UCLouvain

École polytechnique de Louvain

Optimal Football Team Building

A Data-Driven approach to build the best team

Author: **Pierrick VERHOOSEL**
Supervisors: **Jean-Charles DELVENNE, Martin TAMINIAUX**
Readers: **Charles PECHEUR, Marco SAERENS**
Academic year 2023–2024
Master [120] in Mathematical Engineering

Abstract

In this study, we introduce a graph-based representation of a football team, where players and their interactions are modeled as nodes and edges, respectively. This representation allows us to quantify both individual player contributions and the chemistry between players. Then the goal is to find the arrangement of players, which player at which position, that maximizes the overall team performance which is the sum of the individual player scores and the edge scores. By incorporating edge scores to represent player interactions, we enhance the accuracy and relevance of the team selection process. We introduce also a validation method to assess the relative importance of the chemistry between players compared to the individual qualities of the players. Finally, we demonstrate the potential applications of our model, including player recruitment, in-game management, and match outcome prediction. We also discuss the limitations and possible improvements of the model.

Acknowledgments

I would like to thank my supervisors, Jean-Charles Delvenne and Martin Taminiaux, for their guidance and support throughout this project. I would also like to thank my readers, Charles Pecheur and Marco Saerens, for having accepted to read and assess my work.

I would like to thank you especially for enabling me to conclude my studies by combining my passions. I have always been deeply passionate about sports, and being able to apply what I have learned in class to this field is very exciting.

I would also like to thank my family and friends for their support and encouragement throughout my studies.

Disclaimer

Some sentences of this thesis have been reformulated by the use of a large language model, in this case ChatGPT. The usage of this model has been strictly limited to the reformulation of the text and has not been used to generate new ideas or concepts.

Contents

Chapter 1

Introduction

In recent years, sports analytics has become more and more important. Football is one of the most famous sports and thus the use of data analytics is increasingly widespread. Football teams are using data to make key decisions about recruitment, formation choice, and how to adapt tactics during games. The goal of this thesis is to show how data can improve football team building.

Building a successful soccer team involves more than just putting together the best individual players. It requires also to take into account the teamwork, the interactions between players known as chemistry, and other aspects that a coach has to manage. This master's thesis looks at how data and analytics can help make better decisions when building football teams by adding a mathematical aspect to the team selection.

For each game, the coach of a football team has to make a lot of decisions : the choice of the players and the tactic. For each game, the coach can choose from a finite set of players, but each player has its own qualities. There are different types of players : goalkeepers, defenders, midfielders, and forward players. We will assign a score to each player, this score will depend on a lot of parameters. The score of the player is position-dependent because a player can be good in one position and bad in another. The score of the player will be used to quantify the quality of the player at a given position. The second main element of the model is the relationship between the players on the field, indeed some players are more compatible with others. We will assign a score to each of these relations to quantify how good this relationship is.

The main goal of this research is to create a model to build football teams that perform well by considering factors like player skills, team chemistry, and tactical choices. The model will be based on a graph representation of the team, where players are nodes and relationships between players are edges. The goal is to find the arrangement of players that maximizes the overall team performance, which means that we have to find which players and at which position they have to play to maximize the team performance.

The first contribution of this thesis is to provide a new perspective on football

team building, to show how data can help to build the best team, and to provide a framework for football team selection. Another contribution is to try to quantify the chemistry between players and more important to create a method to assess how important the chemistry is compared to the individual qualities of the players. Finally, this thesis can contribute in many ways like recruiting new players, managing the team during the game, or even predicting the outcome of a game. This thesis can be extended to other team sports, offering a new perspective on team selection and performance optimization.

The structure of this thesis is as follows. In the first chapter, we will conduct a state-of-the-art analysis to see what has already been done in the field of football team building. In the second chapter, we will present the data sources and the notations used in this thesis. In the third chapter, we will introduce the problem as a graph problem and present the mathematical modeling of the problem. Then we will present some theories about Machine Learning models and how we can use them efficiently. After that, we will see how the players are rated and how we can quantify the chemistry between players. Next, we will see how we can validate our mathematical model which is based on the players' scores and chemistry and how we can assess the importance of the chemistry between players. By merging all together, we will see some useful applications of this model. Finally, we will conclude the subject and discuss the limitations and the possible improvements of the model.

Chapter 2

State of the Art

First, it is important to conduct a state-of-the-art analysis to learn what has already been done and where we can bring something new. We live in a world where artificial intelligence and data are increasingly present. In recent years, a large number of tasks have been improved due to this phenomenon. This development is present in many fields such as health, finance, the military, and more. Sports are not exempt from this. Indeed, there are many ways to use data and artificial intelligence to improve sports performance.

The use of data in sports fields is not recent, football is maybe the most important sport in the world and thus the use of data in football is widespread. The data can be used in many ways, for example, to analyze the performance of a player, to predict the outcome of a game, to improve the recruitment of players, to estimate the price of the player, and more. As explained in the article written in 2023 by Filip Sekan [24], the presence of data in football has increased with the "Video Boom" in the 2000s. The video analysis has allowed us to have a lot of data about the players and the games. An example of a club that has been successful in using data is Liverpool FC. In 2015, their data scientists have developed a model that can determine whether a player's activities on the field increase the likelihood of scoring. Their offensive trio of Salah, Mané, and Firminho, which was recruited using data analytics, contributed significantly to the important accomplishments they have made including its Champions League and, particularly, English Premier League victories.

Now, we will look at the existing literature on the subject and see what has already been done in the field of football team building. We will also look at the different methods used to build football teams and the different aspects that can be taken into account.

The first paper was written by Danny Leese [13] in 2020, this paper is about team chemistry in basketball. Even if our goal is to build a football team, other sports as basketball have the same problem. His approach was to group all similar players in the NBA. For example, there would be a group of 3-point shooters, a group of defensive big men, and a group of pass-first guards, etc. Then look at all 5-player

lineup combinations over the past 20 years to see which groups made up each lineup. By determining the performance level of a 5-group combination in a lineup, they could quantify the chemistry of that group mixture. This research only focuses on the overall chemistry of the team and does not take into account the individual qualities of the players. He creates a cluster of players based on their qualities but after that, we can not compare 2 players in the same cluster, this gives us a way to improve. For example, if the coach has to choose only one 3-point shooter, he has to know which one is the best but for the moment he can not.

The next relevant paper is introduced in 2023 [12] by Mahdi Nouraie. He proposed to utilize deep neural networks to evaluate the suitability of each player for several positions on the field. Finally, the problem is formulated as finding the maximum weighted matching in a complete bipartite graph. The objective is to obtain the best possible alignment between players and the positions designated by the coach. The model here only considers the individual qualities of the players and does not take into account the chemistry between the players. He considers that the best team is the one that gives the best matching between players and positions. This is a good way to start but we can improve this model by taking into account the chemistry between players.

The last paper is written by Lotte Bransen and Jan Van Haaren [1] . Indeed, they attempted to introduce a method to quantify the offensive and defensive impact of a pair of players when they play together on the field. These scores are based on the positive or negative impact of actions performed between a pair of players. They introduced the concept of VAEP for "Value Added by Estimating Probability". In short, it's about quantifying how much an action increases the chances of one's team scoring and how much it decreases the chances of the opponent's team scoring. After that, they modeled the problem of assembling the best team as one that maximizes the sum of chemistry between all pairs of players with some constraints to have a balanced team. It is important to note here that they create their team only based on chemistry and thus do not take into account the individual qualities of the players. Even if implicitly, it is true that two players with good individual qualities are more likely to form a good pair. This gives us our first way for improvement. It might be interesting to consider both the value of the players and their chemistry. Then we can attempt to quantify how important chemistry is compared to the individual qualities of the players. Another important element they did not address is the validation of their model. They do not demonstrate that by fielding the optimal team, they are more likely to achieve good results. So, these are two significant ways to explore. However, their method for quantifying the chemistry between players is interesting and will be presented in more detail later on.

Chapter 3

Data Sources and Notations

3.1 Data sources

To achieve this project, different data sources were used. These sources are either CSV files or Python packages. One important thing to do is to understand the data from each source so we can use the information correctly and match it between all the different sources.

3.1.1 StatsBomb

First, one of the most important data sources is StatsBomb[3]. This company specializes in football-related data and has provided free access to all data from the 2015/2016 season for the top 5 leagues: France, England, Italy, Spain, and Germany. StatsBomb has also released a Python package to make it easy to work with this data. Thanks to their data, we have access to all the events that happen during the matches, allowing us to create a lot of statistics about the game. We get the information about who did what, where, and how. The events during the match are things like passes, shots, dribbles, tackles, and more. For example, we can see who made the pass, who received it, where it was made, and where it was received. In each match, there are more or less 3000 events, that represent an event every 2 seconds more or less, so we have a lot of data to work with.

3.1.2 FIFA Dataset

The FIFA dataset contains information about the players in the game FIFA. This dataset is available on Kaggle[9]. It contains information about the players' skills, such as their speed, shooting, passing, and more. This dataset is very useful for our project because it allows us to assign a score to each player based on their skills. This score will be used to determine the quality of the player and to help us build the best team possible. The dataset also contains information about the players' positions, which is very important for our project.

3.1.3 Mplsoccer package

The mplsoccer [4] package is a Python package that allows us to create football visualizations. It provides a lot of tools to create different types of plots, such as heat maps, pass maps, and expected goals. This package is very useful for creating visualizations of the data we have. It also provides a way to create the tactical schemes we will use in our project. The package will help us to display the football team and to see the positions of the players on the field. We used the coordinates of the players from the package to represent them on the field.

3.1.4 European Soccer Database

The European Soccer Database [10] is a dataset available on Kaggle that contains information about football matches, players, and teams from several European countries. This dataset is very useful for our project because it contains a lot of information about the matches, such as the score, the betting odds, the players who played, and more. This dataset will help us to validate our model and to see if the team we have built is performing well.

3.2 Notation

First of all, let introduce some notations for the sake of clarity. An important aspect to define is the positions assigned to the players on the field, along with the formations used. It is also important to define all the mappings between different representations in order to work easily with multiple data sources.

3.2.1 Players Positions

First, there are two main types of positions: field players and goalkeepers. Whatever the context, a goalkeeper remains a goalkeeper. However, for field players, the positions may vary slightly depending on the case. In our case, it's important to verify that the positions assigned by FIFA match those defined by the mplsoccer package. It's important to note that in the Kaggle dataset, we only have coordinates for player positions. Therefore, we need to create a mapping between these coordinates and the traditional FIFA positions.

The player positions are usually abbreviated with two or three letters. Here are the positions used by FIFA :

- **GK** (Goalkeeper) : The player who guards the goal.
- **CB** (Center Back) : Defenders in the center.
- **LB/RB** (Left Back/Right Back) : Defenders on the left and right sides of the defense.

- **LWB/RWB** (Left Wing Back/Right Wing Back) : Defenders on the left and right flanks but with a more offensive role.
- **CDM** (Central Defensive Midfielder) : A midfielder whose role is more defensive.
- **CM** (Central Midfielder) : Midfielders in the center.
- **CAM** (Central Attacking Midfielder) : A midfielder whose role is more offensive.
- **LM/RM** (Left Midfielder/Right Midfielder) : Midfielders positioned on the left and right sides.
- **LW/RW** (Left Wing/Right Wing) : A player positioned on the left and right flank with an offensive role.
- **CF** (Center Forward) : A forward positioned centrally.
- **ST** (Striker) : A forward whose primary role is to score goals.

These abbreviations are commonly used to define player roles and positions on the field. These notations will be very useful for the sake of clarity. In the following table, you'll find the mapping between the positions used by FIFA and those used by mplsoccer. It's important to notice that the main difference is that mplsoccer further divides central positions into right-central and left-central.

Position FIFA	**Position mplsoccer**
Right Back (RB)	RB
Center Back (CB)	RCB, CB, LCB
Left Back (LB)	LB
Left Wing Back (LWB)	LWB
Right Wing Back (RWB)	RWB
Central Defensive Midfielder (CDM)	RDM, CDM, LDM
Right Midfielder (RM)	RM
Central Midfielder (CM)	RCM, CM, LCM
Left Midfielder (LM)	LM
Right Wing (RW)	RW
Left Wing (LW)	LW
Central Attacking Midfielder (CAM)	RAM, LAM, CAM
Center Forward (CF)	RCF, LCF
Striker (ST)	RS, LS, ST, SS

As explained earlier, the Kaggle dataset contains only the coordinates of players on the field, not their positions. Therefore, it's necessary to create a mapping between the coordinates and the FIFA positions. In the figure below, you will find two examples of mapping. All the blue points represent the coordinates present in the dataset, and the red points represent the coordinates associated with a certain position. In this case, you will find the mapping for CB and LM.

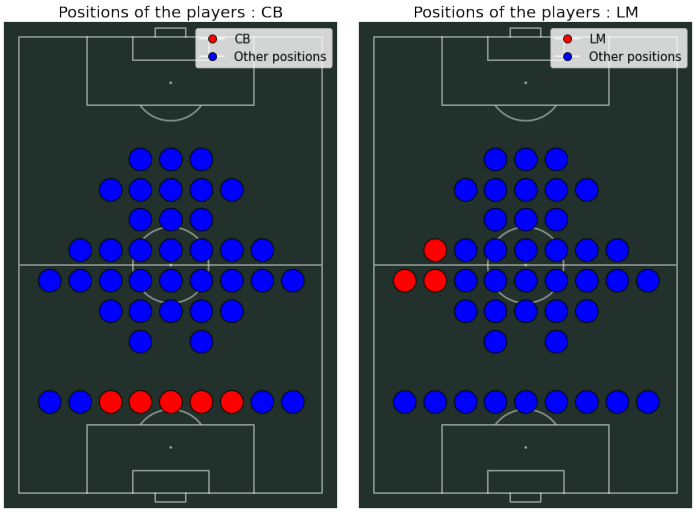


Figure 3.1: Mapping FIFA and mplsoccer positions

3.2.2 Tactical Schemes

Now that we've seen the different possible positions on the field, it's time to analyze how team formations work. The tactical setup of a team corresponds to the positions assigned to players on the field. There are a lot of possible schemes, and each formation can be represented by a code consisting of 3 or 4 digits. This code allows us to read the tactical scheme from top to bottom, meaning that the first digit indicates the number of defenders, followed by the midfielders, and finally the forwards. Given that there's always just one goalkeeper, there is no need to add a 1 at the beginning of the code. In cases where the code contains 4 digits, it indicates that there is an additional line either between the defenders and midfielders or between the midfielders and forwards. For example, the 4141 formation has a player positioned between the defenders and the midfielders. Such a player is known as a defensive midfielder.

Here are the 6 tactical schemes we are considering 3.2:

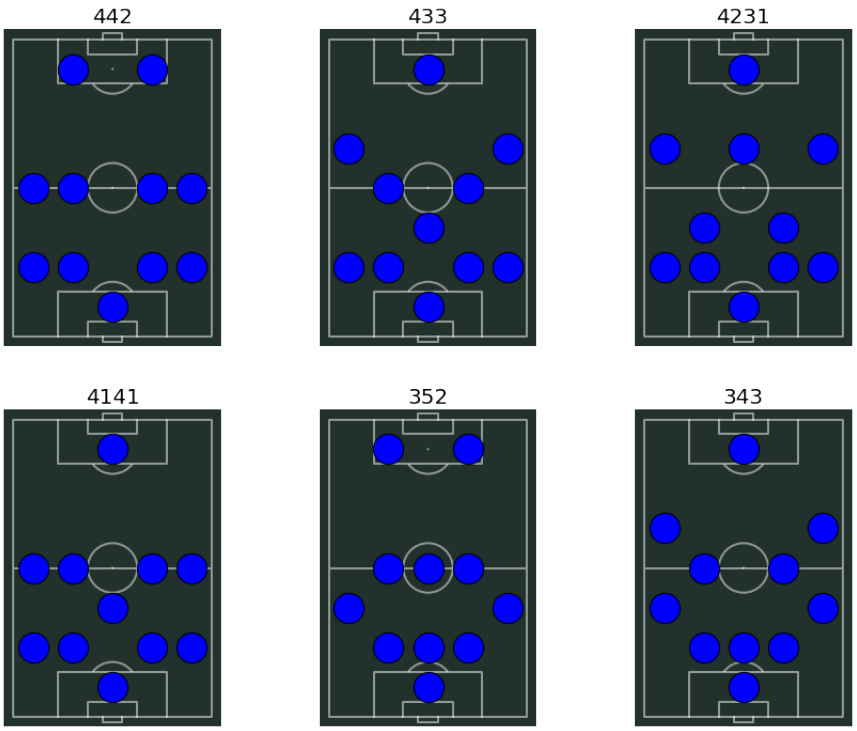


Figure 3.2: Tactical Schemes

These schemes were obtained with the mplsoccer package, that's why we have to use the mapping between these positions and the positions from FIFA. The player positions in these formations can vary depending on the context. In some cases, the players are shown across the entire field, while in others, they are shown on just half of the field. In practice, during a match, the players are often closer than they appear in these figures.

Chapter 4

Modeling football team as a graph problem

In this chapter, the problem of building a football team is introduced as a graph problem. A brief reminder of graph theory is provided, followed by the introduction of the graph representation. The mathematical modeling of the problem is then presented, and the chapter concludes with the implementation of the model.

4.1 Graph representation

4.1.1 Theory

A **graph** is a mathematical structure that represents relationships between pairs of objects. A graph consists of two main components:

- **Nodes**: These are the individual elements or points in the graph.
- **Edges**: These are the connections or links that join pairs of nodes. Edges can be directed or undirected. Directed edges indicate a specific direction (from one vertex to another), while undirected edges represent a mutual or bidirectional relationship.

A graph can be denoted mathematically as $G = (V, E)$, where:

- V represents the set of nodes.
- E represents the set of edges, with each edge defined as a pair of nodes.

A graph may have additional properties that would be useful for us :

- **Weighted Edges**: Each edge has a numerical value. It indicates the strength or the cost of the connection.
- **Weighted Nodes**: Each node has a numerical value. It indicates the strength or the cost of the node.
- **Complete**: A complete graph with n nodes, denoted K_n is a graph where every pair of distinct nodes is connected by a unique edge.

4.1.2 Graph visualization

In the context of our problem, we can represent the football team as a graph. The nodes represent the players, and the edges represent the relationships between players. The edges can be weighted based on the strength of the relationship between players. The nodes can also be weighted based on the quality of the player. First of all, we will visualize our problem on the drawing below 4.1. A football team normally consists of 11 players on the pitch, but on the drawing, it's a team of 4 players for the sake of clarity. The blue circles represent the positions on the pitch. There must be exactly one player chosen for each position. We can also see that there are different choices of players for each position, each represented by a red node. Finally, the orange edges represent the relationships between players in different positions. One more time, we only represent the edges between nodes from close positions for the sake of clarity. This drawing is a simplified version of the overall problem. However, it still gives an idea of the problem.

