Zul-Kifl Alidou, Bryan Nunez, Juan Uscanga, Sedachael Zeleke
Dr. Mingon Kang
CS 489 - Introduction to Machine Learning
08 May 2020

EPL Predictor Final Report

Introduction

For our machine learning project, we wanted to make something that was flexible in its usability but that was also interesting to all of us so that we could enjoy it. So, we decided to try to accurately predict the positions of the English Premier League football teams at the end of a season (2018-2019 season for this model). In order to accomplish this, we first had to figure out how to determine the outcome of a match between two teams. This problem is a multiclass classification problem, as the test data (being the two teams playing each other) will be classified into one of three categories: *H* (if the home team is predicted to win), *A* **(**if the away team is predicted to win), or *D* (if the match is predicted to end in a draw). After predicting the outcomes of all the matches in an entire season, we will rank each team based on our predictions, and compare that to the actual end of season standings for that season.

Brief Description of the Data

The samples in our training data set consist of matches that were played in previous seasons in the English Premier League, starting from the 2011-2012 season and ending in the 2017-2018 season. These samples are made up of statistics from the matches and information about the teams playing each other. The features of these samples include (but are not limited to) full time home team goals, full time away team goals, full time result, half time home team goals, half time away team goals, half time result, home team shots, away team shots, home team shots on target, and away team shots on target. Using these features, we will try to predict the label (or match outcome) for our testing data. In our training data set, the column titled “full time result” will represent the label for each sample and will be taking a value of *H* (to represent a victory for the home team), *A* (to represent a victory for the away team), or *D* (to represent a draw). We have 2,660 of these data samples at our disposal, which can be found at *http://football-data.co.uk/englandm.php*.

Data Preprocessing

To proceed any further, we first needed to properly set up our data. Since we used several seasons worth of data, we started by reading in each csv file into its own variable, and once they were all read in, we combined them into one. We then created two new variables, to be used as

our input (X) and output (y) variables. We proceeded by removing the unnecessary columns from our input data set (i.e., home/away team names, match duration, time/date, and any other non-numerical column) as well as any predictive statistics that would give away the outcome of the matches (i.e., goals scored, match outcome, etc.). The way we did this was by copying predictive and non-predictive features into separate variables in case they would be required later, as opposed to deleting the predictive feature columns from our data set. Lastly, we had to convert our outcome data set into numerical values (i.e., Home Team to '1', Away Team to '0', and Draw to '-1').

Generating Statistics

The first thing we did after polishing our data set was to try predicting each team's statistics for every match in the season. This meant generating for both teams: the number of passes made, the number of fouls committed, the number of shots taken, etc. The idea was that by generating (or predicting) these statistics, we would in turn use them to help us move to the next stage, which is to predict the number of goals scored by both teams in a given match.

To generate the statistics of a team within a match, we looked at all the other features of that same team within that team's last ten matches. For example, if we wanted to predict the number of shots Arsenal will take in a match, we look at all of Arsenal's statistics (except for the number of shots it took) in its last ten matches. If we wanted to predict the number of fouls that Arsenal will commit in a match, we look at all of Arsenal's statistics (except for the number of fouls it committed) in its last ten matches. By following this approach, we generated the values for the statistics of each team in a match.

To predict each of the statistics of both teams in a match, we used a linear regression classifier. During this step, we did try other classifiers such as Logistic Regression and ridge regression, but we got the best (and most realistic) statistics using linear regression. The statistics from a team's ten previous matches were used to train the classifier and generate the value of a specific statistic.

We decided to only look at the last ten matches of a team because we felt like the ten previous matches were a good indication of how well a team would perform in a match. We did not look at more than ten matches because too many matches are not reliable and might be a worse indication of how a team is going to perform currently. We also did not consider looking at fewer than ten matches because it would not provide us with enough data, so ten matches seemed like the perfect number.

Generating Goals

After generating the statistics that we predicted to occur in a match for a team, we used those values to predict the number of goals scored by each team in a match. We do this because by knowing the number of goals each team scores, we can determine the outcome of the match, and by knowing the outcome of the match, we can give the respective teams the correct number of points. For example, in a given match if the home team wins, they get 3 points and the away team gets 0 points, and so on.

