CSGO odds forecasting

(This report must be rendered with MathJax, otherwise, it loses every single mathematical explanation).

Goal

The goal of this project is to predict the outcomes of competitive games by using some features available online that gives some information about the probability of a team to win a given match.

Typical scenario

Team 1 encounters Team 2 on a best-of-1 match. A bookmaker provide odds for the Team i to win, in the format $o_i = x$ where $x \in [1, \infty[$. Meaning that if one bets $b = 1€$ on Team 1 with odds $o_1 = 2.4$ then he earns:

$$G_1 = \begin{cases} b.o_1 - 1 = 1.4€ & if\ Team\ 1\ wins \\ -b = -1€ & otherwise \end{cases}$$

The odds given by the bookmaker imply the probabilities of teams to win the match. If $o_1 = 2.4$ then the implied probability (for the team 1 to win the game) should be $p_1 = \frac{1}{2.4} \approx 42\%$. In fact, it would be like this if the odds were fair, but they are not, so the bookmaker makes a profit. If the odds were fair (*ie.* $p_1 = 1/o_1$) we would have the following gain expectancy:

$$\begin{aligned} \mathbb{E}[G_1] &= p_1(b.o_1 - 1) + (1 - p_1)(-b) \\ &= b - p_1 - b + p_1 \\ &= 0 \end{aligned}$$

So the odds are set unfair by the bookmaker so that $1/o_1 + 1/o_2 = 1 + margin > 1$.
To obtain the implied probability we just have to normalize:

$$p_1 = \frac{\frac{1}{o_1}}{\frac{1}{o_1} + \frac{1}{o_2}},\ p_2 = \frac{\frac{1}{o_2}}{\frac{1}{o_1} + \frac{1}{o_2}},\ p_1 + p_2 = 1$$

As a consequence, without further knowledge over the outcome of the match, the gain expectancy will be strictly negative, proportional to the margin of the bookmaker. If people make bet, it is because they think they have an edge over the bookmaker, so they can "beat the odds".

What we want

Let's denote $\bar{Y} \hookrightarrow Ber(\bar{p}_1)$ the true (unknown but observed) output of the match (equal to 1 if Team 1 wins, and to 0 if Team 2 wins). We want to estimate $\hat{Y} \hookrightarrow Ber(\hat{p}_1)$ with enough precision so we can beat the bookmaker margin.
If we know surely $\hat{p}_1$ we would have the following expectancy of betting on Team 1:

$$\begin{aligned}\mathbb{E}[G_1] &= \hat{p}_1(b.o_1 - 1) + (1 - \hat{p}_1)(-b) \\ &= b.\hat{p}_1 o_1 - \hat{p}_1 - b + \hat{p}_1 \\ &= b(\hat{p}_1 o_1 - 1)\end{aligned}$$

And a simple betting rule would be :

$$\mathbb{E}[G_1] > 0 \iff \hat{p}_1 > 1/o_1$$

Similarly we derive the betting rule on Team 2, we bet if $\hat{p}_2 > 1/o_2$.

NB: Because of the bookmaker margin both conditions may be false together, thus, sometimes there will be matches where the best solution is simply not to bet at all.

Features

To obtain this prediction model we have to calculate some features in order to have an edge over the bookmaker.
Which features influences the outcome of the match ?

In theory

1. *The ranking of the teams* : basically a kind of power ranking of teams, some are just better than others in every field of the game.
2. *The fitness of the teams* : are they on a winning streak, a losing streak ? Did they recently outperformed/underperformed other teams ?
3. *The antecedents of the encounter* : if Team 1 never lost to Team 2 in 20 matches, this could be a key information to take into account.
4. *The map on which the game is played* : like in Tennis, Nadal is better on clay court while Federer is better on hard.

5. *The fitness of the players* : the game is played in teams of five, the performance of the team depends highly of the performance of every single player.
6. *The odds given by the bookmaker* : well it does not really *influence* the outcome of the game, but since the bookmakers make small errors, their predictions are relevant and they should be used for predictions.

The implementation

Information may be hard to find and to aggregate, thus, the only features used up to now are :

- The ranking of the teams,
- The fitness of the teams,
- The antecedents of the encounter (TODO),
- The odds.

