UNIVERSITÄT DES SAARLANDES

Football Team Strength Models

Mathieu Tischer & Maximilian Prinz

Universität des Saarlandes

Data Science and Artificial Intelligence

Chair for Sports Analytics

04.08.2025

Abstract

The goal is to build a robust model to quantify team strength for football teams by using an ELO based model and machine learning classifiers. We use the brier score to evaluate how good each model is at making predictions. In the making, we tackle the problem of having too little data and extracting meaningful predictors for our models. We use open-source python libraries and the free Statsbomb football dataset to achieve this goal. The ELO approach delivers rather mediocre results while on the other hand, a Gradient Boosting Classifier was able to deliver great predictions because of its more in-depth structure.

Table of Contents

Introduction

In modern Football, there are countless methods to quantify how well a team performs. We subconsciously do it every time we watch the sport; What makes this team stronger than the other? How large is the impact of that specific player? Do the two teams have a history of match results? All these questions are important when building an appropriate team strength model, but why do we need a model in the first place? Team strength models are helpful for making predictions for upcoming matches [1] and are used in the making of betting odds.

In this project we dive deep into the task of building the best possible team strength model to try and predict future football match outcomes ourselves. For that, we use two main approaches. The first approach is a model based on the ELO formula mostly known from chess, in which it

has been very useful in recent years. Modern chess websites like chess.com use it to quantify how good each player is at the sport. How it works is it assigns a number to every player (in our case this will be teams) and for every game it assigns an expected value for the two players and afterwards updates the ratings according to the actual outcome of the game [2].

The second solution for the task is to use the dataset we have to find meaningful predictors and train a machine learning classifier using these predictors. In the process, we also develop a way to generate more training data based on the data already available.

Methodology

Data

To start off we needed a dataset to base our work on and we opted for the free open source event dataset from Statsbomb [3]. It consists of event data for every match where every event is a detailed description of what is happening with the ball during each moment of the game, as well as who is handling it, what type of shot it is, etc.... The set contains matches from numerous competitions like the Bundesliga, the Euros, the English Premier League, and more dating all the way back to 1958. The best fit for our task was to train and evaluate everything using the data from the 2015/2016 Bundesliga season because it contained a consistent number of matches for each team for a small period of time, which is not the case for tournaments like the 2024 Euro's. 2015/2016 was also the only season for which the dataset provided all matches for Europe's top leagues.

Code

The main programming language and version we used was Python 3.11.2 as it is very widely used and practical when working with machine learning, and we used Jupyter Notebook to execute files, functions, blocks of code, etc... for an optimal workflow.

To build machine learning models we opted for the Scikit-learn python library [4] which gave us access to the most popular classification models. The data we had consisted of large .json files which we were able to parse using Statsbomb's own python library [5], and process using pandas.DataFrame [6]. Finally, to visualize our work and results we used the Matplotlib python library [7].

Implementation

ELO model

The ELO rating system follows a simple formula for which we had to optimize all parameters. As already explained above, it calculates an expected value E_H for the home team (and E_A for the away team) and updates the teams' ratings after the match. We adapted the classic E_H formula to use more parameters which alter important factors. The following is an overview of the theory behind the parameters and the formula:

- K determines how quickly ELO ratings change (the higher, the more the updated ratings will differ).
- α translates ELO differences into outcome probabilities and is directly related to the scaling factor parameter in the classic ELO formula.
- β controls how much probability mass given to draws.

- γ controls how fast draw probabilities decrease.
- S is the actual outcome of the match.
- Moreover, there is a starting ELO parameter R_H^0.

$$E_H = e^{\alpha(R_H - R_A)} + \frac{1}{2} e^{\beta - \gamma|R_H - R_A|}$$

$$R_H^{n+1} = R_H^n + K(S - E_H)$$

After implementing the formula, we now had to test how well the ELO system performs, but since the starting ELO is the same for every team, we found that it would not accurately represent the teams' original strengths at the start of the season. E.g. one would expect Bayern München to perform way better than Eintracht Frankfurt based on the previous season's results [8]. Since we did not have the data for the 2014/2015 season available to us, we had to work our way around this issue by running the ELO formula through the season a first time and using the final ELO standings as starting ELO's for the evaluation run.

