English Premier League Games and Standings Projections Model

Jeff Rathgeber, Sunan Tajwar

NYU Tandon School of Engineering

CS-UY 3934: Mining Massive Datasets

Professor Gustavo Sandoval

May 9, 2023

TABLE OF CONTENTS

Problem Description and Hypothesis

The English Premier League consisting of 20 English teams including the likes of Manchester United, Liverpool, Manchester City, and Arsenal, is the most-watched football league in the world at the club level. With a total of 380 games played each season, 20 teams play each other on a weekly basis, each matchup being played twice, with each team hosting each matchup on their home ground at least once. The Premier League, as we know it today, has existed since 1992, but these clubs have played against each other since the 1800s. The popularity of the league has led to a plethora of second and third-level statistics, ranging from player activity in the final third to statistics on aerial duels won, that allow clubs to evaluate their strategies and personnel amongst matchups, and give viewers a sense of hope and basis for the €68 billion but into gambling on the English premier league annually (as recorded in 2019-2020). In this paper, we look to observe some of the more surface-level features such as shots on target, corners, fouls, and average goals scored and conceded for teams in fixtures both home and away. To do so we attempt to mold the premier league fixtures into five machine-learning models using the K-Neighbors classifier, XGBoost classifier, Random Forrest classifier, Support Vector Machines, and a Neural Network. In doing so we strive to predict the outcome of individual games, which then leads us to establish a projection on final season standings. We believe by consolidating comparative statistics in each fixture, we will be able to train a model given a set of two teams' statistics, which will allow us to generate an accurate prediction on the match outcome and thus that make the most profitable bets based on weekly premier league money lines from a variety of markets.

Dataset Description

1. Data Extraction

Our dataset consists of sets of 380 match seasons in which we have access to all fixtures in a given season in addition to 24 match descriptive features, 12 per home/away team. These features included team shots, shots on target, corners, fouls committed, offsides, free kicks conceded, yellow cards, red cards, and goals. Some of these features, such as shots, also were split into half-time and full-time statistics. The dataset we used to train our model on was a consolidation of the past twelve premier league seasons dating back to the 2009-2010 season, in which this data was collected. For each fixture, we also had multiple fixture money lines and team odds statistics, derived from various gambling-authorized organizations, including Bet365 and BetBrain. We will be testing data on fixtures of the second half of the current 2022-2023 premier league season.

2. Pre-processing

Given the various features of match statistics provided to us by the original dataset, through feature correlation analysis we determined that we should keep the majority but not all of the features, to train the model. The features we selected were full-time home goals, full-time away goals, full-time result, half-time home goals, half-time away goals, half-time result, home team shots, away team shots, home team shots on target, away team shots on target, home team corners, away team corners, home team fouls committed, away team fouls committed, home team free kicks conceded, away team free kicks conceded, home team yellow cards, away team yellow cards, home team red cards, and away team red cards. The feature we are trying to classify and predict is ‘full-time results’, and it is currently classified as ‘H’, ‘A’, or ‘D’ to indicate whether the home team won, the away team won, or if there was a draw. To fit the

constraints of our model, we numerically encoded the full-time result values ‘H’, ‘A’, and ‘D’ as 0, 1, and 2 respectively. Lastly, we also eventually scaled our training data using a Standard Scaler to try and scale all the features we are learning towards unit variance and prevent any significant influence from outlying statistics or teams.

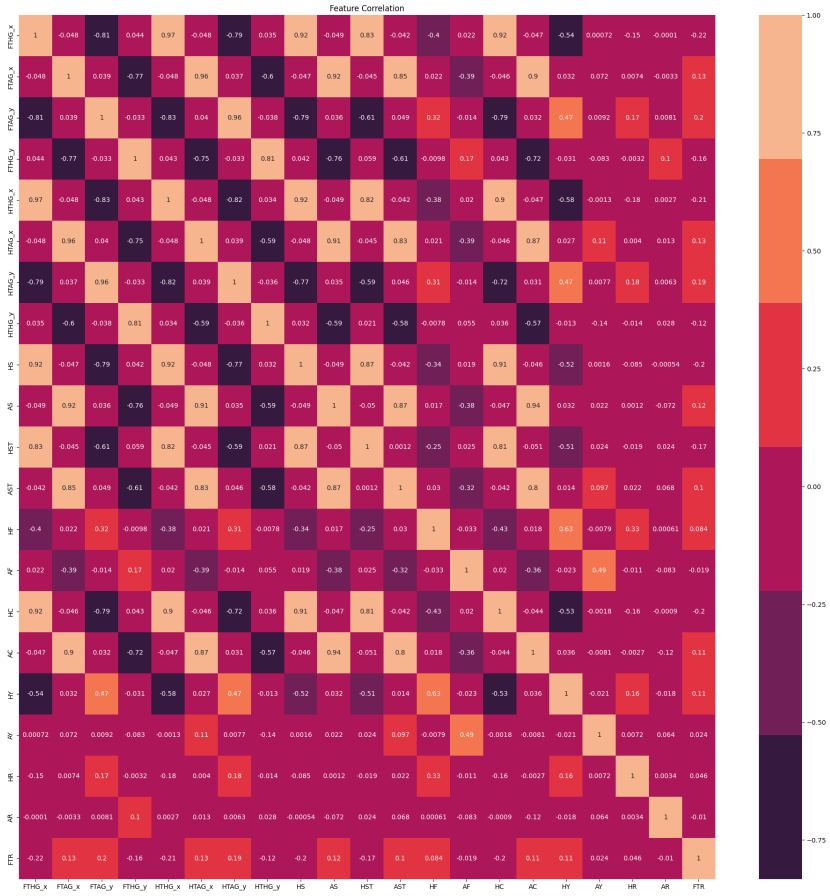

Figure 1: Feature Correlation Matrix

Upon further inspection, we quickly learned that, of the features that showed some correlation to the target, the majority of features had a linear relationship to the number of points a team would end the season with. Below are depicted a few examples of these relationships, and all examined features generally fell into one of the categories: positive, negative, or seemingly unrelated to the naked eye. Although most of these turned out to be fairly intuitive (i.e. teams that score more

