Match Report Analysis

Emilio Zavala Miceli
May, 2024

Table of Contents

1. Introduction

The objective of this project is to make 2 different dashboards scraping data, the two main websites we are going to use are SofaScore and Fotmob. The first one is going to have a general recap of the match, with the main stats and shots of both teams. The second one is only the shots of the teams and the player of the match.

2. Problem Statement

The main problem of the project was to get the data, often the data of the games is too brief or released long after the game or tournament, with these in mind the main objective was to get the most data possible as soon as the match ends. The best way to get the data of a specific match is scraping webs like SofaScore or Fotmob, these webs update the match data minute to minute in their webs, this way is easier to get the data.

3. Methodology

This project is based on web scrapping using Python. First, we need to scrap the webs we are going to use to get the data, with libraries like *'requests'* for the Fotmob[1] data and *'ScraperFC' for the* SofaScore[2] data. These data are going to be plotted in a dashboard generated with *'matplotlib'*.

1. Scrap information from SofaScore and Fotmob
2. Get game data
3. Get both teams stats
4. POM[3] stats
5. Shotmaps of both teams
6. Match Momentum
7. Players and Teams shots
8. Heatmap
9. Dashboard

4. Implementation

First, we need to say the libraries we use, for these, we are going to divide them into 2 categories, the first ones are for scraping and manipulating the data, and the second one is for the visualization of the dashboard. The libraries used to scrap and manage the data were *request, ScraperFC, numpy and pandas.* For the visualization we used *mplsoccer and matplotlib.* Other libraries used for the use of images, twitter and saving files were *os, PIL, io and tweepy.*

4.1 Scraping data

SofaScore web is going to give us the match stats while fotmob is giving us the shotmaps as well as advanced statistics of the game like the xG of each team.

SofaScore

This web is the easier to scrap thanks to the library *ScraperFC* package created by the GitHub user oseymour[4]. This package has the direct function call 'Sofascore()', this helps make some calls like the one used to get the general stats of a match 'sofascore.get_general_match_stats(match_url)'. This gives a DataFrame with all the general stats of the match.

Fotmob

Fotmob is the difficult one, for this website we need to use the request package. First, we must inspect the fotmob web, in the network section we must search for 'matchDetails', here we are going to have the API we want to request. We make two dictionaries, one of the headers and another for the payload (both empty), then me call the function requests.request(HTTP method, API endpoint, headers, data), the HTTP method we are using is "GET", the API endpoint is the URL of matchDetails API, headers is the empty dictionary and data the payload one, this is going to be inside our response variable. The response variable contains the data received from the server as a JSON-formatted string. To convert this JSON string into a Python dictionary ,we utilize the .json() method provided by the response object in the requests library. This method parses the JSON string and returns a dictionary representation of the data. We store this dictionary in a variable name data, which we can then use to access and manipulate the JSON data within our Python code.

We extract shot data from the JSON response by navigating through its structure using a series of dictionary keys (['content']['shotmap']['shots']). This data represents various attributes of shots taken during the match. With the help of the pandas library, we convert this extracted data into a DataFrame (dfShots). Each row in the df corresponds to a shot, and the columns represent different attributes of the shots, such as player name, location of the shot, minute, etc.

4.2 Game data

Match data

We begin by retrieving match data from a dictionary (data). By traversing through the dictionary using specific keys, we extract information such as the round, match time, season, and score. Additionally, we determine if the match outcome was decided by penalties. To obtain the league information, we utilize the 'sofascore.get_match_data(match_url)' function, which returns a dictionary containing various details. From this dictionary, we access the league name using appropriate keys.

Teams data

This data retrieves essential information about both the home and away teams from the provided data. For the home team, it captures details such as the team's name, team ID, team color, and image URL. Similarly, for the away team, the code extracts corresponding information. In this part of the code, we create a function that checks if the colors of both teams are similar, if the colors are similar both teams get a default color (red and blue), which makes the visualization always clear.

4.3 Teams Stats

Firstly, we get the xG of the match, with the data dictionary and a couple of its keys. Next, we declare a list named 'all_stats' to encompass all the statistics we want to know about the teams such as xG, possession, shots, fouls, and big chances.
Statistical data for the home team is gathered into a dictionary named 'home_data'. This includes extracting specific statistical values such as xG, possession, shots, fouls, and big chances from the

DataFrame ‘df_matchStats’. Similarly, statistical data for the away team is collected into a dictionary named ‘away_data’.

4.4 POM Stats

In these case as we get the information from two different sources, we get first the Fotmob POM (‘pomFM’) and then the SofaScore one (‘pomSS’). The stats shown in the dashboard as well as the shots plot is going to be with the fotmob one and the heatmap shown is going to be with the SofaScore player.

For the ‘pomFM’ stats we first define a dictionary name ‘pomDict’ that holds details of the player of the match, such as their name, position, role, and team. These details are sourced from the ‘data’ variable.

