From your current rank and evolution, get to know how well it will end for you

My journey from data scraping to building an app

Introduction

Measures implemented to contain Covid-19 outbreaks impacted all aspects of life. In particular, soccer championships had been forced to pause their season. In France, it meant a final stop after 27 full legs and positions were frozen. The decision was met with outrage by clubs that were not in positions they might have been had the season run to its normal end because it had huge consequences for them; from a sporting point of view to a financial one.
As a soccer fan and a data scientist, I saw a great opportunity to combine both fields in one new original (at least for me!) project : predicting the final ranking given the course of the championship is known up to some leg.
The project is decomposed in 5 steps :

- getting the data
- exploring it and get some insights
- build different type of models and pick the most interesting
- moving the workflow from notebooks to Python scripts
- writing an app with Streamlit.

Each of these steps will be described in their respective sections. The code (notebooks and Python scripts) is available in my Github repository here.

Getting the data

First and foremost, in order to get started with the project, we are going to need data; the more the better !

Since I wanted to take advantage of this project to experiment with tools I usually don't use in my daily work, I decided to build my datasets by scraping it. After some online research I ended up on the website "*L'Équipe*", a french newspaper specialized in sports. There I found what I needed : all soccer results from season 2004-2005 to season 2018-2019 for the following championship:

- Ligue-1 : France's top soccer league
- Ligue-2 : France's second soccer league
- Premier League : England's top soccer league
- Bundesliga : Germany's one
- Serie-A : Italy's one
- La Liga : Spain's one

Actually, there are results from older seasons but they were not relevant due to some structural changes the different leagues had implemented, e.g. going from 18 to 20 teams or

reforming the amount of points won after a game which is currently 3 for a win, 1 for a toss and 0 for a loss.

In order to achieve our first goal, we will need the following imports :

```
from bs4 import BeautifulSoup : to process the GET response from requests
import requests : to get data from the website
import pandas as pd : to build, handle and save the scraped data as dataframes
import re : to extract the relevant piece of information (team name and the number of goals scored)
import random : combined with time, it is used to provide a random number of seconds to wait between 2 requests calls
import time : to monitor the amount of time taken to process one step
from copy import deepcopy : to handle safely dataframes, especially when transformation (e.g. slicing, renaming) is performed while keeping the original dataframe unchanged
import numpy as np : to manipulate pieces of data as arrays.
```

Scraping data from the website requires knowing the template of the urls. For the french leagues it is
'https://www.lequipe.fr/Football/ligue-{level}/saison-{season}/page-calendrier-resultats/{leg}-journee'
whereas for the other leagues it is
'https://www.lequipe.fr/Football/championnat-d-{country}/saison-{season}/page-calendrier-resultats/{leg}-journee'
where country is the French lower cased name of the corresponding country : Germany -> allemagne, Italy -> italie, England -> angleterre and Spain -> espagne.

Given a championship, a season and a leg, our goal is to get from the following screenshot (webpage can be found here[1])

[1] https://www.lequipe.fr/Football/ligue-1/saison-2017-2018/page-calendrier-resultats/36e-journee

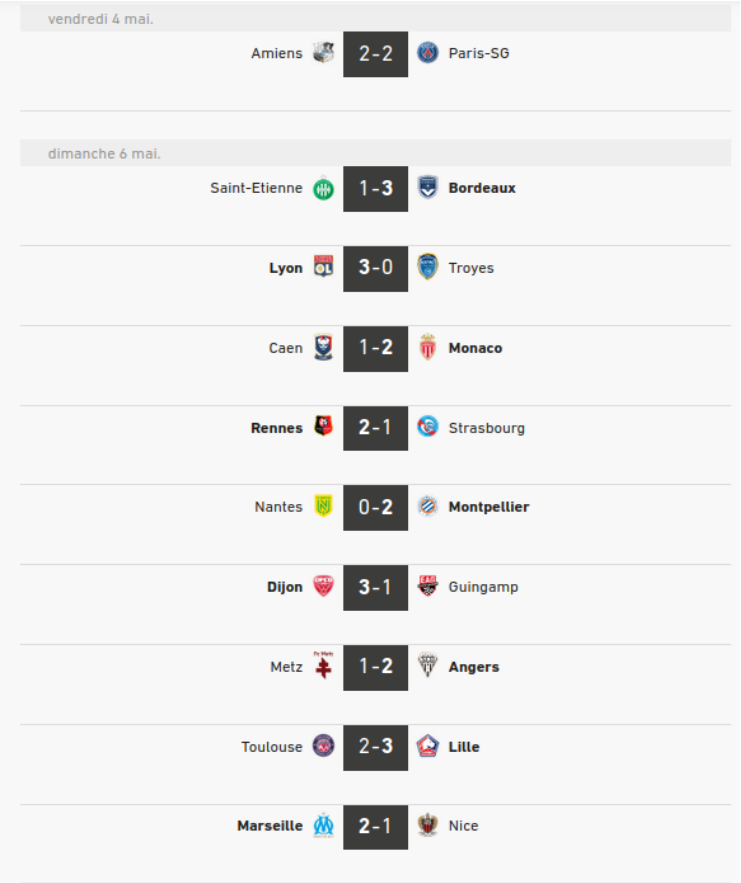

Fig. 1 : Data as displayed by l'Équipe in its website. Screenshot by author.

to the following dataframe

	country	season	leg	team	play	goals_scored	opponent	goals_conceded	nb_points
0	France	2017-2018	36	Amiens	Home	2	Paris-SG	2	1
1	France	2017-2018	36	Paris-SG	Away	2	Amiens	2	1
2	France	2017-2018	36	Saint-Étienne	Home	1	Bordeaux	3	0
3	France	2017-2018	36	Bordeaux	Away	3	Saint-Étienne	1	3
4	France	2017-2018	36	Lyon	Home	3	Troyes	0	3
5	France	2017-2018	36	Troyes	Away	0	Lyon	3	0
6	France	2017-2018	36	Caen	Home	1	Monaco	2	0
7	France	2017-2018	36	Monaco	Away	2	Caen	1	3
8	France	2017-2018	36	Rennes	Home	2	Strasbourg	1	3
9	France	2017-2018	36	Strasbourg	Away	1	Rennes	2	0
10	France	2017-2018	36	Nantes	Home	0	Montpellier	2	0
11	France	2017-2018	36	Montpellier	Away	2	Nantes	0	3
12	France	2017-2018	36	Dijon	Home	3	Guingamp	1	3
13	France	2017-2018	36	Guingamp	Away	1	Dijon	3	0
14	France	2017-2018	36	Metz	Home	1	Angers	2	0
15	France	2017-2018	36	Angers	Away	2	Metz	1	3
16	France	2017-2018	36	Toulouse	Home	2	Lille	3	0
17	France	2017-2018	36	Lille	Away	3	Toulouse	2	3
18	France	2017-2018	36	Marseille	Home	2	Nice	1	3
19	France	2017-2018	36	Nice	Away	1	Marseille	2	0

Fig. 2 : Data after being scrapped. Screenshot by author.

This is done by the function `get_leg_data` that can be found in the notebook `notebooks/scrap_and_build_data.ipynb`

Once the process is operational, we automate the scraping in 3 steps:

1. get all data for 1 championship and 1 season
2. get all data for 1 championship over the 15 seasons
3. get all data for all championships

At the end of the day, our data is made available in 5 csv files : 1 per championship. They are located in the folder `inputs`.

For more information on how to scrap a website, one can have a look here[2] .

At this point, we are ready to move on with the project. I would like to express my gratitude to “L’Équipe” for the historical records and for not blocking me from getting the data automatically.

Exploratory Data Analysis

Now that we have our data spanning from season 2004-2005 to 2018-2019; that is 15 seasons, let us analyze it in order to get insights on how teams perform.

In order to get a feeling about how teams are performing we start with some global insights such that :

- The number of teams having played at least 1 season (out of the 15 we are investigating).
- The number of teams having played all 15 seasons.
- The number of champions.
- The maximum number of titles won by a team.

They will provide hints about how diversified a championship is. Running our app for all the championships we’ve collected data for yields :

Championship	Nb. of participants (played min 1 season)	Nb. of teams having played all seasons	Nb. of distinct champions	Maximum number of titles won by 1 team
Ligue-1 (FR)	41	9	7	6 (PSG)
La Liga (ESP)	41	7	3	10 (FC Barcelona)
Serie-A (IT)	38	6	3	10 (Juventus Turin)
Bundesliga (GER)	34	6	4	11 (Bayern Munich)

[2] https://realpython.com/beautiful-soup-web-scraper-python/

Premier League (ENG)	39	7	4	5 (Chelsea, Man. Utd)
Ligue-2 (FR)	48	0	12	3 (Metz)

It is also worth mentioning that Man. City won the Premier League 4 times out of the 15 considered seasons.

All those informations can be retrieved from the Streamlit app under the EDA section as shown below :

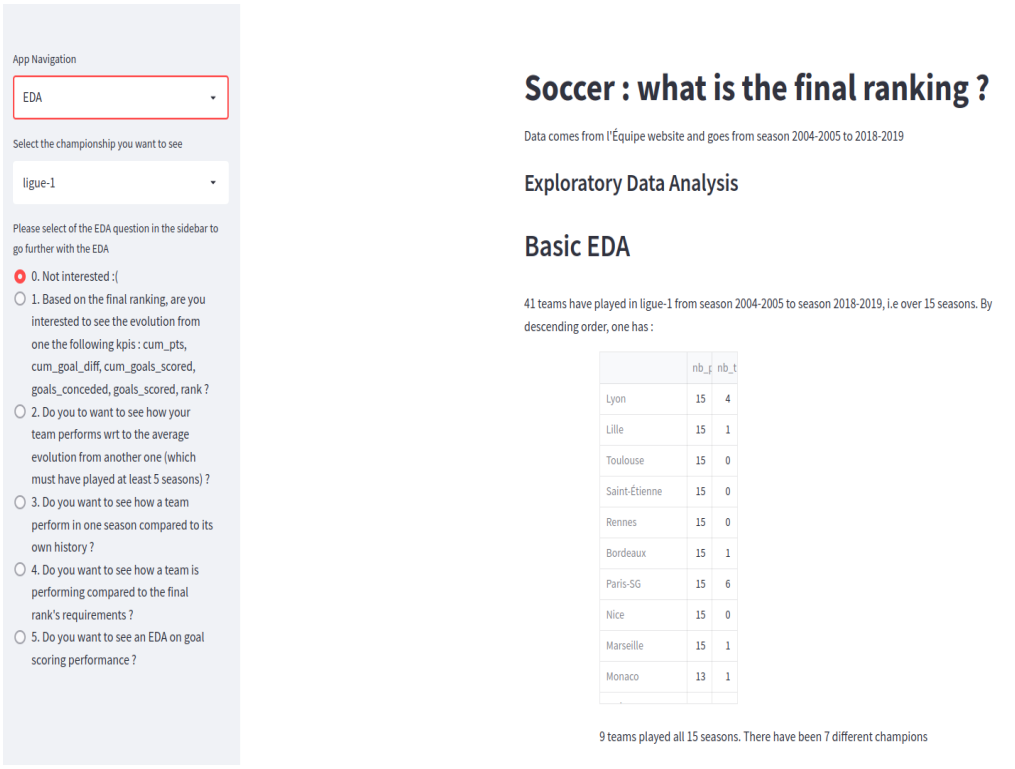

Fig. 3 : Default display from the EDA page from the Streamlit app. Screenshot by author.

Next, let see if we can exhibit trends about the evolution of the following kpis based upon the final rank since there are used to determine the ranking :

- **Points** : this is a main criteria on how to rank the teams; the more, the higher.
- **Goal difference:** it is the difference between the goals scored and the goals conceded. When several teams have the same, then the goal difference is used to decide which one ranks highest. The greatest the difference is the best it is.
- **Goals Scored:** If the two first criterias above turned out to be insufficient to decide the ranking, then the most attacking team goes first … After all, the show must go on :)

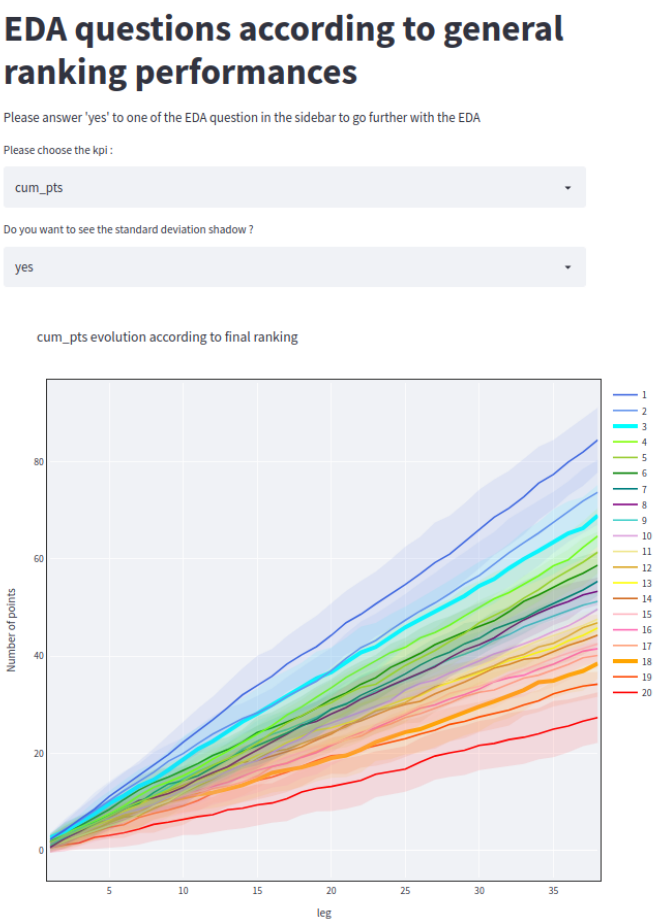

Fig. 4 : Cumulative point evolution's trends for Ligue-1. Standard deviation shadows show how confident one could be when following a trend. Screenshot by author.

Interestingly, and actually quite surprising, is that the top3 and the flop1 trends are isolating themselves from the remaining one early in the season (around legs 9 - 10). Concerning the trend for rank 19, it is around leg 25.

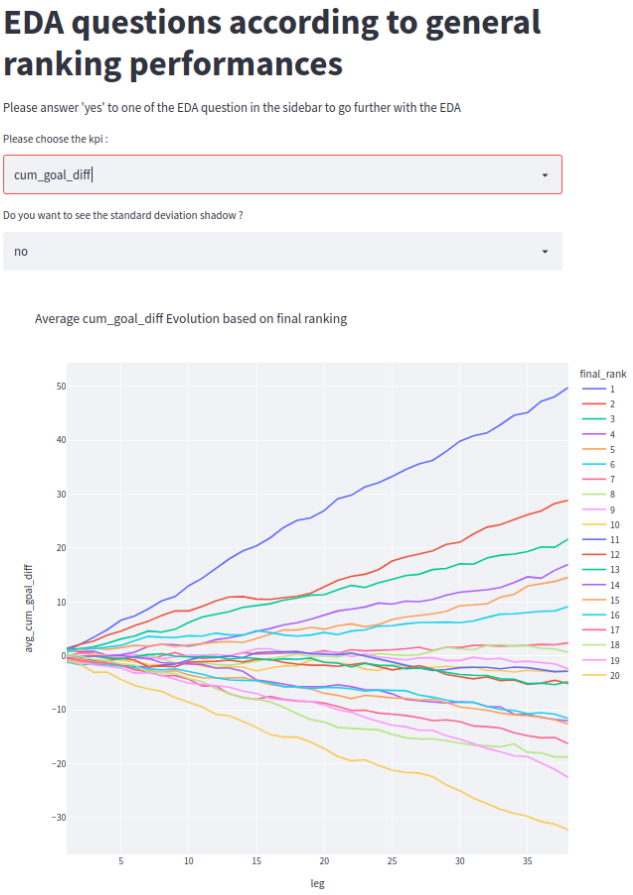

Fig. 5 : Cumulative goal difference evolution's trends for Ligue-1. Screenshot by author.

When talking about the goal difference, top1 and flop1 are visible since leg 5 and are keeping the pace until the end. On average, only the top 8 end the season with a positive difference. An interesting point I had not expected is that one could identify 3 groups at the mid-ranking level where trends end up being almost equal :

- group 1 : ranks 9, 10 and 11 at around -2.6,
- group 2 : ranks 12 and 13 at around -5. Note that rank 13 has a better goal difference than rank 12.
- group 3 : ranks 14-15-16 at around -12.

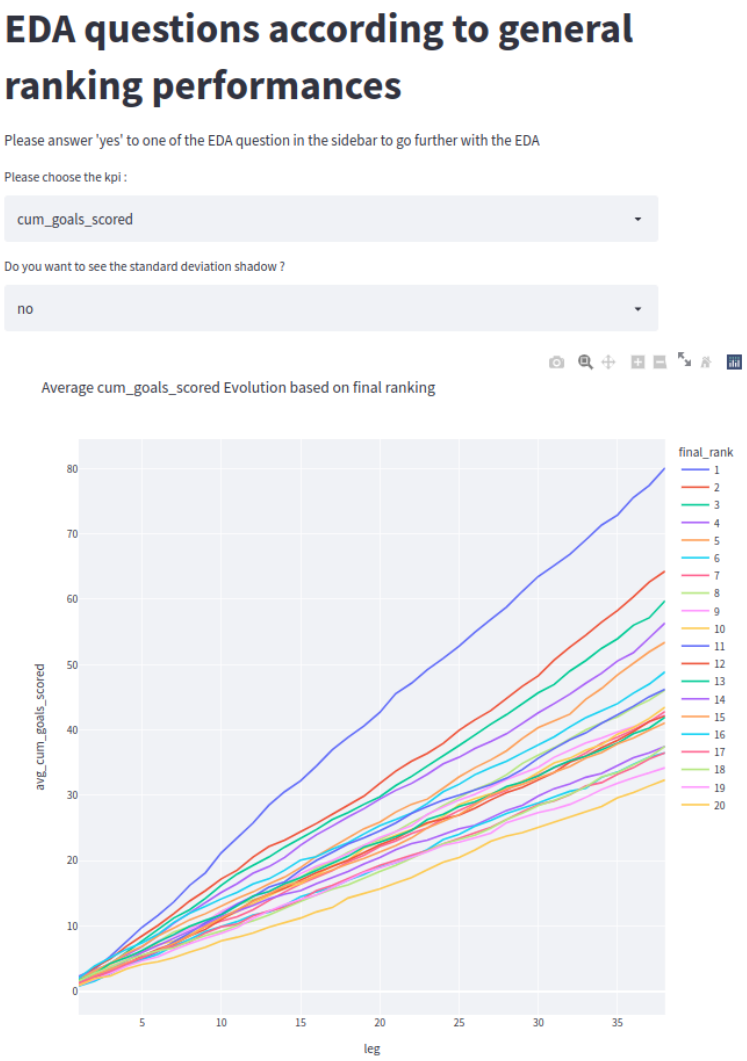

Fig. 6 : Cumulative goal scored evolution's trends for Ligue-1. Screenshot by author.