To predict the number of goals that a team may score, we use a gradient boost regressor to help us determine this value. We did consider a few other models to help us generate this value (including but not limited to ridge regressors and linear regressors), but we seemed to always get the best (and most realistic) goals out of the gradient boost regressor:

We started by training our goal model on the matches that occurred from the 2011-2012 season all the way to the 2017-2018 seasons, along with the goals that occurred in each of those matches. And whenever we wanted to predict the number of goals that a certain team scored in a match, we would feed the goal model with the generated stats for that same team (from the previous section), and whatever output we got would represent the predicted number of goals for that team in a match. We essentially did this operation for both teams in every match that occurred in the 2018-2019 season and assigned a certain number of points to each team based on the number of goals they and their opponents scored.

After doing this for every match, we created a table that ranked the teams, based on the number of points we predicted them to earn. The table below represents how we predicted the season to end:

Pos	Team	Points	Pos	Team	Points	Difference
1	Manchester City	98	1	Liverpool	67	-1
2	Liverpool	97	2	Manchester United	63	-4
3	Chelsea	72	3	Bournemouth	62	-11
4	Tottenham Hotspur	71	4	Huddersfield	61	-16
5	Arsenal	70	5	Leicester	59	-4
6	Manchester United	66	6	Manchester City	59	+5
7	Wolverhampton Wanderers	57	7	Arsenal	57	+2
8	Everton	54	8	Fulham	57	-11
9	Leicester City	52	9	Everton	54	+1
10	West Ham United	52	10	Chelsea	53	+7
11	Watford	50	11	Crystal Palace	50	-1
12	Crystal Palace	49	12	Tottenham Hotspur	48	+8
13	Newcastle United	45	13	Watford	47	+2
14	Bournemouth	45	14	Burnley	46	-1
15	Burnley	40	5	Wolves	46	+8
16	Southampton	39	16	Southampton	45	0
17	Brighton & Hove Albion	36	17	West Ham United	44	+7
18	Cardiff City	34	18	Newcastle United	41	+5
19	Fulham	26	19	Cardiff	36	-1
20	Huddersfield Town	16	20	Brighton & Hove Albion	33	+3

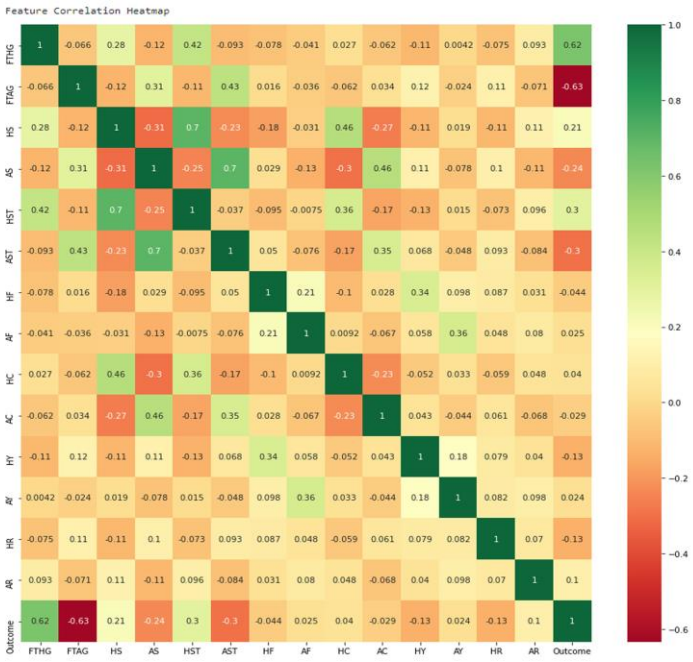
As it can be seen above, we did not perform too well on this particular model. The right-most column (titled “Difference) tells how far our prediction was from the actual ranking of that team.

For example, we predicted Manchester United to end up in second position, but they ended up four places below (in sixth place). That's what the "-4" in the right most column on the second chart means. The only team we correctly placed was Southampton, which we predicted for them to end up in sixteenth place (and they did).

After comparing our simulated table to the actual one, we concluded that our predictions were incredibly far off, and started brainstorming some ideas on how to improve it.

