. Zhang, N. D. W. Cahyani, & N. H. Ab Rahman (Eds.), *Cyber Security Intelligence and Analytics* (Vol. 123, pp. 424–431). Springer International Publishing. https://doi.org/10.1007/978-3-030-96908-0_53

Mahesh, B. (2020). Machine Learning Algorithms—A Review. *International Journal of Science and Research (IJSR)*, *9*(1), 381–386. https://doi.org/10.21275/ART20203995

Maji, A., Velaga, N. R., & Urie, Y. (2018). Hierarchical clustering analysis framework of mutually exclusive crash causation parameters for regional road safety strategies. *International Journal of Injury Control and Safety Promotion*, *25*(3), 257–271. https://doi.org/10.1080/17457300.2017.1416485

Maria Johnsen. (2025). *Super AI*. Independently published. https://books.google.fi/books?hl=en&lr=&id=MzVDEQAAQBAJ&oi=fnd&pg=PA4&dq=what+is+super+AI&ots=0h7hHoAZih&sig=Mchtakisf0ci9DcVxqDG_ND9JIk&redir_esc=y#v=onepage&q=what%20is%20super%20AI&f=false

Michael Cohen & Matthew Sloane. (2015). *Predictive Modeling and Statistical Analysis in sports* [Pomona College]. https://cs.pomona.edu/classes/cs190/archive/fa2022/thesis_examples/Cohen-Sloane.15.pdf

Michelucci, U. (2022). *An Introduction to Autoencoders* (arXiv:2201.03898). arXiv. https://doi.org/10.48550/arXiv.2201.03898

Moin, T. S. (2023). *Overview of Computer Vision*. https://doi.org/10.13140/RG.2.2.13989.68327

Murel, J., & Kavlakoglu, E. (2024, March 25). *What is reinforcement learning?* What Is Reinforcement Learning? https://www.ibm.com/think/topics/reinforcement-learning

Naderisorki, M., Rezapour, M., Psychiatry and Behavioral Sciences Research Center, Addiction Institute, Mazandaran University of Medical Sciences, Sari, Iran., Naderi Soorki, M., & Research Laboratory of Intelligent Wireless Networks, Faculty of Engineering, Shahid Chamran University of Ahvaz, Ahvaz, Iran. (2024). Investigating the Application of Artificial Intelligence in the Pediatric Oncology. *Journal of Pediatrics Review*, *12*(1), 1–4. https://doi.org/10.32598/jpr.12.1.786.3

Naeem, S., Ali, A., Anam, S., & Ahmed, M. M. (2023). An Unsupervised Machine Learning Algorithms: Comprehensive Review. *International Journal of Computing and Digital Systems*, *13*(1), 911–921. https://doi.org/10.12785/ijcds/130172

Nakayama, L. F., Kras, A., Ribeiro, L. Z., Malerbi, F. K., Mendonça, L. S., Celi, L. A., Regatieri, C. V. S., & Waheed, N. K. (2022). Global disparity bias in ophthalmology artificial intelligence applications. *BMJ Health & Care Informatics*, *29*(1), e100470. https://doi.org/10.1136/bmjhci-2021-100470

Nasteski, V. (2017). An overview of the supervised machine learning methods. *HORIZONS.B*, *4*, 51–62. https://doi.org/10.20544/HORIZONS.B.04.1.17.P05

Ogris, G., Leser, R., Horsak, B., Kornfeind, P., Heller, M., & Baca, A. (2012). Accuracy of the LPM tracking system considering dynamic position changes. *Journal of Sports Sciences*, *30*(14), 1503–1511. https://doi.org/10.1080/02640414.2012.712712

Oh, Y. (2024). *Machine Learning-Based Prediction on Heat Transfer Enhancement in Complex Thermal Systems* [Doctoral Dissertation, Rutgers, The State University of New Jersey, School of Graduate Studies]. https://www.proquest.com/docview/3128383054

Patel, A. K., & Chatterjee, S. (2016). Computer vision-based limestone rock-type classification using probabilistic neural network. *Geoscience Frontiers*, *7*(1), 53–60. https://doi.org/10.1016/j.gsf.2014.10.005

Pinheiro, R. (2022). *Labor Market Efficiency and Team Transactions in Major League Baseball*. https://doi.org/10.7302/4639