The interesting insight is that along the entire season, goal scored trends from ranks 10 to 15 are almost equal, with an average number of goals slightly higher than 1 while ranks 16 to 18 are around 1 goal per game while top5 scores on average 1.5 goals per game. Again top1 and flop1 are kind of 'outliers' since leg 10.

Trends are great since they provide valuable lessons on the expected pace to progress to end at a specific rank (think to the Law of Large Numbers). However, randomness in the performance is the spicy element in sport competitions; could one guess all results, then there wouldn't have been any interest in it. So, what if we transpose in one plot the tendencies for a championship together with the evolution of a team for a specific season ? Let's find out

Fig. 7 : Auxerre's point evolution in 2009-2010 compared to the rank trends. Left, without the standard deviation shadow. Right, with the shadow. Screenshot by author.

Figure 7 is interesting because it proves that trends are not predicting everything. If one were just considering the left part, it would have been a natural guess to say that Auxerre should have ranked at position 2. However, when looking at the right side, then we would notice that the rank 2 trend lies within the standard deviation shadow range from the rank 3 trend.

Now that we have a visualization of the performance evolution for a specific team and season with respect to final rank tendencies, let us do (just for the pleasure of exploring) the same thing but with respect to another team's (or even its own) average performance.

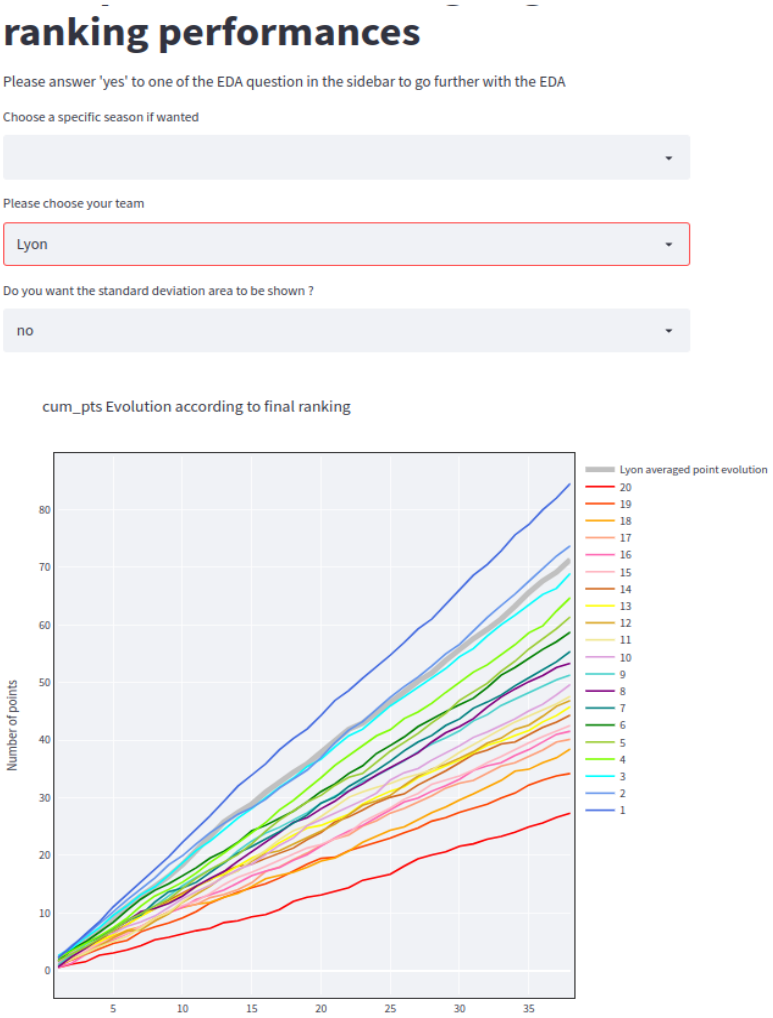

Fig. 8 : Lyon's average point evolution. It is computed over the 15 seasons Lyon played in Ligue-1 between 2004-2005 and 2018-2019. Screenshot by author.

Over the 15 Ligue-1 seasons considered in this project, figure 8 shows that Lyon is a strong team and should rank at position 2 or 3. Even if Paris-SG is almost always champion since the last decade, it is worth reminding Lyon won the title 7 times in a row (from season 2001-2002 to 2007-2008) during the 2000s.

EDA for 2019-2020 season

Since our season of interest is the 2019-2020, we also scrapped data related to it. However it is used apart from all the others since it is incomplete under a Covid free environment setting.

So … Are you ready to explore the 2019-2020 season and get visual insights to guess the final ranking for a couple of teams ?

I propose you to explore :

- **Ligue-1** : Amiens
- **Ligue-2** : Clermont

- **Premier-League** : Man. City and Liverpool
- **Serie-A** : AS Roma
- **Bundesliga** : Eintracht Frankfurt
- **La Liga** : Atlético Madrid

Contrary to the other leagues, the French ones (Ligue-1 and Ligue-2) were not resumed after the lockdown from Spring 2020. Hence, whatever your guess is, it will be open to debate :).

First of all, let us remind what the rankings were when the season was stopped after leg 27 (which is the last leg being completed for Ligue-1) :

- **Amiens** : it was standing at rank 19 with 22 points.
- **Clermont** : it was standing at rank 5 with 47 points.
- **Man. City** : it was 2nd with 57 points.
- **Liverpool** : it was leading the Premier League with 79 points.
- **AS Roma** : it was standing at rank 5 with 48 points.
- **Eintracht Frankfurt** : it was standing at rank 12 with 31 points.
- **Atlético Madrid** : it was standing at rank 6 with 45 points.

Secondly, let us see how teams performed during the season till mid-march 2020 in term of points when comparing it to final rank points average historical evolution:

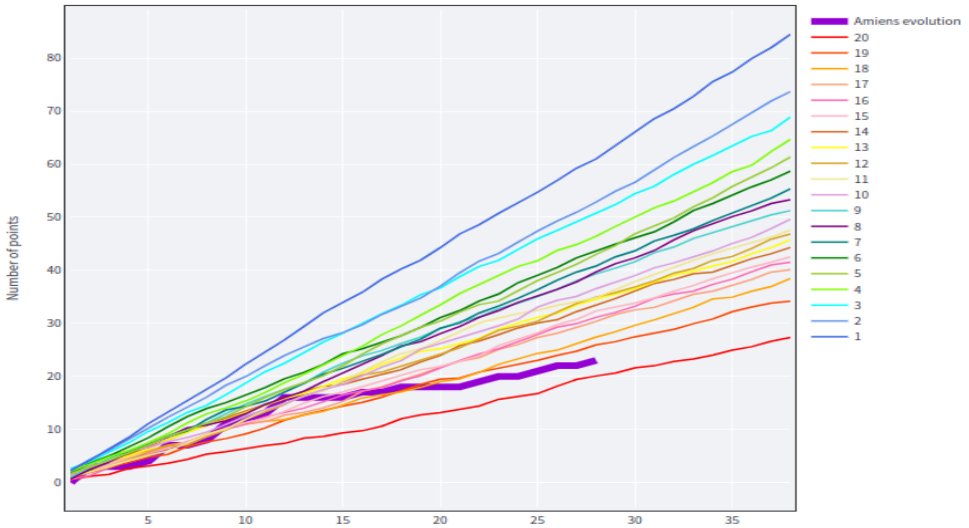

Fig.9.A : Amiens' point evolution during the 2019-2020 season with respect to Ligue-1 final rank points average historical evolution. Screenshot by author.

Amiens' points evolution shows that its season has 2 parts. The first one goes until leg 12 where it had a pretty decent pace which would give expectation to end at a mid-level standing. However the second part is catastrophic simply put. Indeed, if it had continued like that until the end, it would not have been a surprise to see them ending last. This weird evolution suggests something happened. Let see how it looks like on goal difference

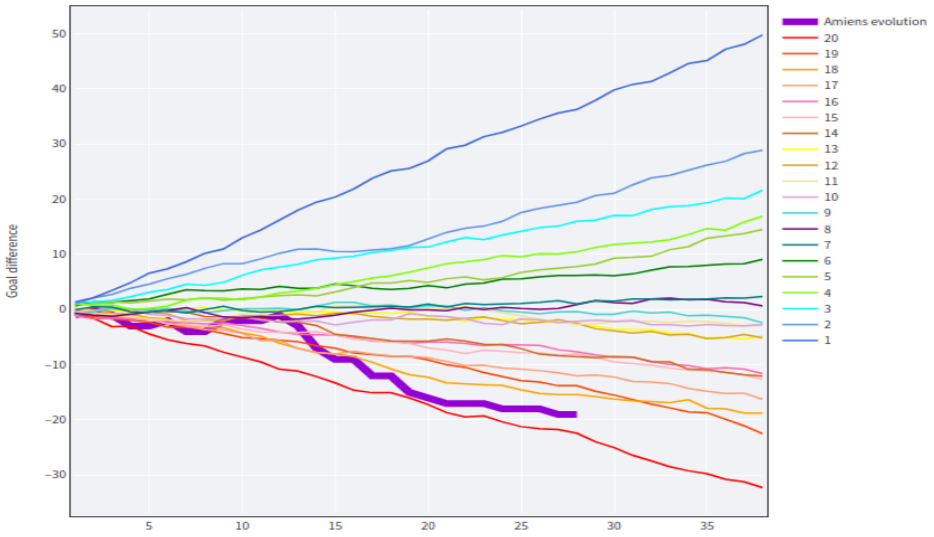

Fig.9.B : Amiens' goal difference evolution during the 2019-2020 season with respect to Ligue-1 final rank points average historical evolution. Screenshot by author.

Interestingly, the same pattern occurs with the goal difference. At this point, an imbalance is developing between goals scored and goals conceded. So let's dive in further.

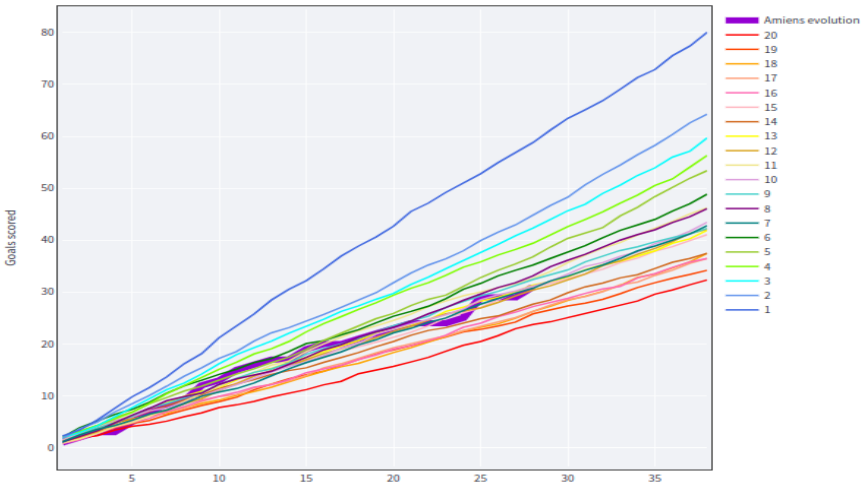

Fig.9.B : Amiens' goals scored evolution during the 2019-2020 season with respect to Ligue-1 final rank points average historical evolution. Screenshot by author

Despite the poor results obtained by Amiens since leg 13, it still seemed able to score as many goals as in their first part. This teaches us that Amiens faced important issues in their defensive sector and that it needed to be solved before the next season in order to live the situation again. The second lesson to be learnt is that soccer requires to be strong in defense in order to expect good results.

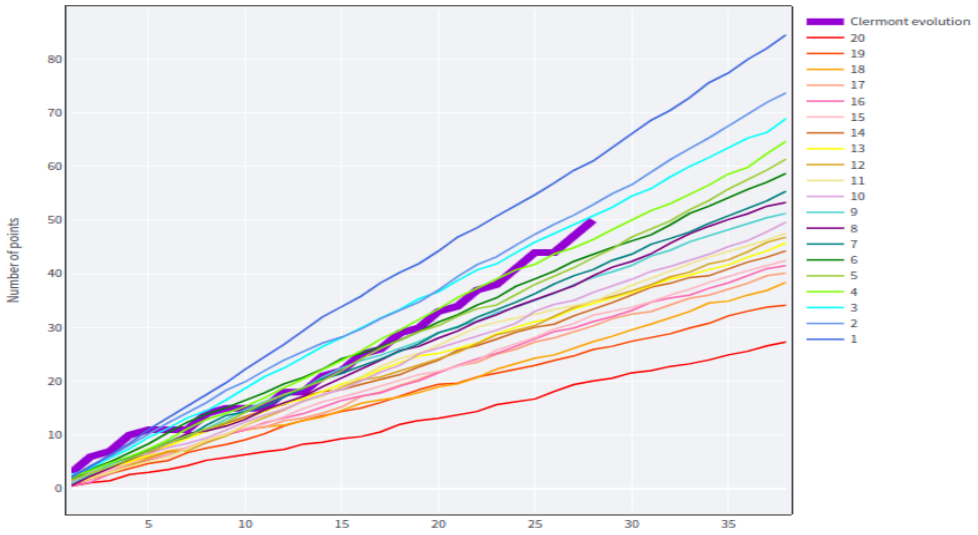

Fig.10 : Clermont's point evolution during the 2019-2020 season with respect to Ligue-2 final rank points average historical evolution. Screenshot by author.

Clermont's evolution showed it was gaining steam in their ability to win points and, according to the historical evolution, it could expect to be 3rd at the time the season was stopped. The fact that Clermont was standing at the 5th position suggests a competitive Ligue-2 season. Due to its dynamic, I think Clermont could have made it to be promoted to Ligue-1 by the end of that season. Unfortunately for them it was decided otherwise. However, their goal was reached the next season (2nd at the end of season 2020-2021).

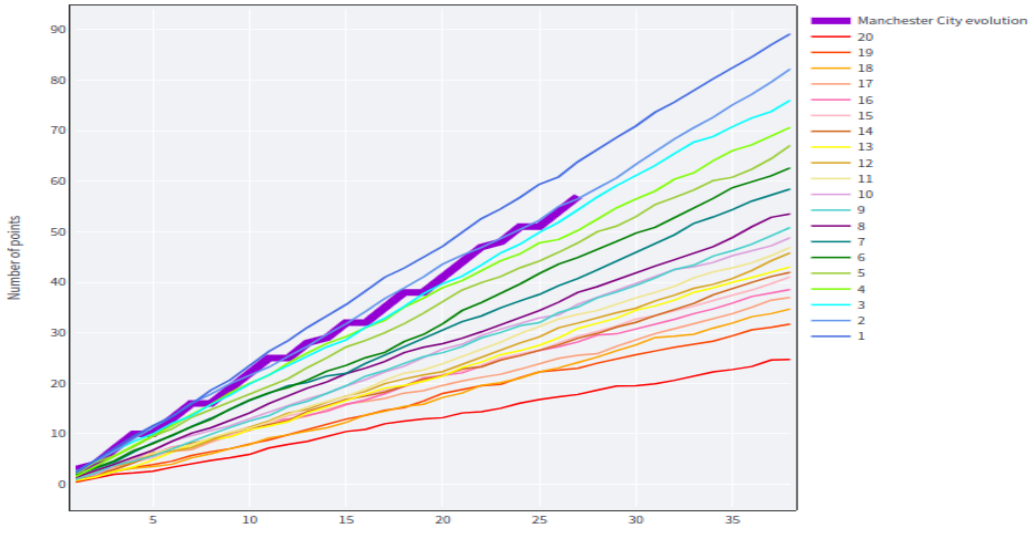

Fig.11 : Man. City's point evolution during the 2019-2020 season with respect to Premier-League final rank points average historical evolution. Screenshot by author.

During the 2019-2020 season, Manchester City is quite impressive in its ability to match very well with the average historical pace to finish second. Therefore, and because of Liverpool's performance that year (see Fig. 12), I would have bet that Man. City completes the season at rank 2.

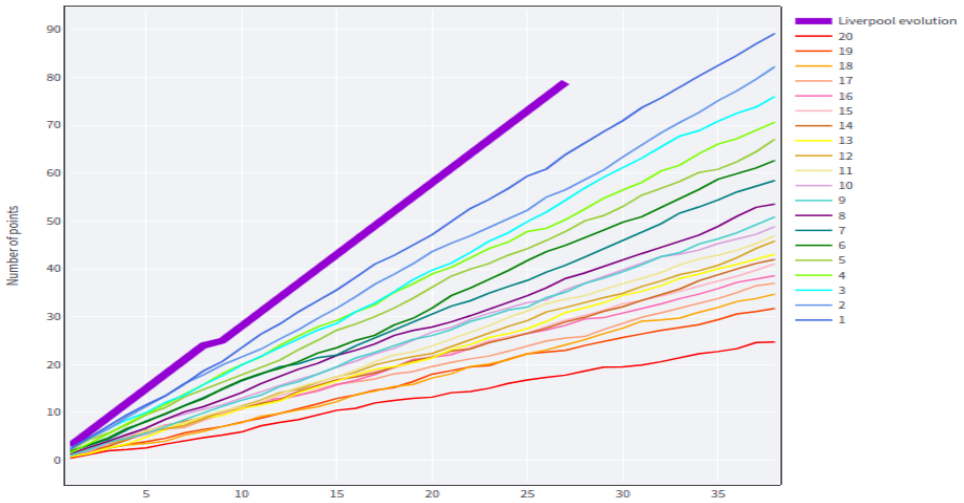

Fig.12 : Liverpool's point evolution during the 2019-2020 season with respect to Premier-League final rank points average historical evolution. Screenshot by author.

I mentioned I found Man. City to be impressive. But what can we say about Liverpool ? It is winning almost every game and is moving much faster than the expected pace to finish first. Therefore, no surprise to say Liverpool would be champion, even at this stage of the season.

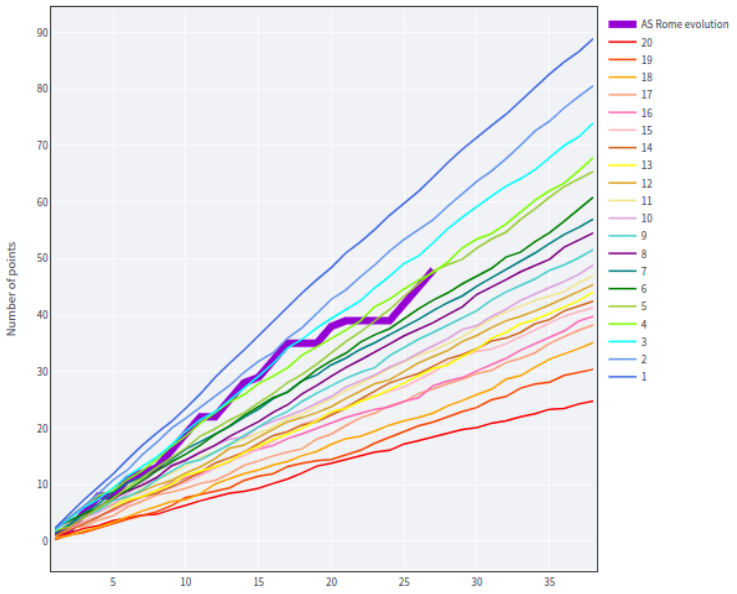

Fig.13 : AS Roma's point evolution during the 2019-2020 season with respect to Serie-A final rank points average historical evolution. Screenshot by author.