Following this, two list are created: one listing the possible positions a player may hold (goalkeeper, defender, midfielder, attacker), and another listing the statistical keys of interest corresponding to each position. These keys represent the specific performance metrics we seek to analyze for the player of the match. Based on the player’s positions ‘pom_position’, we select the appropriate keys from the predefined list, then retrieves the values corresponding to these keys for the player.

To get the ‘pomSS’, we get the ratings of both teams in their respective DataFrame and get the maximum value of rating of each team, then we compared both max rating and get the POM for SofaScore, we create a function to see what the opponent team is of the ‘pomSS’.

4.5 Shotmaps

Teams shotmaps

With the DataFrame containing all the shots in the match ‘dfShots’, we filtered the shots in two separated DataFrames, the home and away shots, for each team then we separated in goals and non-goals shots.

For the shotmap we segregate shots taken by the home and away teams into separate DataFrames, namely ‘home_shots’ and ‘away_shots’, utilizing teams ids. Further categorization ensures with ‘home_goal_shots’ and ‘away_goal_shots’ exclusively containing shots resulting in goals.

Additionally, non-goal shots are filtered out and stored in ‘home_non_goal_shots’ and ‘away_non_goal_shots’.

POM shotmap

For the player shotmap, we get only the columns with the name of the player in the ‘dfShots’, as with the teams, we will divide the shots of the player into the goals and the non-goals.

4.6 Match Momentum

Thanks to ScraperFC library we get functions such as ‘sofascore.match_momentum(match_url)’, these give us a DataFrame with the minute and the value of the momentum, if the momentum is more than 0, the value is for the home team but it the value is less than 0, it means the momentum is on the away team.

4.7 Players and Teams shots plotting

Players

We created a function named ‘playerShots’ that requires the season, first, we created a vertical pitch using the mplsoccer library function ‘VerticalPitch()’, we use a custom pitch type, with 105 of length and 68 of width. First, we use ‘.scatter()’ to plot the goal and the non-goal shots, with an if statement we check if there were shots in both df’s. Then using the request function, we get the image of the player, we predefined the URL above and the only thing changing is the player ID. Then we save the file, and the function returns the file of the figure generated.

Teams

For the teams we have the functions ‘homeShots’ and ‘awayShots’, the only things that change are the DataFrames plotted, and the text used. As well that in the player function, we use the same custom pitch and the scatter function with the else statement. In this case, the URL for the team is like the player one but with the team ID. In this case, we are going to give 3 extra statistics, the xG, the goals, and the shots of the team. Returns the file as well.

The season is required because we used it when we want to save the file. In Fotmob all the links of player's image are the same, the only things changing is ID of the player or the team.

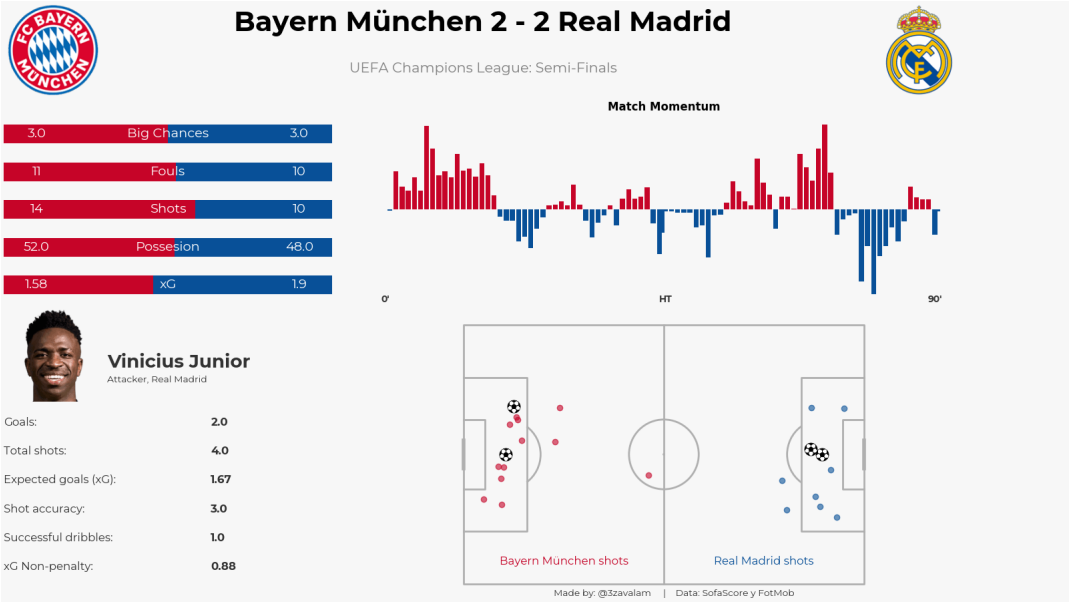
4.8 Heatmap

Sofascore have a function 'sofascore.get_player_heatmap()' that give us a DataFrame with the x and y for each minute. To plot the heatmap, we first declare the pitch using the mplsoccer function 'Pitch()' this pitch type we are going to use is the opta one. Using the hot colormap, the player's movements are graphically represented on the pitch using a kernel density estimation plot. This plot overlays the pitch, with shading applied to indicate varying levels of activity intensity. By providing a concise and visually intuitive representation, this heatmap offers values insights into the player's positioning and involvement throughout the match.

