Université Mohammed V
Faculté des Sciences
Rabat

**Master Science et Ingénierie de Données**

Data Mining Project

**PROJECT REPORT**

---

# Clustering Moroccan Football Players with similar skill set using K-Means

---


**Created by:**

EL BELGHITI Hamza, EL FARKH Salaheddine, HAZIM Khaoula

**Supervised by:**

Dr. RIADSOLH and Prof. LASRI


**Academic year: 2022-2023**

# Table of contents

# ABSTRACT

In this project, the primary objective is to address the challenge of categorizing Moroccan football players based on their skillsets. The project begins by defining the problem statement, which involves clustering players into five distinct categories: goalkeepers, defenders, offensive midfielders, defensive midfielders, and strikers. To accomplish this task, the K-Means clustering algorithm is employed, and the report elaborates on the inner workings of this algorithm. It also provides insights into the FIFA 22 dataset, emphasizing data exploration, analysis, and visualization to gain a comprehensive understanding of the player attributes. The subsequent sections of the project encompass the development of the K-Means clustering algorithm through pseudocode and code implementation, followed by visual representation of the clusters generated. Additionally, the project assesses the performance of the developed algorithm by comparing it to the scikit-learn implementation, thereby offering a comprehensive approach to player clustering based on skillsets.

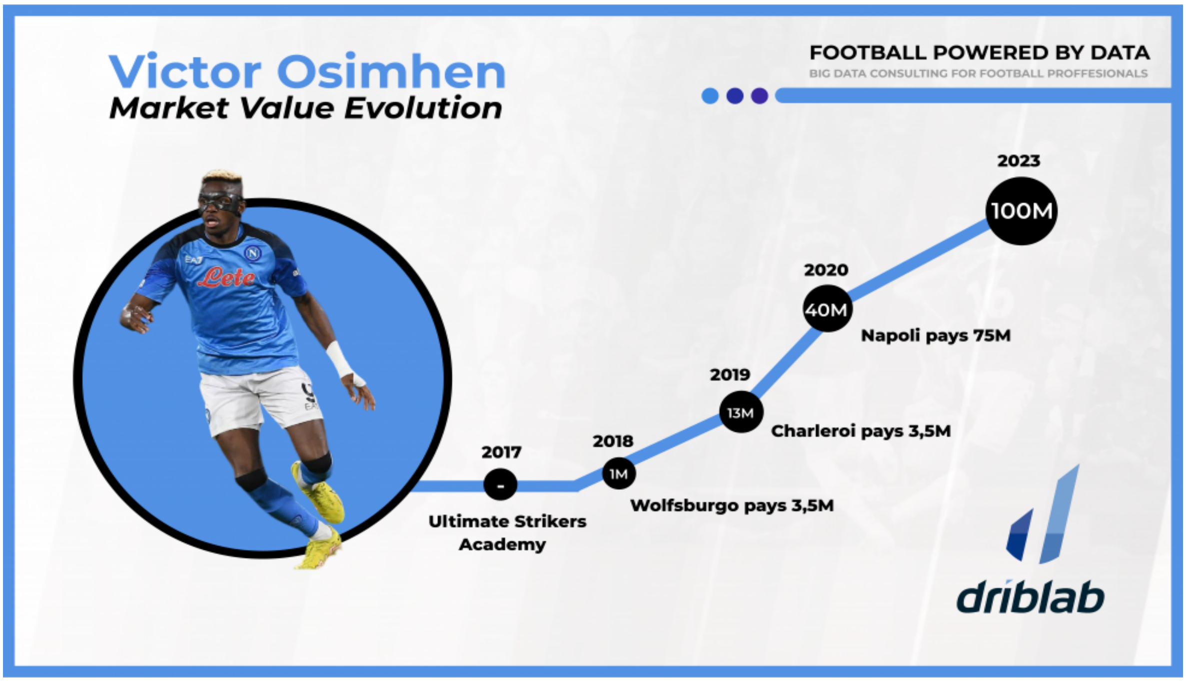
**Keywords:** K-Means Algorithm, Clustering, Data Mining, Football Players

# I. Introduction

The world of football is characterized by its diverse array of players, each possessing a unique set of skills and attributes that contribute to their team's success. In this era of data-driven decision-making, understanding and categorizing football players based on their skillsets is an essential endeavor for clubs, coaches, and analysts. This project delves into the exciting domain of sports analytics, specifically focusing on the task of clustering football players into distinct categories using the K-Means clustering algorithm.

The primary objective of this project is to tackle the challenge of player categorization, a task that holds immense significance in scouting, team formation, and strategic planning. By employing the K-Means clustering technique, we aim to automatically group players with similar skillsets into predefined categories, including goalkeepers, defenders, offensive midfielders, defensive midfielders, and strikers.

In this report, we will delve into the fundamental aspects of K-Means clustering, elucidating its underlying principles and mechanisms. Furthermore, we will introduce the FIFA 22 dataset, which serves as the foundation for our analysis. Data exploration and visualization will play a pivotal role in comprehending the dataset's nuances, enabling us to make informed decisions during the clustering process.



The core of this project will involve implementing the K-Means clustering algorithm, complete with pseudocode for clarity, and visualizing the clusters generated from the player data. To ensure the robustness of our approach, we will also conduct a performance comparison with the scikit-learn implementation of the K-Means algorithm.

# II. Project background and problem statement

### 2.1. Background:

In the realm of professional football, where every player's unique skills contribute significantly to a team's success, the ability to categorize players accurately based on their skillsets is important. Historically, player evaluation and categorization have largely relied on subjective assessments by coaches, scouts, and analysts. However, the advent of comprehensive player statistics and data-driven approaches has opened up new possibilities for objective and data-centric player classification.

### 2.2. Problem:

The problem this project seeks to address revolves around the need for a systematic and data-driven method to cluster football players into specific skill-based categories. Such categorization is vital for various facets of the football industry, including talent scouting, team composition, tactical planning, and performance analysis. Identifying players with similar skillsets allows clubs to make informed decisions about recruitment, optimize team strategies, and identify areas for improvement.

The traditional manual assessment of players, though valuable, is time-consuming, subjective, and prone to biases. As the sport evolves and the demand for data-driven insights grows, there is an increasing need for automated approaches that can objectively and comprehensively assess a player's abilities based on a wide range of attributes.

### 2.3. Solution:

To tackle this challenge, we turn to the field of data mining and clustering. By leveraging the K-Means clustering algorithm, we aim to develop a systematic method for grouping football players into categories that reflect their primary skillsets. This project seeks to bridge the gap between traditional player evaluation methods and modern data analytics, providing football clubs, coaches, and analysts with a powerful tool to enhance their decision-making processes. We will specifically focus on the best Moroccan players for this project.

# III. K-Means Clustering

### 3.1. What is Clustering?

Cluster analysis is a technique used in data mining and machine learning to group similar objects into clusters. K-means clustering is a widely used method for cluster analysis where the aim is to partition a set of objects into K clusters in such a way that the sum of the squared distances between the objects and their assigned cluster mean is minimized.

### 3.2. K-Means Clustering

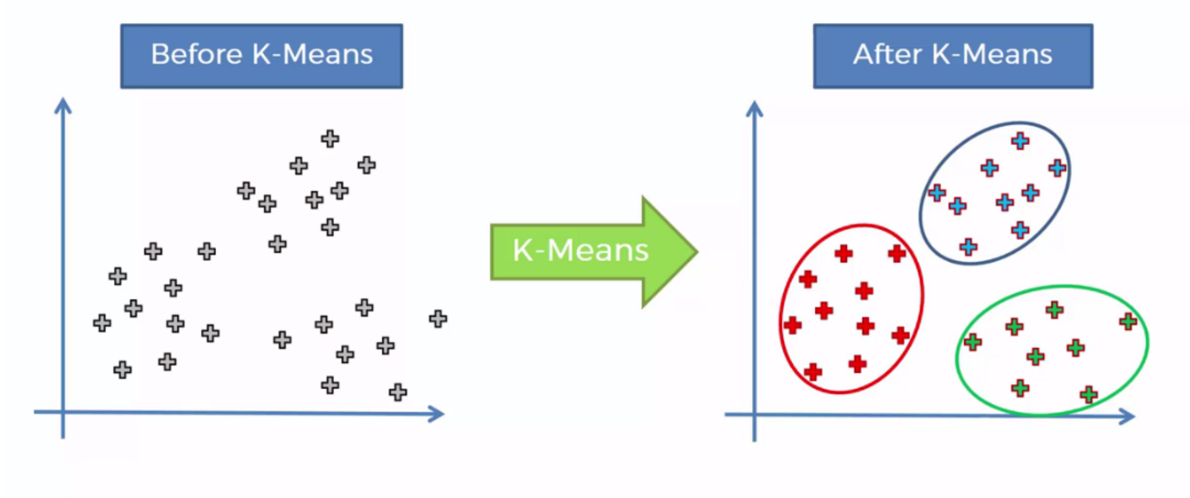
K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labeled, outcomes.

The fundamental idea behind K-Means is to group data points together based on their similarity, with the goal of creating clusters where data points within the same cluster are more similar to each other than they are to those in other clusters. This technique is particularly valuable when dealing with large datasets or when you want to uncover hidden patterns and structures within your data.

A cluster is like a group of similar things. Imagine you want to organize a bunch of objects into different groups. You first decide how many groups you want (let's call this number 'k'). Then, you find a central point for each group (we'll call this point a "center").

Now, you start putting each object into the group with the closest center. You do this so that the groups are as tight and small as possible.

The word "means" in K-Means refers to finding the average or middle point of each group. This helps summarize what each group is like.




3.1 Figure - Clustering data points into 3 clusters using the K-Means Algorithm

### 3.2.1. Example

Let's try understanding this with a simple example. A bank wants to give credit card offers to its customers. Currently, they look at the details of each customer and, based on this information, decide which offer should be given to which customer.

Now, the bank can potentially have millions of customers. Does it make sense to look at the details of each customer separately and then make a decision? Certainly not! It is a manual process and will take a huge amount of time.

So what can the bank do? One option is to segment its customers into different groups. For instance, the bank can group the customers based on their income:

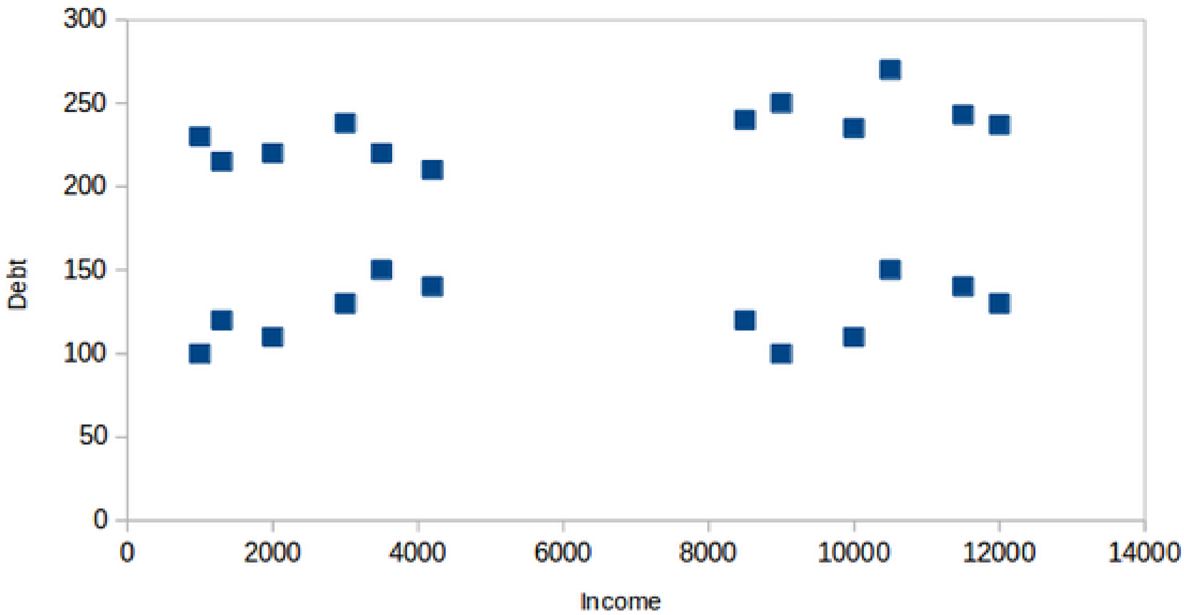

3.2.1 Figure - Different customer income groups

The bank can now make three different strategies or offers, one for each group. Here, instead of creating different strategies for individual customers, they only have to make 3 strategies. This will reduce the effort as well as the time.