AS Roma had a very points winning pace until leg 16 before having a more complicated period that lasted around 8 games. Just before Serie-A was stopped, AS Roma was again on track of winning (in average) a high number of points per game. In case it is able to retrieve this pace when the championship resumes, then one could expect AS Roma to reach a position between 2 and 4.

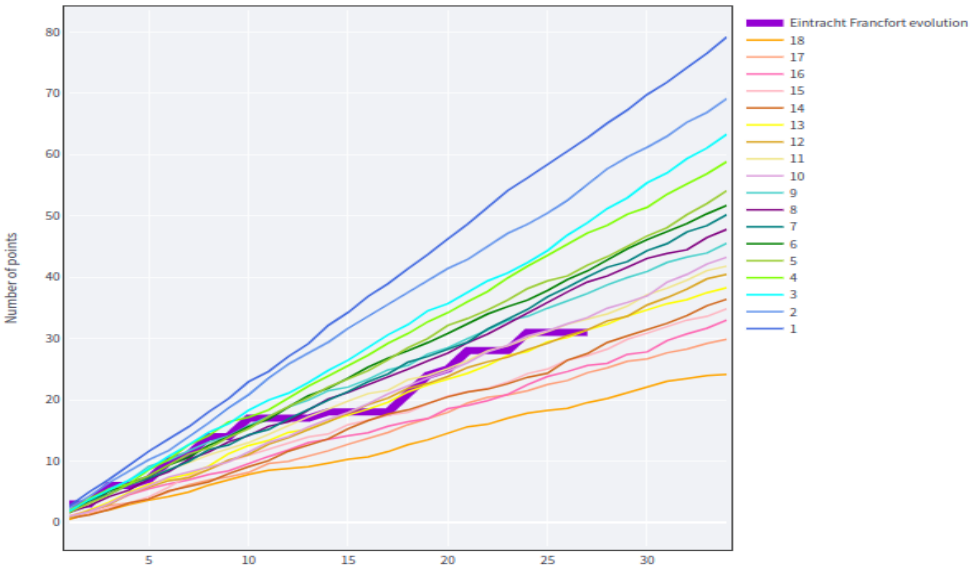

Fig.14.A : Eintracht Frankfurt's point evolution during the 2019-2020 season with respect to Bundesliga final rank points average historical evolution. Screenshot by author.

Depending on the reader, one could identify 3 or 4 distinct phases in the Eintracht Frankfurt season : the first from leg 1 to 10, the second from leg 11 to 17, the third from leg 18 to 25. I would say it is open to debate whether legs 26 and 27 still belong to the third part or are the beginning of the fourth one. Looking at the goal difference and goals scored evolution from figures 14.B and 14.C, I would opt for the second option : there is a 4th part.

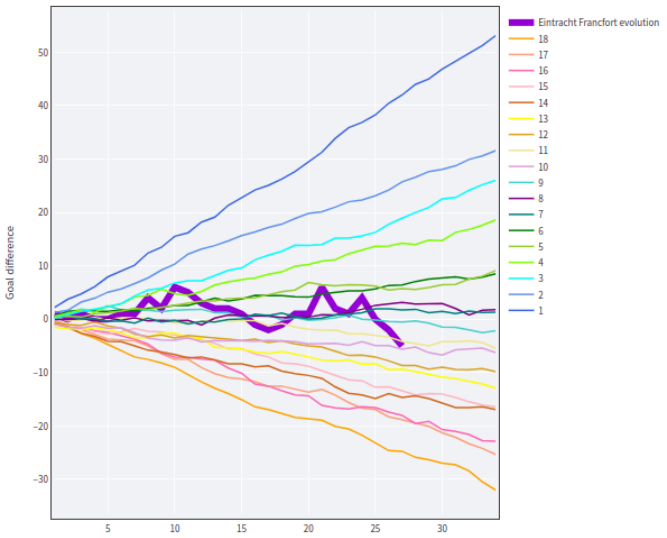

Fig.14.B : Eintracht Frankfurt's goal difference evolution during the 2019-2020 season with respect to Bundesliga final rank points average historical evolution. Screenshot by author.

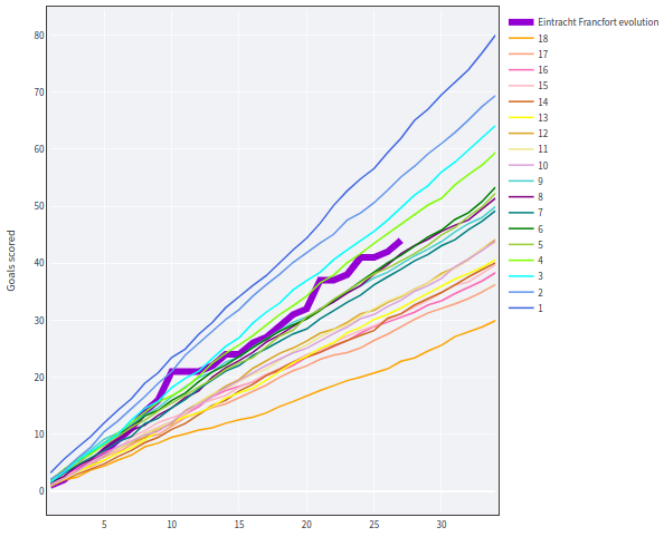

Fig.14.C : Eintracht Frankfurt's goals scored evolution during the 2019-2020 season with respect to Bundesliga final rank points average historical evolution. Screenshot by author.

If phase 4 lasts until the end of the season and looks like phase 2, I would say Eintracht Frankfurt would finish between ranks 15 and 17. On the other hand, if it gets to a fifth phase looking like the third one, I would bet for a rank between 10 and 13.

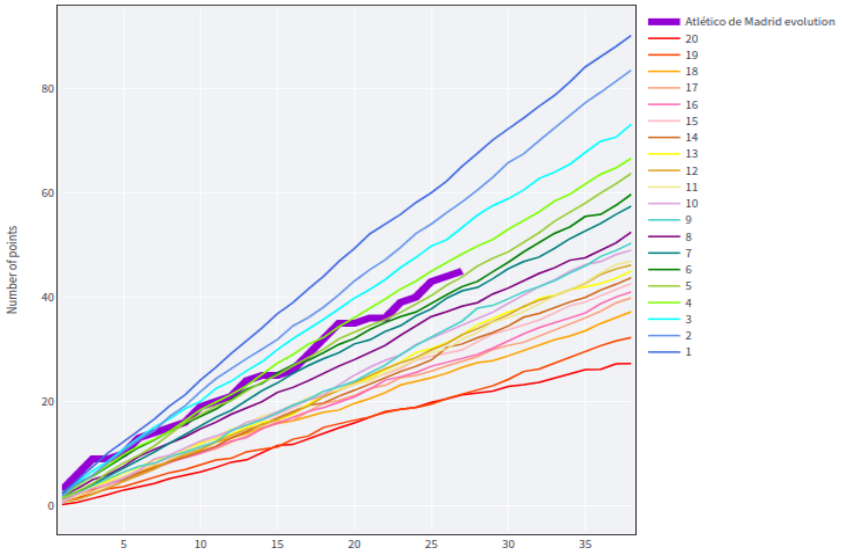

Fig.15 : Atlético Madrid's point evolution during the 2019-2020 season with respect to La Liga final rank points average historical evolution. Screenshot by author.

Despite being 6th after 27 games, Atlético Madrid follows pretty well the curve corresponding to the 5th final standing. If it keeps the same pace, Atlético could expect to finish between ranks 4 and 7.

Now, let us find out if our guesses were good or not. Contrary to the other leagues, the French ones (Ligue-1 and Ligue-2) were not resumed after the lockdown from Spring 2020. Hence, whatever your guess is, it might still be open to debate :). What about the other leagues ? Well, let's have a look at the records :

- Manchester City ended the season as vice-champion with 81 points . As shown in Fig. 15, it was able to keep its pre-covid pace.

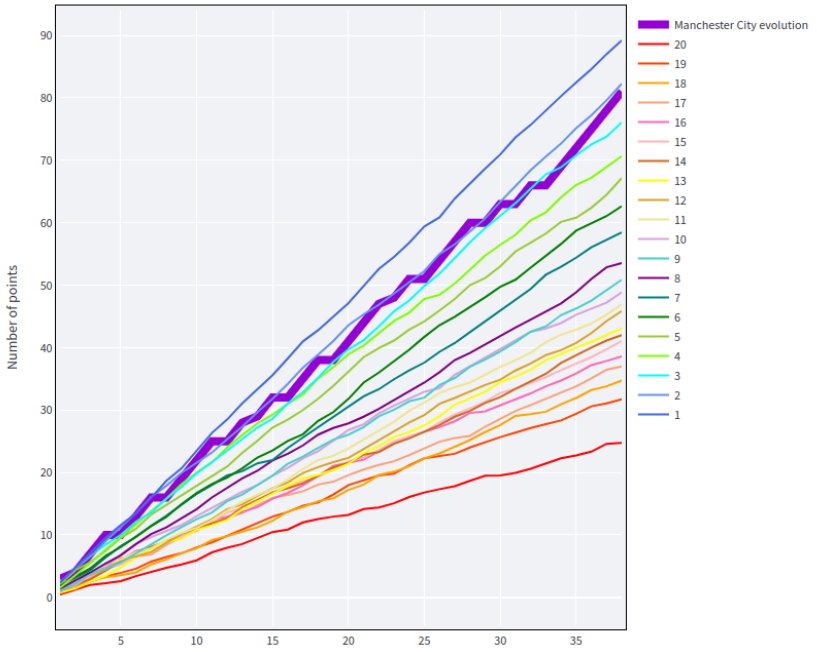

Fig. 16 : Man. City complete points evolution. Screenshot by author.

- Liverpool ended the season as champion with 99 points. However, it looks like it was negatively impacted by the break following the Covid outbreak.

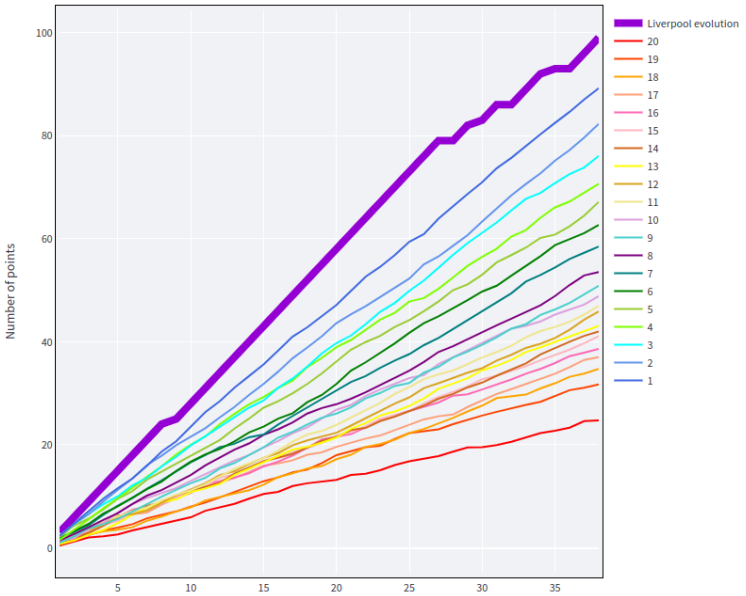

Fig. 17 : Liverpool complete points evolution. Screenshot by author.

- AS Roma completed Serie A at the 5th position with 70 points. Despite being able to almost (it had a small losing streak) continue on the same pace as shortly before Serie-A interruption and ending between the curves for an expected final position 3 and 6. This also shows that the competition level that year should have been quite high.

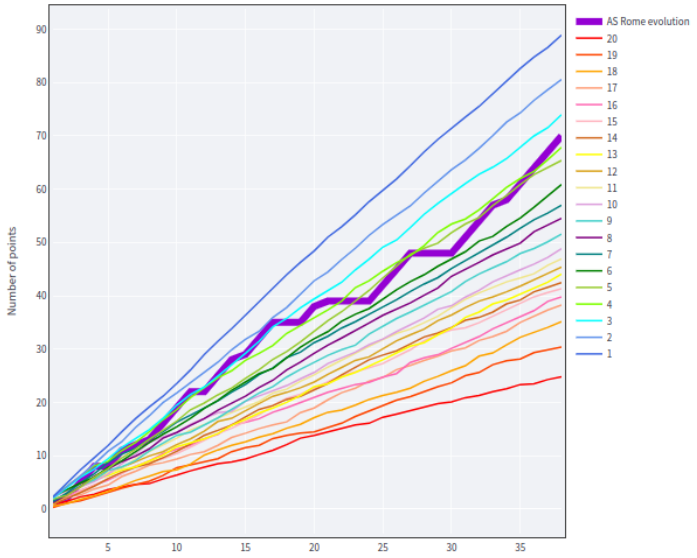

Fig. 18 : AS Roma complete points evolution. Screenshot by author.

- Eintracht Frankfurt ended the season at rank 9 with 45 points. Finally, the restart from Bundesliga marked the start of a fifth phase in Eintracht Frankfurt's ability to win games. It has even gotten a slightly better result as our most optimistic guess.

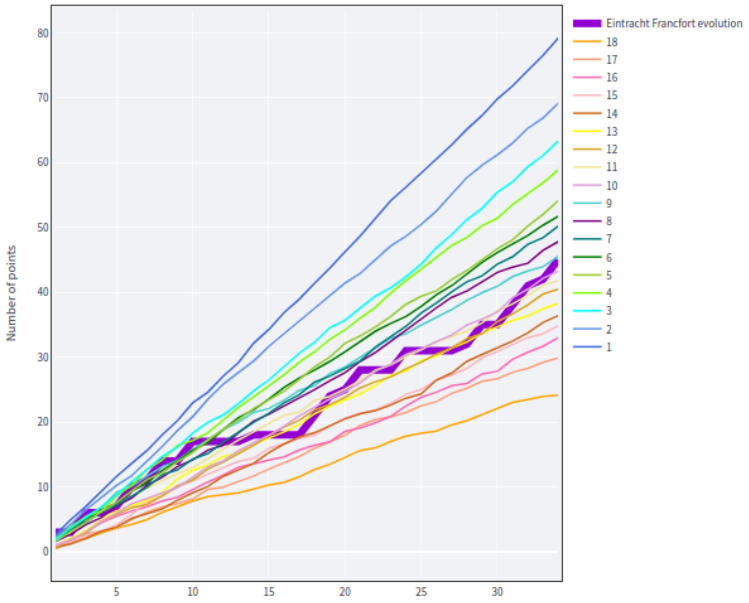

Fig. 19 : Eintracht Frankfurt complete points evolution. Screenshot by author.

- Atlético Madrid ended the season 3rd with 70 points. The forced covid break turned out to be very beneficial to Atlético Madrid since it was able to improve the average number points won per game. As a consequence, it was able to get from the curve corresponding to an expected rank 6 to the expected rank 4 curve. It has to be noted that the actual fourth (Sevilla) also ended with 70 points, but Atlético Madrid had a slightly better goal difference (+24 vs +20).

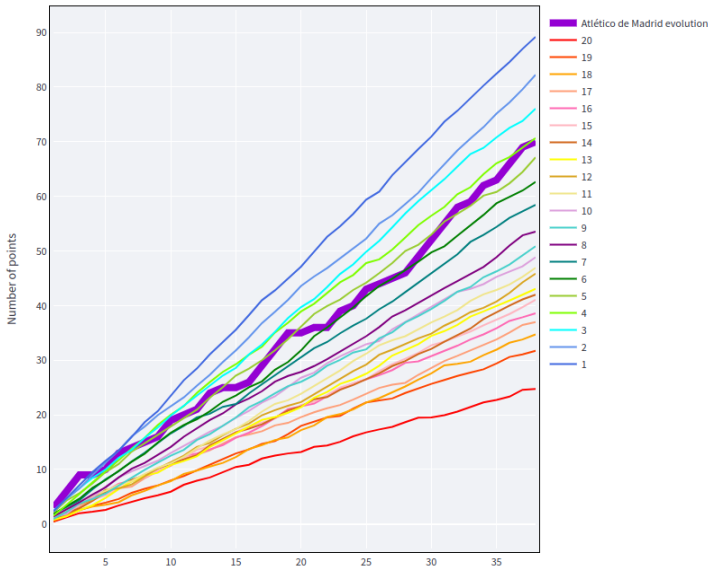

Fig. 20 : Atlético Madrid complete points evolution. Screenshot by author.

Modeling

The second part of this project is to predict a team's final rank given how well it performed up to some leg. Since our motivation is to predict the Ligue-1 final ranking for the 2019-2020 season which was interrupted after 27 complete legs[3], we are assuming that the feature set is composed of how teams performed up to leg 27.

The naive one

This model is the baseline and was chosen by the LFP to determine the final rank. It is very simple since it relies only on the average number of points scored per leg. The formula is

$$final\ number\ of\ points\ =\ championship\ length\ \times\ average\ number\ of\ points\ per\ leg.$$

For Ligue-1, $championship\ length\ =\ 38$ and since the season was stopped after leg 27, $average\ number\ of\ points\ per\ leg\ =\ \frac{total\ nb\ of\ pts\ after\ 27\ games}{27}$.

Despite (or shall I say due to) its simplicity, there is one main flaw that our project tries to address : Dynamics in a season are not taken into account. Indeed, it happens that some teams perform better during the season's 'money time' (let's say the 7-8 last games) than the rest of the season.

So … let us build ML models to solve this issue. We are going to explore 3 approaches : regression, classification and 'traditional' ranking.

[3] Actually, leg 28 was almost completed when the championship was stopped. Therefore, the modeling choice was made to consider 27 as the breaking leg.

Regression type

In soccer, ranking is based on a descending order by points scored. So, if we can predict points for all the competitors, we have our ranking.

As such, the **ultimate target variable** for the models from this family **is the final number of points** a team is going to score by the end of the championship.

It is worth pointing out that the predicted final number of points will almost surely not be an integer value; hence we shall not expect to get *exactly* the value predicted since at each leg a team can win 0, 1 or 3 points. However, that is not an issue because we want to rank. Actually, having non-integer values limits the risk of getting a draw between 2+ teams.

