PA026: Documentation

English Premier League results prediction using Machine Learning

To download the project visit my GitHub repo.

1 Introduction

Predicting sports results is a fairly old discipline, ever since betting became so prominent. A lot of statistical methods were tried, obtaining decent results, possibly even producing profit if used for betting. Nowadays, the bookies have access to the most advanced machine learning models so they can set the best odds for themselves, and of course keep the machine learning know how under a tight seal.

Each sport has its own intricacies and it's usually pretty difficult even for the domain experts to predict results with high certainty. In my case, I've chosen to try and predict football results using machine learning, so I'll foreshadow some specifics of this sport.

Firstly, it's somewhat difficult to obtain good enough data. By that I mean data that contain more than just the result and few extremely basic stats about the game. Thankfully, organisations like OPTA have begun providing really detailed data in the recent two decades, though it's pay to use.

Secondly, even detailed data about matches and player performances is not even close to all the information that human experts have at disposal, and still achieve barely 60 % prediciton accuracies (and that's the best ones, usually it's around 53-55 %). Every league has a different style and different strength of its teams. In my case of EPL, it's regarded as the most competitive league in the world, which means many upsets and shocking results.

This sort of randomness coupled with factors, that can't be easily described by numbers such as

- Coaching staff change
- Key player injury
- Unexpected surge of individual form
- Player fatigue during busy spells
- Locker room drama

leads to machine learning models that struggle to beat the 50 % treshold. There's also only 380 matches played each season, with usually a week long pause between those. That doesn't mean that the teams don't play though, as there are ongoing cup competitions. This adds another difficulty, as fatigue becomes a factor and teams otherwise touted as favourites are forced to field a weaker team and might draw or lose to a team they'd comfortably beat with their starting eleven fully at disposal. I should also briefly mention that teams acquire or lose players quite often, so one can't rely on the team's name, but has to search for independent performance describing features to provide to the machine learning model.

To sum it up, the football of EPL is ripe with unexpected results, few matches played means trouble for models requiring a lot of data, and even getting the data is not that easy in itself.

1.1 My Goals

I've managed to get a fairly detailed dataset of 4 seasons, containing useful match statistics and even individual player stats and ratings. My idea is to try and use team statistics across the whole season played so far, the same stats but from a recent patch of matches (to make use of recent form), coupled with the starting 11's ratings and also a few constructed features to achieve a reasonable prediction accuracy using different types of machine learning models. The main improvement should be the explicit usage of recent form of the teams.

I want to find out key features, the ideal window of recent form, the importance of creating your own features and the performance of different models. I suspect that creative feature engineering is the key to reaching human expert levels of accuracy, and that the recent form window should be around a month at maximum, which means 3 to 5 recent matches.

2 How to run

After obtaining the code and data from my repo, simple **pip install -r requirements.txt** should be enough to install all the necessary libraries. Of course the number one assumption is a working Python 3. Working Tensorflow is also needed for running RNN model testing script.

All of the scripts are in the main folder and I will describe their usage in the later sections.

The folder `eval` contains the outputs of experiments and also a README that further describes the contents and findings. Folder `graphs` is self-describing I hope.

3 Data preparation

Transforming the raw data into usable datasets is handled by `data_prep.py` and `feature_creation.py` scripts. They require no additional input, just the data in `data` folder.

The match data is firstly merged with detailed team and player statistics. I've chosen to remove in my opinion unimportant stats like how many shots were shot at different heights and places in the net. From player statistics, I used only their ratings, to try and evaluate the strength of the starting elevens in each match.

Afterwards, the matches were sorted by their date, so I could more easily create stats about the recent form of the teams and later on input series for RNN models.

For each match, for both of the teams competing, the mean of their stats and player ratings is computed across their matches played so far in that season. In addition, the same thing is computed but only for their latest 3 / 5 / 8 matches, to reflect their recent form. This means that from the raw data I've created 3 datasets, which are different in the length of the recent form which is taken into account. Using normalization on the numerical stats, I have 6 datasets in total (3 normalized, 3 raw) to test machine learning models on, each containing 114 features (constructed included).

For the constructed features, I chose to try and use some I haven't seen in similar projects that are free to look into. For each team I keep its amount of wins, losses and draws (over the course of the season and recent form). I believe that especially recent W/D/L amounts should prove a useful feature, as it's a thing that human experts put a lot of weight on.

