Using algorithms to build the ultimate Fantasy Premier League team

Neel Thakurdas · Follow

11 min read · Aug 11

As the new premier league season kicks off this weekend, it brings with it a fresh new fantasy premier league (FPL) season. For those who don't know what FPL is, it is a virtual football manager game where fantasy managers pick players who score points based on how well they perform in real-life. In my case, having played the game since 2017, the past few years have seen me finishing behind the likes of fantasy teams 'Hakuna Juan Mata' and 'Dijk in Diaz' in my FPL mini-league. With my

traditional approach falling short, I decided I needed a new strategy. Relying on information from 'FPL_JohnCena69' on twitter is no longer a cutting-edge way of playing the game (before you go check, no the twitter handle does not exist). So let me take you through my new approach for 2023–24, which uses machine learning and optimization to recommend the 'optimal' team.

Acquiring the data

As I anticipated spending several days cleaning thousands and thousands of rows of data, I stumbled upon Vaastav Anand on GitHub, who had already done some incredible work on cleaning and web-scraping the data. His dataset consisted of stats for every individual player and the games they played from 2016 onwards. While there were a handful of missing games, there were still over 100,000 rows of data and over 30 columns to work with (shown below).

```
Index(['name', 'position', 'assists', 'bonus', 'bps', 'clean_sheets',
       'creativity', 'element', 'fixture', 'goals_conceded', 'goals_scored',
       'ict_index', 'influence', 'kickoff_time', 'minutes', 'opponent_team',
       'own_goals', 'penalties_missed', 'penalties_saved', 'red_cards',
       'saves', 'selected', 'team_a_score', 'team_h_score', 'threat',
       'total_points', 'transfers_balance', 'transfers_in', 'transfers_out',
       'value', 'was_home', 'yellow_cards', 'GW'],
      dtype='object')
```

Cleaning the data

The main dataset consisted of two separate datasets that I merged together. The first dataset had data from 2016 all the way till 2022, and the second one had data for just the 2022–2023 season. The issue was that unlike the second dataset, the first one did not have any expected data, which considers the probable outcome of an event based on available information. For example, xG, or expected goals, which is a widely used metric nowadays, quantifies the likelihood of a particular shot resulting in a goal based on factors such as shot distance, angle, and various player actions leading up to the shot. Since it wasn't present in the earlier data, I unfortunately had to exclude all the expected columns from the combined dataset. Additionally, I had to re-assign the price of each player in accordance with their new prices set for the upcoming 2023–24 season, as well as change the teams for players that transferred

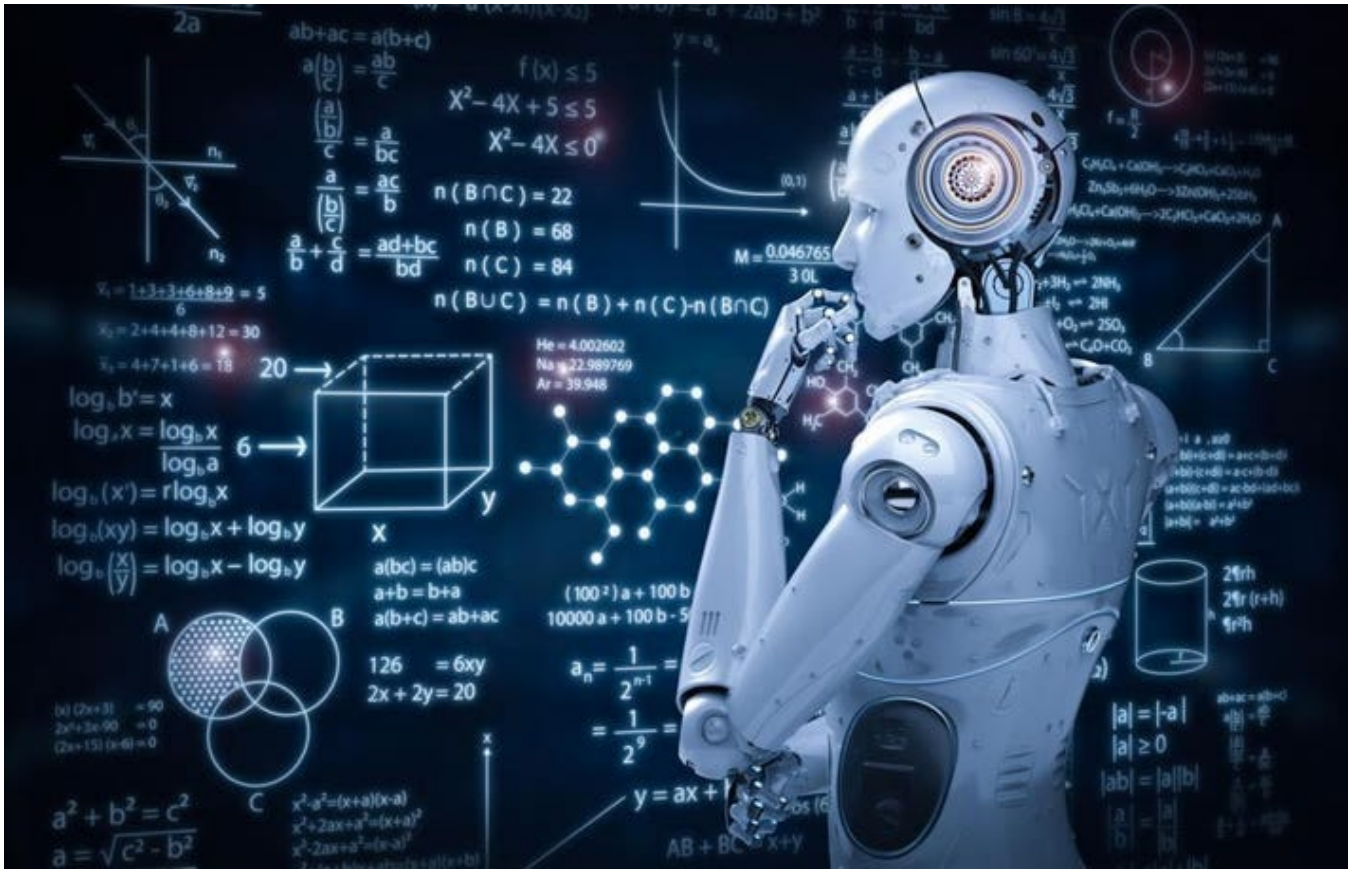
between premier league clubs (ex: Mason Mount from Chelsea to Manchester United).

