Predicting the Outcome of Primeira Liga's Football Matches

Miguel Bandeira

1. Aim

The aim of this project is to explore data from a public dataset regarding European football previous matches and also to develop a model that can predict the outcome of matches with an accuracy of more than 46% (since the home team wins about 46% of the time).

2. Data Description

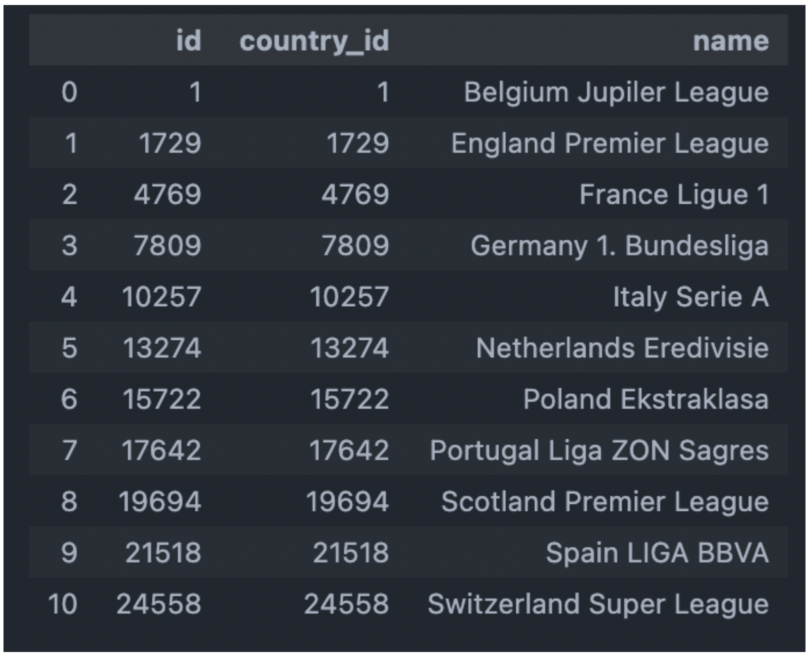
The dataset is publicly available online, and contains:

- +25,000 matches
- +10,000 players
- 11 European Countries with their lead championship
- Seasons 2008 to 2016
- Players and Teams' attributes* sourced from EA Sports' FIFA video game series, including the weekly updates
- Team line up with squad formation (X, Y coordinates)
- Betting odds from up to 10 providers
- Detailed match events (goal types, possession, corner, cross, fouls, cards etc…) for +10,000 matches

The dataset contains 7 tables: Match, Player, Player-Attributes, Teams, Team-Attributes, League, and Country.
The shape of these tables is provided in the table below.

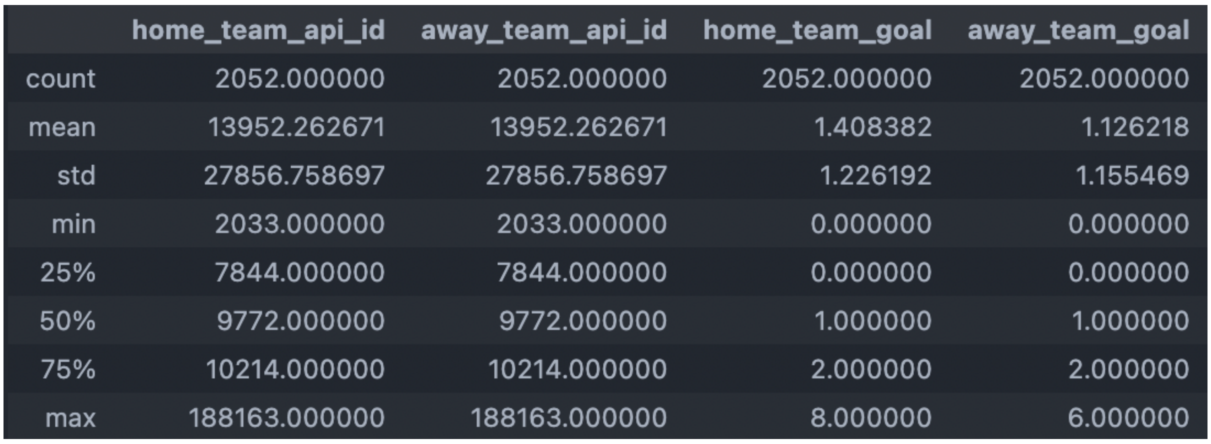
	Count	**Dimensions**
Match	25979	115
Teams	299	5
Team_Attributes	1458	25
League	11	3
Player	11060	7
Player_Attributes	183978	42
Country	11	2