Feature Selection

We had already tried to generate our statistics using all the available features at this point, but the final result was far off from the actual 2018-2019 standings table. Thus, we decided to make some extra implementations in order to increase our model's accuracy. The easiest and most obvious way to do so was to use feature selection and feature extraction. Before getting started with these implementations, we decided to create a correlation matrix that would allow us to see the relationship each feature had with one another as well as with the target variable (match outcome). We color coded this red, yellow, green to represent a heatmap of sorts and make it that much easier for us to visualize the data:



From this heatmap, we were able to see that the seemingly related features were in fact highly correlated (for example, Home Team Shots and Home Team Shots on Target with a 0.7 correlation coefficient). Thus, we decided to get rid of one feature from each pair of highly correlated features, effectively performing a simple feature extraction technique. By modifying one of these pairs at a time, we ran the newly modified list of features into the model to

determine if it would give us a better accuracy, and if it did, we would remove another feature from a different highly correlated pair.

The second implementation approach we took was to use several feature selection functions in order to extract unnecessary features instead of manually testing all the possible subsets. We implemented three different functions: Sklearn's built in feature selection function, Chi-Squared, and Random Forest. After running our original list of non-predictive features into each, we noticed that the rankings for each of these methods were similar, usually having the same top five or so features, just in different order. Since we just needed to test with K number of features, the ranking of the features did not matter, just what those top K features were. However, the rankings/scores did provide us with new information as to how correlated to the output the features were. Again, we tested our accuracies using the top 5, 6, 7, etc. features for each of the three methods and concluded that Chi-Squared gave us the best accuracy with K=6 features. Below are the results of the three methods:

Sklearn	Random Forest	Chi-Squared
1. **HST** - 0.1190	1. **HST** - 0.1155	1. **HST** - 1038.358
2. **AST** - 0.1064	2. **AF** -- 0.1092	2. **AST** - 920.324
3. **HS** -- 0.1035	3. **HS** -- 0.1079	3. **AS** -- 668.393
4. **AS** -- 0.1021	4. **AST** - 0.1064	4. **HS** -- 561.865
5. **AF** -- 0.1007	5. **HF** -- 0.1056	5. **HY** -- 108.262
6. HF -- 0.0999	6. AS -- 0.1037	6. HR -- 105.968
7. HC -- 0.0992	7. HC -- 0.1020	7. AR -- 61.587
8. AC -- 0.0970	8. AC -- 0.0955	8. HC -- 26.992
9. AY -- 0.0765	9. AY -- 0.0682	9. AY -- 22.521
10. HY -- 0.0661	10. HY -- 0.0596	10. AF -- 22.036

From these two different approaches, we got similar final predictions. We decided to try a third approach, which was to combine the two previous by taking the top K features as listed in each of the feature selection functions and then removing one feature from each highly correlated pair - effectively allowing a new feature from lower on a list to be used. We believed this would have been a better way to calculate the predictions, but that was not the case.

Ultimately, the second approach which was to just take the top K features given by Chi-Squared resulted in the highest accuracy during these tests, so that is what we used from this point on. We believe that one of the main reasons why the hypothetical approach for feature selection and extraction was not the best is because of the data sets we chose. In essence, we did not have a lot of features available to us, and the ones that we did have were highly correlated pairs (essentially repeats of one another), meaning they did not offer much new information. It would have been

optimal to have some other more independent statistics such as total ball possession time (or percent), as well as number of passes and successful passes in our data sets.

Support Vector Machine (SVM)

Another approach we took in our experiment to improve our predicted table standings was to experiment with SVM. SVM is a supervised binary linear classification algorithm. SVM works by plotting points on a graph and picks the best line that would divide the points and classify new points depending on which side of the line it is at. But in our case, since we have three outcomes, a binary classification would not necessarily work. Therefore, a multiclass SVM must be used. Luckily, Sklearn has a OneVsRestClassifier that handles multiclass SVM.