Given that machine learning models cannot directly handle string values, I had to preprocess the data. First, I uniquely assigned an ID number to each player, allowing the model to easily differentiate individuals. After dissecting the kickoff time into discrete components: 'Year', 'Month', 'DayOfMonth', 'DayofWeek', and 'Time, I transformed player positions into numerical equivalents, wherein goalkeepers were labeled as 0, defenders as 1, midfielders as 2, and forwards as 3. Similarly, unique codes were also assigned to each premier league team.

Creating and evaluating the ML model

To predict the total points, I extracted diverse features from the dataset, encompassing attributes such as player position, assists, goals scored, and more. Next, using Python's 'Scikit-learn' library, I split the dataset into two parts: a training set to teach the model patterns in the data, and a testing set to evaluate its accuracy. To make predictions, I used a Random Forest Regression model, which is well-suited for predicting numerical outcomes, like the total points a player might score. I trained this model using the training data, enabling it to learn the relationships

between player attributes and their total points. Finally, I deployed the trained model to predict total points on the test set, and the predictions are compared against the actual values to assess the model's predictive capabilities.



Mean Squared Error (MSE) is a statistical metric used to measure the accuracy of a predictive model, like the Random Forest Regression model we employed. It quantifies the average squared difference between the predicted values and the actual values in a dataset. By squaring the differences, MSE emphasizes larger errors and penalizes them more heavily, providing a comprehensive assessment of the model's performance. The model achieved an impressive MSE of 0.055, signifying that the squared difference between our predicted and actual total points is consistently minimal. This result indicates that the model's predictions closely align with the actual points scored by players in the test dataset, reflecting its reasonably high accuracy in making predictions.

Creating the optimization problem

To suggest the 'ideal' team, I used an optimization method. This involves creating a

mathematical model that seeks to maximize a specific goal while following certain rules. By integrating predicted points from the ML model, this mathematical model picks the best players to maximize team points within budget and position limits. I employed the 'PuLP' Python library to build and solve this model, as it's great for tackling linear programming problems such as this one. Before mentioning the constraints, here is a brief overview of the rules of the game (you can skip reading this bit if you already know them):

- The objective of the game is to score the most points.
- Points are scored through goals, assists, clean sheets and saves.
- Points are deducted when a player receives a yellow card, red card or scores an own goal.
- Players are priced according to their points potential. Logically, higher the price, higher the perceived points potential. For example, Erling Haaland is priced at £14.0m, while someone like Elliot Anderson is priced at £4.5m. If you're wondering who Elliot Anderson is, that's exactly my point.
- You have a £100.0m budget to pick a squad of exactly 15 players (2 goalies, 5 defenders, 5 midfielders and 3 forwards.)
- From your squad of 15 players, you have to choose 11 playing players and 4 to sit on the bench. Of these 11 playing players, there has to be exactly 1 goalie, at least 3 defenders, at least 2 midfielders and at least 1 forward. Points scored by players on the bench do not count towards your team's total.
- You cannot pick the same player more than once and cannot have more than 3 players from the same team.
- Each week, you have to decide a captain, which earns you double the points for that player for that week. You are also eligible to make one free transfer within your team each week, with every additional transfer setting you back 4 points.