The sources of information are :

- The API of pandasport.co
- The odds of the bookmaker CSGOLounge

Team rankings

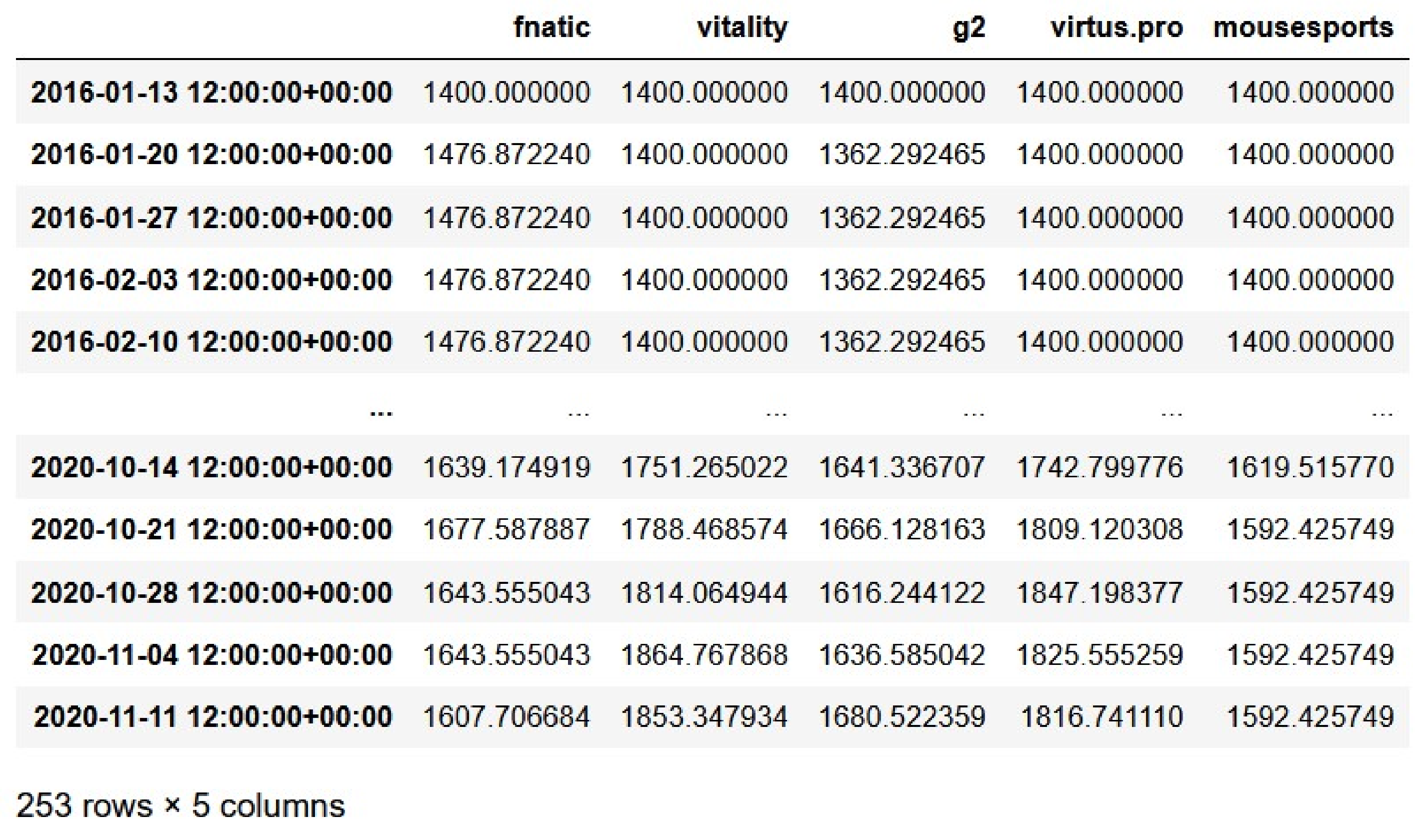
The pandasport API provides us a complete historical data recording almost every CSGO competitive game played since 2016. This is th raw data that we will use to compute the rankings.

NB : We could have choses an existing power ranking made by websites such as HLTV or Gosugamers, but they are including neither every team nor cover a large enough time window.

A word on elo ranking

- **ELO ranking** is a ranking system, broadly use in many games, including chess. In the ELO rating system, the difference in the ratings between two players (here teams) serves as a predictor of the outcome of a match. In this way we can estimate how much the ouput of the match deviates from the theory, and update the scores accordingly.
- Basically, here, **we set the elo origin of teams at 1400** in 2016, and then we compute their elo variation each week. This way we have a good estimation of how powerful a team at week T is, based on its ELO score at week $T-1$.

