COSE362: Term Project Report
“Predicting Soccer Match Result in the Premier League”

Team 16 2018320178 Youngchan Kim 2018320229 Camden Scott 2018320239 Hojun Ryu

1. Introduction

Soccer is one the most popular sports in the world and with the 2022 FIFA World Cup being held in Qatar™, which recently finished, we thought it would be interesting to design a model so that we could predict the winning team of any soccer match. Since soccer matches consist of many interesting features and records dating back many years are available online, we thought it would be a good choice for analyzing using machine learning methods.

2. Problem Definition

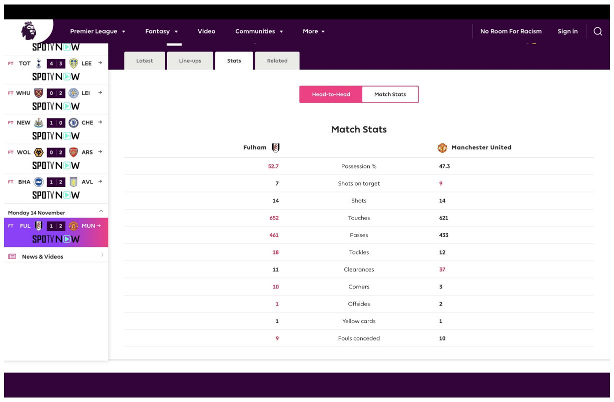
Our goal is to design a machine learning model to predict the outcome of a soccer match. The dataset used was from *premierleague.com*, the official site of the Premier League. One of the expected challenges when designing this model was in extracting meaningful data from the match history and player statistics, which contributes to the overall outcome of each individual match. We attempted two different types of approach: one based on match attributes and one based on individual player attributes. Given features of a match or players, we attempted to find the probability of each individual teams’ chances to win their match.

3. Methodology 1: Match Attribute Approach

3.1. Introduction

There are many factors, attributes, and statistics in any one single soccer match. For example, attributes such as number of passes, number of shots, and ball possession may play an important role in determining victory or defeat in a match. We first focused on the overall match attributes and tried to build a regression model.

3.2. Data Collection



Match Stats

Fulham		Manchester United
52.7	Possession %	47.3
7	Shots on target	9
14	Shots	14
652	Touches	621
461	Passes	433
18	Tackles	12
11	Clearances	37
10	Corners	3
1	Offsides	2
1	Yellow cards	1
9	Fouls conceded	10

Figure 1. Official Premier League Site(match stats)

We collected match statistics from the Official Premier League site using Selenium and saved them in a .csv file. Collected statistics include: Scores, Possession, Shots on target, Shots, Touches, Passes, Tackles, Clearances, Corners, Offsides, Yellow cards, Fouls conceded, Red cards.
Twenty teams compete in the league and each team plays every other team twice during a season. Therefore, there are 380 matches per season and we collected data from 16 seasons(2007-2022).

3.3. Analyzing Features

We collected 12 features for every match, so we attempted to reduce them using the PCA, SVD, and LDA methods. First, we made a correlation map, as shown below. This helped us determine which features are important and how much they affected the final match results. We could check how the possessions, shots on target, shots, touches, and passes are related and attempt to find any correlations between them.

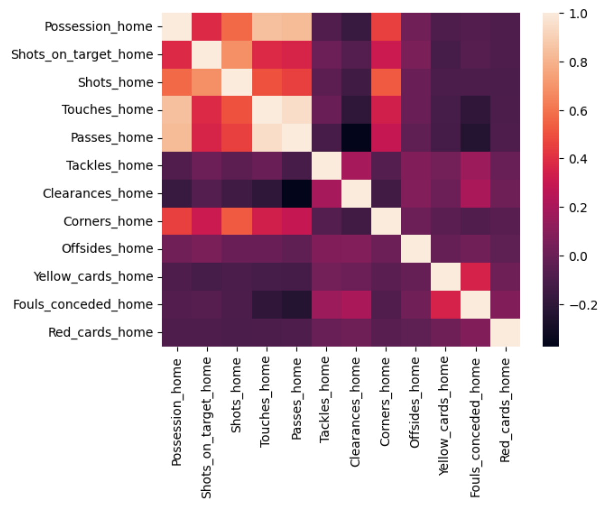


Figure 2. Correlation between features

Next, we tried PCA to calculate which features explained the variance in the dataset.
PCA result:
We concluded that just 6 features could explain over 95% of the variance in the dataset and applied SVD and LDA with 6 features.