There's more rules about chips, bench order, etc., but I won't complicate it further. Using the rules of the game mentioned above, here are the constraints for the

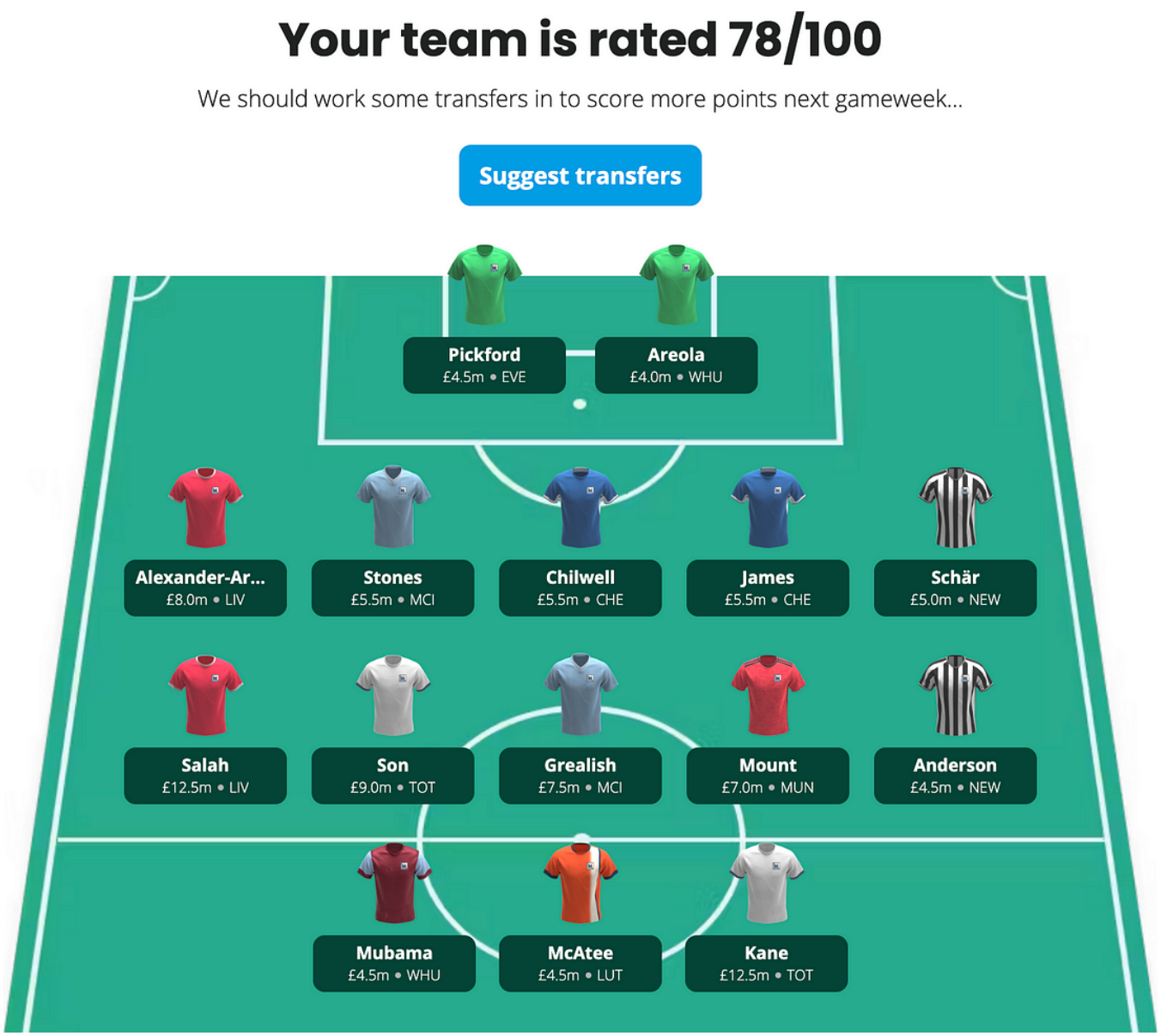
optimization model:

- Pick players that maximize total predicted points
- Exclude players that are no longer playing in the premier league, those that are injured or suspended, and any player who is highly likely to be a rotation risk.
- Pick exactly 11 players. The reason why I want the model to suggest 11 playing players is because it becomes complicated when the concept of a bench is introduced.
- Pick exactly 1 goalie, a minimum of 3 and maximum of 5 defenders, a minimum of 2 and maximum of 5 midfielders, and a minimum of 1 and maximum of 3 forwards.
- Pick a maximum of 3 players from any given team
- Do not exceed a budget of £83.0m (since we are picking 11 and not 15 players, and I prefer to have a bench that does not exceed £17.0m).
- Spend at least £82.0m (As I do not like to have a lot of money in the bank. This only applies to FPL and not real life. If you would like to send me money, please feel free).

The 'optimal' team

With those constraints, here was the team the model suggested:

The model backed the Chelsea defensive double-up of James and Chilwell, a defender and attacker each from Man City and Liverpool, Spurs duo Kane and Son. With the help of Fantasy Football Hub's free tool that evaluates your team using AI, here is how the team fared (I filled in the bench with the cheapest players in each position):



78% is not the result I was expecting, which is a C+ at best. But here is what it is missing: my model weighs the importance of each game equally, which means regardless of whether the game was played in 2016 or 2023, it means all the same to the model. Therefore, a player like Erling Haaland, who broke countless records last season and is arguably a no-brainer pick, is unfairly punished by the model for his lack of longevity in the league. Similarly, despite a rejuvenated Arsenal team that finished an impressive 2nd in the league last season, none of their players make the final 11. The model also does not take fixtures into account, which is an essential part of FPL decision-making. For instance, a player is more likely to perform better when playing Luton at home than Man City away.

