Predicting Outcomes of the Premier League Matches Using Real World and Computer Game Data

Zae Myung Kim **Jooyong Lee** **Juil Chung**
University of Minnesota Twin Cities
`{kim01756, lee02628, chung557}@umn.edu`

1 Introduction

There has been a lot of interest in predicting match outcomes of football (or soccer) games as the sport is hugely popular with a large fan base and enjoys much media attention. Predictions of match outcomes may be useful for stakeholders within the sport, such as managers, journalists, analysts, bookmakers, etc., as such information can be utilized to make risk assessments and plan strategies.

The majority of studies on this topic involve making use of statistics from players and matches, for example, the number of goals scored by a player in a season and the ball possession rate of a team within a match. In this project, we additionally make use of in-game data for players (e.g. finishing ability, passing ability, etc.) from a football simulation game, "Football Manager (FM)," to construct a richer representation for each player. Experimental results confirm that these additional data indeed provide machine learning models with additional meaningful information and boost their performance.

In terms of modeling, we devise a powerful prediction model using a heterogeneous convolutional graph which is an extension of popular graph convolutional networks, allowing us to incorporate varying types of nodes and edges.

2 Problem Setting

We denote a list of HOME players to be $p^h_1, ..p^h_N$ and AWAY players to be $p^a_1, ..p^a_N$ where N is the number of players in each team participating in a match. Each player p_i is represented by an array of d_p features, $p_i \subset \mathbb{R}^{d_p} \in \mathbb{P}$. A team t_i is represented by a randomly initialized embedding of size d_p, and is defined by relations with its players and matches against other teams. The first scenario is to predict a match outcome $y \in \{\text{WIN}, \text{DRAW}, \text{LOSE}\}$ given players ($p^h_{1..N}$ and $p^a_{1..N}$). The second scenario is similar to the first, but the input is augmented with in-game statistics $s \subset \mathbb{R}^{d_s}$ such as ball possession, shots, passes, touches, etc.

3 Approach

As the data is in tabular format, we will first explore this supervised-learning problem with traditional machine learning models, namely, support vector machines (SVM) and Random Forests (RF). Our main modeling contribution is the design of a heterogeneous graph convolutional network (HGCN) that models team-player and team-team relations. Figure 1 shows an example of the network for a single match between home and away teams. The network is heterogeneous because we have three types of nodes: *goalkeeper*, *kicker*, and *team*. Each player (*goalkeeper* and *kicker*) *PlayFor* a *team*, and each *team Uses* 11 players per match. Between the two teams, three types of edges exist: *Win*, *Draw*, and *Lose*. We note that *Win* and *Lose* edges co-occur in pairs, and in the case of a draw, two *Draw* edges exist. Over the matches in each season, a large overall network is drawn by connecting these individual match networks.

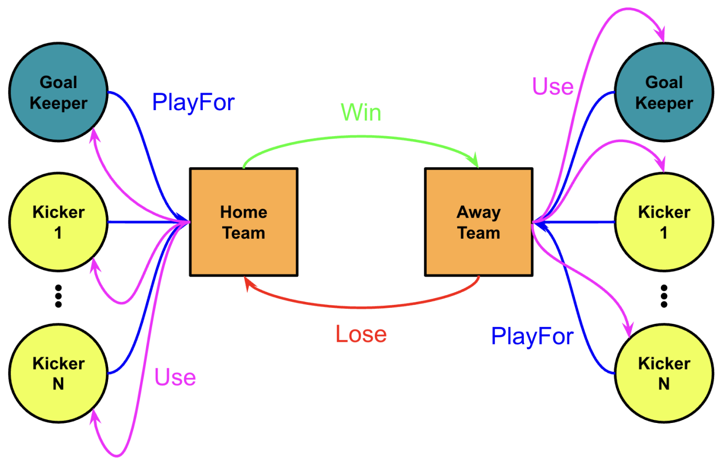

Figure 1: Architecture of heterogeneous graph convolutional network

1st-level attributes	**2nd-level attributes**
entity (dict)	player ID, player information, national team, birth, age, name
stats (dict)	goals conceded, goals, assist, red card, pass, mins played, ...