The only thing missing now is a metric to quantify how good the predictions made by the model were, and for that we used the Brier score. Our ELO model outputs the outcome probabilities

$$p_H = e^{\alpha(R_H - R_A)},\ p_D = e^{\beta - \gamma|R_H - R_A|},\ p_A = e^{-\alpha(R_H - R_A)}$$

and we put these probabilities in a vector to compare with a one-hot encoded actual outcome vector and apply the brier score formula on. In our 3-dimensional case, the formula follows

$$BS = (p_H - o_H)^2 + (p_D - o_D)^2 + (p_A - o_A)^2$$

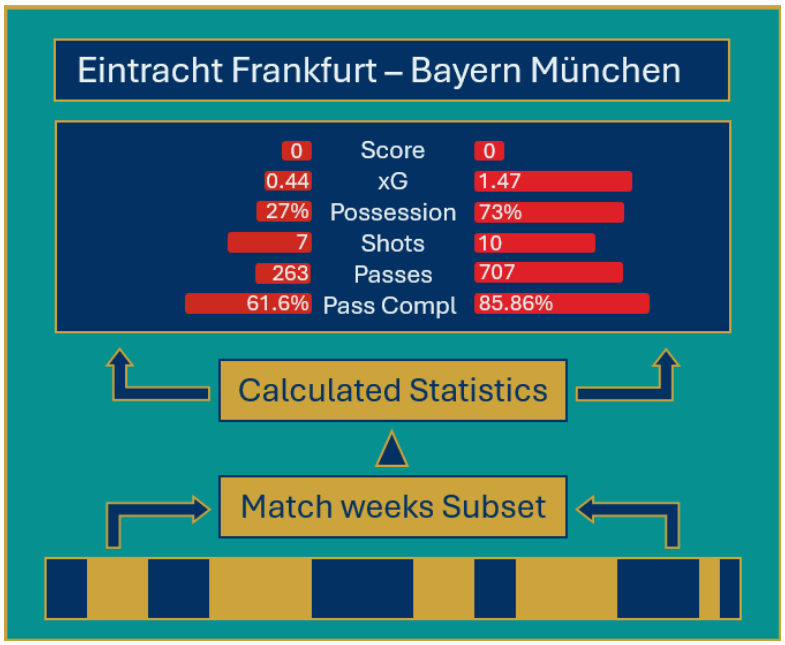
where o is the binary boolean value of each outcome class i.e. Home win, Draw and Home Loss. The brier score takes a value between 0 and 2; the lower the score, the better the prediction was. We do this for every match and average out per matchday and season.

The brier score obviously depends on the parameters, and since we had a lot of them to tune, we used a parameter grid search technique which iterates over every combination of parameters for a given range of values to find the optimal performing combination.

Data generation

As already mentioned, one problem we encountered while working with the dataset is that while it was very detailed, it was unfortunately very little to work with i.e. we only had one season, so it was going to be hard to properly train our classification models.

To solve this issue, we resulted in a method that generates match data we could train our model on based on the data that we already had. We essentially created fake pairings for one match week and used the actual outcome of that pairing in the 2015/2016 season as a label for that data point (visualization seen below). We could then tune the number of weeks we wanted to simulate and chose 510 since that would correspond to having data from the 2000/2001 season onwards.



Feature extraction

Now that the training data was set, the next task was to find predictors for our ML models like average ball possession, xG, average shots on target, etc. from the dataset based on the events in each match. Ideally, these would then help to build a model that outperforms ELO. Using Statsbomb's library's functionalities, simple queries and pandas.DataFrame, we could easily extract, group the statistics by team, and then average them out for our model to use.

ML models

Now that we had everything set up for training and testing, we could easily choose and implement some well-known classifiers using the imported libraries and evaluate their performance, again, using the brier score.

Since we had enough data for training, there was no reason to let the model run twice through the season like in the ELO approach before, so we let the models run through the data, produce probability vectors to evaluate the brier score, and kept track of the best parameters for each one of them. The optimal parameters for each model were, again, found by applying a grid search technique with a given range of values for each parameter.