Adjusting the weight of matches

In order for the optimization model to prioritize more recent matches over older ones, I had to create an additional constraint, where more recent matches have a higher weight and the weight decreases as we move further back in time. Here is the formula I used to calculate the time weight:

Time Weight = e^(-decay rate * days ago)

Where:

- Time Weight is the calculated importance of a match.
- The decay rate constant (which I kept as 0.001) is a parameter that determines how quickly the weight of a match diminishes as time passes.
- 'Days ago' represents the number of days between the match's kickoff time and the day of the most recent game. Therefore, the most recent day would have a weight of 1.

For example, say a game was played exactly a year before the most recent game. Using our decay rate constant of 0.001 and weight formula:

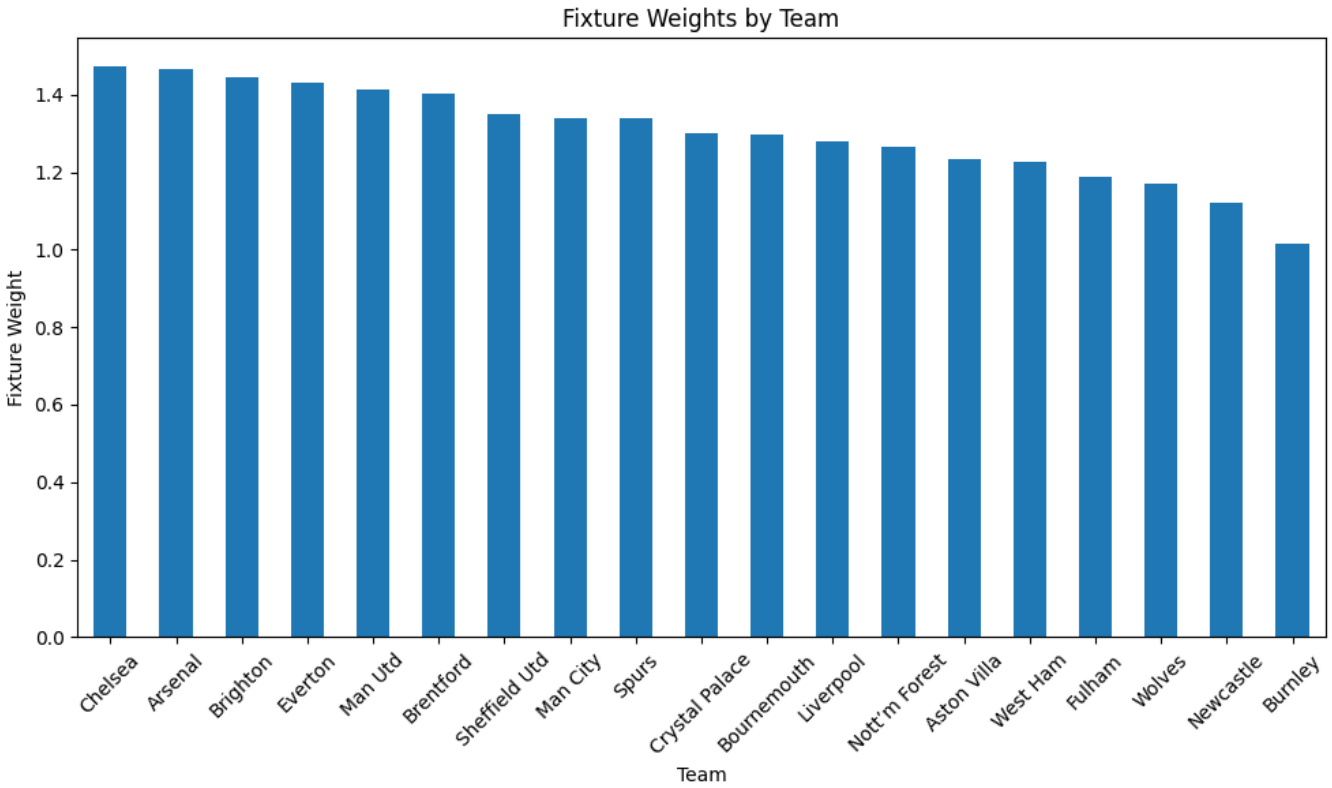
Time Weight = e^(-0.001 * 365) = 0.6942

Therefore, the weight of the game played one year before the most recent game is approximately 0.6942, or it carries around 69.42% of the importance compared to the most recent game. And the reason why I chose 0.001 as the decay constant is because, in my opinion, it falls in that goldilocks zone where it prioritizes more recent games while still placing some sort of emphasis on older games. For instance, if I had increased or decreased the decay rate by a factor of 10 in either direction (0.01 and 0.0001), the same time frame of a year between two games would have resulted in a weight of 2.60% and 96.42% respectively.