I've also computed custom measure of attacking and defending strength of each team. The idea is that home side advantage is a real and observed thing and most teams perform vastly differently when playing at home or away, so for the home team I compute its home attacking and defending strength, based on how many goals they score or concede when playing at home compared to the rest of the teams in the league. Same concept goes for the away team, taking into account its performance when playing away from home and comparing it to the rest of the league. This is of course also done over the whole season played so far and recent form.

More ideas on custom features are in the final section concerning improvement ideas.

4 Classical Models

Since classical machine learning models are the go to of almost every source I've seen tackling any kind of sports prediction nowadays, I also chose to employ them, namely the `scikit-learn` implementations of SVM, Gaussian Bayes, Multinomial Bayes, AdaBoost, kNN, Random Forests and Extra Trees. I've used 25 models in total for the initial testing, differing mostly in the number of estimators.

4.1 Initial testing

The first step was to see which type of model generally performs the best, without much tuning. Using `scikit-learn's SelectKBest` to reduce features (K = 20/30/40/50/60) on each of the 6 datasets I ended up with 36 datasets differing in normalization, recent form window length and feature reduction.

Testing of the models was done in 2 scenarios, both implemented by the `model_test.py` script.

In the first one, every model and dataset combination was trained on a randomly generated 85/15 train/test split 100 times, and the average accuracy was taken as the final result. Figure 1 displays the results of the best models of each type and the length of the recent form window.

The absolutely best performance was an Extra Trees models with 1 000 estimators, on dataset with recent form length of 3 matches, which achieved nearly 54 % accuracy. Extra Trees and Random Forests generally outperformed other models, as the first 22 best achieved accuracies belong to either of those two classes of models. Their best results were achieved on datasets without normalization, feature reduction didn't affect them much either. Interestingly, in most cases the best recent form window length is of 5 matches, but the very best model achieved its result using recent form length of 3. kNN models were capable of promising results too, especially with `k` higher than 100. Detailed results of all model and dataset combinations can be found in the `eval` folder, in the `results_final.csv` file.

Achieving nearly 54 % accuracy is a pretty good result, on par with upper tier human experts and slightly better than most project concerning this topic that I've seen. It's also a very good result given the small size of the initial data (only 4 seasons) and without any hyperparameter tuning.

The second scenario uses the last 50 matches of each season as the testing set and the rest of the matches as the training set, again over 100 iterations. This scenario should in theory achieve better results, as my method relies on computing the mean values of stats over the course of the season among other things, and these values should be more telling when more matches have been played. Figure 2 shows the actual results, once again in the format of the best achieved result by each model type.