During the prototyping phase I experimented with several combinations of feature engineering and different models summarized in the table below (explanations of the variables to be given shortly after) :

Model Name	Model Type	Features	Target
lr_1	Linear Regression	- nb. games to play at home - nb. pts won at break leg	final nb pts
lr_2	Linear Regression	- lr. feat coeff	final nb pts
lr_3	Linear Regression	- lr_feat_coef - nb. pts won at break leg - cumulative goal difference at break leg - rolling 5 avg pts at break leg - nb. games to play at home	final nb pts
lr_4 (*)	Linear Regression	- rolling 5 avg pts at break leg - avg. nb. pts since season start - cumulative goal difference - cumulative goals scored - the cumulative points evolution from start to break leg	final nb pts
lr_5 (*)	Linear Regression	- rolling 5 pts at break leg - avg. cumulative pts since season start - cumulative goal difference - cumulative goals scored - the cumulative points evolution from start to break leg - lr_feat_coeff - nb. games to play at home	final nb pts
lr_6 (**)	Linear	- lr_feat_coef	lr_predict_coeff

	Regression	- nb. pts won at break leg - cumulative goal difference at break leg - rolling 5 avg pts at break leg - nb. games to play at home	
rf_1	Random Forest	- nb. games to play at home - nb. pts won at break leg	final nb pts
rf_2	Random Forest	- lr_feat_coef - nb. pts won at break leg - cumulative goal difference at break leg - rolling 5 avg pts at break leg - nb. games to play at home	final nb pts
rf_3 (*)	Random Forest	- rolling 5 pts at break leg - avg. nb. pts since season start - cumulative goal difference - cumulative goals scored - the cumulative points evolution from start to break leg - lr_feat_coeff - nb. games to play at home	final nb pts
rf_4 (**)	Random Forest	- lr_feat_coef - nb. pts won at break leg - cumulative goal difference at break leg - rolling 5 avg pts at break leg - nb. games to play at home	lr_predict_coeff
xgb_1	XGBoost Regressor	- nb. games to play at home - nb. pts won at break leg	final nb pts
xgb_2	XGBoost Regressor	- lr_feat_coef - nb. pts won at break leg - cumulative goal difference at break leg - rolling 5 avg pts at break leg - nb. games to play at home	final nb pts
xgb_3 (*)	XGBoost Regressor	- rolling 5 pts at break leg - avg. nb. pts since season start - cumulative goal difference - cumulative goals scored - the cumulative points evolution from start to break leg - lr_feat_coeff - nb. games to play at home	final nb pts
xgb_4 (**)	XGBoost Regressor	- lr_feat_coef - nb. pts won at break leg - cumulative goal difference at break leg	lr_predict_coeff

		- rolling 5 avg pts at break leg - nb. games to play at home	

When a model name is followed by a (*) or a (**), it means I tried a quit different approach from my mainstream modeling :

- (*) : In this family of models, I aimed to use the entire history of point evolution until the breaking leg. Pros are that we have more variety in the features and we are closer to the team performance. Cons are the number of variables is not constant and we are more prone to the curse of dimensionality.
  In order to use this approach, we have to pivot our input dataframe (imported from the csv files) on index=['season', 'team'] and columns='leg'.
- (**) : EDA has shown the evolution of the number of points won tends to be linear during a time frame sufficiently large, i.e $nb.\ of\ points \sim A\ x\ leg$. In this approach, the goal is to predict the tendency A for the remaining games. Once computed, the predicted final number of points is given by
  $final\ nb.\ pts\ =\ nb.\ of\ pts\ at\ breaking\ leg\ +\ A(season\ length\ -\ breaking\ leg)$.
  For example, if we interested in Ligue-1, and assume that breaking leg is 27, the nb. of pts at breaking leg is 50 and A=1.8, then the predicted final number of points is

$$50\ +\ 1.8x(38\ -\ 27) = 50\ +\ 1.8\,x\,11\ =\ 69.8$$

As promised before the regression type models' recap table, let me explain briefly the different variables used. Assume that the breaking leg is N.

The main features variables we used are :

- *nb. pts won at break leg* : It is the total number of points won by a team from leg 1 to leg N.
- *cumulative goal difference at break leg* : It corresponds to the difference between the total number of goals scored and the total number of goals conceded from leg 1 to leg N.
- *rolling 5 avg pts at break leg* : The average number of points won during the last 5 games, i.e from legs N, N-1, N-2, N-3 and N-4
- *nb. games to play at home* : The number of games to be played at home till the end of the season
- *lr_feat_coef* : It is the computed tendency of points won between leg 1 and leg N.

For the pivoted approach, we also use the following :

- *avg. nb. pts since season start* : The average number of points won per leg since the start of the season
- *cumulative goal difference* : same as *cumulative goal difference at break leg.*
- *cumulative goals scored* : The number of goals scored since the start of the season.
- *the cumulative points evolution from start to break leg* : This variable gathers in a few words all variables of type *leg-n* and models the evolution of points won game after game since *leg-n* corresponds to the points won from leg 1 to leg n.

Target variables are :

- *final nb pts* : The number of points won at the end of the season
- *lr_predict_coeff* : The tendency of points won from leg N+1 until the end of the season.

Classification type

The idea of the classification approach came to me after I saw the picture below when I applied a reduction dimensional algorithm to the cumulative points evolution from the season start to the break leg :

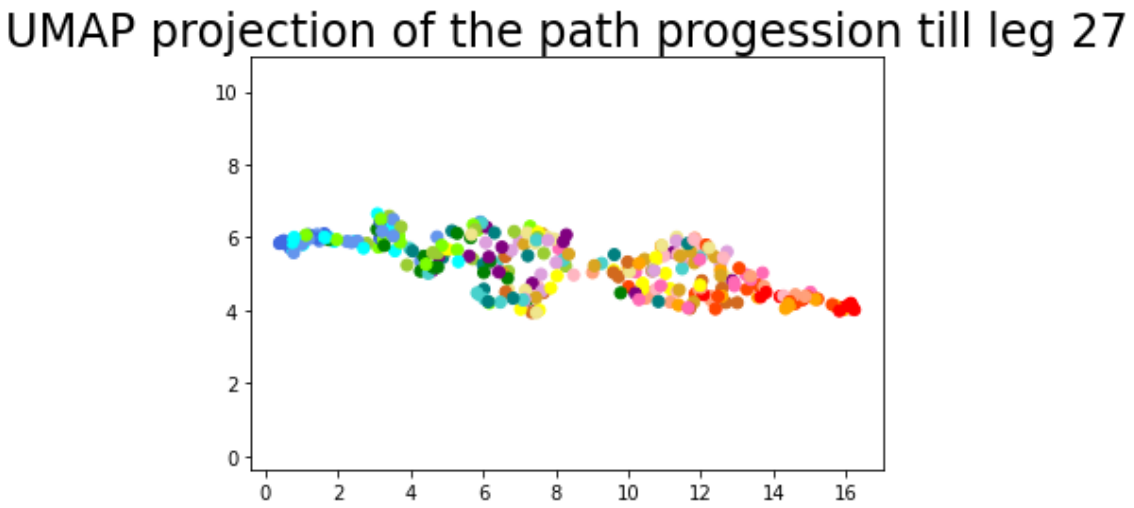

Fig. 21 : UMAP 2D embedding from the cumulative point evolution per team-season pairs in Ligue-1. Colors are related to the final rank (e.g. blue is rank 1 and red is rank 20). Screenshot by author.

Colors are attributed to ranks according to the rule from the following figure

```
color_2_position = {1: "royalblue",
                    2: "aqua",
                    3: "cornflowerblue",
                    4: "chartreuse",
                    5: "yellowgreen",
                    6: "green",
                    7: "teal",
                    8: "purple",
                    9: "mediumturquoise",
                    10: "plum",
                    11: "khaki",
                    12: "goldenrod",
                    13: "yellow",
                    14: "chocolate",
                    15: "lightpink",
                    16: "hotpink",
                    17: "lightsalmon",
                    18: "orange",
                    19: "orangered",
                    20: "red"}
```

Fig. 22 : mapping between colors and positions. Screenshot by author

But how do we get there ? Good question and here is the story.

While I was developing the regression type models using the cumulative points evolution as features, I started to interpret that evolution as a function from $\{1, 2, 3, \ldots, 38\} \rightarrow \{0, 1, \ldots, 114\}$ with some label (the final rank) so that I could represent all the team's evolution from all available seasons in one space. Since the input space is finite, the function can in fact be described as a vector where the j-th component corresponds to the total number of points after j games.

So far so good, but … I want to see in one plot all my evolutions. If N games have been played, that means that I need to represent the vectors from an N-dimensional space in a 2D or 3D plots, i.e. to perform a reduction of dimensionality. Since UMAP[4] is quite recent and has the ability to preserve geometric structures quite well and runs pretty fast, I decided to use it and got the picture above.

As for Regression type models I tried different models with or without dimensionality reduction. They are summarized in the following table

Model Name	Model Type	Features
fuzzy_umap	Custom Fuzzy	- UMAP 2D-embedding
xgb_class_umap	XGBoost Classifier	- UMAP 2D-embedding
xgbrf_class_umap	XGBoost Random Forest Classifier	- UMAP 2D-embedding
xgb_class_no_umap	XGBoost Classifier	- rolling 5 avg pts at break leg - avg. nb. pts since season start - cumulative goal difference - cumulative goals scored - the cumulative points evolution from start to break leg
xgbrf_class_no_umap	XGBoost Classifier	- rolling 5 avg pts at break leg - avg. nb. pts since season start - cumulative goal difference - cumulative goals scored - the cumulative points evolution from start to break leg

While documentation about XGBoost Classifier and XGBoost Random Forest Classifier can be found in the official documentation[5] Custom Fuzzy is self-made. So let us explain how it works in 5 steps. For sake of simplicity, assume that N legs have already been played:

1. Build the training dataset as follows : for every team in each season used for model training, build the N-dimensional season evolution vector by the point evolution from season start till leg N and choose as label the final rank, i.e. the rank the team has once the season is over.
   Prepare the test dataset in the same way.
2. Apply UMAP on the training set in order to learn the embedding from an N-dimensional space to a 2-dimensional one by using UMAP's *fit_transform* method
3. Apply the learnt embedding to the season to test by using UMAP's *transform* method.
4. For every team in the test season, compute the following based on Euclidean distances to historical point and the corresponding label :

[4] https://umap-learn.readthedocs.io/en/latest/
[5] https://xgboost.readthedocs.io/en/stable/python/python_api.html#module-xgboost.sklearn

$$score(TEAM_A) = \sum_{(t,k)} \|Emb(t) - Emb(TEAM_A)\|/k,$$

where the summation is over all team's evolution-label pairs (t,k) in the training set, Emb is the embedding function learnt by UMAP.

5. Rank the teams according to the scores in an ascending order. Hence, the team with the lowest score is ranked first.

Ranking type

One of the projects I worked on in my job was about building a ranking algorithm. When doing literature research, I found some interesting ideas to explore and libraries addressing this challenge. This is how I got to know *xgboost.XGBRanker*.

While my job related problem had to deal with tens (if not hundreds) of thousands of items, the scalability issue didn't allow me to go further with this object. In contrast, this project sounded the perfect playing ground to give it a try.

So how do we use it ?

Data preprocessing steps are the same as for the classification type approach : the initial dataframe is pivoted in order to have for 1 team in a specific season its cumulative point evolution from the season start to the break leg in 1 line.

According to the documentation[6], it is important to sort our dataframe per group whose length has to be specified. Indeed, *XGBRanker* will split along indices the dataframe in several pieces on which the ranking procedure is learned. The screenshot below shows a quite explicit example on how it works; just think of *qid* as the group ID. The length of the group array is equal to the number of groups.

[6] https://xgboost.readthedocs.io/en/stable/python/python_api.html#module-xgboost.sklearn

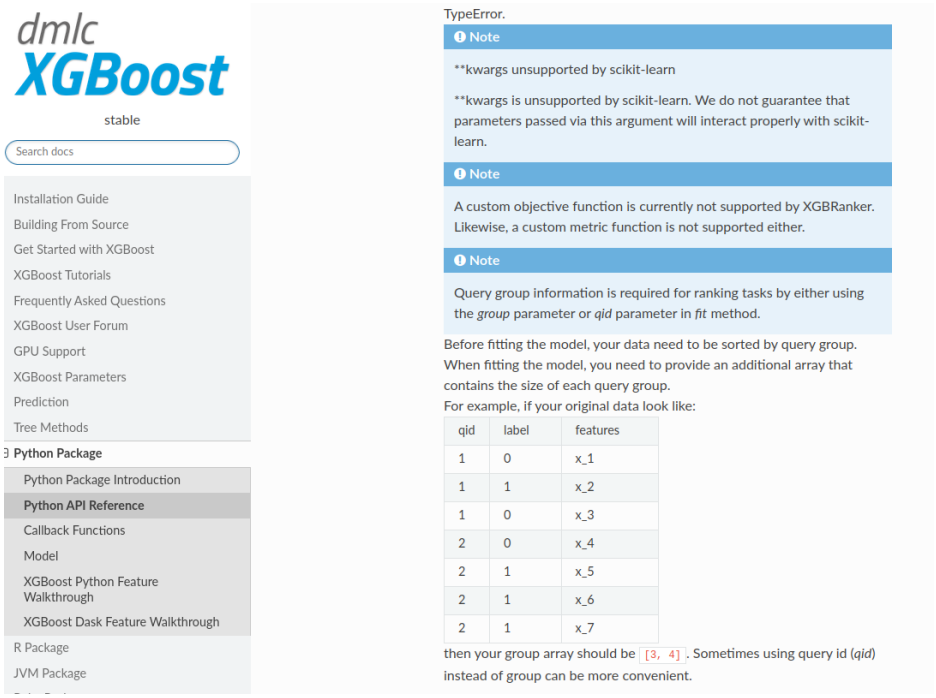

Fig 23. Screenshot taken by the author of the official XGBoost documentation explaining how to use *XGBRanker*

In our case, *qid* is the season (something like 2011-2012) and the 'size' of the group is equal to the number of teams competing in the given championship, i.e. 20 (or 18 if we are looking at Bundesliga). So, we first have to sort by season and make sure to pass to the 'group' keyword a constant array of length equal to the number of seasons.

Like the Classification, I had to decide to get along with one of the following options :

A. I apply the XGBoost Ranker to the original data
B. I apply the XGBoost Ranker after I applied UMAP to the data
C. Well, why should we choose between A and B ? Let's try both

And … I chose option C.

Model Name	Model Type	Features	Target
xgb_ranker_umap	XGBoost Ranker	- UMAP 2D-embedding	Final rank
xgb_ranker_no_umap	XGBoost Ranker	- rolling 5 avg pts at break leg - avg. nb. pts since season start - cumulative goal difference - cumulative goals scored - the cumulative points evolution from start to break leg	Final rank

What metrics ?

This is the question to determine the value of a ranking. But this raises the question of what is a fair value of a position.

In order to answer this question, it worth to remind consequences a position has on a team :

- **Position 1** : Winner of the league. Going to play the Champions League the next season;
- **Position 2** : Qualified to participate in the next Champions League;
- **Position 3** : Depending on the championship, the team is either directly qualified for the Champions League or is going to play the Champions League's qualifying round;
- **Positions 4-5** : Qualified to play an European competition the next season
- **Position 18 (resp. 16) out of 20 (resp. 18)** : The team has to win a play-off in order to remain in the league;
- **Positions 19-20 (resp. 17-18) out of 20 (resp. 18)** : the teams are relegated to the lower league;
- **Other positions** : There aren't any sporting consequences; only financial ones :).

Since Ligue-2 is France's second level, there isn't any access to European competition (Champions League or Europa League) from the final ranking. Instead, the top-2 teams are promoted to Ligue-1 and the 3rd has play-off games against the 18th from Ligue-1 and the winner will play in the highest division the next season.

Usually, items (here teams) are ranked according to a descending order from some gain function. Here, we cannot use such a function due to the sport consequence explained above **and** the following considerations :

- getting wrong between the positions 18 (out of 20) and 19 is worse than being wrong between positions 5 and 6;
- reordering the ranking to take account of the position sport value to fit a traditional framework can skew the way to evaluate soccer rankings since a switch between rank i and i+1 can, for example, mean an inversion between position 4 and 17

Taking them into account, we end up with a gain function that has to be U-shaped. As such, I thought about the following proposals

The first one is generated by the following function in the figure below

```python
def _get_rank_position_scoring(nb_teams: int):
    bonus_position = {1: 300,
                      2: 280,
                      3: 270,
                      4: 260,
                      5: 220,
                      nb_teams: 350,
                      nb_teams - 1: 340,
                      nb_teams - 2: 300,
                      nb_teams - 3: 280}

    for rk in range(6, nb_teams - 3):
        upper = ((nb_teams / 2) - 3) ** 2
        lower = ((nb_teams / 2) - 5) ** 2
        if 2 * rk < nb_teams:
            bonus_position[rk] = int(220 * (rk - nb_teams / 2) ** 2 / lower)
        else:
            bonus_position[rk] = int(280 * ((rk - nb_teams / 2) ** 2) / upper)

    return bonus_position
```

Fig 24.A: Code snippet from author

When applied to a championship with 20 teams, one gets :

```
{1: 300,
 2: 280,
 3: 270,
 4: 260,
 5: 220,
 6: 140,
 7: 79,
 8: 35,
 9: 8,
 10: 0,
 11: 5,
 12: 22,
 13: 51,
 14: 91,
 15: 142,
 16: 205,
 17: 280,
 18: 300,
 19: 340,
 20: 350}
```

Fig. 24.B : Gain distribution according to the rank (used as key) for the second method when 20 teams are competing. Screenshot by author

The second gain function I've considered is obtained by the function in the following snippet

```python
def _get_rank_scoring(nb_teams: int):
    max_bonus = 250 * nb_teams
    return {1 + i // 2 if i % 2 == 0 else nb_teams - i // 2: max_bonus - i * 250 for i in range(nb_teams)}
```

Fig. 25.A: Code snippet from author

Applied to Ligue-1, we get

```
{1: 5000,
 2: 4500,
 3: 4000,
 4: 3500,
 5: 3000,
 6: 2500,
 7: 2000,
 8: 1500,
 9: 1000,
 10: 500,
 11: 250,
 12: 750,
 13: 1250,
 14: 1750,
 15: 2250,
 16: 2750,
 17: 3250,
 18: 3750,
 19: 4250,
 20: 4750}
```

Fig. 25.B : Ligue-1 gain distribution according to the rank (used as key) for the first method. Screenshot by author

One of the metrics commonly used to measure the quality of a ranking is NDCG which stands for *Normalized Discounted Cumulative Gain*. Assume we have to rank n items i_1, …, i_n and denote by σ a permutation[7] of n elements and by σ^{-1} its inverse. Let g_j be the gain generated by the item i_j and assume that $g_i = g_j$ if and only if $i = j$. Finally, let π be the permutation such that :

$$g_{\pi^{-1}(1)} > g_{\pi^{-1}(2)} > \ \dots \ > g_{\pi^{-1}(n)}.$$

Without loss of generality, we can assume that π is the identity permutation, that is $\pi(k) = k$ for all $k = 1, \dots, n$. Hence π^{-1} is also the identity permutation.
With all these definition in hands, we can now introduce DCG(σ), the discounted cumulative gain generated by the ranking σ :

$$DCG(\sigma) = \sum_{k=1}^{nb.\,teams} \frac{g_{\sigma^{-1}(k)}}{log_2(1+k)}$$

As such, the information provided by DCG is only half informative. Indeed, it only says the greatest, the better. The quality information (i.e. how good it is ?) depends on DCG(π). Indeed, if $DCG(\sigma) = 99$ then whether $DCG(\pi) = 100$ or $DCG(\pi) = 1000$, the conclusion won't be the same. To incorporate that piece of information, we *normalize* DCG(σ) by DCG(π) … and gives birth to the normalized discounted cumulative gain generated by the ranking σ :

$$NDCG(\sigma) = \frac{DCG(\sigma)}{DCG(\pi)}.$$

This metric is read as follows : the closest NDCG(σ) to the 1, the better is the ranking σ.