To take fixtures into account, I used Fantasy Football Hub's free fixture analyzer, which is a far more detailed assessment of fixtures than the one on the official

premier league website. I took fixture difficulty data for the first 5 match weeks of the upcoming 2023–24 season, and weighted them based on how recent the fixture is, with higher emphasis being placed on the more recent weeks to account for transfers within the team. With one free transfer a week, you could swap out essentially half your team within 5 weeks. By multiplying the weightage of each fixture with respect to the recency and difficulty, we would have the weighted average fixture difficulty for each team. In layman's terms, say Team A were to play 2 matches, with fixture difficulty ratings of 1.5 and 0.5 (the higher the rating, the easier the fixture, and vice versa) and a weightage of 0.7 for matchweek 1 and 0.3 for matchweek 2. Team A would have a weighted average fixture difficulty of (0.7 * 1.5) + (0.3 * 0.5) = 1.2.

Using a ratio of 0.24 : 0.22 : 0.20 : 0.18 : 0.16 (this ratio was used because each weekly free transfer retains 10/11 or roughly 0.91 of your team, which is roughly equal to the ratio between adjacent weights), here is the weighted average fixture difficulty rating for each team:



Chelsea, Arsenal and Brighton have a far easier start to the season compared to the likes of Burnley, Newcastle and Wolves. With the help of this information, this was

the simple formula used to calculate weighted predicted points:

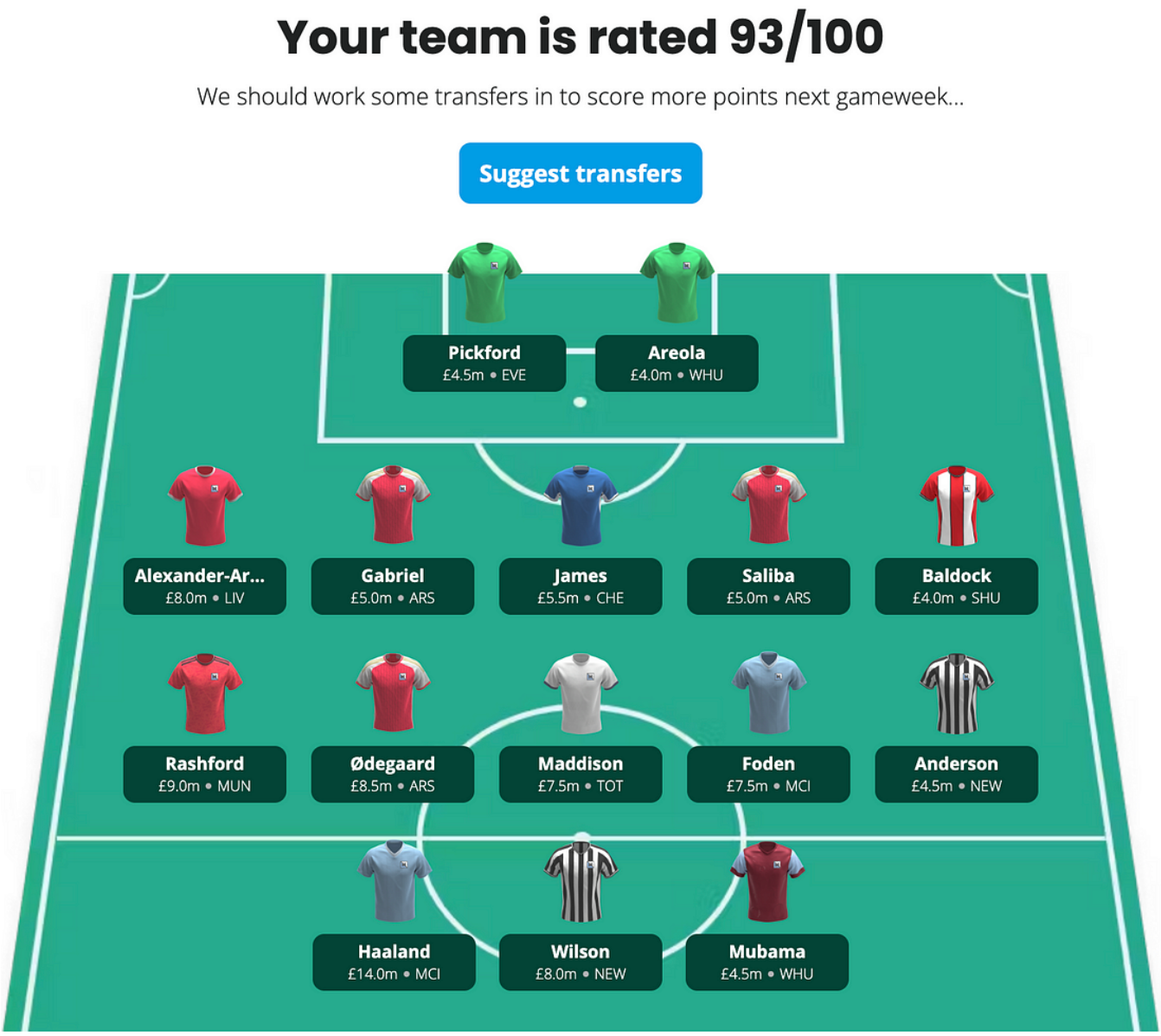
Weighted Predicted Points = Predicted Points *Time Weight * Fixture Weight

Using this, I updated the constraint to maximize weighted predicted points instead of predicted points.