PK, F. A. (1984). *What is artificial intelligence?* https://www.researchgate.net/profile/Janani-Nakkeeran/publication/389004322_Artificial_Intelligence_-_The_Next_Revolution/links/67af8e1e645ef274a48112d9/Artificial-Intelligence-The-Next-Revolution.pdf#page=76

Puterman, M. L., & Patrick, J. (2017). Dynamic Programming. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning and Data Mining* (pp. 377–388). Springer US. https://doi.org/10.1007/978-1-4899-7687-1_77

Python. (n.d.). *What is Python? Executive Summary*. https://www.python.org/doc/essays/blurb/

Qiankun Zhao & Sourav S. Bhowmick. (2003). *Association Rule Mining: A Survey* [Technical]. Nanyang Technological University, Singapore. https://personal.ntu.edu.sg/assourav/Unpublished/UP-ARMSurvey.pdf

Rago, V., Brito, J., Figueiredo, P., Costa, J., Barreira, D., Krustrup, P., & Rebelo, A. (2020). Methods to collect and interpret external training load using microtechnology incorporating GPS in professional football: A systematic review. *Research in Sports Medicine*, *28*(3), 437–458. https://doi.org/10.1080/15438627.2019.1686703

Raschka, S., Patterson, J., & Nolet, C. (2020). *Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence* (arXiv:2002.04803). arXiv. https://doi.org/10.48550/arXiv.2002.04803

Rex Martinez. (2019). *Artificial Intelligence: Distinguishing between Types & Definitions*. 1027–1028.

Richard S. Sutton. (1990). Integrated Modeling and Control Based on Reinforcement Learning and Dynamic Programming. *Advances in Neural Information Processing Systems 3*, 471–478. https://proceedings.neurips.cc/paper_files/paper/1990/file/d9fc5b73a8d78fad3d6dffe419384e70-Paper.pdf

Rico-González, M., Pino-Ortega, J., Méndez, A., Clemente, F., & Baca, A. (2023). Machine learning application in soccer: A systematic review. *Biology of Sport*, *40*(1), 249–263. https://doi.org/10.5114/biolsport.2023.112970

Rodriguez, M. Z., Comin, C. H., Casanova, D., Bruno, O. M., Amancio, D. R., Costa, L. D. F., & Rodrigues, F. A. (2019). Clustering algorithms: A comparative approach. *PLOS ONE*, *14*(1), e0210236. https://doi.org/10.1371/journal.pone.0210236

Rossum, G. van, & Drake, F. L. (2006). *An introduction to Python: Release 2.5* (2. print). Network Theory Limited.

Saharawat, V. (2023, January 16). *What is Unsupervised Learning?* What Is Unsupervised Learning? https://pwskills.com/blog/unsupervised-learning/

Salian, I. (2018, August 2). *SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?* SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning? https://blogs.nvidia.com/blog/supervised-unsupervised-learning/

Sammut, C., & Webb, G. I. (Eds.). (2010). *Encyclopedia of Machine Learning*. Springer US. https://doi.org/10.1007/978-0-387-30164-8

Sangeeta, Prof. (Dr.), Rai, Mr. V., Pathak, Dr. N., & Sharma, Dr. N. (2024). SEARCH STRATEGIES: INFORMED SEARCH STRATEGIES. In Dr. N. Pathak & Dr. N. Sharma (Eds.), *Artificial Intelligence and their Applications* (First, pp. 47–62). Iterative International Publishers, Selfypage Developers Pvt Ltd. https://doi.org/10.58532/nbennurch55

Sarlis, V. (2024). *Data Science for Sports Analytics* [International Hellenic University, School of Science & Technology]. https://www.researchgate.net/profile/Vangelis-Sarlis/publication/382048910_Data_Science_For_Sports_Analytics/links/66899da60a25e27fbc2f88f6/Data-Science-For-Sports-Analytics.pdf

Seweryn, K., Chęć, G., Łukasik, S., & Wróblewska, A. (2024). *Improving Object Detection Quality in Football Through Super-Resolution Techniques* (arXiv:2402.00163). arXiv. https://doi.org/10.48550/arXiv.2402.00163

Smadi, S. M. (2017). *Detection of Online Phishing Email using Dynamic Evolving Neural Network Based on Reinforcement Learning* [Doctoral thesis, Northumbria University]. https://core.ac.uk/download/pdf/196576912.pdf

Smith-Vickery, N. J. (2023). *Frozen in Fear: Machine Learning Unveils Sex Differences in Rats' Freezing Postures and Defensive Reactions*. University of California.