[7] Rank n elements consists of rearranging them according to some rule; that's nothing else than a permutation.

And the winners are …

Going further with so many models does make sense; it only adds more confusion. Consequently, I chose to keep the basic model (which is the one used by LFP[8]) as well as one model from the regression type, one from the classification type and one from the ranking one.
In order to make my selection, I proceeded as follows :

1. for each season S and championship C, compute the NDCG for all models. To do it :
    a. set the test dataset as the soccer results from season S and championship C
    b. set the training dataset as the subset of soccer results from championship C and all seasons but S
    c. train and evaluate all the models
    d. record the NDCG : 1 line contains the season S, the championship C and the NDCG value for each of the models tested.
2. Once we have the $15 \times 6 = 90$ records, we compute the mean NDCG value and its standard deviation for every model in competition.
3. Pick those with the greatest possible mean while being the least volatile possible.

Above, I considered the breaking leg to be 27, which is the last leg being completely played in Ligue-1 before lockdowns were implemented.

When using the gain repartition described in Fig 24.B, we get

championship	bundesliga	liga	ligue-1	ligue-2	premier-league	serie-A
lr_6	98.213333	97.145333	97.069333	96.360667	97.817333	98.013333
lr_4	97.918000	97.364667	97.068000	95.775333	97.766667	97.808667
lr_3	97.978667	97.086667	97.061333	96.216667	97.778000	98.018667
lr_1	98.017333	97.248667	96.983333	95.684000	97.655333	97.847333
lr_5	97.824667	97.309333	96.902000	95.964000	97.792000	97.854667
xgb_class_raw	97.286000	96.356667	96.739333	95.332667	97.936667	96.869333
rf_4	97.642000	97.276000	96.720000	94.656000	97.674000	97.772667
xgb_ranker_raw	96.978000	97.204667	96.674000	94.794000	97.040000	96.674000
rf_2	97.398667	97.290000	96.666000	95.126667	97.988000	97.598667
xgb_2	97.327333	97.046667	96.287333	94.020000	97.404667	97.072000
xgb_4	97.270000	96.494000	96.213333	93.374000	97.388000	97.292000
rf_1	97.893333	97.027333	96.200000	94.548000	97.376667	97.652667
lr_2	97.016000	96.702000	96.099333	95.336000	96.996000	97.311333
naive	97.016000	96.702000	96.099333	95.336000	96.996000	97.311333
xgbrf_class_raw	97.308000	96.714000	95.992667	94.669333	96.574667	97.607333
rf_3	97.722667	97.173333	95.968667	95.162000	97.793333	97.607333
xgb_3	97.700000	96.929333	95.541333	94.580000	96.988000	97.148667
xgb_1	97.447333	97.104000	95.396667	94.233333	97.000000	97.266667
fuzzy_umap	86.676000	88.513333	88.529333	86.808667	85.428667	87.060000
xgbrf_class_umap	84.156667	88.206000	84.703333	85.010000	85.217333	84.832000
xgb_ranker_umap	84.342000	85.658000	83.223333	82.442667	86.840667	82.807333
xgb_class_umap	82.828000	86.817333	82.757333	85.286667	84.446000	83.376000

Fig. 26.A : Average NDCG per championship and algorithm for the gain distribution displayed in Fig. 24.B. Screenshot by author

[8] LFP = Ligue de Football Professionnel = French Professional Football/Soccer League

On the other hand, using the gain repartition displayed in Fig. 25.B, we end with

championship	bundesliga	liga	ligue-1	ligue-2	premier-league	serie-A
lr_3	97.823333	96.870000	97.196000	95.859333	97.716667	98.078000
lr_6	98.218000	96.896000	97.185333	96.257333	97.767333	98.054000
lr_4	97.786667	97.305333	97.128667	95.616000	97.642000	97.862000
lr_1	97.854000	97.102667	97.049333	96.024667	97.684000	97.947333
lr_5	97.672667	97.208000	96.978667	95.696667	97.588667	97.932000
xgb_class_raw	97.456000	96.552000	96.914000	94.900667	97.881333	96.952667
xgb_ranker_raw	96.728000	97.470000	96.706667	94.417333	97.229333	97.044000
rf_4	97.497333	97.378000	96.503333	95.535333	97.672667	97.854667
rf_2	97.288000	97.030667	96.484667	95.444000	97.168000	97.662667
rf_1	97.562000	97.042667	96.409333	94.752000	97.288000	97.786667
lr_2	96.681333	96.512000	96.368667	95.262667	97.128667	97.507333
naive	96.681333	96.512000	96.368667	95.262667	97.128667	97.507333
xgbrf_class_raw	97.247333	96.853333	96.317333	95.258667	96.908000	97.779333
xgb_2	96.914000	96.900667	96.209333	94.200000	97.075333	97.130667
xgb_4	97.218667	96.392000	96.162000	92.867333	97.173333	97.314000
rf_3	97.490667	97.234667	95.956000	95.622667	97.771333	97.486000
xgb_1	97.159333	96.901333	95.594000	94.401333	97.297333	97.435333
xgb_3	97.606000	97.122000	95.345333	93.748000	96.664667	97.414000
fuzzy_umap	89.423333	89.860000	90.150000	87.924000	90.152667	90.222667
xgbrf_class_umap	90.444667	85.011333	88.626000	84.424000	88.195333	87.268667
xgb_ranker_umap	87.446000	84.852667	87.954667	85.182000	86.964667	87.619333
xgb_class_umap	88.408667	84.968000	87.629333	83.467333	87.063333	85.863333

Fig. 26.B : Average NDCG per championship and algorithm for the gain distribution displayed in Fig. 25.B. Screenshot by author

Considering point 3 from our procedure, I made the following selection in addition to the naive model :

- *Regression type* : lr_6
- *Classification type*: xgb_class_raw
- *Ranking type* : xgb_ranker_raw

Notebooks to scripts

Once experiments are over and that I was satisfied enough with my EDA plots and models, the next stage consists of moving them into .py scripts which are themselves related to my Github repository.

Personally, I find it very important to do since it is the moment the different pieces of code become useful in a production-like setting. Moreover, it is also a way to consolidate the logic behind a code that was prototyped on notebooks and to think like a software engineer.

In the project repository, I chose to keep all the notebooks (the rough drafts to the first refactoring parts) I've written since they are part of my reasoning and I think they might be helpful to show the transition into .py scripts.

That being said, let me explain how I organized my code. First, everything is in the *soccer_season_prediction* directory as shown in the figure below, where the different parts of the code are split over several modules.

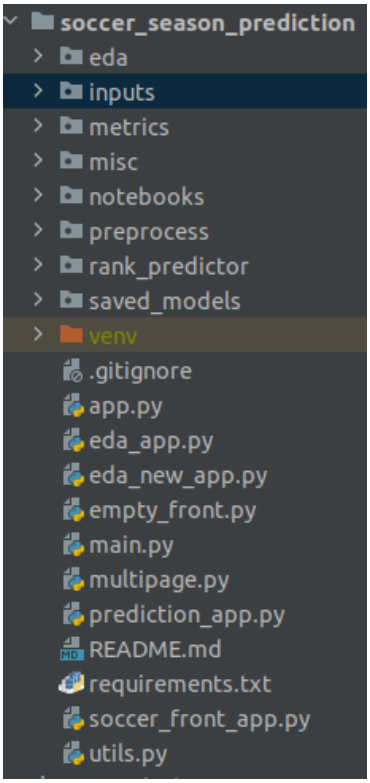

Fig 27 : Project organization. Screenshot by author

As you can see, there are a couple of files directly available. Those containing ‘app’ in their name are related to the Streamlit app to be presented at the next section. The *utils.py* script contains a bunch of general functions that can be used for a general purpose (e.g listing the file from a module, dumping or loading a pickle file).

If I were in a situation where my project would have been used in a “production setting” (requiring an orchestrator such as Airflow), I would have used the *main.py* as my entry point where arguments, if any, would be passed. Since it is not the case, this file only contains a pass command. Finally, there are the traditional .gitignore, requirements.txt and README.md files coming with the use of Github.

The *eda* module contains the scripts used to make all the eda analysis : *colors.py* is here to handle all the color related issues, *basics.py* contains functions to get basic kpis such as the number of participation per team. In *goals_related_eda.py*, one will find 2 aggregation functions related to the number of goals a team scores in a game. The Plotly figures displayed by the app are provided by *figures_eda.py* while layers handling and express plot are written in *utils.py* in order to keep it easier to read.

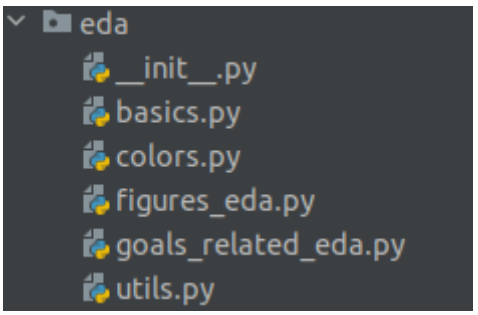

Fig. 28 : Files from the eda module. Screenshot by author

The *inputs* module contains all the data to run the project for all the championships considered and all 15 seasons before the Covid impacted one : 2019-2020. The latter is provided in the module *covid_season*.

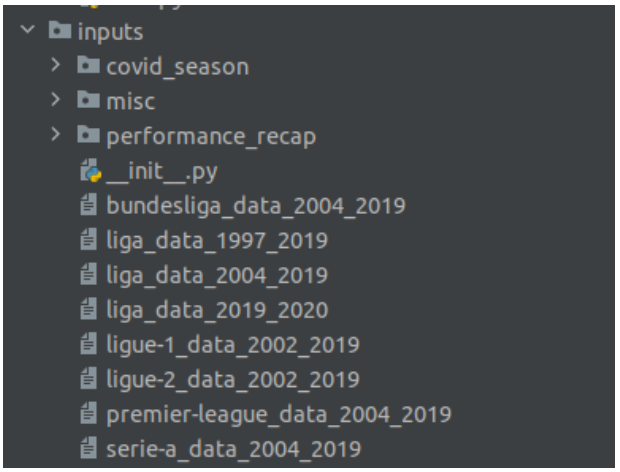

Fig 29. Csv files used as input for the project. Screenshot by author

In *metrics*, one will find a script encompassing all metrics described before. All the notebooks I've written from getting the data, experimenting and prototyping are available between the *notebooks* and the *misc* modules.

The *preprocess* is self-explaining. The most important module is *rank_predictor.* Each of the *naive.py, classification.py, regression.py* and *ranking.py* contain one type of ranker which is defined as an object. These are the one selected in the steps before and are, respectively, named *SoccerNaive, SoccerClassification, SoccerRegression* and *SoccerRanking.* The master file is *ranker.py* which contains the *Ranker* object. It is the one to be called by the user in the app. Depending on the desired type ranking, it will match one the above rankers.

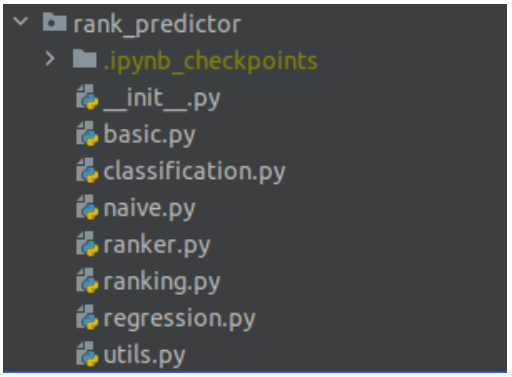

Fig. 30 : The different models available in the rank_predictor module. Screenshot by author

Finally, *saved_models* is here to store the models already being trained as pickle files. The name of those files is made out of the pattern :

{ranking_type}_{championship}_{breaking_leg}_ranker.

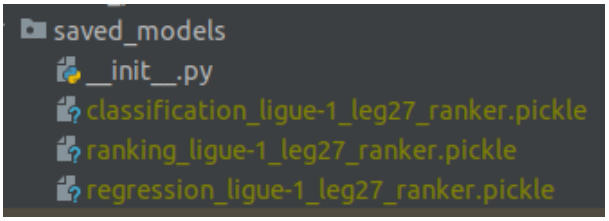

Fig. 31 : folder containing the trained models. Screenshot by author

The Streamlit app :

The cherry of the cake part. After all the work done so far, I wanted to display my results in a beautiful way and practice another important tool for a data scientist : writing an app. Due to its pretty intuitive way to build one, I chose to do it with Streamlit (note that Gradio is another nice framework to build apps and I think it is worth keeping an eye on it in the future), but at the time I had to choose between them, I found Streamlit more adapted to my needs.

So … ladies and gentlemen, let us me introduce you **Soccer Rank Predictor** 🎉🎉🎉

When the app is started, the user is directed to a homepage where he can choose among 3 options what he wants to do. Roughly speaking, the app will either focus on exploratory data analysis or training and using a ML model to make predictions.

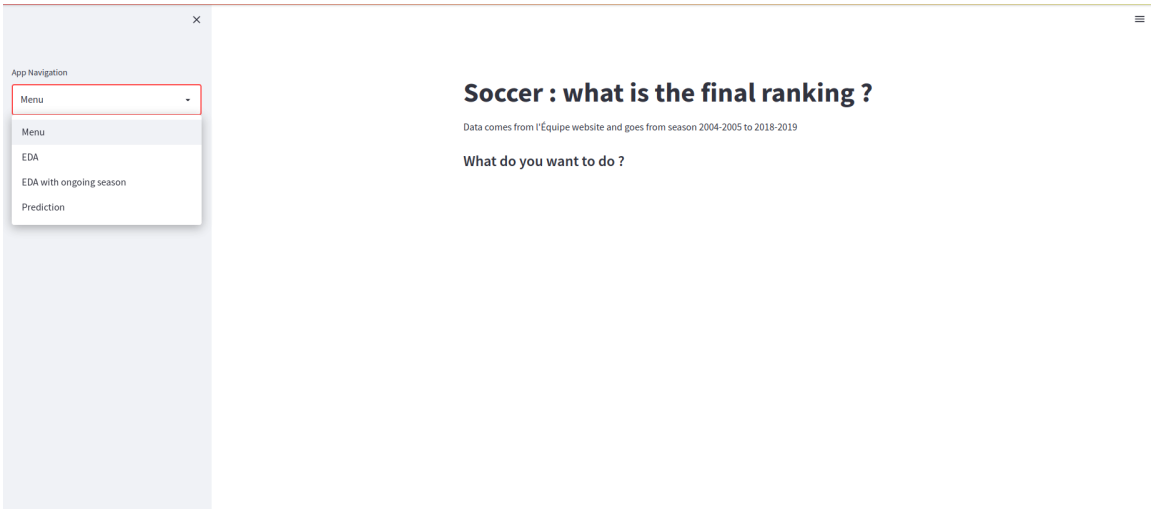

Fig. 32 : Homepage from the app once it has been launched. Left side column allows the user to navigate to another page. Screenshot by author

OPTION 1 : EDA

When selecting this option, the user intends to perform EDA on historical data going from the season 2004-2005 to 2018-2019. After he has clicked on the access the following page, where a basic EDA is provided by default for the selected championship (default is Ligue 1 🇫🇷) :

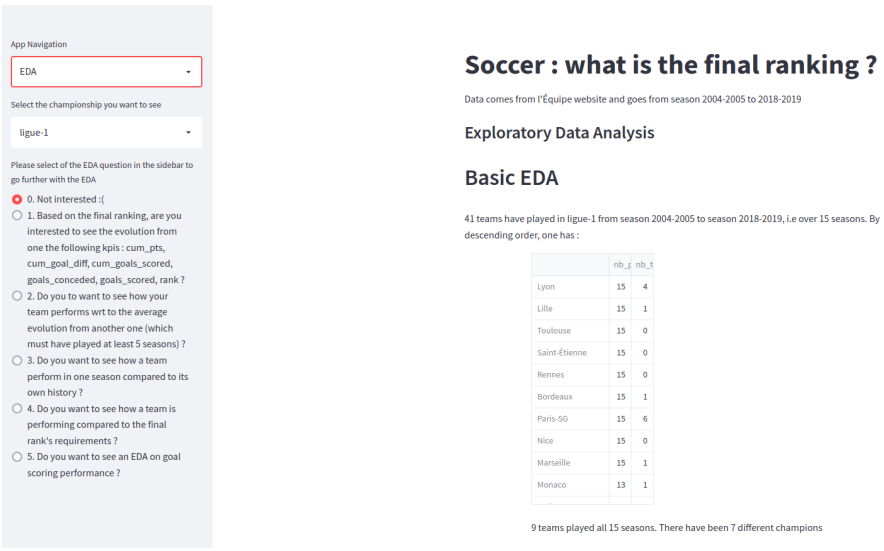

Fig 33.A : On the right column is the first part of the basic EDA. it corresponds to the case the user is not interested in getting deeper insights. This is the default scenario. Screenshot by author

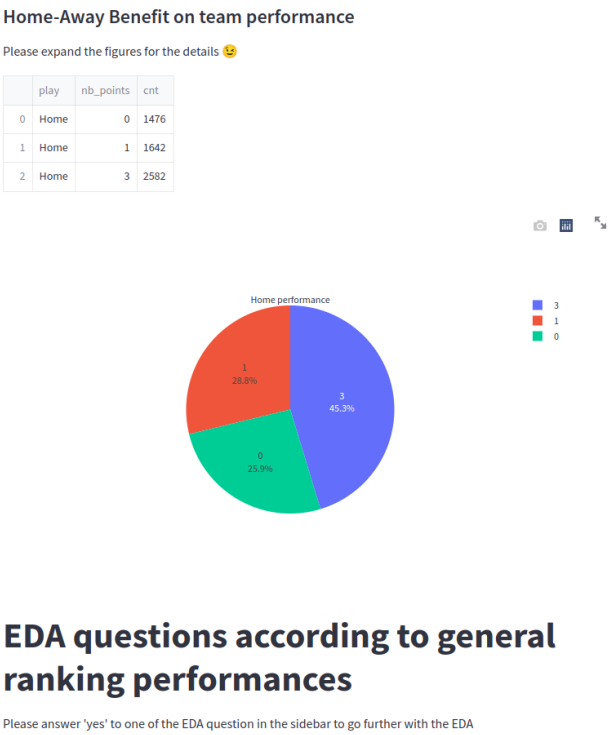

Fig 33.B : second part of the basic EDA which can be accessed by scrolling the app. Screenshot by author

Two pieces of information are provided. The first one recapitulates the number of teams that played at least 1 season, the number of teams having played *all* seasons (in our case 15) and the number of teams having won the championship at least one time. A dataframe is also provided containing the playing teams, their number of participation and the number of times they won the title.