Here, the active reader would have noticed that there are only 299 teams but 1458 Team Attribute rows, as well as 11060 players but 183978 player attributes. This is because the data provider has provided the attributes for some teams and players over a span of several years and.
The leagues present in the dataset are:



	id	country_id	name
0	1	1	Belgium Jupiler League
1	1729	1729	England Premier League
2	4769	4769	France Ligue 1
3	7809	7809	Germany 1. Bundesliga
4	10257	10257	Italy Serie A
5	13274	13274	Netherlands Eredivisie
6	15722	15722	Poland Ekstraklasa
7	17642	17642	Portugal Liga ZON Sagres
8	19694	19694	Scotland Premier League
9	21518	21518	Spain LIGA BBVA
10	24558	24558	Switzerland Super League

For the purpose of this project, we will only focus on two tables: Matches and Team-Attributes and on the Portuguese league.

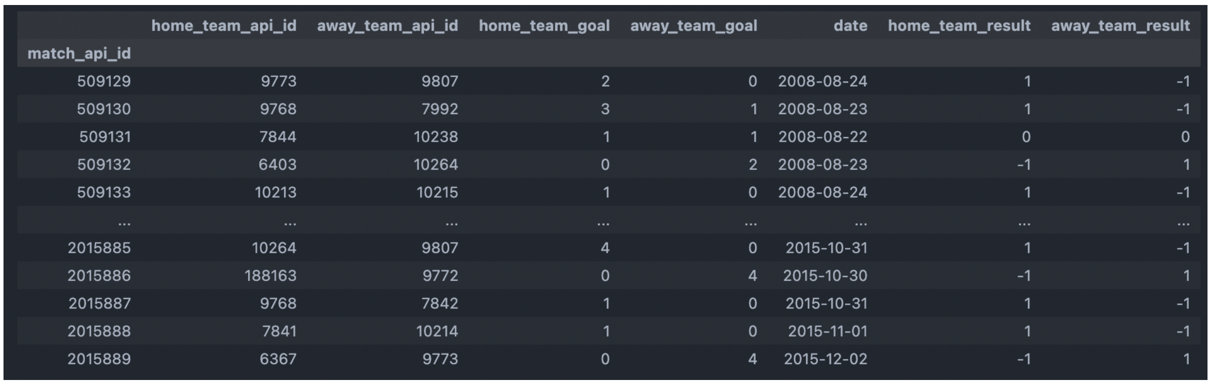


	home_team_api_id	away_team_api_id	home_team_goal	away_team_goal
count	2052.000000	2052.000000	2052.000000	2052.000000
mean	13952.262671	13952.262671	1.408382	1.126218
std	27856.758697	27856.758697	1.226192	1.155469
min	2033.000000	2033.000000	0.000000	0.000000
25%	7844.000000	7844.000000	0.000000	0.000000
50%	9772.000000	9772.000000	1.000000	1.000000
75%	10214.000000	10214.000000	2.000000	2.000000
max	188163.000000	188163.000000	8.000000	6.000000

Finally, we can see that the average goals scored by the home team is higher than the average of the away team, confirming to us that the home teams tend to perform better.