Sorzano, C. O. S., Vargas, J., & Montano, A. P. (2014). *A survey of dimensionality reduction techniques* (Version 1). arXiv. https://doi.org/10.48550/ARXIV.1403.2877

Sport analytics leverage AI and ML to improve the game. (2024, April 8). https://www.cio.com/article/2081595/sport-analytics-leverage-ai-and-ml-to-improve-the-game.html

Suchman, L. (2023). The uncontroversial 'thingness' of AI. *Big Data & Society*, *10*(2), 20539517231206794. https://doi.org/10.1177/20539517231206794

Sun, D. (2023). An Overview of Machine Learning Applications in the Football Field. *Applied and Computational Engineering*, *8*(1), 318–322. https://doi.org/10.54254/2755-2721/8/20230178

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*(1), 9–44. https://doi.org/10.1007/BF00115009

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT Press.

Szeliski, R. (2022). *Computer vision: Algorithms and applications* (Second edition). Springer.

Szepesvári, C. (2010). *Algorithms for Reinforcement Learning*. Springer International Publishing. https://doi.org/10.1007/978-3-031-01551-9

Teixeira, J. E., Forte, P., Ferraz, R., Branquinho, L., Silva, A. J., Monteiro, A. M., & Barbosa, T. M. (2022). Integrating physical and tactical factors in football using positional data: A systematic review. *PeerJ*, *10*, e14381. https://doi.org/10.7717/peerj.14381

Teixeira, J. E., Forte, P., Ferraz, R., Leal, M., Ribeiro, J., Silva, A. J., Barbosa, T. M., & Monteiro, A. M. (2021). Monitoring Accumulated Training and Match Load in Football: A Systematic Review. *International Journal of Environmental Research and Public Health*, *18*(8), 3906. https://doi.org/10.3390/ijerph18083906

Teranishi, M., Tsutsui, K., Takeda, K., & Fujii, K. (2023). Evaluation of Creating Scoring Opportunities for Teammates in Soccer via Trajectory Prediction. In U. Brefeld, J. Davis, J. Van Haaren, & A. Zimmermann (Eds.), *Machine Learning and Data Mining for Sports Analytics* (Vol. 1783, pp. 53–73). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-27527-2_5

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, *38*(3), 58–68. https://doi.org/10.1145/203330.203343

Thea Gasser. (2019). *Bias – A lurking danger that can convert algorithmic systems into discriminatory entities* [HAMK Häme University of Applied Sciences]. https://urn.fi/URN:NBN:fi:amk-201905027218

Thomas, G., Gade, R., Moeslund, T. B., Carr, P., & Hilton, A. (2017). Computer vision for sports: Current applications and research topics. *Computer Vision and Image Understanding*, *159*, 3–18. https://doi.org/10.1016/j.cviu.2017.04.011

Trifunovic, D. (2019). *NLP-based chatbot for HAMK* [HAMK]. https://www.theseus.fi/bitstream/handle/10024/167435/Thesis_Dejan_Trifunovic.pdf?isAllowed=y&sequence=2

Tyagi, K., Rane, C., Sriram, R., & Manry, M. (2022). Unsupervised learning. In *Artificial Intelligence and Machine Learning for EDGE Computing* (pp. 33–52). Elsevier. https://doi.org/10.1016/B978-0-12-824054-0.00012-5

van der Maaten, L., & Hinton, G. (2008). Visualizing Data Using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605.

Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, *109*(2), 373–440. https://doi.org/10.1007/s10994-019-05855-6

Wang, J. (2024). Analyzing Players' Tactical Positions and Movement Trajectories in Soccer Matches Using Machine Vision Algorithms. *Journal of Electrical Systems*, *20*(6s), 2113–2123. https://doi.org/10.52783/jes.3126

Wen, Q. (2024). Advancements of Football Data Analysis Based on Machine Learning Algorithms: *Proceedings of the 1st International Conference on Engineering Management, Information Technology and Intelligence*, 67–71. https://doi.org/10.5220/0012902400004508

Windt, J., Taylor, D., Little, D., & Sporer, B. C. (2021). Making everyone's job easier. How do data scientists fit as a critical member of integrated support teams? *British Journal of Sports Medicine*, *55*(2), 73–75. https://doi.org/10.1136/bjsports-2020-102938

Zandavi, S. M., Koch, F., Vijayan, A., Zanini, F., Mora, F. V., Ortega, D. G., & Vafaee, F. (2021). *Disentangling single-cell omics representation with a power spectral density-based feature extraction*. https://doi.org/10.1101/2021.10.25.465657