goals and concede less end with more points), the various models to be discussed serve the purpose of learning which of these relationships between feature(s) and target are the most relevant, and they can even find more complex relationships that may appear in linear feature space to be nonexistent.

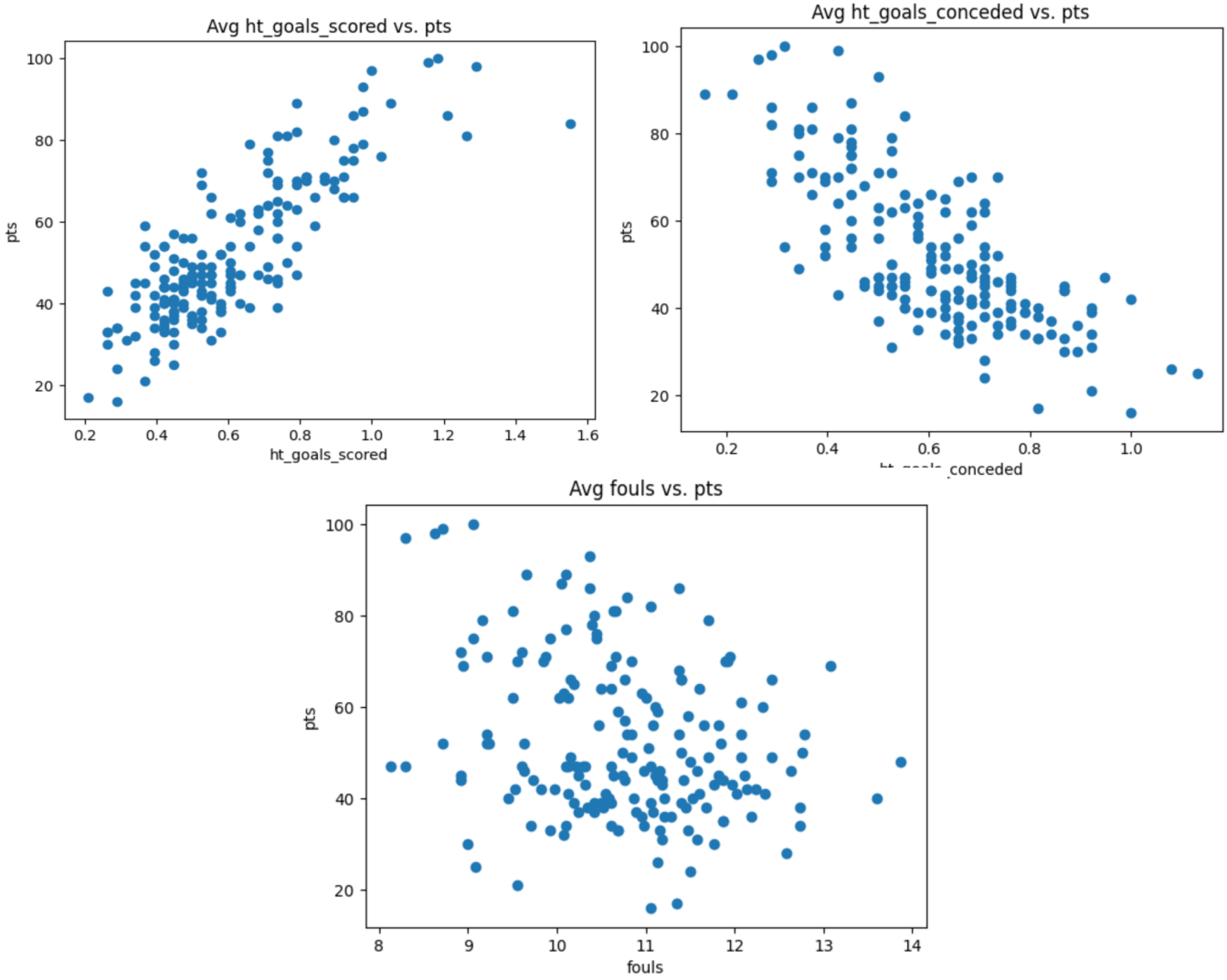

Figure 2: Half-Time Goals Scored/Conceded and Average Fouls vs Season Points

3. Feature Engineering

In order to quantitatively measure team statistics and how they play in individual features, we had to do some feature transformations which were merged into a fixtures dataset that could comparatively analyze the attacking and defensive statistics of each team in a matchup, taking into account whether they are home or away in the fixture. Our objective was to be able to create a predictive model based on recent historical data which could emulate a team's current or season form. We then began the feature engineering by forming a dataset with the average of the statistics for each of the teams in the twelve-season training dataset. We did this for both the teams' home and away performances. We then proceeded to form a cumulative team table, in which we merged these two tables together using the teams as an index. In doing we also engineered some features such as average goals conceded home and away at half-time and full-time. After doing so, we began forming the fixtures/matchups table. We did so by merging home and away statistics per row in the dataset using the 'Home Team' and 'Away Team' columns as indices. Each row was thus a qualitative representation of comparative mean statistics per matchup, taking into consideration home and away performances and team form. After making the matchup table, we removed the team names prior to training the data so that the team name or historical performance would not bias the data when training the model.