3.4. Regression

3.4.1. Linear Regression

We used a linear model for regression with the training set using matches from 2007 to 2020 and a test set with matches from 2021-2022. The size of the train/test split is 0.14.

3.4.2. Linear Regression with Standardization

We standardized the whole dataset to a Gaussian of (0,1). After pre-processing, we split the dataset into the training set and test set with ratios of 0.2, 0.8.

3.4.3. Linear Regression with Standardization & SVD

We applied SVD to the standardized dataset and applied regression to the linear model. The dataset used is the same as before.
The SVD function(TruncatedSVD) is from the scikit-learn decomposition library.

3.4.4. Linear Regression with Standardization & LDA

We applied LDA to the standardized dataset and applied regression to the linear model. The dataset used is the same as before.
The LDA function(LinearDiscriminantAnalysis) is from the scikit-learn discriminant_analysis library.

3.5. Results

3.5.1. RMSE & R2 Score

We calculated the error using RMSE(Root-Mean-Square Error) and a scored model fitted with an R2 Score. The results are as follows in Tables 1 - 4:

	Train	Test
RMSE	1.01	1.00
Score	0.50	0.43

Table 1. RMSE and R2 Score on 3.1.4

	Train	Test
RMSE	1.01	1.00
Score	0.41	0.39

Table 2. RMSE and R2 Score on 3.1.5

	Train	Test
RMSE	1.20	1.18
Score	0.17	0.17

Table 3. RMSE and R2 Score on 3.1.6

	Train	Test
RMSE	1.01	1.01
Score	0.41	0.40

Table 4. RMSE and R2 Score on 3.1.7

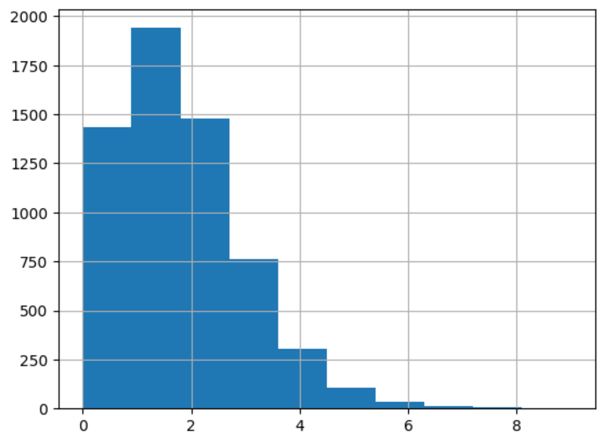


Figure 3. Histogram of Home Team Score

Since the home team's score in most of the matches lies between 0 and 2, an RMSE of about 1 is too big to have confidence in the prediction. Also, it seems that the feature reduction methods were not as effective as expected and the gap between the training error and test error is too small. We concluded that the bias is too high and our model is underfitting the data. It seems like our model is too simple to learn about the structures of a complex soccer game, so the predictions are not as accurate in training sets.

4. Methodology 2: Player Attribute Approach

4.1. Introduction

We tried analyzing with overall match attributes, but it suffered from huge RMSE and there were limitations in regards to real-time prediction. Therefore, we turned our attention to individual player attributes, which are also important and are nearly deterministic before every match. In addition, rather than building a regression model, we just attempted classification of the match output. Classification models we used are Support Vector Machine, Stochastic Gradient Descent, Decision Tree, and Random Forest.

4.2. Dataset collection

We deployed Selenium once again to collect individual player attributes from 2013 to 2022 on the Official Premier League site and split the dataset into a training set(0.8), a test set(0.1), and a validation set(0.1).

The Collection routine is as follows:

1. Scrap all match results.
2. Scrap all player stats with player ID.
3. Standardize each player's feature by appearances (# matches player played in)
4. Scrap all playerIDs for each match.
5. Merge all features and mean features per home and away team with player ID.

The dataset can be found in this link, while the scrap code can be found in this link.