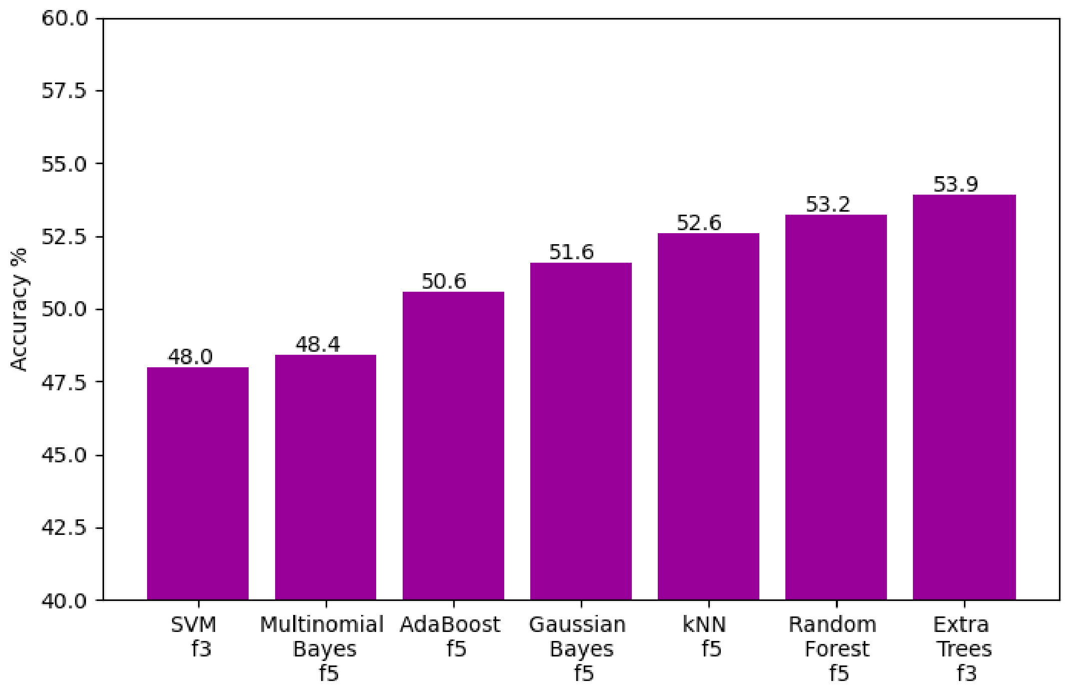


Figure 1: Initial testing, 100 iterations, random 85/15 train/test split

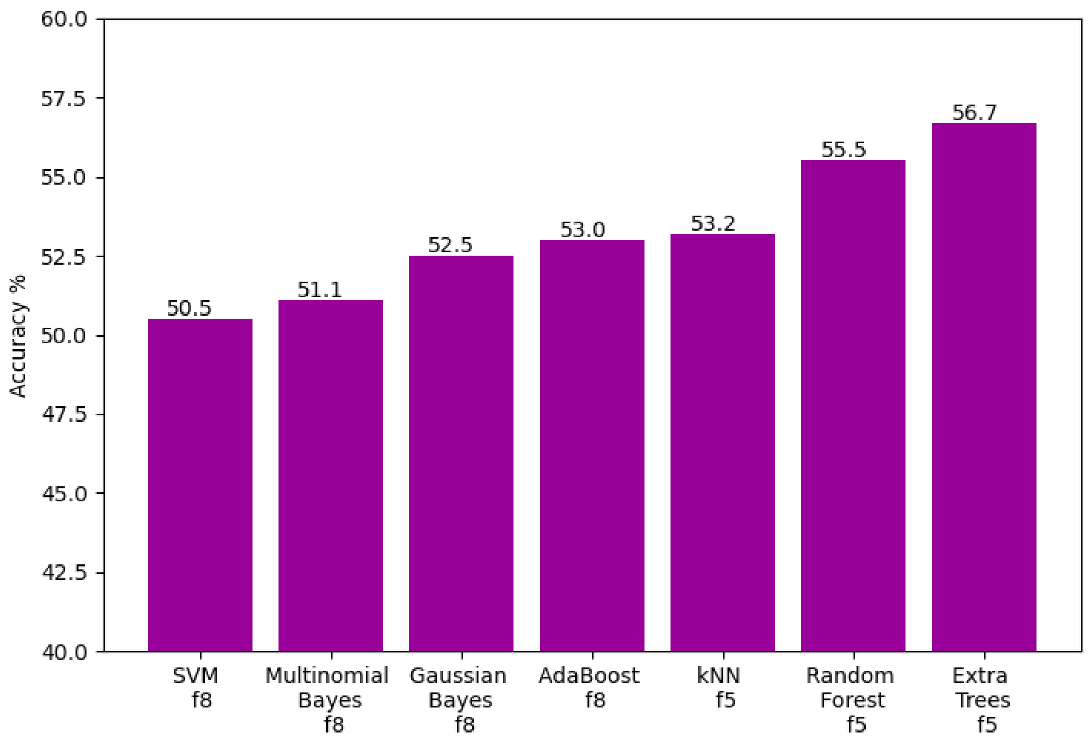
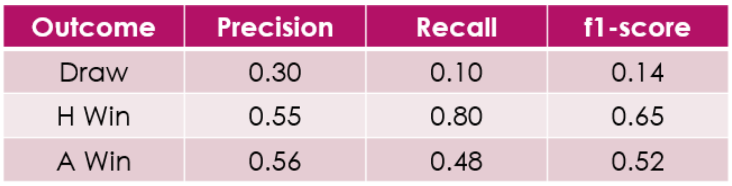


Figure 2: Initial testing, 100 iterations, last 5 weeks of each season used for testing

Once again it's the same duo dominating the rest of the models, taking up the 39 top spots ahead of any other models. This time the recent form window's length is skewed towards 5-8 matches, and the best results were achieved on feature reduced datasets. Detailed results are also stored in the `eval` folder, namely the `end_of_season_results_final.csv` file. And according to expectations, the accuracy is also slightly higher, with the best model reaching almost 57 %. This time it was once again Extra Trees model with 200 estimators, recent form length of 5 matches, and features reduced to the 30 most important.