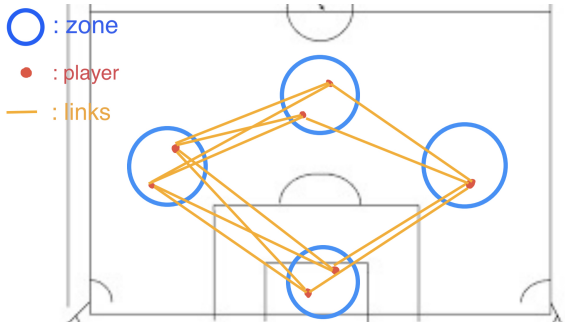


Figure 4.1: Model with a team of 4 players

The graph below 4.2 shows an example of a complete solution to the problem. The graph solution has 11 players, one at each position, and each pair of players is connected by an edge. Thus this is a complete graph of 11 nodes, denoted K_{11}. In this case, we choose a 4-4-2 formation but later we'll look at different tactical schemes and different ways of weighting the importance of the edges according to the position of the players they connect.

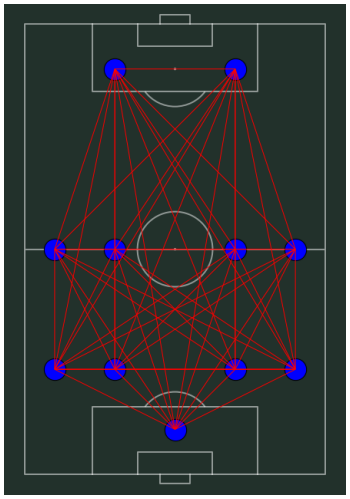


Figure 4.2: Example of complete graph solution 4-4-2

Let $G = (V, E)$ be the graph containing all available players and all the possible relations between them. The solution to the problem is a complete subgraph K_{11} of G with weighted edges and nodes. Therefore, the problem can be formulated as finding the complete subgraph of G that maximizes the sum of the weights of its nodes and edges under some constraints.

4.2 Mathematical modeling

Now that we have introduced the problem and show a visual intuition, it's time to move on to its mathematical modeling. First of all, we will present our initial problem from a new perspective, aiming to find a parallel between our problem and a well-known one. If we can find a way to represent our problem in terms of a more well-known one, then we might be able to discover a more efficient method to solve our problem.

4.2.1 Assignment problem

The *assignment problem* involves the assignment of a set of tasks to a set of agents in such a way that each task is assigned to exactly one agent and each agent is assigned at most one task. This can be represented using the variables:

s_i : the ith task,
t_j : the jth agent,
$c_{i,j}$: the cost or benefit associated with assigning task s_i to agent t_j.

We can model this problem as a bipartite graph as shown below 4.3 for a small example.

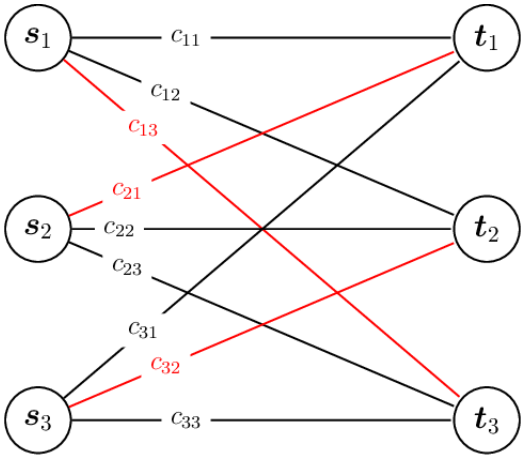


Figure 4.3: Assignment problem [14]

The goal is typically to minimize or maximize a certain objective function that depends on the costs or benefits of the assignments.

In our case, we want to find the association of agents and tasks that maximizes the cost function. The assignment problem can be mathematically formulated as follows:

$$\text{Maximize} \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j}x_{i,j}$$

subject to the constraints:

$$\sum_{j=1}^{m} x_{i,j} = 1, \quad \text{for } i = 1,\ldots,n$$
$$\sum_{i=1}^{n} x_{i,j} \leq 1, \quad \text{for } j = 1,\ldots,m$$
$$x_{i,j} \in \{0,1\}, \quad \text{for } i = 1,\ldots,n \text{ and } j = 1,\ldots,m$$

where:

$$x_{i,j} = \begin{cases} 1 & \text{if task } i \text{ is assigned to agent } j \\ 0 & \text{otherwise} \end{cases}$$

$c_{i,j}$ is the cost or benefit associated with assigning task i to agent j

In the context of our assignment problem, we utilize an incidence matrix, denoted as M, to represent the relationships between tasks and agents. The matrix M is a binary matrix where each row corresponds to a task, each column corresponds to an agent, and the entries $M_{i,j}$ indicate whether task i could be assigned to agent j. Formally, M is defined as follows:

$$M_{i,j} = \begin{cases} 1 & \text{if task } i \text{ could be assigned to agent } j \\ 0 & \text{otherwise} \end{cases}$$

The incidence matrix M provides a concise representation of the assignment relationships, allowing us to formulate and solve the assignment problem efficiently. Using M, the objective of maximizing the total cost and the associated constraints can be expressed easily and concisely.

Additionally, it's important to notice that in the case of a bipartite graph, the incidence matrix M is considered to be totally unimodular. This mathematical property ensures that all square submatrices of M have determinants that are equal to -1, 0, or 1. The total unimodularity of M holds significant implications for optimization problems, particularly linear programming, as it simplifies the computation and guarantees that the solution of the linear programming relaxation remains integer. In the context of our assignment problem, the total unimodularity of M contributes to the efficiency of optimization algorithms applied to find the optimal assignment.

Finally, we can rewrite the final problem incorporating matrix M and use the fact that the matrix is totally unimodular (TU). The assignment problem can be expressed as a linear sum assignment problem with the objective of maximizing the total cost. Let $c_{i,j}$ be the cost associated with assigning task i to agent j, and let $x_{i,j}$ be the continuous assignment variable:

$$\text{Maximize } \sum_i \sum_j c_{i,j} \cdot x_{i,j}$$

subject to the constraints:

$$\sum_j M_{i,j} \cdot x_{i,j} = 1 \quad \text{for each task } i$$

$$\sum_i M_{i,j} \cdot x_{i,j} \leq 1 \quad \text{for each agent } j$$

$$0 \leq x_{i,j} \leq 1 \quad \text{for } i = 1, \ldots, n \text{ and } j = 1, \ldots, m$$

The main advantage is that now our variables are continuous, making the problem easier to solve.

4.2.2 Baseline Model

By modeling our problem as an assignment problem, we might be able to use existing algorithms or optimization techniques that are well-established for solving such problems. It provides a new perspective that could lead to more efficient solutions. First, let's define two lists:

- The first list, called *player*, is the list of players available for selection.
- The second list, called *position*, is the list of the different positions needed on the field.

In order to model our problem, let consider these notations :

- $position_i$: the ith position on the field,
- $player_j$: the jth player,
- $c_{i,j}$: the benefit associated with assigning player $player_j$ to position $position_i$,
- $x_{i,j}$: Binary variable indicator if $player_j$ plays at $position_i$,
- N: Total number of players available.

Then M is the incidence matrix between players and their position on the field. Depending on the version of the problem we consider, we will have a more or less significant number of possible positions for a player. Formally, we can define M as :

$$M_{i,j} = \begin{cases} 1 & \text{if position } i \text{ is assigned to player } j \\ 0 & \text{otherwise} \end{cases}$$

So far, we have only considered the scores of the players. This initial model, which we refer to as the *Baseline Model* (see Equation 4.1), can be written as follows:

$$\text{Maximize} \quad \sum_i \sum_j c_{i,j} \cdot x_{i,j} \tag{4.1}$$

subject to the constraints:

$$\sum_j M_{i,j} \cdot x_{i,j} = 1 \quad \text{for each position } i \tag{4.2}$$

$$\sum_i M_{i,j} \cdot x_{i,j} \leq 1 \quad \text{for each player } j \tag{4.3}$$

$$0 \leq x_{i,j} \leq 1 \quad \text{for } i = 1, \ldots, 11 \text{ and } j = 1, \ldots, N \tag{4.4}$$

The *Baseline Model* 4.2.2 provides a way to easily and efficiently solve our initial problem, also using the property that the matrix M is totally unimodular (TU), and therefore the solution to the relaxed problem is identical to the solution of the integer variable problem.

4.2.3 Complete Model

The main concern is that, at the moment, we completely disregard the scores of the edges between players, and to take them into account, we need to modify our assignment problem. We can keep the current constraints related to the assignment problem, but we need to modify the objective function by adding a term for the edge scores. Additionally, we should introduce constraints related to these edge scores.

We want to ensure that the edge between two players will be selected only if both players are selected. By "selected", we mean that the associated variable has a value of 1 and thus has an impact on the objective function 4.5. Let's analyze a simple case with two players and one edge between them.

Let x be the binary variable indicating if the first player is on field, and let y be the binary variable indicating if the second player is on field. Then, the binary variable z associated with the edge between the two players must equals to 1 if only if both x and y are equals to 1 and equals to 0 otherwise. In order to model this condition, we create 3 constraints :

- $z \leq x$
- $z \leq y$
- $z \geq x + y - 1$

To prove that these constraints guarantee that $z = 1$ if and only if $x = 1$ and $y = 1$, we can examine the cases all possible combinations of x and y.

1. If $x = 0$ and $y = 0$, then the constraints become:

$$\begin{cases} z \leq 0 \\ z \leq 0 \\ z \geq 0 + 0 - 1 \end{cases} \Rightarrow z = 0$$

2. If $x = 0$ and $y = 1$, then the constraints become:

$$\begin{cases} z \leq 0 \\ z \leq 1 \\ z \geq 0 + 1 - 1 \end{cases} \Rightarrow z = 0$$

3. If $x = 1$ and $y = 0$, then the constraints become:

$$\begin{cases} z \leq 1 \\ z \leq 0 \\ z \geq 1 + 0 - 1 \end{cases} \Rightarrow z = 0$$

4. If $x = 1$ and $y = 1$, then the constraints become:

$$\begin{cases} z \leq 1 \\ z \leq 1 \\ z \geq 1 + 1 - 1 \end{cases} \Rightarrow z = 1$$

We can conclude that these constraints verify that $z = 1$ if and only if $x = 1$ and $y = 1$.

Finally, let's introduce the variables $\ell_{i,j,k,l}$ indicating whether the edge between player j at position i and player l at position k has to be selected and $e_{i,j,k,l}$ be the score associated to this edge. For the sake of clarity, let

$$\Omega = \{(i,j) \mid \text{for every player } j \text{ and position } i\}$$

Some relations are more or less important, for example the keeper and the striker are not playing very often together, the weight of the edge between a player at $position_i$ and a player at $position_j$ is $w_{i,j}$. These weights are determined by the weighting scheme 6.3. Finally, we have to introduce some weights w_1 and w_2 in order to give more or less importance to the score of the nodes or the edges. By merging all components together, we arrive at the *Complete Model*, which can be written as follows:

$$\text{Maximize} \quad \underbrace{w_1 \sum_{(i,j)\in\Omega} c_{i,j} \cdot x_{i,j}}_{\text{Players Score}} + \underbrace{w_2 \sum_{(i,j)\in\Omega} \sum_{(k,l)\in\Omega} w_{i,k} \cdot e_{i,j,k,l} \cdot \ell_{i,j,k,l}}_{\text{Chemistry Score}} \tag{4.5}$$

subject to the constraints:

$$\sum_j M_{i,j} \cdot x_{i,j} = 1 \quad \text{for each position } i \tag{4.6}$$

$$\sum_i M_{i,j} \cdot x_{i,j} \leq 1 \quad \text{for each player } j \tag{4.7}$$

$$x_{i,j} \in \{0,1\} \quad \text{for } (i,j) \in \Omega \tag{4.8}$$

$$\ell_{i,j,k,l} \leq x_{i,j} \quad \text{for } (i,j) \in \Omega \text{ and } (k,l) \in \Omega \tag{4.9}$$

$$\ell_{i,j,k,l} \leq x_{k,l} \quad \text{for } (i,j) \in \Omega \text{ and } (k,l) \in \Omega \tag{4.10}$$

$$\ell_{i,j,k,l} \geq x_{i,j} + x_{k,l} - 1 \quad \text{for } (i,j) \in \Omega \text{ and } (k,l) \in \Omega \tag{4.11}$$

$$\ell_{i,j,k,l} = \ell_{k,l,i,j} \quad \text{for } (i,j) \in \Omega \text{ and } (k,l) \in \Omega \tag{4.12}$$

$$\ell_{i,j,k,l} \in \{0,1\} \quad \text{for } (i,j) \in \Omega \text{ and } (k,l) \in \Omega \tag{4.13}$$

It is important to note that we no longer use the TU property, and we are once again working with integer variables. Finally, we have successfully modeled our problem starting from a known problem : the Assignment problem and then adding some specific features related to our situation. Initially, we only considered the scores of the players : *Baseline Model* 4.1, but at the end we have also taken into account the scores of the edges between players : *Complete Model* 4.5. By considering the edge

scores makes our problem more challenging to solve. As the number of variables and constraints increases, the problem becomes more complex. However, the introduction of edge scores allows us to consider the relationships between players and to build a team that is not only composed of the best players but also of players who work well together. This is a significant improvement that can lead to better team performance. We can notice that if we set $w_2 = 0$, we get back to the *Baseline Model* 4.1.

4.3 Implementation

Now that we have derived the formulation of our problem, let's explain how we have implemented it in practice. It is important to note that for the sake of clarity and ease of implementation, we have considered that each player has a unique assigned position. As a reminder, in the FIFA dataset [9], a player has a unique position assigned but in practice, a player can play at different positions. Therefore, if we want to assign multiple possible positions to a player, we need to create twins of that player and assign a unique position for each twin. Hence, we have to create as many twins as additional positions. That means that unlike seen in the formulation of the problem for the *Baseline Model* 4.1 and the *Complete Model* 4.5, the variable associated with a player does **not** depend with the position. This helps us to simplify the implementation because the variables from the objective function 4.1 and 4.5 are not dependent on the position anymore. That does not change anything to the problem, it's just a way to simplify the implementation.

We have assumed that goalkeepers can only be goalkeepers and that outfield players can only be outfield players. The score of outfield players in positions other than their base position will be predicted by regression models detailed later in the Player Scores section 6.1. We have still N players but to take into account the different positions of a player we will assume that the same player playing at 2 different positions is 2 players different. But to keep the problem correct, We are going to add a constraint to prevent the same player from being selected for different positions. Let $N = N_{GK} + N_{OP}$ where N_{GK} is the number of goalkeepers in the set of players and N_{OP} is the number of outfield players. There are only 10 outfield player positions in a team composition, there are 11 players on the pitch but we do not take into account the GK. Then let M be the total number of players that we will create by considering that the same player playing at 2 different positions is counted as 2 players. Finally, we have $M = N_{GK} + 10 \cdot N_{OP}$.

Here is a small example with Lionel Messi and Neymar, you can see in the table below 4.1 that we have created twins for them.

Attribute	**Player/Twin 1**	...	**Player/Twin 10**	**Player/Twin 11**	...
Name	Messi	...	Messi	Neymar	...
Position	Position 1	...	Position 10	Position 11	...
Score	Score 1	...	Score 10	Score 11	...

Table 4.1: Twins example for Lionel Messi and Neymar

If we would like to create this table for the whole team, there would be M columns at the end. Let's create a new list, called *twins* that contains all the twins of the players.

- *twins*: The list of twins of the players available for selection. This list has a size of M. The i^{th} element of the list is the i^{th} twin available for selection.

In the example above with Lionel Messi and Neymar, the players at the positions from 1 to 10 in the *twins* list are the twins of Lionel Messi and the players at the positions from 11 to 20 are the twins of Neymar and so on.

Let us define the following:

- b_i: The score of the i^{th} twin at his given position.
- e_{ij}: The score of the edge between the i^{th} twin and the j^{th} twin. These scores are also determined before the optimization problem as they are parameters of the problem not variables.
- x_i: This variable indicates if the i^{th} twin will be aligned by the coach or not.
- ℓ_{ij}: These binary variables indicate if both the i^{th} twin and the j^{th} twin are playing.

These formulations are very close to the ones we have seen in the previous section for the *Complete Model* 4.5. The main difference is that we remove the dependence of the variables with the position. But we have to add a method to get the set of twins from a unique player and the set of players playing at a given position. We will define two functions δ and γ to get these sets.

The function δ is very useful to get the set of twins from a unique player. The function is defined as follows :

$$\delta(player_j) = \{i | x_i \text{ is a twin of } player_j\}$$

The function γ is also very useful as it allows to get the set of players playing at a given position. This function is defined as follows :

$$\gamma(position_j) = \{i | x_i \text{ is the variable linked to a player at } position_j\}$$

These functions allow us to model the constraints more easily. We emphasize that this does not change the original problem 4.2.3, it is simply an easier way to implement it. Let's now discover these modifications.

4.3.1 Baseline Model Implementation

First, we will implement the simplest version of our problem, which is its baseline version 4.2.2. We can notice that we have now M players instead of N because we take also into account the twins of the players. But the x variables only depend on the player and not on the position as each player or twins of a player can only be assigned to one position.

$$\text{Maximize} \sum_{i=1}^{M} b_i \cdot x_i \tag{4.14}$$

subject to the constraints:

$$\sum_{j \in \gamma(position_i)} x_j = 1 \quad \text{for } i \in \{1, \ldots, 11\} \tag{4.15}$$

$$\sum_{j \in \delta(player_i)} x_j \leq 1 \quad \text{for } i \in \{1, \ldots, N\} \tag{4.16}$$

$$x_i \in \{0, 1\} \quad \text{for } i \in \{1, \ldots, M\} \tag{4.17}$$

The first constraint 4.15 ensures that there is exactly one player at each position. The second constraint 4.16 ensures that each player is assigned to at most one position. It ensures that two twins of the same player can not be on the field in the same time. The objective function is to maximize the sum of the scores of the selected players. By "selected", we mean that the associated variable x_i to $player_i$ has a value of 1 and thus has an impact on the objective function 4.1 because the player is on the pitch.