If the user wishes to have a more in-depth data visualization, he can click on 1 of the buttons in the sidebar on the left where 5 options are available. The details will appear below the basic EDA.

The **first option** allows the user to explore how a set of kpis evolves (in average) given the final rank. The benefit of this option is that it gives us an idea about how a team should follow a trend in order to expect at some final rank R.

The user can choose to plot the average evolution from the following kpis :

- *cum_pts* : the cumulative number of points from the season start to leg N.
- *cum_goal_diff* : the cumulative goal difference (i.e the difference between the number of goals scored and the number of goals conceded) from the season start to leg N.
- *cum_goals_scored :* the cumulative number of goals scored from the season start to leg N.
- *goals_conceded :* the average number of goals conceded per leg.
- *goals_scored :* the average number of goals scored per leg.
- *rank :* the average rank evolution during the season.

Please note that in general the average value displayed cannot be reached since it might be a non-integer value.

In order to get an idea of the fluctuation, the user can opt to display a shadowish interval of length 2 times the standard deviation and centered at the average value.

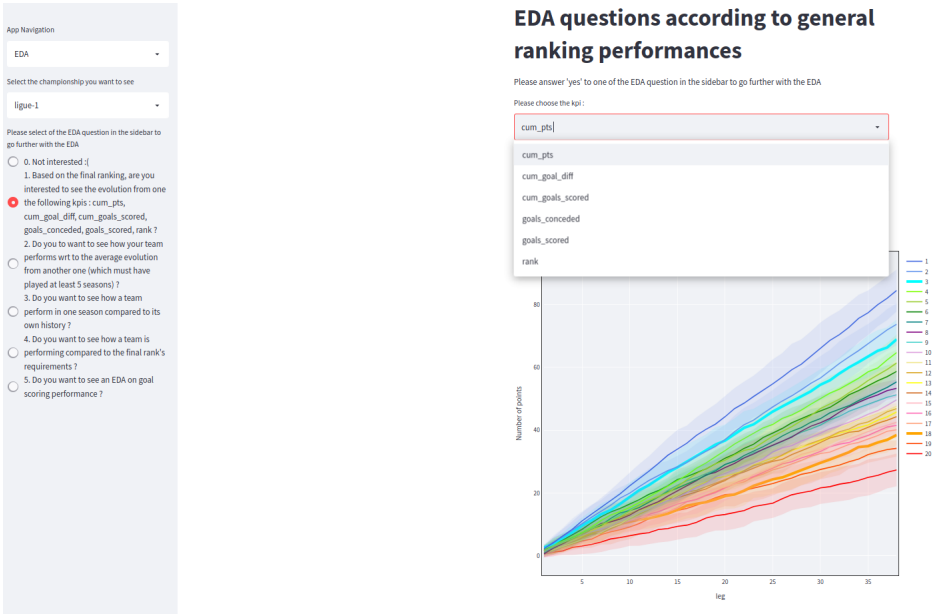

Fig. 34 : View when selecting option 1. Screenshot by author.

The **second option** allows the user to compare the cumulative points evolution from one team for a given season with the average evolution from another one. To be meaningful, the latter team must have played at least 5 seasons among the 15 we are covering.

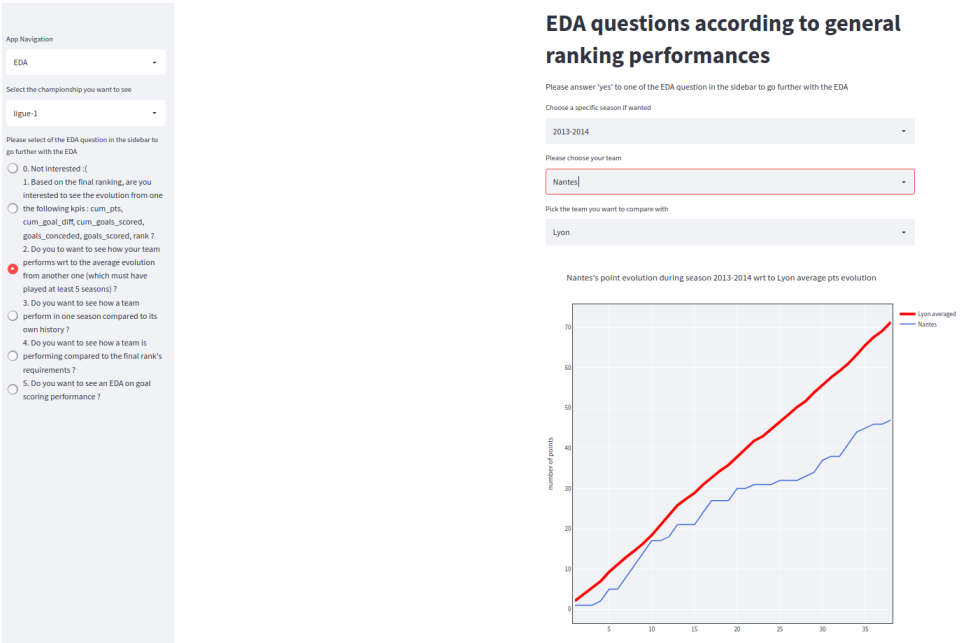

Fig. 35.A : View when selecting option 2. Here Nantes' cumulative point evolution during the season 2013-2014 is compared to the average evolution from Lyon. Screenshot by author.

Of course, it is possible to compare a team with respect to its own average evolution (pending it has played 5 seasons).

Fig. 35.B : Marseille's evolution in 2006-2007 (right) and in 2017-2018 compared to its average evolution. Screenshot by author.

In the plots from figure Fig. 35.B, we can see that Marseille is performing better than its average points progression during the season 2017-2018 whereas in 2006-2007 it was more tedious (though 3 distincts periods of performance can be distinguished: legs 1-10, 11-27 and 28-38).

The **third option** can be seen as complementary to option 2 since it plots *all* performances from a team to be chosen by the user with its average cumulative points evolution. The purpose of this option is to see whether or not a team is consistent year after year.

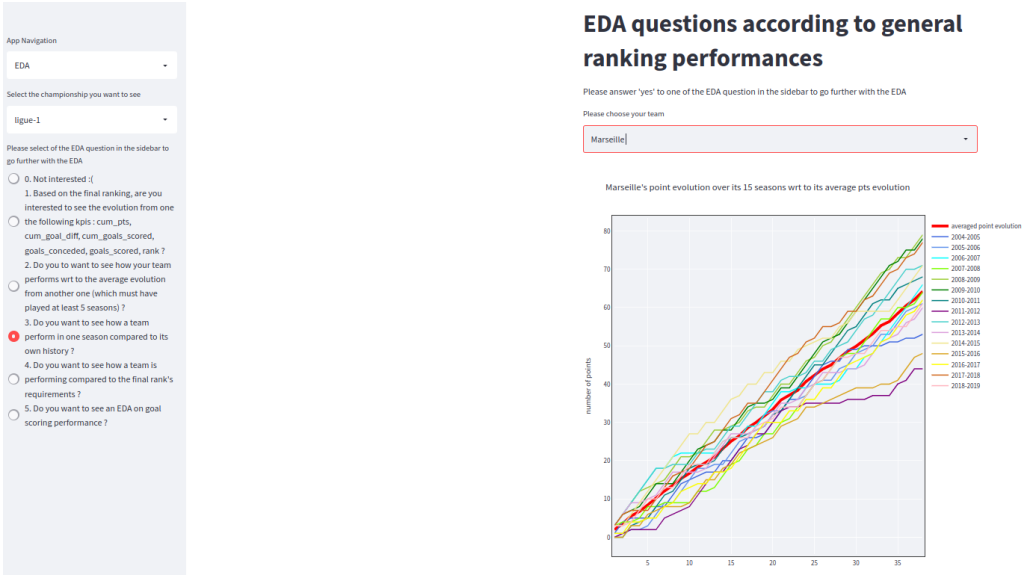

Fig. 36 : View from option 3. Here Marseille's average evolution and all its recorded performances in Ligue-1. Screenshot by author.

The user can choose any team having played at least 5 seasons. Less than this threshold, I don’t find it very informative.

Option 4 is dedicated to see how close the performance from a team is following the cumulative point’s trends based on the final ranking. Users have 3 choices to make : the season, the team and whether or not they wish to see the standard deviation intervals.

When no season is selected, the app will plot the *average* cumulative point evolution as shown below.

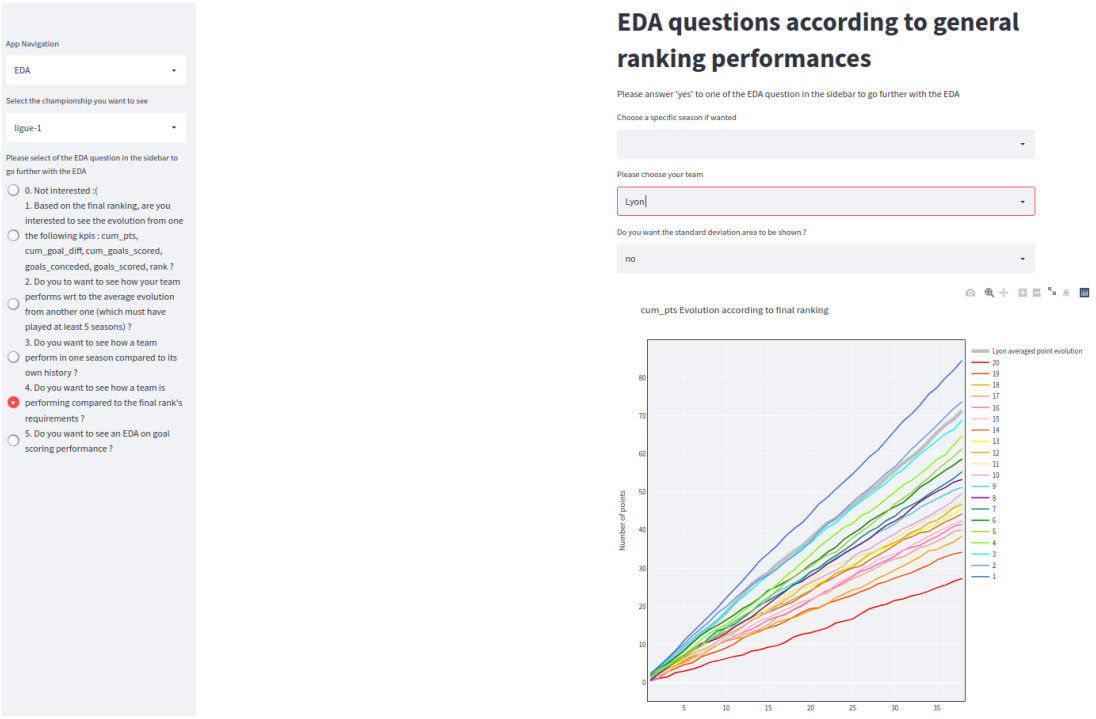

Fig. 37.A : View from option 4 when no season is selected. The plot shows the average cumulative point evolution from a chosen team with respect to the rank related evolution. Screenshot by author.

When a season is specified, only the performance from that season will be displayed.

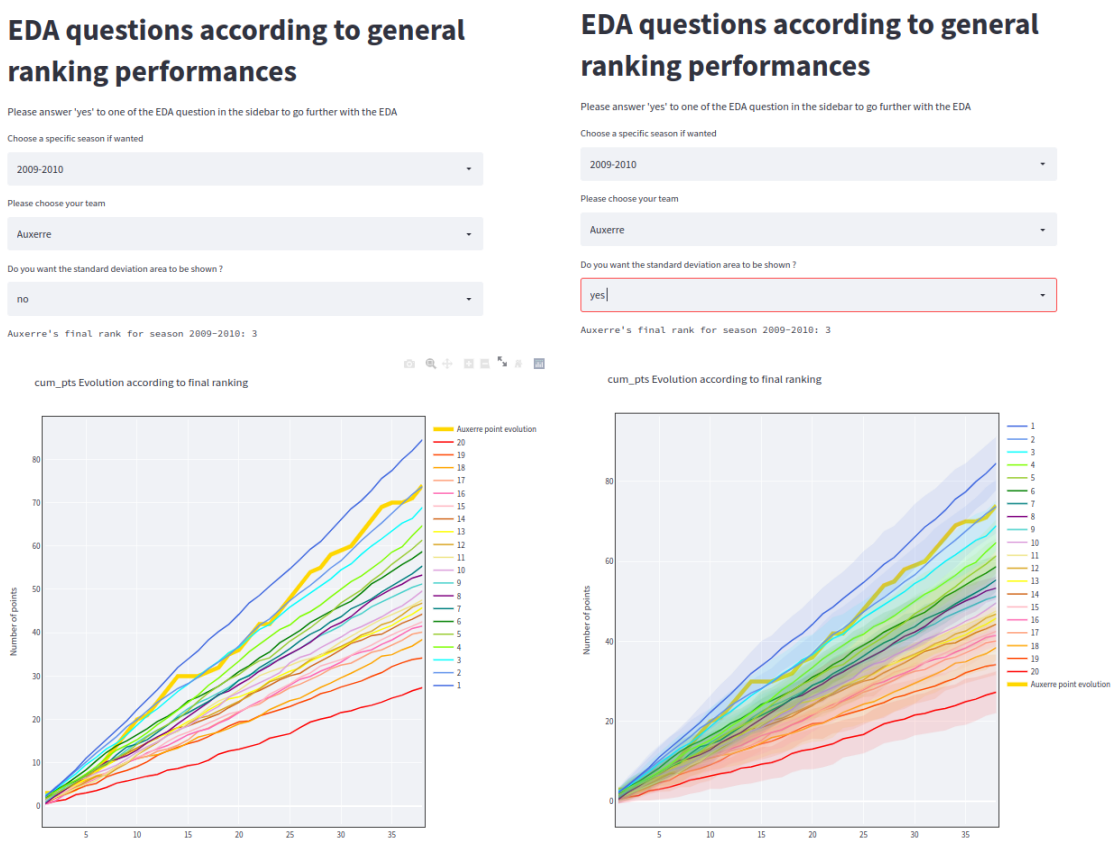

Fig. 37.B : View from option 4 when a season is selected. Left without the standard deviation shades; right, with the shades. Screenshot by author.

As you can see in the right side plot, Auxerre's performance in 2009-2010 should have granted it the second place if we were only looking at trends. However, its final rank is 3. Had we plotted the standard deviation, we would have noticed that the cumulative point evolution was within the error interval from rank 3.

The **fifth option** is quite different from the other options since it focuses on the number of goals a team might expect to score given a set of criteria such as the number of goals scored in the previous game, the average number of goals conceded the last 5 games by the opponent, the number of points won by the team (or opponent) the last games. This list is not exhaustive.

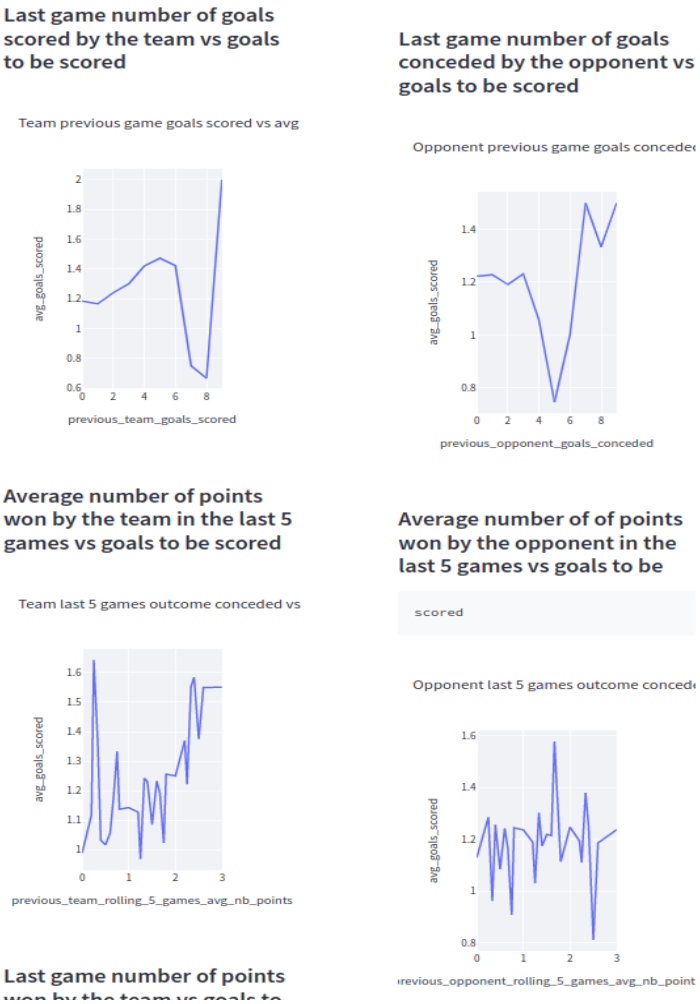

Fig. 38 : Partial view from option 5. Screenshot by author.

OPTION 2 : RANK PREDICTION

The second option is the main purpose of this article : using a ML model (and training it if none is available) to predict the outcome of a season. In order to achieve this, the user needs to provide a couple of pieces of information. The first one is the championship he is interested in. The second point is the kind of model to be used : regression, classification or ranking. We refer the reader to the modeling section for details. Note that the naive model is always included. Finally, and maybe the most interesting parameter, the user is asked to specify the breaking leg, i.e. the last leg where all games are played and their outcome is available.

Once all these parameters are provided, the app will check whether or not a model is available. This is easy to be done since all saved models are named according to the following pattern : {model}_{championship}_{breaking leg}_ranker.pickle

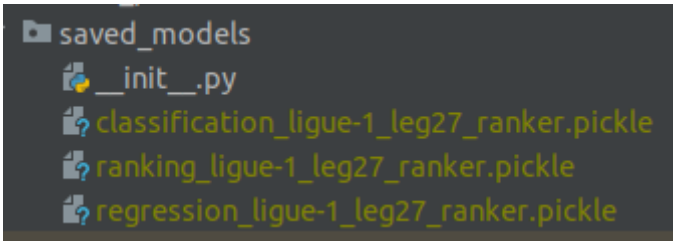

Fig. 39 : folder of saved models. There can only be one model type per championship and known history. Screenshot by author.

The caveat is that only model per triplet (model, championship, breaking leg) is allowed.

