EPL Forecasting: A Mathematical Model for Predicting Premier League Standings in Future Seasons

Elisabeth G. Mortensen

Introduction

The English Premier League stands as one of the most watched and fiercely competitive football leagues globally, captivating millions of fans with its thrilling matches and unpredictable outcomes. As each season approaches and progresses, there's an interest in predicting standings, driven by both fervor of fans, sports analytics, and betting industries. The aim is to build a model that leverages historical data and autoregression to predict end of season standings. Then by applying this model to previous seasons, compare the predictions with real-world final standings to see how well the model does. Ultimately, we want to simulate a final standing of the current 2023-24 season.

Background Information

What makes Premier League such a rich sport is the power of the standings. Securing a spot in the top four not only earns prestige but also grants coveted access to the UEFA Champions League. Conversely, the bottom three teams face the daunting prospect of relegation, leaving the league and moving down to the Championship. The points you earn throughout the season decide your place in the standings and with the 1981 implementation of the 'three points for a win' system, wins have higher value in comparison to draws so teams with more wins are more likely to rank higher than those with less wins but more draws. As teams make their way through the double round robin format of the season these points act the ultimate arbiter and an important metric for the model.

Autoregressive models are helpful tools in the statistical prediction world. Autoregression can be used to model temporal dependencies in time series data, such as historical match results and team/player performance (Oved, 2021).

Modeling Approaches

In this paper, we attempt to use statistical methods to model and predict final standings of the Premier League using 20 years of match data. This data is collected from football-data.co.uk. We are able to combine 20 seasons provided from them into one large data set. From there we added features such as columns for indicating season. The data includes several metrics beyond win/lose such as home goals, home shots, home fouls, etc.

First, we work with a simple autoregression that only considers total points of past seasons. We then determine other notable features in earning points, particularly, number of goals scored and then weigh recent data more heavily in regards to relevance. We then use further autoregression with those features in mind to predict the 2023-24 Premier League season standings, in particular top 4 teams and bottom 3 teams.

While we could test the accuracy of the model by simply comparing the predicted standing of each team in comparison to their actual standing for that season in a true/false manner, it would be beneficial to quantify it.

In order to do that, we take inspiration from the Levenshtein distance. The Levenshtein distance tests the difference between two strings by finding the minimum number of single-character edits it would take to go from one string to the other. We can apply this idea by looping through the predicted standing and comparing the index of each team to their index in the actual standings. Then we can calculate the absolute difference between those indices (e.g. Chelsea actual standing is 11th, prediction has Chelsea in 4th. The $distance = |11th\ place - 4th\ place| = 7\ places$) and sum up the difference from each team (e.g. $Chelsea\ distance + Man\ United\ distance = 7\ places + 5\ places = 12\ places$). The lower that number is, the closer the predicted standing is to the actual. As an interesting data insight exercise, we ran the distance of each of the 20 seasons in the dataset, to see what inherent movement (volatility) one should expect from season to season. Mostly to get a feeling for the likely shakeup in a season prediction.

To further understand the value of this "distance" metric, we can consider a random shuffle of a season. For instance, for the 2020-21 season we can take a completely random shuffle of the teams playing and calculate the "distance". In the example shown below, we obtain a value of 144. That is to say that our random "prediction" is 144 team movements away from the actual standings. Randomness is obviously a bit aggressive and extremely noisy, but it gives us a decent baseline for what a very poor model (random) can produce. If the "distance" value we obtain from our subsequent models is similar to that of a random shuffle of teams, our models wouldn't be any good.

	Actual Team	Random Team
1	Man City	Leeds
2	Man United	Newcastle
3	Liverpool	West Brom
4	Chelsea	Sheffield United
5	Leicester	Everton
6	West Ham	Man City
7	Tottenham	Arsenal
8	Arsenal	Crystal Palace
9	Everton	Tottenham
10	Leeds	Man United
11	Aston Villa	Burnley
12	Newcastle	Liverpool
13	Wolves	West Ham
14	Crystal Palace	Fulham
15	Southampton	Leicester
16	Brighton	Chelsea
17	Burnley	Southampton
18	Fulham	Brighton
19	West Brom	Wolves
20	Sheffield United	Aston Villa

Comparison of actual 2020-21 standings to random 2020-21 standings

Lastly, because of the significance of the top 4 and bottom 3 places of the standings, we will consider those predictions with more scrutiny.

Derivation of Model

Before we make any considerations for the 2023-24 season, we will build and test the model using the previous seasons 2020-21, 2021-22, and 2022-23. This way we can test the accuracy of the model against actual outcomes (leaving out the data for the season we try to predict). The basis of the model is a point count. We start by considering the 2020-21 season and the teams playing during that season. Below is a table displaying the total point count per team per season, it's these values that we will work with.

Season	Man City	Man United	Liverpool	Chelsea	Leicester	West Ham	Tottenham	Arsenal	Everton	Leeds	Aston Villa	Newcastle	Wolves	Crystal Palace	Southampton	Brighton	Burnley	Fulham	West Brom	Sheffield United
S0304	41	75	60	79	33	0	45	90	39	33	56	56	33	0	47	0	0	52	0	0
S0405	52	77	58	95	0	0	52	83	61	0	47	44	0	33	32	0	0	44	34	0
S0506	43	83	82	91	0	55	65	67	50	0	42	58	0	0	0	0	0	48	30	0
S0607	42	89	68	83	0	41	60	68	58	0	50	43	0	0	0	0	0	39	0	38
S0708	55	87	76	85	0	49	46	83	65	0	60	43	0	0	0	0	0	36	0	0
S0809	50	90	86	83	0	51	51	72	63	0	62	34	0	0	0	0	0	53	32	0
S0910	67	85	63	86	0	35	70	75	61	0	64	0	38	0	0	0	30	46	0	0
S1011	71	80	58	71	0	33	62	68	54	0	48	46	40	0	0	0	0	49	47	0
S1112	89	89	52	64	0	0	69	70	56	0	38	65	25	0	0	0	0	52	47	0
S1213	78	89	61	75	0	46	72	73	63	0	41	41	0	0	41	0	0	43	49	0
S1314	86	64	84	82	0	40	69	79	72	0	38	49	0	45	56	0	0	32	36	0
S1415	79	70	62	87	41	47	64	75	47	0	38	39	0	48	60	0	33	0	44	0
S1516	66	66	60	50	81	62	70	71	47	0	17	37	0	42	63	0	0	0	43	0
S1617	78	69	76	93	44	45	86	75	61	0	0	0	0	41	46	0	40	0	45	0
S1718	100	81	75	70	47	42	77	63	49	0	0	44	0	44	36	40	54	0	31	0
S1819	98	66	97	72	52	52	71	70	54	0	0	45	57	49	39	36	40	26	0	0
S1920	81	66	99	66	62	39	59	56	49	0	35	44	59	43	52	41	54	0	0	54
S2021	86	74	69	67	66	65	62	61	59	59	55	45	45	44	43	41	39	28	26	23
S2122	93	58	92	74	52	56	71	69	39	38	45	49	51	48	40	51	35	0	0	0
S2223	89	75	67	44	34	40	60	84	36	31	61	71	41	45	25	62	0	52	0	0

Total Point Count per Team Per Season (2003-2023)

The gradient represents a team's points per season, with darker blue indicating seasons where the team earned more points and lighter colors showing seasons with fewer points. We use all this data and complete a simple autoregression only taking into consideration total points per team per season. With that we predict the point outcome of each team in the 2020-21 season and rank them from highest to lowest. Our autoregressions are of the form $Y_t = \phi Y_{t-1} + \epsilon_t$ in which we predict Y_t points based on the previous seasons Y_{t-1} points.

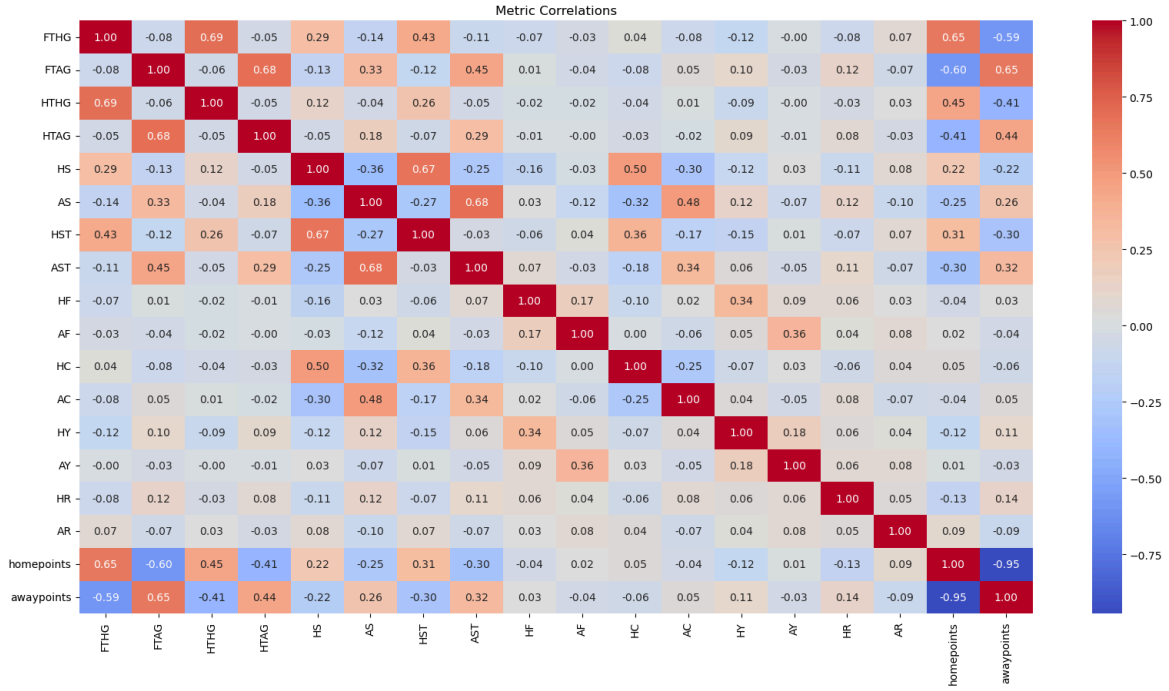
	Team	Autoregression	Predicted points
1	Liverpool	Y_t = 40.11 + 0.46 * Y_t-1	85.73
2	Man City	Y_t = 19.88 + 0.75 * Y_t-1	80.31
3	Chelsea	Y_t = 84.32 + -0.08 * Y_t-1	79.31
4	Man United	Y_t = 33.43 + 0.57 * Y_t-1	70.94
5	Arsenal	Y_t = 55.09 + 0.23 * Y_t-1	67.72
6	Tottenham	Y_t = 32.58 + 0.51 * Y_t-1	62.49
7	Everton	Y_t = 58.34 + -0.03 * Y_t-1	57.06
8	Leicester	Y_t = 5.49 + 0.8 * Y_t-1	55.25
9	Southampton	Y_t = 5.15 + 0.82 * Y_t-1	47.56
10	Brighton	Y_t = 2.89 + 0.93 * Y_t-1	41.06
11	Burnley	Y_t = 8.34 + 0.6 * Y_t-1	40.57
12	Wolves	Y_t = 6.8 + 0.57 * Y_t-1	40.48
13	West Ham	Y_t = 30.86 + 0.24 * Y_t-1	40.20
14	Newcastle	Y_t = 36.37 + 0.08 * Y_t-1	39.79
15	Crystal Palace	Y_t = 7.84 + 0.73 * Y_t-1	39.10
16	Aston Villa	Y_t = 6.38 + 0.8 * Y_t-1	34.21
17	West Brom	Y_t = 17.7 + 0.35 * Y_t-1	17.70
18	Fulham	Y_t = 2.88 + 0.81 * Y_t-1	2.88
19	Leeds	Y_t = 0.0 + 0.0 * Y_t-1	0.00
20	Sheffield United	Y_t = 6.13 + -0.16 * Y_t-1	-2.58