Generally, the ideal case for SVM would be to predict the outcome of matches and tally the scores of each team depending on if it was a win, loss, or draw. In our case, we used the 2012-2018 actual season statistics to train the model. There are two ways of predicting the 2019 season table, one way is using the generated statistics for 2019 and the second way is to use the actual statistics for 2019. We noticed that the actual 2019 statistics gave us a better accuracy than the generated statistics. Here are the results for the actual 2019 team statistics: (left to right: Actual 2019 standings vs. Predicted 2019 standings):

Pos	Team	Points
1	Manchester City	98
2	Liverpool	97
3	Chelsea	72
4	Tottenham Hotspur	71
5	Arsenal	70
6	Manchester United	66
7	Wolverhampton Wanderers	57
8	Everton	54
9	Leicester City	52
10	West Ham United	52
11	Watford	50
12	Crystal Palace	49
13	Newcastle United	45
14	Bournemouth	45
15	Burnley	40
16	Southampton	39
17	Brighton & Hove Albion	36
18	Cardiff City	34
19	Fulham	26
20	Huddersfield Town	16

Pos	Team	Points	Difference
1	Manchester City	83	0
2	Liverpool	83	0
3	Leicester	75	+6
4	Chelsea	74	-1
5	Man United	73	+1
6	Tottenham	63	+2
7	Wolves	62	0
8	Bournemouth	58	+6
9	Arsenal	58	+4
10	Southampton	58	+6
11	Everton	56	-3
12	Newcastle	51	-1
13	West Ham	49	-3
14	Crystal Palace	42	-2
15	Watford	40	-4
16	Brighton	39	+1
17	Burnley	38	-2
18	Huddersfield	37	+2
19	Cardiff City	32	-1
20	Fulham	23	+1

The table that was generated on the right was with a 58% accuracy of how many matches it correctly predicted. As we can see from comparing the tables, SVM did a great job at predicting the table. It did not get a lot of the correct positions, but it did get the top teams correct and bottoms teams as well since the positions were not too far off. We have 3 correct team positions and 6 teams with a +1 and -1 difference. This means that teams were off by one position.

Logistic Regression (LR)

After experimenting with SVM to predict the end of season standings, we decided to also try Logistic Regression. The reason we chose Logistic Regression is because along with SVM, it is a fundamental machine learning algorithm that is used for binary and multiclassification problems. Since the outcome of a match has three possibilities: home team win, away team win, or draw, Logistic Regression is a great choice for this. In terms of architecture, we imported scikit-learn and used their Logistic Regression model. Additionally, we used One vs All for the classifier since our problem is a multiclassification problem. For training the model, the input is the 2012-2018 actual season statistics. For testing the model, we performed two experiments with separate input values. The input for the first experiment consisted of using the actual statistics for the 2019 season and the input for the second experiment consisted of the generated 2019 statistics. In both experiments, I measured accuracy by recording the number of correctly predicted match outcomes and dividing by the total number of match outcomes. The accuracy for experiment one was 62% and the accuracy for the experiment two was 40%. Here are the results using the actual 2019 team statistics: (left to right: Actual 2019 standings vs. Predicted 2019 standings):

Pos	Team	Points
1	Manchester City	98
2	Liverpool	97
3	Chelsea	72
4	Tottenham Hotspur	71
5	Arsenal	70
6	Manchester United	66
7	Wolverhampton Wanderers	57
8	Everton	54
9	Leicester City	52
10	West Ham United	52
11	Watford	50
12	Crystal Palace	49
13	Newcastle United	45
14	Bournemouth	45
15	Burnley	40
16	Southampton	39
17	Brighton & Hove Albion	36
18	Cardiff City	34
19	Fulham	26
20	Huddersfield Town	16

Pos	Team	Points	Difference
1	Manchester City	96	0
2	Liverpool	90	0
3	Manchester United	75	-3
4	Chelsea	75	+1
5	Leicester City	75	-4
6	Wolves	69	-1
7	Everton	66	-1
8	Tottenham	63	+4
9	Crystal Palace	54	-3
10	Arsenal	52	+5
11	Bournemouth	51	-3
12	Newcastle	51	-1
13	Watford	51	+2
14	Southampton	51	-2
15	West Ham	51	+5
16	Cardiff	42	-2
17	Huddersfield	37	-3
18	Brighton	33	+1
19	Burnley	30	+4
20	Fulham	27	+1

Overall, Logistic Regression also produced a well representation of the table with a 62% accuracy of correctly predicted match outcomes for experiment 1. We have 2 correctly predicted teams at the top and a total of 6 teams with a +1 and -1 difference; meaning that those teams were only one position off from their actual position. The worst difference we have is a +5 with teams Arsenal and West Ham.