4.3.2 Complete Model Implementation

Now we will focus on implementing the *Complete Model* 4.2.3, which means incorporating the impact of edges' scores. As in the previous section 4.3.1, we have now M players instead of N but the x variables only depend on the player and not on the position.

$$\text{Maximize} \underbrace{w_1 \sum_{i=1}^{M} b_i \cdot x_i}_{\text{Player Score}} + \underbrace{w_2 \sum_{i=1}^{M} \sum_{j=i+1}^{M} \ell_{i,j} \cdot w_{i,j} \cdot e_{i,j}}_{\text{Edges Score}} \tag{4.18}$$

subject to the constraints :

$$\sum_{j \in \gamma(position_i)} x_j = 1, \quad \text{for } i \in \{1, \ldots, 11\}, \tag{4.19}$$

$$\sum_{j \in \delta(player_i)} x_j \leq 1, \quad \text{for } i \in \{1, \ldots, N\}, \tag{4.20}$$

$$x_i \geq \ell_{i,j}, \quad \text{for } i \in \{1, \ldots, M\} \text{ and } j \in \{i+1, \ldots, M\}, \tag{4.21}$$

$$\ell_{i,j} \geq x_i + x_j - 1, \quad \text{for } i \in \{1, \ldots, M\} \text{ and } j \in \{i+1, \ldots, M\}, \tag{4.22}$$

$$x_i \in \{0, 1\}, \quad \text{for } i \in \{1, \ldots, M\}, \tag{4.23}$$

$$\ell_{i,j} \in \{0, 1\}, \quad \text{for } i \in \{1, \ldots, M\} \text{ and } j \in \{i+1, \ldots, M\}. \tag{4.24}$$

This reformulation of the problem 4.2.3 may be a bit confusing at first, but it helps us to implement the problem more easily. This reformulation will be also helpful for the last chapter about applications 8.

Chapter 5

Model Building and Evaluation methods

5.1 Regression Models

In this section, we look at regression models, an important part of our project. Regression helps us find relationships between different features and make predictions based on these features. We will use different types of regression to understand what influences certain outcomes and to build models that can predict these outcomes for new data. Regression is a form of supervised learning, where we use labeled data to train our models. By learning from existing data, we aim to build models that can accurately forecast outcomes for new inputs.

5.1.1 Linear Regression

Linear regression [25] is a fundamental method in supervised learning that predicts a continuous outcome based on one or more input features. It assumes a linear relationship between the target and the features. In multiple linear regression, the model involves several features, and the model can be represented as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$$

where $x_1, x_2, \ldots, x_n$ are the features, and $\beta_1, \beta_2, \ldots, \beta_n$ are their respective coefficients.

The goal of linear regression is to find the best values for $\beta_0, \beta_1, \ldots, \beta_n$ to minimize the error between the predicted outcomes and the actual outcomes. The most common measure of this error is the cost function. In linear regression, the cost function measures how well the linear model fits the data. It is used to determine the optimal coefficients $(\beta_0, \beta_1, \ldots, \beta_n)$ by minimizing the overall error. The most commonly used cost function in linear regression is the Mean Squared Error (MSE), which calculates the average of the squared differences between the observed values and the predicted values. Here's how it works: Given a dataset with n observations and a linear regression model, the cost function can be defined as:

$$J(\beta_0, \beta_1, \ldots, \beta_n) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where:

- y_i is the actual value for the ith observation,
- $\hat{y}_i$ is the predicted value for the ith observation, calculated as $\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_n x_{in}$,
- n is the total number of observations.

The goal of linear regression is to find the values of $\beta_0, \beta_1, \ldots, \beta_n$ that minimize the cost function $J(\beta_0, \beta_1, \ldots, \beta_n)$. This process, often known as training the model, typically involves optimization techniques like gradient descent, which iteratively adjusts the coefficients to reduce the MSE.

By minimizing the MSE, linear regression aims to create a model that best fits the given data, leading to more accurate predictions and a better understanding of the relationship between features.

5.1.2 Logistic Regression

Logistic regression [26] is a statistical model used for binary classification tasks. It models the probability of a binary outcome (0 or 1) based on one or more predictor variables.

Logistic regression uses the logistic function to model this probability:

$$P(Y = 1|X) = \frac{1}{1 + e^{-z}}$$

where z is a linear combination of the features X weighted by the coefficients of the model:

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

The β's represent the model coefficients, which are estimated from the training data to minimize a loss function, typically the logistic cost function:

$$J(\beta) = -\frac{1}{N} \sum_{i=1}^{N} \left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right)$$

where $\hat{y}_i$ is the model's prediction for observation i and y_i is the true label for observation i.

The coefficients β are often estimated using optimization methods such as gradient descent or its variants, where the goal is to find the values of β that minimize the

loss function $J(\beta)$. Once the coefficients have been estimated, the model can be used to make predictions on new data by using the sigmoid function to compute probabilities and applying a decision threshold to classify observations.

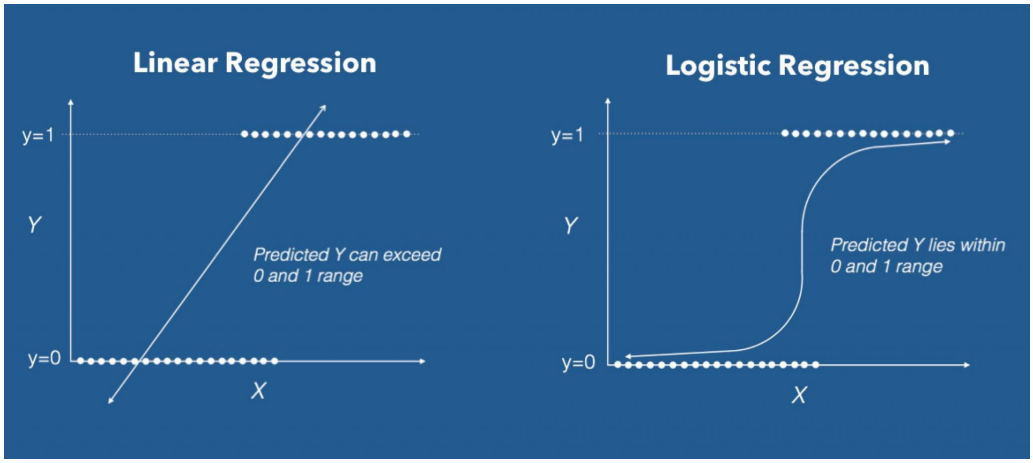


Figure 5.1: Linear and Logistic Regression Comparison [15]

We can see from the image above 5.1 the advantage of using logistic regression over linear regression. Logistic regression allows predicting values only between 0 and 1, which is precisely what we seek since we want to predict a probability.

5.1.3 MLP

A multi-layer perceptron (MLP) [16] is a kind of artificial neural network with several layers of interconnected neurons. These neurons generally use nonlinear activation functions, enabling the network to detect and learn complex patterns in data. MLPs play an important role in machine learning because they can capture nonlinear relationships, making them effective for tasks like classification, regression, and pattern recognition.

A neural network is made of connected nodes, called neurons, which are grouped into layers. Each neuron gets input signals, processes them with an activation function, and then sends an output signal that can go to other neurons in the network. The activation function decides what output a neuron gives based on its input. These functions add nonlinearity to the network, allowing it to learn complex patterns in data. The network is typically organized into several layers. The first one is the input layer, where data is introduced. Followed by several hidden layers where computations are performed. Finally, the output layer where predictions are made. Neurons in neighboring layers are linked by weighted connections. The weights determine how much influence one neuron's output has on the input of the next neuron. During training, the network learns to adjust these weights based on a training dataset. Each neuron also has a bias. On the figure below 5.2, you can find an example of MLP's structure.

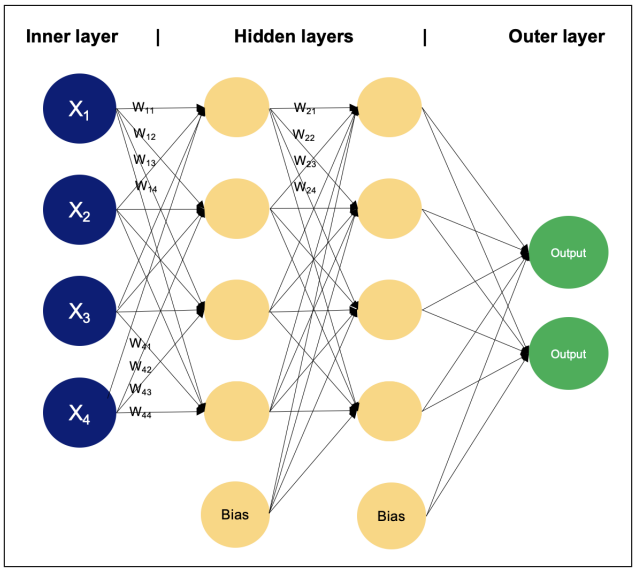


Figure 5.2: MLP scheme [16]

An activation function is used for each neuron. Below 5.3, you'll find the four main activation functions. The choice of this function can depend on the application, the model, and other factors. In our case, to predict probabilities, it's preferable to use the sigmoid function as the output is between 0 and 1.

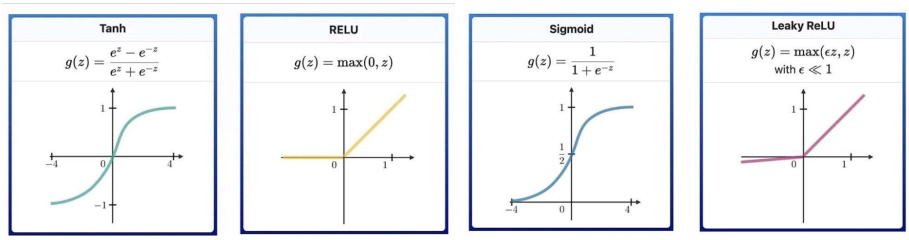


Figure 5.3: Activation functions [17]

Let called σ the sigmoid function and let y_1, y_2, y_3 and y_4 the nodes of the first hidden layer. We can compute their value as :

$$y_j = \sigma(\sum_{i=1}^{4} w_{i,j} x_i) \text{ for } j \in \{1, .., 4\}$$

The same process can be made for every layers. The number of neurons in the output layer depends on the task. In binary classification, there may be either one or two neurons depending on the chosen activation function. They represent the probability of belonging to one class.

Neural networks are trained using feedforward propagation and backpropagation. In feedforward propagation, data moves through the network layer by layer, with each layer processing the data and passing it to the next one. Backpropagation is an algorithm that trains neural networks by adjusting weights and biases to reduce the loss function. The loss function measures how good the model's predictions are. It helps the training process by indicating how well the network is learning. The

goal of training a neural network is to minimize this loss by finding the best weights and biases. This process is driven by an optimization algorithm called stochastic gradient descent. If you want to know more about the training process, you can refer to this source [16].

5.1.4 Random Forest

Random Forest [27] is a well-known ensemble learning technique in machine learning, useful for both classification and regression. It creates multiple decision trees during training and combines their predictions to improve the accuracy and stability. The term "forest" in Random Forest refers to the set of these trees, each uses a random subset of the data and features.

Random Forest works by building multiple individual decision trees, each trained on a random subset of the training data. Furthermore, at each split in the trees, only a random subset of features is used. This randomness helps to create varied trees, and when these trees are combined, they tend to reduce overfitting and improve generalization.

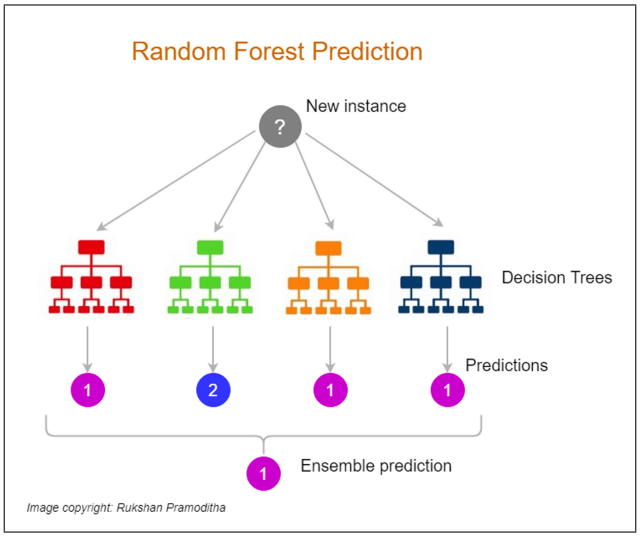


Figure 5.4: Random Forest Algorithm [18]

A decision tree[7] is built by recursively splitting the dataset based on the values of the features. At each step, the algorithm chooses the feature and the split point that best separates the data into different groups or reduces the impurity measure. Starting with the entire dataset, the algorithm considers all possible splits and selects the one that maximizes the separation or reduces the impurity the most. This split becomes the root node of the tree. Then, the dataset is divided into subsets based on this split, and the process is repeated recursively for each subset. At each step, the algorithm evaluates all possible splits and chooses the one that best separates the data. This process continues until a stopping criterion is met, such as reaching a maximum tree depth or when splitting doesn't improve the model's performance.

The resulting tree is a hierarchical structure where each node represents a decision based on a feature, and each leaf represents a class label or a regression value.

5.1.5 XGBoost

XGBoost [8] (eXtreme Gradient Boosting) is a powerful and scalable machine learning library that is widely used for supervised learning tasks, such as classification and regression. It is an implementation of gradient boosting algorithms [19] that are designed to be highly efficient, scalable, and accurate. XGBoost has become popular due to its speed and performance. One of the key features of XGBoost is its ability to handle large datasets with a large number of features, making it suitable for a wide range of applications. It uses a combination of advanced techniques, such as gradient boosting, tree pruning, and parallel computing, to achieve high performance. XGBoost supports both classification and regression tasks, and it provides various hyperparameters that can be tuned to optimize performance for different datasets and tasks.

5.2 Scoring Metrics used

To evaluate our different models, we will use metrics to assess their performance. Root Mean Square Error (RMSE) is one of the most used metric to assess the accuracy of a regression or prediction model. Mean Absolute Error (MAE) is also one of the most known metric. Both methods need the error which is the difference between the actual value and the predicted one as we can see on the plot below 5.5. In the case of classification model, we will use the Receiver Operating Characteristic (ROC) which help us to understand visually the performance of the model.

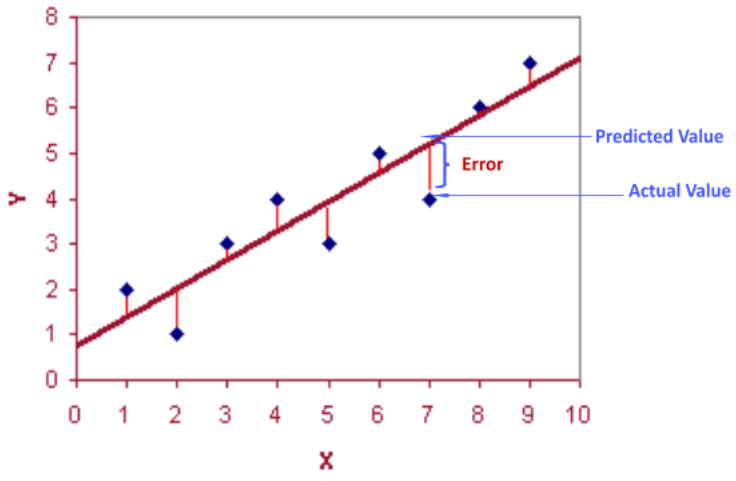


Figure 5.5: Residual Plot [22]

5.2.1 RMSE (Root Mean Square Error)

The Root Mean Square Error is a widely used metric to evaluate the performance of a regression model. Here is the formula :

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

where :

- n is the total number of observations,
- y_i is the true value of the target for observation i,
- $\hat{y}_i$ is the predicted value for observation i.

Here are the advantages and disadvantages of using RMSE: Advantages:

- Easy to interpret
- Sensitive to large errors

Disadvantages:

- Sensitive to outliers
- Does not take into account the error direction
- Sensitive to Overfitting

Despite its drawbacks, RMSE remains one of the most used metric due to its simplicity and interpretability, but it's important to consider its limitations when using it.

5.2.2 MAE (Mean Absolute Error)

Here is the formula :

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where:

- n is the total number of observations,
- y_i is the true value of the target for observation i,
- $\hat{y}_i$ is the predicted value for observation i.

Mean Absolute Error (MAE) is a popular metric because, like RMSE, the units of the error score match the units of the target value that is being predicted. Unlike the RMSE, the changes in MAE are linear and therefore intuitive. RMSE punish larger errors more than smaller errors. This is due to the square of the error value. The MAE does not give more or less weight to different types of errors and instead the scores increase linearly with increases in error.

5.2.3 ROC

A Receiver Operating Characteristic (ROC) curve is a graph that shows the performance of a classification model at all classification thresholds. This curve represents two parameters:

- True Positive Rate (TPR)
- False Positive Rate (FPR)

The True Positive Rate (TPR) is equivalent to recall. It is defined as:

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

The False Positive Rate (FPR) is defined as:

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

An ROC curve plots the TPR and FPR values for different classification thresholds. Decreasing the classification threshold classifies more items as positive, increasing both the number of false positives and true positives.

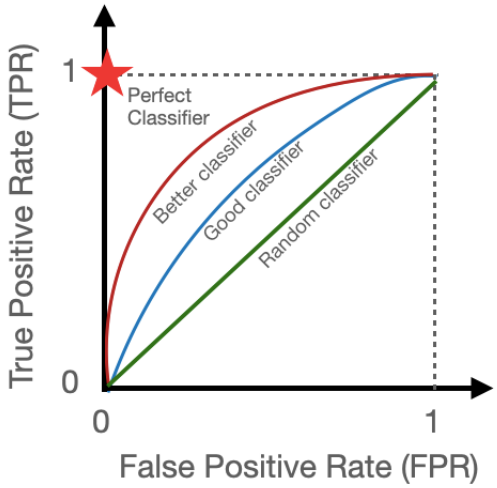


Figure 5.6: ROC Curve and AUC [21]

The Area Under the Curve (AUC) is a metric used to quantify the performance of a binary classification model based on the Receiver Operating Characteristic (ROC) curve. It represents the area under the ROC curve. AUC ranges from 0 to 1, where:

- AUC = 0.5 indicates that the model's predictions are no better than random guessing.
- AUC > 0.5 indicates that the model performs better than random guessing, with higher values indicating better performance.

- AUC = 1 indicates that the model makes perfect predictions.

In summary, the AUC provides a value that summarizes the model's ability to distinguish between positive and negative classes across all possible classification thresholds.

5.3 Model Assesment

To create strong and reliable models, it's essential to use a disciplined approach. In this section, we'll explain how to use correctly the data, the process for selecting the best hyperparameters, and discuss how to prevent overfitting.

5.3.1 Splitting Dataset

In machine learning, to build a good model we have to split our data into different sets. The train set, validation set and test set refer to different subsets of data used in the process of building, tuning, and evaluating machine learning models. These sets play distinct roles in training and validating a model's performance.

- The *train set* is the subset of data used to train a machine learning model. It contains the examples the model learns from. The model's parameters are adjusted during this training phase to minimize the error or loss function.
- The *validation set* is a separate subset used to evaluate the model during the training process. It gives information on the model's performance and adjusts hyperparameters without influencing the training itself. The validation set is crucial to avoid overfitting, as it indicates how well the model generalizes to unseen data.
- The *test set* is the final subset used to assess the model's performance. This data has not been used in training or validation. That provides an unbiased assessment of how well the model performs on unseen data. The test set is critical for estimating the real-world performance of a model and checking that it has not overfitted to the training or validation data.