Model Evaluation

1. K-Nearest Neighbors

The first model we trained on the 12-season dataset was the K-Nearest Neighbors classifier. This classifier determines a classifier by using a distance calculation between datapoints to determine the ‘K’ nearest neighbors for a given data point and creates these classified groupings iteratively. The training dataset was split into training and test sets with an 80-20 split. The training data sets had 3648 matchup instances, and the test data set had 912 matchup instances. When initializing the K-Nearest Neighbors model, we hyperparameters we tuned were ‘n_neighbors’, or the number of neighbors used in the algorithm, and the ‘weight’ calculation method of the data points relative to each other. We did so using the GridSearchCV in which we selected the best combination of trained hyperparameters, and evaluated themes based on their mean five-fold cross-validation accuracy. The best combination of parameters for this model was 49 neighbors, using the Euclidean weight calculation method, which resulted in a mean-accuracy score of 52.28%.

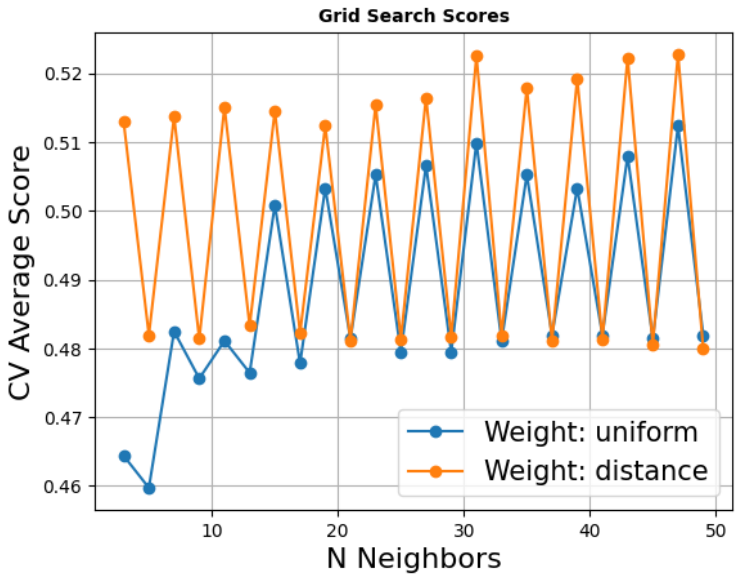

Figure 3: KNN CV Avergae Scores

We then used the model with the optimal parameter to generate a prediction on the test data, which resulted in an accuracy score of 50.22%. It also had a precision, recall, and F1 scores of 0.42, 0.46, and 0.43 respectively. Given the relatively low but similar training and testing accuracy scores, it is reasonable to say that the model was underfitted. This is most likely because the features were not complex enough to accurately model and predict the multitude of variables in a premier league match. The model predicted wins, losses, and draws from the perspective of the home at very similar rates.

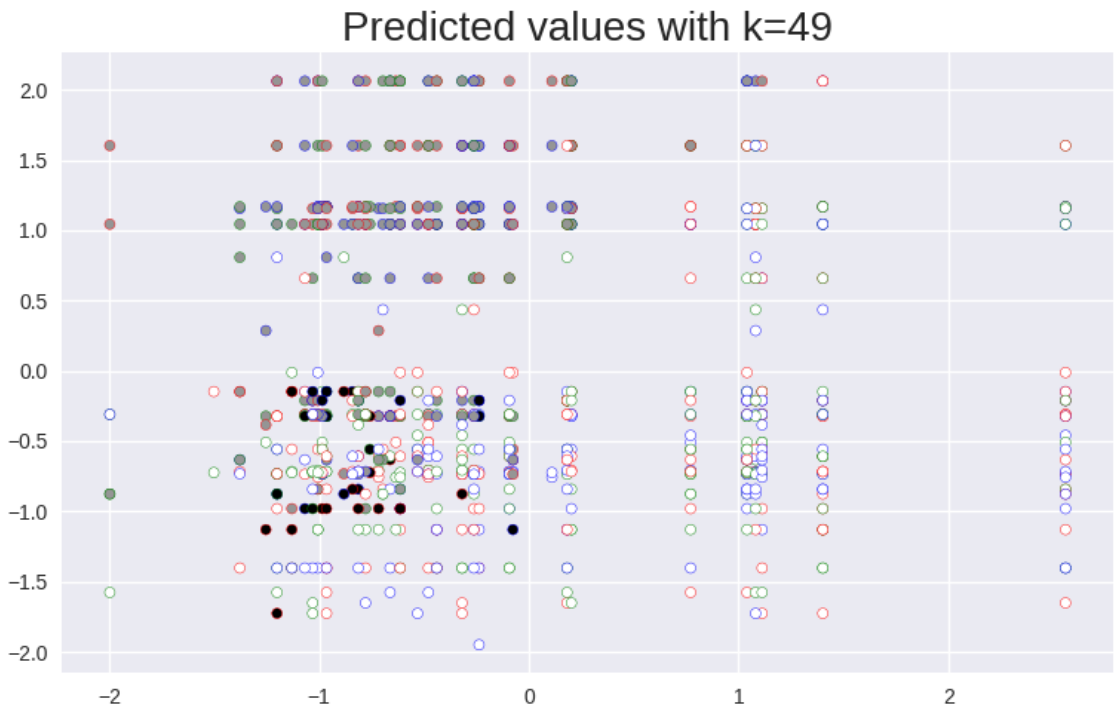

Figure 4: KNN Classification Results

Green - Win, Red - Loss, Blue - Draw (From Home Team Perspective)