The groups shown above are known as clusters, and the process of creating these groups is known as clustering.

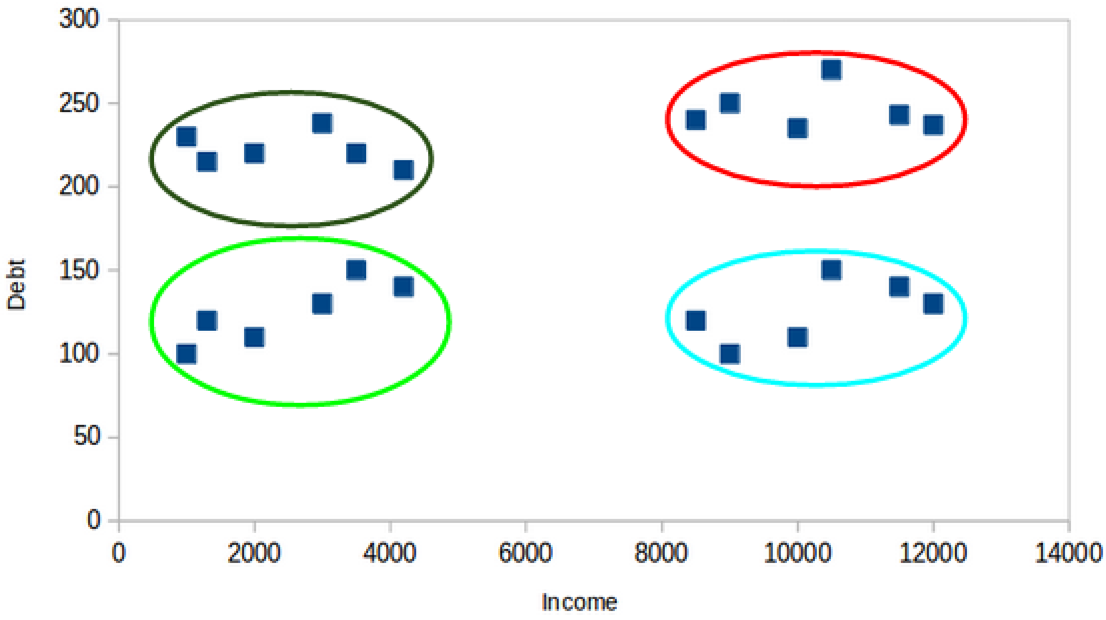
### 3.3. Properties of Clusters

We'll take the same bank as before, which wants to segment its customers. For simplicity purposes, let's say the bank only wants to use the income and debt to make the segmentation. They collected the customer data and used a scatter plot to visualize it:




3.3.1. Figure - Scatter plot of customer data (Debt vs Income)

On the X-axis, we have the income of the customer, and the y-axis represents the amount of debt. Here, we can clearly visualize that these customers can be segmented into 4 different clusters, as shown below:

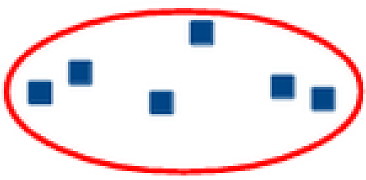



3.3.2. Figure - Customers segmented to 4 different clusters

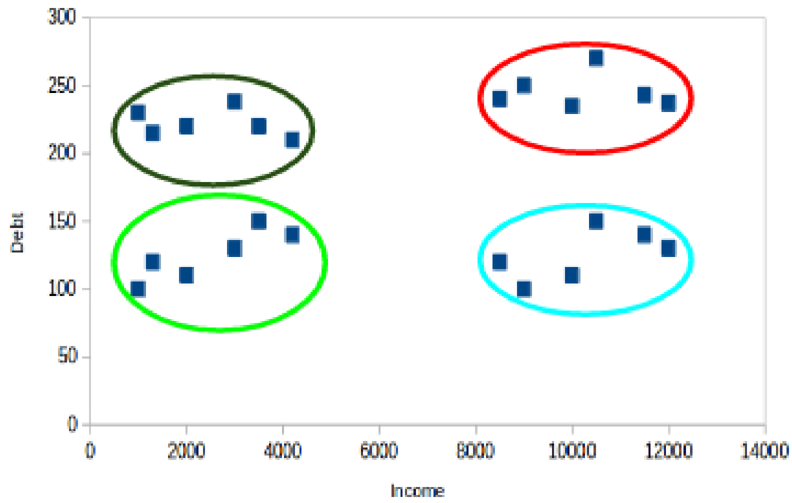
This is how clustering helps to create segments (clusters) from the data. The bank can further use these clusters to make strategies and offer discounts to its customers. So let’s look at the properties of these clusters.

Here's some of the properties of the K-Means CLustering Algorithm:

- **All the data points in a cluster should be similar to each other.** Having similar data points within the same cluster helps the bank to use targeted marketing. You can think of similar examples from your everyday life and consider how clustering will (or already does) impact the business strategy.



- **The data points from different clusters should be as different as possible.** The k-means algorithm uses an iterative approach to find the optimal cluster assignments by minimizing the sum of squared distances between data points and their assigned cluster centroid.

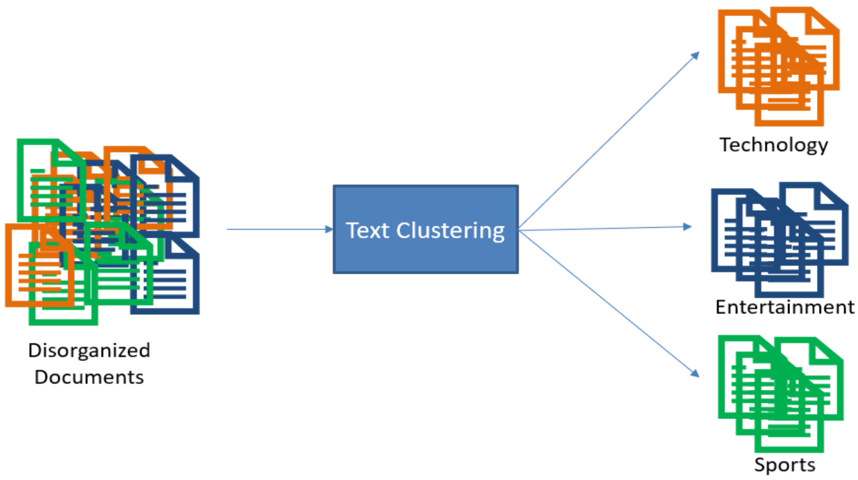



### 3.4. Applications of Clustering in Real-World Scenarios

Clustering is a widely used technique in the industry. It is being used in almost every domain, from banking and recommendation engines to document clustering and image segmentation.

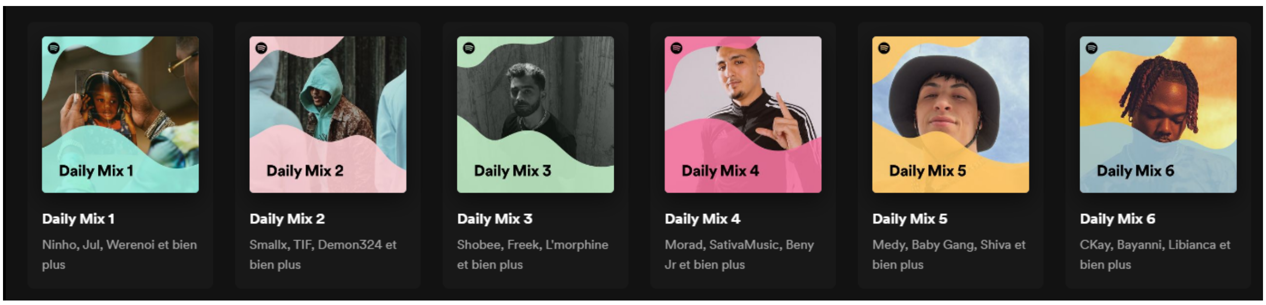
➔ **Customer Segmentation:** We covered this earlier – one of the most common applications of clustering is customer segmentation. And it isn't just limited to banking. This strategy is across functions, including telecom, e-commerce, sports, advertising, sales, etc.


➔ **Document Clustering:** This is another common application of clustering. Let's say you have multiple documents and you need to cluster similar documents together. Clustering helps us group these documents such that similar documents are in the same clusters.




➔ **Image Segmentation:** We can also use clustering to perform image segmentation. Here, we try to club similar pixels in the image together. We can apply clustering to create clusters having similar pixels in the same group.

➔ **Recommendation Engines:** Clustering can also be used in recommendation engines. Let's say you want to recommend songs to your friends. You can look at the songs liked by that person and then use clustering to find similar songs and finally recommend the most similar songs.

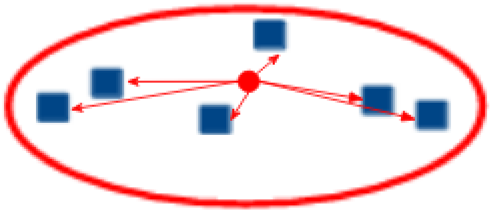


### 3.5. Different Evaluation Metrics for Clustering

- **Inertia:**

Inertia tells us how far the points within a cluster are. So, inertia actually calculates the sum of distances of all the points within a cluster from the centroid of that cluster. Normally, we use Euclidean distance as the distance metric, as long as most of the features are numeric; otherwise, Manhattan distance in case most of the features are categorical.

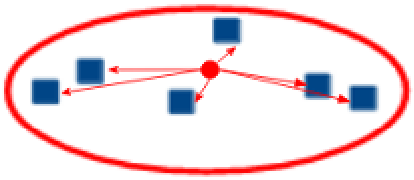
We calculate this for all the clusters; the final inertial value is the sum of all these distances. This distance within the clusters is known as intracluster distance. So, inertia gives us the sum of intracluster distances:



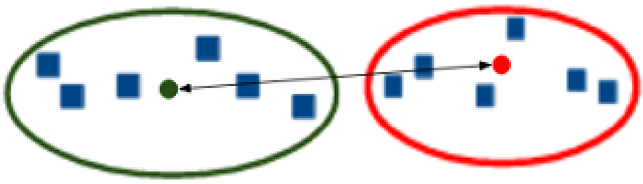
Intra cluster distance

- **Dunn Index**

We now know that inertia tries to minimize the intracluster distance. It is trying to make more compact clusters. If the distance between the centroid of a cluster and the points in that cluster is small, it means that the points are closer to each other. So, inertia makes sure that the first property of clusters is satisfied. But it does not care about the second property – that different clusters should be as different from each other as possible.