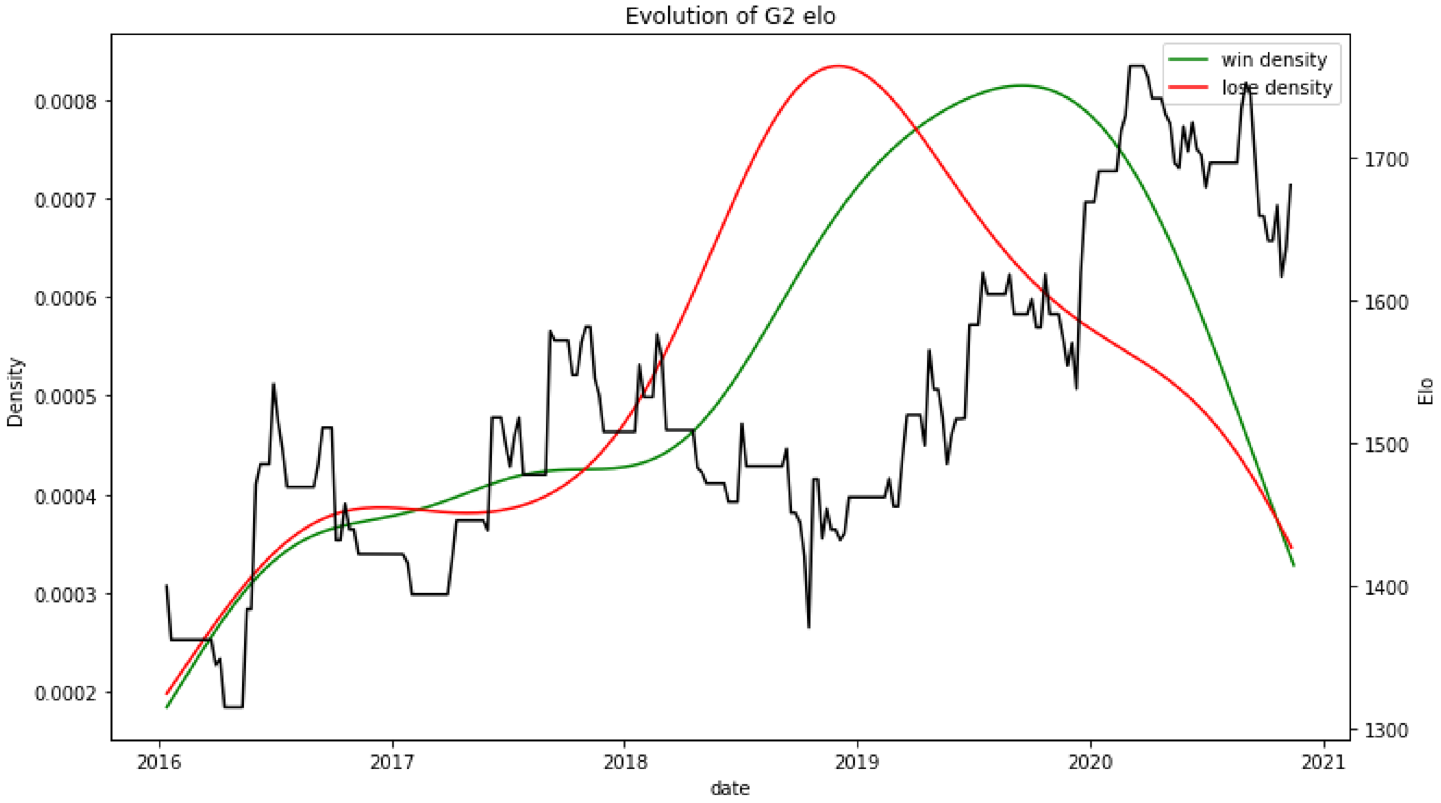
We obtain a ranking in this form



	fnatic	vitality	g2	virtus.pro	mousesports
2016-01-13 12:00:00+00:00	1400.000000	1400.000000	1400.000000	1400.000000	1400.000000
2016-01-20 12:00:00+00:00	1476.872240	1400.000000	1362.292465	1400.000000	1400.000000
2016-01-27 12:00:00+00:00	1476.872240	1400.000000	1362.292465	1400.000000	1400.000000
2016-02-03 12:00:00+00:00	1476.872240	1400.000000	1362.292465	1400.000000	1400.000000
2016-02-10 12:00:00+00:00	1476.872240	1400.000000	1362.292465	1400.000000	1400.000000
...	...	...	...	...	...