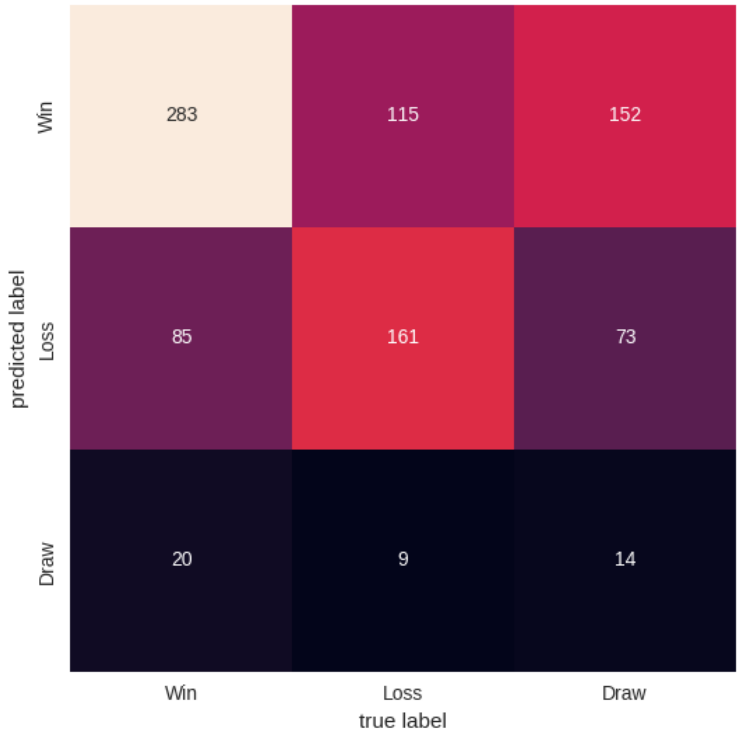

Figure 5: KNN Confusion Matrix

We also trained the K-Neighbors Classification model after a polynomial feature transformation of degree 2. With this, the optimal parameters were 48 for ‘n_neighbors’ and ‘uniform’ for the weight calculation, which had the best mean cross-validation score of 52.41%. This model had an essentially equivalent test accuracy score of 50.1%, which would again suggest an underfit model.

We used this trained model to predict the unplayed premier league matches this season and then used the predicted results for each match to generate the projected final season standings based on our prediction.

HomeTeam	Predicted Points
Arsenal	88
Man City	85
Newcastle	77
Man United	71
Brighton	70
Tottenham	65
Liverpool	59
Brentford	55
Aston Villa	53
Fulham	51
Wolves	46
Chelsea	45
Crystal Palace	42
Bournemouth	42
West Ham	42
Leicester	40
Leeds	38
Everton	36
Nott'm Forest	33
Southampton	32

Table 1: KNN Final Season Standings Projection

2. XGBoost

The second model we trained on the 12-season dataset was with an XGBoost Classifier, which implements a gradient-boosted decision tree. This tree is an ensemble classification model/tool, which utilizes subsequent decision trees, each tree building off of the errors and what it learned from previous models. It builds from weak base learners into strong learners. We used the same training and testing dataset splits and set sizes as the previous model, and used GridSearchCV to tune the algorithm's hyperparameters.

Figure 6: XGBoost Ensemble Decision Tree

The hyperparameters we tuned were ‘max_depth’ which indicated which sets the maximum depth of the decision tree, and ‘n_estimators’ which is the number of runs XGBoost will run to try and learn. The optimal parameters were a max depth of 2 and 100 ‘n_estimators’. When using the model with the optimal hyperparameters on the test set, the prediction accuracy was 51.1%. It also recorded precision, recall, and F1 scores of 0.47, 0.52, and 0.46 respectively. Again we can make the observation from the cross-validation scores and from the training and testing accuracy being relatively similar, which would indicate that model was underfit as it lacked variance, but recorded a high error even in the training set. This would again suggest that the features and metrics used are simply not complex enough to model the full intricacies of a Premier League match. However, from the graph of mean cross-validation scores, we can observe that as the max depth increases, the model starts to overfit the data.