Zhu, X., & Goldberg, A. B. (2009). *Introduction to Semi-Supervised Learning*. Springer International Publishing. https://doi.org/10.1007/978-3-031-01548-9

Zohuri, B. (2023). Artificial Super Intelligence (ASI) The Evolution of AI Beyond Human Capacity. *Current Trends in Engineering Science (CTES)*, *3*(6), 1–5. https://doi.org/10.54026/CTES/1049

Appendix 1: Data management plan

The research data in this thesis consists of video files obtained from a publicly available dataset initially provided by a Kaggle competition titled “DFL Bundesliga Data Shootout.” Although the competition has ended and the original dataset is no longer available through Kaggle, the files have been preserved and redistributed by users on GitHub. The setup script (setup.sh) included in the cloned repository downloads these video files from external sources such as Google Drive. No interviews, surveys, or personal data were collected during the research.

The data has been collected using a creative method that combines object detection models with video analysis. Specifically, pre-trained models for player and ball detection were applied to football match footage to extract positional and behavioral data. These models were downloaded and set up automatically by running the setup.sh script.

The collected data exists in the form of video files and numerical outputs such as coordinates, player movement trajectories, and possession status. The processed results are stored as CSV files, which include frame-level tracking data and performance metrics. This structured data was used for training machine learning models.

Personal or sensitive data: No personal or sensitive data has been collected or processed during the thesis work. All data used for the analysis comes from public match footage that does not contain identifying information of individuals. Since the data does not involve human participants in a manner that requires personal information, no privacy notice or consent process was necessary.

All code, documentation, and support files will be stored in a GitHub repository. Due to large file sizes, data models and testing videos will not be included directly in the repository. Instead, the README.md file will provide download links for the data models and include instructions for running the code.

The research data will not be reused after the completion of the thesis. The processed files and model outputs will be securely stored for one year from the date of thesis approval to allow verification of the results if needed. After that period, all stored data will be permanently deleted. No anonymization or data sharing agreement is required as personal data was not involved.

Appendix 2: requirements.txt File – Python Package Dependencies for the Project

```
supervision==0.6.0
numpy==1.26.4
opencv-python==4.9.0.80
transformers==4.39.3
umap-learn==0.5.5
scikit-learn==1.4.2
tqdm==4.66.2
sentencepiece==0.2.0
protobuf==4.25.3
```

Appendix 3: setup.sh file to install for the project

```bash
#!/bin/bash

# Get the directory where the script is located
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"

# Check if 'data' directory does not exist and then create it
if [[ ! -e $DIR/data ]]; then
    mkdir "$DIR/data"
else
    echo "'data' directory already exists."
fi

# download the models
gdown -O "$DIR/data/football-ball-detection.pt" "https://drive.google.com/uc?id=1isw4wx-MK9h9LMr36VvIWlJD6ppUvw7V"
gdown -O "$DIR/data/football-player-detection.pt" "https://drive.google.com/uc?id=17PXFNlx-jI7VjVo_vQnB1sONjRyvoB-q"
gdown -O "$DIR/data/football-pitch-detection.pt" "https://drive.google.com/uc?id=1Ma5Kt86tgpdjCTKfum79YMgNnSjcoOyf"

# download the videos
gdown -O "$DIR/data/0bfacc_0.mp4" "https://drive.google.com/uc?id=12TqauVZ9tLAv8kWxTTBFWtgt2hNQ4_ZF"
gdown -O "$DIR/data/2e57b9_0.mp4" "https://drive.google.com/uc?id=19PGw55V8aA6GZu5-Aac5_9mCy3fNxmEf"
gdown -O "$DIR/data/08fd33_0.mp4" "https://drive.google.com/uc?id=1OG8K6wqUw9t7lp9ms1M48DxRhwTYciK-"
gdown -O "$DIR/data/573e61_0.mp4" "https://drive.google.com/uc?id=1yYPKuXbHsCxqjA9G-S6aeR2Kcnos8RPU"
gdown -O "$DIR/data/121364_0.mp4" "https://drive.google.com/uc?id=1vVwjW1dE1drIdd4ZSILfbCGPD4weoNiu"
```