2020-10-14 12:00:00+00:00	1639.174919	1751.265022	1641.336707	1742.799776	1619.515770
2020-10-21 12:00:00+00:00	1677.587887	1788.468574	1666.128163	1809.120308	1592.425749
2020-10-28 12:00:00+00:00	1643.555043	1814.064944	1616.244122	1847.198377	1592.425749
2020-11-04 12:00:00+00:00	1643.555043	1864.767868	1636.585042	1825.555259	1592.425749
2020-11-11 12:00:00+00:00	1607.706684	1853.347934	1680.522359	1816.741110	1592.425749

253 rows × 5 columns

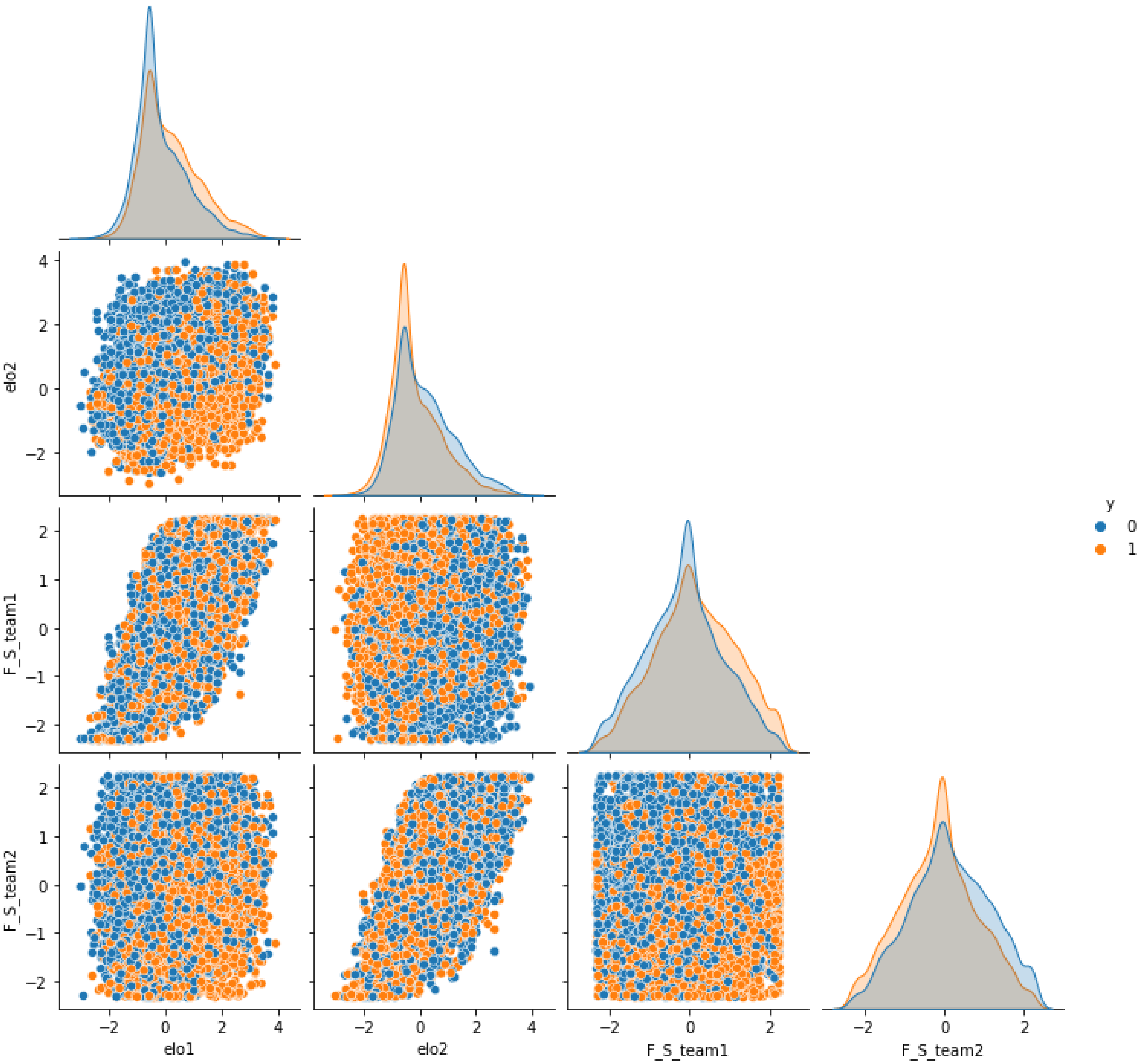
For example, here is the evolution of the ELO ranting of the team G2 esports.