Intra cluster distance



Inter cluster distance

This is where the **Dunn index** comes into action. Along with the distance between the centroid and points, the Dunn index also takes into account the distance between two clusters. This distance between the centroids of two different clusters is known as **inter-cluster distance**. Let's look at the formula of the Dunn index:

$$\text{Dunn Index} = \frac{\min(\text{Inter cluster distance})}{\max(\text{Intra cluster distance})}$$

We want to maximize the Dunn index. The more the value of the Dunn index, the better the clusters will be. Let's understand the intuition behind the Dunn index:

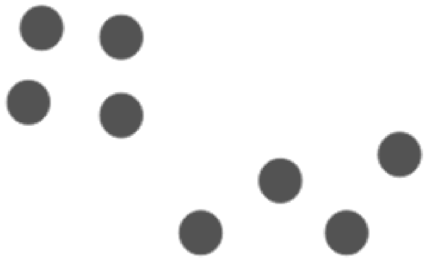
$$\text{Dunn Index} = \frac{\overbrace{\min(\text{Inter cluster distance})}^{\text{Clusters are far apart}}}{\max(\text{Intra cluster distance})}$$

In order to maximize the value of the Dunn index, the numerator should be maximum. Here, we are taking the minimum of the inter-cluster distances. So, the distance between even the closest clusters should be more which will eventually make sure that the clusters are far away from each other.

$$\text{Dunn Index} = \frac{\min(\text{Inter cluster distance})}{\underbrace{\max(\text{Intra cluster distance})}_{\text{Clusters are compact}}}$$

### 3.6. K-Means Clustering Algorithm

Let’s now take an example to understand how K-Means actually works:



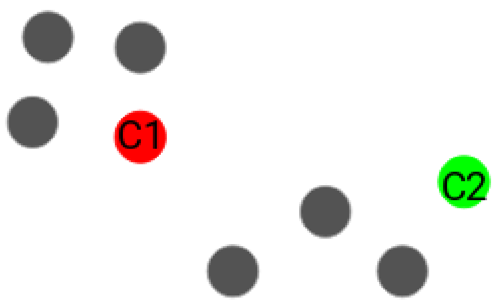
We have these 8 points, and we want to apply k-means to create clusters for these points. Here’s how we can do it.

1. **Choose the number of clusters *k***

The first step in k-means is to pick the number of clusters, **k.**

2. **Select k random points from the data as centroids**

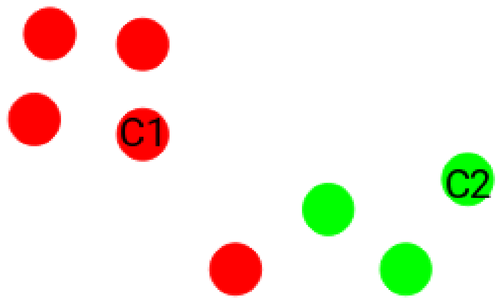
Next, we randomly select the centroid for each cluster. Let’s say we want to have 2 clusters, so k is equal to 2 here. We then randomly select the centroid:




Here, the red and green circles represent the centroid for these clusters.

3. **Assign all the points to the closest cluster centroid**

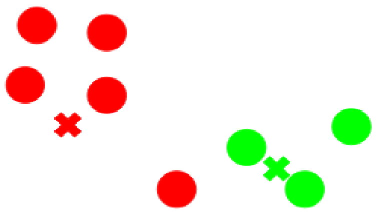
Once we have initialized the centroids, we assign each point to the closest cluster centroid:




Here you can see that the points closer to the red point are assigned to the red cluster, whereas the points closer to the green point are assigned to the green cluster.

**5. Recompute the centroids of newly formed clusters**

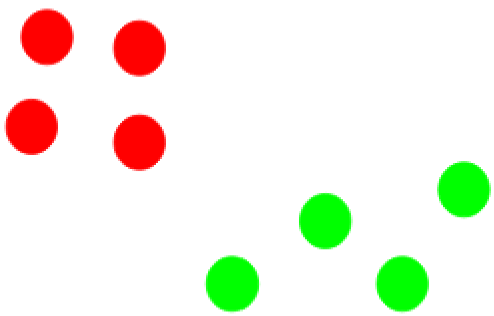
Now, once we have assigned all of the points to either cluster, the next step is to compute the centroids of newly formed clusters:



Here, the red and green crosses are the new centroids.

**6. Repeat steps 3 and 4**

We then repeat steps 3 and 4:



The step of computing the centroid and assigning all the points to the cluster based on their distance from the centroid is a single iteration. Now the question is when should we stop this process? It can't run till eternity, right?

### 3.7 Stopping Criteria for K-Means

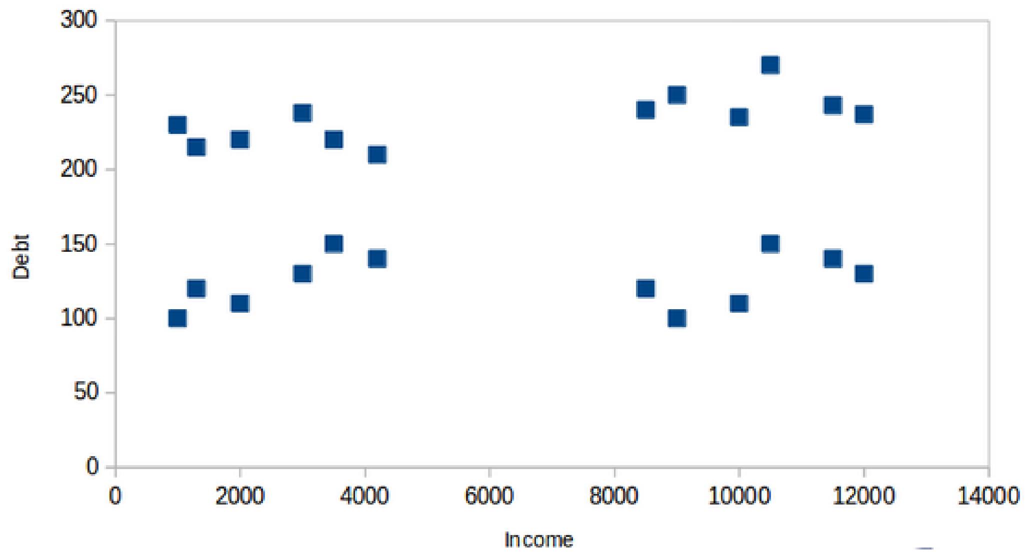
There are essentially three stopping criteria that can be adopted to stop the K-means algorithm:

- Centroids of newly formed clusters do not change
- Points remain in the same cluster
- Maximum number of iterations is reached

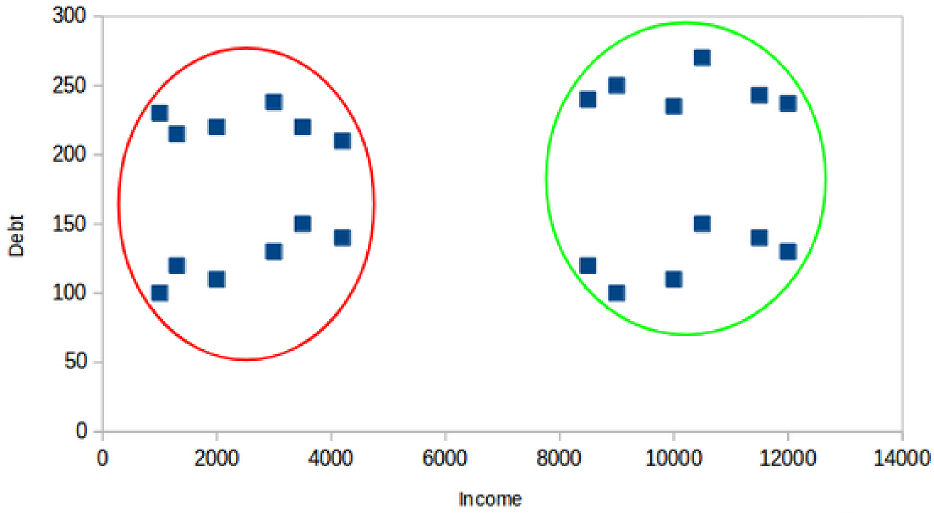
### 3.8 How to choose the right number of clusters in K-Means

One of the most common doubts everyone has while working with K-Means is selecting the right number of clusters.

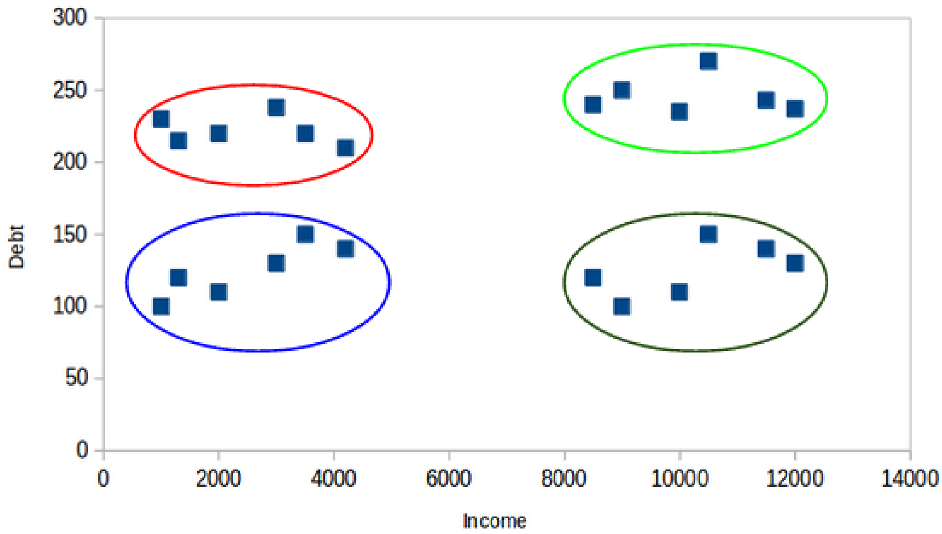
Let's look at a technique that will help us choose the right value of clusters for the K-Means algorithm. Let's take the customer segmentation example that we saw earlier. To recap, the bank wants to segment its customers based on their income and amount of debt:



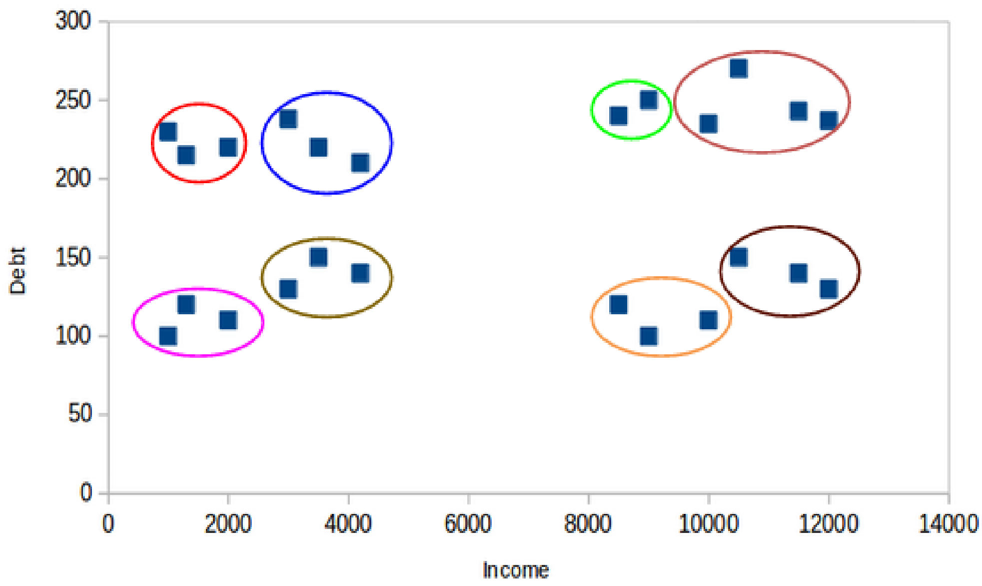
Here, we can have two clusters which will separate the customers as shown below:



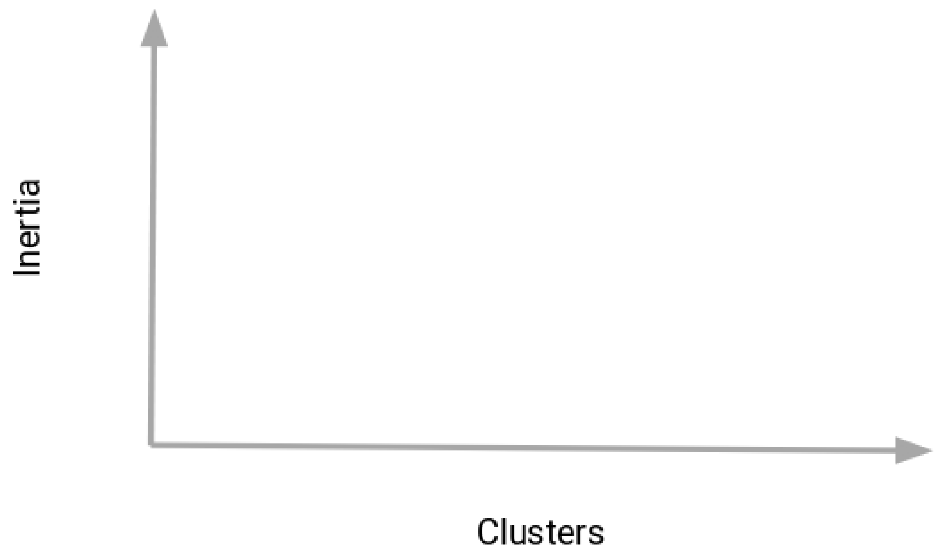

All the customers with low income are in one cluster, whereas the customers with high income are in the second cluster. We can also have 4 clusters:



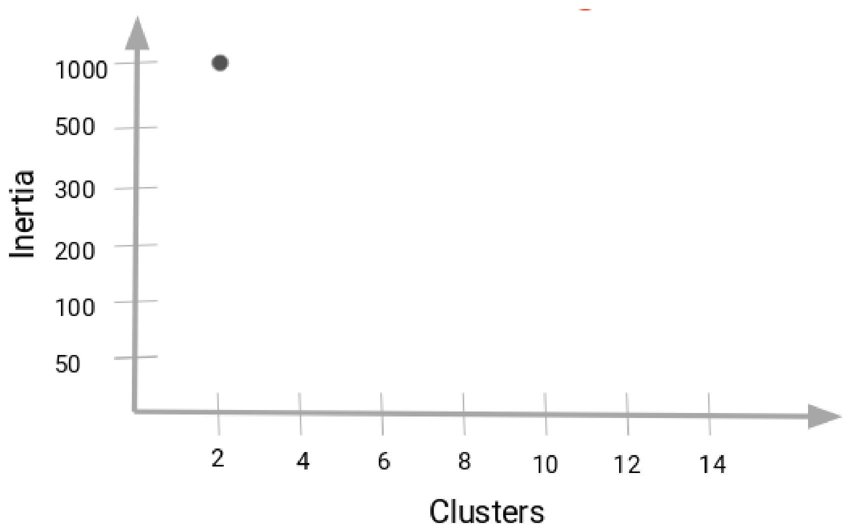

Here, one cluster might represent customers who have low income and low debt; another cluster is where customers have high income and high debt, and so on. There can be 8 clusters as well:



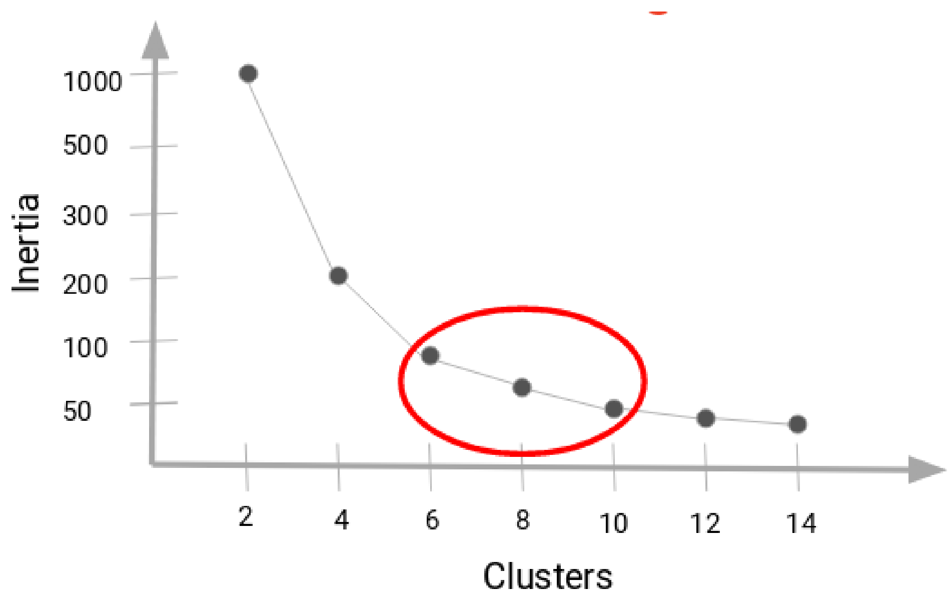

One thing we can do is plot a graph, also known as an elbow curve, where the x-axis will represent the number of clusters and the y-axis will be an evaluation metric. Let's say inertia for now. We can choose any other evaluation metric like the Dunn index as well:




Next, we will start with a small cluster value, say 2. Train the model using 2 clusters, calculate the inertia for that model, and finally plot it in the above graph. Let's say we got an inertia value of around 1000:




Now, we will increase the number of clusters, train the model again, and plot the inertia value. This is the plot we get:




When we changed the cluster value from 2 to 4, the inertia value reduced sharply. This decrease in the inertia value reduces and eventually becomes constant as we increase the number of clusters further. **Here, we can choose any number of clusters between 6 and 10.** We can have 7, 8, or even 9 clusters. We must also look at the computation cost while deciding the number of clusters.

> *the cluster value where this decrease in inertia value becomes constant can be chosen as the right cluster value for our data.*

# IV. Exploratory Data Analysis

### 4.1. About the dataset

**Context:** The dataset used for this project is a comprehensive collection of player data spanning from FIFA 15 to FIFA 22, focusing on the Career Mode of the popular video game series. The dataset can be found in the popular [website Kaggle here](). This dataset provides a unique opportunity to analyze the evolution of player attributes and performances across eight consecutive versions of the game. It opens the door to various intriguing analyses, such as tracking changes in player skills over time, assessing ideal budgets for assembling competitive teams, and exploring the trends among the top percentage of players.

**Content:** The dataset encompasses a wealth of information, offering a holistic view of FIFA players and their attributes. Some key aspects of the dataset include:

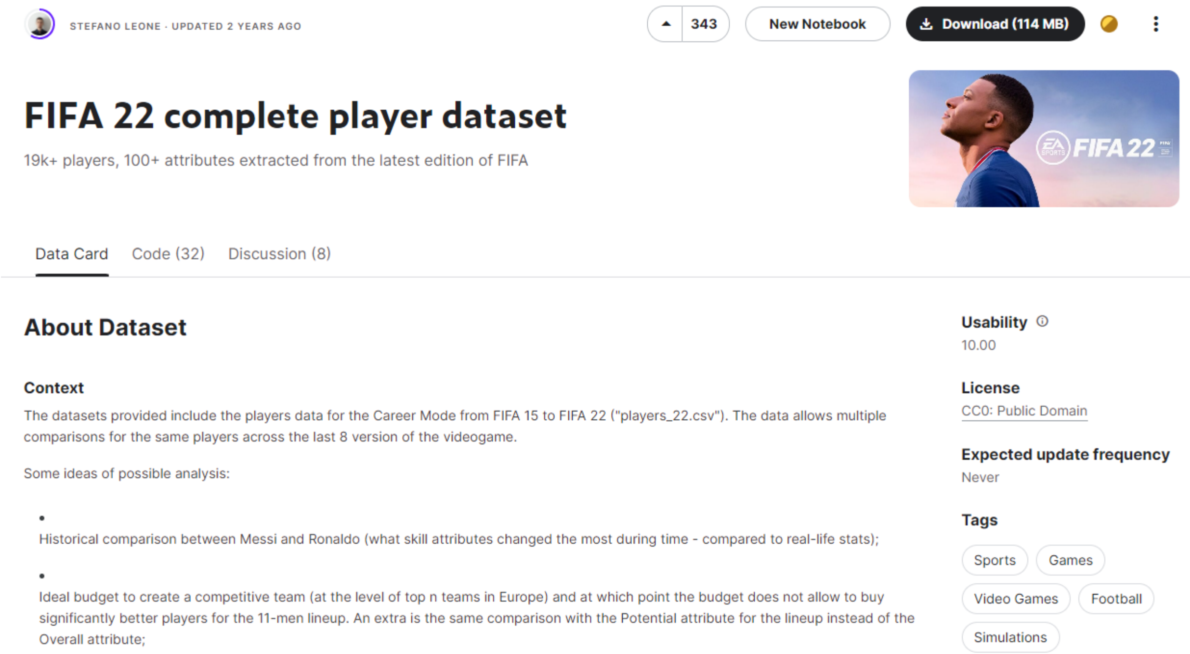
- **Historical Player Data:** It covers all players featured in FIFA 15 through FIFA 22, allowing for in-depth historical comparisons.

- **Attributes:** More than 100 attributes are provided for each player, ranging from attacking and defensive skills to mentality and goalkeeping abilities.

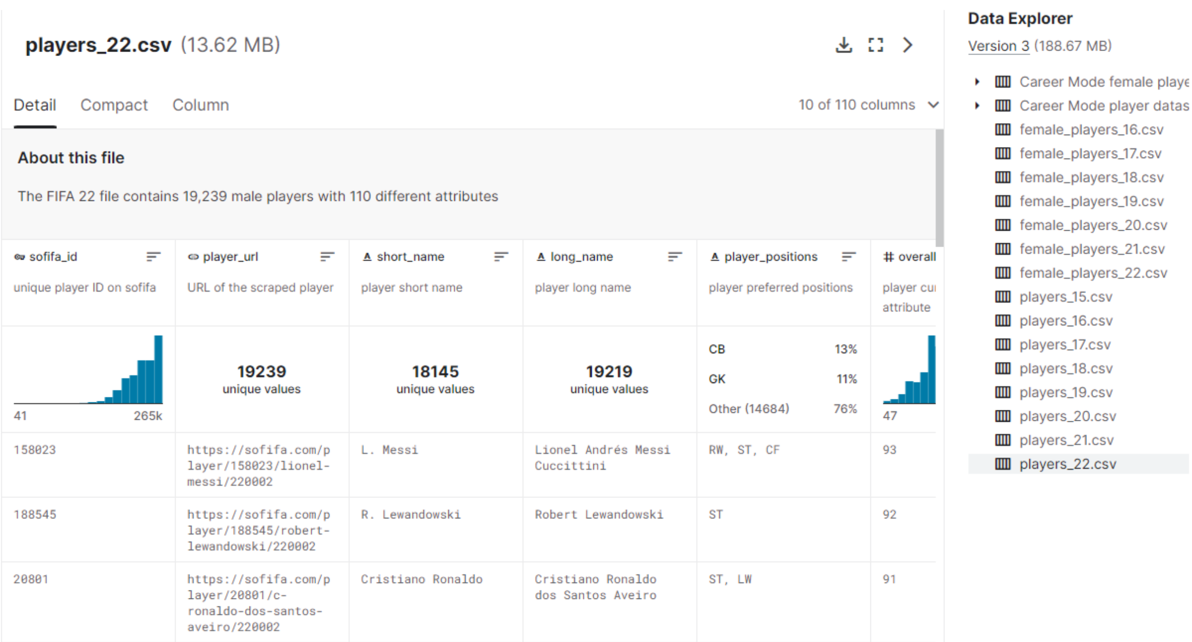
- **URL References:** The dataset includes links to scraped player profiles and associated images, including player faces, club logos, and national team logos.

- **Player Positions:** Information about player positions, along with their roles in club and national teams, provides context for their in-game roles.

- **Personal Data:** Nationality, club affiliation, date of birth, wage, and salary details offer insights into players' real-world backgrounds.




In our case, we're only interested in the FIFA 22 players data. That's why we're going to be using the **players_22.csv** dataset.