In the case of our project, we will use the 2015-2016 season from the 5 major European leagues : Bundesliga, Ligue 1, LaLiga, Serie A and Premier League. This season is the only one that is free to use from StatsBomb [3]. Then we have to split these data into several parts in order to avoid overfitting. As we will create several regression models that are linked together we have to be sure that we do not use the same data for the training of different models. Indeed some models will use the output of another model as input, so if we use the same data for the training of these models, we will have a bias in the evaluation of the model.

In the next chapters, we will use these matches in order to train and test our models. These models will be presented in the next chapters. The *scored model* and *conceded model* from the VAEP section 6.2.1 are trained on the Bundesliga and Ligue

1. The chemistry scores from the *Final Model* 6.2.5 are trained on the first half of the season of the Premier League, LaLiga and Serie A. Finally the Validation chapter will use the second half of the season of the Premier League, LaLiga and Serie A.

5.3.2 Cross-Validation

To create a more robust model, we also use cross-validation. Cross-validation is a technique used to assess the performance and generalization ability of a machine learning model. It divides the dataset into multiple subsets and systematically trains and validates the model on these different subsets. Cross-validation is particularly useful when the dataset is limited in size, allowing for a more robust evaluation of a model's performance.

K-Fold cross-validation is one of the most known methods. The *train set* is divided into k distinct subsets. This is usually done randomly to ensure a representative distribution of the data in each fold. In each iteration, one of the k folds is used as the validation set, and the remaining k-1 folds are merged to create the train set. The model is trained on the train set and validated on the validation set.

5.3.3 Grid Search

Grid search is a technique used in machine learning to find the best combination of hyperparameters for a model. Hyperparameters are parameters that are not learned during training but are set before the training process starts. Examples include learning rates, regularization strengths, and the number of layers or units in a neural network. Grid search operates by exhaustively searching through a specified parameter grid, trying every possible combination of hyperparameters to identify the one that results in the best performance. The main drawback is the computational cost as it builds a model for every combination of hyperparameters. The selection of the hyperparameters may improve the model, this is a very important choice. That's why the Grid Search is very useful.

Chapter 6

Estimating parameters of the model

This chapter is one of the most important parts of this research. The goal is to assign values to the different parameters seen in the *Complete Model* 4.5. We will first focus on the player scores, then we will see how to estimate the edge scores. Finally, we will see the weighting scheme, which gives more or less importance to certain edges in the graph. All these steps are really important to build a strong *Complete Model* 4.5. All these parameters have to be defined and quantified before solving the *Complete Model* 4.5.

6.1 Player Scores

The goal of this section is to attribute a score to each player for each position. It corresponds to $c_{i,j}$ in the *Complete Model* 4.5. This is a very important step because the score of the player is one of the most important parameters of the model. The FIFA dataset [9] provides for each player a lot of statistics as the defending qualities, the attacking qualities, the physical qualities, etc. These statistics are used by FIFA to attribute a score to each player. Given that each player has a fixed position in the dataset, we do not know the score that the player would have at another position. The goal is to create a model that can predict the score of a player at a given position. The player's maximum score is 99. For example, a player like Lionel Messi has a score of 91 as a right winger, this is given in the dataset. But if the same player plays as a striker maybe his score would be 88. The goal is to design regression models to evaluate each player in every position, we have to create one model per outfield position. Given that the score of each player at their initial position is already known, this score is computed by a model designed by FIFA themselves but they do not make it public. That's why we have to create our own model in order to approximate their model as best as possible to compute the score of every player at every position and not be limited to the initial position.

6.1.1 Splitting Dataset

To train our models, we will first need to create a dataset of players for each position. That means that we will create a new dataset containing only the players that play at a given position. After that, we need to split each dataset into a part that will be used to train the model and another part that will be used to evaluate the model. This step is important to evaluate if our model overfits or not. Indeed, we want to ensure that our model can predict scores accurately even for new players. Depending on the position assigned to the model, we have more or less data available. Indeed, some positions are rarer than others. The table 11.1 in the annexes contains the size of the training and test set for the different positions. The target is the score of the player provided in the FIFA dataset and the features are all the statistics provided by the same dataset also.

The FIFA dataset was built with the different statistics and the global score was computed by the *FIFA model* . Their model is a blackbox for us, we do not know how they compute the score of a player from the statistics. That's why we have to create our own model to predict the score of a player at a given position.

Our final dataset for a given position looks like the following table 6.1, the values are just here to give an example of the structure of the dataset. The target is the global score computed by the *FIFA model* 6.1.1.

Player	**Attacking**	**Defending**	**Physic**	**...**	**...**	**Pace**	**Target/Score**
Player 1	85	70	80	...	...	90	88
Player 2	78	85	82	...	...	85	83
Player 3	90	60	75	...	...	95	87
Player 4	88	75	78	...	...	92	85
Player 5	80	80	85	...	...	88	86
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 6.1: Features and Target/Score for Different Players

6.1.2 Feature Selection

Given that the dataset contains a lot of statistics, we decided to keep only the 10 most correlated features with the target. These features are of course position-dependent. That means that the features kept for predicting the global score of a striker are not the same as the features used for a center-back.

For example here are the radar plots 6.1of the features used for ST and CB, we can notice that the features used are really different as expected. As Messi is a more offensive player, we can notice that the values are clearly higher for the features used for the striker position.

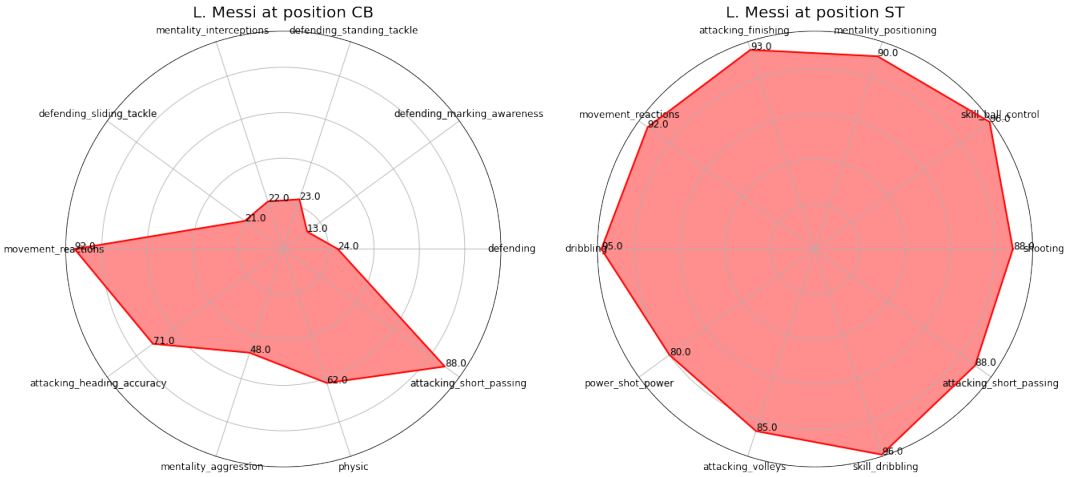


Figure 6.1: Comparison of the features used

6.1.3 Linear Regression

Then we can analyze the linear regression model for a fixed position, in this case CB, by analyzing the MAE (Mean Absolute Error) and the prediction plot on the test set which compares the value of the prediction and the true value.

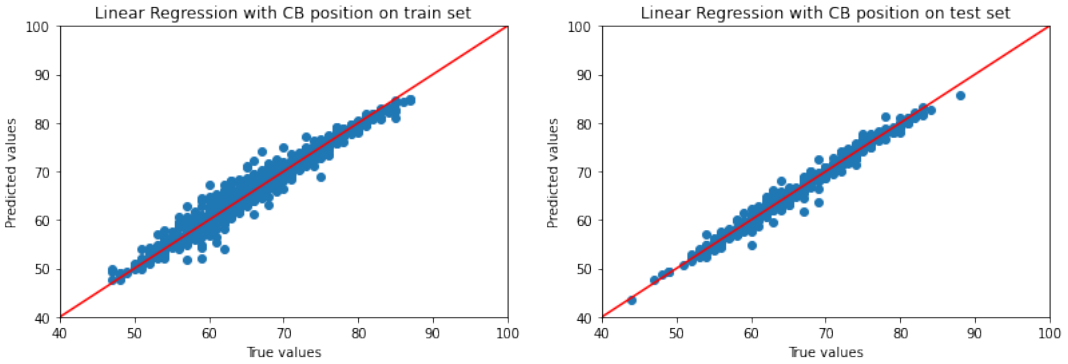


Figure 6.2: Mean absolute error on train set : 0.68 and Mean absolute error on test set : 0.67

Given that the objective is to predict scores up to 99, we can be satisfied with the model found, given the low value of MAE. By comparing the plots and the MAE, we can conclude that the model doesn't overfit which is a very important fact. Given that FIFA uses a model to compute the scores, it's expected that we'll be able to predict accurately. As there exists a true relation between the features and the target.

In the table 11.2, you will find the RMSE and MAE values for the different models on the train and test set and for each different position. Whatever the position, the results are more or less the same. The MAE is around 1.2 on average and RMSE is around 1.5. The very good thing is that the models give the same error on the train and the test set which means that the models don't overfit.

An important point to note is that the scores of the players provided in the FIFA dataset are always integers, however, our model predicts continuous values. So even though there is a true model, *FIFA model*, on their side to calculate the scores, due to rounding to the nearest integer, it complicates the task for our regression model to approximate the *FIFA model* as best as possible.

6.1.4 MLP

Although the previous model already provided very good results, it could be interesting to try a second model to see if we can achieve even better performance. A widely used model is the MLP. However, to obtain a good model, it's important to perform hyperparameter tuning. To achieve that, we use cross-validation and grid search combined.

Unfortunately, the results obtained 11.3 aren't better than those obtained with the Linear Regression. As you can see in the figures below 6.3, the results are really close to those of the previous ones for the CB position but for some positions, the results are worse.

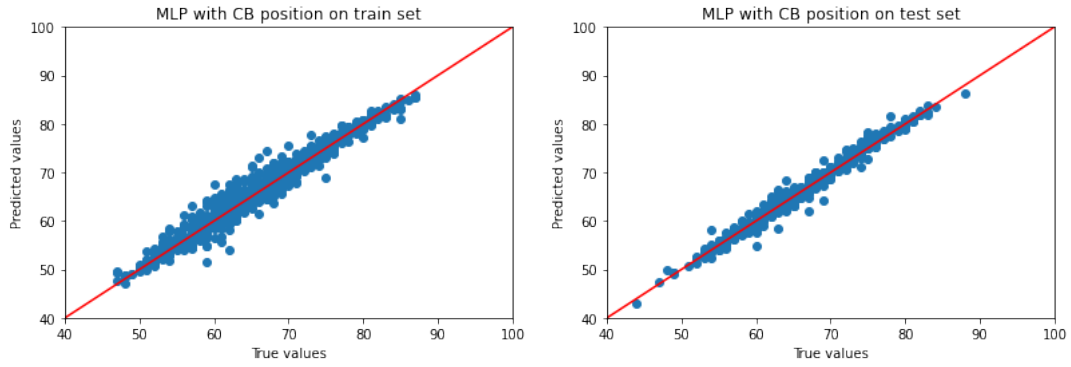


Figure 6.3: Mean absolute error on train set : 0.72 and Mean absolute error on test set : 0.69

Since the MLP doesn't success to obtain better results and that the initial results were already good, we decided to use the Linear Regression as our final model for predicting the scores of the players at a given position.

6.2 Chemistry Scores

The other important aspect of determining parameters is the score associated with the edges. Here, the goal is to determine the value of the parameters $e_{i,j}$ from the objective function of the *Complete Model* 4.5. Unlike the players' scores, where we already had a target established in the FIFA dataset [9], this time we have to define it ourselves. The challenge is to first define the chemistry between 2 players then define how we can compute it. To quantify the chemistry between a pair of players, we will draw upon certain elements from the document proposed by Lotte Bransen and Jan Van Haaren [1], as presented earlier in the State of the Art chapter 2. The goal will be to compute the chemistry from a dataset containing a large number of matches. We will compute the chemistry score for each pair of players who have played together thanks to the data from past matches. Finally we have computed the chemistry score for each pair of players who have played together, we will use this data to train a model that will predict the chemistry score for pairs of players who have never played together.

As we have computed the chemistry score for each pair of players who have played together, it may seem useless to create a model to predict this score. However, the goal is to predict also the chemistry score for pairs of players who have never played together. Given that both players have never played together, we do not have a chemistry score computed from the last matches. That's why we have to create a model to predict the chemistry score for these pairs of players.

6.2.1 VAEP

The first thing to introduce is the VAEP (Valuing Actions by Estimating Probabilities) [1], this framework was introduced recently and allows us to quantify how good or bad an action is. To achieve this, we need to introduce two different models that predict, from a given game situation, the probabilities of a goal being scored or conceded in the next 5 actions. As reminder an action could be a pass, shot, dribble and more. Thanks to the StatsBomb [3] Python package, we have free access to the data from the top 5 leagues for the 2015/2016 season. For these models, we will use only Bundesliga and Ligue 1 to keep some data unused for future models. Therefore, we will retrieve all the actions that occurred in matches throughout the entire season. We convert our match event data into the SPADL (Soccer Player Action Description Language) with the help of the *socceraction* [2] python package. You can find in the annexes all the variables used by the SPADL format 11.1, but in short, it contains information about the type of action, who performed it, and where it occurred. We will now create our game states. In fact, we are not just considering the current state of the match but also looking at the previous 3 actions. Indeed, the previous actions can also have an impact. Therefore, we will use these game states as features for our models. The targets are binary variables that indicate whether a goal was scored or conceded in the next 5 actions. Each row of the dataset corresponds to a unique gamestate with the binary target variable corresponding. By splitting our

dataset into a train and test set and by training a XGBoost [8] model for each case, we can analyze the results on the plot below6.4.

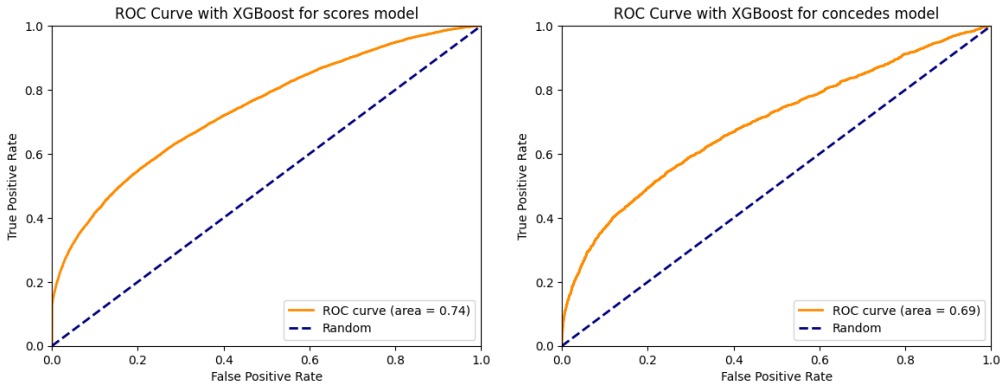


Figure 6.4: Comparison of the ROC between the models

First, we can notice that the AUC of the first classifier is slightly better than the second. We can see that both models perform better than random guessing. AUC=0.5 represents random guessing, so the higher the AUC value, the better the model performs. The random classifier, making predictions by chance, would have an AUC close to 0.5. Therefore, both models, with AUC values higher than 0.5, show some level of ability to distinguish between positive and negative instances.

Let's introduce the metrics that are based on the models above. Let's first introduce a_i as the i-th action in the data and S_i as the game state at the moment when the action a_i is played.

- The change in short-term scoring probability from the pre-action to the post-action gamestate : *Offensive value* (OV)
- The change in short-term conceding probability from the pre-action to the post-action gamestate : *Defensive value* (DV)
- The Valuing Actions by Estimating Probabilities : *VAEP*

You can find the different formulas to compute these metrics on the figure below 6.5. You can notice that *VAEP* is the difference between the *offensive value* and *defensive value*. *VAEP* offers a compromise between the offensive and defensive impact of an action.

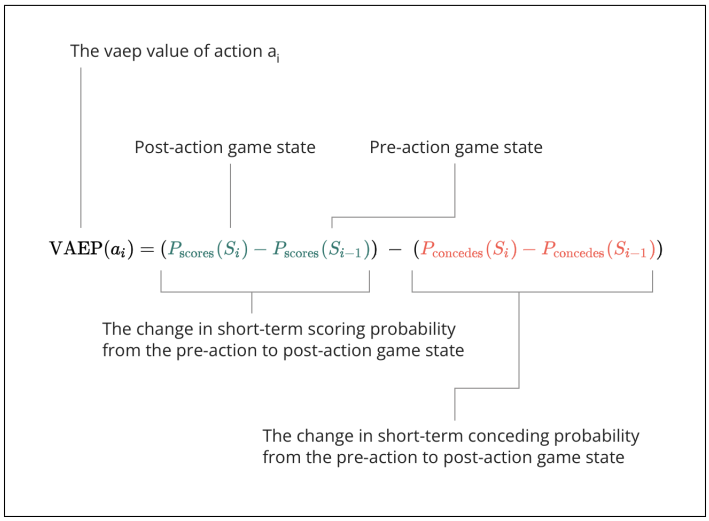


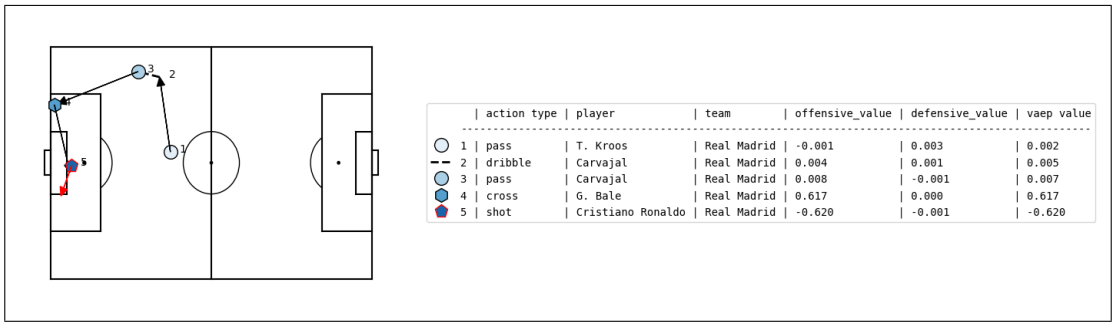
Figure 6.5: VAEP formula [29]

Here is the table 6.2 that contains the top 5 players in LaLiga by VAEP value. We can see that the top 5 players are all well-known players and that the values are really high. This is a good sign that the model is working well. The VAEP value is computed as the sum of the VAEP values for all actions performed by the player during the season. We can notice that the top 5 players are all offensive players. In the last column, we can see the VAEP value by minutes. This value is computed by dividing the VAEP value by the total number of minutes played by the player during the season. This allows us to compare players who have played different amounts of time. Some players may have a high VAEP value because they played more minutes, but the VAEP value by minutes allows us to compare them more fairly.