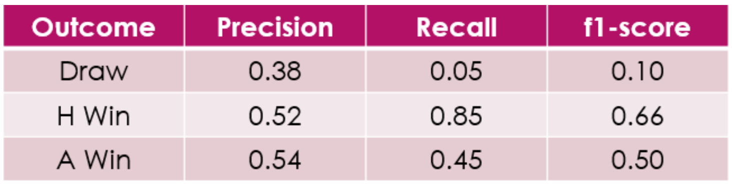
The `eval` folder also contains information about the precision, recall and f1 scores of models form the second scenario. Figures 3 and 4 show averages of these scores for the combined top performing Extra Trees and Random Forests (they're an extremely similar type of model, so I lump them together) and kNNs respectively.

All of the models struggled with draws, even the best ones, forests and trees, managed to achieve only 0.30 precision and 0.10 recall. This is somewhat understandable, as draws are notoriously hard to predict, and are usually happening due to a team overperforming or underperforming, something that's almost impossible to expect, especially just from match stats. Home victory predictions achieved impressive precisions of mostly around 0.55 and even more impressive recalls of about 0.80, at least in the cases of forests and trees. Away victories managed similar precision, but recall was way worse, generally around 0.48. These findings point to a similarity with human predictions, as draws are the most difficult to predict, while home side victories the easiest.



Outcome	Precision	Recall	f1-score
Draw	0.30	0.10	0.14
H Win	0.55	0.80	0.65
A Win	0.56	0.48	0.52

Figure 3: Forest scores



Outcome	Precision	Recall	f1-score
Draw	0.38	0.05	0.10
H Win	0.52	0.85	0.66
A Win	0.54	0.45	0.50

Figure 4: kNN scores

It can be seen that kNNs peformed similarly, but a bit worse in general, as was the case when accuracy was used as the scoring metric. Rest of the models performed way worse, and detailed information can be found in the `detailed_results.csv` file.

Overall, the achieved results didn't reach the perhaps overly optimistic expected values of around 60 %, but did not disappoint either, as they match all but the best human experts and quite a lot of related work on much larger datasets.

4.2 Grid search

Knowing the clearly best models, I've decided to try hyperparemeter tuning through Grid Search. I've decided to try both Extra Trees and Random Forests, as they performed the best across all the datasets. Once again I employed feature reduced and normalized datasets too. Hyperparameter grid consisted of different values of:

- **max_features** - {6, 10, 20, "auto"}
- **min_samples_leaf** - {3, 10, 20, 40}
- **min_samples_split** - {6, 10, 16, 40}
- **n_estimators** - {100, 500, 1000}

Information about each hyperparameter can be found in the official documentation, as I think it's nicely explained there and unnecessary to add here.

Testing and training was similar to the first scenario, with a random 85/15 split. Each combination of hyperparameters was tested on each of the 36 datasets using 3 fold cross-validation, using `sklearn`'s `GridSearchCV`. The whole ordeal is performed by the `model_grid_search.py` script.

For each dataset, the optimal hyperparameter combination was found, and the 15 best performing configs are stored to be easily viewed in `grid_results_combined_top15.txt` file. Unfortunately, there was no pattern among the hyperparameter configs and or datasets, or discernable improvement. Which leads to the conclusion that instead of hyperparameter tuning, one should look to construct better features and/or reduce available features in a worthwile manner.

4.3 Ablation testing

Finally, to see which features are the most valuable, I've done ablation testing. I chose to use an average performing Extra Trees model and the last matches of each season as the testing set scenario. This time over 50 iterations, and only on dataset containing all features and recent form length of 5. Done using the `model_ablation_test.py` script.