Appendix 4: Pseudocode for run_all: Player Tracking and Possession Detection

```
def run_all(source_video_path, device):
    # Load YOLO models for player, ball, and pitch detection
    player_model = YOLO(player_model_path).to(device)
    ball_model = YOLO(ball_model_path).to(device)
    pitch_model = YOLO(pitch_model_path).to(device)

    # Read all frames and video metadata
    frame_generator = get_video_frames(source_video_path)
    video_info = get_video_info(source_video_path)

    # Collect player crops from sampled frames for team
classification
    crops = []
    for frame in get_sampled_frames(source_video_path,
stride=STRIDE):
        players = detect_players(player_model, frame)
        crops += extract_player_crops(frame, players)

    # Train the team classifier
    team_classifier = TeamClassifier(device=device)
    team_classifier.fit(crops)

    # Initialize tracker and state variables
    tracker = ByteTrack(minimum_consecutive_frames=3)
    player_team_map = {}
    ball_trail = []
    csv_rows = []

    # Process each frame
    for frame_idx, frame in enumerate(frame_generator, start=1):
        # Detect pitch keypoints and compute transform
        keypoints = detect_pitch_keypoints(pitch_model, frame)
        if not valid_keypoints(keypoints):
            continue
        transformer = compute_view_transform(keypoints,
CONFIG.vertices)

        # Detect and track all players, GKs, and referees
        detections = detect_players(player_model, frame)
        tracked_detections = tracker.update(detections)
        players, goalkeepers, referees =
split_by_class(tracked_detections)

        # Predict team IDs
        player_crops = extract_player_crops(frame, players)
        player_team_ids = team_classifier.predict(player_crops)
        goalkeeper_team_ids = resolve_goalkeeper_teams(players,
player_team_ids, goalkeepers)

        # Detect ball and select the most probable candidate
        ball_detections = detect_ball(ball_model, frame)
        ball_position = select_valid_ball(ball_detections)
```

```
        # Update ball trail visualization
        if ball_position:
            update_ball_trail(ball_trail, ball_position)

        # Identify the closest player to the ball
        closest_player_id =
find_closest_player_with_ball(tracked_detections, ball_position)

        # Save detection data with transformed coordinates
        for i, detection in enumerate(tracked_detections):
            tid = detection.tracker_id
            if not tid:
                continue
            team_id = assign_team_id(tid, detection.class_id, i,
player_team_ids, goalkeeper_team_ids, player_team_map)
            position = transform_to_pitch(transformer,
detection.foot_position)
            has_ball = (tid == closest_player_id)

            csv_rows.append({
                "time": round(frame_idx / video_info.fps, 2),
                "teamId": team_id,
                "playerId": tid,
                "x": float(position[0]),
                "y": float(position[1]),
                "hasBall": has_ball
            })

        # Annotate and yield frame
        annotated_frame = annotate_frame(frame, tracked_detections,
player_team_ids, ball_trail)
        yield annotated_frame

    # Save CSV output
    write_to_csv("output.csv", csv_rows)
```

Appendix 5: Code for Drawing Radar-style Player Movement Path on the Pitch

```
from sports.configs.soccer import SoccerPitchConfiguration
from sports.annotators.soccer import draw_pitch,
draw_points_on_pitch
from supervision.draw.color import Color
import numpy as np
import matplotlib.pyplot as plt


player_id = 9
player_data = df_pos[df_pos["playerId"] ==
player_id].sort_values(by="time")

config = SoccerPitchConfiguration()
pitch_img = draw_pitch(config=config)

points = player_data[["x", "y"]].to_numpy().astype(np.float32)

face_color = Color(r=255, g=255, b=0)  # vàng
edge_color = Color(r=0, g=0, b=0)      # đen

pitch_img = draw_points_on_pitch(
    config=config,
    xy=points,
    face_color=face_color,
    edge_color=edge_color,
    radius=15,
    pitch=pitch_img
)

plt.figure(figsize=(10, 6))
plt.imshow(pitch_img)
plt.title(f"Radar-style Path for Player {player_id}")
plt.axis("off")
plt.show()
```

Appendix 6: Code for Generating Heatmap of Player Movement on Radar-style Football Pitch