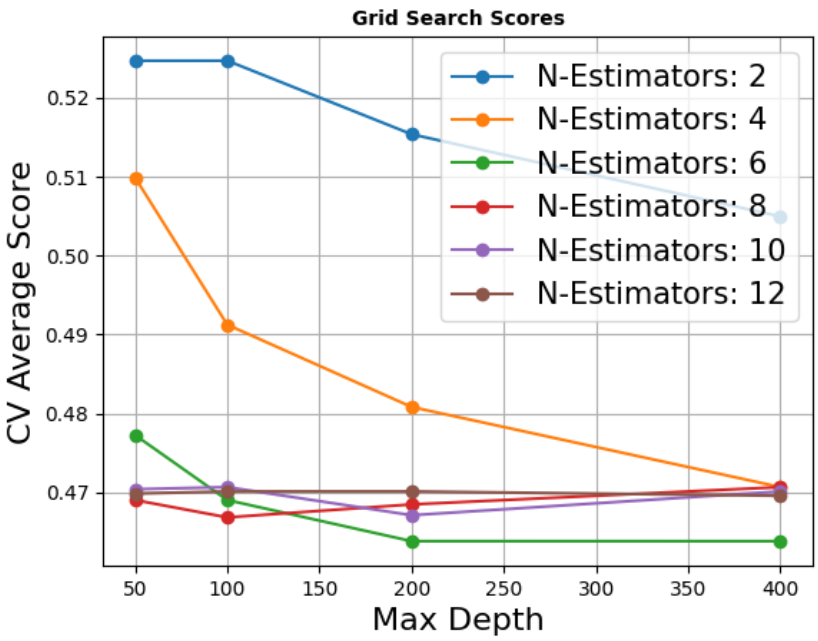

Figure 7: XGBoost CV Average Scores

Lastly, we again used the trained model to predict the outcome of the unplayed Premier League games this season. Using these results we generated a projection for the final season standings of the 2022-2023 Premier League season.

HomeTeam	Predicted Points
Arsenal	85
Man City	82
Man United	71
Tottenham	68
Newcastle	68
Aston Villa	59
Brighton	58
Liverpool	56
Brentford	55
Chelsea	51
Fulham	51
Wolves	46
Crystal Palace	45
Bournemouth	42
West Ham	42
Leeds	41
Nott'm Forest	39
Southampton	38
Leicester	37
Everton	36

Table 2: XGBoost Final Season Standings Projection

3. Random Forest

The next model we tested utilized a Random Forrest Classifier, which is essentially a collection of decision trees, each sampling a subset of features and data points. The aggregation of the decision trees is made to prevent overfitting relative to a standard decision tree, and thus create a more robust model. Final classifications using the Random Forest model are based on a majority vote among the decision trees. In our model, GridSearchCV was used once again to tune the model's hyperparameters which included 'n_estimators', 'max_features', 'max_depth', and 'criterion'. The 'n_estimators' and 'max_depth' hyperparameters had the same purpose as described in previous models. The 'max_features' parameter is the function used to determine

the number of features to consider when looking for the best split during the classification, and the ‘criterion’ parameter is the function used to measure the quality of the split, and for our model, we were choosing between the ‘gini’ and ‘entropy’ criterion. Upon running the GridSearchCV algorithm, we determined that the optimal hyperparameters for the model were 400 ‘n_estimators’, a ‘max_depth’ of 10, a ‘max_features’ constraint of ‘sqrt’, and the ‘gini criterion’. This combination of hyperparameters achieved a mean cross-validation score of 49.92%.

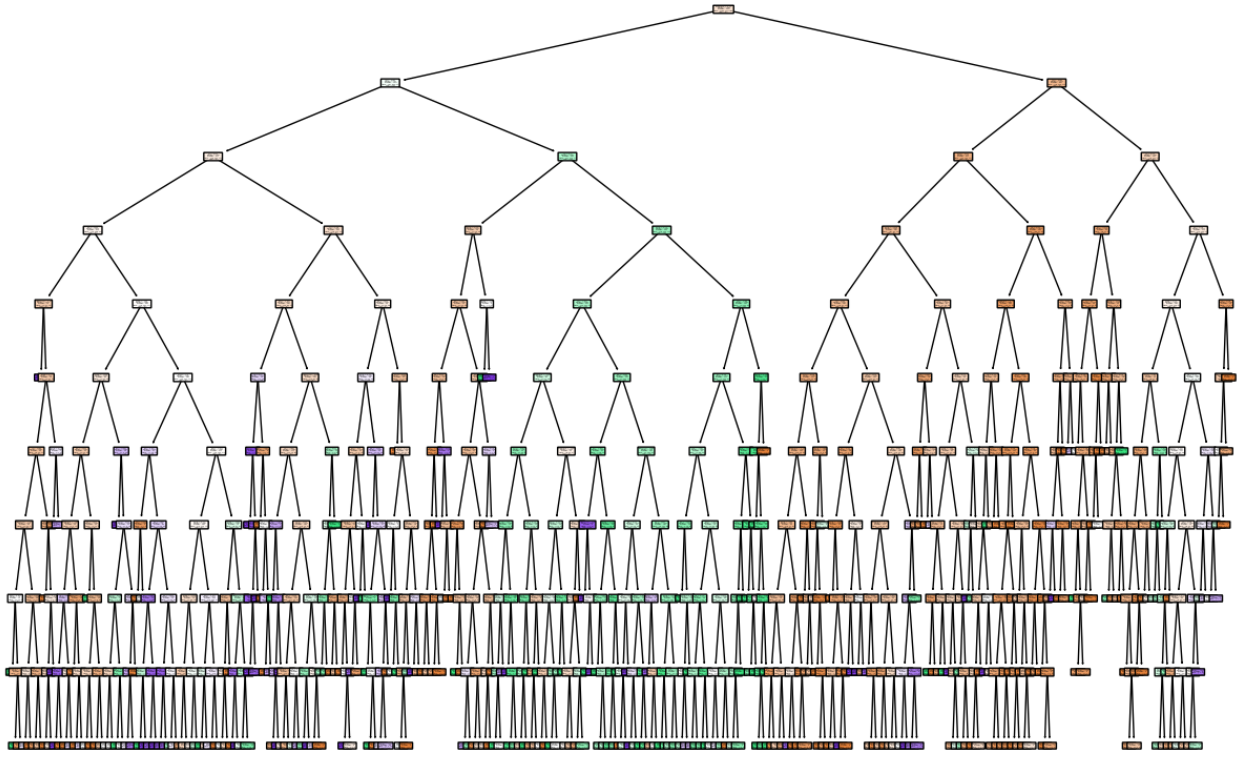

Figure 8: Random Forest Ensemble of Decision Trees

Upon training the model using the 12-season dataset, we tested the model with optimal parameters on the test set and achieved an accuracy score of 46.1% and had precision, recall, and F1 scores of 0.42, 0.46, and 0.43 respectively. This would suggest an even more underfit model than the XGBoost classifier which is a singular decision tree, as we have a lower accuracy score,