Teams fitness

This score is based on the 5 most recents matches of a team, it is like an ELO ranking but just with a finite numbers of records in the past. This feature is likely to take into account the fact that teams in winning streak are more likely to perform well than teams in losing streak.

Below this is a pairplot of the 2 main features (ocoputed on both teams). Recall that $y = 1$ means the team 1 has won the match and $y = 0$ means that team 2 won the match. We can observe a significant difference between the 1's and the 0's distributions.



NB : features have been symmetrized (because team 1 and team 2 can be swap, and this can generate "new samples"). This is why in distribution the scores between team 1 and team 2 are perfectly symmetric.

This score is based on the history of the two teams that are going to get involved together in a competitive match. We use again an ELO-like computation to derive this score.

Details can be found in the feature engineering notebook.

Odds

This feature is very important since we can get a lot of information from the (unknown) odds computation of a bookmaker. Here we are using the data from CSGOLounge, what's specific with this odd system is the fact that it is only a crowd-wisdom odd sytem, as there is no prior used by the website. It only depends on the amount already bet on each teams by the gamblers.

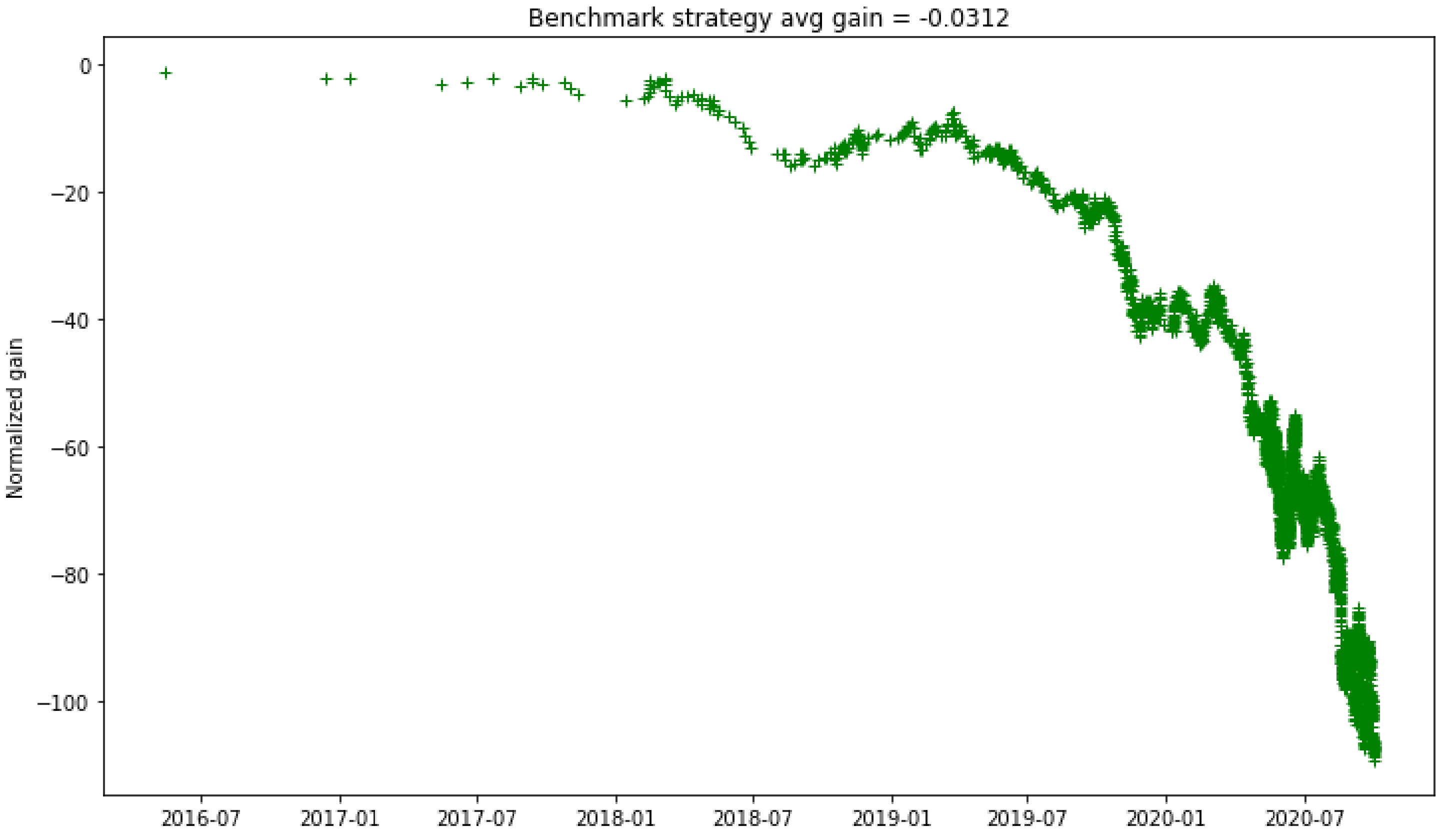
The whole scrapping and storage process are meant to work with a Django server, which also provides a graphical interface for the project.