When no model is found with the inputs provided by the user, the app will ask him to train one by unchecking 3 seasons to be used for the evaluation. The remaining 12 seasons are composing the training set. Once completed, a model training performance score is displayed as well as a message inviting the user to click to one of the parameter boxes in order to access the prediction part. The reason is that Streamlit runs the app script sequentially from top to bottom at each action.

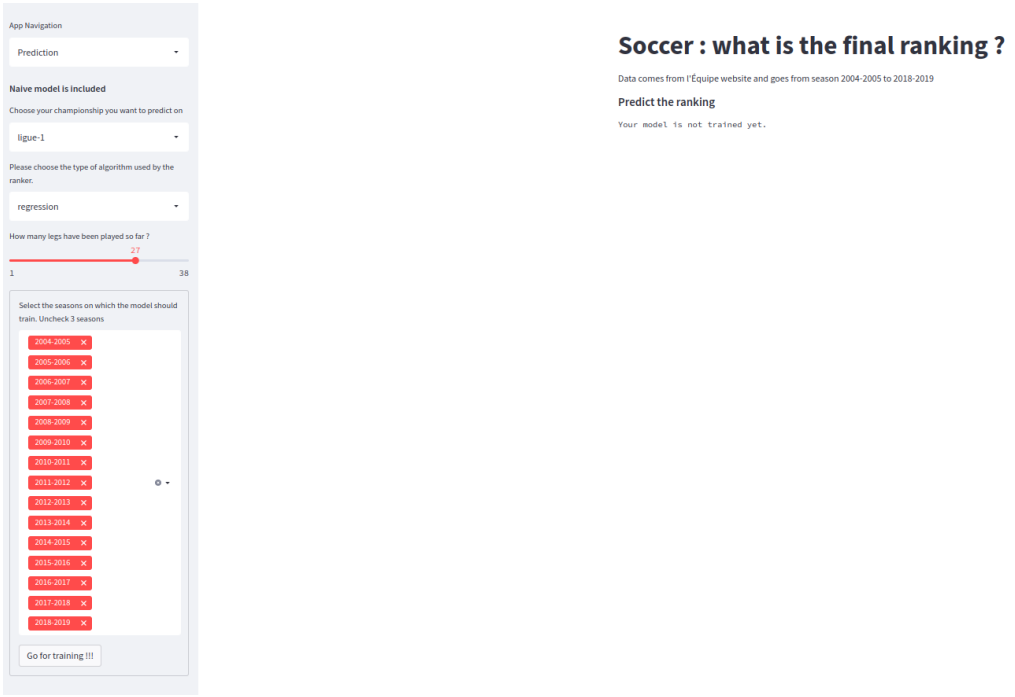

Fig. 40 : View from the prediction page when no model corresponding to the selected model type (here regression) is available. Screenshot by author.

In case a model has already been trained, the user can choose to either use it or train a new one.

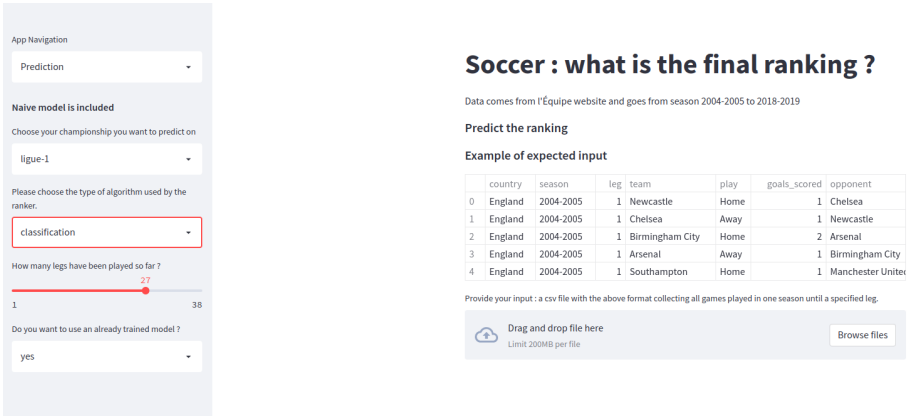

Fig. 41 : View from the prediction page when there exists a model corresponding to the selected model type (here classification) is available. Screenshot by author.

When using an existing model, the only remaining thing to do is to provide an input file containing data for the season the user is interested in and having the same column names and structure as the snipped dataframe provided as example. Then … tada predictions are displayed with a recap of the current standing after the breaking leg is over as well as the naive model's outcome.

OPTION 3 : EDA ON COVID SEASON

Exploratory data analysis during the 2019-2020 season is different from the one for previous seasons. Indeed, only partial pre-covid data is available. In order to get insights from the past, the choice has been made to display only 1 figure. Of course, it can be used for any other (incomplete) season. It is not limited to the covid season even though it was primarily intended for it.

In order to use it, the user needs to choose the championship he is interested in and to provide an input file for the season he wants to analyze and for which the season is not over. The first input allows us to get the historical records so that we can make the comparison. One could select one championship and provide a file about another one; but it would not be informative … except in the case one is interested in the question "Had team A played in championship C2 instead of C1, how would it have positioned itself in C2 with its performance from C1 ?"

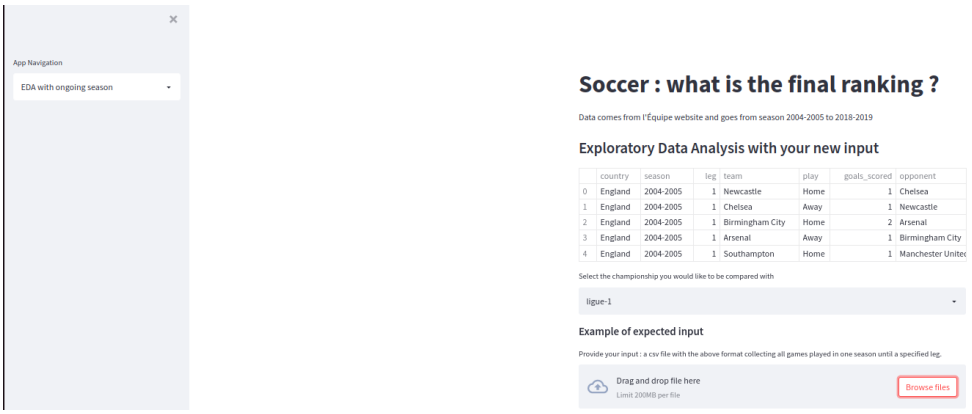

Fig. 42: View from the "*EDA on ongoing season"* page when landing on it. Screenshot by author.

As a use case, I thought about the first covid impacted season. However it can be used for any other seasons (past or future) as long as the number of teams matches with the one from seasons 2004-2005 to 2018-2019.

So far so good, inputs have been provided. So what next ? Well, the user can compare the performance of one team from the input file provided and compare its evolution with the trends of 3 metrics based on the final rank. Those are imputed from the historical record.

The default display is about the evolution of cumulative points. Now, if the user wants to investigate further and look at other kpis, it can be done by selecting the corresponding metric. Moreover, it is also possible to replay a past season by using the slider related to the “known part of the season”, i.e. a number of played legs.

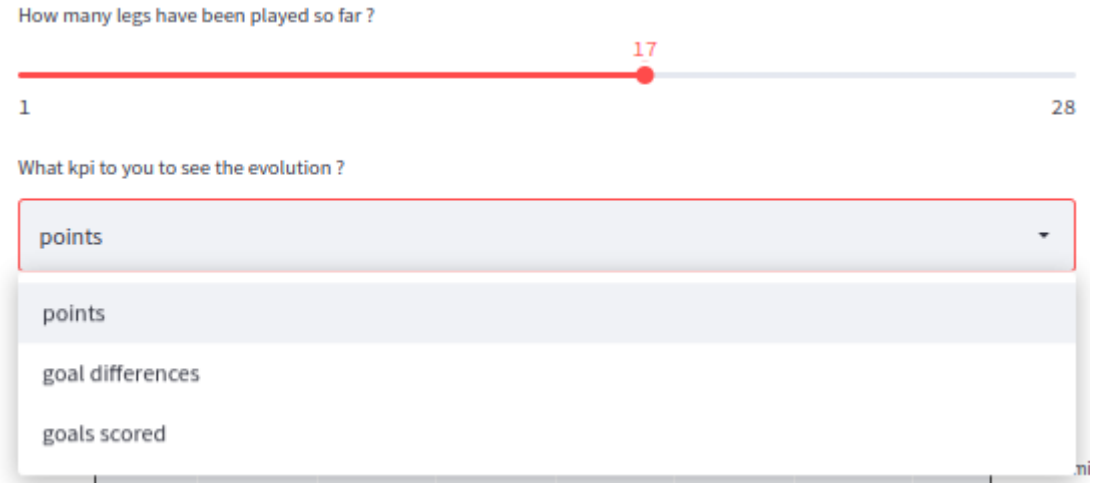

Fig. 43: Parameters to be used to select another metric or vary the length of the known part of the season. Screenshot by author.

As in the EDA option, the user can also display an uncertainty shade. Again, it is built from the standard deviation.

A slider is also added in order to play with an already completed season.

In terms of scripts, it required me to write 5 files :

- *soccer_front_app.py* : the main file to be called to run the app. It will instantiate a MultiPage object allowing us to build an app with several pages.
- *eda_app.py* : the file contains all the code to perform the EDA work described in option 1.
- *eda_new_app.py* : this file is similar to the previous one, but it is designed for a season that is not complete, i.e. the number of legs completed is lower than the championship’s length. It will be called for option 3.
- *prediction_app.py* : this file is run when option 2 is selected. It contains the interface for training and using a model to predict the ranking.
- *multipage_app.py* : this file contains a single class, named MultiPage, with two methods. The first one, *add_page* appends a list with a dictionary like {“title” : my_title, “func”: my_function}. The *my_function* variable is a function to render the

page in Streamlit. The second method, called *run*, calls the my_function function from the selected page. All the code allowing a user to interact with the Streamlit interface from a particular page is contained in my_function.
Hence, one can see each page as an app.

```python
import os
import logging
import streamlit as st
from multipage import MultiPage
import eda_app as eda_app
import prediction_app as prediction_app
import eda_new_app as eda_new_app
import empty_front as empty_front

# Create Streamlit Application
app = MultiPage()

# Add Logger
LOGLEVEL = os.environ.get("LOGLEVEL", "INFO").upper()
logging.basicConfig(format='%(asctime)s - %(levelname)s - %(message)s',
                    level=LOGLEVEL)
logger = logging.getLogger()

# Title of the main page
st.title('Soccer : what is the final ranking ?')
st.markdown(" Data comes from l'Équipe website and goes from season 2004-2005 to 2018-2019")

# Add all your applications (pages) here
app.add_page("Menu", empty_front.app)
app.add_page("EDA", eda_app.app)
app.add_page("EDA with ongoing season", eda_new_app.app)
app.add_page("Prediction", prediction_app.app)

# The main my_app
app.run()
```

Fig. 44 : Content from soccer_front_app.py file. Screenshot by author.

Launching the Streamlit app requires to run the following command line in a terminal :

```
streamlit run soccer_front_app.py
```

where we assumed it to be in the folder where *soccer_season_prediction* is located. If you are positioned somewhere else in your terminal you can either move to the *soccer_season_prediction* folder using the cd command and then run the above line or you run it directly from your position by providing the relative path to the app file to run.

Running the 2019-2020 end of season

We are finally getting to the point where we can make our prediction (with a nice interactive interface :)) on Ligue-1 final ranking had the season been played till the end. We are going to use models being trained on all seasons from 2004-2005 to 2018-2019 but 2007-2008, 2011-2012 and 2016-2017.

 ….

When Ligue-1 and Ligue-2 were paused (before being freezed) due to the Covid-19 outbreak, the standings after the last leg where all planned games were played were :

leg	team	rank	cum_pts	cum_goals_scored	cum_goal_diff
27	Paris-SG	1	68.0	75.0	51.0
27	Marseille	2	55.0	39.0	12.0
27	Rennes	3	47.0	33.0	9.0
27	Lille	4	46.0	34.0	7.0
27	Lyon	5	40.0	42.0	16.0
27	Montpellier	6	40.0	35.0	6.0
27	Monaco	7	40.0	43.0	1.0
27	Reims	8	38.0	25.0	4.0
27	Nice	9	38.0	39.0	2.0
27	Strasbourg	10	38.0	32.0	0.0
27	Nantes	11	37.0	28.0	-1.0
27	Bordeaux	12	36.0	39.0	6.0
27	Angers	13	36.0	26.0	-7.0
27	Brest	14	34.0	34.0	-2.0
27	Metz	15	31.0	25.0	-9.0
27	Saint-Étienne	16	29.0	28.0	-16.0
27	Dijon	17	27.0	25.0	-11.0
27	Nîmes	18	27.0	28.0	-14.0
27	Amiens	19	22.0	29.0	-19.0
27	Toulouse	20	13.0	21.0	-35.0

Fig.45.A : Ligue-1 standing after 27 legs. Highlighted is Amiens' rank. Screenshot by author

and

leg	team	rank	cum_pts	cum_goals_scored	cum_goal_diff
27	Lorient	1	54	45	21
27	Lens	2	50	38	14
27	Troyes	3	50	33	9
27	AC Ajaccio	4	49	37	15
27	Clermont	5	47	33	8
27	Le Havre	6	41	37	12
27	Guingamp	7	39	39	8
27	Valenciennes	8	39	21	2
27	Grenoble	9	35	26	0
27	Auxerre	10	34	31	2
27	Caen	11	34	32	0
27	Sochaux	12	34	28	0
27	Chambly	13	34	24	-6
27	Nancy	14	33	26	1
27	Châteauroux	15	31	20	-17
27	Rodez	16	29	29	-4
27	Paris FC	17	27	21	-18
27	Niort	18	25	29	-11
27	Le Mans	19	25	28	-15
27	Orléans	20	19	21	-21

Fig. 45.B : Ligue-2 standing after 27 legs. Highlighted is Clermont's rank. Screenshot by author.

Curious to see what the models predicted ? Let's find out the predictions (I've summarized the results in one table per championship to ease the comparison) :

- **Ligue-1**

RANK	NAIVE	REGRESSION TYPE	CLASSIFICATION TYPE	RANKING TYPE
1	Paris-SG	Paris-SG	Paris-SG	Paris-SG
2	Marseille	Marseille	Marseille	Marseille
3	Rennes	Rennes	Rennes	Rennes
4	Lille	Lille	Lille	Lille
5	Monaco	Lyon	Montpellier	Nice
6	Montpellier	Montpellier	Bordeaux	Montpellier
7	Lyon	Monaco	Nice	Lyon
8	Strasbourg	Nice	Strasbourg	Nantes
9	Nice	Reims	Angers	Monaco
10	Reims	Strasbourg	Nantes	Bordeaux
11	Nantes	Nantes	Reims	Strasbourg
12	Bordeaux	Bordeaux	Lyon	Reims
13	Angers	Angers	Monaco	Saint-Étienne
14	Brest	Brest	Brest	Angers
15	Metz	Metz	Saint-Étienne	Brest
16	Saint-Étienne	Saint-Étienne	Metz	Nîmes
17	Dijon	Dijon	Nîmes	Dijon
18	Nîmes	Nîmes	Dijon	Metz
19	Amiens	Amiens	Amiens	Amiens
20	Toulouse	Toulouse	Toulouse	Toulouse

I colored with green the champion, in blue the rank to be directly qualified to the UEFA Champions League, in light purple the ranks giving access to european competition (Europa League or qualifying games to UEFA champions League or Europa League). I used red to highlight the ranks going directly to Ligue-2 and in salmon the play-off position.

The interesting point is that the top-4 and the flop-2 are the same regardless of the model type. From what can be seen in figure [Fig. 45.A], there were tough competitions between different groups of teams forming the middle range of the ranking. Figure [Fig. 46] below shows how close predictions were in terms of expected final number of points.

season	team	naive_predicted_rank	predicted_final_nb_pts
2019-2020	Paris-SG	1	95.7037
2019-2020	Marseille	2	77.4074
2019-2020	Rennes	3	66.1481
2019-2020	Lille	4	64.7407
2019-2020	Monaco	5	56.2963
2019-2020	Montpellier	6	56.2963
2019-2020	Lyon	7	56.2963
2019-2020	Strasbourg	8	53.4815
2019-2020	Nice	9	53.4815
2019-2020	Reims	10	53.4815
2019-2020	Nantes	11	52.0741
2019-2020	Bordeaux	12	50.6667
2019-2020	Angers	13	50.6667
2019-2020	Brest	14	47.8519
2019-2020	Metz	15	43.6296
2019-2020	Saint-Étienne	16	40.8148
2019-2020	Dijon	17	38.0000
2019-2020	Nîmes	18	38.0000
2019-2020	Amiens	19	30.9630
2019-2020	Toulouse	20	18.2963

season	team	regression_predicted_rank	predicted_final_nb_pts
2019-2020	Paris-SG	1	90.5220
2019-2020	Marseille	2	71.0684
2019-2020	Rennes	3	61.9423
2019-2020	Lille	4	59.8906
2019-2020	Lyon	5	54.3156
2019-2020	Montpellier	6	53.4800
2019-2020	Monaco	7	52.6007
2019-2020	Nice	8	50.9795
2019-2020	Reims	9	50.9246
2019-2020	Strasbourg	10	50.3469
2019-2020	Nantes	11	49.5720
2019-2020	Bordeaux	12	49.0728
2019-2020	Angers	13	47.6455
2019-2020	Brest	14	45.8315
2019-2020	Metz	15	41.4971
2019-2020	Saint-Étienne	16	39.3934
2019-2020	Dijon	17	36.9467
2019-2020	Nîmes	18	35.8586
2019-2020	Amiens	19	30.5572
2019-2020	Toulouse	20	18.7953

Fig. 46: Expected points predictions for Ligue-1. Left is the naive model; right the regression type model. Screenshot by author.

- **Ligue-2**

RANK	NAIVE	REGRESSION TYPE	CLASSIFICATION TYPE	RANKING TYPE
1	Lorient	Lorient	Lorient	Lorient
2	Troyes	Lens	Lens	Lens
3	Lens	Troyes	Troyes	AC Ajaccio
4	AC Ajaccio	AC Ajaccio	Clermont	Clermont
5	Clermont	Clermont	AC Ajaccio	Guingamp
6	Le Havre	Le Havre	Le Havre	Troyes
7	Guingamp	Guingamp	Guingamp	Le Havre
8	Valenciennes	Valenciennes	Valenciennes	Valenciennes
9	Grenoble	Sochaux	Sochaux	Chambly
10	Sochaux	Grenoble	Grenoble	Auxerre
11	Caen	Auxerre	Nancy	Rodez
12	Chambly	Caen	Auxerre	Grenoble
13	Auxerre	Nancy	Rodez	Caen

14	Nancy	Chambly	Chambly	Nancy
15	Châteauroux	Rodez	Caen	Sochaux
16	Rodez	Châteauroux	Châteauroux	Châteauroux
17	Paris FC	Niort	Niort	Le Mans
18	Niort	Paris FC	Paris FC	Niort
19	Le Mans	Le Mans	Le Mans	Paris FC
20	Orléans	Orléans	Orléans	Orléans

I colored with green the 2 positions going directly to Ligue-1, in light green those having to play 2 rounds of play-off games (the first one against each other, then the winner plays against the Ligue-1 team ranked 18) in order to be promoted to Ligue-1. Again I used red to highlight the ranks going directly to Ligue-2 and in salmon the play-off position.