3. Match table

The Match table is made of 115 columns, most of them are either not useful for our purpose or have a lot of null values, so the first step was to only load that columns that would be used for the model, and those are: *match_api_id, home_team_api_id, away_team_api_id, home_team_goal, away_team_goal, date*. We achieve this simplified table by only loading those columns through a sql query. Also, as mentioned before, I focused on the Portuguese league, which *id* can be found in the table League. These columns were selected because they do not have null values and they would allow me to come up with new features for the model. The first two columns I created were the result of the match in perspective of the home team. For instance, if the home team had won the match, the value of the *home_team_result* would be 1. If the away team won, the value would be -1 and in case of a draw, a 0. Same logic was applied for *away_team_result* but in reverse. The Match table with these changes looked like the following:

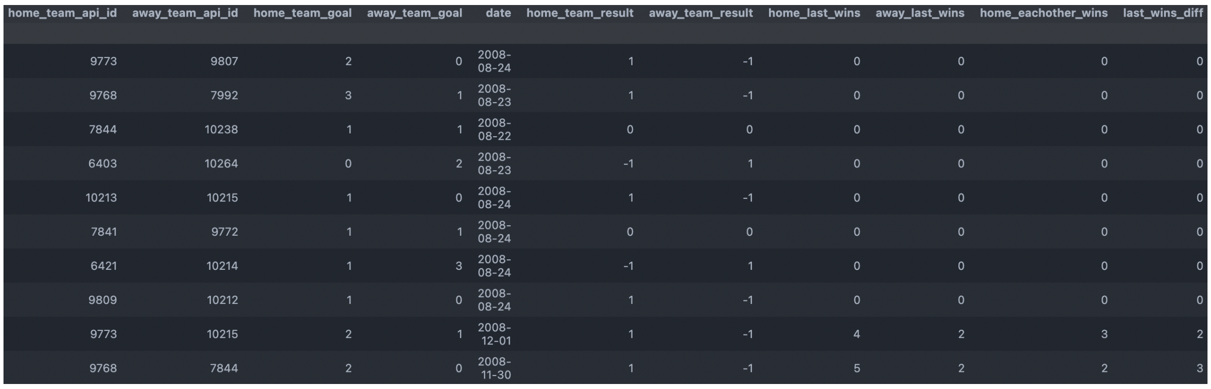


	home_team_api_id	away_team_api_id	home_team_goal	away_team_goal	date	home_team_result	away_team_result
match_api_id							
509129	9773	9807	2	0	2008-08-24	1	-1
509130	9768	7992	3	1	2008-08-23	1	-1
509131	7844	10238	1	1	2008-08-22	0	0
509132	6403	10264	0	2	2008-08-23	-1	1
509133	10213	10215	1	0	2008-08-24	1	-1
...	...	...	...	...	...	...	...
2015885	10264	9807	4	0	2015-10-31	1	-1
2015886	188163	9772	0	4	2015-10-30	-1	1
2015887	9768	7842	1	0	2015-10-31	1	-1
2015888	7841	10214	1	0	2015-11-01	1	-1
2015889	6367	9773	0	4	2015-12-02	-1	1

Next relevant information worth analyzing was the results of the last 10 matches of both the home team and away team. I decided to use the last 10 matches because I feel like it shows a somewhat accurate representation of the team's momentum. Fewer matches would be too little, more matches would be too outdated.

After that, it was necessary to compare the difference between the results of the last matches of both teams, in order to create some sort of index of their performance. This was a basic calculation achieved by subtracting the numbers of wins of the home team with the wins of the away team and store it in a new column called *last_wins_diff.*

Finally, I had to calculate and evaluate the previous matches between both teams in the perspective of the home team.
The final result of the table Match is:



home_team_api_id	away_team_api_id	home_team_goal	away_team_goal	date	home_team_result	away_team_result	home_last_wins	away_last_wins	home_eachother_wins	last_wins_diff
9773	9807	2	0	2008-08-24	1	-1	0	0	0	0
9768	7992	3	1	2008-08-23	1	-1	0	0	0	0
7844	10238	1	1	2008-08-22	0	0	0	0	0	0
6403	10264	0	2	2008-08-23	-1	1	0	0	0	0
10213	10215	1	0	2008-08-24	1	-1	0	0	0	0
7841	9772	1	1	2008-08-24	0	0	0	0	0	0
6421	10214	1	3	2008-08-24	-1	1	0	0	0	0
9809	10212	1	0	2008-08-24	1	-1	0	0	0	0
9773	10215	2	1	2008-12-01	1	-1	4	2	3	2
9768	7844	2	0	2008-11-30	1	-1	5	2	2	3

3.2 Prediction results

After processing the data in a way that would be suitable for the intended task, only thing left to do was to use various classification and regression models to come up with the outcome given two different teams and their respective attributes.
Since I had turned the problem into a prediction for the home team, the target variable would obviously be the home team result.
Moreover, the features used were the difference between the results of the last 10 matches of each team (*last_wins_diff)* and the result of the last 10 matches between both teams (*home_eachother_wins*).