Predicted points per team for 2020-21 season with Autoregression

It's important to clarify that although our prediction is based on points, it does not forecast the exact number of points a team will earn. Instead, it evaluates the team's performance relative to others for ranking purposes. Think of it as a prediction score. That is to say in the prediction above, we do not predict Chelsea to earn 79 points in the 2020-21 season rather we predict Chelsea will earn a value of points that places them in 3rd place compared to the other teams.

Now we can apply the accuracy testing mentioned earlier inspired by Levenshtein distance. When we follow that method of comparing the prediction to the actual standings for 2020-21 season we obtain a value of 60. We can pinpoint some team movements predicted by the autoregression that increased this value such as the prediction of Southampton in 9th while actually placing in 15th. When we compare that to the distance value of 144 for a random shuffle of the teams, that accuracy doesn't seem too bad but we should try to make it better.

To expand on our autoregression we built a correlation matrix heat-map of several different metrics to consider in a football match. By considering the relationship of different finer metrics we can gain further insight into what plays a part in the points a team earns.



Correlation matrix heat-map of finer metrics in Premier League (2003-2023)

From this we can point out some interesting insights. For instance, HS (home team shots) and HC (home team corners) have a relatively strong positive correlation of 0.5, same goes for AS (away team shots) and AC (away team corners). However, we don't see a notable relationship between HS (home team shots) and home team points with a correlation of 0.22, same goes for AS (away team shots) and away team points with a correlation of 0.26. We see a similar pattern of low correlations between points and other metrics. This is not to say that these finer metrics do not affect the games and in turn the points a team is likely to earn. If we found a very strong correlation between corners and points it would suggest that it was a game of corners. Any decent correlation with home points and away points suggests it aids to some extent even if not "strongly" and that these little impacts imply varied game.

With that said, we do see a good correlation between FTHG (full time home goals) and home team points, similarly with FTAG (full time away goals) and away team points. While this seems obvious, having many goals does not guarantee a 3 point win.This serves as an Inspiration for adjusting our autoregression by adding a weight to the goals scored by teams.

An article by The Athletic speaks on how we can value goals when taking into consideration things like how much time has elapsed in a game and the type of game state-changing goal scored (equalizing or putting their team ahead). One of their calculations is of the highest average points earned per goal from the top 50 goal scorers in the Premier League when considering decisive goals that

influence the final score (Cary, 2022). Based on their calculation, we can test a 0.25 average point value per goal scored.

To take this goal weight into account we recalculate match points. We keep the 3 points for a win, 1 point for a draw, and 0 for a loss. However, for both the home team and away team we add to those points $0.25\ points\ per\ goal \times (full\ time\ goal\ count) = additional\ points$. With this we have a new total point count for the 2020-21 season participating teams with a goal weight which we can perform an autoregression with.

	Team	Autoregression	Predicted points
1	Man City	Y_t = 27.87 + 0.77 * Y_t-1	129.24
2	Liverpool	Y_t = 63.95 + 0.41 * Y_t-1	122.49
3	Chelsea	Y_t = 139.47 + -0.22 * Y_t-1	116.93
4	Arsenal	Y_t = 97.64 + 0.09 * Y_t-1	105.18
5	Man United	Y_t = 44.74 + 0.6 * Y_t-1	104.43
6	Tottenham	Y_t = 47.62 + 0.51 * Y_t-1	93.44
7	Leicester	Y_t = 7.72 + 0.81 * Y_t-1	85.44
8	Everton	Y_t = 85.59 + -0.03 * Y_t-1	83.16
9	Southampton	Y_t = 8.4 + 0.8 * Y_t-1	70.51
10	West Ham	Y_t = 47.0 + 0.23 * Y_t-1	61.87
11	Brighton	Y_t = 4.09 + 0.95 * Y_t-1	61.85
12	Newcastle	Y_t = 56.64 + 0.04 * Y_t-1	58.98
13	Wolves	Y_t = 10.24 + 0.55 * Y_t-1	57.05
14	Burnley	Y_t = 12.73 + 0.56 * Y_t-1	55.09
15	Aston Villa	Y_t = 10.66 + 0.78 * Y_t-1	54.04
16	Crystal Palace	Y_t = 12.02 + 0.71 * Y_t-1	53.32
17	West Brom	Y_t = 28.1 + 0.32 * Y_t-1	28.10
18	Fulham	Y_t = 5.32 + 0.79 * Y_t-1	5.32
19	Leeds	Y_t = 0.0 + 0.0 * Y_t-1	0.00
20	Sheffield United	Y_t = 8.5 + -0.16 * Y_t-1	-3.07

Predicted points per team for 2020-21 season with Goal Weight Autoregression

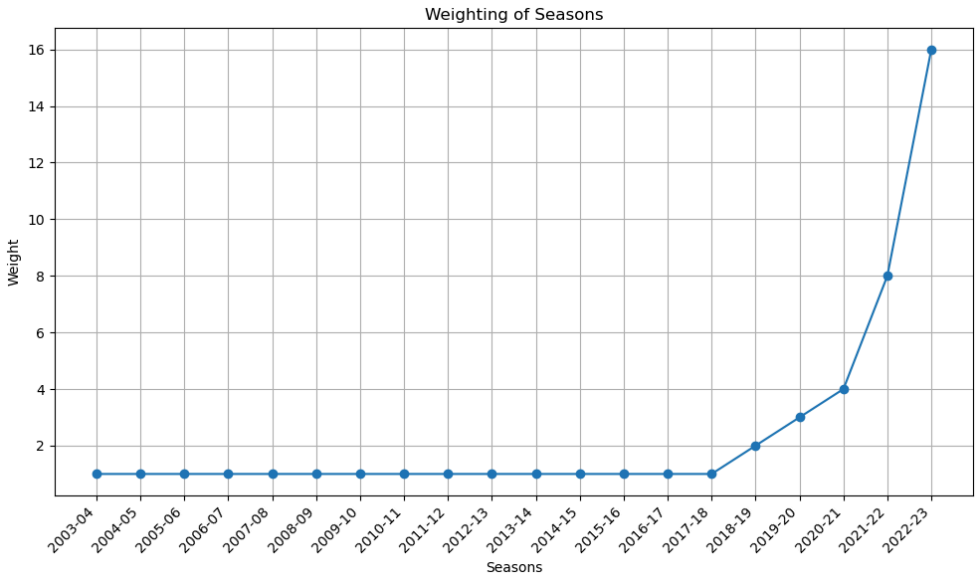
To reiterate, we are not predicting specific number of points a team will earn especially now with the added goal weight, we focus on their place in the standings, which is why a negative number is not an issue. When we perform the accuracy distance we obtain a value of 48. This is already much better than the autoregression we performed that only took into consideration history of total points where we had an accuracy value of 60.

To go further with this model we can take into consideration how recent data is likely to provide a better look into what a future season may look like. The teams we study in our historical data back in the 2003/2004 season are not the same teams we see in the 2023/2024 season. Clubs and their talent can change drastically over just a few years other or more examples of changes overtime that can occur. By

reducing the weight of this now less reliable data, we can hold on to valuable data while recognizing how the random noise that this older contains should be reduced (Dorhauer, 2017).

Because our model considers 20 individual seasons and not 7,600 individual games, weighting with exponential decay to not be particularly useful. Rather, we take a more simplistic approach in which that we consider the most recent 5 seasons to be the most important (Nguyen, 2021). The weighting is as follows: Every season from 2003-04 to 2017-18 are given weight 1. The weight then increases by 1 for each one of 2018-19, 2019-20, and 2020-21. After that, for 2021-22 the weight is multiplied by 2, and then multiplied by 2 again for 2022-23 season (Nguyen, 2021).