Challenges

One of the greatest challenges that we faced during our endeavor had to do with our data. From initially inspecting it, we believed that what had been offered to us would be the most important factors we could have at our disposal to predict a future season. However, that was not the case

at all. Some of the statistics or features that we would have loved to work with are ball possession time or percent (what percent of the match time did a team hold the ball in their possession), corner kicks, free kicks, and penalty kicks to name a few. Furthermore, some other statistics we could have used are things such as team net worth and budget (can serve as a different measure of how 'good' players might be), as well as roster changes or some percent of how many players have been changed since the previous season. To further improve our accuracy, we also thought about adding individual player statistics to help predict the outcome of a match. For example, depending on the different set of players that are on the field for a given match, there may or may not be more synergy as a whole unit which can also influence the outcome of the match. Or there may be certain individuals that can have great influences on the match outcome. As a team, we spent a great deal of time talking about other 'outside' factors that might influence the outcome of a football season. One of those things that came up was betting - throughout history, the outcome of a sports match can be swayed for the right price. Some other influencers could also be politics, a country's economy, laws, etc. There are just so many things we thought about that might have great impacts on a match and season end results. There is also the other side to this being that all these players are professionals, and with that comes a certain expectation that must be met. Then again, these players are all human, and with it comes a certain level of unpredictability.

Another challenge that we ran into resulted from being unsure on how to correctly evaluate our model. This was a challenge because our final output consisted of a table representing team standings at the end of the season. Therefore, we needed to figure out a scientific way to measure how accurate our table was when compared to the season's actual table. In addition to just comparing each team's standing in the predicted table standings with the actual table standings, we also decided to try recording our model accuracy. In this case, the predicted 'y' value is the outcome of a predicted match and the ground truth 'y' value is the actual outcome of a match. By using this method of evaluation, we are able to look at both the accuracy of the model along with the table standings it produced. Even though our main concern is the result of our final table standings, the accuracy of the match outcomes is also something we took into consideration when optimizing the model.

Conclusion and Results

Set on predicting the English Premier League football team standings at the end of the 2019 season, we worked diligently as a team, discussing and implementing different ideas and solutions to help overcome the many challenges we faced on our journey. We started by first predicting the statistics for both teams in all matches and using those stats to predict the number of goals scored by each team. This initial implementation was more of a brute force approach (given the fact that our predictions were incredibly far off from the actual rankings), so we decided to take it back to the drawing board to try and improve our method. We took some of

what we had previously learned in class to try and improve the results. The first improvement we thought to add was feature selection/extraction. It was during this time that we came to realize that our data sets were incomplete and left much to be desired in terms of feature variety. However, we decided that working with a smaller set of features would help us be able to understand everything better rather than adding more features and complicating things for ourselves. Moving on, we decided to do a little bit of cleanup with our data set and remove unnecessary and redundant data (HST vs. HS). Again, removing some of the highly correlated features in an already limited data set showed a decrease in accuracy, so we kept all of these features. Next, we decided to implement feature selection methods (i.e., Chi-Squared) and use only K amount of features to predict match outcomes. From our results, this seemed to be the optimal choice as our predictions started to look more like the real deal. Once we had improved the accuracy of our model, we moved on to implementing support vector machines and Logistic Regression. Compared to our initial approach, we saw just how inaccurate at predicting team positions it was compared to SVM and Logistic Regression. From our extensive testing, Logistic Regression (compared to SVM) resulted in a higher "accuracy". However, our SVM model's predicted standings table was closer to the actual 2018-2019 standings table. Thus, since our goal was about producing an accurate standings table (as opposed to optimizing "accuracy"), we concluded that SVM was the better choice to work off of if we were to continue improving our model. For future reference, we now know to use better data sets with a greater variety of features as well as implementing additional statistics that are not considered totally relevant.