In summary, this dataset offers a comprehensive and meticulously updated collection of FIFA player data, spanning eight game editions and encompassing diverse attributes and player information. Its richness and historical depth provide an excellent foundation for in-depth analyses and insights into the evolution of virtual football players' skills and attributes.

## 4.2. Importing the dataset and libraries

Enough theory, now let's dive deep into the code. We will first import the required libraries:

I. Importing libraries

```
# Data analysis and wrangling
import pandas as pd
import numpy as np
import random as rnd
import json

# Visualization
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
import seaborn as sns
%matplotlib inline

# Pre processing
from sklearn import preprocessing

# Machine Learning
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans

from pylab import rcParams
rcParams['figure.figsize'] = 15, 10
```

[3] ✓ 0.0s Python

Now, we're gonna import the dataset from our local folder:

II. Importing the dataset

```
df = pd.read_csv("../data/players_22.csv", low_memory=False)
df.head(3)
```

[4] ✓ 1.5s

| | sofifa_id | player_url | short_name | long_name | player_positions | overall | potential | value_eur | wage_eur | age | ... | lcb | cb | rcb |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 158023 | https://sofifa.com/player/158023/lionel-messi/... | L. Messi | Lionel Andrés Messi Cuccittini | RW, ST, CF | 93 | 93 | 78000000.0 | 320000.0 | 34 | ... | 50+3 | 50+3 | 50+3 |
| 1 | 188545 | https://sofifa.com/player/188545/robert-lewand... | R. Lewandowski | Robert Lewandowski | ST | 92 | 92 | 119500000.0 | 270000.0 | 32 | ... | 60+3 | 60+3 | 60+3 |
| 2 | 20801 | https://sofifa.com/player/20801/c-ronaldo-dos-... | Cristiano Ronaldo | Cristiano Ronaldo dos Santos Aveiro | ST, LW | 91 | 91 | 45000000.0 | 270000.0 | 36 | ... | 53+3 | 53+3 | 53+3 |

3 rows × 110 columns

The dataset has 19,239 rows and 110 columns.

### 4.3. Data Cleaning

First we're gonna delete columns that have a lot of missing values (>50%)

III. Cleaning the Dataset

Deleting columns with more than 50% of missing values

```python
cols_to_drop = []
for i in df.columns:
    missing = np.abs((df[i].count() - df[i].shape[0])/df[i].shape[0] * 100)
    if missing > 50:
        print('{} - {}%'.format(i, round(missing)))
        cols_to_drop.append(i)
```

[6] ✓ 0.0s Python

```
club_loaned_from - 94%
nation_team_id - 96%
nation_position - 96%
nation_jersey_number - 96%
player_tags - 93%
player_traits - 51%
goalkeeping_speed - 89%
nation_logo_url - 96%
```

Let's delete the columns **club_loaned_from, nation_team_id, nation_position, nation_jersey_number, player_tags, player_traits, goalkeeping_speed** who have more than 50% of missing values.

```python
df.drop(columns=cols_to_drop, inplace=True)
print(df.shape)
```

[7] ✓ 0.0s Python

```
(19239, 102)
```

Next, we're only going to keep the top Moroccan players with a rating >70.

Now let's keep only players with the Moroccan MA Nationality

```python
df = df[df.nationality_name == 'Morocco']
```

[683] ✓ 0.0s Python

```python
df.shape
```

[684] ✓ 0.0s Python

```
(101, 102)
```

To have a nicer visualization of the clusters afterwards, we'll be keeping only the top players (rating over 70).

```python
df = df[df.overall > 70]
print(df.shape)
```

[685] ✓ 0.0s Python

```
(43, 102)
```

Now we're only left with 43 rows (43 players) and 107 columns (features of a player).

After that, we're going to drop unnecessary columns like the one who contains the urls, and we're going to replace null values with the mean.

Dropping columns with urls in them.

```python
df = df[df.columns.drop(list(df.filter(regex='url')))]

df.shape
```

[686] ✓ 0.0s Python

```
(43, 97)
```

Replacing null values with the mean

```python
df.isnull().sum()
```

[687] ✓ 0.0s Python

```
sofifa_id           0
short_name          0
long_name           0
player_positions    0
overall             0
                   ..
lcb                 0
cb                  0
rcb                 0
rb                  0
gk                  0
Length: 97, dtype: int64
```

```python
df = df.fillna(df.mean())
```

[688] ✓ 0.0s Python

Here's how our dataset looks now after all theses changes:

```python
df = df[cols]
df.head()
```

[691] ✓ 0.0s

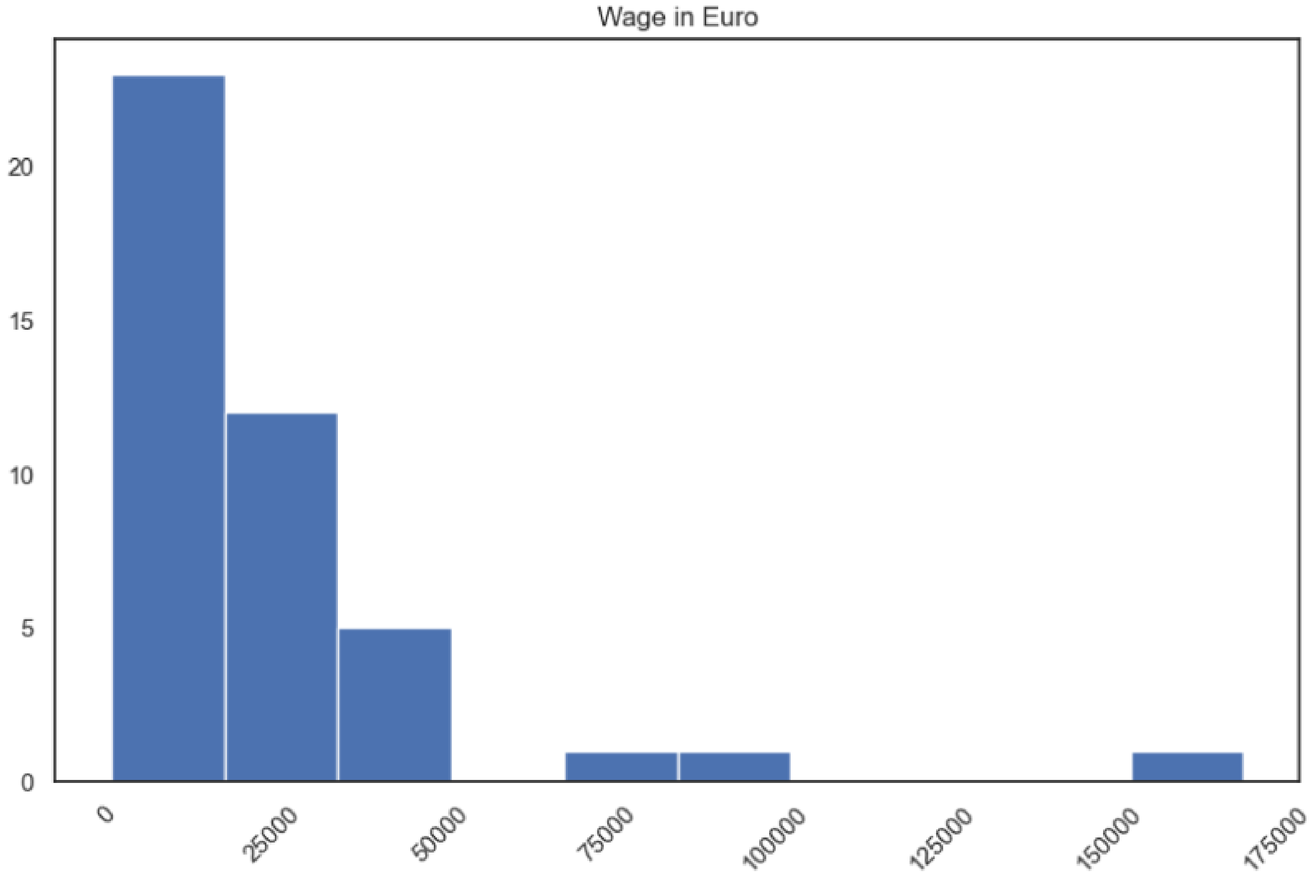
| | short_name | player_positions | overall | potential | value_eur | wage_eur | age | dob | height_cm | weight_kg | ... | ldm | cdm | rdm | rwb | lb | lcb | cb | rcb | rb | gk |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 95 | A. Hakimi | RB, RWB | 85 | 88 | 69500000.0 | 100000.0 | 22 | 1998-11-04 | 181 | 73 | ... | 79+2 | 79+2 | 79+2 | 83+2 | 82+2 | 77+2 | 77+2 | 77+2 | 82+2 | 17+2 |
| 123 | H. Ziyech | RW, CAM | 84 | 84 | 42500000.0 | 170000.0 | 28 | 1993-03-19 | 181 | 65 | ... | 70+3 | 70+3 | 70+3 | 70+3 | 66+3 | 57+3 | 57+3 | 57+3 | 66+3 | 18+3 |
| 238 | Y. Bounou | GK | 82 | 82 | 20500000.0 | 29000.0 | 30 | 1991-04-05 | 192 | 78 | ... | 32+2 | 32+2 | 32+2 | 28+2 | 27+2 | 28+2 | 28+2 | 28+2 | 27+2 | 81+1 |
| 276 | Y. En-Nesyri | ST, LW | 82 | 86 | 44000000.0 | 37000.0 | 24 | 1997-06-01 | 188 | 78 | ... | 54+2 | 54+2 | 54+2 | 55+2 | 53+2 | 52+2 | 52+2 | 52+2 | 53+2 | 19+2 |
| 399 | M. Benatia | CB | 80 | 80 | 7000000.0 | 19000.0 | 34 | 1987-04-17 | 189 | 94 | ... | 73+3 | 73+3 | 73+3 | 69+3 | 71+3 | 79+1 | 79+1 | 79+1 | 71+3 | 15+3 |

5 rows × 95 columns

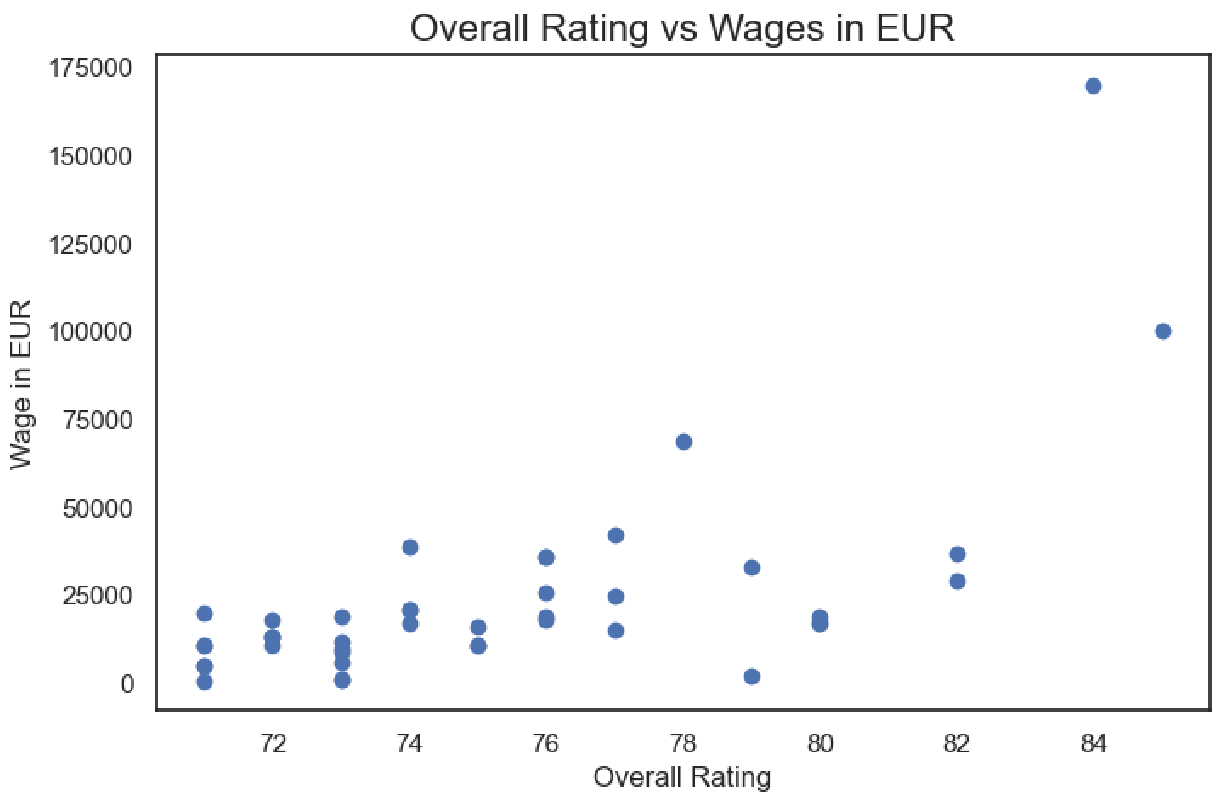
## 4.4. Exploratory Data Analysis

Let's explore our data set by visualizing some interesting features and analyzing them.