For our model we apply a similar process. For the 2023-24 season, the weights are assigned as follows: Seasons from 2003-04 to 2017-18 each receive a weight of 1. The weight then increases by 1 for each subsequent season—2018-19, 2019-20, and 2020-21. For the 2021-22 season, the weight is doubled, and it is doubled again for the 2022-23 season. With these adjustments, we can now conduct an autoregression that takes into account both goal weight and the recency of the data. We can adjust the weights accordingly when applying them to autoregressions of previous seasons.



Weight values across seasons from 2003-04 to 2022-23

	Team	Autoregression	Predicted points
1	Liverpool	Y_t = -84.02 + 2.1 * Y_t-1	1935.60
2	Man City	Y_t = -39.84 + 1.69 * Y_t-1	1397.63
3	Leicester	Y_t = -10.23 + 2.0 * Y_t-1	1249.03
4	Man United	Y_t = -77.31 + 1.87 * Y_t-1	1159.39
5	Chelsea	Y_t = -69.83 + 1.8 * Y_t-1	1131.95
6	Wolves	Y_t = 7.92 + 1.9 * Y_t-1	1098.18
7	Southampton	Y_t = -22.37 + 1.99 * Y_t-1	1009.48
8	Burnley	Y_t = 4.68 + 1.77 * Y_t-1	922.60
9	Tottenham	Y_t = -35.38 + 1.6 * Y_t-1	915.74
10	Arsenal	Y_t = -60.48 + 1.71 * Y_t-1	898.06
11	Everton	Y_t = -50.6 + 1.82 * Y_t-1	824.30
12	Brighton	Y_t = 9.16 + 1.8 * Y_t-1	739.69
13	Newcastle	Y_t = -14.29 + 1.55 * Y_t-1	647.36
14	Crystal Palace	Y_t = 1.16 + 1.53 * Y_t-1	623.33
15	West Ham	Y_t = -9.47 + 1.51 * Y_t-1	608.27
16	Fulham	Y_t = 45.09 + -0.04 * Y_t-1	45.09
17	West Brom	Y_t = 29.03 + 0.32 * Y_t-1	29.03
18	Leeds	Y_t = 0.0 + 0.0 * Y_t-1	0.00
19	Aston Villa	Y_t = 92.04 + -0.58 * Y_t-1	-117.13
20	Sheffield United	Y_t = 37.07 + -0.81 * Y_t-1	-373.89

Predicted points per team for 2020-21 season with Goal and Recency Weight Autoregression

The results of applying our model to the 2020-21 season gives us a “distance” value of 72. We did not accurately predict the winner of the Premier League title. We accurately predicted 3 of the top 4 teams and 1 of the bottom 3 teams.

Now to further test the accuracy of our model we apply the above methods and final autoregression with goal and recency weight to the 2021-22 and 2022-23 seasons. The results of applying our model to the 2021-22 season gives us a “distance” value of 78. We did not accurately predict the winner of the Premier League title. We accurately predict 1 of the top 4 teams and 1 of the bottom 3 teams.The results of applying our model to the 2022-23 season gives us a “distance” value of 84. We accurately predict the winner of the Premier League title. We accurately predict 1 of the top 4 teams and none of the bottom 3 teams.

It should be noted that despite having 20 years of data to study, because certain teams have only been in the league for a short period of time, their autoregressions are likely to cause outliers in the prediction whether it's underestimating their points or overestimating them.

Results

Now that we have built a model that is satisfactory in predicting the final standings of the Premier League seasons, we can apply the model to now predict the standings of the 2023-24 season.

	Team	Predicted points
1	Arsenal	2191.49
2	Newcastle	1706.60
3	Man City	1653.60
4	Brentford	1470.50
5	Man United	1429.64
6	Brighton	1424.76
7	Aston Villa	1106.90
8	Tottenham	1097.72
9	Liverpool	1048.65
10	Crystal Palace	713.07
11	Wolves	566.00
12	Chelsea	535.52
13	West Ham	524.39
14	Everton	496.12
15	Bournemouth	115.78
16	Nott'm Forest	20.00
17	Burnley	15.63
18	Sheffield United	9.09
19	Fulham	-405.19

Predicted points per team for 2023-24 season with Goal and Recency Weight Autoregression

As mentioned earlier, there is higher consequence in the top 4 and bottom 3 standings. Our model predicts the winner of the Premier League title in 1st place to be Arsenal. The 4 teams we predict to qualify for the Champions League are Arsenal, Newcastle, Manchester City, and Brentford. The 3 teams we predict to be relegated are Burnley, Sheffield United, and Fulham.

The standings above only include 19 of the 20 teams. We made a choice to not include Luton Town in the 2023/2024 autoregression. Of the three teams promoted for the 23/34 season, Luton Town is the only team who has never been present in the 20 years of Premier League historical data applied in the model. Because of this any of their historical data for the past 20 years will be from lower divisions of English football that do not relate to the model. So instead let us consider their likelihood of being relegated due to their newcomer status. In a model done by Ryan Alty, of the 3 teams promoted each season, between 1.35 and 2.05 teams will suffer relegation the season after achieving promotion (2019). Considering Luton Town's lack of experience in the Premier League and the probability of relegation of

newly promoted teams, we can make the assumption that Luton Town would be in the bottom three of the standings.

Discussion

Predicting Premier League match outcomes and standings consists of a large audience. How these models are built and what metrics/features are considered in the model depends on who's making it. For those in the betting industry, their models are the best of the best and are required to be with what is at stake.

For curiosity and interest, we tested our models predictions of the 2020-21, 2021-22, and 2022-23 seasons alongside the predictions of The Guardian for those same seasons. The Guardian's predictions for the 2020-21 season had a distance value of 56. They missed the league winner but got all top 4 and 2 of the bottom 3 right (Wilson, 2021). In 2021-22, their distance value was 60. They got the league winner right, along with 3 of the top 4 and 2 of the bottom 3 (Wilson, 2022). In 2022-23, the distance value was 94. They got the league winner but only 1 of the top 4 and 1 of the bottom 3 correct (Wilson, 2023).When studied, The Guardian’s accuracy of predictions is not very far off from our models predictions.

We can study the matches, the teams, and all the finer metrics, but ultimately the volatility and unpredictability of the Premier League is hard to predict. This is visible in the accuracy of our models. The 2022-23 season illustrates this well, particularly with teams like Chelsea that saw a significant and surprising drop in their standings. This decline was influenced by factors including heavy spending (team overhaul), changes in management, and a recent change in ownership.

For future work and improvement in this model, work in Monte Carlo methods can be used to obtain prediction that account for uncertainty in the metrics studied. Additionally, performance of the model can also be improved with Poisson Modeling and perhaps expanding the use of historical data from season by season to individual matches. Additionally, from testing we did for our model with previous seasons, the recency weighting could improve. Exploring a logistic weighting could be a next step for improvement.

References

Alty, R. (2019). The Probability of Relegation in the First Three Seasons in The Premier League. *Leicester Undergraduate Mathematical Journal*, *1*. https://journals.le.ac.uk/ojs1/index.php/lumj/article/view/3440/3065

Cary, M., & Whitehead, J. (2022). *Can we measure the value of a goal?* The Athletic. https://theathletic.com/3244648/2022/04/24/can-we-measure-the-value-of-a-goal/

Dorhauer, A., & McDaniel, R. (2017). *The Math of Weighting Past Results | The Hardball Times*. The Hardball Times. https://tht.fangraphs.com/the-math-of-weighting-past-results/

football-data.co.uk. (n.d.). *Data Files: England*. https://www.football-data.co.uk/englandm.php

Nguyen, Q. (2021). Poisson Modeling and Predicting English Premier League Goal Scoring. *arXiv.org*, (Ithaca: Cornell University Library). https://doi.org/10.48550/arxiv.2105.09881

Oved, N., Feder, A., & Reichart, R. (2020). Predicting in-game actions from interviews of NBA players. (). Ithaca: Cornell University Library, arXiv.org. https://doi.org/10.48550/arxiv.1910.11292

Wilson, J. (2021). *Premier League 2020-21 season review: our predictions versus reality*. The Guardian. https://www.theguardian.com/football/2021/may/25/premier-league-2020-21-season-review-our-predictions-versus-reality

Wilson, J. (2022). *Premier League 2021-22 season review: our predictions versus reality*. The Guardian. https://www.theguardian.com/football/2022/may/27/premier-league-2021-22-season-review-our-predictions-versus-reality

Wilson, J. (2023). *Premier League 2022-23 season review: our predictions versus reality*. The Guardian. https://www.theguardian.com/football/2023/may/30/premier-league-2022-23-season-review-our-predictions-versus-reality

Appendix

Standings for each season (2003-2023)

In [17]:
```
import pandas as pd

file_path = 'pm_stats.csv'
data = pd.read_csv(file_path)

def calculate_match_points(row):
    if row['FTR'] == 'H':
        return {row['HomeTeam']: 3, row['AwayTeam']: 0}
    elif row['FTR'] == 'A':
        return {row['HomeTeam']: 0, row['AwayTeam']: 3}
    elif row['FTR'] == 'D':
        return {row['HomeTeam']: 1, row['AwayTeam']: 1}
    else:
        return {row['HomeTeam']: 0, row['AwayTeam']: 0}

match_points = data.apply(calculate_match_points, axis=1)
points_df = pd.DataFrame(match_points.tolist())

points_summary = points_df.fillna(0).groupby(data['Season']).sum()

seasonal_team_lists = {}

for season in points_summary.index:
    season_data = points_summary.loc[season]
    valid_teams = season_data[season_data > 0].sort_values(ascending=False)
    seasonal_team_lists[season] = list(valid_teams.index)

#seasonal_team_lists
```