4.3. Dataset

We have 25 features per team and these features are:

'Clean sheets', 'Goals Conceded', 'Tackles', 'Tackle success 'Last man tackles', 'Blocked shots', 'Interceptions', 'Clearances', 'Headed Clearance', 'Clearances off line', 'Recoveries', 'Duels won', 'Duels lost', 'Successful 50/50s', 'Aerial battles won', 'Aerial battles lost', 'Own goals', 'Errors leading to goal', 'Assists', 'Passes', 'Passes per match', 'Big Chances Created', 'Crosses', 'Cross accuracy %', 'Through balls', 'Accurate long balls', 'Yellow cards', 'Red cards', 'Fouls', 'Offsides', 'Goals', 'Headed goals', 'Goals with right foot', 'Goals with left foot', 'Hit woodwork', 'Goals per match', 'Penalties scored', 'Freekicks scored', 'Shots', 'Shots on target', 'Shooting accuracy %', 'Big chances missed', 'Saves', 'Penalties Saved', 'Punches', 'High Claims', 'Catches', 'Sweeper clearances', 'Throw outs', 'Goal Kicks'

4.4. Pre-processing

In order to compare these features in our learning model, each feature is subtracted like so: 'home_feature' - 'away_feature'. In this way we have a total of 25 features in Dataframe X. Using the score difference, each piece of match data is labeled with ['home','draw','away'], to show the winner of the match. Now we have Dataframe X for feature, and Dataframe y for label.

4.5. Results of Classification without Feature Selection

We used GridSearchCV from the sklearn module to identify and tune an optimal hyperparameters set.

4.5.1. Support Vector Machine

We first tried a support vector machine on the player attributes. We used SVM from sklearn module. Hyperparameters used in this method were C, gamma, and kernel function. After comparison, the optimal hyperparameter was the Radial Basis Function for kernel function, with 9 for C and 0.0001 for gamma.

	C	gamma	kernel	Score
1	9	0.0001	rbf	0.553439
2	8	0.0001	rbf	0.553439
3	7	0.0001	rbf	0.552734
4	2	0.0010	rbf	0.552381
5	1	0.0010	rbf	0.552381

Table 5. Top 5 SVM result for GridSearch

4.5.2. Stochastic Gradient Descent

We then deployed a stochastic gradient descent algorithm to train our classifier model. SGD classifier from sklearn model is a linear classifier model with stochastic gradient descent training. Hyperparameters used in this method were loss function and max iteration. After comparison, the optimal hyperparameter was huber for loss function with a 4500 max iteration.

	loss function	max_iter	Score
1	huber	4500	0.525926
2	huber	500	0.508289
3	huber	2500	0.506173
4	huber	4000	0.505820
5	huber	1000	0.500529

Table 6. Top 5 SGD result for GridSearch

4.5.3. Decision Tree

We attempted a decision tree model in order to classify each match output. We compared various depth values and regularized the tree by setting the maximum depth at 2, which was found to be optimal. Best resulting tree is as

	max_depth	Score
1	2	0.529806
2	3	0.529453
3	4	0.520635
0	1	0.506173
4	5	0.504409

Table 7. Top 5 Decision Tree result for GridSearch

below:

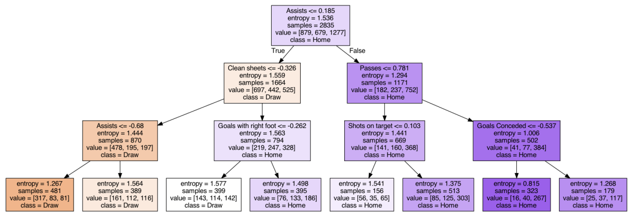


Figure 4. Decision Tree

4.5.4. Random Forest

Finally, we tried a Random Forest model in order to classify the match outputs. We compared various depth values and regularized the tree by setting the maximum depth as 2, which was found to be optimal.

	max_depth	Score
1	2	0.529806
2	3	0.529453
3	4	0.522046
0	1	0.506173
4	5	0.505115

Table 8. Top 5 Random Forest result for GridSearch

4.6. Results of Classification with Feature Selection

The Random Forest method was used in Feature selection. Based on importance, the top 12 features were selected, while the other features were dropped.

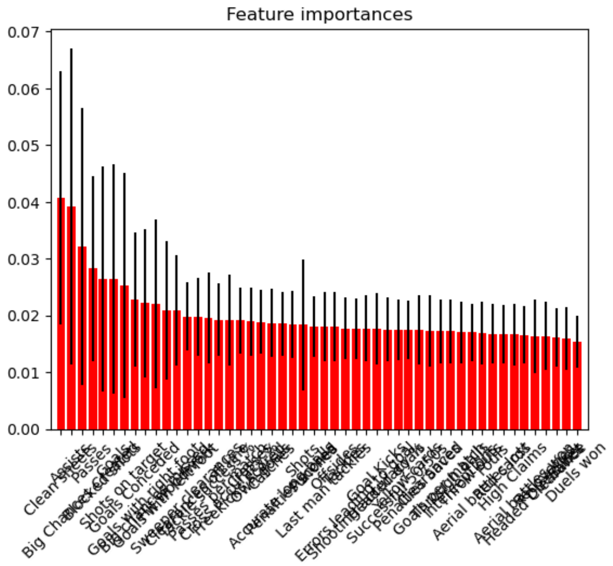


Figure 5. RandomForest Feature Importances

Features used in this classification are:

Clean sheets (0.041)
Assists (0.039)
Big Chances Created (0.032)
Passes (0.028)
Blocked shots (0.026)
Goals (0.026)
Shots on target (0.025)
Goals Conceded (0.023)
Goals with right foot (0.022)
Big chances missed (0.022)
Goals with left foot (0.021)
Hit woodwork (0.021)

4.6.1. Support Vector Machine

Since there are so many different attributes, we selected some features using the random forest classifier. Interestingly, the clean sheets score was found to be the most significant feature, while number of goals and assists followed behind. We deployed SVM with 15 highly correlated features and dropped other features.

	C	gamma	kernel	Score
1	3	0.0010	rbf	0.551323
2	1	0.0001	linear	0.551323
3	2	0.0001	linear	0.551323
4	2	0.0010	rbf	0.551323
5	2	0.0010	linear	0.551323

Table 9. Top 5 SVM result for GridSearch

4.6.2. Stochastic Gradient Descent

After feature selection, we deployed a stochastic gradient algorithm.

	loss function	max_iter	Score
1	log_loss	1000	0.548854
2	hinge	2500	0.548148
3	hinge	500	0.544621
4	log_loss	500	0.544268
5	log_loss	3500	0.542152

Table 10. Top 5 SGD result for GridSearch

4.6.3. Decision Tree

We also tried a decision tree model after feature selection. The best resulting tree is shown below.

	max_depth	Score
1	3	0.530511
2	4	0.525573
3	2	0.524868
4	5	0.513580
5	6	0.512169

Table 11. Top 5 Decision Tree result for GridSearch

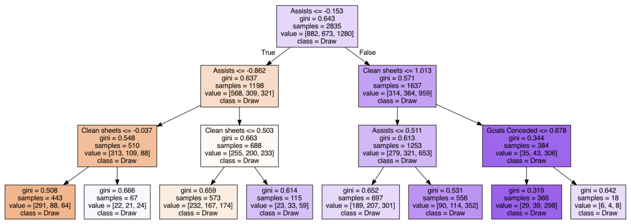

Assists <= -0.153 gini = 0.643 samples = 2835 value = [882, 673, 1280] class = Draw
True
False
Assists <= -0.862 gini = 0.637 samples = 1198 value = [568, 309, 321] class = Draw
Clean sheets <= 1.013 gini = 0.571 samples = 1637 value = [314, 364, 959] class = Draw
Clean sheets <= -0.037 gini = 0.548 samples = 510 value = [313, 109, 88] class = Draw
Clean sheets <= 0.503 gini = 0.663 samples = 688 value = [255, 200, 233] class = Draw
Assists <= 0.511 gini = 0.613 samples = 1253 value = [279, 321, 653] class = Draw
Goals Conceded <= 0.678 gini = 0.344 samples = 384 value = [35, 43, 306] class = Draw
gini = 0.508 samples = 443 value = [291, 88, 64] class = Draw
gini = 0.666 samples = 67 value = [22, 21, 24] class = Draw
gini = 0.659 samples = 573 value = [232, 167, 174] class = Draw
gini = 0.614 samples = 115 value = [23, 33, 59] class = Draw
gini = 0.652 samples = 697 value = [189, 207, 301] class = Draw
gini = 0.531 samples = 556 value = [90, 114, 352] class = Draw
gini = 0.319 samples = 366 value = [29, 39, 298] class = Draw
gini = 0.642 samples = 18 value = [6, 4, 8] class = Draw

Figure 6. Decision Tree

4.6.4. Random Forest

We tried the Random Forest model after feature selection.

	max_depth	Score
1	3	0.530511
2	4	0.525573
3	2	0.524868
4	5	0.514286
5	6	0.511111

Table 12. Top 5 Random Forest result for GridSearch

4.7. Accuracy

Accuracy is the portion of correctly classified data over the whole dataset.