The updated 'optimal' team:

Taking recency of play and fixture difficulty into account, this is the 11 the model suggested. The biggest difference between the two teams being that the model now recommends an Arsenal defensive double-up, as well as the Norwegian duo of Martin Ødegaard and of course, Erling Haaland. Despite having relatively poor 2022–23 seasons by their high standards, Alexander-Arnold and Reece James still had the backing of the model, considering how explosive they can be. Jordan Pickford also retained his spot between the goalposts, with Rashford, Foden and

Maddison replacing their teammates Mount, Grealish and Son respectively. To finish the team, Callum Wilson got the nod despite Newcastle's tricky fixtures. Here's how the team performed:



93%! That's an A+, which is a huge improvement! However, to keep in mind, a higher team rating score does not automatically guarantee more FPL points, it just means you are giving yourself a better probability of scoring more points. So, if it doesn't perform well, I can always blame it on luck.

Future improvements

While I am happy the model was able to score a 93%, here is what I think can be

done to the model to improve that:

- **Incorporate New Player Data:** By incorporating data on incoming players from foreign leagues and assigning appropriate weights, the model could make more comprehensive player selections.
- **Utilize Advanced Metrics:** By integrating advanced metrics like expected goals (xG), expected assists (xA), and player involvement in critical events, the model [illegible] insights into player performance.
- **Address Injuries and Suspensions:** Factoring in player injuries and suspensions would be the model to avoid players who might miss upcoming matches and thereby enhance overall team performance.
- **Dynamic Data Integration:** Enabling the model to adapt to changing circumstances by incorporating dynamic features like player prices and recent form would improve both short and long-term decision-making capabilities.
- **[illegible]mize Captaincy and Transfer Strategy:** Extending the model's optimization [illegible]ilities to recommend optimal captaincy choices, transfer strategies and [illegible]p usage would account for immediate and long-term gains.
- **Consider Contextual Insights:** Arguably the most important, the model could enhance decision-making by incorporating contextual factors such as changes in team formations and player roles. For instance, if there is a formation switch within a team and a certain player has a more attacking role, the model should know this. Incorporating a feature that considers community insights would be of significant help.

FPL Data Science

Follow

Written by Neel Thakurdas

2 Followers

More from Neel Thakurdas

Overall, the model does well in producing a competitive FPL team. Now, I'm curious to find out how this team fares in real-life. Only time will tell.

The code and data for this project can be found here.