Table 1: Details of a sample data in *all_players_official_stats.json*

4 Dataset Construction

4.1 Web Crawling

To train and test the prediction models, real-world data, such as match results and player-team statistics, are necessary. While these data had been already collected and made available on some websites such as Kaggle, they are largely out of date[1]. Thus, we crawled the following most recent data from the official website of the Premier League: (1) player statistics[2] and (2) match results[3].

In addition, we augmented the real-world data with in-game data from a computer game–*"Football Manager"*. These in-game data include player skills such as finishing, dribbling, passing, and tackling abilities at normalized scales and can be considered as hand-crafted features by domain experts. We note that these pieces of information were crawled from FMINSIDE website.[4]

4.2 Raw Data

A match result consists of all the details of a match that took place between two teams. We gathered all the matches for every season starting from 2012-13 to 2021-22, totaling 3,034 matches.[5] Seasons prior to 2012-13 were not considered as most players who did not have common 21 attributes did not feature after 2011-12 season.

The statistics and attributes for a player are shown in Table 1, and they are merged with the in-game data as shown in Table2. Each player's data includes 3 attributes: name, players' official statistics, and in-game attributes. A field player's official statistics contains 124 attributes, and a goalkeeper contains 58: the number of yellow card per game, the rate of successful pass, the number of a goal per game, the number of assists per game, and the number of turnover per game. For in-game data, a field player contains 36 attributes, such as corners, crossing, and dribbling, and a goalkeeper contains 38 attributes.

4.3 Data Pre-Processing

We observed that not every player has the same diversity of all-season official statistics. Therefore, selecting official statistics as players' attributes was necessary to ensure all values were present in the player dataset. For the selection, we sorted official statistics in descending

[1]The available dataset on Kaggle was released six years ago.
[2]`https://www.premierleague.com/players`
[3]`https://www.premierleague.com/results`
[4]`https://fminside.net/players`
[5]380 matches take place every season.

1st-level attributes	2nd-level attributes
name (str)	
official overall stat (dict)	yellow card, normalized pass, goal per game, assist per game, turnover per game
fm attributes (dict)	corners, crossing, dribbling, finishing, first touch, free kick taking, heading, long shots, long throws, marking, ...

Table 2: Merged attributes for each player in *merged_player_data.json*

order in terms of how many times each statistic is observed in the player data and calculated the percentage of players who contains the top' n' statistics. We decided to use the top 21 statistics and sampled approximately 41% of players who included all of those statistics. The significant common characteristic of most players not included in the sample is that the last season those players played were before season 2012-13. Therefore, we decided to discard players who never had a match from season 2012-13 to 2021-22.

After discarding players, there were some players who still needed all or some of the top 21 official statistics. For in-game attributes, imputation is also required because Football Manager only has players currently attached to premier league teams. Therefore, missing all-season official statistics and missing players' in-game attributes were imputed using *KNN* algorithms using 3 neighbors.

All-season official attributes were normalized using min-max normalization except age. Ages were divided by 35 to normalize. In-game attributes representing a player's skill level were divided by 20, which is the maximum value of each level. Ability and potential representing the player's overall score were divided by their maximum value of 100. Other attributes such as length, weight, salary, and wage were normalized using min-max normalization.

4.4 Final Dataset

As shown in Table3, the in-game data for a player is merged with the data from Premier League. Each player's data includes 2 attributes: players' all season official statistics and in-game attributes. For all season official statistics, a field player contains 22 attributes, such as touches, successful_final_third_passes, poss_lost_all, and accurate_pass, and a goalkeeper contains 22 attributes as goal_kicks, saves, ball_recovery, and saves. For in-game data, a field player contains 42 attributes, such as corners, crossing, and dribbling, and a goalkeeper contains 44 attributes, such as aerial reach, punching, rushing out, one-and-one, and kicking.

is_goalkeeper	premier_stats	fm_stats
1	age, appearances, losses, wins, draws, min_played, touches, ...	corners, crossing, dribbling, finishing, first touch, ...
0	age, appearances, losses, wins, mins_played, keeper_throws, ...	aerial Reach, handling, kicking, one on ones, ...