I tested several classifications models, including K-Nearest Neighbors, Logistic Regression, Decision Tree, Random Forest, Neural Net, AdaBoost and Naive Bayes with a random split of 80% of the matches as the training set and 20% as the testing set.

In the table below are present accuracy of each model.

K-Nearest Neighbors	~ 47%
Logistic Regression	~ 51%
Decision Tree	~ 49%
Random Forest	~ 44%
Neural Net	~ 50%
AdaBoost	~ 50%
Naive Bayes	~ 50%

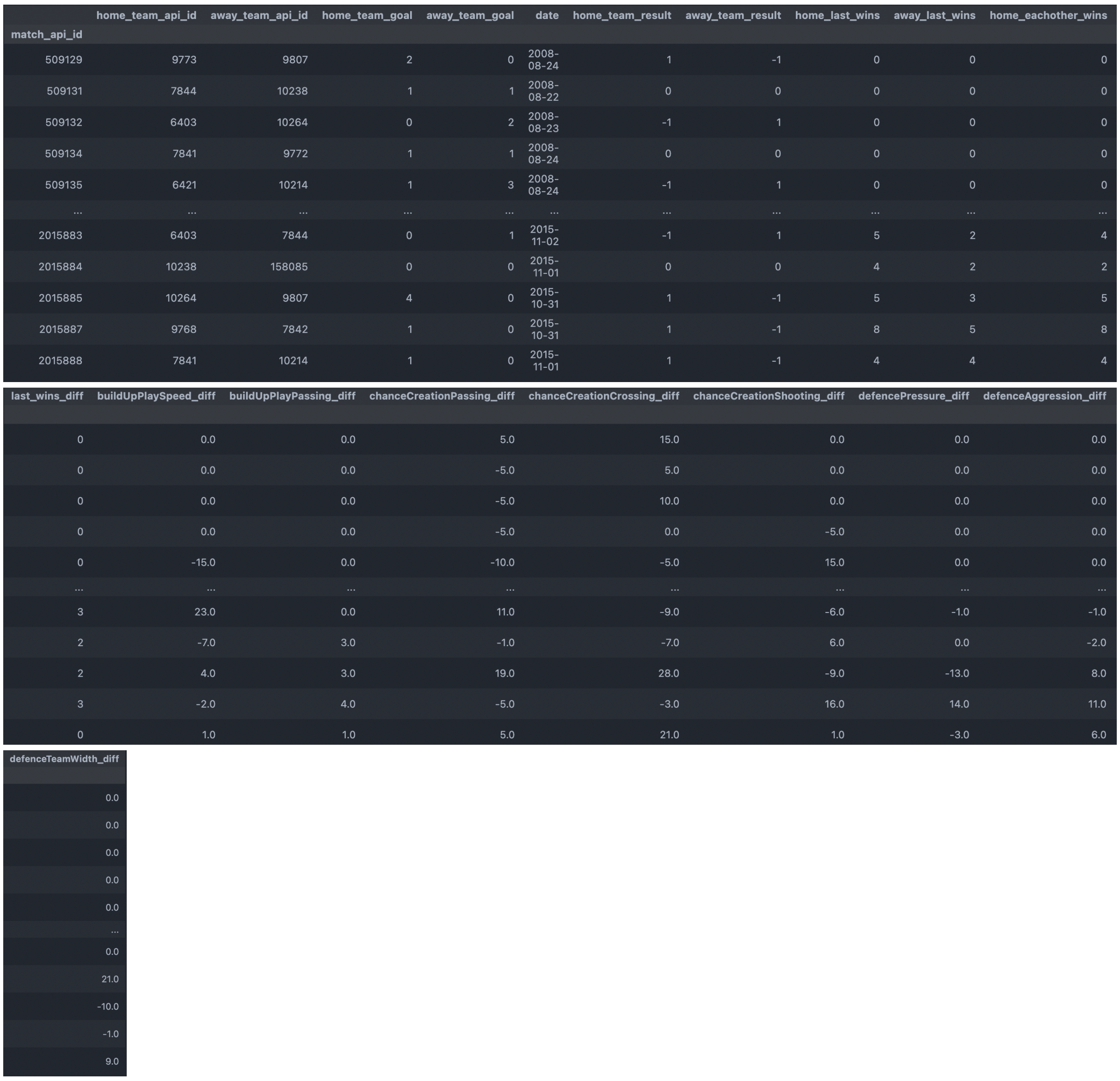
After running the models, the results were promising and very close to each other, with the Logistic Regression being the most promising with a whopping 51% accuracy score, beating our goal of 46%. However, there was another hypothesis that needed to be tested.