Player Name	VAEP Value	VAEP Value by minutes
Luis Suárez	26.460	0.0084
Lionel Messi	25.232	0.009
Gareth Bale	21.529	0.012
Cristiano Ronaldo	18.165	0.0057
Antoine Griezmann	17.895	0.0059

Table 6.2: Top 5 Players in LaLiga by VAEP Value

Finally, let's analyze a practical example that we can visualize with the *mplotsoccer* python package [6]. Here is a sequence of actions 6.6 performed by Real Madrid. You will find information such as the player's name, the type of action performed, and on the pitch, you will see the starting and ending points of each action. In the table, you will find the values of *offensive value*, *defensive value*, and *VAEP*. We can notice that the cross performed by Bale increased the probability of scoring a goal by more than 0.6. While Ronaldo's last action, where he missed his shot, decreased the probability of scoring a goal by more than 0.6 too.



	action type	player	team	offensive_value	defensive_value	vaep value
1	pass	T. Kroos	Real Madrid	-0.001	0.003	0.002
2	dribble	Carvajal	Real Madrid	0.004	0.001	0.005
3	pass	Carvajal	Real Madrid	0.008	-0.001	0.007
4	cross	G. Bale	Real Madrid	0.617	0.000	0.617
5	shot	Cristiano Ronaldo	Real Madrid	-0.620	-0.001	-0.620

Figure 6.6: Example of action

6.2.2 Building Targets

Now that we have presented the different metrics and what they represent, we will focus on the chosen targets to quantify the chemistry between 2 players. Throughout an entire season, for each pair of players who have played together, we will analyze their performances and the impacts they have had on their team's results. For each pair, we will obviously take into account the time spent playing together on the field in order to have meaningful metrics. These metrics obtained from real data will then be used to train models that can predict these metrics for pairs of players who have never played together, for example.

Offensive Value

The first metric created is called *offensive value*. This metric is based on the previous results and on the paper of Lotte Bransen [1]. A match is described as a sequence of actions $\{a_1^p, \ldots, a_n^q\}$, where n is the number of actions in the match and $p \in P$ refers to the player who performed the first action of the match and $q \in P$ refers to the player who performed the last action of the match, with P being the set of all players in the match. Each action a_i^p can be one of five types: pass, cross, dribble, take-on, or shot. Formally, we define the j-th interaction between two players p and q in a match m as a subsequence of two consecutive actions as follows:

$$I_j^m(p,q) = (a_i^p, a_{i+1}^q) \tag{6.1}$$

where a_i^p represents the i-th action in match m performed by player p.

Moreover, we extend the concept of the *offensive value* from individual actions to interactions between players. We calculate the *offensive value* for an interaction by summing the *offensive values* for the two following actions. We define this extension as :

$$OV(I_j^m(p,q)) = OV(a_i^p) + OV(a_{i+1}^q) \tag{6.2}$$

We determine the combined offensive value (COV) for a pair of players during a match by summing the offensive values (OV) for their interactions. Formally, we define $COV_m(p,q)$ for a player pair (p,q) in a match m as follows:

$$COV_m(p,q) = \sum_{i=1}^{k} OV(I_j^m(p,q)) + \sum_{i=1}^{l} OV(I_j^m(q,p)) \tag{6.3}$$

where k represents the number of interactions where player p initiates the first action and player q initiates the second action, and l represents the number of interactions where player q initiates the first action and player p initiates the second action.

In order, to make relevant comparison between pairs of players who spent different amounts of time on the pitch together within a season, we introduce $SCOV(p, q)$, which represents the season combined offensive value for a pair of players (p, q) in that season. Formally, we define $SCOV(p, q)$ as follows:

$$SCOV(p, q) = \frac{\sum_m COV_m(p, q)}{\sum_m MINS_m(p, q)} \tag{6.4}$$

We sum over all matches m in a season, where $MINS_m(p, q)$ represents the total number of minutes that players p and q spent together on the pitch during match m.

Defensive Value

This second metric is actually the same as the first one but for the defensive value. In deed, there are two aspects in football: scoring goals and preventing the opponent from scoring. That's why a good pair of players can be one that prevents the opponents from scoring goals. With the same development as before, we can obtain a formula for the seasonal combined defensive value ($SCDV$) :

$$SCDV(p, q) = \frac{\sum_m CDV_m(p, q)}{\sum_m MINS_m(p, q)} \tag{6.5}$$

Balanced Value

As mentioned before, the goal of a football match for a team is to score one more goal than the opponent. By combining the two previous metrics, we can achieve a balance between scoring a lot of goals, which is good, and not conceding too many at the same time. We will train a model to predict for each pair of player the SCOV and another for the SCDV . Then, we will merge these results to create our final metric allowing us to quantify the chemistry between two players. Let's create the seasonal combined balanced value :

$$SCBV(p, q) = SCOV(p, q) + SCDV(p, q) \tag{6.6}$$

This score will be used to quantify the chemistry between 2 players, but we won't train a model to predict this score because it depends on both offensive and defensive skills. That's why it is difficult to predict it.

6.2.3 Features Engineering and Selection

Now that we have decided and presented the targets we will use for our models, it is important to decide which features will be chosen to try to predict these targets. It is an important step because if we don't choose the features correctly, it will make the task difficult for the model. As we want to predict the chemistry for a pair of players, we will take as features the physical and technical characteristics of

each player. Indeed, with this information, the model could learn certain patterns between pairs of players with similar profiles. We can also add some binary features to indicate if both players have the same nationality, and speak the same language,...

Finally, we will proceed a feature selection to keep only the relevant features for each target. This selection is based on keeping the 10 best features correlated with the target. We made also a verification of the sense of the features to be sure that they are relevant.

6.2.4 Model Prediction

As explained earlier, we have to create a model to predict the SCOV and SCDV for each pair of players. By testing different types of models, we can determine which one is the best for our task. On the figure below 6.7, you can find the plot of the prediction vs the true value for the SCOV . We can notice that the model has some difficulties to predict the values correctly, even if we can see a tendency as the values are increasing slightly. The selected model is the Random Forest because it has the best performance even if other models have similar results.

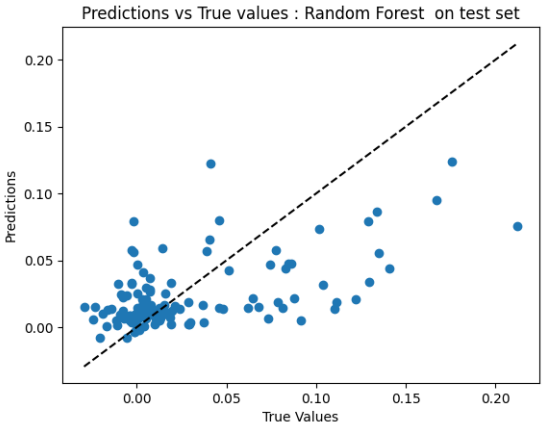


Figure 6.7: SCOV prediction on test set

On the table below 6.3, you can find the RMSE and MAE values for the model above and the *dummy model* which returns always the mean of the target of the train set. We can notice that even if our model isn't perfect, it has better results than the *dummy model*.

	Random Forest	Dummy Model
RMSE	0.039	0.047
MAE	0.028	0.035

Table 6.3: Scoring Metrics Results for SCOV

The same process was applied to the SCDV. Unfortunately the results are poorer than the previous ones as you can see on the figure below 6.8.

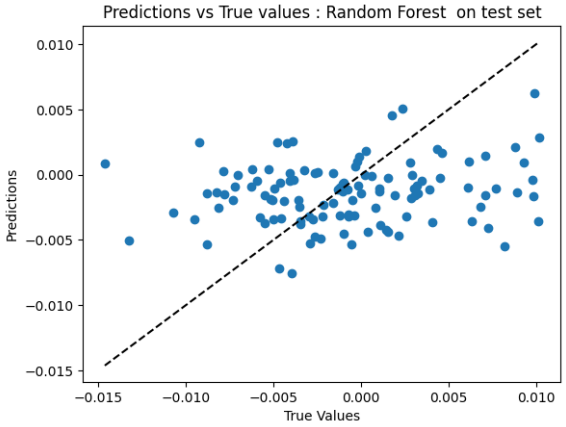


Figure 6.8: SCDV prediction on test set

The model is not able to predict the values, the RMSE and MAE of our model are more or less the same as the *dummy model*. As we have seen on the figure above 6.8, this result was expected. The table below 6.4 contains the RMSE and MAE values for the Random Forest model and the dummy model for the SCDV.

	Random Forest	Dummy Model
RMSE	0.012	0.011
MAE	0.009	0.009

Table 6.4: Scoring Metrics Results for SCDV

6.2.5 Final Model

As we have seen, we can't really rely on the random forest model to predict the defensive value and thus to predict the chemistry between two players. That's why we will introduce a new method to predict the chemistry. Most of the time, the coach has to choose between players who have already played together. That's why the prediction of the chemistry score can be replaced by the true score of the pair of players. In deed, we can use the chemistry score obtained from the last season or the last games. The prediction of the chemistry score is therefore not necessary. The prediction of the chemistry remains important for new pairs of players as the coach does not have any information about the chemistry between these players.

We aim to predict $SCBV\_pred(p,q)$ using the following conditional model:

$$SCBV\_pred(p,q) = \begin{cases} SCBV(p,q) & \text{if time\_played}(p,q) \geq 855, \\ SCOV\_pred(p,q) + SCDV\_pred(p,q) & \text{if time\_played}(p,q) < 855. \end{cases}$$

In order to create the models for SCDV and SCOV, we will use the Random Forest model trained on the games of the first half of the season. SCBV will be computed by summing the SCOV and SCDV values for the first half of the season. If a pair of player has played together for more than 855 minutes during the first

part of the season, then the *Final Model* will predict the value theyobtained for the first half of the season.

The time threshold of 855 minutes corresponds to half of the total time of a half season, in deed there are 19 games in a half season and each game lasts 90 minutes. This threshold is chosen to ensure that the model has enough data to predict the chemistry score. If the pair of players has played together for more than 855 minutes, we will use the real chemistry score. Otherwise, we will use the predicted chemistry score.

On the figure below 6.9 we can see the comparison of the chemistry score for each pair of players computed on the first and the second half of the season. The values for the second part of the season are the true ones and not the predicted ones. Unfortunately, we can notice that even if a pair of players get a high chemistry score for the first part of the season, they can have a bad chemistry score during the second part of the season and inversely.

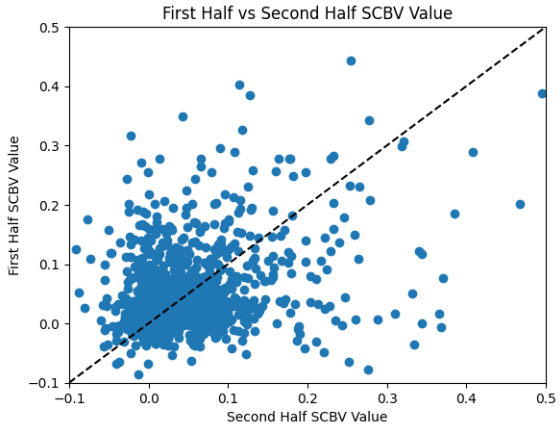


Figure 6.9: Comparison of the chemistry score between the first and the second half of the season

These results show that the chemistry between two players can change a lot during the season. This helps us also in order to understand why it is hard to predict chemistry from the skills of the players as the players have the same skills during all the season but obtained different chemistry scores between the first and second half.

An idea to avoid these fluctuations would be to compute the chemistry for a pair of players over several seasons. This would allow us to have a more stable chemistry score. Unfortunately, we don't have enough data to do this as StatsBomb [3] only provides free data for the 2015/2016 season.

6.2.6 Summary of the Chemistry

In this chapter, we have introduced different metrics : SCDV 6.2.2 ,SCOV 6.2.2 and SCBV 6.2.2. They are used to quantify the chemistry between two players. We have also presented the different steps to create the dataset and the models to

predict these metrics. The results obtained are not very satisfying, the models have some difficulties in predicting the values correctly. The main point is that it is quite difficult to predict the chemistry for players who have never played together. Indeed, the chemistry between two players is not only based on their physical and technical characteristics but also on their understanding of the game, their communication, their positioning, etc. These are all elements that are difficult to quantify and moreover there is also a randomness factor. That's why the models have difficulties predicting the values correctly. But for the players who have already played together, we can use their past performances to assign a chemistry score and then use it to compute the final score of the team. The importance of the chemistry prediction is for new pairs of players but most of the time the coach has to choose between players who have already played together.

That's why the prediction of the chemistry score can be replaced by the true score of the pair of players most of the time. Indeed, we can use the chemistry score obtained from the last season or the last games for example. The prediction of the chemistry score is then not necessary as we can compute it from past results. Unfortunately, we saw that the chemistry between two players can change a lot during the season. This helps us in order to understand why it is hard to predict chemistry from the skills of the players as the players have the same skills during the entire season but obtained different chemistry scores between the first and second half. The solution would be to compute the chemistry for a pair of players over several seasons, in order to get a more robust chemistry. This would give us a more stable chemistry score but we don't have enough data to do this as we have only access to the 2015/2016 season [3].

6.3 Weighting Scheme

Now we're going to discuss the parameter values chosen to give more or less importance to the weight of certain edges. These parameters corresponds to the $w_{i,j}$ in the objective function 4.3.2. As reminder $w_{i,j}$ is the weight of the edge between the i^{th} position and the j^{th} position. It's important to note that here we're not modifying the edge score we saw earlier 6.2, but we are modifying the weight that an edge has in the objective function. We'll call this set of parameters the weighting scheme.

6.3.1 Uniform Weighting Scheme

First of all, we'll use the most basic scheme available. We're going to assign the same weight to each edge, so we'll call it the *Uniform weighting scheme*. The weight between the i^{th} player and the j^{th} player is equal to :

$$w_{i,j} = \frac{1}{55} \tag{6.7}$$

We have normalized the weight in order to have : $\sum_{(i,j)} w_{i,j} = 1$ and the complete graph K_{11} has 55 edges. On the figure below 6.10, you can find two examples of the uniform weighting scheme where the width of the edge corresponds to the importance of the edge.

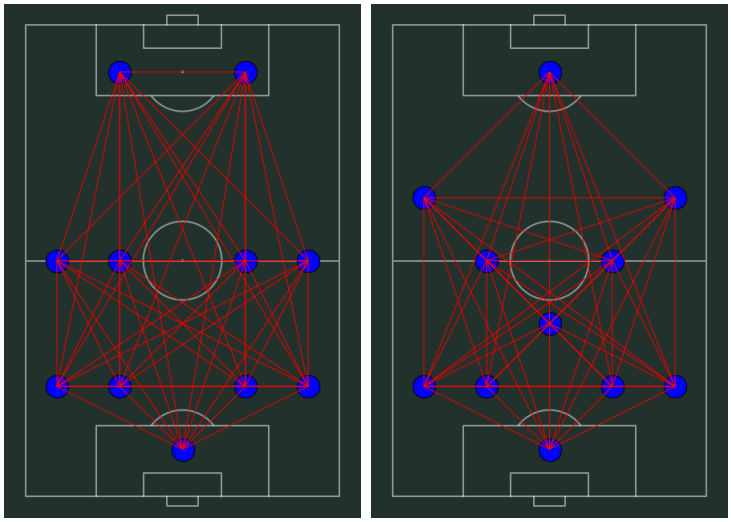


Figure 6.10: Uniform weighting scheme

The *uniform weighting scheme* is the most basic version possible. Although it has advantages due to its simplicity, it also has disadvantages such as giving equal importance to relationships between two players even if they are close or far apart on the field.

6.3.2 Distance Weighting Scheme

That is why we are introducing the *distance weighting scheme* . This scheme will assign weights inversely proportional to the distance between players on the field. So, the closer the players are, the greater the weight will be. You can verify this on the figures below 6.11, where the width of the edges corresponds to the weight assigned to that edge. The weight between 2 players at position (x_i, y_i) and (x_j, y_j) :

$$\tilde{w}_{i,j} = \frac{1}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \tag{6.8}$$

Finally we normalized these weights in order to have : $\sum_{(i,j)} \tilde{w}_{i,j} = 1$ and $w_{i,j}$ is the normalized version of $\tilde{w}_{i,j}$. For the objective function 4.3.2, we will use the $w_{i,j}$ and not the $\tilde{w}_{i,j}$. The normalized weight is computed as follows:

$$w_{i,j} = \frac{\tilde{w}_{i,j}}{\sum_{(i,j)} \tilde{w}_{i,j}} \tag{6.9}$$

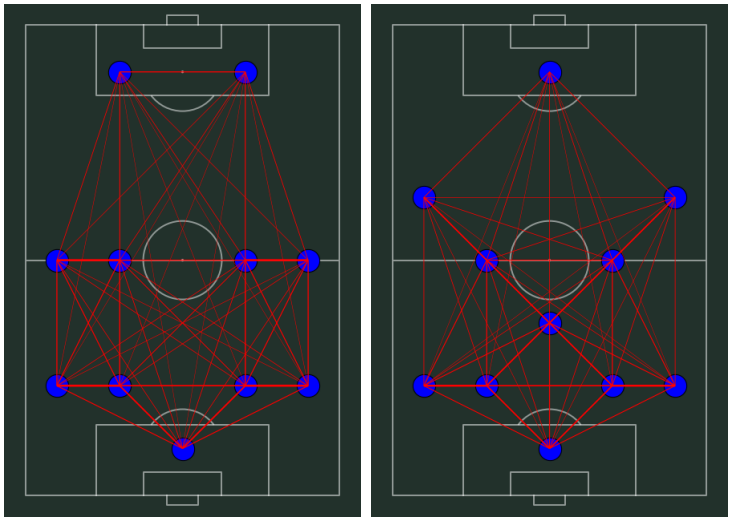


Figure 6.11: Distance weighting scheme

To conclude, we have introduced two different weighting schemes. The first one is the *uniform weighting scheme* where all the edges have the same weight. The second one is the *distance weighting scheme* where the weight of the edge is inversely proportional to the distance between the players. The *distance weighting scheme* is more complex to compute but it is more relevant as it gives more importance to the players who are close to each other on the field. It is important to recall that the weights are not the edge scores but the importance of the edge in the objective function 4.3.2. The *distance weighting scheme* will be used for the rest of the research as it gives more importance to the players who are close to each other on the field and this is more relevant for our task. The *distance weighting scheme* remains simple to compute and doesn't require a lot of computational power. It is very flexible and can be adapted to any formation.