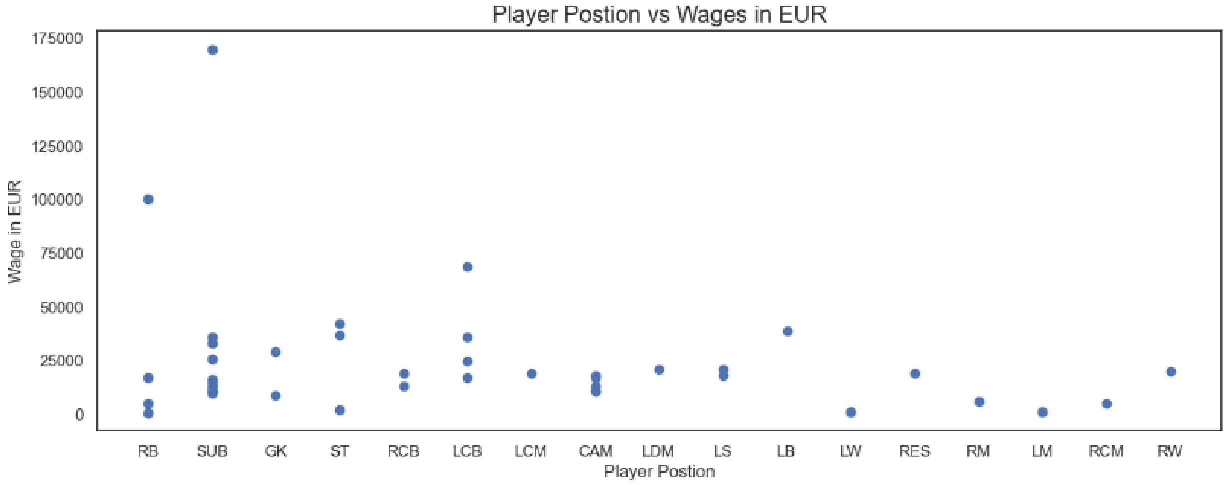
Starting with the distribution of the annual wage in euro the top 50 Moroccan football players get.



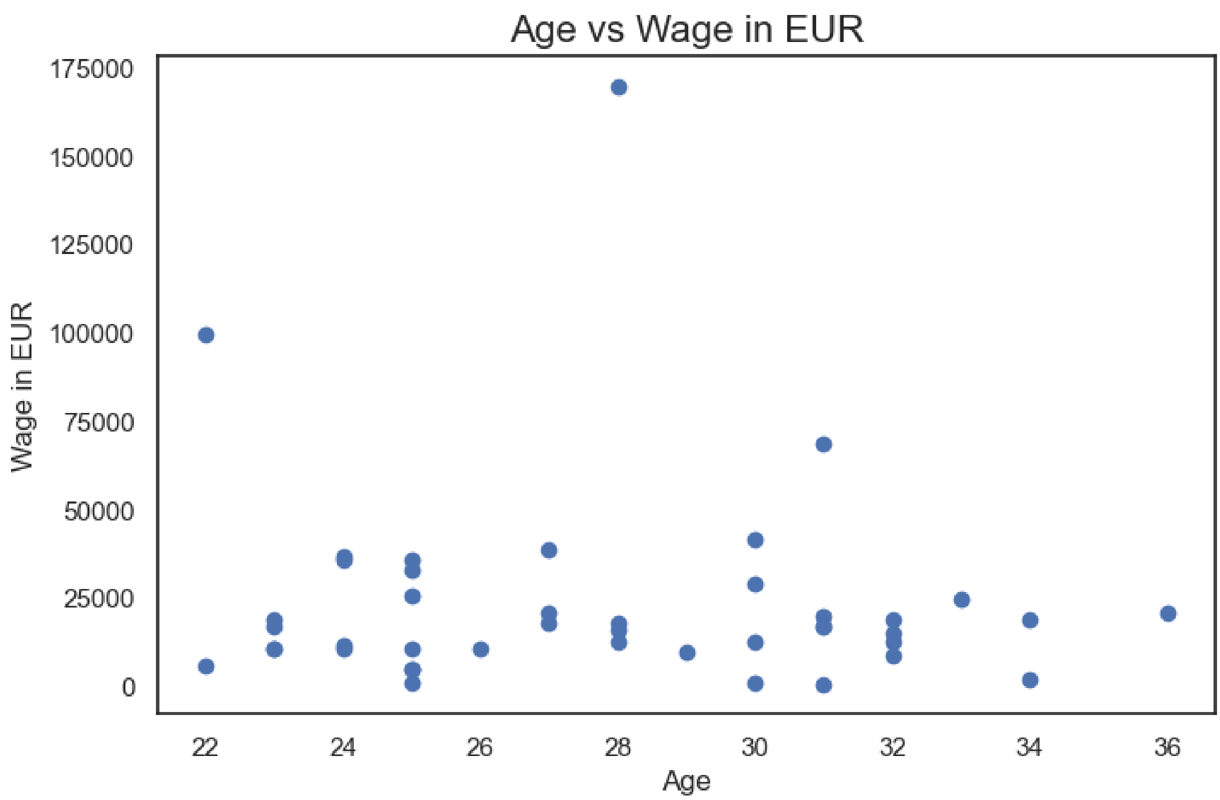

We can see the majority of the players make less than 50,000 Euros per year. Now let's see how the skill set of the player influences his wage.



There's a positive correlation between the overall rating of a player and his annual wage. Let's see now if the player position makes a difference in the wage.

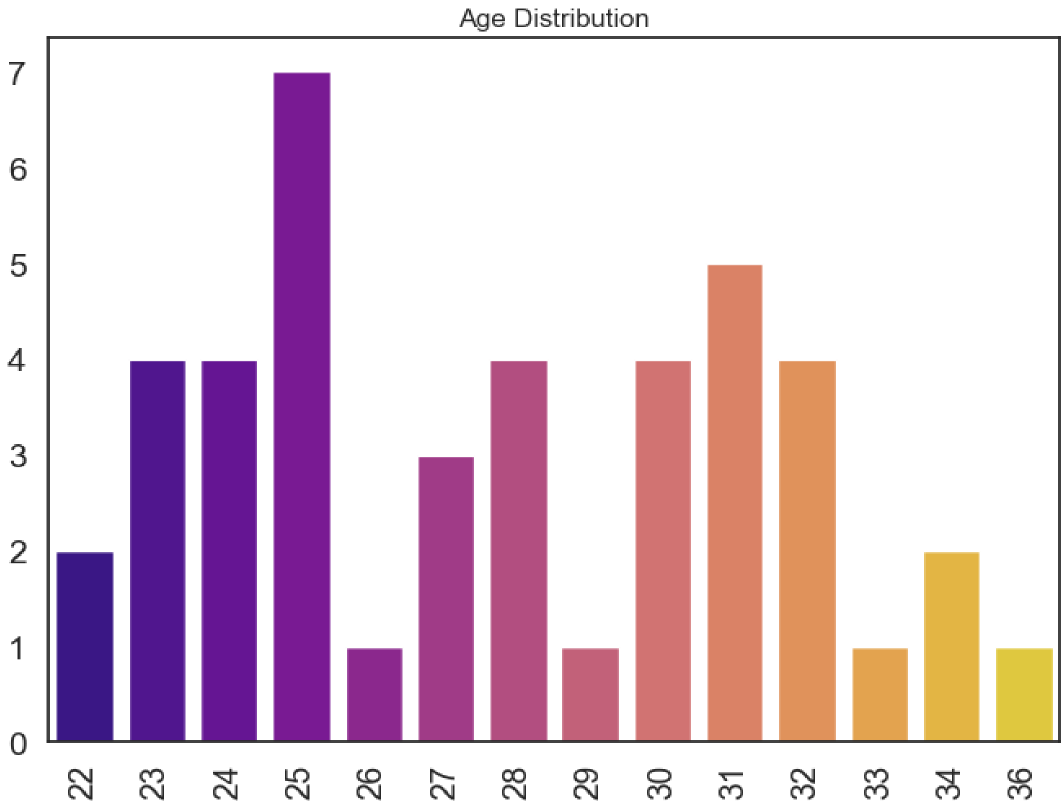



We can clearly see that the player position has no influence on the annual wage. Maybe the age of a player has a role in that?

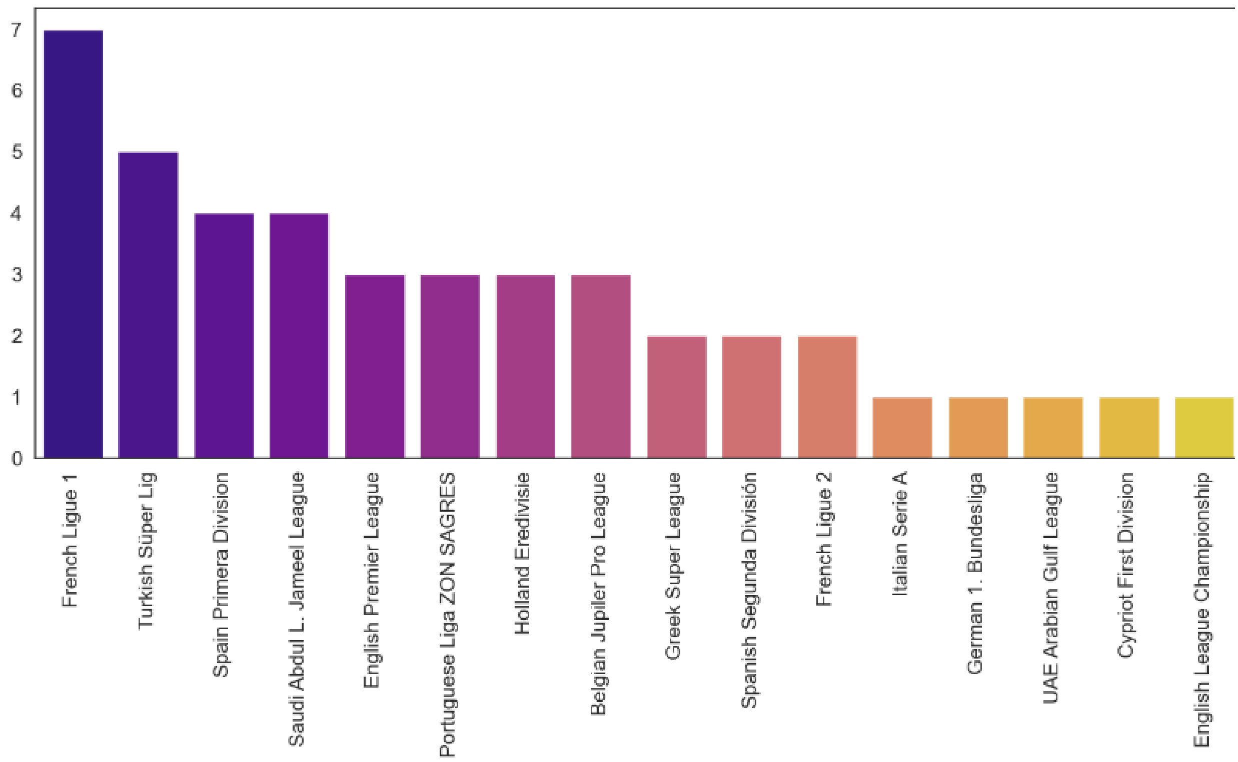



With just a few players we can't make a generalization but it seems that players between 24 and 31 years make slightly more money than the others. This is explained by the player in his prime in that period of time, after that his skills decrease and with it his wage.