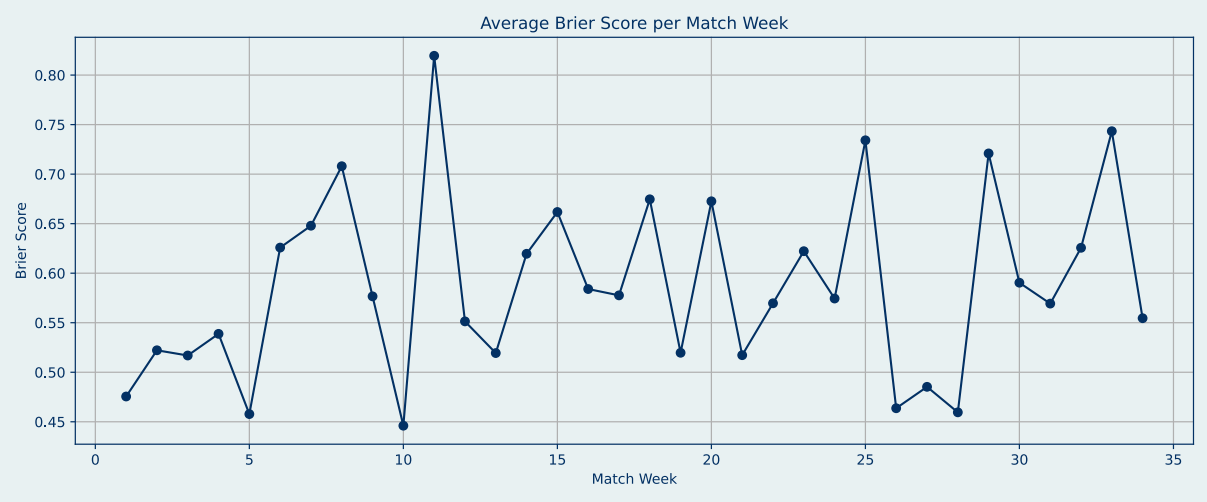
Results

ELO model results

After applying grid search to numerous parameter ranges, the best combination of parameters for the ELO model was

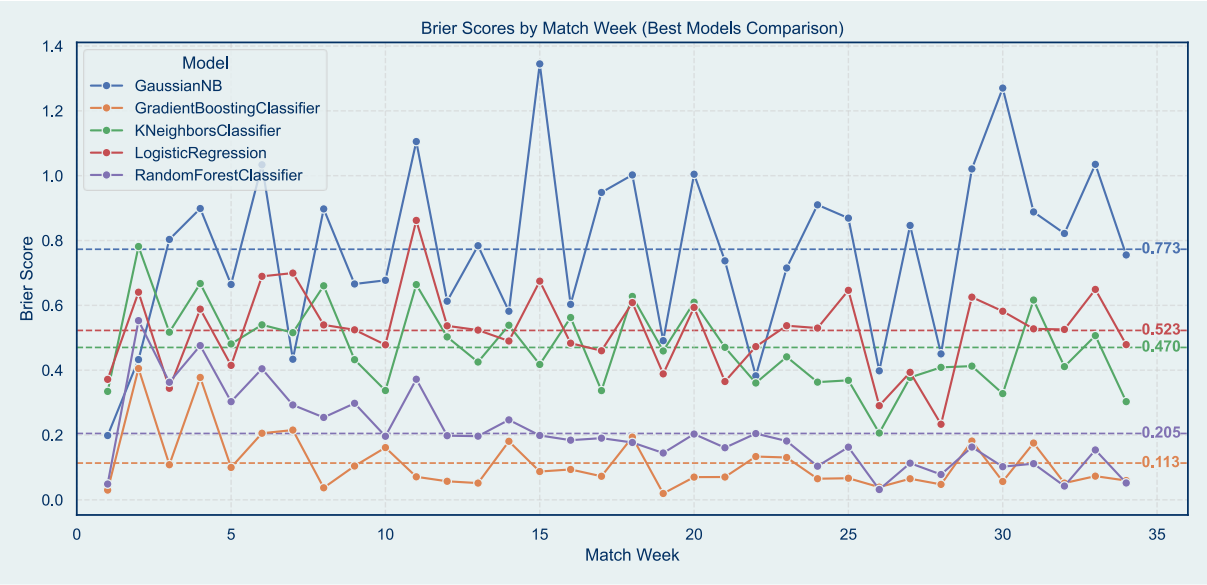
$$K = 1, \alpha = 0.057, \beta = 0.0, \gamma = 0.1, R^0 = 1500$$

and yielded us a brier score of around 0.58. We have to note that small tweaks in the parameter values would sometimes not significantly affect the score, so while these parameters were optimal, there were a lot of other combinations that achieved almost the same result. The plot below shows the evolution of the brier score across the season using the optimal parameters.