Chapter 7

Validation

This section is really important in order to prove that the *Complete Model* 4.2.3 with all the parameters computed in the last chapter 6 is relevant. We will assess that the different weights and parameters like the player's scores 6.1 and the edge's scores 6.2 are well chosen. The aim here is to validate our *Complete Model* and then determine good values for the weights w_1 and w_2 present in the objective function 4.5 giving more or less importance to the score of the nodes or the edges. The goal is to prove that maximizing our team's score will give us a better chance of obtaining good results. For the moment we're maximizing a score without showing that this score is indeed relevant to show that a team works well or not. We need to prove that the score has a real impact.

First, we have access to the odds of sports betting for each match. The odds offer us a way to get the different probabilities of the outcomes of the match. We will consider the probability of the home team winning from sports betting odds as the target probability. For all the matches from the second half of the season 2015/2016 of LaLiga, Serie A and Premier League, we have the odds of the home team winning and the composition of both teams. From these compositions, we can compute the score of both teams for each match with the *Complete Model* 4.2.3. Finally, we will create a model that will take as input the scores of the teams and attempt to predict the probability that the home team will win. The model will be a Logistic Regression, where the target is a binary value indicating whether or not the home team has won. The model will allow us to determine the values of the weights w_1 and w_2 to weigh the importance of the player's scores and edges in the objective function of the *Complete Model* 4.5. Finally, we will validate the model by comparing the predicted probabilities for each match with the probabilities obtained from the odds. This is important to use probabilities of the home team winning and not to just use the results of the matches. Indeed, this is not always the best team that wins, there is a lot of randomness in football. That is why it is better to use probabilities to validate our model.

7.1 Betting odds

First of all, for this part of the research, we need to know the probability that the home team will win. Therefore the betting odds are very important for us as they give us a very good reference for the probabilities. First, we will briefly introduce the sports betting odds system. For each possible outcome of the match, a probability is assigned. In a football match, there are only 3 possible outcomes: victory for the home team, a draw, or victory for the away team. The odds are really good approximates of the game probabilities. The odds are computed by the bookmakers and they take into account many factors such as the team's form, the players' form, the injuries, etc. Then the bookmakers will update the odds according to the bets placed by the betters. If a lot of people bet on the home team, the odds of the home team winning will decrease [23]. Now, let's analyze a small example.

	Home	Draw	Away
odds	1.5	4	10

The sports betting odds are expressed as above. If you bet 100 euros on the home team winning and you win the bet, then you earn $100 \times 1.5 = 150$ euros, i.e. a 50% profit. Let's now translate these odds into probability. The probability that the home team wins the match is $\frac{1}{odds_{Home}} = \frac{1}{1.5} \approx 0.67$. By rewriting the table above, but this time taking into account the probabilities, we have:

	Home	Draw	Away
probability	0.67	0.25	0.1

If the probabilities were exact, then their sum should equal 1, as this is one of the main properties of probabilities. Checking reveals that the sum of these probabilities : $0.67 + 0.25 + 0.1 = 1.02$. We can observe that this sum is greater than 1. Now we can calculate the player return rate : $\frac{1}{1.02} = 0.98$. This is important to determine the margin that the bookmakers take.

Now, let's talk about the *true probabilities* of each outcome, without taking into account the margin taken by the bookmakers. Indeed, taking into account the bookmakers' margin in the odds does not give us the actual probabilities. To obtain them, we must multiply the probabilities derived from the odds by 0.98. The *true probabilities*, for our example, would therefore be:

	Home	Draw	Away
Odds Probability	0.67	0.25	0.1
True Probability	0.65	0.24	0.098

Therefore, we can conclude that these *true probabilities* are lower than those offered by the betting odds. In our case, this doesn't change the results significantly, but some bookmakers take very large margins, and this can have a huge impact.

7.2 Logistic Regression

The purpose of this section is to validate our model using Logistic Regression [26], but most importantly, it will enable us to determine the values of the weights w_1 and w_2 to weigh the importance of players' scores and edges from the *Complete Model* 4.5. Hence, we will train our Logistic Regression Model with 4 features: the player-related score for each team and the edge-related score for each team.

Let's say we are going to use the following features:

- X_1 : Player-related score for Team 1
- X_2 : Edge-related score for Team 1
- X_3 : Player-related score for Team 2
- X_4 : Edge-related score for Team 2

In our case, we consider the binary output Y as whether the home team wins or not. In the context of predicting the home team's victory, we can define the probability $P(Y = 1|X)$ that the home team wins, given the features X (score of the teams, current form, ...).

7.2.1 Split Teams Model

Hence, our first logistic regression model called *Split Teams Model* will have the following form:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\beta_3 X_3+\beta_4 X_4)}} \tag{7.1}$$

Where β_1, β_2, β_3, and β_4 are the coefficients of our model and β_0 the intercept. These coefficients represent the importance of each feature in predicting the probability of the home team winning. The name of the model comes from the fact that we split the features for both teams and thus the coefficients β are different for the home team and the away team. Our weights w_1 and w_2 will actually be the coefficients β of the logistic regression model. It's important to note that the value of these coefficients would be different for a team playing at home compared to playing away. Let find in the table below 7.1 the values of the coefficients for the home team and the away team. This table means that in function of we consider a team playing at home or away, we will use different coefficients w_1 and w_2 for the objective function of the *Complete Model* 4.5.

Parameter	**Home Team**	**Away Team**
w_1	β_1	β_3
w_2	β_2	β_4

Table 7.1: Weights translation for Home Team and Away Team

Given that the scores of players and the scores of edges can be of completely different orders of magnitude, it's important to scale our features. In this case, we will use one of the most known scaling methods:

Standard Scaling refers to the process of transforming data in order to has a mean of zero and a standard deviation of one. This is done by subtracting the mean and dividing by the standard deviation for each feature. Mathematically, the formula for standard scaling is:

$$z = \frac{x - \mu}{\sigma}$$

where:

- x represents the original data point,
- μ is the mean of the data,
- σ is the standard deviation of the data,
- z is the standardized value.

Standard scaling is often used in machine learning to bring different features onto a common scale. This is very important in order to improve the performance and stability of algorithms that are sensitive to the distribution and scale of data.

It is important to note that once we work with standardized features, the parameters of our *Split Teams Model* 7.2.1 are also computed on these standardized features. Therefore, they must be converted to non-standardized parameters to be used in our objective function. It's crucial to convert the model's parameters back to their original (non-standardized) form to understand their real-world interpretation.

In order to map the values from the standardized domain to the initial one, the coefficients have to verify this equation for all the data where n is the number of features :

$$P(Y = 1|X_{scaled}) = P(Y = 1|X) \tag{7.2}$$

$$\beta_0 + \sum_{i=1}^{n} \beta_i z_i = \beta'_0 + \sum_{i=1}^{n} \beta'_i x_i \tag{7.3}$$

By applying the variable transformation from standardization, we obtain:

$$\beta_0 + \sum_{i=1}^{n} \beta_i (\frac{x_i - \mu_i}{\sigma_i}) = \beta'_0 + \sum_{i=1}^{n} \beta'_i x_i \tag{7.4}$$

where :

- β'_0 is the non-standardized intercept from (β_0)
- β'_i is the non-standardized parameter from (β_i)

- σ_i is the standard deviation used to standardize the feature x_i
- μ_i is the mean of the feature x_i

By solving the above equation through identification, we obtain the values of the parameters β'_0 and β'_i.

$$\beta'_0 = \beta_0 - \sum_{i=1}^{n} \beta_i \times \frac{\mu_i}{\sigma_i}$$

$$\beta'_i = \frac{\beta_i}{\sigma_i} \qquad i \in \{1, \ldots, n\}$$

Dividing by the standard deviation reintroduces the scale of the original feature. This is crucial because it allows us to find the values of the coefficients to use in our objective function while using a logistic regression model that operates on standardized features.

Here is just a small example to show the mapping from the standardized space to the initial one. First here is the data for the example, we use only one feature :

X	$X_{\textbf{scaled}}$	**y**
-5	-1.581	0
-4	-1.265	0
-3	-0.948	0
-2	-0.632	0
-1	-0.316	0
0	0	1
1	0.316	1
2	0.632	1
3	0.948	1
4	1.265	1
5	1.581	1

Table 7.2: Scaling table

On the figures below 7.1, you can compare the results obtained with the standardized and non standardized data. The first graph is the logistic regression curve derived from the standardized values of X. The second graph is the curve obtained by converting the standardized parameters to non-standardized parameters, as explained earlier. We can check that the MAE are exactly the same in both cases. We can clearly see that the curve is identical in both graphs, but the scale on the x-axis is different. Here are the 2 functions :

$$P(Y = 1|X_{scaled}) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_{scaled})}}$$

$$P(Y = 1|X) = \frac{1}{1+e^{-(\beta'_0+\beta'_1 X)}}$$

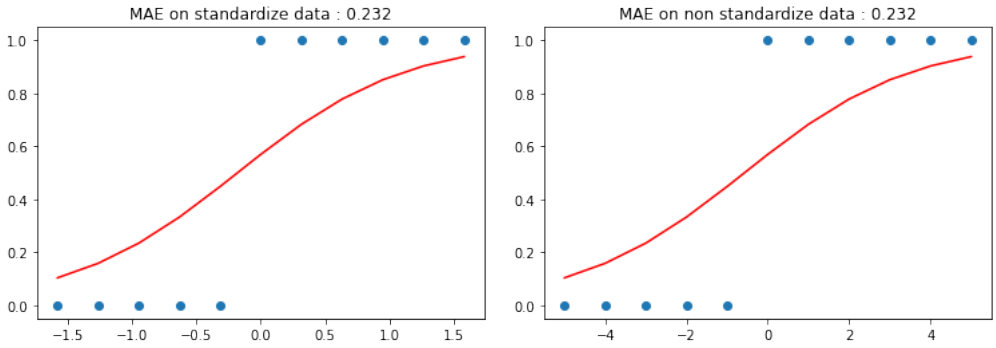


Figure 7.1: Comparison of the logistic regression curve on standardized and non-standardized data

The goal will be to compare the predicted probabilities with the probabilities derived from sports betting. It's important to notice that, as explained earlier, we need to consider the *true probabilities* from sports betting and not the ones that are displayed for the betting which includes the bookmaker's margin.

In order to avoid overfitting, we will use as data the second half of the season for the LaLiga, Premier League, and Serie A. We will then split these games in order to have a training set and a test set. The training set will be used to train the model and the test set to evaluate the model's performance. It is important to only consider the second half of the season because the first part of the season has been used to train the model to predict the chemistry score 6.2. We have used the *Final Model* 6.2.5 to predict the chemistry between players, most of them played together during the first part of the season.

The dataset will have the following form containing the different features and the binary target Y :

Match	X_1	X_2	X_3	X_4	**Y**
Match 1	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
Match 2	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
Match 3	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 7.3: Scores and Outcomes for Matches

A Logistic Regression model will be trained on the train set, in order to obtain our *Split Teams Model*. On the figure below 7.2, you can find the predictions of the model on the test set. The x-axis represents the *true probabilities* from sports betting and the y-axis the predicted probabilities from the *Split Teams Model* 7.2.1.

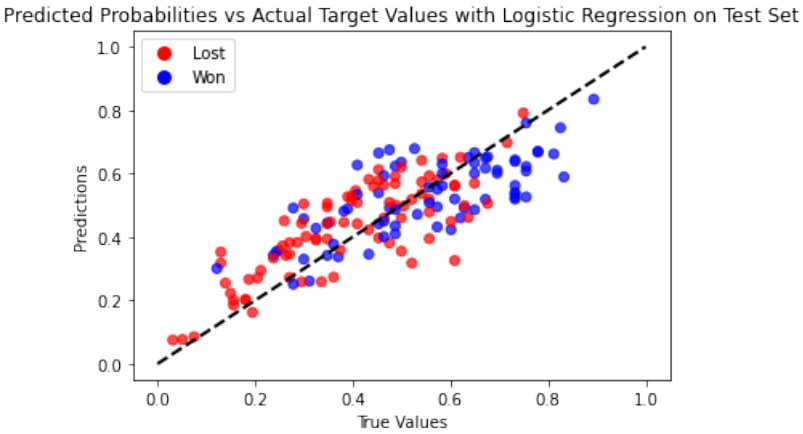


Figure 7.2: Predictions vs True Values on Test set

We can notice from the figure above 7.2 that the model performs quite well on the test set. The predictions are quite close to the *true probabilities* from bookmakers. The model can predict the probabilities of the home team winning based on the scores of the players and the edges of both teams. You can see the color of the points on the figure allowing us to know if the match was won by the home team or not.

In the table below 7.4, you can find the β parameters of the *Split Teams Model*. The standardized and non-standardized coefficients are presented. The standardized coefficients are used to predict the probabilities, while the non-standardized coefficients are used to calculate the team's score. The mapping between the two is done as explained earlier. We can compare the β coefficients from the home team and the away team. We can notice that the scale of the coefficients is the same for both teams but the sign is different. This is because the model is trained to predict the probability of the home team winning. The Non-Standardized coefficients are used in our objective function to calculate the team's score to give the true importance of the players and the edges.

Coefficient	**Standardized**	**Non-Standardized**
β_0	-0.19	0.64
β_1	0.39	0.01
β_2	0.10	0.61
β_3	-0.42	-0.01
β_4	-0.28	-1.69

Table 7.4: Standardized and Non-Standardized Coefficients

Let's now calculate the mean of the nodes scores and the edges scores for the home and away teams. The mean scores are then ponderated by their non-standardized coefficients in order to obtain their true importance. The results are presented in the table below 7.4.

We can notice that the nodes score is more important than the edges score but the edges score isn't negligible. The mean score for the home team is positive while

the mean score for the away team is negative. This is due to the fact that the model is trained to predict the probability of the home team winning. We can notice that the mean score for the home team is lower than the mean score for the away team but there is a non-standardized intercept that is positive.

Then if we compute : $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 = 0.64 + 8.53 + 0.29 - 8.85 - 0.8 = -0.19$ we can notice that the mean prediction $P(Y = 1|X) = \frac{1}{1+e^{-(-0.19)}}$ is equal to 0.45 which is the probability of the home team winning. Finally, on the figure below 7.3, you can find the distribution of the results on the train set. We can notice that the fraction of winning games for the home team : $\frac{166}{166+99+99} = 46\%$. This is quite close to the probability of the home team winning.

We can finally discuss the fact that the away team has a higher mean score than the home team even by taking into account the non-standardized intercept. That doesn't mean that this is more likely to win when we are playing away. Indeed, in this case, a draw is considered as a defeat for the home team so the away team has more chance to win. As reminder, Y is a binary variable indicating whether the home team wins or not. So if the match is a draw, this result is considered as a defeat for the home team. This is why the away team has a higher mean score than the home team.

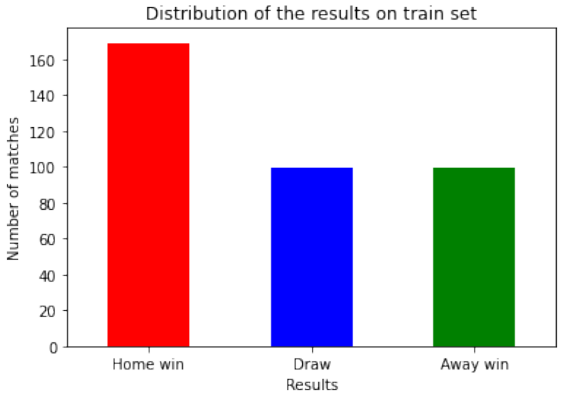


Figure 7.3: Distribution of the results on the test set

	Nodes	Edges	Total
Home	8.53	0.29	8.82
Away	-8.85	-0.80	-9.65

Figure 7.4: Mean Scores for Home and Away

7.2.2 Equal Weights

Our last model, the *Split Teams Model* 7.2.1 lets the freedom that the importance of the players and the edges can be different for the home or away team as we have a coefficient for each score. We can also consider that the importance of the players and the edges is the same for the home and away team. In this case, we will use the same coefficient for the home and away team. We just have to add a constraint to our model to ensure that the coefficients have the same absolute value but sign opposite for the home and away team :

$$\beta_1 = -\beta_3$$
$$\beta_2 = -\beta_4$$

The new logistic regression model called *Equal Weights* will have the following form :

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1(X_1 - X_3) + \beta_2(X_2 - X_4))}} \tag{7.5}$$

where β_1 is the coefficient for the player-related score and β_2 is the coefficient for the edge-related score.

We have to create new features for this new model, we will use the difference between the player-related score from the home team and the player-related score from the away team and the difference between the edge-related score from the home team and the edge-related score from the away team. The main point is that we solve the same problem as before but with a constraint that the coefficients for the home and away team are the same but with opposite signs. This will allow us to have a more robust model in order to quantify the importance of the players and the edges.

Coefficient	**Standardized**	**Non-Standardized**
β_0	-0.19	-0.18
β_1	0.58	0.01
β_2	0.26	1.13

Table 7.5: Standardized and Non-Standardized Coefficients with Equal Weights

We can notice on the table above 7.5 that the coefficients are quite similar to the previous model 7.4. We can use these coefficients to calculate the mean scores for the home and away team. The results are presented in the table below 7.6. These scores are the mean score for the home and away team ponderate by the non-standardized coefficients. This table allows us to understand the impact of the nodes and the edges in the objective function 4.5.

	Nodes	**Edges**	**Total**
Home	8.68	0.54	9.22
Away	8.68	0.54	9.22

Table 7.6: Mean Scores for Home and Away with Equal Weights

Chapter 8

Applications

In this chapter, we will present different applications of the model. We will first analyse how to use the model to try to beat the bookmakers. Then, we will discuss how to use the model in order to recruit new players. Finally, the model is also helpful for the national teams, we will see how a national team can be built using the model and compare the results with the true national teams aligned during the Euro 2016. All the applications will be based on the *Complete Model* 4.5 with the *Distance Weighting Scheme* 6.3.2 and the coefficients w_1 and w_2 from the objective function of the *Complete Model* giving more or less importance to nodes or edges are determined by the method *Equal Weights* 7.2.2. The score of the players are computed as seen in the chapter 6.1 and the chemistry between players as seen in the chapter 6.2 with the *Final Model* 6.2.5.

8.1 Betting Analysis

In the previous section, we explored the connection between our validation model and sports betting odds. Now, it's time to analyze whether our predicted probabilities from our models can attempt to beat the bookmakers. First, we need to define the concept of a value bet and then present the strategy to implement.

8.1.1 Value Bet

A **value bet** occurs when the probability of a specific outcome, as assessed by the bettor, is greater than the implied probability reflected in the odds provided by the bookmaker. In other words, if the bettor believes that the likelihood of an event happening is higher than what the odds suggest, placing a bet on that outcome would be considered a **value bet**.

In our case, we will identify a value bet when our model predicts a probability of the home team winning that is greater than the odds offered by the bookmakers. On the plot below 8.1, the green points represent the value bets present in the matches of the second part of the season 2015/2016 of the LaLiga, Serie A and Premier League, according to our *Equal Weights model* 7.2.2.