Results follow:

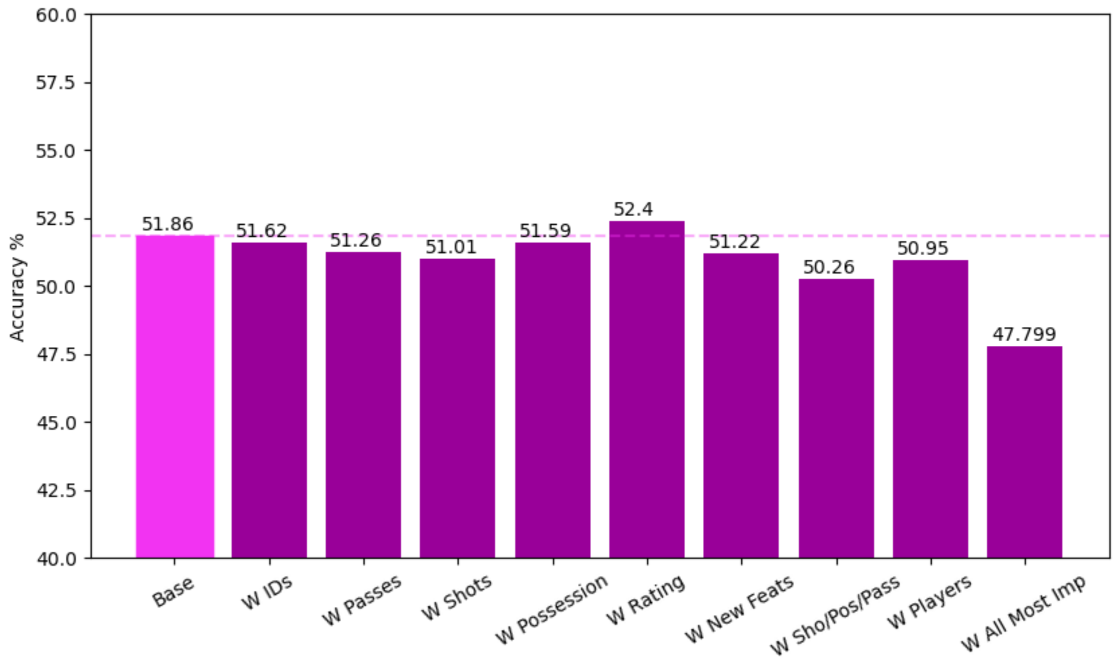


Figure 5: Ablation testing, 50 iterations, 85/15 split, W means Without

It's interesting to see that the ID of teams is actually useful in itself. Human experts of course are hugely influenced in their decisions by recognizing expected favourites, and it seems to translate positively to machine learning models too. The model itself values stats about shots, passes and possession the most, followed by my custom made features about team strengths and recent W/D/L. But only one stat does not seem to have a huge say, as removing it lowers the accuracy just slightly, only removing more of them drops it down significantly.

Slightly more hard to obtain stats, like the number of fouls and corners won, also hold some value while not necessarily being the most important ones. But it can be seen that just these stats and player ratings can still achieve "only" about 5 % worse accuracy, which is honestly suprising. Penalties saved information proved to be useless though, and it also kind of makes sense, when considering the nature of such event.

Player ratings seem to be also important, but when checking what the model itself finds important in explaining the result, it's interesting to see that it's only the attackers and midfielders that the model finds valuable, with the ratings of defenders and goalkeepers at the bottom of the standings.

The model also usually finds the recent form stats a bit more important than their seasonal equivalents. These can be seen by running the script and watching the console output.

Probably the most interesting finding is that the removal of the team ratings actually helps the accuracy. It is kind of understandable, since these ratings are subjective and not computed using a formula on team stats. It might be a good idea to use one's own, custom method of calculating a team's rating, based on hard data.

It seems that the most important stats are pretty intuitive, and that it is also necessary to construct useful features if one wants to reach outstanding accuracies. It might also be a good idea to try and find performance disturbing features and remove them, like is the case with the team ratings here.

5 RNNs

RNNs haven't really been tried that often from what I've seen when it comes to sports results predictions, which is surprising, given their penchant for working with data series. In essence, they should be capable of understanding recent form without me having to provide it for them, like with the classical models. I've decided to give them a try.

First I had to construct data series from the raw data. This is the purpose of the `rnn_data_prep.py` script, which generates for each match an input series of length specified in a command line argument, with normalized stats, which should help the network learn faster. I've experimented with series lengths of 2/3/4/5/8.

I tried using 3 main types architectures, all using LSTM cells. One shallow with single LSTM, second was several stacked LSTMs and the third one as a single bidirecitonal LSTM. Experimenting with different dropouts, epoch numbers, batch sizes and several different numbers of units in each LSTM was handled by the `rnn_test.py` script.