Table 3: Details of a sample data in *final_player_data.json*

5 Experiments

5.1 Hyperparameter Tuning

We note that both randomized searching and grid searching can be used to optimize the hyperparameters of our models in different ways. The former takes a few random parameters while the latter requires a manually specified subset of parameters. In general, it is known that the randomized search method outperforms the grid search method when a small set of hyperparameters are considered. As this is the case for our setting, we opted for the randomized searching method when finding the hyperparameters.

5.2 Experiment 1: Elo Rating System

Elo rating system, created by Arpad Elo, is an effective method to calculate the relative strengths of players or teams with respect to their opponents. After fitting the system to learn each team's relative strengths, we can predict the probability of the outcome of the match.

A few adjustments were made to our model. Firstly, we set a threshold. If the difference between home team winning probability and its opponents was higher than the positive threshold, then it was determined as a win for the home team. Likewise, if it was lower than the negative threshold, then it was a loss. For other cases, it counted as a draw. Secondly, we used two approaches to evaluate the accuracy of our model: all seasons and individual seasons. In "all seasons" method, we collected 3793 matches from the 2012-13 to 2021-22 season. This model had an advantage with a large pool of training data to optimize our parameters. However, one of its shortcomings was that this model did not take into account the trend in performance and strength of each team by seasons. On the other hand, for "individual season approach", we took 380 matches from every season and ran them on our model. This model was able to reflect on the performance of each team season-wise. However, it had only 300 matches for the training set.

		2012-13	2013-14	2014-15	2015-16	2016-17	2017-18	2018-19	2019-20	2020-21	2021-22	All
Elo	**Acc.**	0.30	0.21	0.42	0.31	0.34	0.31	0.55	0.36	0.46	0.46	0.42
	F1	0.22	0.19	0.41	0.30	0.33	0.31	0.47	0.37	0.45	0.46	0.44
HGCN	**Acc.**	0.56	0.60	0.53	0.51	0.59	0.56	0.61	0.59	0.57	0.59	0.57
	F1	0.54	0.45	0.45	0.51	0.54	0.49	0.50	0.54	0.48	0.54	0.52

Table 4: Prediction performance at season-level for Elo and HGCN models.

Model	Input Variation	# Features	Precision	Recall	F1
SVM	Premier	484	0.53	0.53	0.53
	FM	928	0.52	0.52	0.52
	Premier + FM	**1412**	**0.55**	**0.55**	**0.55**
RF	Premier	484	0.53	0.53	0.53
	FM	928	0.5	0.5	0.5
	Premier + FM	**1412**	**0.55**	**0.55**	**0.55**
SVM	33 Premier	372	0.53	0.53	0.53
	7 FM	68	0.48	0.48	0.48
	33 Premier + 7 FM	440	0.54	0.54	0.54
RF	33 Premier	372	0.54	0.54	0.54
	7 FM	68	0.51	0.51	0.51
	33 Premier + 7 FM	440	0.53	0.53	0.53

Table 5: Prediction performance of SVM and RF with input variations (# features indicates the total size of the input).

To optimize our training system, We randomly assigned 80% of the collected data as a training set and the rest as a test set. Then, We optimized a threshold as a hyperparameter through *sklearn.randomizedCV*. Our motivation for this model over grid search is discussed in the next section. The experiment result is tabulated in Table4.

This result reveals two interesting trends. Firstly, the all-seasons model ended up with higher accuracy than many individual seasons. This likely means that despite its failure to capture the trend in season-wise performance of each team, its large training data set managed to overcome the shortcoming. Secondly, while a lack of training data for individual seasons caused a wide fluctuation in its accuracy, some of its high values suggested that it was able to take into account the strength of each team in that specific year.

5.3 Experiment 2: SVM & RF

We conducted 6 baseline experiments for each model to predict match results when:

1. $all\_season\_official\_statistics$ are given to each player.
2. $in\_game\_attributes$ are given to each player.
3. $all\_season\_official\_statistics$ and $in\_game\_attributes$ are given to each player.
4. Selected $all\_season\_official\_statistics$ (field player: 17, goalkeeper: 16) are given to each player.
5. Selected $in\_game\_attributes$ (field player: 3, goalkeeper: 4) are given to each player.
6. Selected $all\_season\_official\_statistics$ and $in\_game\_attributes$ are given to each player.