Deriving a winning strategy

Now that we have our features, the goal is to quantify their performance in predicting the outcomes of the games. More details and code to be found here.

What happens if we follow the bookmaker

Nothing good at all, because of the bookmaker margin we will enentually end up with a negative average gain.

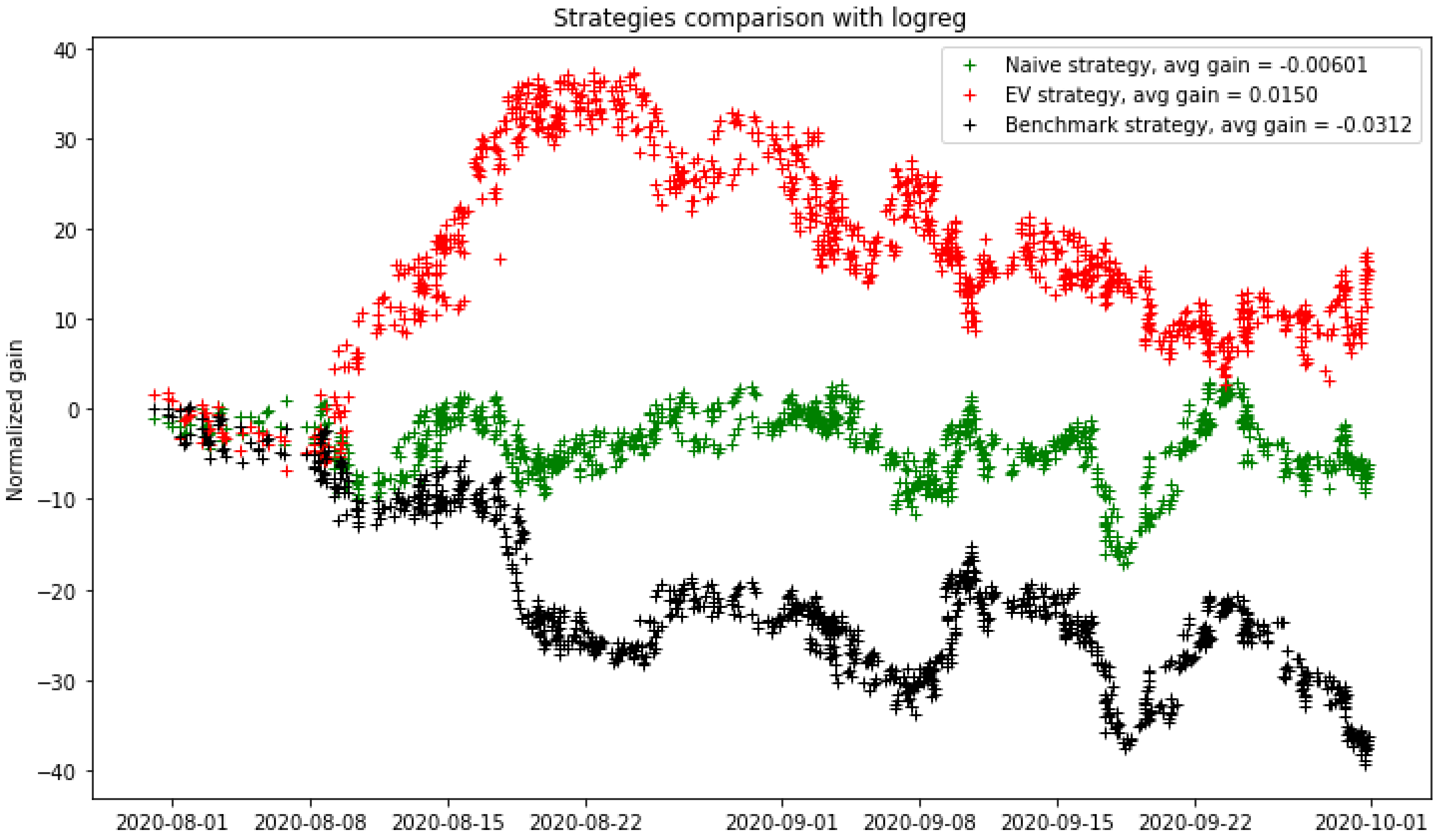


No surprises ! This is very bad, our very goal is to do better than that.

Logistic regression

Now we wan't to compare a model based on our features, with this benchmark 'follow the bookmaker' strategy.

We chose the logistic regression because of it's good property to predict well the probabilities and not just the targets. (See https://scikit-learn.org/stable/modules/calibration.html)
Below are the result of the simulation, where we trained the model on 70% of the first games in our database, then we evaluated the profit on the remaining 30%.



Our simple model makes a positive profit ! Although this is just on a certain interval of time, this is pretty encouraging. We can see that the 'expected value' strategy outclasses the naive strategy, this is good news too.