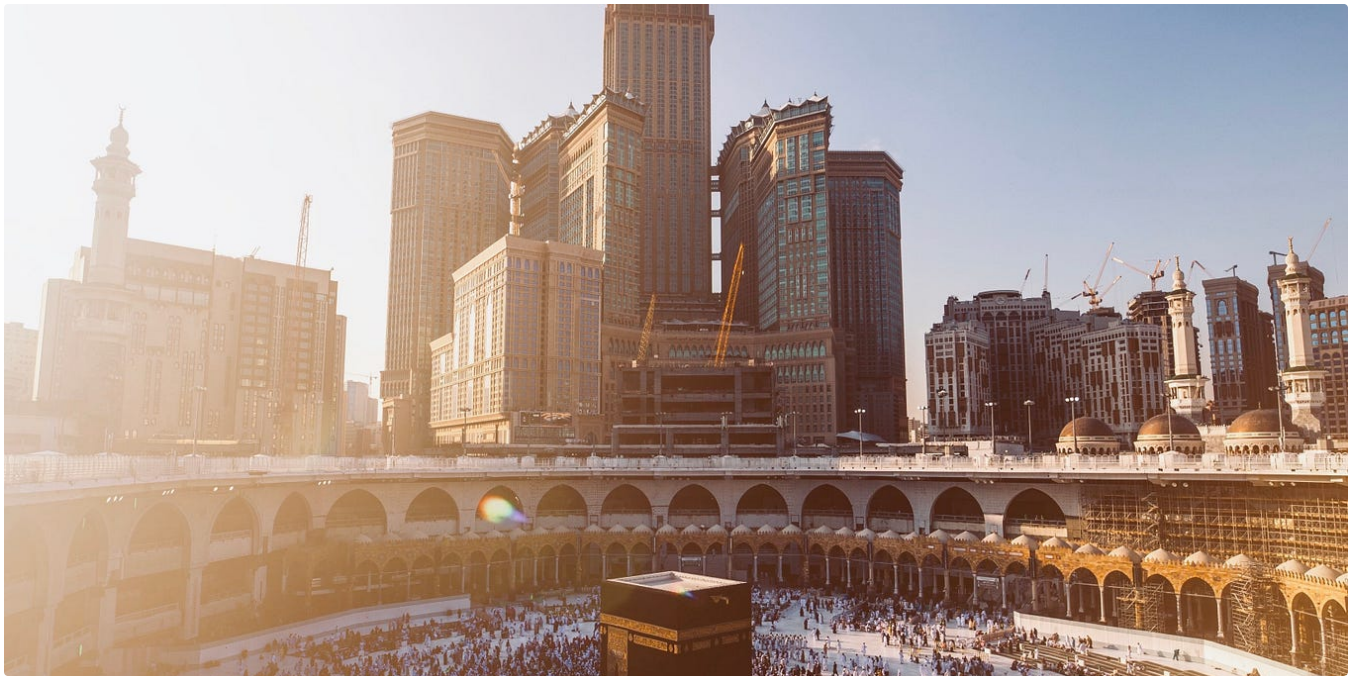
Neel Thakurdas

What actually affects the chances of a stolen base?

Exploring the factors that drive successful baseball base stealing strategies

7 min read · Aug 12

50

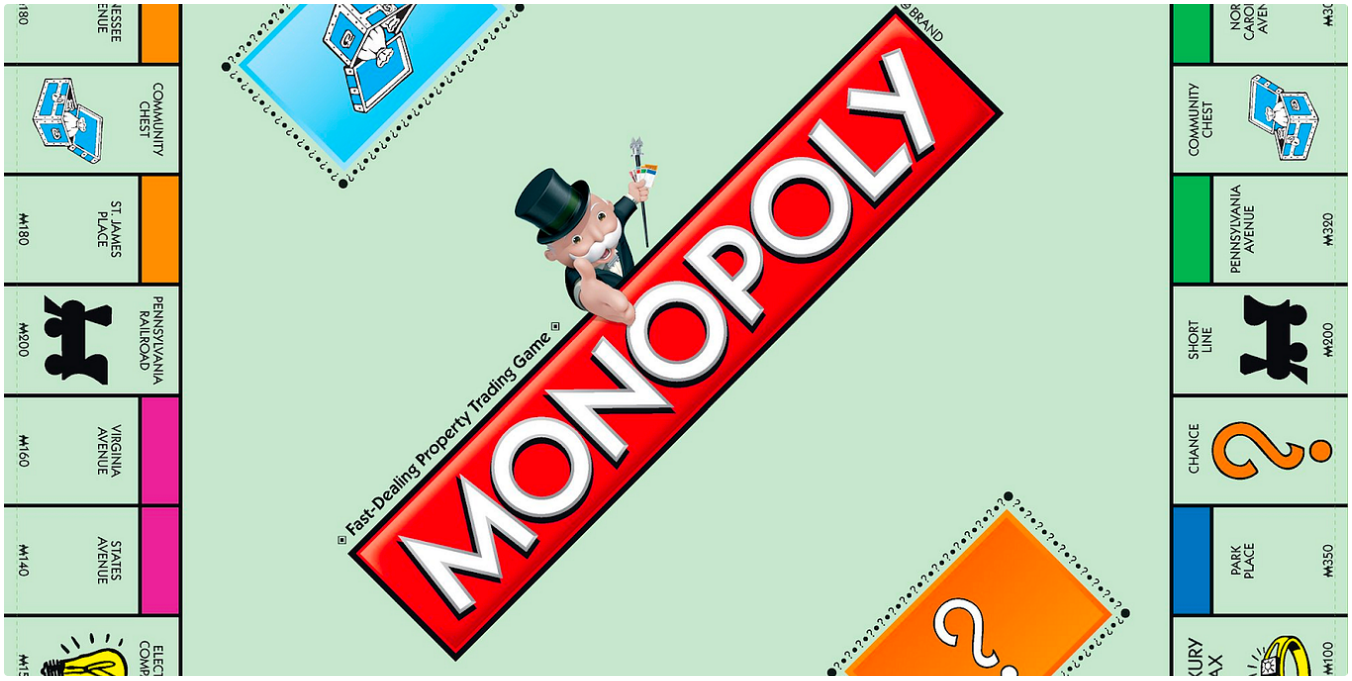


Neel Thakurdas

Why is Saudi Arabia investing so heavily in diversification?

In recent years, Saudi Arabia has directed substantial investments, amounting to billions of dollars, towards various sporting events like...

3 min read · Aug 24



Neel Thakurdas

How to win Monopoly: A guide on how to lose friends

Monopoly, the iconic board game that has graced countless living room tables for decades. But beneath its charming facade of colorful...