4. Team_Attributes table

Altough we results were already above our goal, I wanted to use the team attributes and compare results.
For this, I selected only the following columns: *date, buildUpPlaySpeed, buildUpPlayPassing, chanceCreationPassing, chanceCreationCrossing, chanceCreationShooting, defencePressure, defenceAggression and defenceTeamWidth* because they were the only Nominal ones, except for the case of *buildUpPlayDribbling* that was also nominal but had 969 null values, so it did not classify.
So, for this step, I created two auxiliary functions that grabbed the most recent team attributes, since each team had a set of multiple attributes that changed throughout time.
After that, I created new columns, one for each attribute, that calculated the difference between the same attribute of each team, so we could use these differences in our models. Essentially:

$$attribute\ Difference = home\ team\ Attribute - away\ \ team\ Attribute$$

Lastly, I merged this table with the Match table calculated before, so I could also have the matches results, in order to train my model. Below you can find the result:



match_api_id	home_team_api_id	away_team_api_id	home_team_goal	away_team_goal	date	home_team_result	away_team_result	home_last_wins	away_last_wins	home_eachother_wins
509129	9773	9807	2	0	2008-08-24	1	-1	0	0	0
509131	7844	10238	1	1	2008-08-22	0	0	0	0	0
509132	6403	10264	0	2	2008-08-23	-1	1	0	0	0
509134	7841	9772	1	1	2008-08-24	0	0	0	0	0
509135	6421	10214	1	3	2008-08-24	-1	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...
2015883	6403	7844	0	1	2015-11-02	-1	1	5	2	4
2015884	10238	158085	0	0	2015-11-01	0	0	4	2	2
2015885	10264	9807	4	0	2015-10-31	1	-1	5	3	5
2015887	9768	7842	1	0	2015-10-31	1	-1	8	5	8
2015888	7841	10214	1	0	2015-11-01	1	-1	4	4	4

match_api_id	last_wins_diff	buildUpPlaySpeed_diff	buildUpPlayPassing_diff	chanceCreationPassing_diff	chanceCreationCrossing_diff	chanceCreationShooting_diff	defencePressure_diff	defenceAggression_diff	defenceTeamWidth_diff
509129	0	0.0	0.0	5.0	15.0	0.0	0.0	0.0	0.0
509131	0	0.0	0.0	-5.0	5.0	0.0	0.0	0.0	0.0
509132	0	0.0	0.0	-5.0	10.0	0.0	0.0	0.0	0.0
509134	0	0.0	0.0	-5.0	0.0	-5.0	0.0	0.0	0.0
509135	0	-15.0	0.0	-10.0	-5.0	15.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...
2015883	3	23.0	0.0	11.0	-9.0	-6.0	-1.0	-1.0	0.0
2015884	2	-7.0	3.0	-1.0	-7.0	6.0	0.0	-2.0	21.0
2015885	2	4.0	3.0	19.0	28.0	-9.0	-13.0	8.0	-10.0
2015887	3	-2.0	4.0	-5.0	-3.0	16.0	14.0	11.0	-1.0
2015888	0	1.0	1.0	5.0	21.0	1.0	-3.0	6.0	9.0

Now, all was left to do was to run the same classifiers and compare the results. The target was, again, the home team result and the features were the columns with the difference between the attributes of both teams. So, the features list was *buildUpPlaySpeed_diff, buildUpPlayPassing_diff, chanceCreationPassing_diff, chanceCreationCrossing_diff, chanceCreationShooting_diff, defencePressure_diff, defenceAggression_diff, defenceTeamWidth_diff*. Also using again an 80-20 split between the training set and the testing set. These were the obtained results:

K-Nearest Neighbors	~ 45%
Logistic Regression	~ 48%
Decision Tree	~ 50%
Random Forest	~ 44%

Neural Net	~ 45%
AdaBoost	~ 50%
Naive Bayes	~ 50%

As we can see, I got slightly worst results than with the previous model so this hypothesis was refuted.