Total points per season per team for 20/21, 21/22, and 22/23

In [14]:
```
import pandas as pd
import dataframe_image as dfi

file_path = 'pm_stats.csv'
data = pd.read_csv(file_path)

specified_teams_2021 = ['Man City','Man United','Liverpool','Chelsea','Leiceste
                        'West Ham','Tottenham','Arsenal','Everton','Leeds',
                        'Aston Villa','Newcastle','Wolves','Crystal Palace','Sc
                        'Brighton','Burnley','Fulham','West Brom','Sheffield Un
specified_teams_2122 = ['Man City','Liverpool', 'Chelsea','Tottenham','Arsenal
                        'Man United','West Ham','Leicester','Wolves','Brighton
                        'Newcastle','Crystal Palace','Brentford','Aston Villa',
                        'Southampton','Everton','Leeds','Burnley','Watford','No
specified_teams_2223 = ['Man City','Arsenal','Man United','Newcastle','Liverpoo
                        'Brighton','Aston Villa','Tottenham','Brentford','Fulha
```

```
                        'Crystal Palace','Chelsea','Wolves','West Ham','Bourner
                        "Nott'm Forest",'Everton','Leicester','Leeds','Southam

def calculate_match_points(row):
    if row['FTR'] == 'H':
        return {row['HomeTeam']: 3, row['AwayTeam']: 0}
    elif row['FTR'] == 'A':
        return {row['HomeTeam']: 0, row['AwayTeam']: 3}
    elif row['FTR'] == 'D':
        return {row['HomeTeam']: 1, row['AwayTeam']: 1}
    else:
        return {row['HomeTeam']: 0, row['AwayTeam']: 0}

match_points = data.apply(calculate_match_points, axis=1)
points_df = pd.DataFrame(match_points.tolist())

points_summary = points_df.fillna(0).groupby(data['Season']).sum()
filtered_points_summary_2021 = points_summary[specified_teams_2021]
filtered_points_summary_2122 = points_summary[specified_teams_2122]
filtered_points_summary_2223 = points_summary[specified_teams_2223]

styled_df = filtered_points_summary_2021.style.background_gradient().format("{
dfi.export(styled_df, 'dataframe_styled_2021.png')
styled_df = filtered_points_summary_2122.style.background_gradient().format("{
dfi.export(styled_df, 'dataframe_styled_2122.png')
styled_df = filtered_points_summary_2223.style.background_gradient().format("{
dfi.export(styled_df, 'dataframe_styled_2223.png')
```

Random Standings for 20/21, 21/22, and 22/23

In [8]:

```
import random

season1 = seasonal_team_lists.get('S2021', [])
season2 = seasonal_team_lists.get('S2122', [])
season3 = seasonal_team_lists.get('S2223', [])

random_sorts_2021 = []
random_list_1 = season1.copy()
random.shuffle(random_list_1)
random_sorts_2021.append(random_list_1)
print('Random 20/21 standings:', random_sorts_2021)

random_sorts_2122 = []
random_list_2 = season2.copy()
random.shuffle(random_list_2)
random_sorts_2122 = [random_list_2]
print('Random 21/22 standings:', random_sorts_2122)

random_sorts_2223 = []
random_list_3 = season3.copy()
random.shuffle(random_list_3)
random_sorts_2223 = [random_list_3]
print('Random 22/23 standings:', random_sorts_2223)
```

```
Random 20/21 standings: [['Sheffield United', 'Tottenham', 'Liverpool', 'Burnl
ey', 'Southampton', 'Leicester', 'West Brom', 'Brighton', 'Arsenal', 'Man Unit
ed', 'Wolves', 'Aston Villa', 'Chelsea', 'Man City', 'Fulham', 'Leeds', 'West
Ham', 'Everton', 'Crystal Palace', 'Newcastle']]
Random 21/22 standings: [['Aston Villa', 'Chelsea', 'Brighton', 'Wolves', 'Man
City', 'Burnley', 'Newcastle', 'Liverpool', 'Watford', 'Tottenham', 'Norwich',
'West Ham', 'Leeds', 'Leicester', 'Southampton', 'Crystal Palace', 'Everton',
'Arsenal', 'Brentford', 'Man United']]
Random 22/23 standings: [["Nott'm Forest", 'Man City', 'Tottenham', 'Crystal P
alace', 'Brentford', 'Chelsea', 'Leeds', 'Liverpool', 'West Ham', 'Arsenal',
'Man United', 'Leicester', 'Southampton', 'Aston Villa', 'Fulham', 'Brighton',
'Wolves', 'Everton', 'Bournemouth', 'Newcastle']]
```

Distance Values for Random to Actual standings for 20/21, 21/22, and 22/23

In [9]:

```
season1 = seasonal_team_lists.get('S2021', [])
season2 = seasonal_team_lists.get('S2122', [])
season3 = seasonal_team_lists.get('S2223', [])

random_season1 = ['Sheffield United', 'Tottenham', 'Liverpool', 'Burnley', 'Sou
random_season2 = ['Aston Villa', 'Chelsea', 'Brighton', 'Wolves', 'Man City',
random_season3 = ["Nott'm Forest", 'Man City', 'Tottenham', 'Crystal Palace',

distance1 = 0
distance2 = 0
distance3 = 0

for team in random_season1:
    index1 = season1.index(team)
    index2 = random_season1.index(team)
    distance1 += abs(index1 - index2)
print('Distance value for random 20/21: ', distance1)

for team in random_season2:
    index1 = season2.index(team)
    index2 = random_season2.index(team)
    distance2 += abs(index1 - index2)
print('Distance value for random 21/22: ', distance2)

for team in random_season3:
    index1 = season3.index(team)
    index2 = random_season3.index(team)
    distance3 += abs(index1 - index2)
print('Distance value for random 22/23: ', distance3)
```

```
Distance value for random 20/21:  144
Distance value for random 21/22:  130
Distance value for random 22/23:  134
```

Autoregression for 20/21

In [18]:

```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import dataframe_image as dfi

predictions_2021 = pd.DataFrame(columns=['Team', 'Autoregression', 'Predicted
```

```
filtered_points_summary = points_summary[specified_teams_2021]
row_index = 1

for team_name in filtered_points_summary.columns:
    match_points = filtered_points_summary[team_name].dropna().reset_index(drop
    match_points = match_points[0:-3]
    match_points.index += 1

    Y = match_points

    if len(Y) > 1:
        model = AutoReg(Y, lags=1)
        model_fitted = model.fit()
        intercept = round(model_fitted.params[0], 2)
        lag_coefficient = round(model_fitted.params[1], 2)
        autoregression_equation = f'Y_t = {intercept} + {lag_coefficient} * Y_
        prediction = round(model_fitted.predict(start=len(Y), end=len(Y), dynam
        predictions_2021.loc[row_index] = [team_name, autoregression_equation,
        row_index += 1
    else:
        print(f'Not enough data to predict for {team_name}')

predictions_2021.sort_values(by='Predicted points', ascending=False, inplace=Tr
predictions_2021.reset_index(drop=True, inplace=True)
predictions_2021.index += 1
#dfi.export(predictions_2021.head(20), 'dataframe_autoreg2021.png')
predictions_2021
```

Out[18]:

	Team	Autoregression	Predicted points
1	Liverpool	Y_t = 40.11 + 0.46 * Y_t-1	85.73
2	Man City	Y_t = 19.88 + 0.75 * Y_t-1	80.31
3	Chelsea	Y_t = 84.32 + -0.08 * Y_t-1	79.31
4	Man United	Y_t = 33.43 + 0.57 * Y_t-1	70.94
5	Arsenal	Y_t = 55.09 + 0.23 * Y_t-1	67.72
6	Tottenham	Y_t = 32.58 + 0.51 * Y_t-1	62.49
7	Everton	Y_t = 58.34 + -0.03 * Y_t-1	57.06
8	Leicester	Y_t = 5.49 + 0.8 * Y_t-1	55.25
9	Southampton	Y_t = 5.15 + 0.82 * Y_t-1	47.56
10	Brighton	Y_t = 2.89 + 0.93 * Y_t-1	41.06
11	Burnley	Y_t = 8.34 + 0.6 * Y_t-1	40.57
12	Wolves	Y_t = 6.8 + 0.57 * Y_t-1	40.48
13	West Ham	Y_t = 30.86 + 0.24 * Y_t-1	40.20
14	Newcastle	Y_t = 36.37 + 0.08 * Y_t-1	39.79
15	Crystal Palace	Y_t = 7.84 + 0.73 * Y_t-1	39.10
16	Aston Villa	Y_t = 6.38 + 0.8 * Y_t-1	34.21
17	West Brom	Y_t = 17.7 + 0.35 * Y_t-1	17.70
18	Fulham	Y_t = 2.88 + 0.81 * Y_t-1	2.88
19	Leeds	Y_t = 0.0 + 0.0 * Y_t-1	0.00
20	Sheffield United	Y_t = 6.13 + -0.16 * Y_t-1	-2.58

Autoregression for 21/22

In [25]:

```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import dataframe_image as dfi

predictions_2122 = pd.DataFrame(columns=['Team', 'Autoregression', 'Predicted |
filtered_points_summary = points_summary[specified_teams_2122]
row_index = 1

for team_name in filtered_points_summary.columns:
    match_points = filtered_points_summary[team_name].dropna().reset_index(dro|
    match_points = match_points[0:-2]
    match_points.index += 1
    Y = match_points

    if len(Y) > 1:
        model = AutoReg(Y, lags=1)
        model_fitted = model.fit()
        intercept = round(model_fitted.params[0], 2)
        lag_coefficient = round(model_fitted.params[1], 2)
        autoregression_equation = f'Y_t = {intercept} + {lag_coefficient} * Y_
```

```
        prediction = round(model_fitted.predict(start=len(Y), end=len(Y), dynar
        predictions_2122.loc[row_index] = [team_name, autoregression_equation,
        row_index += 1
    else:
        print(f'Not enough data to predict for {team_name}')

predictions_2122.sort_values(by='Predicted points', ascending=False, inplace=Tr
predictions_2122.reset_index(drop=True, inplace=True)
predictions_2122.index += 1
#dfi.export(predictions_2122.head(20), 'dataframe_autoreg2122.png')
#predictions_2122
```