but still minimal variance between the training and test accuracy. This would suggest again that there were not significant enough splits in the weighing and combination of the features using a consolidation of multiple decision trees that resulted in a significantly higher accuracy score. Although the variance remained low, this was not an issue with the other models either. Together this again suggests that the lack of complexity of the features was not best suited accurately model the outcomes of premier league games effectively.

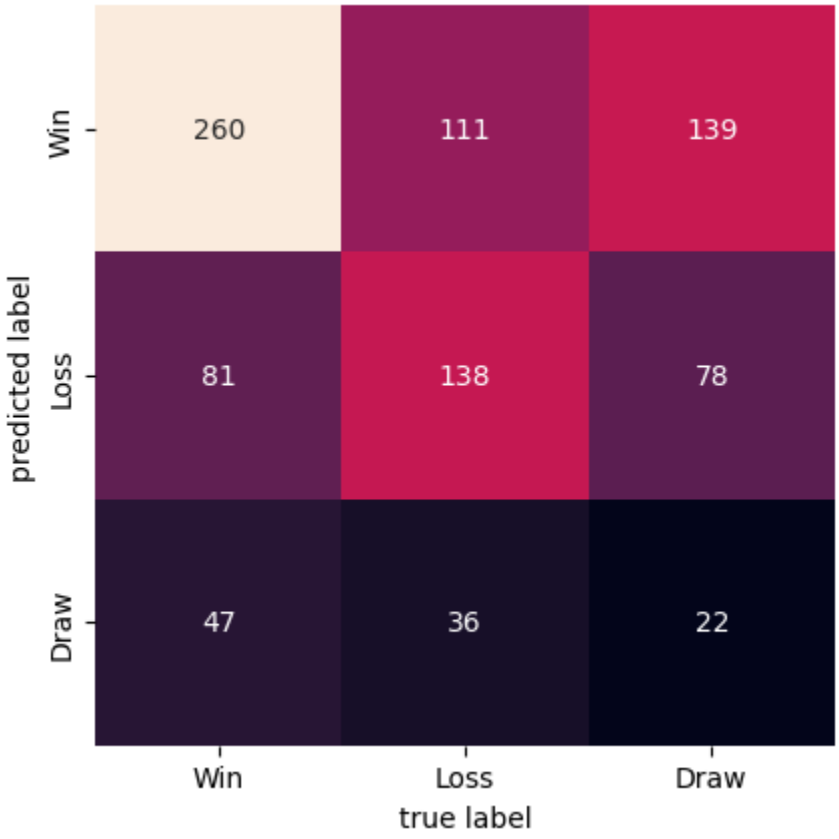

Figure 9: Random Forest Confusion Matrix

Nonetheless, we still used this model to predict the outcomes of the unplayed matches for this premier league season and used those predicted results to generate a projection for the final season standings.

HomeTeam	Predicted Points
Arsenal	85
Man City	79
Man United	71
Tottenham	68
Newcastle	68
Aston Villa	59
Brighton	58
Liverpool	56
Brentford	55
Chelsea	51
Fulham	51
Wolves	46
Crystal Palace	45
Bournemouth	42
West Ham	42
Leeds	41
Everton	39
Nott'm Forest	39
Southampton	38
Leicester	37

Table 3: Random Forest Final Season Standings Projections

4. Support Vector Machines

The goal of support vector machines is to use a relatively small portion of training examples (support vectors) chosen from the rest to optimize a hyperplane for either classification or, in this case, regression analysis.

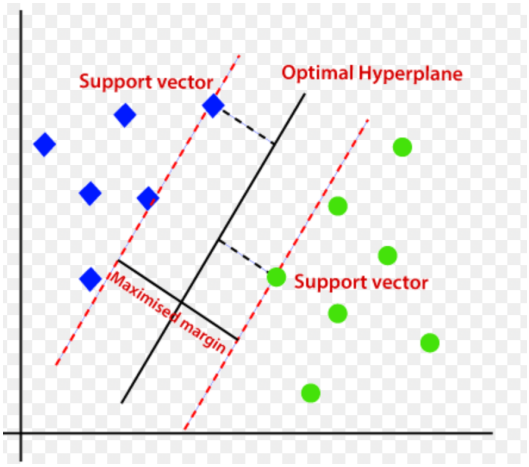

Figure 10: Support Vector Machine Diagram

Given the relatively low complexity of the relationship between our features and target, the support vector machine model turned out to be slight overkill, as SVM's are particularly good at learning nonlinear relationships upwards of 5 and even 10 degrees. In this case, where most of the data was linearly related, the RMSE did not change any greater than precision 0.00001 when changing the degree of the model; effectively, it did not change.