Note that when Ligue-2 was interrupted (see figure [Fig. 45.B]), we had several groups of teams being very close in terms of points won. The biggest was composed of six teams of which four teams had the same number of points -34-. Hence, it is far from being trivial to determine how the final ranking would look like. Nevertheless models agree on their predictions of the most important ranks. Interestingly, they were able to make quite similar decisions : between quatuor Lens-Troyes-Clermont-AC Ajaccio as well as the trio Niort-Paris FC-Le Mans.

Contrary to France, championships from Germany, England, Italy and Spain were resumed and teams had to play the remaining games between early June and end of July. Even though conditions were quite different than before the pause (empty stadium and health protocols, teams having positive players, different pace of competition, players who could improve their fitness during the break, etc … as if a new competition started), it is still interesting to see the difference between the actual final rankings and the ones predicted by our models 😅.

When other championships were paused, standings were :

leg	team	rank	cum_pts	cum_goals_scored	cum_goal_diff
27	Liverpool	1	79	64	47
27	Manchester City	2	57	68	39
27	Leicester	3	50	54	27
27	Chelsea	4	44	45	8
27	Manchester United	5	41	41	12
27	Tottenham	6	40	44	8
27	Sheffield Utd	7	40	29	4
27	Wolverhampton	8	39	38	6
27	Arsenal	9	37	39	3
27	Burnley	10	37	33	-6
27	Everton	11	36	36	-5
27	Southampton	12	34	34	-14
27	Crystal Palace	13	33	24	-8
27	Newcastle	14	31	24	-17
27	Brighton	15	28	32	-7
27	Bournemouth	16	26	26	-17
27	Aston Villa	17	25	34	-18
27	West Ham	18	24	32	-16
27	Watford	19	24	24	-19
27	Norwich	20	18	24	-27

leg	team	rank	cum_pts	cum_goals_scored	cum_goal_diff
27	Juventus Turin	1	66	52	28
27	Lazio Rome	2	62	62	36
27	Inter Milan	3	58	54	26
27	Atalanta Bergame	4	54	77	40
27	AS Rome	5	48	53	17
27	Naples	6	42	43	7
27	Parme	7	39	37	4
27	AC Milan	8	39	32	-3
27	Hellas Vérone	9	38	31	2
27	Cagliari	10	35	43	1
27	Bologne	11	34	38	-6
27	Sassuolo	12	33	45	-1
27	Fiorentina	13	31	33	-4
27	Torino	14	31	30	-16
27	Udinese	15	28	21	-17
27	Sampdoria Gênes	16	26	30	-18
27	Genoa	17	25	32	-19
27	Lecce	18	25	35	-25
27	SPAL	19	18	20	-25
27	Brescia	20	17	23	-27

Fig. 47.b : Left, Premier League standing after 27 legs. Right, Serie-A. Highlighted are Liverpool's and AS Roma's ranks. Note that Man. City is 2nd.

leg	team	rank	cum_pts	cum_goals_scored	cum_goal_diff
27	FC Barcelone	1	58	63	32
27	Real Madrid	2	56	49	30
27	Séville FC	3	47	39	10
27	Real Sociedad	4	46	45	12
27	Getafe	5	46	37	12
27	Atlético de Madrid	6	45	31	10
27	Valence CF	7	42	38	-1
27	Villarreal	8	38	44	6
27	Grenade CF	9	38	33	1
27	Athletic Bilbao	10	37	29	6
27	Osasuna	11	34	34	-4
27	Betis Séville	12	33	38	-5
27	Levante	13	33	32	-8
27	Alavés	14	32	29	-8
27	Valladolid	15	29	23	-10
27	Eibar	16	27	27	-14
27	Celta Vigo	17	26	22	-12
27	Majorque	18	25	28	-16
27	Leganés	19	23	21	-18
27	Espanyol Barcelone	20	20	23	-23

leg	team	rank	cum_pts	cum_goals_scored	cum_goal_diff
27	Bayern Munich	1	61	80	52
27	Borussia Dortmund	2	57	74	41
27	RB Leipzig	3	54	68	41
27	Bayer Leverkusen	4	53	52	20
27	Borussia M'Gladbach	5	52	53	19
27	Wolfsburg	6	39	36	3
27	Fribourg	7	37	35	-2
27	Schalke 04	8	37	33	-10
27	Hoffenheim	9	36	36	-11
27	FC Cologne	10	34	43	-6
27	Hertha Berlin	11	34	39	-9
27	Eintracht Francfort	12	31	44	-5
27	Augsbourg	13	30	40	-14
27	Union Berlin	14	30	32	-15
27	Mayence	15	27	36	-24
27	Fortuna Düsseldorf	16	24	29	-23
27	Werder Brême	17	21	29	-33
27	Paderborn	18	18	31	-24

Fig. 47.c : Left, Liga standing after 27 legs. Right, Bundesliga. Highlighted are Atletico de Madrid's and Eintracht Frankfurt's ranks

The predictions our models made are given below. As for Ligue-1, we will use green to highlight the champion, blue the ranks providing a direct access to the UEFA Champions League, in light purple those opening the door to an european competition (Europa League or qualifying games to UEFA champions League or Europa League), red the unfortunate positions leading to the lower level championship and salmon the play-off position. For the true result, the final cumulative number of points is specified as well as the number of points won since the season's new start. We'll use the following syntax :

TEAM (final number of points, + points won since restart)

- **Premier-League : it remained 11 games to be played which means up to 33 points to be won**

RANK	NAIVE	REGRESSION TYPE	CLASSIFICATION TYPE	RANKING TYPE	True result
1	Liverpool	Liverpool	Liverpool	Liverpool	**Liverpool (99, +20)**
2	Man. City	Man. City	Man. City	Man. City	**Man. City (81, +24)**
3	Leicester	Leicester	Leicester	Leicester	**Man. United (66, +25)**
4	Chelsea	Chelsea	Chelsea	Chelsea	**Chelsea (66, +22)**
5	Man. United	Man. United	Arsenal	Burnley	**Leicester (62, +12)**
6	Tottenham	Tottenham	Man. United	Tottenham	**Tottenham (59, +19)**
7	Sheffield Utd	Sheffield Utd	Wolverhampton	Man. United	**Wolverhampton (59, +20)**
8	Wolverhampton	Wolverhampton	Burnley	Sheffield Utd	**Arsenal (56, +19)**
9	Arsenal	Arsenal	Tottenham	Arsenal	**Sheffield Utd (54, +14)**
10	Burnley	Burnley	Sheffield Utd	Wolverhampton	**Burnley (54, +17)**
11	Everton	Everton	Everton	Newcastle	**Southampton (52, +18)**
12	Southampton	Crystal Palace	Crystal Palace	Everton	**Everton (49, +13)**
13	Crystal Palace	Southampton	Brighton	Crystal Palace	**Newcastle (44, +13)**
14	Newcastle	Brighton	Bournemouth	Bournemouth	**Crystal Palace (43, +10)**
15	Brighton	Newcastle	Southampton	West Ham	**Brighton (41, +13)**
16	Bournemouth	Bournemouth	Newcastle	Brighton	**West Ham (39, +15)**
17	Aston Villa	West Ham	Aston Villa	Southampton	**Aston Villa (35, +10)**
18	West Ham	Aston Villa	West Ham	Aston Villa	**Bournemouth (34, +8)**
19	Watford	Watford	Watford	Watford	**Watford (34, +10)**
20	Norwich	Norwich	Norwich	Norwich	**Norwich (21, +3)**

When looking at the performance post Covid-19 lockdown, it is interesting to see that Man. United and West Ham exhibit very good results relative to their ranks whilst Leicester and Sheffield United had more troubles to restart and had significant consequences on what they could have expected had the championship not been stopped.

- **Serie-A : it remained 11 games to be played which means up to 33 points to be won**

RANK	NAIVE	REGRESSION TYPE	CLASSIFICATION TYPE	RANKING TYPE	True result
1	Juventus Turin	Juventus Turin	Juventus Turin	Juventus Turin	**Juventus Turin (83, +17)**
2	Lazio Rome	Lazio Rome	Inter Milan	Inter Milan	**Inter Milan (82, +24)**
3	Inter Milan	Inter Milan	Lazio Rome	Lazio Rome	**Atalanta Bergame (78, +24)**
4	Atalanta Bergame	Atalanta Bergame	AS Rome	Atalanta Bergame	**Lazio Rome (78, +16)**
5	AS Rome	AS Rome	Atalanta Bergame	AS Rome	**AS Rome (70, +22)**
6	Naples	Naples	Naples	Bologne	**AC Milan (66, +27)**
7	Parme	Parme	Hellas Vérone	Parme	**Naples (62, +20)**
8	AC Milan	Hellas Vérone	Parme	Hellas Vérone	**Sassuolo (51, +18)**
9	Hellas Vérone	AC Milan	AC Milan	Naples	**Hellas Vérone (49, +11)**
10	Cagliari	Cagliari	Bologne	Cagliari	**Fiorentina (49, +18)**
11	Bologne	Bologne	Cagliari	AC Milan	**Parme (49, +10)**
12	Sassuolo	Sassuolo	Sassuolo	Sassuolo	**Bologne (47, +13)**
13	Torino	Fiorentina	Udinese	Torino	**Udinese (45, +17)**
14	Fiorentina	Torino	Fiorentina	Fiorentina	**Cagliari (45, +10)**
15	Udinese	Udinese	Torino	Udinese	**Sampdoria Gênes (42, +16)**
16	Sampdoria Gênes	Sampdoria Gênes	Genoa	Genoa	**Torino (40, +9)**
17	Genoa	Genoa	Sampdoria Gênes	Brescia	**Genoa (39, +14)**

18	Lecce	Lecce	Lecce	Lecce	**Lecce (35, +10)**
19	SPAL	SPAL	Brescia	Sampdoria Gênes	**Brescia (25, +8)**
20	Brescia	Brescia	SPAL	SPAL	**SPAL (20, +2)**

In Italy, the break was beneficial to Genoa, Sampdoria Gênes and AC Milan in their respective races. On the other hand, Lazio Rome, Hellas Vérone and Cagliari lost lots of points to their immediate competitors. Nonetheless it didn't impact the belonging to the groups of high values

- **La Liga : it remained 11 games to be played which means up to 33 points to be won**

RANK	NAIVE	REGRESSION TYPE	CLASSIFICATION TYPE	RANKING TYPE	True result
1	FC Barcelone	FC Barcelone	FC Barcelone	Real Madrid	**Real Madrid (87, +31)**
2	Real Madrid	Real Madrid	Real Madrid	FC Barcelone	**FC Barcelone (82, +24)**
3	Séville FC	Real Sociedad	Real Sociedad	Séville FC	**Atlético de Madrid (70, +25)**
4	Getafe	Séville FC	Getafe	Atlético de Madrid	**Séville FC (70, +23)**
5	Real Sociedad	Getafe	Villarreal	Real Sociedad	**Villarreal (60, +22)**
6	Atlético de Madrid	Atlético de Madrid	Séville FC	Getafe	**Real Sociedad (56, +10)**
7	Valence CF	Valence CF	Atlético de Madrid	Villarreal	**Grenade CF (56, +18)**
8	Grenade CF	Villarreal	Grenade CF	Valence CF	**Getafe (54, +8)**
9	Villarreal	Grenade CF	Valence CF	Athletic Bilbao	**Valence CF (53, +11)**
10	Athletic Bilbao	Athletic Bilbao	Athletic Bilbao	Grenade CF	**Osasuna (52, +18)**
11	Osasuna	Osasuna	Bétis Séville	Osasuna	**Athletic Bilbao (51, +14)**
12	Levante	Bétis Séville	Levante	Levante	**Levante (49, +16)**
13	Bétis Séville	Levante	Osasuna	Alavés	**Valladolid (42, +13)**
14	Alavés	Alavés	Alavés	Bétis Séville	**Eibar (42, +15)**

15	Valladolid	Valladolid	Valladolid	Celta Vigo	**Bétis Séville (41, +8)**
16	Eibar	Celta Vigo	Majorque	Valladolid	**Alavés (39, +7)**
17	Celta Vigo	Eibar	Celta Vigo	Majorque	**Celta Vigo (37, +11)**
18	Majorque	Majorque	Eibar	Eibar	**Leganés (36, +13)**
19	Leganés	Leganés	Espanyol Barcelone	Espanyol Barcelone	**Majorque (33, +8)**
20	Espanyol Barcelone	Espanyol Barcelone	Leganés	Leganés	**Espanyol Barcelone (25, +5)**

With 31 points out of 33 possible, Real Madrid had a very strong restart/finish allowing them to beat FC Barcelona in their race to the title. Another team for which the break turned out to be positive is Atlético Madrid. Indeed, following it, Atlético Madrid had the same pace as the top 4 allowing the team to get from rank 8 (no qualification to an european competition) to a place on the podium (hence a ticket to the next UEFA Champions League). On the other end of the ranking, Eibar found a second wind to save its belonging to La Liga since it had only 2 points more than the first team in the relegation area when the championship was interrupted. Leganés performed well too, but it was just too short to save itself.

Real Sociedad and Getafe were the teams having lost the most with the restart since they lost 8 to 15 points to their most dangerous competitor. As a consequence they were prevented from taking to the European competitions they could have expected to play the next season. The third team for which the new start went bad is Valence CF. Indeed, having lost 8 points to Grenade and 11 points to Villareal, it couldn't secure a rank to play the Europa League.

- **Bundesliga : it remained 7 games to be played which means up to 21 points to be won**

RANK	NAIVE	REGRESSION TYPE	CLASSIFICATION TYPE	RANKING TYPE	True result
1	Bayern Munich	Bayern Munich	RB Leipzig	Bayern Munich	**Bayern Munich (82, +21)**
2	Borussia Dortmund	Borussia Dortmund	Bayern Munich	Borussia Dortmund	**Borussia Dortmund (69, +12)**
3	RB Leipzig	RB Leipzig	Borussia Dortmund	Borussia M'Gladbach	**RB Leipzig (66, +12)**
4	Bayer Leverkusen	Bayer Leverkusen	Borussia M'Gladbach	RB Leipzig	**Borussia M'Gladbach (65, +13)**

5	Borussia M'Gladbach	Borussia M'Gladbach	Bayer Leverkusen	Bayer Leverkusen	**Bayer Leverkusen (63, +10)**
6	Wolfsburg	Wolfsburg	Wolfsburg	Wolfsburg	**Hoffenheim (52, +16)**
7	Schalke 04	Fribourg	Schalke 04	Fribourg	**Wolfsburg (49, +10)**
8	Fribourg	Schalke 04	Fribourg	FC Cologne	**Fribourg (48, +11)**
9	Hoffenheim	Hoffenheim	Eintracht Francfort	Hertha Berlin	**Eintracht Francfort (45, +14)**
10	Hertha Berlin	FC Cologne	FC Cologne	Schalke 04	**Hertha Berlin (41, +7)**
11	FC Cologne	Hertha Berlin	Union Berlin	Eintracht Francfort	**Union Berlin (41, +11)**
12	Eintracht Francfort	Eintracht Francfort	Hoffenheim	Hoffenheim	**Schalke 04 (39, +5)**
13	Augsbourg	Augsbourg	Augsbourg	Union Berlin	**Mayence (37, +10)**
14	Union Berlin	Union Berlin	Hertha Berlin	Augsbourg	**FC Cologne (36, +2)**
15	Mayence	Mayence	Mayence	Mayence	**Augsbourg (36, +6)**
16	Fortuna Düsseldorf	Fortuna Düsseldorf	Fortuna Düsseldorf	Fortuna Düsseldorf	**Werder Brême (31, +10)**
17	Werder Brême	Werder Brême	Werder Brême	Paderborn	**Fortuna Düsseldorf (30, +6)**
18	Paderborn	Paderborn	Paderborn	Werder Brême	**Paderborn (20, +2)**

Bayern Munich was perfect in their ability to restart. Indeed it won all games … and the UEFA Champions League. With 16 points out of 21, Hoffenheim won 5 points (resp. 6 points) more than Fribourg (resp. Wolfsburg) in order to save a position for an European competition. On the other end of the ranking, the break was also beneficial to Werder Brême which won 4 points more than Fortuna Düsseldorf and won the right to a last fight to save its belonging to Bundesliga. It has to be noted that Fortuna Düsseldorf had the same pace as Augsboug and was better than Schalke 04 and FC Cologne. These last 2 teams together with Hertha Berlin were among the flop 3 from the new start regarding to what the ranks and performance until the championship was interrupted.

Miscellaneous

As an additional way to achieve my objective to predict final ranking, I wanted to predict the number of goals a team would score at its next game. Since it turned out not to be conclusive, I won't develop it. The funny fact didn't know was that around 30% of the time a team would score 1 goal in a given game.

Conclusion

In no way, this project is aimed to be used for any claim. It was just a project designed by myself to practice on a topic completely different from my daily tasks.

That being said, the fact that the quality of predicted ranks are not as good as expected at the start is a good thing for the interest in soccer even if results about the content of groups of high interest are quite decent. Indeed, the uncertainty about games' outcome makes it an exciting sport to follow and argue with friends (and other fans). When looking at how close some teams ended their respective championships (less than 2 points difference), I'm not sure that any model could have made the right guess given that there remained 11 games to be played (7 for Bundesliga). Moreover, as seen for the 2019-2020 final predictions for La Liga, Premier-League, Bundesliga and Serie-A, some teams performed much better than before the interruption while others seemed unable to get back to their pre Covid-19 pace. Is it due to the different Covid sanitary measures, the ability teams to keep players fit during the lockdown without the possibility of doing collective training sessions to practice some strategies or would it have happened anyway regardless of Covid-19 (kind of money times effect) ?

Throughout the project presented in this article, it allowed me to cover most of a data scientist's spectrum skills :

- defining a problem
- getting the data
- cleaning and exploring the data in order to get insights
- data visualization
- data engineering, building models and picking the best ones
- translating the pipeline from notebooks to scripts and building an app
- storytelling the journey (not by best skill, but working on it :))

What I haven't covered here is model deployment and monitoring. One thing that could be improved is data and model versioning when experimenting.

Side note

I consider my notebooks as rough drafts before turning to .py script with content I deem sufficient; so notebooks are not designed to be user-friendly.

Best practices require providing docstrings and variable typing. As it is personal and has been written over the course of several months (when I had sufficient free time and

motivation to go on with the project) and is not meant to be used widely by the public or go to production, it didn't seem to me to be as important.