8 min read · Sep 11

Neel Thakurdas

Become a Property Tycoon in the Big Apple

Analyzing Housing Sales in New York City

5 min read · Aug 23

Recommended from Medium

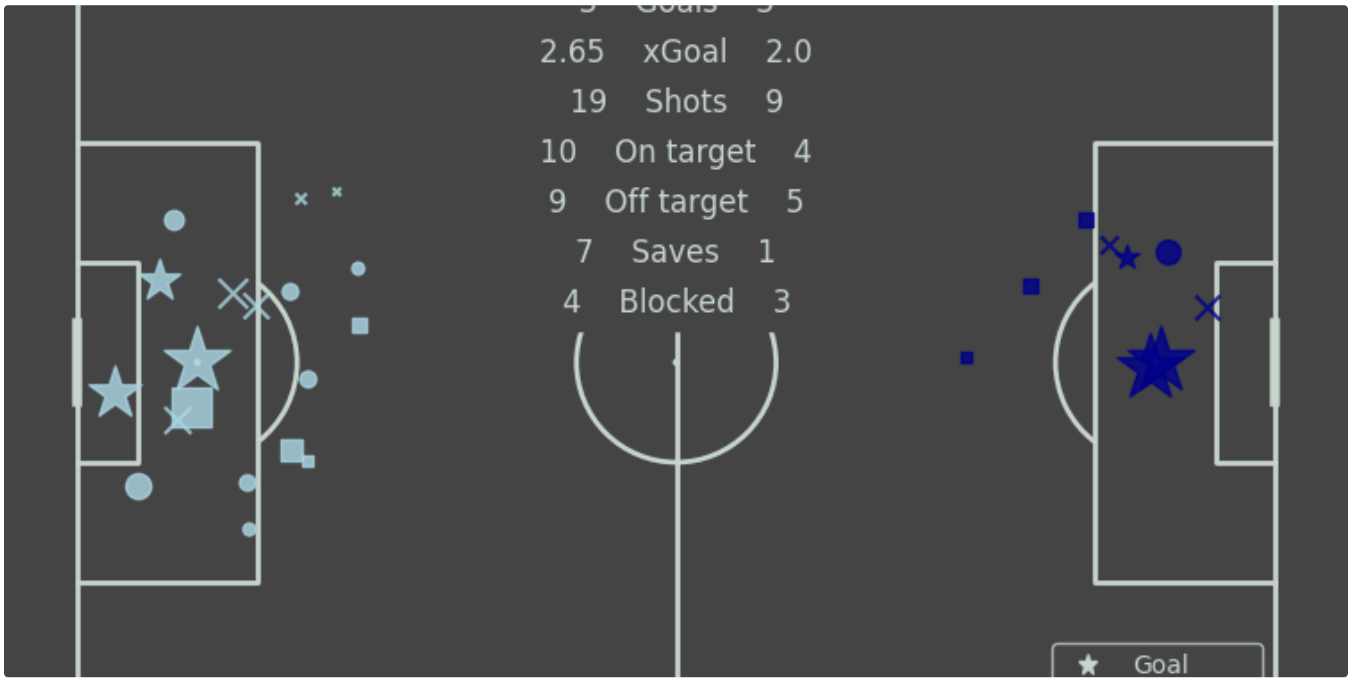
Federico Marchi

Football and PCA: A Step-by-Step Guide Analyzing Serie A Teams' data

Learning Data Science with football data: episode 1

8 min read · May 16

29



Roland Kovács

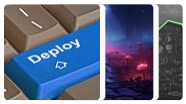
Visualising shots using football match event data in Python

This article describes how you can create you own shot map in Python using football match event data.

8 min read · Apr 25

3 1

Lists



Predictive Modeling w/ Python

20 stories · 397 saves

New_Reading_List

174 stories · 110 saves

Practical Guides to Machine Learning

10 stories · 457 saves

Coding & Development

11 stories · 180 saves

Jorge Mendoza Rivilla

Collecting data for football analysis

Data resources for performance analyst or enthusiastic about Data Football.

2 min read · May 1

14

Away_scoring		0	1	2	3	4	5
Home_scoring							
0	0	0.021	0.021	0.011	0.004	0.001	0.000
1	1	0.065	0.065	0.035	0.012	0.002	0.000
2	2	0.087	0.087	0.046	0.016	0.003	0.001
3	3	0.081	0.081	0.043	0.015	0.003	0.001
4	4	0.057	0.057	0.030	0.011	0.002	0.000

Aritra

Football Odds data analysis using MonteCarlo simulation in Python—Part 2

Introduction

6 min read · Mar 21

10

Matt Chapman in Towards Data Science

The Portfolio that Got Me a Data Scientist Job

Spoiler alert: It was surprisingly easy (and free) to make

· 10 min read · Mar 24

Brentford	11	7	1	40	25	11	14
Brighton	7	9	3	30	27	21	6
Aston Villa	6	9	4	27	25	22	3
Tottenham	6	9	4	27	27	23	4
Chelsea	5	6	8	21	12	16	-4
Fulham	3	9	7	18	20	29	-9
West Ham	2	10	7	16	16	24	-8

Amrit Vignesh

Predicting Past 22/23 Premier League Home and Away Tables Using Statistical Principles with Comps

3 min read · Aug 19

 5 

See more recommendations