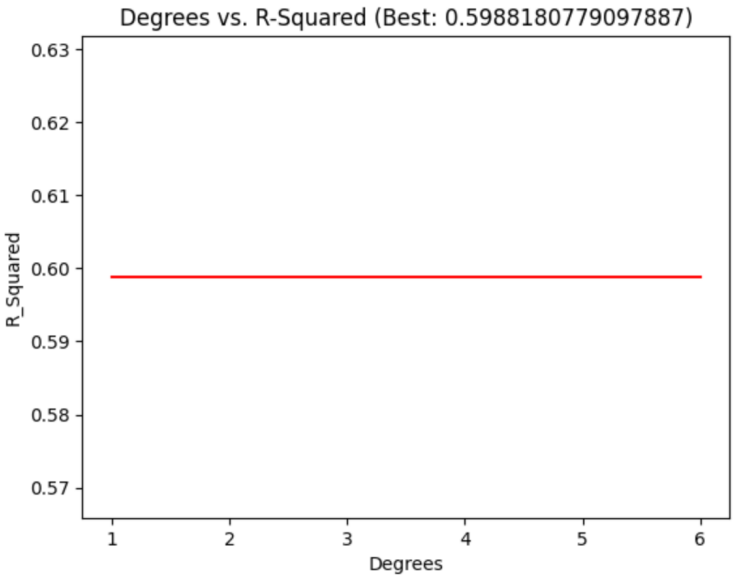

Figure 11: RMSE vs. SVM Degree

Every SVM model we tried would underfit the data to various degrees, but this lack of compatibility became even more evident when testing on the data from the current season. Below are the results with known standings from the 2019-2020 season (left) as well as the projected results from this season (right).

2019-2020 Season

	Team	Predicted	Actual
0	Liverpool	62.330434	99
16	Man City	62.451925	81
18	Chelsea	63.138543	66
9	Man United	62.756657	66
7	Leicester	57.414483	62
6	Tottenham	57.569005	59
19	Wolves	46.897825	59
10	Arsenal	53.730430	56
3	Burnley	43.557299	54
17	Sheffield United	49.014929	54
15	Southampton	46.231294	52
13	Everton	49.613696	49
8	Newcastle	42.314274	44
4	Crystal Palace	42.899491	43
12	Brighton	41.733054	41
1	West Ham	45.767222	39
11	Aston Villa	42.109512	35
5	Watford	41.264577	34
2	Bournemouth	41.574266	34
14	Norwich	41.120836	21

Predicted 2022-2023 Season

	Team	Predicted
16	Brentford	53.727656
4	Newcastle	53.727649
13	Man City	53.727647
19	Liverpool	53.727647
12	Brighton	53.727647
11	Arsenal	53.727647
8	Man United	53.727647
5	Tottenham	53.727647
3	Leeds	53.727647
0	Crystal Palace	53.727647
17	Nott'm Forest	53.727647
15	Wolves	53.727647
18	Chelsea	53.727647
14	Southampton	53.727647
10	Aston Villa	53.727647
1	Fulham	53.727647
2	Bournemouth	53.727647
7	Leicester	53.727647
6	Everton	53.727647
9	West Ham	53.727559

Table 4 & 5: SVM 19-20 (L) and Current Season (R) Standings Projections

As can be seen in the data, the target range of the 2019-2020 season was almost 4 times larger than the range produced by the projected results from the model. As for the current season, the standard deviation was a matter of over 7 decimal places, but nonetheless still managed to produced a final standing that made sense for most teams, especially in the bottom half of the table.

5. Neural Networks

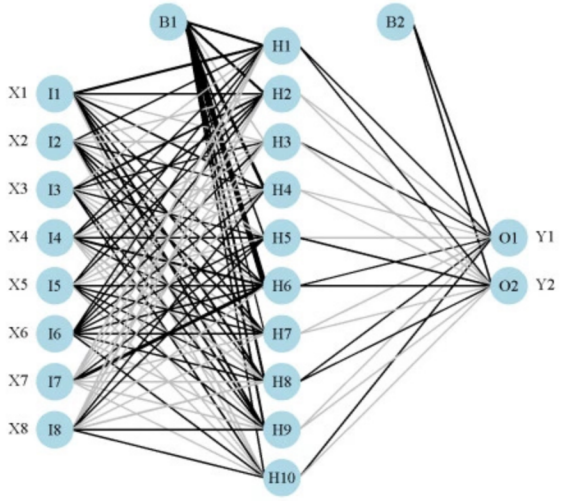

Figure 12: Example Neural Network Diagram

Lastly, we decided to train a neural network regressor which combines tens to upwards of hundreds of neurons, each with their own set of weights to train. This complex system allows the model to be able to learn very complex relationships between variables, especially nonlinear. We realized very early that most of the features we were working with had linear relationships, and it became evident that in training the neural network, these correlations were learned very quickly, and after about 100 training iterations very little improvement was able to be made.