We trained Support Vector Machine (SVM) and Random Forests (RF) models for the baseline experiments using *scikit-learn* Python library. Hyperparameters for all models are tuned using *RandomizedSearchCV*. As shown in table 5, SVM and Random forest show the best performance on all sections (precision, recall, f1-score, support) when the player dataset includes all season official statistics and in-game attributes,

5.4 Experiment 3: Feature Importance

To better understand the models we trained, we observed the importance of each feature using the function RandomForestClassifier.feature_importances_. As shown in Figure 2, there are features that have high importance commonly in terms of player role between

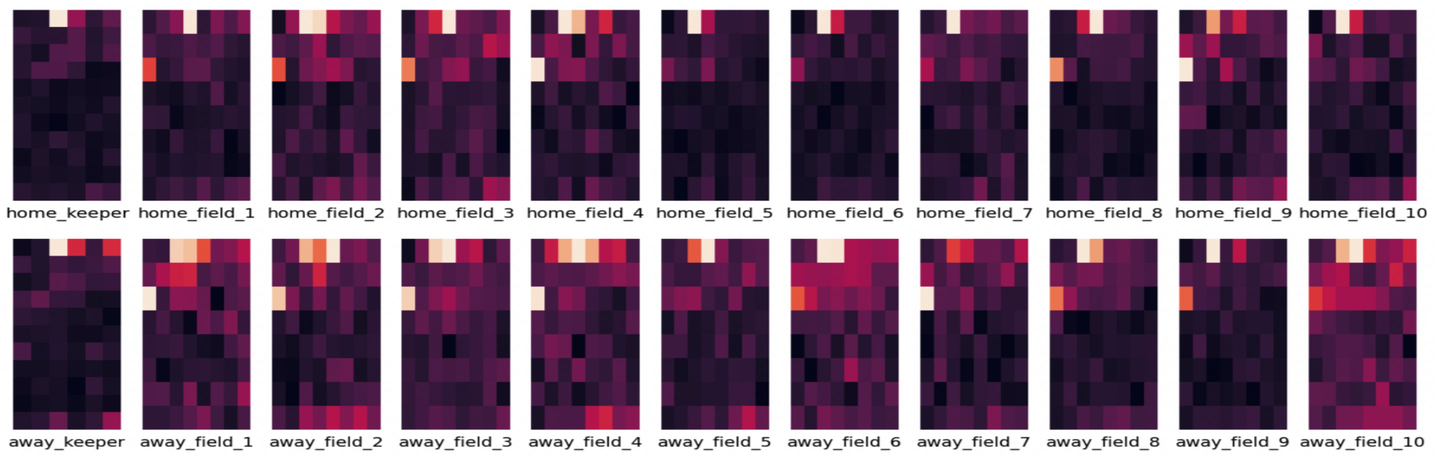

Figure 2: Heatmap showing the importance of each feature

field player and goalkeeper. Features of a field player in descending order are *losses*, *wins*, *attemts_conceded_ibox*,*draws*, *mins_played*. Features of a goalkeeper in descending order are *losses*, *wins*, *attempts_conceded_ibox*, *potential*, *saves*, *poss_lost_all*, *poss_lost_ctrl*.

5.5 Experiment 4: Heterogeneous Graph Convolutional Network (HGCN)

As illustrated in Section 3, we created a large HGCN network for each season and evaluated the network at per season level. The task here is to classify the type of edges between team-team nodes. We note that while the In Table 4, we observe that HGCN greatly outperforms the Elo system in all seasons as well as the SVM and RF models (Table 5). This confirms our hypothesis that adding team-team and team-player relations helps a lot with the prediction task.

6 Discussions

Following the feedback received from the instructor, TA, and peers, we experimented with preparing the inputs with player attributes to build a dense representation vector, conducted hyperparameter tuning, and implemented the proposed HGCN network.

In summary, we contributed a new dataset for the task of football match prediction and proposed a powerful model based on HGCN network. In future work, we will incorporate a temporality in the match data, and experiment with position-specific attributes such as team formation, etc.