The age distribution of the moroccan players:

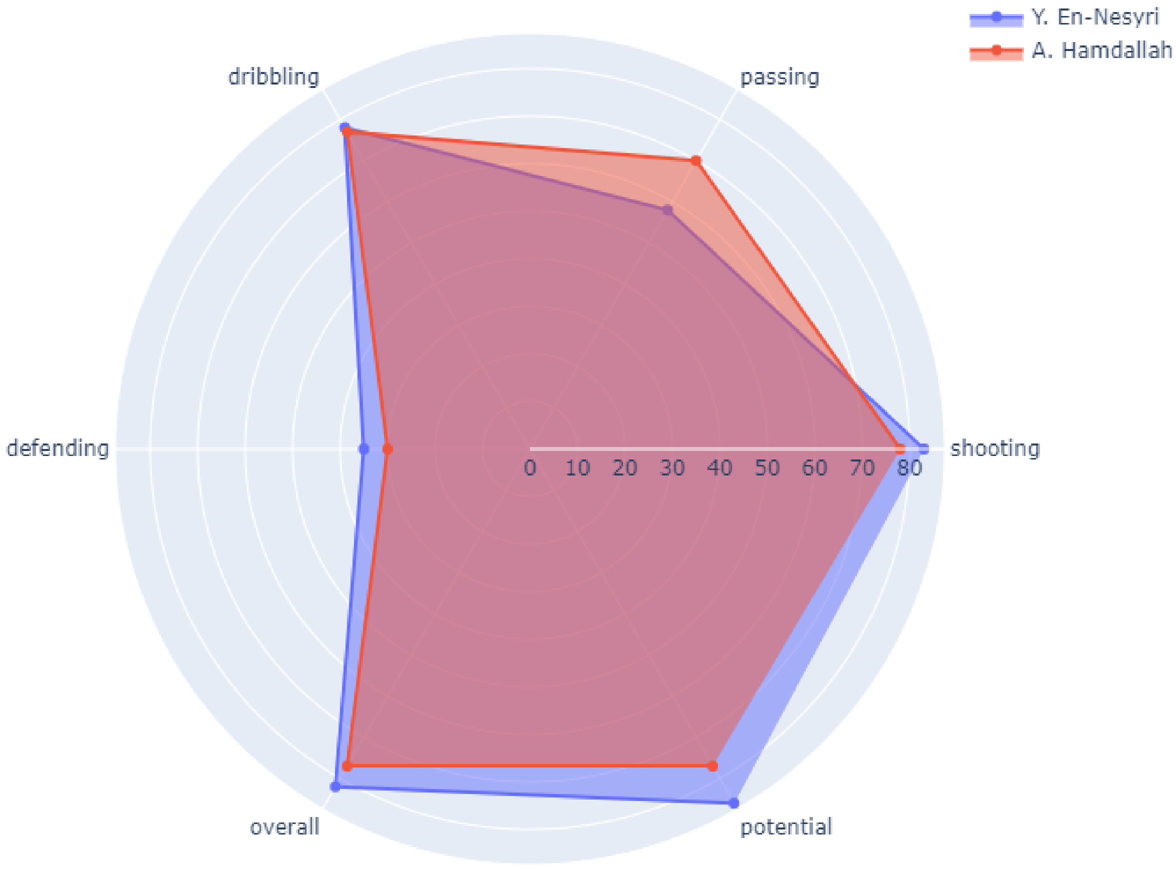



Most players are between the ages of 22 and 28 years old.




The French league has the most Moroccan football players with 7, the Turkish league with 5 and the Spanish with 4 equal with the Saudi league.

Now let's do a fun comparison between two of the most controversial players in our Football National Team of Morocco, Youssef En-Nesyri and Hamdallah who are both strikers but the public opinion on both of them are so different.




En-Nesyri is more skilled than Hamdallah overall, except for the passing when Hamdallah is slightly better. I will let you be the judge of who should be in the starting XI.

# V. Model Building

## 5.1. Data pre-processing

First thing, let's prepare our datasets for our model. We're gonna save the short names of our players before dropping all non numeric columns.

```python
names = df.short_name.tolist()

df = df.drop(['short_name'], axis = 1)
```

[702] ✓ 0.0s Python

```python
df.head()
```

[703] ✓ 0.0s Python

| | player_positions | overall | potential | value_eur | wage_eur | age | dob | height_cm | weight_kg | club_team_id | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 95 | RB, RWB | 85 | 88 | 69500000.0 | 100000.0 | 22 | 1998-11-04 | 181 | 73 | 73.0 | ... |
| 123 | RW, CAM | 84 | 84 | 42500000.0 | 170000.0 | 28 | 1993-03-19 | 181 | 65 | 5.0 | ... |
| 238 | GK | 82 | 82 | 20500000.0 | 29000.0 | 30 | 1991-04-05 | 192 | 78 | 481.0 | ... |
| 276 | ST, LW | 82 | 86 | 44000000.0 | 37000.0 | 24 | 1997-06-01 | 188 | 78 | 481.0 | ... |
| 399 | CB | 80 | 80 | 7000000.0 | 19000.0 | 34 | 1987-04-17 | 189 | 94 | 111117.0 | ... |

5 rows × 94 columns

Now we're gonna make a new processed dataset where we remove all non numeric columns.

Now we're gonna remove all non numeric colums.

```python
numeric_columns = df.select_dtypes(include=['number']).columns
df_processed = df[numeric_columns]

df_processed.head()
```

[704] ✓ 0.0s Python

| | overall | potential | value_eur | wage_eur | age | height_cm | weight_kg | club_team_id | league_level | club_jersey_nu |
|---|---|---|---|---|---|---|---|---|---|---|
| 95 | 85 | 88 | 69500000.0 | 100000.0 | 22 | 181 | 73 | 73.0 | 1.0 | |
| 123 | 84 | 84 | 42500000.0 | 170000.0 | 28 | 181 | 65 | 5.0 | 1.0 | |
| 238 | 82 | 82 | 20500000.0 | 29000.0 | 30 | 192 | 78 | 481.0 | 1.0 | |
| 276 | 82 | 86 | 44000000.0 | 37000.0 | 24 | 188 | 78 | 481.0 | 1.0 | |
| 399 | 80 | 80 | 7000000.0 | 19000.0 | 34 | 189 | 94 | 111117.0 | 1.0 | |

5 rows × 56 columns

### 5.2. Manuel Implementation of K-Means Algorithm

**Pseudocode:**

1. Scale data to standardize values

2. Initialize random centroids

3. Get labels for each data point

4. Create new centroids

5. Plot the centroids

6. Repeat 3-5 until the centroids stop changing

First we're gonna make a new dataframe with only few features:

```
features = ["overall", "potential", "wage_eur", "value_eur", "age"]

df_man = df[features]

df_man.head()
```

[758] ✓ 0.0s

| | overall | potential | wage_eur | value_eur | age |
|---|---|---|---|---|---|
| 95 | 85 | 88 | 100000.0 | 69500000.0 | 22 |
| 123 | 84 | 84 | 170000.0 | 42500000.0 | 28 |
| 238 | 82 | 82 | 29000.0 | 20500000.0 | 30 |
| 276 | 82 | 86 | 37000.0 | 44000000.0 | 24 |
| 399 | 80 | 80 | 19000.0 | 7000000.0 | 34 |

Second, we are gonna initialize random centroids.

```python
def random_centroids(data, k):
    centroids = []
    for i in range(k):
        centroid = data.apply(lambda x: float(x.sample()))
        centroids.append(centroid)
    return pd.concat(centroids, axis=1)
```
[759] ✓ 0.0s Python

```python
centroids = random_centroids(df_man, 5)
centroids
```
[760] ✓ 0.0s Python

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| overall | 78.0 | 74.0 | 77.0 | 72.0 | 76.0 |
| potential | 79.0 | 76.0 | 80.0 | 81.0 | 79.0 |
| wage_eur | 6000.0 | 26000.0 | 5000.0 | 69000.0 | 11000.0 |
| value_eur | 4000000.0 | 9000000.0 | 5500000.0 | 29500000.0 | 20500000.0 |
| age | 24.0 | 25.0 | 23.0 | 25.0 | 30.0 |

Now we are gonna get the labels and create new centroids.

```python
def get_labels(data, centroids):
    distances = centroids.apply(lambda x: np.sqrt(((data - x) ** 2).sum(axis=1)))
    return distances.idxmin(axis=1)
```
[778] ✓ 0.0s Python

```python
labels = get_labels(df_man, centroids)
```
[779] ✓ 0.0s Python

```python
labels.value_counts()
```
[780] ✓ 0.0s Python

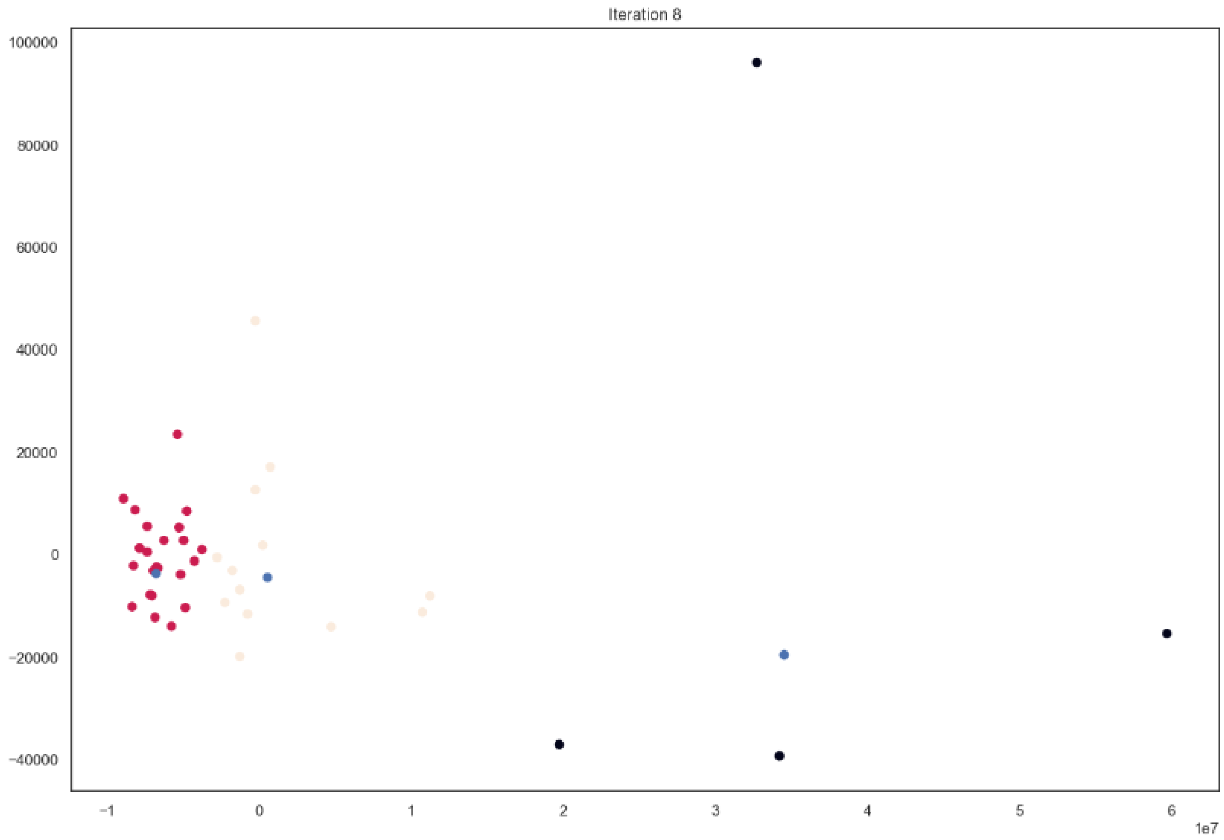
```
2    25
1    14
0     4
dtype: int64
```

```python
def new_centroids(data, labels, k):
    centroids = data.groupby(labels).apply(lambda x: np.exp(np.log(x).mean())).T
    return centroids
```
[764] ✓ 0.0s Python