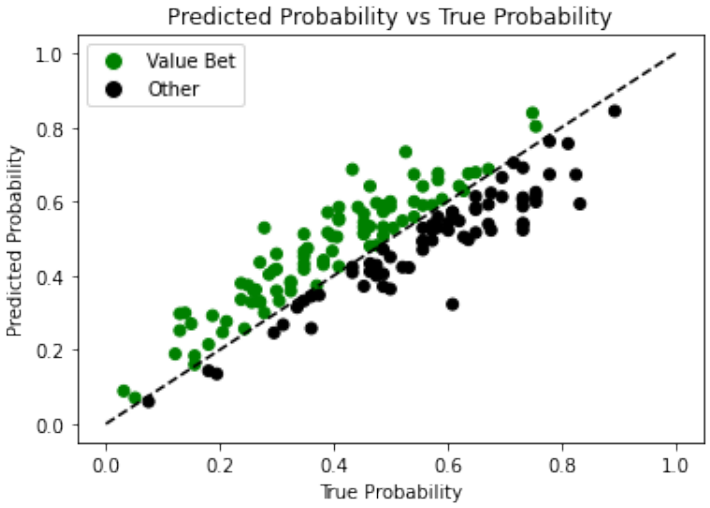


Figure 8.1: Value bets on test set with Logistic Regression

8.1.2 Betting Strategy

It is now time to explain the strategy implemented and analyze the results obtained through a simulation. Initially, we will only bet on the value bets identified earlier 8.1 using *Equal Weights model* 7.2.2. The simulation is based on two parameters : the initial amount of money and the amount used for each bet. In this case, the initial amount is 1000 € and every bet costs 20 € .

On the table below 8.3, you can find the results of the simulation. The simulation is made by considering the odds of the bookmaker without their margin. We can notice that the final amount is 730 € which means that we lost 270 € .

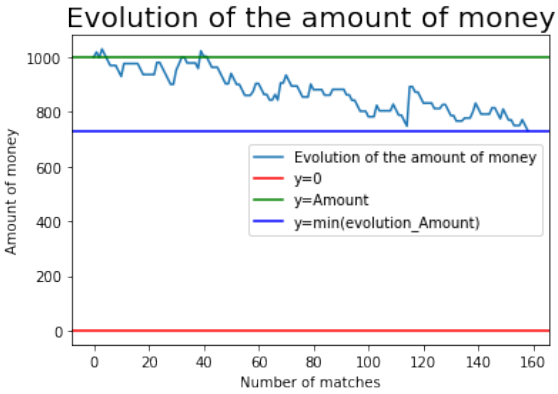


Figure 8.2: Simulation with the Logistic Regression model

Number of games	158
Total Bets	86
Bets win	26
Final Amount	730

Figure 8.3: Simulation Results

We can notice that the number of bets is quite low compared to the number of games. This is due to the fact that we only bet when we have a value bet. The number of bets won is quite low, which explains the loss of money. The final amount is quite low, which means that the strategy is not profitable. In fact, it is quite hard

to beat the bookmaker. The model is quite good at predicting the probability of the home team winning, but it is not enough to beat the bookmaker. Even if we have removed the margin taken by the bookmaker, we are still losing. That means that the value bets identified by the model are not really value bets, the *Equal Weights model* 7.2.2 predicts a larger probability of the home team winning than the bookmaker. However, the bookmaker is right in this case.

8.2 Recruitment

A practical application of the model is also related to recruiting new players. Indeed, at the moment, we create the best possible team based on the players present in the club. However, nothing prevents us from adding players from other clubs to the list and seeing if they will be chosen to be in the optimal team. There are several scenarios we can explore. First, if we want to recruit a specific player, we can check if he would be integrated into the optimal team or not. Another scenario would be if we are undecided between several players and don't know which one to recruit, we would choose the player that allows us to obtain the best score for the optimal team. Finally, the last scenario would be a slightly more practical case. Indeed, during transfer periods, each club allocates a certain budget reserved for transfers. They are not necessarily looking for a specific position, so they can make a certain number of transfers depending on the budget available. This case requires to add a constraint to the model as we cannot spend more money that we have :

$$\sum_{player_j \in \Omega} \sum_{i \in \delta(player_j)} \sigma_i x_i \leq \text{budget}$$

where Ω is the set of new players available and σ_i the price of the i^{th} player. The player's price are from the same dataset as the FIFA score [9]. The price of a player is an estimation based on several factors, such as the player's age, the player's position, the player's statistics, the player's potential, etc

Let see now an example of application of this case. Imagine we are FC Barcelona and we have a budget of 130 million euros. We would like to buy new players from our rival, Real Madrid. By solving the problem presented earlier, the result indicates that we should buy Ronaldo and Modric. On the figure below, you can find the optimal team by recruiting these new players.

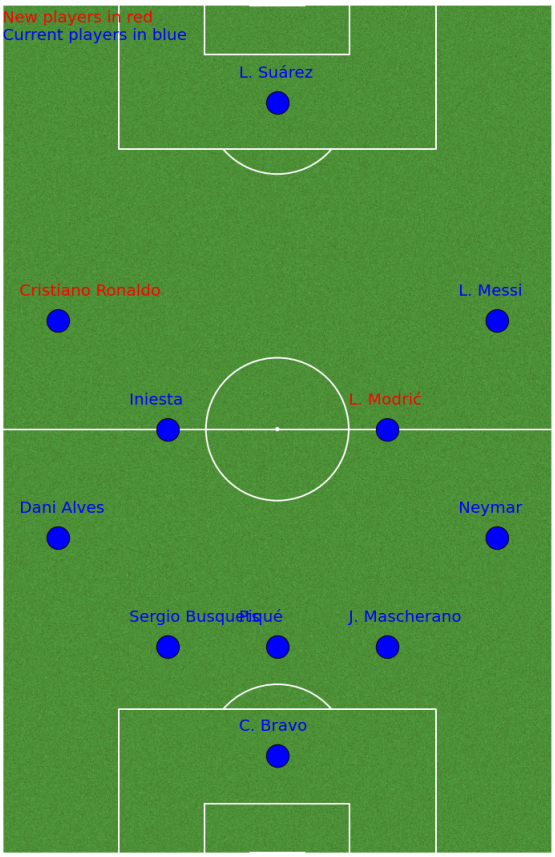


Figure 8.4: Optimal Recruitment

8.3 National Team Example

A concrete example of application is also the team selection in the context of national teams. In this case, the coach has a much larger pool of players. Indeed in a club, a coach can usually rely on a maximum of 25 players. However, in a national team, a coach can choose from all players with the desired nationality. Therefore, there is a huge number of players eligible to play for countries like France, Spain, Brazil, etc.

Initially, we will analyze the Belgian national team. We will compare the team that was fielded during Euro 2016 with the team that would achieve the best score according to our model. We assumed that the coach could choose from among the 50 best Belgian players. First, we need to remove players who are unavailable due to injury. In our case, we need to remove the inclusion of Vincent Kompany. On the left, you can see the team fielded for the Belgium-Sweden match during the group stage. On the right, you can see the predicted team for the same 4-2-3-1 formation.

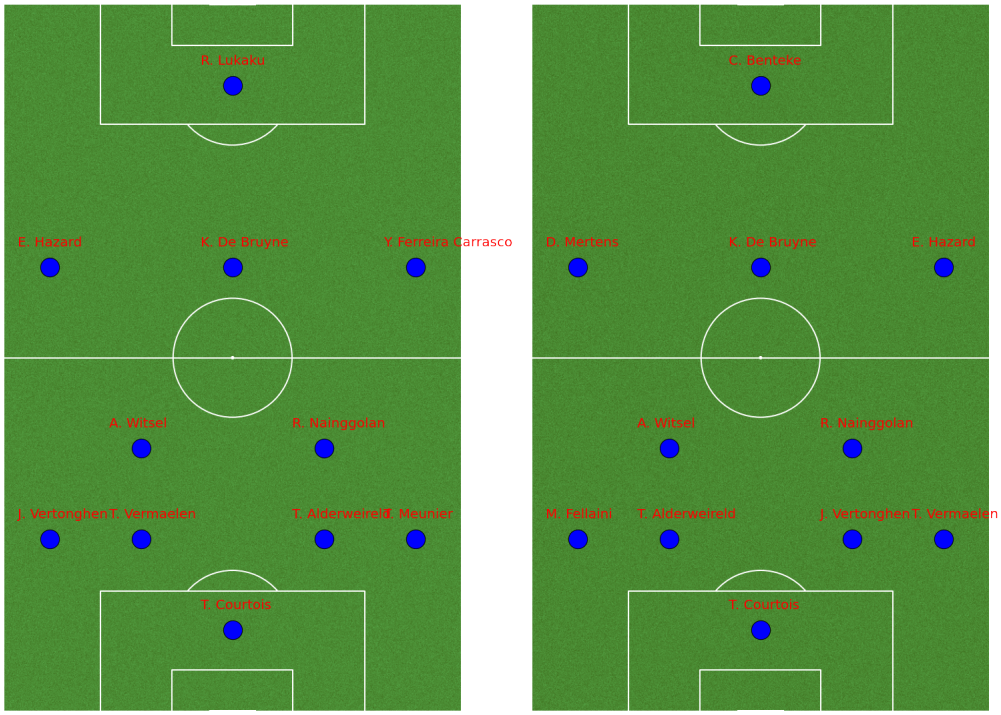


Figure 8.5: Comparison between the fielded team and the predicted team

Overall, the two teams are quite similar. The main differences in the predicted team are:

- Thomas Vermaelen on RB
- Marouane Fellaini on LB
- Dries Mertens on LW
- Benteke on ST

The other differences are just swaps between two closed positions or switching left and right. The main differences showed above are interesting, mainly for the LB position. The other changes are consistent. Indeed, since Thomas Vermaelen is a defender, it's likely to see him as a RB. Dries Mertens plays a usual role for him and Benteke has more or less the same level as Lukaku, so everything is normal until now. However, what is quite surprising at first sight is seeing Fellaini in the left-back position, given that he's typically a midfielder. This can be explained by the fact that Fellaini is a very complete player. He has strong statistics across various skill sets. That's why his score at every position is quite good.

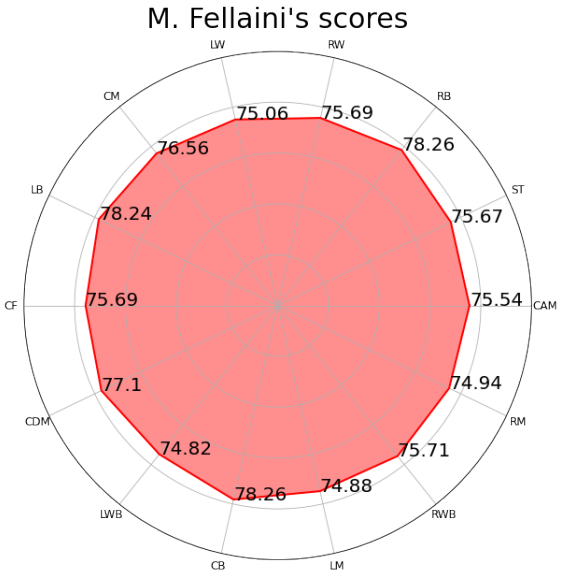


Figure 8.6: Fellaini's scores by position

As you can see above, regardless of the position on the field, the score is almost identical. This type of player is very useful for a coach because he can play in various positions. In case of injury or tactical change, this player can be repositioned in different areas on the field while still maintaining a high score. However, it's worth noting that in practice, it's highly unlikely to see such a decision. The right-back and left-back positions require habits and reflexes that a midfielder typically doesn't have, even if they possess the technical and physical abilities.

In the second part, we will consider the French national team. We will compare the team that was fielded during the opening match of Euro 2016 against Romania 8.7. As done previously, on the left, you'll find the French team's lineup during their 2-1 victory, and on the right, you'll find the lineup predicted by the model.

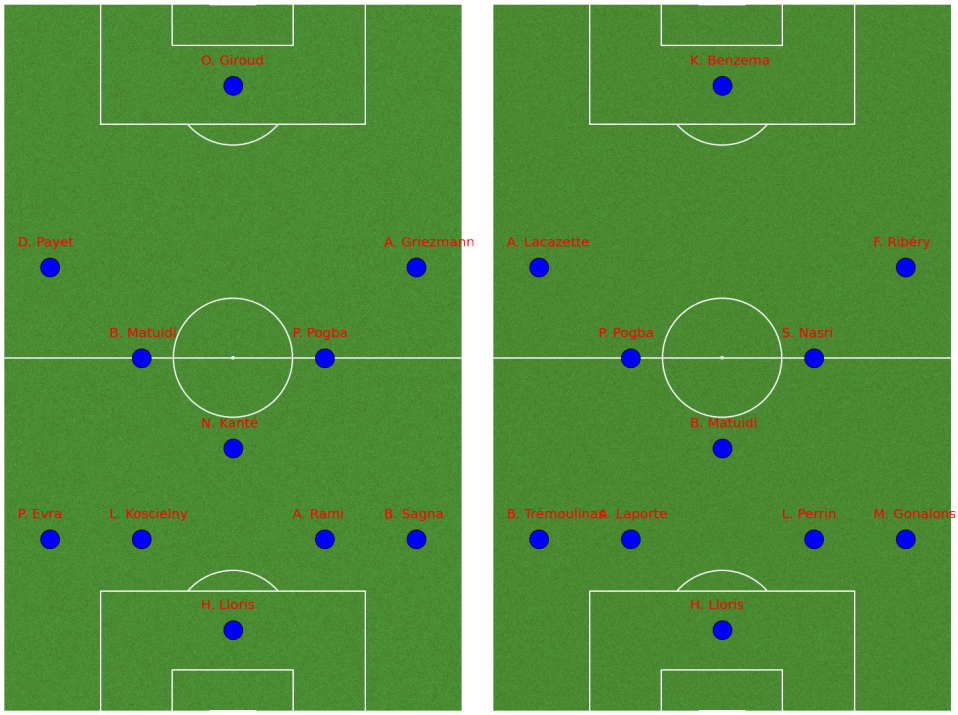


Figure 8.7: Comparison between the fielded team and the predicted team

Unlike Belgium, this time there are many differences between the two teams. The main differences to note are:

- Benzema on ST
- Kanté on CDM
- Ribéry on RW

We can also see that the entire defense is different. Now let's try to explain the reason for these various changes. First, the inclusion of Benzema in our lineup is explained by the fact that he's one of the best forwards in the world, playing for Real Madrid. However, he's not included in the French national team due to personal issues. The model doesn't account for such issues, and while his presence would certainly be beneficial from a qualitative perspective, it's essential to ensure that his presence doesn't impact team relations. Ribéry's absence from the national team is because he chose to retire from international play, he no longer wants to represent France. Many players make this decision as they get older, preferring to leave room for younger talent and to focus on the end of their club careers. These first two absences are due to external factors, even though these players are still reachable, and a coach could negotiate with them for their return to the national team. For Kanté, the reason for his inclusion in the French team is different. The Euro is a tournament held at the end of the season, while the player statistics used in our model were calculated at the beginning of the season. A player like Kanté, who was relatively unknown at the start of the season but then became a Premier League champion with Leicester, was underestimated and made significant progress throughout the year. The early-season

statistics no longer reflect his current level. That's why the model does not include Kanté in the team.

Finally, we can see that the predicted defense is completely different from the one that was actually used. To understand why, let's look at how player scores are spread out 8.8. For France, many players have very similar scores, while for Belgium, there's a wider range. This explains why there are fewer surprises in the Belgian lineup. Because of the large gaps in skill levels, there are fewer choices for putting together a strong team. France, on the other hand, can create many different lineups with players of similar skill levels. This is why there's a bigger difference between the French team that was fielded and the one predicted by the model.

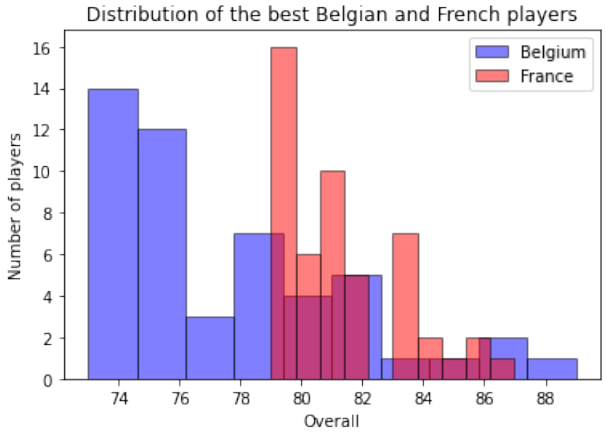


Figure 8.8: Comparison of the player's score distribution

In conclusion, we can see that there are common points between the fielded and predicted teams, but also differences, for which we've tried to explain possible causes. The model does not take into account for more human and extra-sporting factors, which the coach considers. This has negative aspects too, as the coach may lack objectivity in certain situations. A coach usually has favorite tactics, formations, and player combinations. These preferences affect how they choose the team and make decisions during games. Even if a model suggests a different lineup based on stats, a coach might stick to what they know works best for their team. This familiarity with certain players and tactics helps create a sense of consistency and teamwork.

Chapter 9

Visualisation tool

In this chapter, we will explore the visual interface implemented in Python using the Dash package. This interface will allow you to use and visualize a dynamic tool for football team design management based on the model presented in previous sections. First, we will introduce the design of the interface and its various features. Then, we will examine and detail a practical and interesting use case. This tool offers a more useful and easier-to-use way to build efficient team.

The first view 9.1 offers us the ability to first select the league and the team that we want to visualise. In this case, the chosen league is La Liga and the team is Barcelona. Next, there is the option to know all the players in Barcelona's squad. For example, if a player is injured or suspended, or if you want to generally exclude a player from the group, you can deselect them. Finally, you can either check the "best lineup" box, which will give the best team composition regardless of formation, or you can leave this box unchecked, in which case the resulting composition will be the best lineup for a fixed formation that you choose just below.

Figure 9.1: First view

The second view 9.2 of the interface offers us the result of the choices made before on the first view. The blue circles represent the positions associated with the tactical formation selected earlier, and in red, you'll find the name of the chosen player for each position.