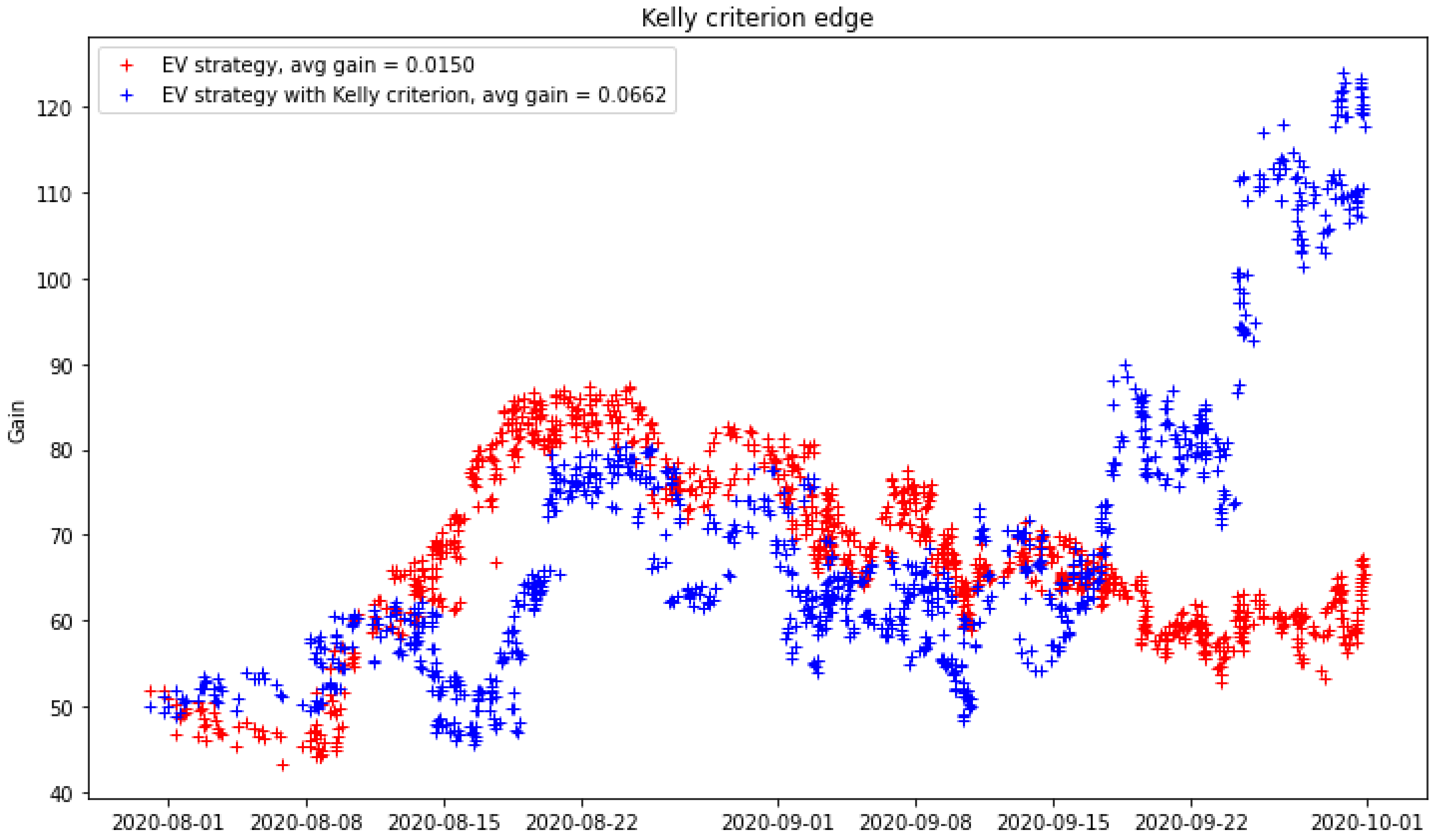
The Kelly criterion

The latter simulation we made was using a single constant amount for every bet. Now, what if we consider our bankroll limited of size B, and we want to maximize our bankroll through time, trying of course to avoid bankruptcy.

The Kelly criterion states that, in ordre to maximize the bankroll B we should bet at each step a fraction f of the bankroll :

$$f = \frac{edge}{odd}$$

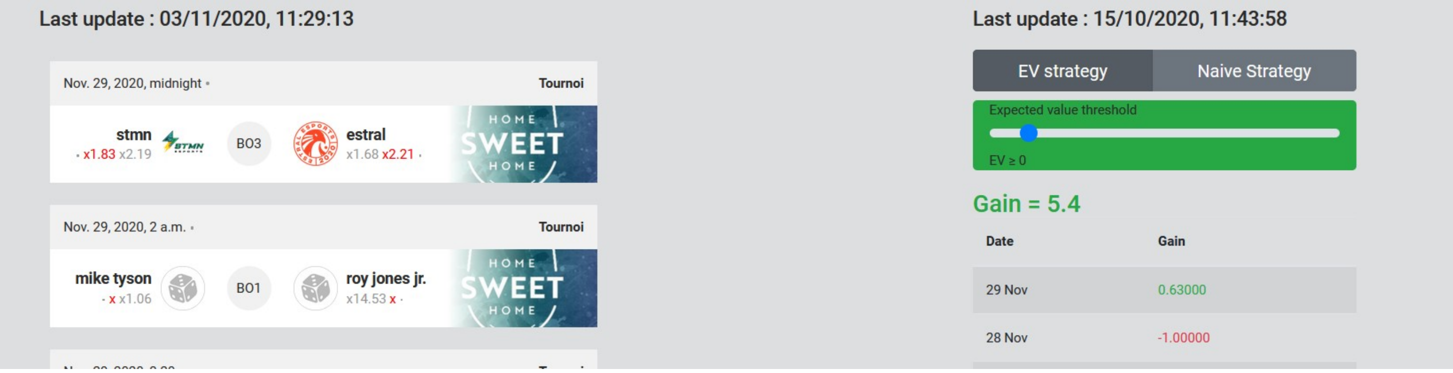
Using this strategy we can rerun an experiment and compare it to the vanilla constant value bet.



Here we start with $B = 50$. And we can clearly see that this strategy increases the global income. This strategy could be profitable if it was implemented.

The application

I built an app with this project to monitor the strategy in live. This is a Django application, if you are not familiar with it see https://www.djangoproject.com/start/. It comes with a tool that simulate online bets. More information in utils.py and in the notebook auto_betting_simu.ipynb. The whole code is not quite well documented as i don't really have time to do it, but don't hesitate to contact me if you have any questions regarding any part of the project.



Last update : 03/11/2020, 11:29:13

Nov. 29, 2020, midnight • Tournoi

stmn • x1.83 x2.19 BO3 estral x1.68 x2.21 •

Nov. 29, 2020, 2 a.m. • Tournoi

mike tyson • x x1.06 BO1 roy jones jr. x14.53 x •

Last update : 15/10/2020, 11:43:58

EV strategy | Naive Strategy

Expected value threshold

EV ≥ 0

Gain = 5.4

Date	Gain
29 Nov	0.63000
28 Nov	-1.00000

To go further

If you want to play with CSGO forecasting, you can fork the project and use the Django application to make your own models and predictions.

You will just need an PandasScore API key, and an account on csgolounge.

- In module_scrapping you will find a tool to recursively scrap games from csgo_lounge with selenium, thus it is possible to populate a database (to obtain the odds).
- In the notebooks the main features are detailled, they have to be run in django shell_plus in order to work properly with the database.