ML classifiers results

The results for the ML models differed a lot depending on the model used. In general, most of the models outperformed the ELO approach – some of them gave poor results while others excelled in expectations. We will focus on the best Classifier we could find which was the Gradient Boosting Classifier with a brier score of 0.113 and parameters n_estimators = 100, which decides how many trees the model should grow, and learning rate = 0.5 which determines how much each tree contributes. The plot below shows some of the best (and worst) performing models.



Discussion

Result interpretation

The ELO model only uses match outcome labels to quantify team strength and suffers a lot under the events of surprising wins and draws - and the probability vectors it puts out are not tweakable in a way that addresses these problems. That is why its' brier score took a rather disappointing value of 0.58 (for reference: random guessing would yield a score of 0.66).

In contrary to that, the Gradient boosting classifier uses meaningful features to make predictions. The way it works is it builds a strong model by combining many small decision trees

in sequence. It starts with a simple initial prediction and then repeatedly looks at mistakes made so far to fit a new tree to predict those errors using said input features. The values in each tree's leaves are chosen to reduce the overall loss, and each tree's contribution is scaled before being added to the model [9]. This graduate way of keeping track of mistakes helps it model consistent performances of certain teams more accurately.

Limitations

Unfortunately, some of the predictors shared a higher correlation than we would have wished for, but we had to keep them since it was not possible to get some of the most important predictors like Starting XI because of how the data is structured. This predictor for example is one that betting websites must heavily consider when calculating odds because of how large the impact of even the substitution of only one player is.

As we addressed before, the data we had was very little to work with, so we had to rely on our data generation to train the models, which was fine for the purpose of this project but made our models a little more biased. The participant list changes every season and while some teams consistently perform well over many years, most teams in the mid-table region achieve very different placements every season. To generalize better, we would have needed the data from a few of the previous (or even following) seasons.

Conclusion

All in all, we can say that we achieved our goal on this project - while we had to work with some limitations, we were able to build a decent model for making predictions in football. Obviously, we could have optimized our model to achieve perfect results each time, but this was not the goal of the project. The problems and limitations we encountered along the way are common when working with ML, and we were able to take away a few things which could help us in future projects.

For a future improved implementation of our project, we would consider using noise to make the models generalize a bit better so that surprising wins, for example, don't affect the performance too much. We would like to work on our data generation, i.e. finding better/more creative ways to make data generation more robust.

References

Hudl Statsbomb – The World's Most Advanced Football Data. (2025). https://www.hudl.com/en_gb/products/statsbomb

[1] B. Owusu, 'How to develop a global team strength model?', Matchmetrics. Accessed: Aug. 07, 2025. [Online]. Available: https://matchmetrics.com/insights/team-strength-model/
[2] 'Elo Rating System - Chess Terms', Chess.com. Accessed: Aug. 07, 2025. [Online]. Available: https://www.chess.com/terms/elo-rating-chess
[3] *statsbomb/open-data*. (Aug. 07, 2025). StatsBomb. Accessed: Aug. 07, 2025. [Online]. Available: https://github.com/statsbomb/open-data
[4] 'scikit-learn: machine learning in Python — scikit-learn 1.7.1 documentation'. Accessed: Aug. 07, 2025. [Online]. Available: https://scikit-learn.org/stable/
[5] *statsbomb/statsbombpy*. (Aug. 05, 2025). Python. StatsBomb. Accessed: Aug. 07, 2025. [Online]. Available: https://github.com/statsbomb/statsbombpy
[6] 'pandas.DataFrame — pandas 2.3.1 documentation'. Accessed: Aug. 07, 2025. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

[7] ‘Matplotlib — Visualization with Python’. Accessed: Aug. 07, 2025. [Online]. Available: https://matplotlib.org/
[8] ‘Bundesliga 2014/15 - Tabelle | 34. Spieltag’, kicker. Accessed: Aug. 07, 2025. [Online]. Available: https://www.kicker.de/bundesliga/tabelle/2014-15/34
[9] ‘Gradient Boosting in ML’, GeeksforGeeks. Accessed: Aug. 08, 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/ml-gradient-boosting/

Appendix

Project Code: https://github.com/MaxPrinz98/FootballDSAI