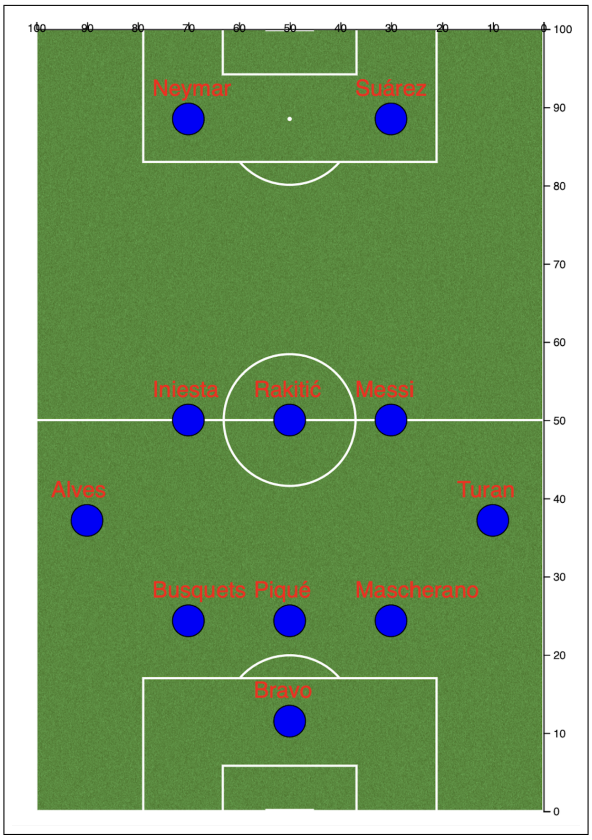


Figure 9.2: Second view

The first part of the interface was therefore useful for creating the lineup of a team at the beginning of a match. In the second part, the interface provides a tool for managing the team during the game. Indeed, following an injury or poor performance from one or more players, a coach can make substitutions and bring in players who have been on the bench until now. On the view below 9.3, you will find the list of players on the field. This list contains the players obtained in the first part of the interface. You can deselect the players you no longer want on the field. Below, you can find the players who are on the bench. Then, if you have deselected players above, you will find them in the list called "Players who are out". As example, we have deselected Busquets and Neymar. Finally, as in the first part, you can either decide on a specific formation or choose the best lineup regardless of formation by checking the "best lineup" box. In this case, by checking the box, the best formation is 433. You can find just below the 2 players who are joining the team. We can notice that even though Neymar is an offensive player, he will be replaced by a more defensive-minded player, in deed both Mathieu and Alba are defenders. This is also due to the change in the team's formation by moving from 352 to 433.

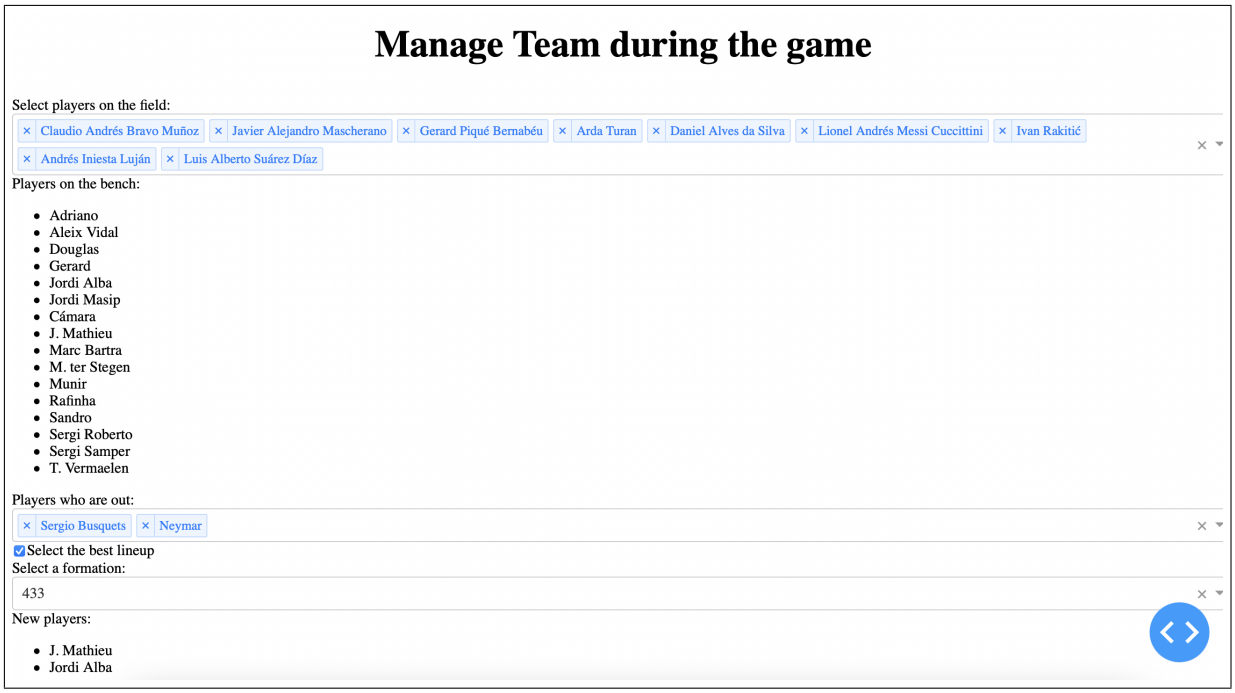


Figure 9.3: Third view

Finally, the last part of the interface is the visualisation of the new team after the substitutions as explained before. To obtain the lineup that maximizes the team's score while keeping the players who haven't been substituted on the field, we need to add new constraints to the *Complete Model* 4.3.2. The constraints are as follows :

- $\sum_{j \in \delta(player_i)} x_j = 1$ for each $player_i$ on the field
- $\sum_{j \in \delta(player_i)} x_j = 0$ for each $player_i$ off the field

As reminder, $\delta(player_i)$ 4.3 is the set of twins of the $player_i$. The first constraint is useful for keeping the players who are already on the field from being substituted but the players can change of position on the field. Without this constraint, additional changes would be required, and a coach is only allowed a limited number of substitutions. The second constraint ensures that players who have been taken off the field cannot be put back on, preventing a player from replacing themselves.

Finally, the last view of the interface 9.4 shows the new team after the substitutions. We can see that Neymar and Busquets have been replaced by Mathieu and Alba.

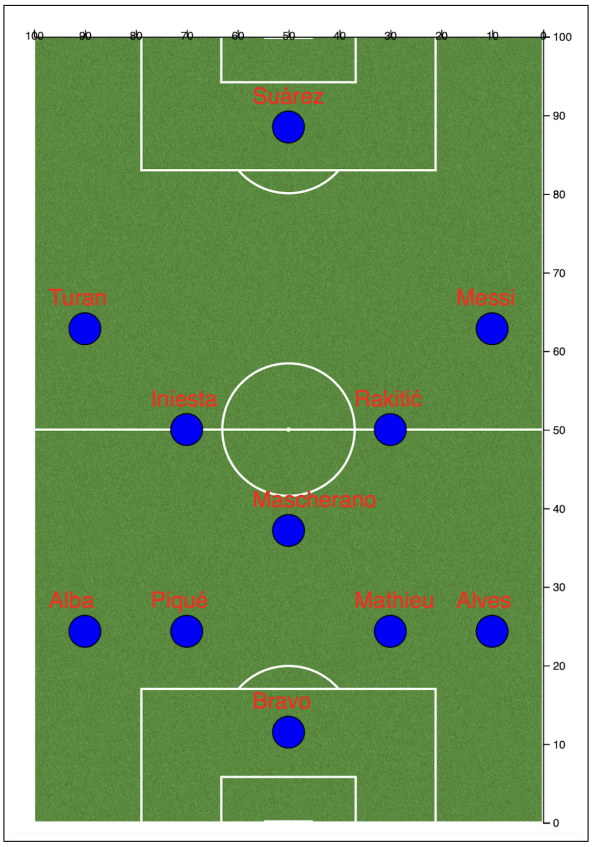


Figure 9.4: Fourth view

The interface we have explored provides a powerful and flexible tool for managing football teams. It allows you to create and adjust team lineups dynamically, taking into account injuries, tactical changes, and real-time needs during matches. With its user-friendly approach, it facilitates strategic decision-making while optimizing team score. To take our interface even further, we could offer a tool for managing a team after a red card. This would require adjustments to the model as well as to the tactical formation.

Chapter 10

Conclusion and Perspectives

In this thesis, we addressed the complex task of optimizing football team composition using a data-driven approach. Our goal was to improve team performance by using quantitative methods. We created a model based on graph theory to represent players and their interactions, focusing on maximizing team synergy. Our objective was to develop a framework that helps make informed decisions for team selection by considering both individual player skills and their interactions.

We achieved several significant contributions through this work. We developed a graph-based model to represent football team compositions, where nodes represent players and edges represent interactions. The model takes into account the individual qualities of the players but also the chemistry between them. The introduction of edge scoring to quantify the quality of player interactions enhanced the accuracy of our team performance predictions. We validated the model using historical performance data from various football leagues, demonstrating its effectiveness in predicting team success. Moreover, we developed a method in order to quantify the relative importance of the players and the chemistry between them. Additionally, there are a lot of applications of the model, such as betting analysis, player recruitment, and national team selection. We also developed a visual interface to facilitate team management and decision-making for substitutes during the match.

The main observed limitation of the model is the small impact of the chemistry on the total score of the team. Despite our efforts to quantify and incorporate player chemistry, the contribution of these interactions to the overall team performance score was less significant than expected. This limitation suggests that while individual skills are critical, our current method of measuring chemistry might not fully capture the complexities of real-world dynamics. Another limitation is the lack of external factors in the *Equal Weights* Model 7.2.2 as we only consider the composition of the team. The validation part of the model tries to predict the outcome of the match based on the team composition. However, the outcome of a match is influenced by many other factors such as the coach's strategy, the motivation of the players, the weather conditions, the referee's decisions, etc. These factors are not taken into account in our model and are hard to quantify, which limits its predictive power.

To address these limitations and enhance the model, future research could explore several promising directions. One approach is to investigate the use of directed edges in the graph model, where interactions in one direction could represent specific targets like passing, and in the opposite direction could represent other targets like defending. This method would provide a more detailed understanding of player interactions and their contributions to different aspects of the game.

Additionally, implementing a dynamic scoring system that varies the team's score according to tactical needs, such as shifting focus between attacking and defending, could prove beneficial. This approach is particularly useful in scenarios where match outcomes or tournament progression depend on specific results, such as in knockout stages or home-and-away legs.

Exploring alternative player rating systems beyond the traditional FIFA scores could also enhance the model. By incorporating more comprehensive and real-time performance metrics, we could provide a better assessment of player abilities and potential contributions to team success.

An other extension would be to take into account the fact that the players are not playing at their best level all the time. Indeed, a player can have a bad day and not play at his usual level. This could be taken into account by adding a random variable to the player's score. This random variable would be a normal distribution centered on the player's score and with a certain standard deviation. This would allow us to simulate the fact that a player can play better or worse than his usual level. You can find more details in the annexes 11 .

Finally, it is crucial to integrate data on the opposing team's strengths, weaknesses, and playing style into the team selection process. This consideration would create a more strategic and adaptable team composition, better suited to the specific challenges posed by different opponents.

In conclusion, our work represents a significant step toward optimizing football team composition using data-driven methods. By combining graph theory, player scoring, and chemistry metrics, we developed a robust model that allows us to build an efficient team. While there are limitations to the current model, future research directions offer promising opportunities to refine and expand our approach.

Bibliography

[1] Lotte Bransen, Jan Van Haaren (2020) : https://www.janvanhaaren.be/assets/papers/mitssac-2020-chemistry.pdf

[2] Socceraction python package : https://github.com/ML-KULeuven/socceraction/

[3] StatsBomb datasources and python package : https://statsbomb.com

[4] mplsoccer python package :https://mplsoccer.readthedocs.io/en/latest/

[5] https://xgboost.readthedocs.io/en/stable/

[6] matplotsoccer python package : https://pypi.org/project/matplotsoccer/

[7] Decision Tree Classification Algorithm : https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

[8] XGBoost : https://www.nvidia.com/en-us/glossary/xgboost/

[9] Kaggle datasets containing FIFA scores : https://www.kaggle.com/datasets/joebeachcapital/fifa-players

[10] European Soccer Database : https://www.kaggle.com/hugomathien/soccer

[11] Soudeep Deb, Shubhabrata Das (2023) , Optimal selection of the starting lineup for a football team : https://arxiv.org/pdf/2303.12385

[12] Mahdi Nouraie (2023), Intelligent team formation and player selection: a data-driven approach for football coaches : https://www.researchgate.net/publication/375668203_Intelligent_team_formation_and_player_selection_a_data-driven_approach_for_football_coaches

[13] Quantifying NBA Player Chemistry, Danny Leese (2020) : https://medium.com/the-sports-scientist/quantifying-nba-player-chemistry-d6c4fa8f016e

[14] Assignment Problem figure : https://www.researchgate.net/figure/Bipartite-graph-formulation-of-a-linear-sum-assignment-problem-LSAP_fig1_335990041

[15] Logistic Regression : https://www.machinelearningplus.com/machine-learning/logistic-regression-tutorial-examples-r/

[16] Multi-Layer Perceptron : https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning

[17] Activation Function : https://ai-artificial-intelligence.webyes.com.br/most-used-activation-functions-in-neural-networks/

[18] Random Forests : https://www.almabetter.com/bytes/tutorials/data-science/random-forest

[19] Gradient Boosting : https://www.geeksforgeeks.org/ml-gradient-boosting/

[20] Regression Metrics : https://machinelearningmastery.com/regression-metrics-for-machine-learning/

[21] ROC AUC : https://towardsdatascience.com/a-quick-guide-to-auc-roc-in-machine-learning-models-f0aedb78fbad

[22] Residual Plot from this page : https://medium.com/@DilaneKombou/presentation-of-mse-mean-squared-error-a4ee9b6cff49

[23] Odds System : https://www.compare-bet.fr/guides/les-cotes-aux-paris-sportifs.html

[24] History of data analysis, Filip Sekan , 2023 : https://medium.com/@filip.sekan/short-history-of-data-analysis-in-football-ce1963e428ae

[25] Understanding The Linear Regression!!!! : https://medium.com/analytics-vidhya/understanding-the-linear-regression-808c1f6941c0

[26] Understanding Logistic Regression : https://medium.com/@novus_afk/understanding-logistic-regression-a-beginners-guide-73f148866910

[27] Random Forest Algorithm in Machine Learning : https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/

[28] Regression Metrics for Machine Learning : https://machinelearningmastery.com/regression-metrics-for-machine-learning/

[29] VAEP,DTAI SPORTS ANALYTICS LAB : https://dtai.cs.kuleuven.be/sports/vaep

Chapter 11

Annexes

The table below 11.1 contains the size of the training and test sets from the section Players' scores prediction 6.1 .

Position	**Train Set Size**	**Test Set Size**
RB	1534	384
CDM	1817	455
RWB	72	18
RM	1653	414
LM	1684	422
ST	2286	572
LB	1466	367
CF	332	84
RW	613	154
LW	575	144
CAM	1570	393
LWB	79	20
CB	2520	631
CM	2536	634

Table 11.1: Size of Train and Test Sets for Different Positions

The table below 11.2 contains the performance metrics for the players' scores prediction using linear regression. 6.1

Position	**MAE Train**	**RMSE Train**	**MAE Test**	**RMSE Test**
RB	1.567	2.152	1.597	2.221
CDM	1.631	2.245	1.701	2.369
RWB	1.514	1.929	1.827	2.340
RM	1.367	1.832	1.321	1.782
LM	1.335	1.835	1.305	1.808
ST	1.015	1.409	0.959	1.279
LB	1.462	2.032	1.566	2.219
CF	1.130	1.438	1.154	1.441
RW	1.096	1.495	1.233	1.638
LW	1.035	1.403	1.048	1.365
CAM	1.007	1.439	0.904	1.232
LWB	1.295	1.807	1.775	2.173
CB	0.683	1.058	0.671	0.946
CM	1.734	2.286	1.748	2.365

Table 11.2: Performance Metrics by Position with Linear Regression

The table below 11.3 contains the performance metrics for the players' scores prediction using MLP. 6.1

Position	MAE Train	RMSE Train	MAE Test	RMSE Test
RB	1.462	1.909	1.563	2.050
CDM	1.111	1.468	1.109	1.500
RWB	8.766	9.669	9.682	10.944
RM	1.286	1.738	1.234	1.718
LM	1.293	1.741	1.291	1.788
ST	1.076	1.479	1.034	1.363
LB	1.323	1.720	1.365	1.846
CF	1.109	1.432	1.089	1.376
RW	1.140	1.574	1.325	1.775
LW	1.228	1.634	1.235	1.625
CAM	1.060	1.489	0.988	1.329
LWB	6.160	6.959	6.430	6.923
CB	0.713	1.087	0.699	0.989
CM	1.625	2.178	1.698	2.326

Table 11.3: Performance Metrics by Position with MLP

The figure below 11.1 shows the SPADL representation of an action. This is used in the VAEP section 6.2.1.

Attribute	Description
game_id	the ID of the game in which the action was performed
period_id	the ID of the game period in which the action was performed
seconds	the action's start time
player	the player who performed the action
team	the player's team
start_x	the x location where the action started
start_y	the y location where the action started
end_x	the x location where the action ended
end_y	the y location where the action ended
action_type	the type of the action (e.g., pass, shot, dribble)
result	the result of the action (e.g., success or fail)
bodypart	the player's body part used for the action

Figure 11.1: SPADL

Stochastic Optimization

Now we will analyze an extension of our initial problem. We have assumed that the players' scores and relationships were deterministic. That means that, for each match, we considered that the player would evolve at the same level. However, in reality, this is quite different. Indeed, some players perform more or less at the same level in each match, while others are able of sometimes playing very well but also very poorly.

These changes in performance levels can be analyzed and may have various sources.That's why an extension of our problem would be to consider scenario probabilities, and based on these scenarios, each player would have a different score. The resolution of this problem is therefore different. Indeed, depending on the scenario that occurs, a team can be more or less successful. Various approaches exist to solve this type of problem. For example, one can consider the expected score of the players, and at that point, we come back to the initial problem. However, there are many different approaches to solving this type of problem, and that is not the focus of this work. Still, it would be interesting to explore the further implications of this idea.

Updated Model

Let redefine the scores b_i and $e_{i,j}$. We could consider, for example, that these scores follow normal distributions, and it's the standard deviation of the distribution that will determine to what extent a player plays at a relatively constant level. The choice of these distributions is just an example. Let's use these following random variables :

- $c_i \sim \mathcal{N}(b_i, \sigma_i^2)$

- $f_{i,j} \sim \mathcal{N}(e_{i,j}, \sigma^2_{i,j})$

The model remains very similar to the initial one. The constraints remain identical but the objective function is updated. The scores in the objective function aren't deterministic but are random variables . In red, you will find the modifications made.

$$\text{Maximize } w_1 \sum_{i=1}^{M} c_i \cdot x_i + w_2 \sum_{i=1}^{M} \sum_{j=i+1}^{M} l_{i,j} \cdot f_{i,j}$$

In the case presented above, all variables are considered independent, whereas in reality, this is not the case. A better approach could be to segment the different possible scenarios and assign probabilities to them. It's rare to observe that only one player performs worse than usual, most of the time there are several. One could create, for example, three scenarios: a pessimistic scenario where each player has a slightly or significantly lower level. The second scenario would represent an average case, where the players perform at their usual level. The last scenario is the optimistic one, where some players perform significantly better than usual while others perform only slightly better.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**