Autoregression for 22/23

In [26]:
```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import dataframe_image as dfi

predictions_2223 = pd.DataFrame(columns=['Team', 'Autoregression', 'Predicted p
filtered_points_summary = points_summary[specified_teams_2223]
row_index = 1

for team_name in filtered_points_summary.columns:
    match_points = filtered_points_summary[team_name].dropna().reset_index(drop
    match_points = match_points[0:-1]
    match_points.index += 1
    Y = match_points

    if len(Y) > 1:
        model = AutoReg(Y, lags=1)
        model_fitted = model.fit()
        intercept = round(model_fitted.params[0], 2)
        lag_coefficient = round(model_fitted.params[1], 2)
        autoregression_equation = f'Y_t = {intercept} + {lag_coefficient} * Y_
        prediction = round(model_fitted.predict(start=len(Y), end=len(Y), dynar
        predictions_2223.loc[row_index] = [team_name, autoregression_equation,
        row_index += 1
    else:
        print(f'Not enough data to predict for {team_name}')

predictions_2223.sort_values(by='Predicted points', ascending=False, inplace=Tr
predictions_2223.reset_index(drop=True, inplace=True)
predictions_2223.index += 1
#dfi.export(predictions_2223.head(20), 'dataframe_autoreg2122.png')
#predictions_2223
```

Distance values for predicted and actual standings for 20/21, 21/22, and 22/23

In [42]:
```
season1 = seasonal_team_lists.get('S2021', [])
season2 = seasonal_team_lists.get('S2122', [])
season3 = seasonal_team_lists.get('S2223', [])

distance1 = 0
distance2 = 0
distance3 = 0
```

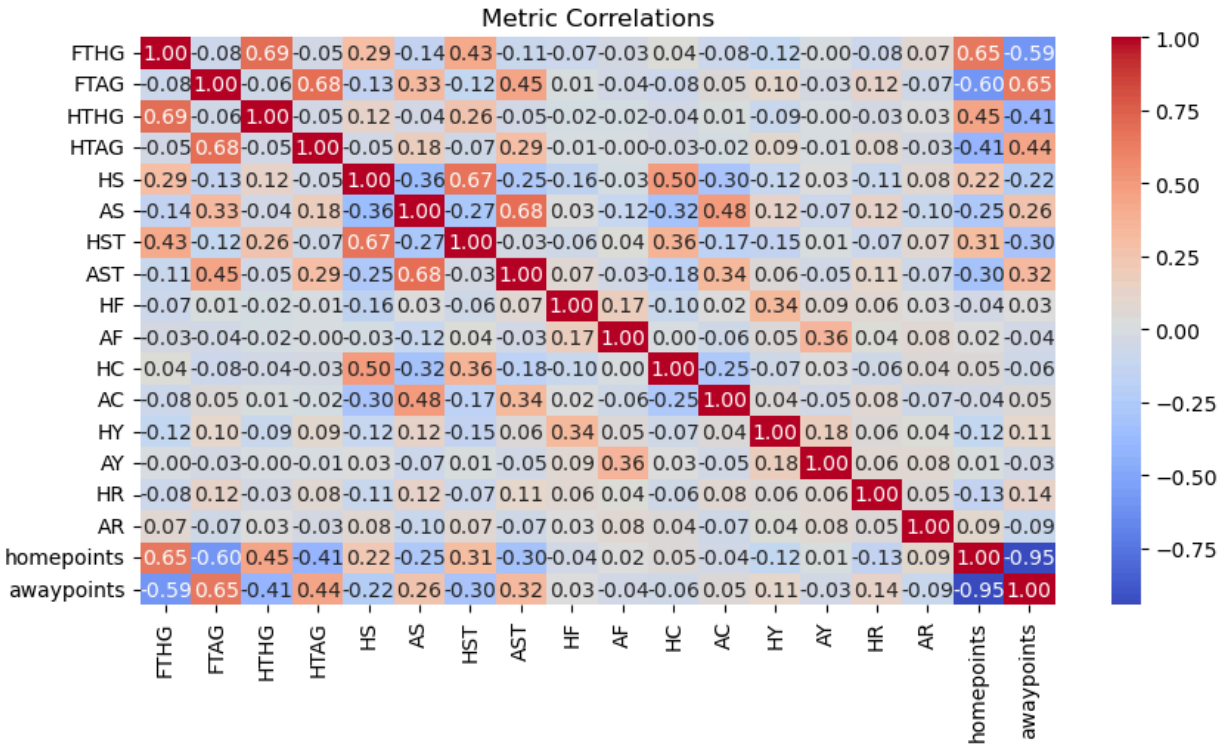
```
for team in predicted_season_2021:
    index1 = season1.index(team)
    index2 = predicted_season_2021.index(team)
    distance1 += abs(index1 - index2)
print('Distance value for predicted 20/21: ', distance1)

for team in predicted_season_2122:
    index1 = season2.index(team)
    index2 = predicted_season_2122.index(team)
    distance2 += abs(index1 - index2)
print('Distance value for predicted 21/22: ', distance2)

for team in predicted_season_2223:
    index1 = season3.index(team)
    index2 = predicted_season_2223.index(team)
    distance3 += abs(index1 - index2)
print('Distance value for predicted 22/23: ', distance3)
```

```
Distance value for predicted 20/21:  60
Distance value for predicted 21/22:  50
Distance value for predicted 22/23:  92
```

Correlaton Matrix Heat Map of Finer Metrics

In [20]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

file_path = 'pm_stats.csv'
data = pd.read_csv(file_path)

data['homepoints'] = data['FTR'].apply(lambda x: 3 if x == 'H' else 0 if x ==
data['awaypoints'] = data['FTR'].apply(lambda x: 3 if x == 'A' else 0 if x ==

specified_cols = ['FTHG', 'FTAG', 'HTHG', 'HTAG', 'HS', 'AS', 'HST', 'AST', 'HI
                  'HC', 'AC', 'HY', 'AY', 'HR', 'AR', 'homepoints', 'awaypoints

specified_dataset = data[specified_cols]

plt.figure(figsize=(10, 5))
sns.heatmap(specified_dataset.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Metric Correlations')
plt.show()
```



Total points per season with goal weight for 20/21, 21/22, and 22/23 teams

In [28]:

```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg

file_path = 'pm_stats.csv'
data = pd.read_csv(file_path)

gw = 0.25

def calculate_match_points(row):
    if row['FTR'] == 'H':
        return {row['HomeTeam']: 3 + row['FTHG'] * gw, row['AwayTeam']: 0 + row
    elif row['FTR'] == 'A':
        # Assuming away goals should also affect the calculation if needed
        return {row['HomeTeam']: 0 + row['FTHG'] * gw, row['AwayTeam']: 3 + row
    elif row['FTR'] == 'D':
        # In case of a draw, both teams get their respective goals times 0.1 a
        return {row['HomeTeam']: 1 + row['FTHG'] * gw, row['AwayTeam']: 1 + row
    else:
        return {row['HomeTeam']: 0, row['AwayTeam']: 0}

match_points = data.apply(calculate_match_points, axis=1)
points_df = pd.DataFrame(match_points.tolist())
goal_points_summary = points_df.fillna(0).groupby(data['Season']).sum()

goal_points_summary_2021 = goal_points_summary[specified_teams_2021]
goal_points_summary_2122 = goal_points_summary[specified_teams_2122]
goal_points_summary_2223 = goal_points_summary[specified_teams_2223]
```

Autoregression w/ goal weight for 20/21

In [62]:

```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import dataframe_image as dfi

predictions_goal_2021 = pd.DataFrame(columns=['Team', 'Autoregression', 'Predi
row_index = 1

for team_name in goal_points_summary_2021.columns:
    match_points = goal_points_summary_2021[team_name].dropna().reset_index(dr
    match_points = match_points[0:-3]
    match_points.index += 1
    Y = match_points
    if len(Y) > 1:
        model = AutoReg(Y, lags=1)
        model_fitted = model.fit()
        intercept = round(model_fitted.params[0], 2)
        lag_coefficient = round(model_fitted.params[1], 2)
        autoregression_equation = f'Y_t = {intercept} + {lag_coefficient} * Y_
        prediction = round(model_fitted.predict(start=len(Y), end=len(Y), dyna
        predictions_goal_2021.loc[row_index] = [team_name, autoregression_equa
        row_index += 1
    else:
        print(f'Not enough data to predict for {team_name}')

predictions_goal_2021.sort_values(by='Predicted points', ascending=False, inpl
predictions_goal_2021.reset_index(drop=True, inplace=True)
predictions_goal_2021.index += 1
dfi.export(predictions_goal_2021.head(20), 'dataframe_goal_autoreg2021.png')
predictions_goal_2021
```

Out[62]:

	Team	Autoregression	Predicted points
1	Man City	Y_t = 23.38 + 0.77 * Y_t-1	104.85
2	Liverpool	Y_t = 51.87 + 0.43 * Y_t-1	104.13
3	Chelsea	Y_t = 112.23 + -0.17 * Y_t-1	98.32
4	Man United	Y_t = 38.5 + 0.59 * Y_t-1	87.58
5	Arsenal	Y_t = 75.24 + 0.16 * Y_t-1	86.19
6	Tottenham	Y_t = 39.61 + 0.52 * Y_t-1	77.93
7	Leicester	Y_t = 6.58 + 0.81 * Y_t-1	70.41
8	Everton	Y_t = 72.43 + -0.04 * Y_t-1	70.18
9	Southampton	Y_t = 6.75 + 0.81 * Y_t-1	59.06
10	Brighton	Y_t = 3.49 + 0.95 * Y_t-1	51.46
11	West Ham	Y_t = 38.93 + 0.24 * Y_t-1	51.04
12	Newcastle	Y_t = 46.43 + 0.05 * Y_t-1	49.37
13	Wolves	Y_t = 8.51 + 0.56 * Y_t-1	48.78
14	Burnley	Y_t = 10.5 + 0.58 * Y_t-1	47.92
15	Crystal Palace	Y_t = 9.91 + 0.72 * Y_t-1	46.21
16	Aston Villa	Y_t = 8.4 + 0.79 * Y_t-1	44.13
17	West Brom	Y_t = 22.89 + 0.33 * Y_t-1	22.89
18	Fulham	Y_t = 4.04 + 0.8 * Y_t-1	4.04
19	Leeds	Y_t = 0.0 + 0.0 * Y_t-1	0.00
20	Sheffield United	Y_t = 7.32 + -0.16 * Y_t-1	-2.82

Autoregression w/ goal weight for 21/22

In [29]:

```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import dataframe_image as dfi

predictions_goal_2122 = pd.DataFrame(columns=['Team', 'Autoregression', 'Predi
row_index = 1

for team_name in goal_points_summary_2122.columns:
    match_points = goal_points_summary_2122[team_name].dropna().reset_index(dr
    match_points = match_points[0:-2]
    match_points.index += 1
    Y = match_points
    if len(Y) > 1:
        model = AutoReg(Y, lags=1)
        model_fitted = model.fit()
        intercept = round(model_fitted.params[0], 2)
        lag_coefficient = round(model_fitted.params[1], 2)
        autoregression_equation = f'Y_t = {intercept} + {lag_coefficient} * Y_
        prediction = round(model_fitted.predict(start=len(Y), end=len(Y), dynar
        predictions_goal_2122.loc[row_index] = [team_name, autoregression_equa
```

```
        row_index += 1
    else:
        print(f'Not enough data to predict for {team_name}')

predictions_goal_2122.sort_values(by='Predicted points', ascending=False, inpla
predictions_goal_2122.reset_index(drop=True, inplace=True)
predictions_goal_2122.index += 1
#dfi.export(predictions_goal_2122.head(20), 'dataframe_goal_autoreg2223.png')
#predictions_goal_2122
```

Autoregression w/ goal weight for 22/23

In [30]:
```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import dataframe_image as dfi

predictions_goal_2223 = pd.DataFrame(columns=['Team', 'Autoregression', 'Predi
row_index = 1

for team_name in goal_points_summary_2223.columns:
    match_points = goal_points_summary_2223[team_name].dropna().reset_index(dr
    match_points = match_points[0:-1]
    match_points.index += 1
    Y = match_points
    if len(Y) > 1:
        model = AutoReg(Y, lags=1)
        model_fitted = model.fit()
        intercept = round(model_fitted.params[0], 2)
        lag_coefficient = round(model_fitted.params[1], 2)
        autoregression_equation = f'Y_t = {intercept} + {lag_coefficient} * Y_
        prediction = round(model_fitted.predict(start=len(Y), end=len(Y), dynar
        predictions_goal_2223.loc[row_index] = [team_name, autoregression_equa
        row_index += 1
    else:
        print(f'Not enough data to predict for {team_name}')

predictions_goal_2223.sort_values(by='Predicted points', ascending=False, inpla
predictions_goal_2223.reset_index(drop=True, inplace=True)
predictions_goal_2223.index += 1
#dfi.export(predictions_goal_2223.head(20), 'dataframe_goal_autoreg2223.png')
#predictions_goal_2223
```

Distance values for goal weight autoregression model for 20/21, 21/22, and 22/23

In [72]:
```
season1 = seasonal_team_lists.get('S2021', [])
season2 = seasonal_team_lists.get('S2122', [])
season3 = seasonal_team_lists.get('S2223', [])

distance1 = 0
distance2 = 0
distance3 = 0

for team in predicted_season_weight_2021:
    index1 = season1.index(team)
    index2 = predicted_season_weight_2021.index(team)
```

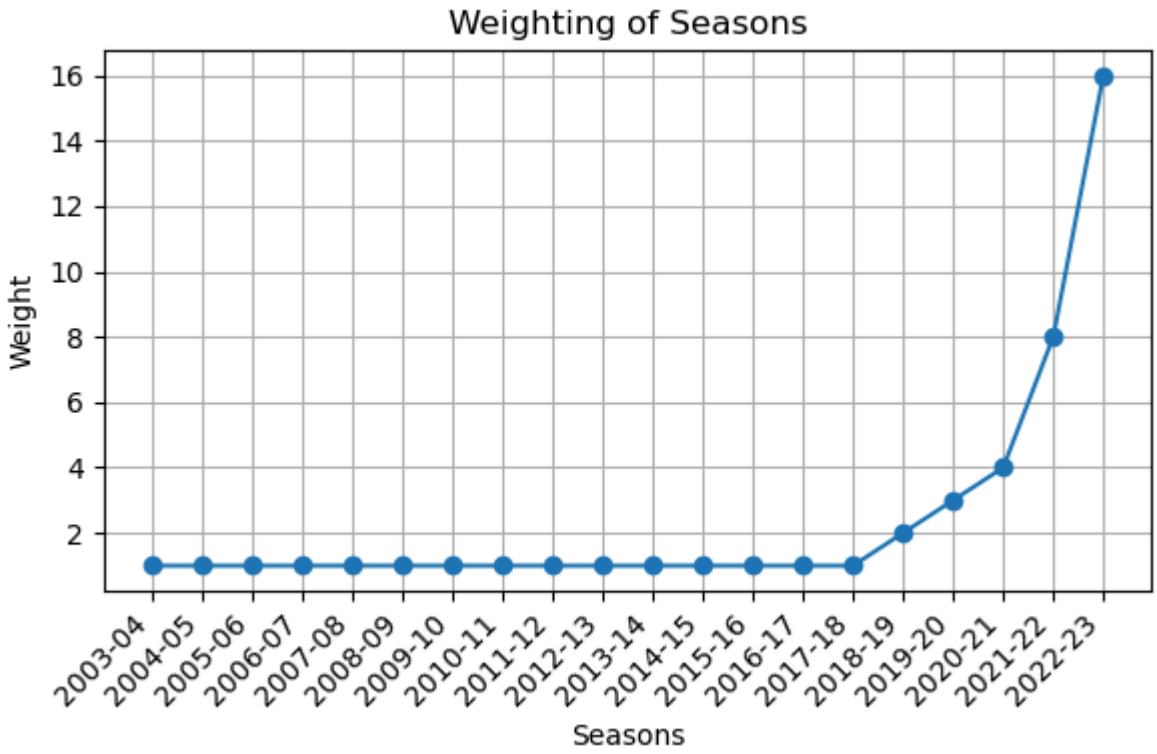
```
    distance1 += abs(index1 - index2)
print('Distance value for predicted 20/21: ', distance1)

for team in predicted_season_weight_2122:
    index1 = season2.index(team)
    index2 = predicted_season_weight_2122.index(team)
    distance2 += abs(index1 - index2)
print('Distance value for predicted 21/22: ', distance2)

for team in predicted_season_weight_2223:
    index1 = season3.index(team)
    index2 = predicted_season_weight_2223.index(team)
    distance3 += abs(index1 - index2)
print('Distance value for predicted 22/23: ', distance3)
```

```
Distance value for predicted 20/21:  48
Distance value for predicted 21/22:  54
Distance value for predicted 22/23:  94
```

Recency weight graph

In [22]:

```
import matplotlib.pyplot as plt

seasons = ["2003-04", "2004-05", "2005-06", "2006-07", "2007-08", "2008-09", "
           "2011-12", "2012-13", "2013-14", "2014-15", "2015-16", "2016-17", "
           "2019-20", "2020-21", "2021-22", "2022-23"]

weights = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 4, 8, 16]

plt.figure(figsize=(6, 4))
plt.plot(seasons, weights, marker='o', linestyle='-')
plt.title('Weighting of Seasons')
plt.xlabel('Seasons')
plt.ylabel('Weight')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.grid(True)
plt.show()
```



Total points per season with goal weight and recency weight 20/21

In [15]:

```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import matplotlib.pyplot as plt

file_path = 'pm_stats.csv'
data = pd.read_csv(file_path)
gw = 0.25
recency_multipliers = {'S0304': 1, 'S0405': 1, 'S0506': 1, 'S0607': 1, 'S0708'
                       'S0809': 1, 'S0910': 1, 'S1011': 1, 'S1112': 1, 'S1213'
                       'S1314': 1, 'S1415': 1, 'S1516': 1, 'S1617': 2, 'S1718'
                       'S1819': 4, 'S1920': 8, 'S2021': 0, 'S2122': 0,'S2223':

def calculate_match_points(row, recency_multiplier):
    home_points = (3 if row['FTR'] == 'H' else 1 if row['FTR'] == 'D' else 0) +
    away_points = (3 if row['FTR'] == 'A' else 1 if row['FTR'] == 'D' else 0) +

    home_points *= recency_multiplier
    away_points *= recency_multiplier

    return {row['HomeTeam']: home_points, row['AwayTeam']: away_points}

match_points = data.apply(lambda row: calculate_match_points(row, recency_mult
points_df = pd.DataFrame(match_points.tolist())
points_summary = points_df.fillna(0).groupby(data['Season']).sum()
points_weight_recency_summary2021 = points_summary[specified_teams_2021]
points_weight_recency_summary2021
```

Out[15]:

Season	Man City	Man United	Liverpool	Chelsea	Leicester	West Ham	Tottenham	Arsenal	Everton	Le
S0304	54.75	91.00	73.75	95.75	45.0	0.00	56.75	108.25	50.25	
S0405	63.75	91.50	71.00	113.00	0.0	0.00	63.75	104.75	72.25	
S0506	53.75	101.00	96.25	109.00	0.0	68.00	78.25	84.00	58.50	
S0607	49.25	109.75	82.25	99.00	0.0	49.75	74.25	83.75	71.00	
S0708	66.25	107.00	92.75	101.25	0.0	59.50	62.50	101.50	78.75	
S0809	64.50	107.00	105.25	100.00	0.0	61.50	62.25	89.00	76.75	
S0910	85.25	106.50	78.25	111.75	0.0	46.75	86.75	95.75	76.00	
S1011	86.00	99.50	72.75	88.25	0.0	43.75	75.75	86.00	66.75	
S1112	112.25	111.25	63.75	80.25	0.0	0.00	85.50	88.50	68.50	
S1213	94.50	110.50	78.75	93.75	0.0	57.25	88.50	91.00	76.75	
S1314	111.50	80.00	109.25	99.75	0.0	50.00	82.75	96.00	87.25	
S1415	99.75	85.50	75.00	105.25	52.5	58.00	78.50	92.75	59.00	
S1516	83.75	78.25	75.75	64.75	98.0	78.25	87.25	87.25	61.75	
S1617	196.00	165.00	191.00	228.50	112.0	113.50	215.00	188.50	153.00	
S1718	379.50	294.00	288.00	256.50	183.0	162.00	286.50	244.50	180.00	
S1819	487.00	329.00	477.00	351.00	259.0	260.00	351.00	353.00	270.00	
S1920	852.00	660.00	962.00	666.00	630.0	410.00	594.00	560.00	480.00	
S2021	0.00	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	
S2122	0.00	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	
S2223	0.00	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.00	

Total points per season with goal weight and recency weight 21/22

In [23]:

```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import matplotlib.pyplot as plt

file_path = 'pm_stats.csv'
data = pd.read_csv(file_path)
gw = 0.25
recency_multipliers = {'S0304': 1, 'S0405': 1, 'S0506': 1, 'S0607': 1, 'S0708'
                       'S0809': 1, 'S0910': 1, 'S1011': 1, 'S1112': 1, 'S1213'
                       'S1314': 1, 'S1415': 1, 'S1516': 1, 'S1617': 1, 'S1718'
                       'S1819': 3, 'S1920': 4, 'S2021': 8, 'S2122': 0,'S2223':

def calculate_match_points(row, recency_multiplier):
    home_points = (3 if row['FTR'] == 'H' else 1 if row['FTR'] == 'D' else 0) *
    away_points = (3 if row['FTR'] == 'A' else 1 if row['FTR'] == 'D' else 0) *
```

```
    home_points *= recency_multiplier
    away_points *= recency_multiplier

    return {row['HomeTeam']: home_points, row['AwayTeam']: away_points}

match_points = data.apply(lambda row: calculate_match_points(row, recency_mult
points_df = pd.DataFrame(match_points.tolist())
points_summary = points_df.fillna(0).groupby(data['Season']).sum()
points_weight_recency_summary2122 = points_summary[specified_teams_2122]
#points_weight_recency_summary2122
```

Total points per season with goal weight and recency weight 22/23

In [24]:
```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import matplotlib.pyplot as plt

file_path = 'pm_stats.csv'
data = pd.read_csv(file_path)
gw = 0.25
recency_multipliers = {'S0304': 1, 'S0405': 1, 'S0506': 1, 'S0607': 1, 'S0708'
                       'S0809': 1, 'S0910': 1, 'S1011': 1, 'S1112': 1, 'S1213'
                       'S1314': 1, 'S1415': 1, 'S1516': 1, 'S1617': 1, 'S1718'
                       'S1819': 2, 'S1920': 3, 'S2021': 4, 'S2122': 8,'S2223':

def calculate_match_points(row, recency_multiplier):
    home_points = (3 if row['FTR'] == 'H' else 1 if row['FTR'] == 'D' else 0) +
    away_points = (3 if row['FTR'] == 'A' else 1 if row['FTR'] == 'D' else 0) +

    home_points *= recency_multiplier
    away_points *= recency_multiplier

    return {row['HomeTeam']: home_points, row['AwayTeam']: away_points}

match_points = data.apply(lambda row: calculate_match_points(row, recency_mult
points_df = pd.DataFrame(match_points.tolist())
points_summary = points_df.fillna(0).groupby(data['Season']).sum()
points_weight_recency_summary2223 = points_summary[specified_teams_2223]
#points_weight_recency_summary2223
```

Autoregression w/ goal and recency weight 20/21

In [16]:
```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import dataframe_image as dfi

predictions_recency_2021 = pd.DataFrame(columns=['Team', 'Autoregression', 'Pr
row_index = 1

for team_name in points_weight_recency_summary2021.columns:
    match_points = points_weight_recency_summary2021[team_name].dropna().reset_
    match_points = match_points[0:-3]
    match_points.index += 1
    Y = match_points
    if len(Y) > 1:
```

```
        model = AutoReg(Y, lags=1)
        model_fitted = model.fit()
        intercept = round(model_fitted.params[0], 2)
        lag_coefficient = round(model_fitted.params[1], 2)
        autoregression_equation = f'Y_t = {intercept} + {lag_coefficient} * Y_
        prediction = round(model_fitted.predict(start=len(Y), end=len(Y), dynar
        predictions_recency_2021.loc[row_index] = [team_name, autoregression_e
        row_index += 1
    else:
        print(f'Not enough data to predict for {team_name}')

predictions_recency_2021.sort_values(by='Predicted points', ascending=False, i
predictions_recency_2021.reset_index(drop=True, inplace=True)
predictions_recency_2021.index += 1
dfi.export(predictions_recency_2021.head(20), 'dataframe_goal_recency_autoreg2(
predictions_recency_2021
```

Out[16]:

	Team	Autoregression	Predicted points
1	Liverpool	Y_t = -84.02 + 2.1 * Y_t-1	1935.60
2	Man City	Y_t = -39.84 + 1.69 * Y_t-1	1397.63
3	Leicester	Y_t = -10.23 + 2.0 * Y_t-1	1249.03
4	Man United	Y_t = -77.31 + 1.87 * Y_t-1	1159.39
5	Chelsea	Y_t = -69.83 + 1.8 * Y_t-1	1131.95
6	Wolves	Y_t = 7.92 + 1.9 * Y_t-1	1098.18
7	Southampton	Y_t = -22.37 + 1.99 * Y_t-1	1009.48
8	Burnley	Y_t = 4.68 + 1.77 * Y_t-1	922.60
9	Tottenham	Y_t = -35.38 + 1.6 * Y_t-1	915.74
10	Arsenal	Y_t = -60.48 + 1.71 * Y_t-1	898.06
11	Everton	Y_t = -50.6 + 1.82 * Y_t-1	824.30
12	Brighton	Y_t = 9.16 + 1.8 * Y_t-1	739.69
13	Newcastle	Y_t = -14.29 + 1.55 * Y_t-1	647.36
14	Crystal Palace	Y_t = 1.16 + 1.53 * Y_t-1	623.33
15	West Ham	Y_t = -9.47 + 1.51 * Y_t-1	608.27
16	Fulham	Y_t = 45.09 + -0.04 * Y_t-1	45.09
17	West Brom	Y_t = 29.03 + 0.32 * Y_t-1	29.03
18	Leeds	Y_t = 0.0 + 0.0 * Y_t-1	0.00
19	Aston Villa	Y_t = 92.04 + -0.58 * Y_t-1	-117.13
20	Sheffield United	Y_t = 37.07 + -0.81 * Y_t-1	-373.89

Autoregression w/ goal and recency weight 21/22

In [31]:

```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import dataframe_image as dfi
```

```
predictions_recency_2122 = pd.DataFrame(columns=['Team', 'Autoregression', 'Pre
row_index = 1

for team_name in points_weight_recency_summary2122.columns:
    match_points = points_weight_recency_summary2122[team_name].dropna().reset_
    match_points = match_points[0:-2]
    match_points.index += 1
    Y = match_points
    if len(Y) > 1:
        model = AutoReg(Y, lags=1)
        model_fitted = model.fit()
        intercept = round(model_fitted.params[0], 2)
        lag_coefficient = round(model_fitted.params[1], 2)
        autoregression_equation = f'Y_t = {intercept} + {lag_coefficient} * Y_
        prediction = round(model_fitted.predict(start=len(Y), end=len(Y), dynar
        predictions_recency_2122.loc[row_index] = [team_name, autoregression_e
        row_index += 1
    else:
        print(f'Not enough data to predict for {team_name}')

predictions_recency_2122.sort_values(by='Predicted points', ascending=False, in
predictions_recency_2122.reset_index(drop=True, inplace=True)
predictions_recency_2122.index += 1
#dfi.export(predictions_recency_2122.head(20), 'dataframe_goal_recency_autoreg2
#predictions_recency_2122
```

Autoregression w/ goal and recency weight 22/23

In [32]:
```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import dataframe_image as dfi

predictions_recency_2223 = pd.DataFrame(columns=['Team', 'Autoregression', 'Pr
row_index = 1

for team_name in points_weight_recency_summary2223.columns:
    match_points = points_weight_recency_summary2223[team_name].dropna().reset_
    match_points = match_points[0:-1]
    match_points.index += 1
    Y = match_points
    if len(Y) > 1:
        model = AutoReg(Y, lags=1)
        model_fitted = model.fit()
        intercept = round(model_fitted.params[0], 2)
        lag_coefficient = round(model_fitted.params[1], 2)
        autoregression_equation = f'Y_t = {intercept} + {lag_coefficient} * Y_
        prediction = round(model_fitted.predict(start=len(Y), end=len(Y), dynar
        predictions_recency_2223.loc[row_index] = [team_name, autoregression_e
        row_index += 1
    else:
        print(f'Not enough data to predict for {team_name}')

predictions_recency_2223.sort_values(by='Predicted points', ascending=False, i
predictions_recency_2223.reset_index(drop=True, inplace=True)
predictions_recency_2223.index += 1
#dfi.export(predictions_recency_2223.head(20), 'dataframe_goal_recency_autoreg
#predictions_recency_2223
```

Distance values for goal and recency weight autoregression for 20/21, 21/22, and 22/23

In [88]:
```
season1 = seasonal_team_lists.get('S2021', [])
season2 = seasonal_team_lists.get('S2122', [])
season3 = seasonal_team_lists.get('S2223', [])

distance1 = 0
distance2 = 0
distance3 = 0

for team in predicted_season_weight_recency_2021:
    index1 = season1.index(team)
    index2 = predicted_season_weight_recency_2021.index(team)
    distance1 += abs(index1 - index2)
print('Distance value for predicted 20/21: ', distance1)

for team in predicted_season_weight_recency_2122:
    index1 = season2.index(team)
    index2 = predicted_season_weight_recency_2122.index(team)
    distance2 += abs(index1 - index2)
print('Distance value for predicted 21/22: ', distance2)

for team in predicted_season_weight_recency_2223:
    index1 = season3.index(team)
    index2 = predicted_season_weight_recency_2223.index(team)
```