Model	Score before feature selection
SVM	0.60
SGD	0.56
DT	0.57
RF	0.57

Table 13. Validation Score before Feature Selection

Model	Score before feature selection
SVM	0.57
SGD	0.53
DT	0.53
RF	0.53

Table 14. Validation Score after Feature Selection

5. Conclusion

5.1. Match Attribute Approach

The number of goals can vary very wildly between each individual match for many reasons. In addition to that, teams might change their tactics mid-game depending on the current score and what result they are trying to achieve. For example, a team that is currently winning may try to sit back and play more defensively, which leads to less goals, while

a team that is losing will pressure harder in an attempt to tie the game. Therefore, we need to edit the prediction model to consider both teams' scores and the final result. This should lower the error and variance significantly.
Another change we need to do is to add a weighted model to the data to consider more recent results more heavily. We have data from 16 seasons, but the team rosters and results have changed very significantly in that time frame. By weighting data according to more recent results, we can more accurately predict a team's current strength and form. We are currently considering two possible models for weighting the data: linear and exponential. We will attempt both in addition to looking at other potential models for weighting data by recency.
As we observed, it seems that our model is too simple to learn about the appropriate structures of the data. To improve the performance of our model, we should provide more appropriate features by utilizing feature engineering, such as appending each team's statistics and player information. On the Premier League site, we can use our heuristics and find all information above that we could analyze and use to make our predictions better. Plus, since our model is underfitting the data, we could possibly try stronger models with more parameters. We may try polynomial regression with an analysis of the learning curve with respect to the highest order term.

5.2. Player Attribute Approach

There are 25 attributes per player, which results in very large datasets. So, we applied Random Forest when selecting features. However, the scores after feature selection decreased compared to before. There are a few more feature selecting methods, and other methods could be used to get better results. We think random forest was not an appropriate method and with a more appropriate method, the score could increase.
With Support Vector Machine, the score was 0.6, which is the best of all of our attempted methods. Tree based methods performed worse than other methods. The reason for this performance is that we had only three results to classify each match into. It could be not classified as well through the tree when deciding which team will win.
When we first changed our approach, the dataset was the main problem. Scrapping the first time took a long time, and after the scrap was done, it was December. So there was not enough time to analyze the player attributes more thoroughly. There is not enough analysis within each individual player attribute before classifying. If we were to do more analysis on each attribute and select a better classifying method, we could more accurately analyze the data to determine which features are the most important.

6. Future Direction

First, the methodology should be augmented. More suitable processing of data using machine learning techniques or experts' heuristics might help. As scores are between zero and two in most of the matches, our model was quite accurate on those matches. However, there are some outlier matches with more than four points and these data points tended to disturb the model greatly. Therefore, applying local models to handle these outliers could also possibly help.
Since many adversarial sports games have similar structures and attributes, we hope researchers could apply transfer learning to predict or analyze matches utilizing the information from the trials and errors we have made through this project. Moreover, sports agents, professional teams, sports analyzers, or sportscasts could possibly deploy these types of methodologies to predict upcoming soccer(or any other sports) matches.

7. Timeline

7.1. October

Planned for experimentation. Data collection, data cleansing, data processing, and selection of learning model.

7.2. November

Done collecting match data. Match attribute approach - trained, fixed, and validated model.
More data collection and processing for player attribute approach.

7.3. December

Player attribute approach - trained, validated, and tested some classification models. Summarization and postmortem.

8. Code Repository

GitHub repository

9. References

M. Silverio,(2020) "My findings on using machine learning for sports betting: Do bookmakers always win?", https://towardsdatascience.com/my-findings-on-using-machine-learning-for-sports-betting-do-bookmakers%-always-win-6bc8684baa8c.

M. C. Purucker, "Neural network quarterbacking," in IEEE Potentials, vol. 15, no. 3, pp. 9-15, Aug.-Sept. 1996, doi: 10.1109/45.535226.

A. McCabe and J. Trevathan, "Artificial Intelligence in

Sports Prediction," Fifth International Conference on Information Technology: New Generations (itng 2008), 2008, pp. 1194-1197, doi: 10.1109/ITNG.2008.203.

Tax, Niek & Joustra, Yme. (2015). Predicting The Dutch Football Competition Using Public Data: A Machine Learning Approach. 10.13140/RG.2.1.1383.4729.

Dorhauer, Adam. (2017). The Math of Weighting Past Results, https://tht.fangraphs.com/the-math-of-weighting-past-results/.