Let's now plot the centroids. Here's the function that does the plotting.

```python
def plot_clusters(data, labels, centroids, iteration):
    pca = PCA(n_components=2)
    data_2d = pca.fit_transform(data)
    centroids_2d = pca.transform(centroids.T)
    clear_output(wait=True)
    plt.title(f'Iteration {iteration}')
    plt.scatter(x=data_2d[:,0], y=data_2d[:,1], c=labels)
    plt.scatter(x=centroids_2d[:,0], y=centroids_2d[:,1])
    plt.show()
```
✓ 0.0s Python

Here's the final plot with 8 iterations:




```
centroids
```

[782] ✓ 0.0s

| | 0 | 1 | 2 |
|---|---|---|---|
| overall | 8.272762e+01 | 7.282334e+01 | 7.754433e+01 |
| potential | 8.573731e+01 | 7.468393e+01 | 7.912666e+01 |
| wage_eur | 5.718405e+04 | 9.781769e+03 | 2.026673e+04 |
| value_eur | 4.424988e+07 | 2.925955e+06 | 1.031069e+07 |
| age | 2.414793e+01 | 2.775817e+01 | 2.821005e+01 |

We can see with our manual implementation that we got 3 clusters.

### 5.3. Implementation using Scikit-learn

First we're going to normalize the data to make sure our algorithm does not make assumptions about the data having a Gaussian distribution.

```python
x = df_processed.values
scaler = preprocessing.MinMaxScaler()
x_scaled = scaler.fit_transform(x)
X_norm = pd.DataFrame(x_scaled)
```
✓ 0.0s Python

Second we're going to do a PCA to reduce the numerous columns into 2 to get a better visualization after.

```python
pca = PCA(n_components = 2) # 2D PCA for the plot
reduced = pd.DataFrame(pca.fit_transform(X_norm))
```
✓ 0.0s

Now we can initialize our KMeans model with 5 clusters and then fit the reduced data to the model, get the labels and the centroid values same as the manual pseudocode before.

```python
kmeans = KMeans(n_clusters=5)

# Fit the input data to the model
kmeans = kmeans.fit(reduced)

# Get the cluster labels
labels = kmeans.predict(reduced)

# Centroid values
centroid = kmeans.cluster_centers_

# Cluster values
clusters = kmeans.labels_.tolist()
```

Now let's see the different clusters and where each player belongs to.

```python
reduced['cluster'] = clusters
reduced['name'] = names
reduced.columns = ['x', 'y', 'cluster', 'name']
reduced.head(10)
```
✓ 0.0s

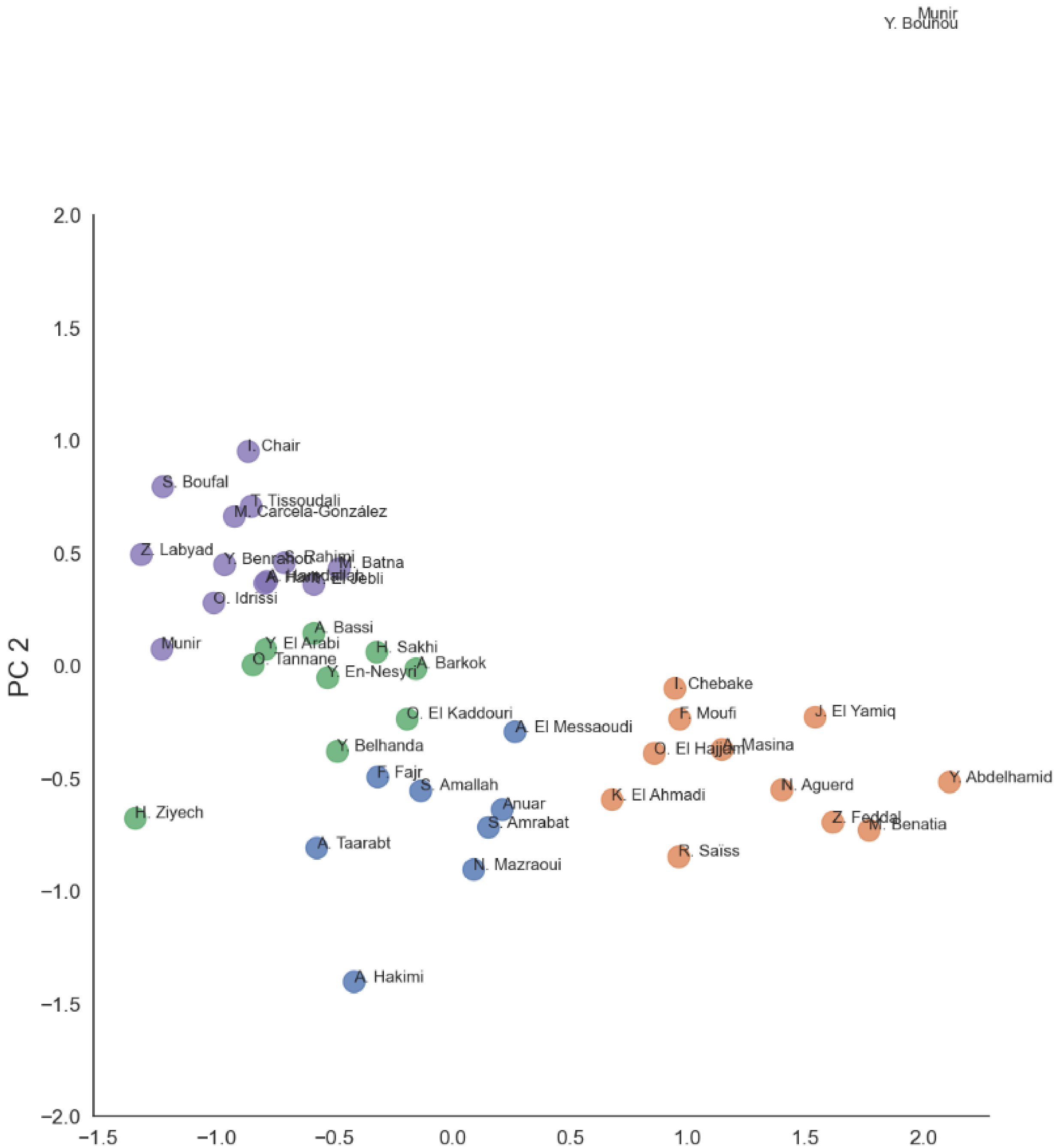
| | x | y | cluster | name |
|---|---|---|---|---|
| 0 | -0.416940 | -1.402227 | 0 | A. Hakimi |
| 1 | -1.347289 | -0.674665 | 2 | H. Ziyech |
| 2 | 1.833563 | 2.825214 | 3 | Y. Bounou |
| 3 | -0.530509 | -0.050283 | 2 | Y. En-Nesyri |
| 4 | 1.767683 | -0.727340 | 1 | M. Benatia |
| 5 | 1.614146 | -0.695639 | 1 | Z. Feddal |
| 6 | 0.089846 | -0.904948 | 0 | N. Mazraoui |
| 7 | -0.792849 | 0.075327 | 2 | Y. El Arabi |
| 8 | -1.233885 | 0.076687 | 4 | Munir |
| 9 | 0.961730 | -0.845448 | 1 | R. Saïss |

We have now grouped our player into five clusters. If you are familiar with these players and their positions, you can see that they're correctly grouped.

For example:

- Noussair Mazraoui and Achraf Hakimi are two Right Backs (RB) in the same cluster (0).
- Fedal and Benatia are both Central Back (CB) they are in the same cluster (1).
- H. Ziyech and Y. En-Nesyri both Strikers (ST) are in the same cluster (2).

Let's see the plot of these clusters:



In this plot, we can see:

🥅 In the top right, Y. Bounou and Munir Mhammedi the two goalkeepers of the national team.

🟧 In orange we have our strikers like H. Ziyech, Y. En-Nesyri and S. Boufal.

🟩 In green offensive midfielders like A. Taarabt and S. Amallah and also offensive left and right backs like A. Hakimi and N. Mazraoui.

🟪 In purple we have the defensive midfielders like K. El Ahmadi.

🟦 And in the blue we have the central backs like M. Benatia and N. Aguerd

Let's confirm again that those clusters are indeed accurate:

```
df_decision = pd.merge(df, reduced, on="name")
df_decision = df_decision[['name', 'player_positions', 'cluster']]

df_decision.head(8)
```

✓ 0.0s

| | name | player_positions | cluster |
|---|---|---|---|
| 0 | A. Hakimi | RB, RWB | 0 |
| 1 | H. Ziyech | RW, CAM | 2 |
| 2 | Y. Bounou | GK | 3 |
| 3 | Y. En-Nesyri | ST, LW | 2 |
| 4 | M. Benatia | CB | 1 |
| 5 | Z. Feddal | CB | 1 |
| 6 | N. Mazraoui | RB | 0 |
| 7 | Y. El Arabi | ST | 2 |

We can clearly see that players with similar positions are indeed in the same cluster. To see all the details, please see [the notebook here.]

# VI. Conclusion

In this project, our focus was exclusively to Moroccan football players within the FIFA gaming series. By applying the K-Means clustering algorithm to this specific subset, we aimed to illuminate the varying skillsets and attributes of these players. With this analysis we discovered significant insights and valuable takeaways.

Our exploration unveiled the multifaceted talents of Moroccan football players featured in FIFA. Through skill-based clustering, we gained a nuanced understanding of their abilities, from offensive prowess to defensive strengths. This newfound clarity provides a robust foundation for player categorization, a crucial aspect of talent assessment.

The implementation of the K-Means clustering algorithm has proven to be an effective tool for organizing Moroccan players into meaningful clusters. This data-driven approach empowers decision-makers in the football world by offering a structured method for player classification, aiding in team formation and strategic planning.

Beyond the immediate clustering results, our project opens doors to deeper analysis. By delving into the defining attributes of each cluster, we can identify trends and changes in Moroccan players' skillsets across different FIFA editions. This valuable information can provide insights into the virtual representation of players and potentially correlate with real-world football trends.

Also, our project underscores the pivotal role of data-driven decision-making in the realm of sports analytics. By harnessing data mining and clustering techniques, football clubs, scouts, and analysts can make informed choices, fine-tune team compositions, and gain a competitive edge in both virtual and real-world football.

# Bibliographie


https://www.driblab.com/scouting-platform/scouting-platform-the-power-of-data-as-investment-value/

https://www.mygreatlearning.com/blog/how-football-scouts-leverage-data-science/

https://medium.datadriveninvestor.com/k-means-clustering-4a700d4a4720?gi=c487028fadc7

https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

https://www.kaggle.com/datasets/stefanoleone992/fifa-22-complete-player-dataset

https://realpython.com/k-means-clustering-python/

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

https://www.geeksforgeeks.org/principal-component-analysis-with-python/

https://www.fifplay.com/encyclopedia/position/