```
    distance3 += abs(index1 - index2)
print('Distance value for predicted 22/23: ', distance3)
```

```
Distance value for predicted 20/21:  72
Distance value for predicted 21/22:  78
Distance value for predicted 22/23:  84
```

Total points per season with goal weight and recency weight 23/24

In [2]:

```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import matplotlib.pyplot as plt

file_path = 'pm_stats.csv'
data = pd.read_csv(file_path)

specified_teams_2324 = ['Arsenal', 'Liverpool', 'Man City', 'Aston Villa', 'To
                        'Man United', 'West Ham', 'Newcastle', 'Brighton', 'Wo
                        'Bournemouth', 'Chelsea', 'Fulham', 'Crystal Palace',
                        'Everton', 'Nott\'m Forest', 'Burnley', 'Sheffield Uni

gw = 0.25
recency_multipliers = {'S0304': 1, 'S0405': 1, 'S0506': 1, 'S0607': 1, 'S0708'
                       'S0809': 1, 'S0910': 1, 'S1011': 1, 'S1112': 1, 'S1213'
                       'S1314': 1, 'S1415': 1, 'S1516': 1, 'S1617': 1, 'S1718'
                       'S1819': 1, 'S1920': 2, 'S2021': 3, 'S2122': 4,'S2223':

def calculate_match_points(row, recency_multiplier):
    home_points = (3 if row['FTR'] == 'H' else 1 if row['FTR'] == 'D' else 0) *
    away_points = (3 if row['FTR'] == 'A' else 1 if row['FTR'] == 'D' else 0) *

    home_points *= recency_multiplier
    away_points *= recency_multiplier

    return {row['HomeTeam']: home_points, row['AwayTeam']: away_points}

match_points = data.apply(lambda row: calculate_match_points(row, recency_mult
points_df = pd.DataFrame(match_points.tolist())
points_summary = points_df.fillna(0).groupby(data['Season']).sum()
points_weight_recency_summary2324 = points_summary[specified_teams_2324]
points_weight_recency_summary2324
```

Out[2]:

Season	Arsenal	Liverpool	Man City	Aston Villa	Tottenham	Man United	West Ham	Newcastle	Brighton	W
S0304	108.25	73.75	54.75	68.00	56.75	91.00	0.00	69.00	0.00	
S0405	104.75	71.00	63.75	58.25	63.75	91.50	0.00	55.75	0.00	
S0506	84.00	96.25	53.75	52.50	78.25	101.00	68.00	69.75	0.00	
S0607	83.75	82.25	49.25	60.75	74.25	109.75	49.75	52.50	0.00	
S0708	101.50	92.75	66.25	77.75	62.50	107.00	59.50	54.25	0.00	
S0809	89.00	105.25	64.50	75.50	62.25	107.00	61.50	44.00	0.00	
S0910	95.75	78.25	85.25	77.00	86.75	106.50	46.75	0.00	0.00	
S1011	86.00	72.75	86.00	60.00	75.75	99.50	43.75	60.00	0.00	
S1112	88.50	63.75	112.25	47.25	85.50	111.25	0.00	79.00	0.00	
S1213	91.00	78.75	94.50	52.75	88.50	110.50	57.25	52.25	0.00	
S1314	96.00	109.25	111.50	47.75	82.75	80.00	50.00	59.75	0.00	
S1415	92.75	75.00	99.75	45.75	78.50	85.50	58.00	49.00	0.00	
S1516	87.25	75.75	83.75	23.75	87.25	78.25	78.25	48.00	0.00	
S1617	94.25	95.50	98.00	0.00	107.50	82.50	56.75	0.00	0.00	
S1718	81.50	96.00	126.50	0.00	95.50	98.00	54.00	53.75	48.50	
S1819	88.25	119.25	121.75	0.00	87.75	82.25	65.00	55.50	44.75	
S1920	140.00	240.50	213.00	90.50	148.50	165.00	102.50	107.00	101.50	1
S2021	224.25	258.00	320.25	206.25	237.00	276.75	241.50	169.50	153.00	1
S2122	337.00	462.00	471.00	232.00	353.00	289.00	284.00	240.00	246.00	2
S2223	848.00	686.00	900.00	590.00	620.00	716.00	404.00	704.00	640.00	3

Autoregression w/ goal and recency weight 23/24

In [94]:

```
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
import dataframe_image as dfi

predictions_recency_2324 = pd.DataFrame(columns=['Team', 'Autoregression', 'Pre
row_index = 1

for team_name in points_weight_recency_summary2324.columns:
    match_points = points_weight_recency_summary2324[team_name].dropna().reset_
    match_points.index += 1
    Y = match_points
    if len(Y) > 1:
        model = AutoReg(Y, lags=1)
        model_fitted = model.fit()
        intercept = round(model_fitted.params[0], 2)
        lag_coefficient = round(model_fitted.params[1], 2)
        autoregression_equation = f'Y_t = {intercept} + {lag_coefficient} * Y_
```

```
        prediction = round(model_fitted.predict(start=len(Y), end=len(Y), dynar
        predictions_recency_2324.loc[row_index] = [team_name, autoregression_e
        row_index += 1
    else:
        print(f'Not enough data to predict for {team_name}')

predictions_recency_2324.sort_values(by='Predicted points', ascending=False, i
predictions_recency_2324.reset_index(drop=True, inplace=True)
predictions_recency_2324.index += 1
#dfi.export(predictions_recency_2324.head(20), 'dataframe_goal_autoreg2223.png
predictions_recency_2324
```

Out[94]:

	Team	Autoregression	Predicted points
1	Arsenal	Y_t = -164.52 + 2.78 * Y_t-1	2191.49
2	Newcastle	Y_t = -72.6 + 2.53 * Y_t-1	1706.60
3	Man City	Y_t = -69.93 + 1.92 * Y_t-1	1653.60
4	Brentford	Y_t = 12.89 + 2.48 * Y_t-1	1470.50
5	Man United	Y_t = -103.61 + 2.14 * Y_t-1	1429.64
6	Brighton	Y_t = -4.87 + 2.23 * Y_t-1	1424.76
7	Aston Villa	Y_t = -35.38 + 1.94 * Y_t-1	1106.90
8	Tottenham	Y_t = -62.65 + 1.87 * Y_t-1	1097.72
9	Liverpool	Y_t = -40.31 + 1.59 * Y_t-1	1048.65
10	Crystal Palace	Y_t = -6.42 + 1.64 * Y_t-1	713.07
11	Wolves	Y_t = -0.56 + 1.45 * Y_t-1	566.00
12	Chelsea	Y_t = -18.72 + 1.29 * Y_t-1	535.52
13	West Ham	Y_t = -0.4 + 1.3 * Y_t-1	524.39
14	Everton	Y_t = -23.57 + 1.46 * Y_t-1	496.12
15	Bournemouth	Y_t = 33.37 + 0.21 * Y_t-1	115.78
16	Nott'm Forest	Y_t = 20.0 + 0.0 * Y_t-1	20.00
17	Burnley	Y_t = 15.63 + 0.57 * Y_t-1	15.63
18	Sheffield United	Y_t = 9.09 + 0.33 * Y_t-1	9.09
19	Fulham	Y_t = 102.45 + -0.97 * Y_t-1	-405.19

Distance Values for The Guardian prediction for 20/21, 21/22, and 22/23

In [91]:

```
season1 = seasonal_team_lists.get('S2021', [])
season2 = seasonal_team_lists.get('S2122', [])
season3 = seasonal_team_lists.get('S2223', [])

guardian_prediction_2021 = ["Liverpool", 'Man City', 'Chelsea', 'Man United',
                            'Tottenham', 'Leicester', 'Wolves', 'Everton', 'Le
                            'Sheffield United', 'Southampton', 'Burnley', 'Wes
                            'Brighton', 'Newcastle', 'Crystal Palace', 'Fulham
```

```
guardian_prediction_2122 = ['Man City', 'Man United', 'Chelsea', 'Liverpool',
                            'Arsenal', 'Tottenham', 'Everton', 'Leeds', 'West H
                            'Aston Villa', 'Brighton', 'Wolves', 'Southampton',
                            'Newcastle', 'Burnley', 'Crystal Palace', 'Brentfo
                            'Norwich', 'Watford']
guardian_prediction_2223 = ['Man City', 'Liverpool', 'Tottenham', 'Chelsea',
                            'Arsenal', 'Man United', 'Newcastle', 'West Ham',
                            'Leicester', 'Aston Villa', 'Crystal Palace', 'Brig
                            'Wolves', 'Everton', 'Nott\'m Forest', 'Brentford',
                            'Leeds', 'Fulham', 'Bournemouth']
distance1 = 0
distance2 = 0
distance3 = 0

for team in guardian_prediction_2021:
    index1 = season1.index(team)
    index2 = guardian_prediction_2021.index(team)
    distance1 += abs(index1 - index2)

for team in guardian_prediction_2122:
    index1 = season2.index(team)
    index2 = guardian_prediction_2122.index(team)
    distance2 += abs(index1 - index2)

for team in guardian_prediction_2223:
    index1 = season3.index(team)
    index2 = guardian_prediction_2223.index(team)
    distance3 += abs(index1 - index2)

print('Distance Value for The Guardian Prediction 20/21: ', distance1)
print('Distance Value for The Guardian Prediction 21/22: ', distance2)
print('Distance Value for The Guardian Prediction 22/23: ', distance3)
```

```
Distance Value for The Guardian Prediction 20/21:  56
Distance Value for The Guardian Prediction 21/22:  60
Distance Value for The Guardian Prediction 22/23:  78
```