The results were far from satisfactory, as the best accuracy achieved was meager 48.5 %, comparable to the worst classical models. The networks overfitted far too much and no amount of dropout or different activations nor unit counts helped. All three architectures achieved pretty much the same accuracy, with the stacked one having a slightly lesser loss. Interestingly enough, all networks preferred series lengths of 2 and 3, which produced very similar results and scored the best. The longer the series the worse the accuracy, which makes some sense, as form in football rarely lasts for longer than 4 matches. Each architecture's loss and accuracies can be seen in the `rnn_results.txt` file, with values ordered to correspond to growing input series' length.

It would seem that for neural networks to work, one would need a much bigger dataset, which depending on the sport might be a problem. This also kind of explains the lack of use of RNNs in such studies, however well they might be suited for predicting based on a series of results.

6 Results and Improvement ideas

To recap my findings. I've managed to achieve decent accuracies of nearly 54 % (57 %) on random (end of season) testing matches, using a mix of intuitively important stats, custom created features, info about recent form (which I haven't seen anyone else use) and ratings of the players of the starting 11. Considering the size of the dataset, that's quite a nice result.

By far the best models proved to be Extra Trees, closely followed by Random Forests and kNNs trailing further behind. The models struggled with predicting draws but fared really well when predicting home victories, which is something strikingly similar to human predictions.

The best features proved to be shots on target, accurate passes, recent W/D/L, possession percentage and my computed home and away attacking and defensive strength. Recent form of stats was usually slightly more important than its seasonal counterpart. Player ratings help, but it's mostly the attacking players. Recent form was best computed over the last 3-5 matches. Feature reduction also seemed to help, normalization not so much. Hyperparameters also don't seem to matter that much, as grid search proved to be futile.

RNNs proved to have potential, but in my case couldn't match the other classifiers because of insufficient data. No amount of architecture and hyperparameter tweaking could overcome that. Their usage might be interesting in other sports, where data is more readily available and perhaps more matches per season are played.

The most obvious way to improve the results would be additional custom features. My idea would be to include goal difference stat, information about winning or losing streak (simple 1 for yes, 0 for no), perhaps points per match home and away. Player ratings could also be aggregated into offense / midfield / defense ratings, as they by themselves didn't amount to much, especially defenders'. It might be also interesting to include an indicator if the starting eleven is far from the optimal one, as that's most likely happen to injuries and rotations and would mean significant weakening of the side. This would require some manual input of expertise though. In real world, derbies hold a special importance to most teams, and they frequently produce results that don't make sense when looking only at the stats of the teams, so a simple indicator if the match is a derby could maybe prove to be useful too. Lastly, custom team rating in the form of some function over team stats could be also productive, as the subjective rating of experts only seemed to confuse the models.

The scripts themselves could also use some refactoring work to allow for better experiments, as they currently require too much manual editing.

7 Related work

Most of the related work I've seen uses basic stats combined with some custom computed features, to reach slightly above 50 % accuracy. The most advanced research I've come across uses extremely detailed and sizeable datasets with features like expected goals (xG), which is computed using formulas that require GPS data - not something easy to come by. The key attributes of performances of individual players is also something I've seen data scientists experiment with. As well as Twitter posts analysis to take into account pre-match news. While interesting ideas, I thought them to be outside of the scope of this project, due to time constraints. Sadly, the absolute peak research is done in secrecy by betting companies.

I've seen RNNs used only to predict the outcome of already running match, based on the events happening in it. They performed really well on that, but then again, it's rather easy to predict the outcome of a match that's running and one can see the performances and stats. I haven't come across any research using them to predict the outcome of a match before it starts.

Below I've listed some of the interesting stuff I came across when doing my initial research on this topic:

- https://www.researchgate.net/publication/257048220
- https://www.researchgate.net/publication/338090323
- https://www.imperial.ac.uk/media/imperial-college/faculty-of-engineering/computing/public/1718-ug-projects/Corentin-Herbinet-Using-Machine-Learning-techniques-to- predict-the-outcome-of-profressional-football-matches.pdf
- https://pdfs.semanticscholar.org/e556/af01e86c3414042aa69831ea5fb398e66f94.pdf
- https://www.sciencedirect.com/science/article/pii/S2210832717301485
- https://towardsdatascience.com/what-ive-learnt-predicting-soccer-matches-with-machine-learning-b3f8b445149d
- http://kickoff.ai/