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import cv2
from io import BytesIO
from PIL import Image

from sports.configs.soccer import SoccerPitchConfiguration
from sports.annotators.soccer import draw_pitch

df_pos = pd.read_csv("radar_output.csv")
player_id = 9
player_data = df_pos[df_pos["playerId"] ==
player_id].sort_values(by="time")

config = SoccerPitchConfiguration()
pitch_img = draw_pitch(config=config)
pitch_h, pitch_w = pitch_img.shape[:2]

pitch_img[:] = (0, 100, 0)

vertices = np.array(config.vertices)
max_x = vertices[:, 0].max()
max_y = vertices[:, 1].max()

x_scaled = player_data["x"] / max_x * pitch_w
y_scaled = player_data["y"] / max_y * pitch_h

fig = plt.figure(figsize=(pitch_w / 100, pitch_h / 100), dpi=100)
ax = fig.add_axes([0, 0, 1, 1])
sns.kdeplot(x=x_scaled, y=y_scaled, cmap="hot",
fill=True,alpha=1.0,bw_adjust=0.3,levels=50,thresh=0.001)
ax.set_xlim(0, pitch_w)
ax.set_ylim(pitch_h, 0)
ax.axis("off")

buf = BytesIO()
plt.savefig(buf, format='png', bbox_inches='tight', pad_inches=0,
transparent=True)
plt.close(fig)
buf.seek(0)
heatmap_np = np.array(Image.open(buf).convert("RGB"))
heatmap_np = cv2.resize(heatmap_np, (pitch_w, pitch_h))

final_img = cv2.addWeighted(pitch_img, 1.0, heatmap_np, 0.5, 0)

plt.figure(figsize=(10, 6))
plt.imshow(cv2.cvtColor(final_img, cv2.COLOR_BGR2RGB))
plt.title(f"Heatmap over Radar Pitch for Player {player_id}")
plt.axis("off")
plt.show()
```

Appendix 7: Code for K-Means Clustering and Visualization of Player Behavior

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

df_output = pd.read_csv("output.csv")
df_radar = pd.read_csv("radar_output.csv")

df_hasball = df_output[df_output["hasBall"] ==
True].sort_values(by="time").reset_index(drop=True)
df_hasball["new_phase"] = (df_hasball["playerId"] !=
df_hasball["playerId"].shift(1)) | \
                          (df_hasball["teamId"] !=
df_hasball["teamId"].shift(1))
df_hasball["phaseId"] = df_hasball["new_phase"].cumsum()

phases = df_hasball.groupby("phaseId").agg({
    "time": ["first", "last"],
    "playerId": "first",
    "teamId": "first"
}).reset_index()
phases.columns = ["phaseId", "start_time", "end_time", "playerId",
"teamId"]
phases["duration"] = phases["end_time"] - phases["start_time"]
phases = phases[phases["duration"] > 0]
phases["to_player"] = phases["playerId"].shift(-1)
phases["to_team"] = phases["teamId"].shift(-1)
phases["event"] = phases.apply(lambda row:
    "hold" if pd.isna(row["to_player"])
    else "pass" if row["teamId"] == row["to_team"]
    else "turnover", axis=1)

summary = phases.groupby("playerId").agg(
    passes=("event", lambda x: (x == "pass").sum()),
    turnovers=("event", lambda x: (x == "turnover").sum()),
    avg_reaction_time=("duration", "mean")
).reset_index()

# Step 2: Movement data from radar
movement = df_radar.groupby("playerId").agg(
    x_range=("x", lambda x: x.max() - x.min()),
    y_range=("y", lambda y: y.max() - y.min())
).reset_index()

# Step 3: Merge all player IDs
all_players =
pd.Series(df_output["playerId"].unique()).sort_values()
summary_all = pd.DataFrame({"playerId": all_players})
summary_all = summary_all.merge(summary, on="playerId",
how="left").fillna(0)
summary_all = summary_all.merge(movement, on="playerId",
how="left").fillna(0)
```

```
# Select relevant features
features = summary_all[["passes", "turnovers",
"avg_reaction_time"]]

# Step 2: Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)

# Step 3: Run KMeans clustering
kmeans = KMeans(n_clusters=3, random_state=42
summary_all["cluster"] = kmeans.fit_predict(X_scaled)

# Step 4: Visualize clusters
plt.figure(figsize=(8, 5))
sns.scatterplot(data=summary_all, x="passes",
y="avg_reaction_time", hue="cluster", palette="Set2", s=100)

# Add playerId labels
for _, row in summary_all.iterrows():
    plt.text(row["passes"] + 0.2, row["avg_reaction_time"],
f'ID:{int(row["playerId"])}', fontsize=9)

plt.title("K-Means Clustering of Player Tendencies")
plt.xlabel("Pass Count")
plt.ylabel("Avg Reaction Time (s)")
plt.grid(True)
plt.tight_layout()
plt.show()
```