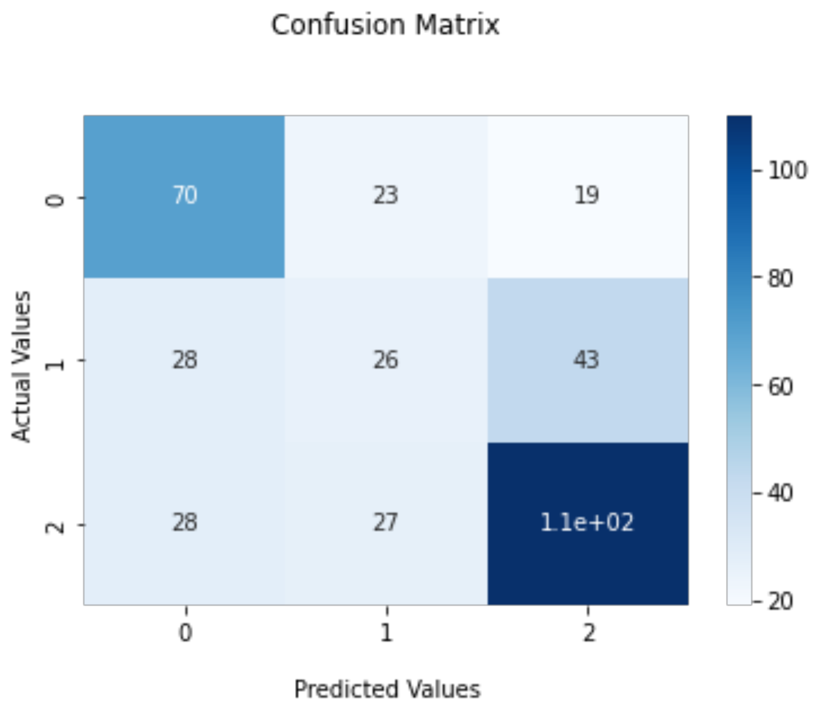
5. Final Model

Because the previous tests were, overall, around or above the 46% accuracy mark, I decided to try one last hypothesis: merging both my Matches and Team_Attributes tables. This way I would have in consideration not only the results of matches but also the differences between each team's attributes, resulting in a more complex set of features. Also, because I had already merged both tables in the previous section, this was an effortless test and very worth a shot.
So, same procedure as the one in the previous section, some target (*home_team_result)* and the features set being a combination of the ones of both testes hypothesis (*buildUpPlaySpeed_diff, buildUpPlayPassing_diff, chanceCreationPassing_diff, chanceCreationCrossing_diff, chanceCreationShooting_diff, defencePressure_diff, defenceAggression_diff, defenceTeamWidth_diff, last_wins_diff, home_eachother_wins).*

These were the obtained results:

K-Nearest Neighbors	~ 47%
Logistic Regression	~ 55%
Decision Tree	~ 46%
Random Forest	~ 48%
Neural Net	~ 48%
AdaBoost	~ 53%
Naive Bayes	~ 50%

As we can see, the overall results were better, with the Logistic Regression being the most accurate, with a whopping 55%, followed by Naive Bayes.



Looking at the confusion matrix above, we conclude that the model has trouble predicting a draw, confusing it, plenty of times, with wins or losses.

6. Clarification

Since I am delivering the project late, I just wanted to clarify that none of the code was improved based on the feedback given by the professor during the presentations as that would be unfair. All the additions made to the code were to help produce the report, since it was only half done by the time the delivery was planned. With this being said, I will also keep the original jupyter notebook file that was submitted before the class in addition to the final notebook, so that the professor can confirm what is being stated here.
The final file to take into consideration is called final_project.ipynb.