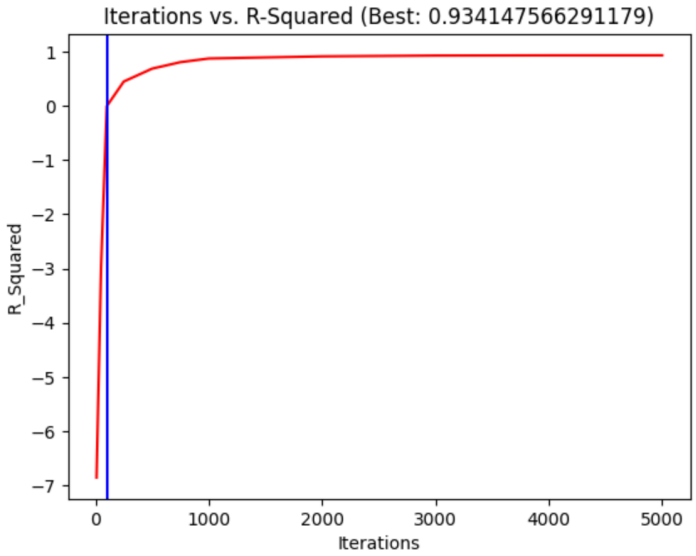

Figure 13: RMSE vs. # Training Iterations

After experimenting with various hyperparameters, the best mean squared error we were able to achieve was roughly 0.934. This number means that ~93.4% of the variance observed can be explained by the predicting variables. As it turned out, the model with the highest RMSE turned out to overfit when testing on the current incomplete Premier League season. It preserved a reasonable ordering of the teams final placement, but the predicted point values exaggerated the difference between each standing. For more the a few thousand iterations, the predicted point values for the first and second place team would be upwards of 130 to even over 160, when the max amount of points a team could get from a 38 games season is 114 (38 wins times 3 points per win). Similarly, decreasing the amount of iterations too low, such as below 10, did not give the model the chance to learn well enough not to underfit the data, and the standard deviation of the predicted scores was exponentially lower. Settling on 100 iterations bore values that were representative of both the training data, and what could be expected to be observed in the real world. These are the final predictions:

Predicted 2022-2023 Season

:

	Team	Predicted
11	Arsenal	92.457777
13	Man City	90.673106
5	Tottenham	89.857748
19	Liverpool	89.117420
8	Man United	87.792983
12	Brighton	86.640760
3	Leeds	84.465512
4	Newcastle	83.709225
18	Chelsea	80.510791
10	Aston Villa	79.301195
14	Southampton	78.410305
15	Wolves	77.900903
0	Crystal Palace	77.223720
1	Fulham	76.636499
7	Leicester	76.445451
17	Nott'm Forest	74.500422
9	West Ham	74.053749
6	Everton	72.679697
16	Brentford	71.387796
2	Bournemouth	69.409503

Table 6: Neural NEtwork Final Season Standings Projections

Conclusions and Discussion

The common theme throughout all of the models we trained was that there was very minimal variance presence as we transition from training to validation to testing data, but there was always a significant amount of bias and high error during the training. This would strongly suggest that our models were simply under fitted and could not precisely model all the variables that factor into the outcome of a Premier League match. This claim is backed by the rather low precision, recall, and F1 scores observed in the K-Nearest Neighbors, XGBoost, and Random Forrest classifiers. The 0.93 R-squared value observed in the Neural Network Model, suggests that roughly 93% of the variance observed can be attributed to the training data, and since we had minimal variance, we may consider our data to be somewhat surface-level in its analysis and breakdown of premier league matches. More complex features that illustrate the intricacies of the game such as individual duels won, key possessions and chances created, managerial tendencies, and even a club's financial expenses in different parts of the pitch can help model the various intricacies of each fixture rather than our somewhat simple consolidation of shots, fouls, corners, half-time performances, and home/away performances.

The overwhelming final season projections of our models would suggest that Arsenal and Manchester City will finish top-2 this season in that order respectively. In addition, teams such as Everton and Nottingham Forrest also appear to be common bottom-three candidates for relegation amongst all of our models as well. In comparison with the current season standings, this would seem to be a relatively well-fitted model at the extremes of the table, but it is the minor ordering and intricacies in mid-table where our models begin to falter and inaccurately calculate the number of points of projected standings of teams.

Although our model may not have been the best fitted regarding as it demonstrated high bias, when using it to predict the outcome of every match from the past 12 seasons while placing a $10 bet using Bet365 odds on who we predicted to be the winner, our model simulation generated a profit for the K-Nearest Neighbors, XGBoost, and Random Forest Classifier models. The simulation results were total winnings of $25,095.70, $29,647.70, and $40,515.60 for the K-Nearest Neighbors, XGBoost, and Random Forest Classifiers respectively. Interestingly enough, although the Random Forest Classifier had lower training and testing accuracy scores than the other two models, it produced the highest winnings during the simulations. This may suggest that the Random Forest model may have been better suited to predict matches with more skewed odds, which leads to higher payouts.

Ultimately our models statistically were not complex enough to accurately model the beautiful yet unpredictable game. However, in a practical measure, our models were able to predict the top and bottom of the Premier League table relatively well and have demonstrated utility through the Bet365 simulations in being able to generate winnings with model accuracy scores around and above 50%. The possible next steps in our model would be to integrate more complex features as mentioned previously, as we attempt to better model different aspects of every Premier League match from the flow of the game, team form, tactics, and individual battles. We would hope that incorporating such features in addition to more historical data, would help better train the model to fit more complexities of the game, generating a greater number of accurately predicted outcomes. In addition, being able to stochastically update the model as games are played may also help improve its accuracy and converge towards a more optimal model and its parameters. Lastly, the expansion of our data and features may benefit from scaling up our models allowing them to further tune their hyperparameters and computations necessary to fit the model well.

References

❖

- https://www.football-data.co.uk/englandm.php
- https://www.ibm.com/topics/knn
- https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/
- https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest
- https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html
- https://scikit-learn.org/stable/modules/svm.html