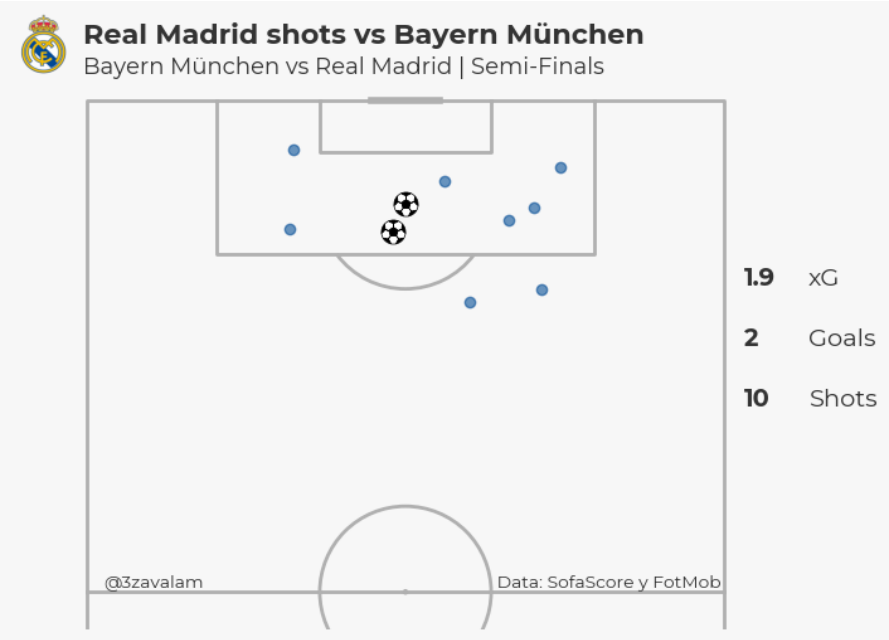
4.9 Dashboard

After creating the figure, we divided in 8 subplots 'fig.add_subplot()'. At the top of the dashboard, we plot the game result, teams, names, etc. Then we have a subplot for the stats of both teams, match momentum, shotmap of both teams and stats of the POM.

5. Results



Img 1. Dashboard of Bayern Munich vs Real Madrid. Champions League 23/24 Semifinals



Img 2. Shotmap of Real Madrid vs Bayern Munich. Champions League 23/24 Semifinals

Img 3. Vinicius Junior Shotmap vs Bayern Munich. Champions League 23/24 Semifinals

6. Discussion

The data used for these dashboards and shotmaps are thanks to SofaScore and Fotmob. They both get their data via opta, this is one of the best companies for data in the football industry. The methodology used is effective thanks to the possibility of changing the display in the dashboards or the statistics used. The main limitations are in the 'ScraperFC' package, Sofascore makes constant changes to their API, these cause some problems if the library isn't updated. There where instance where the API modify and only gives positive values in the match momentum DataFrame.
These dashboards are aim to the public, which want to know a bit more about the match but not going to deep, for a after match analysis is perfect for the public. But it has a lack of context and analysis, that why you need to go deeper if you want to make an analysis of the match for a coach or players. This may be an opportunity for next projects, add more specific analysis and statistics to make a more focused analysis of the game to give to the staff members of a team.

7. Conclusion

Making the API call in the Fotmob web was the more challenging part of the project, ones I have the data in a variable it was just matter of trying the different keys of the dictionary to get the data I want. For the SofaScore web was easier thanks to the library. Is interesting know how the webs works and the scraping part, the web scraping have the ethical dilemma, but we are not going to talk about that because we can write another paper about that.
The project was firstly aimed to only making the dashboard, but seeing all the data gathered only one dashboard seems like wasting data. The main objective for the outputs were purely informatic, for a public who likes football and want to have a little bit more information about the match they just saw or to know how a match ended without seeing it.

8. References

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95.

McKinney, W., & others. (2010). Data structures for statistical computing in python.
In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).

Rowlinson, A. (2024). Mplsoccer: Matpllitlib Soccer.
https://github.com/andrewRowlinson/mplsoccer

Full code implementation available on GitHub: https://github.com/3zavalam/Project-1-

9. Annotations

1. https://www.fotmob.com/en
2. https://www.sofascore.com/
3. POM: Player Of the Match
4. https://github.com/oseymour